Best Practices for XML Internationalization

W3C Working Group Note 13 February 2008

This version:
: http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/

Latest version:
: http://www.w3.org/TR/xml-i18n-bp/

Previous version:
: http://www.w3.org/TR/2007/WD-xml-i18n-bp-20071031/

Editors:
: Yves Savourel, ENLASO Corporation
: Jirka Kosek, Invited Expert
: Richard Ishida, W3C

This document is also available in these non-normative formats: PDF version[1] and XHTML Diff markup to publication from 31 October 2007[2].

Abstract

This document provides a set of guidelines for developing XML documents and schemas that are internationalized properly. Following the best practices describes here allow both the developer of XML applications, as well as the author of XML content to create material in different languages.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

1 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xml-i18n-bp.pdf
2 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xml-i18n-bp-diff.html
3 ⇢ http://www.w3.org/Consortium/Legal/ipr-notice#Copyright
4 ⇢ http://www.w3.org/
5 ⇢ http://www.csail.mit.edu/
6 ⇢ http://www.ercim.org/
7 ⇢ http://www.keio.ac.jp/
8 ⇢ http://www.w3.org/Consortium/Legal/ipr-notice#Legal_Disclaimer
9 ⇢ http://www.w3.org/Consortium/Legal/ipr-notice#W3C_Trademarks
10 ⇢ http://www.w3.org/Consortium/Legal/copyright-documents

This is a W3C Working Group Note[11] of "Best Practices for XML Internationalization". This document was developed by the Internationalization Tag Set (ITS) Working Group[12], part of the W3C Internationalization Activity[13].

Feedback about this document is encouraged. Send your comments to www-i18n-comments@w3.org. Use "[Comment on xml-i18n-bp WD]" in the subject line of your email, followed by a brief subject. The archives[14] for this list are publicly available.

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy[15]. W3C maintains a public list of any patent disclosures[16] made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s)[17] must disclose the information in accordance with section 6 of the W3C Patent Policy[18].

11 ⇢ http://www.w3.org/2005/10/Process-20051014/tr.html#WGNote
12 ⇢ http://www.w3.org/International/its/
13 ⇢ http://www.w3.org/International/Activity
14 ⇢ http://lists.w3.org/Archives/Public/www-i18n-comments/
15 ⇢ http://www.w3.org/Consortium/Patent-Policy-20040205/
16 ⇢ http://www.w3.org/2004/01/pp-impl/37139/status
17 ⇢ http://www.w3.org/Consortium/Patent-Policy-20040205/#def-essential
18 ⇢ http://www.w3.org/Consortium/Patent-Policy-20040205/#sec-Disclosure

Table of Contents

1 Introduction....5
- 1.1 Who should use this document....5
- 1.2 How to use this document....5
  - 1.2.1 Designers and developers of XML applications....5
  - 1.2.2 Users and authors of XML content....5

2 When Designing an XML Application....6
- Best Practice 1: Defining markup for natural language labelling....10
- Best Practice 2: Defining markup to specify text direction....13
- Best Practice 3: Avoiding translatable attribute values....15
- Best Practice 4: Indicating which elements and attributes should be translated....17
- Best Practice 5: Defining markup to override translate information....19
- Best Practice 6: Providing information related to text segmentation....21
- Best Practice 7: Defining markup for ruby text....23
- Best Practice 8: Defining markup for notes to localizers....25
- Best Practice 9: Defining markup for unique identifiers....28
- Best Practice 10: Identifying terminology-related elements....29
- Best Practice 11: Defining markup for specifying or overriding terminology-related information....31
- Best Practice 12: Working with multilingual documents....32
- Best Practice 13: Naming elements and attributes....34
- Best Practice 14: Defining a span-like element....36
- Best Practice 15: Documenting internationalization and localization features of your schema....37

3 When Authoring XML Content....39
- Best Practice 16: Specifying the language of content....40
- Best Practice 17: Specifying text directionality....43
- Best Practice 18: Overriding information about what should be translated....45
- Best Practice 19: Assigning unique identifiers....48
- Best Practice 20: Avoiding CDATA sections....49
- Best Practice 21: Providing notes for localizers....52
- Best Practice 22: Working with inserted text....54
- Best Practice 23: Identifying terms....57
- Best Practice 24: Storing markup from another format....59

4 Generic Techniques....61
- 4.1 Writing ITS Rules....61
  - 4.1.1 Precedence and Inheritance....61
  - 4.1.2 Dealing with namespaces....62
  - 4.1.3 Create your XPath expressions with care....64
- 4.2 Example of adding an attribute to an existing schema....65
  - 4.2.1 Including xml:lang in XML Schema....65
  - 4.2.2 Including xml:lang in RELAX NG....65
  - 4.2.3 Including xml:lang in an XML DTD....66

5 ITS Applied to Existing Formats....67
- 5.1 ITS and XHTML 1.0....67
  - 5.1.1 Integrating ITS into XHTML....67
  - 5.1.2 Using XHTML Modularization 1.1 for the Definition of ITS....69
  - 5.1.3 Using NVDL to integrate ITS into XHTML....74
  - 5.1.4 Associating existing XHTML markup with ITS....76
- 5.2 ITS and TEI....78
  - 5.2.1 Integrating ITS into TEI....78

5.3 ITS and XML Spec....80
5.3.1 Integration of ITS into XML Spec....80
5.3.2 Associating existing XML Spec markup with ITS....83
5.4 ITS and DITA....84
5.4.1 Integration of ITS into DITA....84
5.4.2 Associating existing DITA markup with ITS....87
5.5 ITS and GladeXML....88
5.5.1 Integration of ITS into GladeXML....89
5.5.2 Associating Existing GladeXML Markup with ITS....90
5.6 ITS and DocBook....90
5.6.1 Integration of ITS into DocBook....90
5.6.2 Associating existing DocBook markup with ITS....93

Appendices

A References (Non-Normative) 96
B Acknowledgements (Non-Normative) 98

1 Introduction

This document is a complement to the W3C Recommendation *Internationalization Tag Set (ITS) Version 1.0* [ITS]. However, not all internationalization-related issues can be resolved by the special markup described in ITS. The best practices in this document therefore go beyond application of ITS markup to address a number of problems that can be avoided by correctly designing the XML format, and by applying a few additional guidelines when developing content.

This document and *Internationalization Tag Set (ITS) Version 1.0* [ITS] implement requirements formulated in *Internationalization and Localization Markup Requirements* [ITS REQ].

This set of best practices does not cover all topics about internationalization for XML. Other useful reference material includes: *Character Model for the World Wide Web 1.0: Fundamentals* [CharMod], and *Unicode in XML and other Markup Languages* [Unicode in XML].

1.1 Who should use this document

This document is divided into two main sections:

- The first one is intended for the *designers and developers of XML applications (also referred to here as 'schemas' or 'formats')*.
- The second is intended for the *XML content authors*. This includes users modifying the original content, such as translators.

1.2 How to use this document

1.2.1 Designers and developers of XML applications

Section 2: When Designing an XML Application on page 6 provides a list of some of the important design choices you should make in order to ensure the internationalization of your format.

Section 4: Generic Techniques on page 61 provides additional generic techniques such as writing ITS rules or adding an attribute to a schema. Such techniques apply to many of the best practices.

Section 5: ITS Applied to Existing Formats on page 67 provides a set of concrete examples on how to apply ITS to existing XML based formats. This section illustrates many of the guidelines in this document.

1.2.2 Users and authors of XML content

Section 3: When Authoring XML Content on page 39 provides a number of guidelines on how to create content with internationalization in mind. Many of these best practices are relevant regardless of whether or not your XML format was developed especially for internationalization.

Section 4.1: Writing ITS Rules on page 61 provides practical guidelines on how to write ITS rules. Such techniques may be useful when applying some of the more advanced authoring best practices.

2 When Designing an XML Application

Designers and developers of XML applications should take into account the following best practices:

Best Practice	Implementing as a new feature	Handling legacy markup
Defining markup for natural language labelling on page 10	Make sure the `xml:lang` attribute is defined for the root element of your document, and for any element where a change of language may occur.	Provide an ITS Rules document where you use the `its:langRule` element to specify what attribute or element is used instead of `xml:lang`.
Defining markup to specify text direction on page 13	Make sure the `its:dir` attribute is defined for the root element of your document, and for any element that has text content.	Provide an ITS Rules document where you use the `its:dirRule` element to associate the different directionality indicators with their equivalents in ITS.
Avoiding translatable attribute values on page 15	Make sure you store all translatable text as element content, not as attribute values.	Provide an ITS Rules document where you use the `its:translateRule` element to specify what attributes are translatable.
Indicating which elements and attributes should be translated on page 17	Provide an ITS Rules document where you use `its:translateRule` elements to indicate which elements have non-translatable content.	
Defining markup to override translate information on page 19	• Make sure the `its:translate` attribute is defined for the root element of your documents, and for any element that has text content. • It is also recommended that you define the `its:rules` element in your schema, for example in a header if there is one, and within that the `its:translateRule` element. Content authors can then use these elements to globally change the default translate rules for specific elements and attributes.	Provide an ITS Rules document where you use the `its:translateRule` element to associate this mechanism with the ITS Translate data category.
Providing information related to text segmentation on page 21	Provide an ITS Rules document where you use `its:withinTextRule` elements to indicate which elements should be treated as either part of their parents, or as a nested but independent run of text. By default, element boundaries are assumed to correspond to segmentation boundaries.	

Best Practice	Implementing as a new feature	Handling legacy markup
Defining markup for ruby text on page 23	Make sure the `its:ruby` element and its children are defined for all elements where there is text content.	Provide an ITS Rules document where you use the `its:rubyRule` element to associate your ruby markup with its equivalent in ITS.
Defining markup for notes to localizers on page 25	• Make sure the attributes `its:locNote`, `its:locNoteType` and `its:locNoteRef` are defined in your schema. This markup allows content authors to provide localization-related notes as `its:locNote` attribute values, or to point to the location of the relevant note text using `its:locNoteRef`. • It is also recommended that you define the `its:rules` element in your schema, for example in a header if there is one, and within that the `its:locNoteRule` element and its related markup. Content authors can use this markup to specify localization-related notes. Within the `its:locNoteRule` element, notes can be stored in the `its:locNote` element.	Provide an ITS Rules document where you use the `its:locNoteRule` element to associate your notes markup with its equivalent in ITS.
Defining markup for unique identifiers on page 28	Make sure that elements with translatable content can be associated with a unique identifier.	
Identifying terminology-related elements on page 29	Provide an ITS Rules document where you use `its:termRule` elements to indicate which elements are terms and information related to them (e.g. definitions).	
Defining markup for specifying or overriding terminology-related information on page 31	• Make sure the `its:term` and the `its:termInfoRef` attributes are defined for any element that has text content. • It is also recommended to define the `its:rules` element in your schema, for example in a header if there is one. The `its:rules` element provides access to the `its:termRule` element which can be used to override terminology-related information globally.	

Best Practice	Implementing as a new feature	Handling legacy markup
Working with multilingual documents on page 32	For documents that need to go through some localization tasks, always store the localized version of the text in a separate document.	
Naming elements and attributes on page 34	• Make sure the names of the elements and attributes of your schema reflect their functions, rather than one possible way of rendering their content. • Also, if possible, avoid element names which do not follow a fixed naming scheme (for example, element names that serve also as identifiers).	Not applicable
Defining a span-like element on page 36	Make sure you define a `span`-like element in your schema that will allow authors to associate arbitrary content with properties such as directionality, language information, etc.	If no `span`-like element already exists in your schema, you may be able to use `its:span`.
Documenting internationalization and localization features of your schema on page 37	Make sure you document the internationalization and localization aspects of your schema by providing a set of relevant ITS rules in a single standalone ITS Rules document.	

Where it says "How to implement this as a new feature", this section describes how to create new schemas or add new features to existing schemas. When doing this you may need to take into account the following:

- Think twice before creating your own schema. Seriously consider using existing formats such as DITA, DocBook, Open Document Format, Office Open XML, XML User Interface Language, Universal Business Language, etc. Those formats have many useful insights already built in.
- Check carefully whether an existing format comes with a built-in capability for modification. DocBook and DITA, for example, come with their own set of features for adapting their format to special needs.
- The modification mechanisms available will depend on the schema language (DTD, XML Schema, RELAX NG, etc.) For example, namespace-based modularization of schemas is difficult to achieve with DTDs.

  NVDL is an example of a meta-schema language was designed especially to allow integration of several existing vocabularies into a single XML vocabulary without the need to know the details of source schemas. This means that with NVDL you can usually create a schema for compound documents more easily than with other schema technologies.

- Each schema language provides different ways of extending or modifying existing schemas. Some examples are the include[19], import[20] or redefine[21] mechanisms in XML Schema.

- Some processors do not implement support for all schema language constructs, due to erroneous implementations or differences in conformance profiles (e.g. see the conformance requirements to XML Schema part 1[22]). Therefore a schema which works in one environment may not work in a different one.

- What is possible also depends on the features of the schema which the modification is targeting. For example:

  - An XML Schema `redefine` is only possible if the modified schema has been created with named types.

  - If you are working with XML Schema, you can only apply the technique of 'chameleon' or 'proxy' schemas (see http://www.xfront.com/ZeroOneOrManyNamespaces.html) if the 'chameleon' schemas have no namespace. For example, the XML Schema document for ITS XML Schema document for ITS[23] has a target namespace and therefore cannot be a 'chameleon' schema.

Note: The considerations above are only a portion of what you need to take into account. You need to know a lot more when diving into schema modularization.

19 ⇢ http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/#ref23
20 ⇢ http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/#ref31
21 ⇢ http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/#ref52
22 ⇢ http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#concepts-conformance
23 ⇢ http://www.w3.org/TR/2007/REC-its-20070403/its.xsd

Best Practice 1: Defining markup for natural language labelling

Provide a way for authors to specify the natural language of content using ITS markup, or document equivalent legacy markup in an ITS Rules document.

The XML namespace provides the `xml:lang` attribute and the ITS Language Information data category provides the `its:langRule` element to address this requirement.

How to implement this as a new feature

Make sure the `xml:lang` attribute is defined for the root element of your document, and for any element where a change of language may occur.

For examples of how to add attributes in your existing schema see Section 4.2: Example of adding an attribute to an existing schema on page 65.

Some XML documents may be designed to store data without natural language content. In these cases, there is no need for the `xml:lang` attribute.

The scope of the `xml:lang` attribute applies to both the attributes and the content of the element where it appears, therefore one cannot specify different languages for an attribute and the element content. ITS does not provide a remedy for this. Instead, it is recommended that you avoid translatable attributes.

Make sure that the definition of the `xml:lang` attribute allows for empty values. That is:

- In a DTD you must not use `NMTOKEN` as the data type, instead use `CDATA`.
- In XML Schema the built-in data type `language` does not allow empty values. However, the declaration for `xml:lang` in the XML Schema document for the XML namespace at http://www.w3.org/2001/xml.xsd does allow for empty values and therefore can be used.

It is *not recommended* to use your own attribute or element to specify the language of the content. The `xml:lang` attribute is supported by various XML technologies such as XPath and XSLT (e.g. the `lang()` function). Using something different would diminish the interoperability of your documents and reduce your ability to take advantage of some XML applications.

> **Note:** If you need to specify language as data or meta-data about something external to the document, do it with an attribute different from `xml:lang`. For more information see the article xml:lang in XML document schemas[24].

Example 1: Language information not applicable to the content of the element where it is used

In XHTML the language of a file linked with the `a` element is indicated with a `hreflang` attribute because it does not apply to the content of the `a` element.

```
<a xml:lang="en" href="german.html" hreflang="de">Click here for German</a>
```

24 → http://www.w3.org/International/questions/qa-when-xmllang

If you have different languages in the attribute values and content of an element, consider nesting elements, if possible. See Handling attribute values and element content in different languages[25].

Handling markup not in the ITS namespace

If you are working with an existing schema where there is a way to specify content language that uses something other than the `xml:lang` attribute (but still uses the same values as `xml:lang`), you should provide an ITS Rules document where you use the `its:langRule` element to specify what attribute or element is used instead of `xml:lang`.

Example 2: Dealing with a non-standard way of declaring language information

In this document the `langcode` element is used to specify the language of the `text` element. The `langcode` element has no inheritance behavior equivalent to the one of `xml:lang`.

Note: This example is a multilingual document, which has its own set of issues (see Best Practice 12: Working with multilingual documents on page 32).

```
<myRes>
 <messages>
  <msg id="1">
   <langcode>en</langcode>
   <text>Cannot find file.</text>
  </msg>
  <msg id="2">
   <langcode>fr</langcode>
   <text>Fichier non trouvé.</text>
  </msg>
 </messages>
</myRes>
```

The corresponding ITS Rules document contains an `its:langRule` element that specifies that the `langcode` element holds the same values as the `xml:lang` attribute and applies to the `text` element.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:langRule selector="//text[../langcode]" langPointer="../langcode"/>
</its:rules>
```

Why do this

Information about the language of content can be very important for correctly rendering or styling text in some scripts, applying spell-checkers during content authoring, appropriate selection of voice for text-to-speech systems, script-based processing, and numerous other reasons. You must provide a standard way to specify the language for the document as a whole, but also for parts of the document where the language changes.

Resources:

Background information

- Internationalization FAQ: xml:lang in XML document schemas.

25 ⇢ http://www.w3.org/TR/i18n-html-tech-lang/#ri20050128.175100333

http://www.w3.org/International/questions/qa-when-xmllang
- Mechanisms for declaring language in HTML
  http://www.w3.org/TR/i18n-html-tech-lang/#ri20050208.095812479

Reference links

- Description of the language identification mechanism in the XML specification.
  http://www.w3.org/TR/REC-xml/#sec-lang-tag
- The "Language Information" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#language-information

Best Practice 2: Defining markup to specify text direction

Provide a way for authors to specify the direction of text using ITS markup, or document equivalent legacy markup in an ITS Rules document.

In scripts such as Arabic and Hebrew characters may run from both left to right and right to left when displayed. Directional markup allows you to manage the flow of characters. For an example of how directional markup is used see Creating (X)HTML Pages in Arabic & Hebrew[26].

The ITS Directionality data category provides the `its:dir` attribute and the `its:dirRule` element to address this requirement.

How to implement this as a new feature

Make sure the `its:dir` attribute is defined for the root element of your document, and for any element that has text content.

For examples of how to add attributes in your existing schema see Section 4.2: Example of adding an attribute to an existing schema on page 65.

Handling markup not in the ITS namespace

If you are working with an existing schema where there is a way to specify text directionality that is not implemented using the `its:dir` attribute, you should provide an ITS Rules document where you use the `its:dirRule` element to associate the different directionality indicators with their equivalents in ITS.

Example 3: Specifying text directionality where non-ITS markup has been used.

In this document the `textdir` attribute is used to specify directionality of a text run.

```
<text xml:lang="en">
 <body>
  <par>In Hebrew, the title
     <quote xml:lang="he" textdir="r2l">פעילות הבינאום, W3C</quote>
     means <quote>Internationalization Activity, W3C</quote>.</par>
 </body>
</text>
```

Note: This example shows the directionality of the source text correctly. This is to ensure that you understand the concepts being described. For such display, you need a sophisticated editor that resolves directionality of the source text correctly. Many editors are not yet this sophisticated. See the related discussion about Problems with bidirectional source text[27] in [Bidi in X/HTML].

The corresponding ITS Rules document contains a set of `its:dirRule` elements that specifies the relationships between the `textdir` attribute and the ITS Directionality data category.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:dirRule selector="//*[@textdir='l2r']" dir="ltr"/>
 <its:dirRule selector="//*[@textdir='r2l']" dir="rtl"/>
 <its:dirRule selector="//*[@textdir='lro']" dir="lro"/>
```

26 → http://www.w3.org/International/tutorials/bidi-xhtml/
27 → http://www.w3.org/TR/i18n-html-tech-bidi/#d2e283

```
    <its:dirRule selector="//*[@textdir='rlo']" dir="rlo"/>
  </its:rules>
```

Why do this

Generally the Unicode bidirectional algorithm will produce the correct ordering of mixed directionality text in scripts such as Arabic and Hebrew. Sometimes, however, additional help is needed. For instance, in the sentence of Example 4 the 'W3C' and the comma should appear to the left side of the quotation. This cannot be achieved using the bidirectional algorithm alone.

Example 4: Sentence where bidirectional markup is needed for a proper display

The following will display incorrectly, since no directional markup has been used:

The title says "פעילות הבינאום, W3C" in Hebrew.

The text 'W3C' and the comma should be to the left of the quoted Hebrew text. If your browser supports bidirectional display, the following should appear correctly, since directional markup has been added to the element surrounding the quote:

The title says "‮פעילות הבינאום, W3C‬" in Hebrew.

The desired effect can be achieved using Unicode control characters, but this is not recommended (See *Unicode in XML and other Markup Languages* [Unicode in XML]). Markup is needed to establish the default directionality of a document, and to change that where appropriate by creating nested embedding levels.

Markup is also occasionally needed to disable the effects of the bidirectional algorithm for a specified range of text.

Resources:

Background information

- Internationalization FAQ: What you need to know about the bidi algorithm and inline markup
  http://www.w3.org/International/articles/inline-bidi-markup/
- Authoring Techniques for XHTML & HTML Internationalization: Handling Bidirectional Text 1.0
  http://www.w3.org/TR/i18n-html-tech-bidi/#ri20030728.094313871
- Unicode Technical Report #20: Unicode in XML and other Markup Languages
  http://www.w3.org/TR/2007/NOTE-unicode-xml-20070516/

Reference links

- The "Directionality" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#directionality

Best Practice 3: Avoiding translatable attribute values

Do not define attribute values that will contain user readable content. Use elements for such content.

How to implement this as a new feature

Make sure you store all translatable text as element content, not as attribute values.

Example 5: Avoiding translatable attribute values

It is bad design to use the `desc` attribute to store the alternate descriptive text for the `image` element, as in this example.

```
<image src="elephants.png" desc="Elephants bathing in the Zambezi River."/>
```

Instead, define the content of `image` itself to hold the text you need. This way there is no translatable text in an attribute.

```
<image src="elephants.png">Elephants bathing in the Zambezi River.</image>
```

Note: In many cases, using translatable element content instead of translatable attributes will result in one sentence being embedded within another one. For instance, in Example 5 the description of the image will be embedded inside the text of the paragraph that contains it. In such cases, do not forget to declare the relevant element (here `image`) as 'nested', as described in Best Practice 6: Providing information related to text segmentation on page 21.

Handling markup not in the ITS namespace

If you are working with an existing schema where there are attributes with translatable values, you should provide an ITS Rules document where you use the `its:translateRule` element to specify what attributes are translatable. See Best Practice 4: Indicating which elements and attributes should be translated on page 17 for more information about how to do this.

Why do this

There are a number of issues related to storing translatable text in attribute values. Some of them are:

- The language identification mechanism (i.e. `xml:lang`) applies to both the content and to the attribute values of the element where it is declared. If the text of an attribute is in a different language than the text of the element content, one cannot set the language for both correctly.
- It may be necessary to apply some language-related properties, such as directionality and language identification, to only part of the text in an attribute value. This requires the use of a `span`-like element, but elements cannot be used within an attribute value.
- It is difficult to apply meta-information, such as no-translate flags, author's notes, etc., to the text of an attribute value

- The difficulty of attaching unique identifiers to translatable attribute text makes it more complicated to use ID-based leveraging tools.
- It can be problematic to prepare translatable attributes for localization because they can occur within the content of a translatable element, breaking it into different parts, and possibly altering the sentence structure.

All these potential problems are less likely to occur when the text is the content of an element rather than the value of an attribute.

Resources:

Background information

- "Attributes and translatable text" in "Requirements for Localizable DTD Design"
  http://people.w3.org/rishida/localizable-dtds/#attributes

Reference links

- The "Translate" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#trans-datacat
- The "Element Within Text" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#elements-within-text

Best Practice 4: Indicating which elements and attributes should be translated

Document in an ITS Rules document which elements and attributes need to be translated, and which do not, when this differs from the ITS defaults.

The ITS Translate data category provides the `its:translateRule` element to address this requirement.

How to do this

Provide an ITS Rules document where you use `its:translateRule` elements to indicate which elements have non-translatable content.

If you are working with a schema where there are translatable attributes (something that is not recommended), you should also use `its:translateRule` to specify these translatable attributes.

> **Note:** Where the language of content is given as `xml:lang="zxx"`, where `zxx` indicates content that is not in a language, the element in question is probably not to be translated. You should provide a rule for this.

Example 6: Document where default ITS "Translate" rules do not apply

In the following document, the content of the `head` element should not be translated, and the value of the `alt` attribute should be translated. In addition, the content of the `del` element should not be translated.

```
<myDoc xml:lang='en'>
 <head>
  <id xml:lang="zxx">H4-A3-F8-A1</id>
  <author>Robert Griphook</author>
  <rev>v13 2007-10-27</rev>
 </head>
 <par>To start click <ins>the <ui>Start</ui>
  button</ins><del>green icon</del>
  and fill the form labeled by the following icon:
  <ref file="vat.png" alt="Value Added Tax Form"/></par>
</myDoc>
```

The following rules specify exceptions from the default ITS behavior for documents like the one above.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:translateRule selector="/myDoc/head" translate="no"/>
 <its:translateRule selector="//*/@alt" translate="yes"/>
 <its:translateRule selector="//del" translate="no" />
 <its:translateRule selector="//@*[ancestor::del]" translate="no"/>
 <its:translateRule selector="//*[lang('zxx')] | //@*[lang('zxx')]" translate="no"/>
</its:rules>
```

- First `translateRule`: The content of `head` in `myDoc` is not translatable. By inheritance, the child elements of `head` are also assumed not translatable.
- Second `translateRule`: All the `alt` attributes are translatable.

- Third `translateRule`: The content of `del` is not translatable.
- Fourth `translateRule`: The non-translatability of `del` applies also to any attribute that may have been set as translatable by a prior rule (i.e. the second rule).
- Fifth `translateRule`: Any element or attribute with their language set to `zxx` is not translatable.

Why do this

By default, ITS assumes that the content of all elements is translatable and that all attributes have non-translatable values. If your XML document type does not correspond to this default assumption it is important to indicate what are the exceptions. Doing so can significantly improve translation throughput.

Resources:

Reference links

- The "Translate" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#trans-datacat

Best Practice 5: Defining markup to override translate information

Provide a way for authors to override translate defaults, using ITS markup, or document equivalent legacy markup in an ITS Rules document.

The ITS Translate data category provides the `its:translate` attribute and the `its:translateRule` element to address this requirement.

How to implement this as a new feature

Make sure the `its:translate` attribute is defined for the root element of your documents, and for any element that has text content.

For examples of how to add attributes in your existing schema see Section 4.2: Example of adding an attribute to an existing schema on page 65.

It is also recommended that you define the `its:rules` element in your schema, for example in a header if there is one, and within that the `its:translateRule` element. Content authors can then use these elements to globally change the default translate rules for specific elements and attributes.

Handling markup not in the ITS namespace

If you are working with a schema where there is a way to override translate information that is not `its:translate`, the authors of the documents should use it. In addition, you should provide an ITS Rules document where you use the `its:translateRule` element to associate this mechanism with the ITS Translate data category.

For example, DITA offers a `translate` attribute, and Glade provides a `translatable` attribute. Both have the same semantics as `its:translate`, ie. the translation information applies to element content, including child elements, but excluding attribute values.

Example 7: DITA translation information

The following rules indicate how to associate the DITA `translate` attribute with the ITS Translate data category. The order in which the rules are listed is important:

- First `translateRule`: Indicates that the content of any element with a `translate` attribute set to `no` is not translatable.
- Second `translateRule`: Indicates that any attribute value of any element with a `translate` attribute set to `no` is not translatable. This is needed because some attributes are translatable in DITA and we need to make sure they are not translated when `translate="no"` is used in the elements where they are.
- Third `translateRule`: Indicates that the content of any element with a `translate` attribute set to `yes` is translatable. This takes care of the cases where `translate="yes"` is used to override a prior `translate="no"`.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:translateRule selector="//*[@translate='no']" translate="no"/>
```

```
    <its:translateRule selector="//*[@translate='no']/descendant-or-self::*/@*"
     translate="no"/>
    <its:translateRule selector="//*[@translate='yes']" translate="yes"/>
  </its:rules>
```

You can find a more complete example of how DITA markup is associated with ITS in Section 5.4.2: Associating existing DITA markup with ITS on page 87.

Why do this

In some cases, the author of a document may need to change the translatability property on parts of the content, overriding ITS default behavior, or the general rules for the schema that you have specified when applying Best Practice 4: Indicating which elements and attributes should be translated on page 17.

Resources:

Reference links

- The "Translate" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#trans-datacat

Best Practice 6: Providing information related to text segmentation

Document in an ITS Rules document how elements should be handled with regard to segmentation.

Segmentation refers to how text is broken down, from a linguistic viewpoint, into units that can be handled by processes such as translation.

The ITS Element Within Text data category provides the `its:withinTextRule` element to address this requirement.

How to do this

Whether you are creating a new schema or documenting legacy markup, provide an ITS Rules document where you use `its:withinTextRule` elements to indicate which elements should be treated as either part of their parents, or as a nested but independent run of text. By default, element boundaries are assumed to correspond to segmentation boundaries.

Example 8: A DITA document with formatting and footnote elements.

In the following DITA document:

- The elements `term` and `b` should be treated as part of their parent.
- The element `fn` should be treated as an independent run of text.

```
<concept id="myConcept" xml:lang="en-us">
 <title>Types of horse</title>
 <conbody>
  <ol>
   <li>Palouse horse:<p><term>Palouse horses</term><fn>A palouse horse
    is the same as an <b>Appaloosa</b>.</fn> have spotted coats.
    The <term>Nez-Perce</term> Indians have been key in breeding this
    type of horse.</p></li>
  </ol>
 </conbody>
</concept>
```

The `its:withinTextRule` element is used to specify the behavior of three elements, all other elements are assumed to have the value `its:withinText="no"`:

- First `withinTextRule`: The elements `term` and `b` are defined as part of the text flow.
- Second `withinTextRule`: The element `fn` is defined as a separate bit of content nested inside its parent element.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:withinTextRule selector="//term | //b" withinText="yes"/>
 <its:withinTextRule selector="//fn" withinText="nested"/>
</its:rules>
```

These rules applied to the DITA document above will result in four distinct runs of text:

```
1. title: "Types of horse"

2. li: "Palouse horse:"

3. p: "{term}Palouse horses{/term}{fn/} have spotted coats. The
   {term}Nez-Perce{/term} Indians have been key in breeding this type
   of horse."

4. fn: "A palouse horse is the same as an {b}Appaloosa{/b}."
```

Why do this

Many applications that process content for linguistic-related tasks need to be able to perform a basic segmentation of the text content. They need to be able to do this without knowing the semantics of the elements.

While in many cases it is possible to detect mixed content automatically, there are some situations where the structure of an element makes it impossible for tools to know for sure where appropriate segmentation boundaries fall. For example, the boundaries of some inline elements, such as emphasis, do not typically correspond to segmentation boundaries; on the other hand, some inline elements embedded in a parent element, such as footnotes or quotations, may define segments that should be handled separately from the text in which they are embedded.

Intelligent segmentation is particularly important in translation to successfully match source text against translation-memory databases.

Resources:

Reference links

- The "Element Within Text" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#elements-within-text

Best Practice 7: Defining markup for ruby text

Provide a way for authors to mark up ruby text using ITS markup, or document equivalent legacy markup in an ITS Rules document.

Ruby text is used to provide a short annotation of an associated base text. It is most often used to provide a reading (pronunciation) guide.

The ITS Ruby data category provides the elements `its:ruby` and `its:rubyRule` and their children to address this requirement. The definition of this data category is compliant with the specification of Ruby in [Ruby Annotation].

How to implement this as a new feature

Make sure the `its:ruby` element and its children are defined for all elements where there is text content.

Handling markup not in the ITS namespace

If you are working with an existing schema where there is a way to specify ruby text that has the same semantics as the ITS Ruby data category (for example the *Ruby Annotation* [Ruby Annotation]), you should provide an ITS Rules document where you use the `its:rubyRule` element to associate your ruby markup with its equivalent in ITS.

Example 9: Document with ruby-like elements.

In this document, the `rubyBlock` element has the same functionality as `its:ruby`, `rBase` as `its:rb`, `rParen` as `its:rp`, and `rText` as `its:rt`.

```
<text>
 <para>この本は <rubyBlock>
  <rBase>慶応義塾大学</rBase>
  <rParen>(</rParen>
  <rText>けいおうぎじゅくだいがく</rText>
  <rParen>)</rParen>
 </rubyBlock>の歴史を説明するものです。</para>
</text>
```

This `its:rubyRule` element indicates that the `rBase` element has the same functionality as `its:rb` and that the elements `its:ruby`, `its:rt` and `its:rt` have equivalent elements as well.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:rubyRule selector="//rBase" rubyPointer=".."
  rpPointer="../rParen" rtPointer="../rText" />
</its:rules>
```

Why do this

Ruby is a type of annotation for text. It can be used with any language, but is very commonly used with East Asian scripts to provide phonetic transcriptions of characters that are likely to be unfamiliar to a reader. For example it is widely used in educational materials and children’s texts. It is also occasionally used to convey information about meaning.

Because ruby annotation may be needed when localizing into Japanese or Chinese, it is a good idea to make provision for it in your schema, even if your original documents are to be developed into a language that does not use such markup.

Resources:

Background information

- Internationalization FAQ: Ruby
  http://www.w3.org/International/questions/qa-ruby
- "Implementing the Ruby Module", a personal note.
  http://www.w3.org/People/mimasa/test/schemas/NOTE-ruby-implementation

Reference links

- The "Ruby" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#ruby-annotation
- Ruby Annotation
  http://www.w3.org/TR/ruby/

Best Practice 8: Defining markup for notes to localizers

Provide a way for authors to specify notes for localizers using ITS markup, or document equivalent legacy markup in an ITS Rules document.

The ITS Localization Note data category provides the attributes `its:locNote`, `its:locNoteType` and `its:locNoteRef`, as well as the `its:locNoteRule` element to address this requirement.

How to implement this as a new feature

Make sure the attributes `its:locNote`, `its:locNoteType` and `its:locNoteRef` are defined in your schema. This markup allows content authors to provide localization-related notes as `its:locNote` attribute values, or to point to the location of the relevant note text using `its:locNoteRef`.

For examples of how to add attributes in your existing schema see Section 4.2: Example of adding an attribute to an existing schema on page 65.

Example 10: An illustration of how an author could point to localization notes with `its:locNoteRef`

The `its:locNote` element specifies that the message with the identifier `NotFound` has a corresponding explanation note in an external HTML file. The URI for the exact location of the note is stored in the `its:locNoteRef` attribute.

```
<myRes>
 <head>
  <its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
   <its:locNoteRule locNoteType="description"
    selector="//msg[@id='NotFound']"
    locNoteRef="EX-devlocnotes-4.html#NotFound" />
  </its:rules>
 </head>
 <body>
  <msg id="NotFound">Cannot find {0} on {1}.</msg>
 </body>
</myRes>
```

The HTML file with the localization notes is a simple document with the anchor elements corresponding to the identifiers in the referring XML document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <meta http-equiv="Content-Language" content="en-us">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Localization Notes</title>
 </head>
 <body lang="en">
 <p><a name="NotFound"></a>{0} is a filename<br />
  {1} is a directory name</p>
 </body>
</html>
```

It is also recommended that you define the `its:rules` element in your schema, for example in a header if there is one, and within that the `its:locNoteRule` element and its related markup. Content authors can use this markup to specify localization-related notes. Within the `its:locNoteRule` element, notes can be stored in the `its:locNote` element.

The `its:locNoteRule` element also allows you to specify notes in the current XML document via the `locNotePointer` attribute, or to provide an existing reference to notes via the `locNoteRefPointer` attribute.

Example 11: An illustration of how an author could store localization notes in `its:locNoteRule`

The `its:locNoteRule` element associates the content of the `its:locNote` element with the message that has the identifier 'DisableInfo', and flags it as important. This would also work if the rule was in an external file, allowing content authors to provide notes without modifying the source document.

```
<myDoc>
 <head>
  <its:rules xmlns:its="http://www.w3.org/2005/11/its"
   version="1.0" its:translate="no">
   <its:locNoteRule locNoteType="alert" selector="//msg[@id='DisableInfo']">
   <its:locNote>The variable {0} has three possible values: 'printer',
    'stacker' and 'stapler options'.</its:locNote>
   </its:locNoteRule>
  </its:rules>
 </head>
 <body>
  <msg id="DisableInfo">The {0} has been disabled.</msg>
 </body>
</myDoc>
```

Note: The example includes `its:translate="no"` in the `its:rules` tag, to prevent translators from attempting to translate the notes themselves.

Storing notes as element content has advantages over storing notes as `its:locNote` attribute values: markup for such things as language and directionality can be associated with the text of the content of an element, or parts of the text when a span-like element is also available, but you cannot do these things with attribute text.

Storing notes in an `its:locNote` element can therefore offer these advantages as long as there is a mechanism to associate the notes with the relevant content. On the other hand, it can be easier to scan documents, in some cases, if the note text is stored in elements or attributes alongside the content it refers to.

Although ITS provides the `its:locNote` attribute to store note text, offering the possibility of closely associating the note with the relevant content, using this approach makes it difficult to annotate the notes themselves for language, directionality, etc.

It can be argued that notes, being metadata, have different requirements to the content itself. Schema developers should carefully consider which approach to use. If all notes will always be written by English-speaking content developers, it may be acceptable to use attribute values, but if notes may be written by content developers in Arabic or Hebrew, they are almost certainly going to want to use directional markup and span elements in the notes themselves, so an element-based approach would almost certainly be better.

Handling markup not in the ITS namespace

If you are working with an existing schema where there is a way to provide notes to the localizers that is not implemented using ITS, you should provide an ITS Rules document where you use the `its:locNoteRule` element to associate your notes markup with its equivalent in ITS.

Example 12: Document with custom localization notes

In this document the `comment` element is a note for its sibling `text` element.

```
<messages>
 <msg id="ERR_NOFILE">
  <text>The file '{0}' could not be found.</text>
  <comment>The variable {0} is the name of a file.</comment>
 </msg>
</messages>
```

The `its:locNoteRule` element specifies that the `text` elements have an associated localization description in their sibling `comment` elements.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:locNoteRule selector="//msg/text" locNoteType="description"
  locNotePointer="../comment"/>
</its:rules>
```

Why do this

To assist the translator to achieve a correct translation, authors may need to provide information about the text that they have written. For example, the author may want to do the following:

- Tell the translator how to translate part of the content (e.g. "Leave text in uppercase").
- Expand on the meaning or contextual usage of a particular element, such as what a variable refers to or how a string will be used on the UI.
- Clarify ambiguity and show relationships between items sufficiently to allow correct translation (e.g. in many languages it is impossible to translate the word 'enabled' in isolation without knowing the gender, number and case of the thing it refers to.)
- Explain why text is not to be translated, point to text reuse, or describe the use of conditional text.
- Indicate why a piece of text is emphasized (important, sarcastic, etc.)

Resources:

Reference links

- The "Localization Note" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#locNote-datacat

Best Practice 9: Defining markup for unique identifiers

Provide a way for authors to assign unique identifiers to localizable elements.

How to do this

Make sure that elements with translatable content can be associated with a unique identifier.

It is strongly recommended that you define such identifiers as attributes of type ID, following the rules described in *xml:id Version 1.0* [xml:id]. This allows XML applications to take advantage of the built-in processes associated with that datatype, such as validation.

It is also recommended that you name such attributes `xml:id`[28] to increase interoperability.

> **Note:** Unique identifiers are most useful when their values are *globally unique* (i.e. unique across any documents) and *persistent* (i.e. ones which do not change over time).

Why do this

In order to most effectively reuse translated text where content is reused (for example across updates) it is necessary to have a unique and persistent identifier associated with the element.

This identifier allows the translation tools to correctly track an item from one version or location to the next. After ensuring that this is the same item, the content can be examined for changes, and if no change has taken place the potential for reuse of the previous translation is very high.

Change analysis of this kind constitutes an extremely powerful productivity tool for translation when compared to typical source matching techniques (a.k.a. translation memory). These techniques simply look for similar source text in a multilingual database without, most of the time, being able to tell whether the context of its use is the same.

Identifiers can also be helpful to track displayed text back to its underlying source. For example, when reviewing a translated user interface, the identifiers can be used as temporary prefixes to the text so that any correction can be efficiently done on the proper strings.

Resources:

Reference links

- W3C Recommendation: xml:id
  http://www.w3.org/TR/2005/REC-xml-id-20050909/

28 ⇢ http://www.w3.org/TR/2005/REC-xml-id-20050909/

Best Practice 10: Identifying terminology-related elements

Document in an ITS Rules document what elements are related to terms and term-related information.

The ITS Terminology data category provides the `its:termRule` element to address this requirement.

How to do this

Provide an ITS Rules document where you use `its:termRule` elements to indicate which elements are terms and information related to them (e.g. definitions).

> **Note:** The information identified through the `its:termInfoRef` can be of any type (e.g. human-readable or machine-specific). It is up to the application processing the data to make the distinction.

Example 13: Document with terminology-related elements

In this document, the elements `term` and `dt`, as well as any element with a `syn` attribute, denote terms. In addition, they can all have associated information.

```
<myDoc>
 <body>
  <p>A <term def="d001" syn="#alterego">doppelgänger</term>
  is basically <def xml:id="d001">the counterpart of a
  person</def>. It is almost the same as an
  <emph syn="#alterego">alter ego</emph>, but with a more sinister
  connotation. Sometimes the word <emph syn="#alterego">fetch</emph>
  is also used.</p>
 </body>
 <definitions>
  <entry xml:id="alterego">
   <dt>alter ego</dt>
   <dd>A second self. Figurative sense: trusted friend.</dd>
   <origin>Latin, literally: "second I"</origin>
  </entry>
 </definitions>
</myDoc>
```

The set of ITS rules below indicates:

- First `termRule`: The `term` element is a term and its associated information can be accessed in the node that has the identifier corresponding to the value in its `def` attribute.
- Second `termRule`: Any element with a `syn` attribute is considered a term and the `syn` attribute contains a URI location where some associated information can be found.
- Third `termRule`: The `dt` element is a term and its associated information is in its sibling element `dd`.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:termRule selector="//term" term="yes" termInfoPointer="id(@def)"/>
 <its:termRule selector="//*[@syn]" term="yes" termInfoRefPointer="@syn"/>
 <its:termRule selector="//dt[../dd]" term="yes" termInfoPointer="../dd"/>
</its:rules>
```

Why do this

The capability of specifying terms within the source content is important for terminology management and beneficial to translation and localization quality. For example, term identification facilitates the creation of glossaries and allows the validation of terminology usage in the source and translated documents.

Term identification is also useful for change management and to ensure source language quality.

Terms may require various associated information, such as part of speech, gender, number, term types, definitions, notes on usage, etc. To avoid associated information to be repeated throughout a document, it should be possible for identified terms to link to externalized attribute data, such as glossary documents and terminology database.

Resources:

Background information

- "Markup and Terminological Databases" in the Cover Pages
  http://xml.coverpages.org/terminology.html
- "Saving Money Through Terminology Management", article.
  http://www.lisa.org/globalizationinsider/2003/11/the_terms_of_bu.html

Reference links

- The "Terminology" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#terminology

Best Practice 11: Defining markup for specifying or overriding terminology-related information

Provide a way for authors to specify or override terminology-related information using ITS markup, or document equivalent legacy markup in an ITS Rules document.

The ITS Terminology data category provides the attributes `its:term` and `its:termInfoRef`, as well as the `its:termRule` element to address this requirement.

How to do this

Make sure the `its:term` and the `its:termInfoRef` attributes are defined for any element that has text content.

For examples of how to add attributes in your existing schema see Section 4.2: Example of adding an attribute to an existing schema on page 65.

It is also recommended to define the `its:rules` element in your schema, for example in a header if there is one. The `its:rules` element provides access to the `its:termRule` element which can be used to override terminology-related information globally.

Why do this

In some cases, the author of a document may need to change the information indicating what is a term or how to point to term information, overriding the general rules for the schema that you have specified when applying Best Practice 10: Identifying terminology-related elements on page 29.

Resources:

Background information

- "Markup and Terminological Databases" in the Cover Pages
  http://xml.coverpages.org/terminology.html
- "Saving Money Through Terminology Management", article.
  http://www.lisa.org/globalizationinsider/2003/11/the_terms_of_bu.html

Reference links

- The "Terminology" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#terminology

Best Practice 12: Working with multilingual documents

Avoid document formats that store multiple localized versions of content within the same document.

This best practice refers specifically to situations where copies of the same content are stored in multiple languages in a single document. It is perfectly acceptable to have multilingual text in a document otherwise.

How to do this

For documents that need to go through some localization tasks, always store the localized version of the text in a separate document.

Example 14: Avoiding multilingual documents

This is an example of bad design. It shows a single document that contains multiple translations of the same content:

```
<messages>
 <msg xml:id='fileNotFound'>
  <text xml:lang="en">File not found.</text>
  <text xml:lang="fr">Fichier non trouvé.</text>
 </msg>
</messages>
```

Instead, use one document for each language. Here one in English, and the other one in French. Other languages would go in similar separate documents.

```
<messages xml:lang="en">
 <msg xml:id='fileNotFound'>
  <text>File not found.</text>
 </msg>
</messages>

<messages xml:lang="fr">
 <msg xml:id='fileNotFound'>
  <text>Fichier non trouvé.</text>
 </msg>
</messages>
```

Note: It is admissible to store multilingual copies of a content in a single document **before** the document to send to localization, or **after** all localization tasks are done. For example, a final resource file could be constructed by collating the different language entries.

Note: It is admissible to provide the localizer with multilingual documents in XML formats that are specifically designed for localization, and are industry standards, like the XML Localisation Interchange File Format [XLIFF 1.2].

Why do this

There are two main reasons to avoid sending documents for localization if the source material is located in parallel with the different translations in the same document:

1. It is difficult to manage concurrent translations in all languages. It is very likely that each translation will be done by a different translator, in a different location. To facilitate this,

the document will have to be broken down into separate parts and reconstructed later on. This adds processing time, increases cost and provides more opportunities for the introduction of errors.

2. Depending on the point in the document's lifecycle, such a document may already contain translations, some up-to-date and some outdated (because the source material may have changed). In order to identify what parts need to be localized and what parts should be left alone, the document would then also need to contain custom information about localization state, which may or may not be supported by localization tools.

Best Practice 13: Naming elements and attributes

Use a meaningful and non-dynamic naming scheme for your elements and attributes.

How to do this

Make sure the names of the elements and attributes of your schema reflect their functions, rather than one possible way of rendering their content.

Example 15: Using meaningful names

This is an example of bad design. The element `b` is used for several purposes.

```
<doc>
 <p>To run the application, click the <b>Start</b> button.</p>
 <p><b>Make sure to enter your username</b>, and then
  press <b>OK</b>.</p>
</doc>
```

Instead, define different elements based on their functions rather than a pre-supposed rendering.

```
<doc>
 <p>To run the application, click the <ui>Start</ui> button.</p>
 <p><emph>Make sure to enter your username</emph>, and then
  press <ui>OK</ui>.</p>
</doc>
```

Also, if possible, avoid element names which do not follow a fixed naming scheme (for example, element names that serve also as identifiers).

Example 16: Avoid dynamic names

This is an example of bad design. The names of the elements also serve as text identifiers.

```
<strings>
 <str1>Input path:</str1>
 <str2>Help</str2>
 <str3>OK</str3>
 <str4>Cancel</str4>
</strings>
```

Instead, use elements names that follow a fixed naming scheme, and use `xml:id`[29] for the identifiers.

```
<strings>
 <str xml:id="str1">Input path:</str>
 <str xml:id="str2">Help</str>
 <str xml:id="str3">OK</str>
 <str xml:id="str4">Cancel</str>
</strings>
```

29 → http://www.w3.org/TR/2005/REC-xml-id-20050909/

Why do this

The name of an element should indicate what its function is, not how its content will be presented, because presentation may vary depending on different factors such as language, script, medium, or accessibility.

Using documents where elements or attributes do not follow a predictable naming pattern may cause problems when using XSLT-driven processes. It may also be an issue for translation tools. This is especially true if not all parts of the document are to be translated, since it would be more difficult to specify rules to distinguish the translatable nodes from the non-translatable ones.

Best Practice 14: Defining a span-like element

Provide a way for authors to annotate arbitrary content using its:span or equivalent markup.

A `span`-like element is an element that can be used to mark up arbitrary content and associate it with various properties such as directionality or language information. Examples of such an element include the `span` element in XHTML, or the `phrase` element in DocBook.

How to do this

Make sure you define a `span`-like element in your schema that will allow authors to associate arbitrary content with properties such as directionality, language information, etc.

If your schema does not already provide such an element, you could provide the `its:span` element.

The definition of the `its:span` element in the ITS Specification lists a set of ITS attributes that should be allowed on a span-like element.

Why do this

Some properties of content are applied using attributes. Directionality, terminology, localization notes, translate information, and language identification are examples of such properties. There is a need for a neutral element to delimit the run of text to which such attributes apply, since the appropriate boundaries are sometimes not delimited by other markup that is present, or perhaps those attributes are not permitted on other markup that is present.

Resources:

Reference links

- W3C Recommendation: Internationalization Tag Set (ITS)
  http://www.w3.org/TR/2007/REC-its-20070403/

Best Practice 15: Documenting internationalization and localization features of your schema

Provide an ITS Rules document containing all the ITS rules needed to interpret legacy markup, and identify translate, terminology and text segmentation information in your format.

How to do this

Make sure you document the internationalization and localization aspects of your schema by providing a set of relevant ITS rules in a single standalone ITS Rules document.

Your ITS Rules document should include the following information, when applicable:

- The correspondence between any proprietary mechanism you have to specify the language of content and `xml:lang` (see Best Practice 1: Defining markup for natural language labelling on page 10).
- The correspondence between any proprietary mechanism you have to indicate text directionality and `its:dir` (see Best Practice 2: Defining markup to specify text direction on page 13).
- What markup has translate rules different from the default expectation that elements are to be translated and attributes are not (see Best Practice 4: Indicating which elements and attributes should be translated on page 17).
- The correspondence between any proprietary mechanism you have to override translatability information and `its:translate` (see Best Practice 5: Defining markup to override translate information on page 19).
- The list of elements that should be treated as "nested" or "within text" from a segmentation viewpoint (see Best Practice 6: Providing information related to text segmentation on page 21).
- The correspondence between any proprietary mechanism you have to mark up ruby text and `its:ruby` (See Best Practice 7: Defining markup for ruby text on page 23).
- What part of your markup holds notes for the localizers (see Best Practice 8: Defining markup for notes to localizers on page 25).
- What part of your markup denotes terms and term-related information (see Best Practice 10: Identifying terminology-related elements on page 29).

You can find some examples of ITS Rules documents for existing XML formats in Section 5: ITS Applied to Existing Formats on page 67.

Why do this

Although some XML vocabularies are easy to understand or process, it is often helpful or necessary to provide explicit information about a given vocabulary. If such a vocabulary is to be used in a multilingual context, it is of high importance to provide information, such as which elements contain translatable content, because general information on purpose, general structure, and node types very often are not sufficient. In a way, this need for explicit information is related to the general good practice of documenting source code.

In XML it should come naturally to use a well-defined, structured format to capture such information. For information related to internationalization and translation, ITS Rules documents are a good choice for the following reasons:

- They are designed to take into account many important aspects of internationalization and translation.
- They capture information precisely (for example, selectors identify to which nodes a data category pertains).
- They can be processed by ITS-aware applications.
- They can be easily combined with additional structured information (e.g. related to version control, as shown in the example below).

Example 17: ITS rules embedded in a customized information file

An ITS processor should still be able to process a file as an external ITS rules file if the format of the file contains your own customized information in addition to the ITS rules. The following is an example of that.

```
<myFormatInfo xmlns:its="http://www.w3.org/2005/11/its">
 <desc>ITS rules used by the Open University</desc>
 <hostVoc>http://www.example.com/ns/myFormat</hostVoc>
 <rulesId>98ECED99DF63D511B1250008C784EFB1</rulesId>
 <rulesVersion>v 1.81 2006/03/28 07:43:21</rulesVersion>
 <its:rules version="1.0">
  <its:translateRule selector="//header" translate="no"/>
  <its:translateRule selector="//term" translate="no"/>
  <its:termRule selector="//term" term="yes"/>
  <its:withinTextRule withinText="yes" selector="//term|//b"/>
 </its:rules>
</myFormatInfo>
```

Resources:

Reference links

- W3C Recommendation: Internationalization Tag Set (ITS) http://www.w3.org/TR/2007/REC-its-20070403/

3 When Authoring XML Content

Authors of XML content should consider the following best practices:

Best Practice	Summary
Specifying the language of content on page 40	Use `xml:lang` (or its equivalent in your schema) on the root element of the document, and on each element where the language of the content changes.
Specifying text directionality on page 43	By default the text directionality in an XML document is assumed to be left-to-right. Use `its:dir` (or its equivalent in your schema) on the root element of any document where the text runs predominantly from right-to-left, and on elements where the Unicode bidirectional algorithm needs help to achieve proper display of bidirectional text.
Overriding information about what should be translated on page 45	Use `its:translate` (or its equivalent in your schema) on each element for which the translatability property is different from the defaults set for your schema.
Assigning unique identifiers on page 48	Use unique identifiers in the way provided by your schema on each element that constitutes a segmentation boundary. If possible use *globally unique* and *persistent* values as identifier values.
Avoiding CDATA sections on page 49	Do not put content that will be translated into CDATA sections.
Providing notes for localizers on page 52	Use `its:locNote`, `its:locNoteType` and `its:locNoteRef` (or their equivalents in your schema) to provide notes to the localizer.
Working with inserted text on page 54	Use inserted text only when the text is self-contained and does not affect its surrounding context. For example, titles and quotations are inserted text that, usually, would not cause problems. Avoid using inserted text that has any dependence on the context where it is inserted.
Identifying terms on page 57	Use `its:term` and `its:termInfoRef` (or their equivalent in your schema) to mark terms and supply term-related information.
Storing markup from another format on page 59	If possible, use the XML namespace mechanism to store different vocabularies inside a single XML document.

A number of these practices can be followed only when the XML application has been internationalized properly using the design guidelines in Section 2: When Designing an XML Application on page 6.

Best Practice 16: Specifying the language of content

Specify the natural language of your content using xml:lang, or an equivalent mechanism provided by your document format.

Your schema should provide the `xml:lang` attribute (or an equivalent mechanism) for specifying the language of content. See Best Practice 1: Defining markup for natural language labelling on page 10 for more information.

How to do this

Use `xml:lang` (or its equivalent in your schema) on the root element of the document, and on each element where the language of the content changes. The elements without declarations inherit the language information from their parents. Attribute values are deemed to be in the same language as the element where they are declared.

Make sure the values of `xml:lang` conform to *Tags for Identifying Languages* [BCP 47]. For a brief introduction to how to form language values using BCP 47 see Language tags in HTML and XML[30].

Example 18: Declaring language information with `xml:lang`

In this example, the main content of the document is in English, while a short citation in the `q` element is identified as French using `xml:lang` set to `fr`.

```
<document xml:lang="en">
 <para>The motto of Québec is the short phrase:
  <q xml:lang="fr">Je me souviens</q>. It is chiseled on
  the front of the Parliament Building.</para>
</document>
```

If the schema you are using does not provide an `xml:lang` attribute, use the equivalent attribute.

Example 19: Declaring language information with a non-standard mechanism

In this example, the schema for this document type uses a non-standard way to specify language: a `code` attribute. Authors should use that mechanism, not `xml:lang`, because the developer of the `stringList` document type should provide, along with the schema, an ITS Rules document (shown below) that declares `code` to be equivalent to `xml:lang` when used with the `lang` element.

```
<stringList>
 <msg id="connected">
  <lang code="cs">Jste připojeni k Internetu.</lang>
  <lang code="de">Sie sind an das Netz angeschlossen.</lang>
  <lang code="fr">Vouz êtes connecté à la Toile.</lang>
  <lang code="it">Sei connesso al Web.</lang>
  <lang code="ja">インターネットに接続しました。</lang>
  <lang code="ko">웹에 연결되었습니다.</lang>
  <lang code="ru">Вы подключены к Интернету.</lang>
```

30 → http://www.w3.org/International/articles/language-tags/

```
  </msg>
</stringList>
```

Note: This example is a multilingual document, which has its own set of issues as described in Best Practice 12: Working with multilingual documents on page 32.

The developer of the `stringList` document type should provide an ITS Rules document in compliance with Best Practice 1: Defining markup for natural language labelling on page 10 for existing schemas. Here the `its:langRule` element defines the `code` attribute of the `lang` element to be equivalent to `xml:lang`.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:langRule selector="//lang[@code]" langPointer="@code" />
</its:rules>
```

Note: In some cases, a change in language has implications for translation. For example, content in a different language may have to remain untranslated, or require specific handling. Such information could be provided to the localizer using `its:translate` or `its:locNote` (or their equivalents in your schema). For more details, see Best Practice 18: Overriding information about what should be translated on page 45 and Best Practice 21: Providing notes for localizers on page 52.

Why do this

Knowing the language of the content is very important in many situations. These include:

- Selection of a proper font (e.g. for Traditional or Simplified Chinese.)
- Processing of the text for wrapping and hyphenation.
- Providing spell-checking or grammar verification of the text.
- Selecting proper automated selections of text such as quotation marks or other punctuation signs.
- Using the text with voice browsers.

Resources:

Background information

- Internationalization FAQ: xml:lang in XML document schemas.
  http://www.w3.org/International/questions/qa-when-xmllang

Reference links

- BCP 47 - Tags for Identifying Languages.
  http://www.rfc-editor.org/rfc/bcp/bcp47.txt
- Description of the language identification mechanism in the XML specification.
  http://www.w3.org/TR/REC-xml/#sec-lang-tag
- Language tags in HTML and XML.
  http://www.w3.org/International/articles/language-tags/

- Tagging text with no language.
  http://www.w3.org/International/questions/qa-no-language

Test data

- I18N Tests: Automatic font assignment for CJK text (for XHTML).
  http://www.w3.org/International/tests/sec-cjk-fonts.html

Best Practice 17: Specifying text directionality

Use dedicated markup to specify the directionality of your text content.

Your schema should provide `its:dir` (or an equivalent mechanism) to manage directionality. See Best Practice 2: Defining markup to specify text direction on page 13.

How to do this

By default the text directionality in an XML document is assumed to be left-to-right. Use `its:dir` (or its equivalent in your schema) on the root element of any document where the text runs predominantly from right-to-left, and on elements where the Unicode bidirectional algorithm needs help to achieve proper display of bidirectional text.

You can get additional guidance on when and how to use directional markup in Internationalization Best Practices: Handling Right-to-left Scripts in XHTML and HTML Content[31] and What you need to know about the bidi algorithm and inline markup[32]. Although the first of these references is aimed at (X)HTML authors, the advice is generally relevant for authors of most XML-based documents too.

Example 20: Declaring text directionality

In this example, the attribute `its:dir` is used to specify the directionality of a right-to-left text run in a document that is by default left-to-right.

```
<text
  xmlns:its="http://www.w3.org/2005/11/its"  xml:lang="en"
  its:version="1.0">
 <body>
  <par>In Hebrew, the title
     <quote xml:lang="he"
     its:dir="rtl">פעילות הבינאום, W3C</quote>
     means <quote>Internationalization Activity, W3C</quote>.</par>
 </body>
</text>
```

Without the markup, the Hebrew title will display incorrectly. The text 'W3C' and the comma will be to the right of the quoted Hebrew text, rather than to its left. The markup provides the contextual information that tells the user agent that the comma and 'W3C' text are part of a right-to-left flow of text.

> **Note:** This example shows the directionality of the source text correctly. This is to ensure that you understand the concepts being described. For such display, you need a sophisticated editor that resolves directionality of the source text correctly. Many editors are not yet this sophisticated. See the related discussion about Problems with bidirectional source text[33] in [Bidi in X/HTML].

31 ⇢ http://www.w3.org/TR/i18n-html-tech-bidi/
32 ⇢ http://www.w3.org/International/articles/inline-bidi-markup/
33 ⇢ http://www.w3.org/TR/i18n-html-tech-bidi/#d2e283

Note: In XML documents, using markup is more appropriate than using Unicode Bidi Embedding Controls[34]. See Bidi formatting codes vs. markup for bidi support[35] for a more detailed explanation.

You also need to use dedicated markup to apply directional information, rather than just applying CSS direction properties to ordinary elements. See CSS vs. markup for bidi support[36] for further information.

Why do this

User agents should use the Unicode Bidirectional (bidi) Algorithm and its knowledge of the directional properties of characters to decide whether a sequence of characters should flow to the left or to the right. The bidi algorithm can also handle simple cases where right-to-left and left-to-right text are mixed. However, situations commonly arise where higher level contextual information is needed to achieve the desired layout of bidirectional text. This contextual information can be provided by markup in XML. Such markup also affects page layout behavior. For example, in a right-to-left context, table columns are ordered right-to-left, list bullets appear to the right of text, the page is right-aligned, and so forth.

Note: Directionality information cannot be deduced from language markup:

- There is not necessarily a one-to-one match between a given language and what directionality to use. For example, Azerbaijani can be written using both right-to-left and left-to-right scripts, and the language code `az` is relevant for either.
- The values of inline directional markup are not necessarily aligned with the values of markup about the language. For example, a part of a document might be declared as having right-to-left directionality, but there might be only a general language declaration for a left-to-right script language available, like `fr`.
- Markup used to indicate directionality has values that indicate that the normal directionality should be overridden; it is not possible to indicate that using language related values.

Resources:

Background information

- Internationalization FAQ: What you need to know about the bidi algorithm and inline markup
  http://www.w3.org/International/articles/inline-bidi-markup/
- Unicode Technical Report #20: Unicode in XML and other Markup Languages
  http://www.w3.org/TR/2007/NOTE-unicode-xml-20070516/

Reference links

- The "Directionality" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#directionality

34 → http://www.w3.org/TR/2007/NOTE-unicode-xml-20070516/#Bidi
35 → http://www.w3.org/International/questions/qa-bidi-controls
36 → http://www.w3.org/International/questions/qa-bidi-css-markup

Best Practice 18: Overriding information about what should be translated

Use available markup to specify any content where the choice to translate or not is different from the default for your schema.

Your schema should provide `its:translate` (or an equivalent mechanism) to allow you to override defaults. See Best Practice 5: Defining markup to override translate information on page 19.

The ITS default is that element content should be translated and attribute content should not. Developers of your schema should also have documented any schema-specific defaults for your document type where these differ from the ITS default.

How to do this

Use `its:translate` (or its equivalent in your schema) on each element for which the translatability property is different from the defaults set for your schema.

Example 21: Overriding default translation rules

In the following document, although the content of the `par` elements should normally be translated, in this instance the last `par` should remain in English. Using `its:translate` the author can indicate that the last `par` should not be translated.

Note that the author does not need to specify that the `head` element should not be translated, because this is defined for all documents of type `myDoc` by the ITS Rules document provided by the developer of the `myDoc` schema (see just below).

```
<myDoc xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
<head>
  <lastRev>2007-10-23 041254Z</lastRev>
  <docID>1A454AE4-7EB8-4ed2-A58E-1EC7F75BB0D5</docID>
 </head>
 <par>To apply these terms to you library, attach the following notice.
  It is safest to attach it to the start of each source file to most
  effectively convey the exclusion of warranty; and each file should
  have at least the "copyright" line and a pointer to where the full
  notice is found.</par>
  <par>The notice should read (preferably in English):</par>
  <par its:translate="no">This library is free software; you can
  redistribute it and/or modify it under the terms of the GNU Lesser
  General Public License as published by the Free Software Foundation;
  either version 2.1 of the License, or (at your option) any later
  version. This software is distributed as open source under LGPL.</par>
 </myDoc>
```

This is the ITS Rules document created by the developer of the `myDoc` document type (implementing Best Practice 4: Indicating which elements and attributes should be translated on page 17). These rules override the ITS default that all element content should be translated, but attribute values should not.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <its:translateRule selector="/myDoc/head" translate="no"/>
 <its:translateRule selector="//img/@alt" translate="yes"/>
</its:rules>
```

This is what the rules mean:

- First `translateRule`: The `head` element and its children should not be translated.
- Second `translateRule`: The `alt` attribute of any `img` element should be translated.

To override translate information for attributes, you have to use an `its:translateRule` element in your document.

Example 22: Overriding default translation rules for attributes

This document is of the same type as the one in Example 21 and uses the same ITS rules, therefore the `alt` attribute should normally be translated. Because in this specific document the images refer to a user interface that will not be translated (whereas the document will be), the author needs to override the rule that all `alt` attributes should be translated. This is done at the top of the document, using `its:translateRule`.

```
<myDoc xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
 <head>
  <lastRev>2007-11-12 234503Z</lastRev>
  <docID>D1EA7453-DC53-488a-B950-137BE0EF5253</docID>
  <its:rules>
   <its:translateRule selector="//img[@role='ui']/@alt" translate="no"/>
  </its:rules>
 </head>
 <par>Once you have selected your options, click the
  <img xml:lang="en-us" src="runBtn.png" role="ui" alt="Run"/> button
  to start the process.</par>
</myDoc>
```

The `its:translateRule` element says that the alt text of images referring to UI buttons in the document should be left untranslated.

Note: Authors should NOT use `its:translate` to tag single words or terms that (they think) are likely to remain the same when translated into a given target language (e.g. loan-words). This type of decision is normally made during translation.

Authors may decide *what* is translatable, but not *how* to translate it.

Example 23: XML document with inappropriate usage of `its:translate`.

This is an example of bad design. In this document `its:translate` is used to markup a proper name and two loan words in an attempt to indicate that they should not be translated. You should **not** do this.

```
<book xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
 <body>
  <p>Everything started when <span its:translate="no">Zebulon</span>
  discovered that he had a <span its:translate="no">doppelgänger</span>
  who was a serious baseball <span its:translate="no">aficionado</span>.</p>
 </body>
</book>
```

It may, however, be useful to the translator to mark up loan-words or any special words in this example as *terms*, as described in the section Best Practice 23: Identifying terms on page 57.

Why do this

Although the set of ITS rules provided with the schema should specify any exceptions to the default ITS translation rules for a given schema (see Best Practice 4: Indicating which elements and attributes should be translated on page 17), there are cases where these general rules need to be overridden for specific elements, in specific documents. It is up to the author of the content to indicate these cases using markup.

Best Practice 19: Assigning unique identifiers

Assign a unique identifier to elements that correspond to segmentation boundaries.

Your schema should provide `xml:id`[37] (or an equivalent mechanism) to allow you to assign unique identifiers to elements. See Best Practice 9: Defining markup for unique identifiers on page 28.

Segmentation refers to how text is broken down, from a linguistic viewpoint, into units that can be stored separately and handled by processes such as translation. The schema author ought to create a list of these elements where they differ from the ITS defaults (see Best Practice 6: Providing information related to text segmentation on page 21).

How to do this

Use unique identifiers in the way provided by your schema on each element that constitutes a segmentation boundary.

> **Note:** Often, ids are automatically assigned by authoring or content management applications. Thus, authors may not have to worry about them in some cases.

If possible use *globally unique* and *persistent* values as identifier values.

Why do this

Providing unique identifiers can be very useful for change analysis, text tracking, and various other tasks often utilized during the authoring and the localization of documents.

This is explained in more detail in Best Practice 9: Defining markup for unique identifiers on page 28.

Resources:

Background information

- "Give unique identifiers to elements in XML documents", note.
  http://www.ibm.com/developerworks/xml/standards/x-xmlidspec.html

Reference links

- W3C Recommendation: xml:id
  http://www.w3.org/TR/2005/REC-xml-id-20050909/

37 → http://www.w3.org/TR/2005/REC-xml-id-20050909/

Best Practice 20: Avoiding CDATA sections

Avoid using CDATA sections for content that will be translated.

CDATA sections are often used to place programming code or other special vocabularies in XML with minimal effort. There are often better ways of including such content.

How to do this

Do not put content that will be translated into CDATA sections.

Example 24: Avoiding the use of CDATA sections

This is an example of bad design. In this document, part of the content is in a CDATA section. It is no longer possible to mark up that content for language changes, terms, text direction, translate information, or any of the other things that may be needed to facilitate localization.

```
<myData>
 <item course="12" page="2">
  <title>Accessing the R&amp;D facilities</title>
  <body><![CDATA[The R&D facilities are located in the South wing
   of Building 12-W, in the East quarter of the section Q.
   IMPORTANT ==> These facilities are accessible only to personal with
   Class Omega-45Q1 clearance.]]></body>
 </item>
</myData>
```

Instead, use normal XML for your content. This allows you to tag the content as needed. For instance, here the author has added some terminology markup.

```
<myData xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
 <item course="12" page="2">
  <title>Accessing the R&amp;D facilities</title>
  <body>The R&amp;D facilities are located in the South wing
   of Building 12-W, in the East quarter of the section Q.
   IMPORTANT ==&gt; These facilities are accessible only to personal with
   <span its:term="yes">Class Omega-45-Q1</span> clearance.</body>
 </item>
</myData>
```

If the CDATA section encloses a large, self-contained block of data, such as a script or an XML example, you may be able to replace the section by some inclusion mechanism such as XInclude or XLink.

Example 25: Replacing CDATA sections with XLink

In SVG you can place a script directly into an SVG document, in which case you usually use CDATA sections to avoid having to escape characters in the script's code.

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="6cm" height="5cm" viewBox="0 0 600 500"
     xmlns="http://www.w3.org/2000/svg" version="1.1">
  <!-- Script is inlined and enclosed in CDATA section  -->
  <script type="text/ecmascript"> <![CDATA[
    function circle_click(evt) {
      var circle = evt.target;
```

```
        var currentRadius = circle.getAttribute("r");
        if (currentRadius < 100)
          circle.setAttribute("r", currentRadius*2);
        else
          circle.setAttribute("r", currentRadius*0.5);
      }
    ]]> </script>
    <rect x="1" y="1" width="598" height="498" fill="none" stroke="blue"/>
    <circle onclick="circle_click(evt)" cx="300" cy="225" r="10"
            fill="red"/>
    <text x="300" y="480"
          font-family="Verdana" font-size="35" text-anchor="middle">
      Click on circle to change its size
    </text>
  </svg>
```

Instead, you could use XLink to store the script in a separate file and reference it from the SVG document.

```
  <?xml version="1.0" encoding="utf-8"?>
  <svg width="6cm" height="5cm" viewBox="0 0 600 500"
       xmlns="http://www.w3.org/2000/svg" version="1.1"
       xmlns:xlink="http://www.w3.org/1999/xlink">
    <!-- Script is included from external file  -->
    <script type="text/ecmascript" xlink:href="animate.js"/>
    <rect x="1" y="1" width="598" height="498" fill="none" stroke="blue"/>
    <circle onclick="circle_click(evt)" cx="300" cy="225" r="10"
            fill="red"/>
    <text x="300" y="480"
          font-family="Verdana" font-size="35" text-anchor="middle">
      Click on circle to change its size
    </text>
  </svg>
```

Example 26: Replacing CDATA sections with XInclude

It is quite common to use CDATA sections to put examples of source code into XML documents. The following example shows how to do this using DocBook.

```
  <?xml version="1.0" encoding="utf-8"?>
  <example xmlns="http://docbook.org/ns/docbook">
    <title>Skeleton of XHTML page</title>
    <programlisting><![CDATA[<html xmlns="http://www.w3.org/1999/xhtml"
     xml:lang="en">
    <head>
      <title>… page title goes here …</title>
    </head>
    <body>
      … page content goes here …
    </body>
  </html>]]></programlisting>
  </example>
```

Instead, you could use XInclude to store the example code in a separate file and include it during at processing time. Note that you have to use `parse="text"` to treat the included file as plain text rather than markup.

```
  <?xml version="1.0" encoding="utf-8"?>
  <example xmlns="http://docbook.org/ns/docbook"
```

```
    xmlns:xi="http://www.w3.org/2001/XInclude">
    <title>Skeleton of XHTML page</title>
    <programlisting><xi:include href="EX-xhtml-skeleton.xhtml"
           parse="text"
           encoding="utf-8"/></programlisting>
  </example>
```

If you must use CDATA sections:

- Document the type of content (for example with an attribute set to the appropriate MIME-type). This may help tools to use an appropriate parser to process the content.
- Aim to produce well-formed content. This will allow parsers to process the content more easily.

Note: CDATA is often used to store textual content containing HTML or XML tags. This is not recommended. See Best Practice 24: Storing markup from another format on page 59 for more details.

Note: Using CDATA does not affect whether white-space is preserved or not by XML processors. To preserve white-space use the `xml:space` attribute with the value `preserve`.

Why do this

The use of CDATA sections prevents the insertion of markup for internationalization or localization purposes. For example, tags to denote change of directionality, or language, or to add localization notes, cannot be used with the content inside CDATA sections.

Numeric character references and entity references are not supported in CDATA sections either. This could lead to a possible loss of data if the document is converted from one encoding to another, or when translating.

Mixing content in CDATA sections and content not in CDATA sections in the same document causes more work when doing some tasks with non-XML-aware tools. For example, when searching for the text "R&D" the user has to search both for `R&D` (for the CDATA sections) and `R&amp;D` (for the normal content).

Best Practice 21: Providing notes for localizers

Use dedicated markup to provide notes where you can communicate useful information for the localizer

Your schema should provide `its:locNote`, `its:locNoteType`, and `its:locNoteRef` (or equivalent mechanisms) to allow you to communicate with those who will localize your content. See Best Practice 8: Defining markup for notes to localizers on page 25.

How to do this

Use `its:locNote`, `its:locNoteType` and `its:locNoteRef` (or their equivalents in your schema) to provide notes to the localizer.

This is especially important for content with inserted text where the translator will need context to translate more accurately.

Example 27: Annotating an XML document for localization

In this document two ITS local attributes are used to annotate an XSLT template:

- `its:locNoteRef` is used to point to an explanation of the acronym RFID.
- `its:locNote` is used to indicate what kind of value the element `<xsl:value-of select="PNum"/>` corresponds to.

Note: When working with XSLT, you need to decide whether the ITS markup should be in the output or not, and may have to use different markup accordingly. In this example, the ITS attributes do not appear in the output.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:its="http://www.w3.org/2005/11/its"
     its:version="1.0">
 <xsl:template match="/data">
  <xsl:variable name="Lang" select="Lang"/>
  <xsl:variable name="EMail" select="EMail"/>
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="{$Lang}" lang="{$Lang}">
   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Login</title>
   </head>
   <body>
    <p>Login Into Queztal-Systems</p>
    <form method="POST">
     <table border="0" id="table2">
      <tr><td>First, place your pass card in front of the reader to scan your
       <xsl:text its:locNoteRef="http://en.wikipedia.org/wiki/RFID">RFID</xsl:text>.
       When the light turns green, enter your password in the box below, and
       click Submit.</td></tr>
      <tr><td><input type="password" name="pword" size="25"/></td></tr>
     </table>
     <p><input type="submit" value="Submit" name="go"/></p>
    </form>
```

```
      <p>If you have difficulties login in, please call
       <xsl:value-of select="PNum" its:locNote="Toll-free phone number"/>,
       or send an email to
       <a href="mailto:{$EMail}"><xsl:value-of select="EMail"/></a>.</p>
     </body>
    </html>
   </xsl:template>
  </xsl:stylesheet>
```

Why do this

There are many reasons to provide information to localizers. You may want to:

- Expand on the meaning or contextual usage of a particular element, such as what a variable refers to or how a string will be used in the user interface.
- Clarify ambiguity and show relationships between items sufficiently to allow correct translation. For example, in many languages it is impossible to translate the word "enabled" in isolation without knowing the gender, number and case of the thing it refers to.
- Explain why text is not translated, point to text reuse, or describe the use of conditional text.
- Indicate why a piece of text is emphasized (important, sarcastic, etc.)

Using XML comments for doing this may not be enough as they may get stripped out or ignored during the localization process.

Best Practice 22: Working with inserted text

Make sure that any piece of inserted text is grammatically independent of its surrounding context.

Inserted text refers to any text that is marked by a placeholder in the source XML document and automatically inserted within text content when the document is processed.

Types of inserted text include:

- Boilerplate text reused in different contexts.
- Various parts of a sentence composed by bringing together separate pieces of text.
- Variable placeholders that are replaced by their values when the document is processed.

The implementation of such text can be done in different ways in XML. Some examples are:

- Using entity references.
- Using XSLT processing.
- Using XInclude mechanisms.
- Using XLink mechanisms.
- Using a custom mechanism specific to a given format (e.g. the `conref` attribute in [DITA 1.0]).

How to do this

Use inserted text only when the text is self-contained and does not affect its surrounding context. For example, titles and quotations are inserted text that, usually, would not cause problems.

Avoid using inserted text that has any effect or dependence on the context where it is inserted.

For additional background information about issues and approaches related to text insertion and re-use of text see the articles Working with Composite Messages[38] and Re-using Strings in Scripted Content[39].

If you do insert text, use `its:locNote` or `its:termInfoRef` (or their equivalents in your schema) to provide the localizers with some context. See Best Practice 21: Providing notes for localizers on page 52 and Best Practice 23: Identifying terms on page 57.

Example 28: Providing context to variables.

In this example, in the first message, the element `var` is used to insert the name of a printer. In the second example, it is used to insert a filename. The `its:locNote` attribute is utilized to provide a description of what the variables represent. This may help in deciding how to translate each message.

38 → http://www.w3.org/International/articles/composite-messages/
39 → http://www.w3.org/International/articles/text-reuse/

```
<strings xmlns:its="http://www.w3.org/2005/11/its"
 xml:lang="en" its:version="1.0">
 <msg id="pmAdded">The printer <var arg="0" its:locNote="Printer name"/>
  has been added to the list.</msg>
 <msg id="fmAdded">The file <var arg="0" its:locNote="Filename"/>
  has been added to the list.</msg>
 </strings>
```

This is a French translation of the document shown above. The context provided allowed to disambiguate the variable and to get a more accurate translation.

```
<strings xmlns:its="http://www.w3.org/2005/11/its"
 xml:lang="fr" its:version="1.0">
 <msg id="pmAdded">L'imprimante <var arg="0" its:locNote="Printer name"/>
  a été ajoutée à la liste.</msg>
 <msg id="fmAdded"><var arg="0" its:locNote="Filename"/>
  a été ajouté à la liste.</msg>
</strings>
```

Why do this

If not used properly, inserted text can cause important (and sometimes unresolvable) problems during localization. Consider the following:

Example 29: Using `conref` in DITA

This is an example of bad design. In this example, the author, working with the DITA format [DITA 1.0], decided to reference a term in a termbase by using the `conref` mechanism. In this case, the term `t123` in `termbase.xml` has the value 'hydraulic lift'.

```
<p>Using a <term conref="termbase.xml#t123"/>, raise the vehicle from the ground.</p>
```

At a first glance the example above seems to work fine in English. However, such a construction has several problems:

- You should not separate the article from the noun. If "hydraulic lift" is independently replaced in the future by some other term, you may need to change the article to 'an' or remove it.
- The article/noun separation also causes trouble for the translators. Without any easy way to see the actual term when translating the paragraph, they may not be able to decide the gender or number of the article.
- If it is used at the beginning of a different sentence, the term would need to be capitalized.
- The term is singular in the termbase, but it may need to be plural somewhere else in the document.
- In inflected languages the form required in the text may be different from the form stored in the termbase. For example, in Polish the term would be stored in its nominative form ("dźwignia hydrauliczna"), while it should be in its instrumental form once inserted in this context: "Używając dźwignię hydrauliczną podnieś pojazd z ziemi."

Resources:

Background information

- Internationalization article: Working with Composite Messages. http://www.w3.org/International/articles/composite-messages/
- Internationalization article: Re-using Strings in Scripted Content http://www.w3.org/International/articles/text-reuse/

Best Practice 23: Identifying terms

Use dedicated markup to identify any terminology-related content.

What constitutes a term depends on many factors specific to each organization and project. Terms may include for example names of features, programs, services, and so forth. They also may include words or expressions that are specific to the domain to which the content pertains, such as technical terms, or legal terms, and they may include terms that simply occur often and should be translated consistently.

How to do this

Use `its:term` and `its:termInfoRef` (or their equivalent in your schema) to mark terms and supply term-related information.

Your schema should provide `its:term` and `its:termInfoRef` (or equivalent mechanisms). See Best Practice 11: Defining markup for specifying or overriding terminology-related information on page 31.

You should also override default terminology rules as needed.

Example 30: Identifying terminology-related content

In this document, terms are normally denoted with a `term` element. Following Best Practice 10: Identifying terminology-related elements on page 29, the developer of the schema has provided an ITS Rules document that defines such property for `term`.

However, in this specific document, the author wants to indicate the following:

- The content of any `ui` element should be seen as a term.
- The text `Vector Files` in the title is a term.

In the first case, the author uses a `its:termRule` element in the header of the document to indicate that any `ui` element in this document is a term. This is more efficient than adding an attribute for each instance of `ui` in the body of the document.

In the second case, because the schema does not allow the element `term` to be used in `title` (an oversight of the developer), the author uses a simple `span` element with `its:term` and `its:termInfoRef` to associate `Vector Files` with its corresponding term information.

```
<myManual xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
 <head>
  <its:rules>
   <its:termRule selector="//ui" term="yes"/>
  </its:rules>
  <title>Generating <span its:term="yes" its:termInfoRef="#vFile">Vector
   Files</span></title>
 </head>
 <body>
  <par>Select the command <ui>Build Output Files</ui> from the
   <ui>Tasks</ui> menu to generate the final <term ref="vFile">vector
   files</term>.</par>
 </body>
```

```
    <extra>
     <terms>
      <termDef xml:id="vFile">A <emph>vector file</emph> is a binary document
       that contains the complete set of vectors needed to draw the background
       layer of a map.</termDef>
     </terms>
    </extra>
   </myManual>
```

This ITS Rules document is the one created by the developer of the `myManual` document type (in implementing Best Practice 10: Identifying terminology-related elements on page 29). It provides one `termRule` element indicating that any `term` element is a term and its associated information is located in the element that is identified with the value stored in the `ref` attribute of `term`.

```
   <its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
    <its:termRule selector="//term" term="yes" termInfoRef="id(@ref)"/>
   </its:rules>
```

Why do this

If you do not indicate what words are terms of interest in the content, the translators will not know that these terms need to be translated consistently. Often, multiple translators are working on different files in a given project, and the way they choose to translate specific words can be inconsistent with the way that other translators have translated them. If important terms are marked in the content, they can extract these terms from the content before the content is translated, and pre-translate them in the form of a shared electronic dictionary. This ensures consistency of translation of important terms.

While markup denoting terms for a given schema level should be specified in a set of ITS rules provided with the schema (See Best Practice 10: Identifying terminology-related elements on page 29), there are cases where these general rules need to be overridden or complemented for specific elements, in specific documents. It is up to the author of the content to provide such overriding markup.

Resources:

Reference links

- The "Terminology" data category in ITS.
  http://www.w3.org/TR/2007/REC-its-20070403/#terminology

Best Practice 24: Storing markup from another format

Avoid escaping markup to enable storage of markup from another format.

How to do this

If possible, use the XML namespace mechanism to store different vocabularies inside a single XML document.

Example 31: How to avoid including markup in escaped form

In this document, the elements `top` and `body` both contain HTML markup coded as text. There is no easy way to make the distinction between the HTML markup and the HTML text content.

```
<pages>
 <row>
  <key>ENConvClasses</key>
  <top>&lt;span class="h1"&gt;Elibur Library&lt;/span&gt; - Conversation Groups</top>
  <body><![CDATA[<p>These small discussion groups meet <b>weekly</b> and are for
people learning English. Each group is led by a volunteer who is a native speaker
of American English. Groups converse about books, articles, and other materials.</p>
<p>Space is limited. Ask for availability to <a href="mailto:enconv@elibur-lib.com">
enconv@elibur-lib.com</a>.</p>]]></body>
 </row>
</pages>
```

Instead, use the XML namespace mechanism. Here the content of `top` and `body` is now a mix of text and XHTML elements. This avoid any confusion between text and HTML tags.

```
<pages xmlns:h="http://www.w3.org/1999/xhtml">
 <row>
  <key>ENConvClasses</key>
  <top><h:span class="h1">Elibur Library</h:span> - Conversation Groups</top>
  <body><h:p>These small discussion groups meet <h:b>weekly</h:b> and are for
people learning English. Each group is led by a volunteer who is a native
speaker of American English. Groups converse about books, articles, and
other materials.</h:p>
<h:p>Space is limited. Ask for availability to <h:a
href="mailto:enconv@elibur-lib.com">enconv@elibur-lib.com</h:a>.</h:p></body>
 </row>
</pages>
```

Another alternative to using markup as text is to store it externally and include it into the document using a mechanism such as XInclude or XLink.

If you must include markup as text content:

- Make sure to document the type of content, for example with an attribute set to the appropriate MIME-type. This may help tools to use a more appropriate parser to process the given content.
- Aim at having the content well-formed. This will allow parsers to process it more easily.

Why do this

Some XML documents are used to store different types of data for purposes such as exchange or export. In some cases such data is itself XML data. For example, some XHTML content stored in a database can be exported to an XML container file for localization and re-imported back into the database.

> **Note:** The use of escaping for literal examples of markup is not a problem. The issue is only for large volume of XML/HTML data contained in another XML document.

Storing such XML data inside XML elements as text content (i.e. with its markup tags escaped), has several drawbacks:

- Any handling of such content is made difficult by the impossibility to separate text from markup without extra processing.
- Often, such content is put in CDATA sections, which has its own set of issues. See Best Practice 20: Avoiding CDATA sections on page 49.
- The escaped markup cannot be validated.
- If there is a process turning markup into escaping, there is the danger of double escaping.

4 Generic Techniques

This section provides a set of generic techniques that are applicable to various guidelines; for example, how to add ITS attributes to different types of schemas, or how to optimize XPath expressions for the ITS `selector` attribute.

4.1 Writing ITS Rules

Whether they are external or embedded, there are a few things you should take into consideration when writing ITS rules.

- Try to keep the number of nodes to be overridden to a minimum. This improves performance. For example, if most of a document should not be translated, it is better to set the root element to be non-translatable than to set all elements. The inheritance mechanism will have the same effect for a much lower computing cost.
- Because a rule has precedence over the ones before, you should start with the most general rules first and progressively override them as needed. Some rules may be more complex if they need to take into account all the aspects of inheritance.

4.1.1 Precedence and Inheritance

ITS 1.0 defines the precedence of ITS information for data categories. The precedence order for selection is as follows (starting with the highest precedence) and will be explained using the ITS Translate data category:

1. ITS local attributes on a specific element, for example the `its:translate` attribute, have the highest precedence.
2. Next are global rules, for example a set of `its:translateRule` elements for the ITS Translate data category. Individual rules in an `its:rules` element have an inherent precedence which depends on their position in the `its:rules` element: the rules at the bottom have a higher precedence than rules at the top. In addition, the rules inside a given `its:rules` element have a higher precedence than the rules linked via an `xlink:href` attribute in that same `its:rules` element.
3. Inherited ITS information constitutes the third level of precedence. The kind of inheritance is data category specific. For example, if an element has been labelled as "do not translate" using one of the means described via 1) or 2) above, this information is inherited by its child elements, but not by attributes.
4. ITS information which originates in data category specific defaults is the one with the lowest precedence. For example, the default for the ITS Translate data category is that element content is to be translated and attribute values are not to be translated.

The following example shows the usage of local and global ITS markup and how the precedence described above comes into play.

Example 32: Precedence and inheritance in ITS

In this document, all child elements within the `<text>` element are set as to 'do not translate' by the first `its:translateRule` element. However, the second and last `its:translateRule` element has higher precedence than the one before, so it can be used to describe an exception: all `<p>` elements are still to be translated. This shows the interplay between different rules and demonstrates that the last one always "wins".

Another exception to the first `its:translateRule` element is expressed with the local `its:translate` attribute on the `<notes>` element. It specifies that the content of this element should be translated. Without the `its:translate` attribute, the information from the first `its:translateRule` element would be inherited, and this `<notes>` element would not be translatable.

Finally, the content of the `<documentation>` element within the `<head>` element is also translatable, but not the content of any attributes in the document. This demonstrates the role of defaults for the ITS Translate data category.

```
<doc xmlns:its="http://www.w3.org/2005/11/its">
 <head>
  <documentation>Some translatable text.</documentation>
  <its:rules version="1.0">
   <its:translateRule selector="//text" translate="no"/>
   <its:translateRule selector="//p" translate="yes"/>
  </its:rules>
 </head>
 <text>
  <data>Some data with <code>coded parts</code> (<notes
its:translate="yes"> and translatable text</notes>).</data>
  <p>Some text with <b>bolded words</b>.</p>
 </text>
</doc>
```

4.1.2 Dealing with namespaces

When writing rules for documents that use XML namespaces you must make sure that you declare the namespaces, and use the relevant prefixes in the different XPath expressions.

Example 33: Applying ITS rules on a document containing namespaces

The first document uses several different XML vocabularies:

- The host format is not associated with any namespace. Its elements have no prefix.
- The "inventory-book" vocabulary is associated with the namespace `http://www.example.com/inventory-book`. The elements belonging to that namespace have a `bk` prefix.
- The XHTML vocabulary is associated with the namespace `http://www.w3.org/1999/xhtml`. The elements belonging to that namespace ave a `h` prefix.

- The XLink vocabulary is associated with the namespace `http://www.w3.org/1999/xlink`. There is one attribute belonging to that namespace and it has a `xlink` prefix.
- The ITS vocabulary is associated with the namespace `http://www.w3.org/2005/11/its`. There is one element belonging to that namespace and it has an `its` prefix.

```
<inventory xmlns:bk="http://www.example.com/inventory-book"
 xmlns:h="http://www.w3.org/1999/xhtml"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:its="http://www.w3.org/2005/11/its">
 <header>
  <identity>3E039D7D-B416-47e8-83B3-3F4DF9EDDB87</identity>
  <lastUpdate>2007-11-12</lastUpdate>
  <desc>Inventory made by Joan, for shelves H to K only.</desc>
  <its:rules version="1.0" xlink:href="EX-namespaces-2.xml" xlink:type="simple"/>
 </header>
 <list>
  <bk:book xml:id="item00A83">
   <bk:isbn>0312875819</bk:isbn>
   <bk:quantity>2</bk:quantity>
   <bk:type>HIST</bk:type>
   <bk:author>Bradshaw, Gillian</bk:author>
   <bk:pub>Forge Books; New Ed edition (June 2, 2001)</bk:pub>
   <bk:title>The Sand-Reckoner</bk:title>
   <bk:desc>
    <h:p>Building on a few antique facts, Bradshaw ably recreates the extraordinary
life of Archimedes, the great mathematician and engineer who lived in Syracuse from
287 to 212 B.C. After a few years studying in Alexandria, Archimedes returns home
where his father is dying and his city at war with the Romans.
<h:img src="0312875819large.png" alt="The Sand-Reckoner (by Gillian Bradshaw)"/>
</h:p>
   </bk:desc>
  </bk:book>
 </list>
</inventory>
```

The XLink and ITS namespaces are just used for associating this document with the external ITS rules file shown below.

The ITS Rules document contains several rules that determine what parts of the inventory document should be translated. The rules use XPath expressions where the elements are prefixed. These prefixes are associated with the namespaces used in the inventory. Here is a description of each `its:translateRule`, from top to bottom:

- The first indicates that the `inventory` element should not be translated. This is inherited by all the children of `inventory`. Most of the content of the inventory is not to be translated, so the easiest way to define the proper rules for this type of document is to say that the root element should not be translated, and then list all the exceptions.
- The second indicates that the `desc` element of the host format should be translated.
- The third indicates that the `title` of the `http://www.example.com/inventory-book` namespace should be translated.

- The fourth indicates that the `desc` element of the `http://www.example.com/inventory-book` namespace should be translated.
- The last indicates that the `alt` attribute in the HTML `img` element should be translated.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0"
 xmlns:book="http://www.example.com/inventory-book"
 xmlns:html="http://www.w3.org/1999/xhtml">
 <its:translateRule selector="//inventory" translate="no"/>
 <its:translateRule selector="//desc" translate="yes"/>
 <its:translateRule selector="//book:title" translate="yes"/>
 <its:translateRule selector="//book:desc" translate="yes"/>
 <its:translateRule selector="//html:img/@alt" translate="yes"/>
</its:rules>
```

4.1.3 Create your XPath expressions with care

ITS uses XPath expressions in several contexts to identify nodes. The most prominent contexts are selectors, and pointer attributes like those shown in the following rules:

```
<its:translateRule selector="//term" translate="no"/>
```

or

```
<its:locNoteRule locNoteType="description" selector="//msg/data"
 locNotePointer="../notes"/>
```

When writing ITS-related XPath expressions like the ones above, the following should be considered:

- ITS XPath expressions pertain to XPath 1.0 or its successor
- The values of ITS selector attributes are XPath absolute location paths
- The values of ITS pointer attributes are XPath relative location paths. The ITS pointer attributes are: `locNotePointer`, `locNoteRefPointer`, `its:termInfoPointer`, `its:termInfoRefPointer`, `its:rubyPointer`, `its:rtPointer`, `its:rpPointer`, `its:rbcPointer`, `its:rtcPointer`, `its:rbspanPointer`, and `its:langPointer`.

In environments where XSLT is used to process ITS-related XPath expressions, it is important to know about the subset of XPath which is termed 'XSLT patterns' (see the note in the section Global Approach of the ITS Specification). Using only XSLT patterns in ITS selector attributes helps to avoid issues which may arise with respect to the `match` attribute in XSLT `template` elements.

In addition to this general advice, you should take into account best practices related to writing XPath expressions (see for example the XPath tutorial[40]).

40 → http://www.zvon.org/xxl/XPathTutorial/General/examples.html

4.2 Example of adding an attribute to an existing schema

This example shows how to add an attribute (here `xml:lang`) to an existing document type. We will add the attribute to an element called `para`.

Note that this example only shows a few ways of adding attributes. There are many others, depending on the schema language and the modularization techniques used in the existing schema.

4.2.1 Including `xml:lang` in XML Schema

To include the `xml:lang` attribute in your XML Schema document, import the W3C xml.xsd schema into your own schema using the `xs:import` element.

Example 34: Importing the `xml:lang` declaration in XML Schema.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <!-- Import for xml:lang and xml:space -->
 <xs:import namespace="http://www.w3.org/XML/1998/namespace"
            schemaLocation="http://www.w3.org/2001/xml.xsd"/>
 ...
```

Once the xml.xsd schema is imported, you can use the reference to `xml:lang` in any of your element declarations.

Example 35: Using `xml:lang` in XML Schema.

```
...
<xs:element name="para">
 <xs:complexType>
  <xs:sequence maxOccurs="unbounded">
    ...
  </xs:sequence>
  <xs:attribute ref="xml:lang" use="optional"/>
 </xs:complexType>
</xs:element>
...
```

4.2.2 Including `xml:lang` in RELAX NG

Declare `xml:lang` directly in your schema. There is no existing declaration of, or standardized schema fragment defining, the `xml:lang` attribute in RELAX NG. You have to declare `xml:lang` directly in your schema and specify the choice of values to be either the XML Schema `language` datatype or an empty value.

Example 36: Declaration of `xml:lang` in RELAX NG.

```
<element name="para"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <attribute name="xml:lang">
    <choice>
      <data type="language"/>
      <value></value>
    </choice>
  </attribute>
  ...
</element>
```

4.2.3 Including `xml:lang` in an XML DTD

Add the `xml:lang` directly in the attribute list of your element.

For example, to add `xml:lang` to a `para` element you can specify the following in a DTD:

Example 37: Declaration of `xml:lang` in a DTD.

```
<!ELEMENT para (#PCDATA)>
<!ATTLIST para
          xml:lang CDATA #IMPLIED>
```

5 ITS Applied to Existing Formats

This section presents several examples of how ITS can be used to enhance the internationalization readiness of some well-known XML document types. These examples are only illustrative and may have to be adapted to fit the needs of each specific user.

Two topics are covered for each format:

- How should ITS be integrated in specific markup schemas? For example, for XHTML it promotes the interoperability of ITS implementations if you specify that the ITS `rules` element will always be part of the content model of the `head` element.
- How should ITS data categories be associated with existing markup declarations in a schema that have identical or overlapping purposes? For example, DITA [DITA 1.0] already has an attribute to indicate translatability of text, but doesn't have a global selection mechanism for indicating what parts of an XML document the ITS translate data category and its values should be applied to.

The following XML vocabularies are discussed:

- Section 5.1: ITS and XHTML 1.0 on page 67
- Section 5.2: ITS and TEI on page 78
- Section 5.3: ITS and XML Spec on page 80
- Section 5.4: ITS and DITA on page 84
- Section 5.5: ITS and GladeXML on page 88
- Section 5.6: ITS and DocBook on page 90

5.1 ITS and XHTML 1.0

XHTML [XHTML 1.0] is a reformulation of the three HTML 4 document types as applications of XML 1.0. HTML is an SGML (Standard Generalized Markup Language) application, widely regarded as the standard publishing language of the World Wide Web.

5.1.1 Integrating ITS into XHTML

In XHTML 1.0, the XHTML namespace may be used with other XML namespaces as per *Namespaces in XML* [XML Names], but such documents are no longer strictly conformant[41] XHTML 1.0.

Here is an example of a document containing ITS rules which is a *non-conformant* XHTML 1.0 document.

41 → http://www.w3.org/TR/2002/REC-xhtml1-20020801/#well-formed

Example 38: ITS rules in a non-conformant XHTML 1.0 document

```
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:its="http://www.w3.org/2005/11/its" lang="en" xml:lang="en">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="keywords" content="ITS example, XHTML translation" />
  <its:rules version="1.0" xmlns:h="http://www.w3.org/1999/xhtml">
   <its:translateRule selector="//h:meta[@name='keywords']/@content"
    translate="yes" />
   <its:termRule selector="//h:span[@class='term']" term="yes" />
  </its:rules>
  <title>ITS Working Group</title>
 </head>
 <body>
  <h1>Test of ITS on <span class="term">XHTML</span></h1>
  <p>Some text to translate.</p>
  <p its:translate="no">Some text not to translate.</p>
 </body>
</html>
```

There are three ways to use ITS with XHTML and keep the XHTML document conformant:

1. Use XHTML Modularization [XHTMLMod1.1]. See Section 5.1.2: Using XHTML Modularization 1.1 for the Definition of ITS on page 69 for details.

2. Use external ITS global rules, as shown in the following example. Even local information within the document that would be handled by ITS attributes can be set indirectly.

Example 39: ITS external rules for XHTML

These rules illustrate some of the ITS data categories you can associate with specific XHTML markup. The first `its:translateRule` indicates that the attribute `content` of the `meta` element should be translated if the attribute `name` is set to "keywords". The second `its:translateRule` indicates that no `p` with a `class="notrans"` should be translated. And the `its:termRule` indicates that any `span` element with `class="term"` is a term.

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0"
 xmlns:h="http://www.w3.org/1999/xhtml">
 <its:translateRule selector="//h:meta[@name='keywords']/@content"
  translate="yes" />
 <its:translateRule selector="//h:p[@class='notrans']"
  translate="no" />
 <its:termRule selector="//h:span[@class='term']" term="yes" />
</its:rules>
```

The corresponding document:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="keywords" content="ITS example, XHTML translation" />
  <title>ITS Working Group</title>
 </head>
 <body>
  <h1>Test of ITS on <span class="term">XHTML</span></h1>
```

```
      <p>Some text to translate.</p>
      <p class="notrans">Some text not to translate.</p>
    </body>
  </html>
```

3. Use NVDL. See Section 5.1.3: Using NVDL to integrate ITS into XHTML on page 74 for details.

5.1.2 Using XHTML Modularization 1.1 for the Definition of ITS

This section describes how to use XHTML Modularization 1.1 [XHTMLMod1.1] for the definition of ITS. It first defines an ITS abstract module which is then implemented in the XML Schema format. The module is meant to be integrated in existing or new schemas which rely on XHTML Modularization 1.1.

5.1.2.1 Abstract Definition of ITS Markup

The following is the abstract definition of the elements for global ITS markup, which is consistent with the XHTML Modularization framework [XHTMLMod1.1]. Further definitions of XHTML abstract modules can be found in the XHTML Modularization specification [XHTMLMod1.1].

Note that this definition does not contain the ruby element and the dir attribute, since these are already available in XHTML. Such existing markup should be associated with ITS data categories using an `its:rules` element. See Section 5.1.4: Associating existing XHTML markup with ITS on page 76.

Elements	Attributes	Minimal Content Model
rules	version (CDATA), xlink:href (URI), xlink:type ("simple")	(translateRule | locNoteRule | termRule | dirRule | rubyRule | langRule | withinTextRule)*
translateRule	Selector, translate ("yes"|"no")	EMPTY
locNoteRule	Selector, locNotePointer (CDATA), locNoteType ("alert"| "description"), locNoteRef (URI), locNoteRefPointer (CDATA)	locNote?
locNote	translate ("yes"|"no"), locNote (CDATA), locNoteType ("alert" | "description"), locNoteRef (URI), termInfoRef (URI), term ("yes" | "no"), dir ("ltr" | "rtl" | "lro" | "rlo")	(PCDATA | ruby)*
termRule	Selector, term ("yes" | "no"), termInfoRef (URI), termInfoRefPointer (CDATA), termInfoPointer (CDATA)	EMPTY
dirRule	Selector, dir ("ltr" | "rtl" | "lro" | "rlo")	EMPTY
rubyRule	Selector, rubyPointer (CDATA), rtPointer (CDATA), rpPointer (CDATA), rbcPointer (CDATA), rtcPointer (CDATA), rbspanPointer (CDATA)	rubyText
rubyText	translate ("yes"|"no"), locNote (CDATA), locNoteType ("alert"|"description"), locNoteRef (URI), term ("yes"	PCDATA

Elements	Attributes	Minimal Content Model
	| "no"), termInfoRef (CDATA), dir ("ltr" | "rtl" | "lro" | "rlo"), rbspan (CDATA)	
langRule	Selector, langPointer (CDATA)	EMPTY
withinTextRule	Selector, withinText ("yes"|"no"|"nested")	EMPTY

The following are the abstract definitions of two attribute groups: the selector attribute used within global rules, and ITS attributes to be used locally. Again these definitions make use of XHTML Modularization 1.1.

Collection	Attributes in Collection
Selector	selector (CDATA)
ITSLocal	translate ("yes"|"no"), locNote (CDATA), locNoteType ("alert"|"description"), locNoteRef (URI), termInfoRef (URI), term ("yes" | "no")

5.1.2.2ITS XML Schema Module Implementation

The following schema contains the implementation of the abstract markup module in XML Schema.

Example 40:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3.org/2005/11/its"
    xmlns:its="http://www.w3.org/2005/11/its"
    xmlns:h="http://www.w3.org/1999/xhtml" elementFormDefault="qualified"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
    <xs:import namespace="http://www.w3.org/1999/xhtml"
        schemaLocation="xhtml-schemas/xhtml-ruby-1.xsd"/>
    <xs:simpleType name="translate.type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="term.type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="locNoteType.type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="alert"/>
            <xs:enumeration value="description"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="dir.type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ltr"/>
            <xs:enumeration value="ltr"/>
            <xs:enumeration value="lro"/>
            <xs:enumeration value="rlo"/>
        </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="withinText.type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="yes"/>
        <xs:enumeration value="no"/>
        <xs:enumeration value="nested"/>
    </xs:restriction>
</xs:simpleType>
<xs:attributeGroup name="its.Selector.attlist">
    <xs:attribute name="selector" type="xs:string" use="required"/>
</xs:attributeGroup>
<xs:attributeGroup name="its.ITSLocal.attlist">
    <xs:attribute name="translate" form="qualified" use="optional"
     type="its:translate.type"/>
    <xs:attribute name="locNote" type="xs:string" form="qualified"
     use="optional"/>
    <xs:attribute name="locNoteType" form="qualified" use="optional"
     type="its:locNoteType.type"/>
    <xs:attribute name="locNoteRef" type="xs:anyURI" form="qualified"
     use="optional"/>
    <xs:attribute name="termInfoRef" type="xs:string" form="qualified"
     use="optional"/>
    <xs:attribute name="term" type="its:term.type" form="qualified"
     use="optional"/>
</xs:attributeGroup>
<xs:element name="rules" type="its:rules.type"/>
<xs:complexType name="rules.type" mixed="false">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="its:translateRule"/>
        <xs:element ref="its:locNoteRule"/>
        <xs:element ref="its:termRule"/>
        <xs:element ref="its:dirRule"/>
        <xs:element ref="its:rubyRule"/>
        <xs:element ref="its:langRule"/>
        <xs:element ref="its:withinTextRule"/>
    </xs:choice>
    <xs:attributeGroup ref="its:rules.attlist"/>
</xs:complexType>
<xs:attributeGroup name="rules.attlist">
    <xs:attribute name="version" use="required" type="xs:string"/>
    <xs:attribute ref="xlink:href" use="optional"/>
    <xs:attribute ref="xlink:type" use="optional"/>
</xs:attributeGroup>
<xs:element name="translateRule" type="its:translateRule.type"/>
<xs:complexType name="translateRule.type">
    <xs:attributeGroup ref="its:its.Selector.attlist"/>
    <xs:attribute name="translate" use="required" type="its:translate.type"/>
</xs:complexType>
<xs:element name="locNoteRule" type="its:locNoteRule.type"/>
<xs:complexType name="locNoteRule.type">
    <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:element ref="its:locNote"/>
    </xs:sequence>
    <xs:attributeGroup ref="its:its.Selector.attlist"/>
    <xs:attribute name="locNotePointer" type="xs:string" use="optional"/>
    <xs:attribute name="locNoteType" use="required" type="its:locNoteType.type"/>
    <xs:attribute name="locNoteRef" type="xs:anyURI" use="optional"/>
    <xs:attribute name="locNoteRefPointer" type="xs:string" use="optional"/>
</xs:complexType>
<xs:element name="locNote" type="its:locNote.type"/>
<xs:complexType name="locNote.type" mixed="true">
    <xs:attribute name="translate" use="optional" type="its:translate.type"/>
```

```
        <xs:attribute name="locNote" type="xs:string" use="optional"/>
        <xs:attribute name="locNoteType" use="optional" type="its:locNoteType.type"/>
        <xs:attribute name="locNoteRef" type="xs:anyURI" use="optional"/>
        <xs:attribute name="termInfoRef" type="xs:anyURI" use="optional"/>
        <xs:attribute name="term" use="optional" type="its:term.type"/>
        <xs:attribute name="dir" use="optional" type="its:dir.type"/>
    </xs:complexType>
    <xs:element name="termRule"/>
    <xs:complexType name="termRule.type">
        <xs:attributeGroup ref="its:its.Selector.attlist"/>
        <xs:attribute name="term" type="its:term.type" use="required"/>
        <xs:attribute name="termInfoRef" type="xs:anyURI" use="optional"/>
        <xs:attribute name="termInfoRefPointer" type="xs:string" use="optional"/>
        <xs:attribute name="termInfoPointer" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:element name="dirRule" type="its:dirRule.type"/>
    <xs:complexType name="dirRule.type">
        <xs:attributeGroup ref="its:its.Selector.attlist"/>
        <xs:attribute name="dir" type="its:dir.type" use="required"/>
    </xs:complexType>
    <xs:element name="rubyRule"/>
    <xs:complexType name="rubyRule.type">
        <xs:sequence>
            <xs:element ref="its:rubyText"/>
        </xs:sequence>
        <xs:attributeGroup ref="its:its.Selector.attlist"/>
        <xs:attribute name="rubyPointer" type="xs:string" use="optional"/>
        <xs:attribute name="rtPointer" type="xs:string" use="optional"/>
        <xs:attribute name="rpPointer" type="xs:string" use="optional"/>
        <xs:attribute name="rbcPointer" type="xs:string" use="optional"/>
        <xs:attribute name="rtcPointer" type="xs:string" use="optional"/>
        <xs:attribute name="rbspanPointer" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:element name="rubyText" type="its:rubyText.type"/>
    <xs:complexType name="rubyText.type" mixed="true">
        <xs:attribute name="translate" type="its:translate.type" use="optional"/>
        <xs:attribute name="locNote" type="xs:string" use="optional"/>
        <xs:attribute name="locNoteType" type="its:locNoteType.type" use="optional"/>
        <xs:attribute name="locNoteRef" type="xs:anyURI" use="optional"/>
        <xs:attribute name="term" type="its:term.type" use="optional"/>
        <xs:attribute name="termInfoRef" type="xs:string" use="optional"/>
        <xs:attribute name="dir" type="its:dir.type" use="optional"/>
        <xs:attribute name="rbspan" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:element name="langRule"/>
    <xs:complexType name="langRule.type">
        <xs:attributeGroup ref="its:its.Selector.attlist"/>
        <xs:attribute name="langPointer" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="withinTextRule"/>
    <xs:complexType name="withinTextRule.type">
        <xs:attributeGroup ref="its:its.Selector.attlist"/>
        <xs:attribute name="withinText" type="its:withinText.type"/>
    </xs:complexType>
</xs:schema>
```

The following is a driver file which can be used to evoke the schema above.

Example 41:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xhtml="http://www.w3.org/1999/xhtml"
    targetNamespace="http://www.w3.org/1999/xhtml"
    xmlns:its="http://www.w3.org/2005/11/its"
    xmlns="http://www.w3.org/1999/xhtml" blockDefault="#all">
    <xs:annotation>
        <xs:documentation> This is the XML Schema Driver for new Document Type
         XHTML Basic 1.0 + ITS
         $Id: Overview.html,v 1.10 2008/02/12 04:55:09 fsasaki Exp $
        </xs:documentation>
        <xs:documentation
         source="http://www.w3.org/TR/xml-i18n-bp/#integration-its-xhtmlmod"/>
    </xs:annotation>
    <xs:import namespace="http://www.w3.org/2005/11/its"
     schemaLocation="its-module.xsd"/>
    <xs:redefine schemaLocation="xhtml-schemas/xhtml-basic10.xsd">
        <xs:group name="HeadOpts.mix">
            <xs:choice>
                <xs:group ref="HeadOpts.mix"/>
                <xs:element ref="its:rules"/>
            </xs:choice>
        </xs:group>
        <xs:attributeGroup name="Common.attrib">
            <xs:attributeGroup ref="Common.attrib"/>
            <xs:attributeGroup ref="its:its.ITSLocal.attlist"/>
        </xs:attributeGroup>
    </xs:redefine>
</xs:schema>
```

The file below is an instance which can be validated against this schema.

Example 42:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:its="http://www.w3.org/2005/11/its">
    <head>
        <title> </title>
        <its:rules version="1.0">
            <its:locNoteRule locNoteType="alert" selector="..." locNoteRef="...">
            </its:locNoteRule>
            <its:locNoteRule locNoteType="alert" selector="...">
                <its:locNote> </its:locNote>
            </its:locNoteRule>
            <its:termRule selector="..." term="yes"/>
        </its:rules>
    </head>
    <body>
        <h3> </h3>
        <table>
            <tr>
                <td> </td>
            </tr>
        </table>
        <ul>
            <li its:locNote="..." its:translate="no"> </li>
        </ul>
    </body>
</html>
```

5.1.2.3Conformance statement

This schema conforms to Conformance Type 1 of the ITS specification.

The schema adds the following ITS element to the XHTML schema:

- `its:rules`

The schema adds the following local ITS attributes to the XHTML schema:

- `its:translate`
- `its:locNote`
- `its:locNoteType`
- `its:locNoteRef`
- `its:term`
- `its:termInfoRef`

5.1.3 Using NVDL to integrate ITS into XHTML

As you have seen in the previous section it might sometimes be quite laborious to integrate ITS into an existing vocabulary using only modularization and the customization features of particular schema language. In such situations you can use the NVDL schema language instead.

In NVDL you can create a sort of "meta-schema" which defines how to combine and provide additional rules for existing schemas. An NVDL schema can be used in the same way as schemas written in other languages, such as DTDs, RELAX NG or XML Schema. You can then use such a schema to validate your document instances or so that an XML editor can guide you while you are editing documents. The NVDL.org site[42] provides additional information about the language. You can also find there a list of applications which support the NVDL language.

Adding ITS to XHTML involves allowing the `its:rules` element inside the `head` element and allowing the ITS local attributes to appear on every existing XHTML element.

Example 43: NVDL script for XHTML with ITS

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0"
       startMode="xhtml">

  <!-- Validation starts here -->
  <mode name="xhtml">
    <!-- XHTML elements are validated against XHTML schema -->
    <namespace ns="http://www.w3.org/1999/xhtml">
      <validate schema="../xhtml-schemas/xhtml11.xsd">
        <!-- Inside head element its:rules element is allowed -->
        <context path="head" useMode="its-rules"/>
```

42 → http://nvdl.org/

```
      </validate>
    </namespace>

    <!-- ITS attributes are validated against separate schema -->
    <namespace ns="http://www.w3.org/2005/11/its" match="attributes">
      <validate schema="its-attributes-for-xhtml.rng"/>
    </namespace>
  </mode>

  <!-- Handling of ITS markup in head is different
       because its:rules should be allowed -->
  <mode name="its-rules">
    <namespace ns="http://www.w3.org/2005/11/its">
      <validate schema="its-rules.rng"/>
    </namespace>
    <namespace ns="http://www.w3.org/2005/11/its" match="attributes">
      <validate schema="its-attributes-for-xhtml.rng"/>
    </namespace>
  </mode>
</rules>
```

The NVDL script references three schemas. One for XHTML and two supplementary ones for ITS. The first supplementary schema defines local attributes which are needed for XHTML.

Example 44: Schema defining ITS local attributes suitable for XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

  <!-- Include schema with all ITS building blocks -->
  <include href="its.rng"/>

  <!-- Pull out only definitions of ITS attributes
       which are useful for XHTML -->
  <start>
    <group>
      <ref name="its-att.translate.attributes"/>
      <ref name="its-att.locNote.attributes"/>
      <ref name="its-att.term.attributes"/>
      <optional>
        <ref name="its-att.version.attributes"/>
      </optional>
    </group>
  </start>

</grammar>
```

The second supplementary schema defines the `its:rules` element.

Example 45: Schema defining ITS local attributes suitable for XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

  <!-- Include schema with all ITS building blocks -->
  <include href="its.rng"/>
```

```
    <!-- Pull out only definition of its:rules element -->
    <start>
      <ref name="its-rules"/>
    </start>

  </grammar>
```

5.1.3.1Conformance statement

This schema conforms to Conformance Type 1 of the ITS specification.

The schema adds the following ITS element to the XHTML schema:

- `its:rules`

The schema adds the following local ITS attributes to the XHTML schema:

- `its:translate`
- `its:locNote`
- `its:locNoteType`
- `its:locNoteRef`
- `its:term`
- `its:termInfoRef`

5.1.4 Associating existing XHTML markup with ITS

A number of XHTML constructs implement the same semantics as some of the ITS data categories. In addition, some of the attributes in XHTML need to be translated, which is not the default for XML documents according to the default translate settings in ITS. These attributes need to be identified as needing translation.

An external ITS rules element can summarize these relationships. Because XHTML use is widespread and covers a large amount of legacy material the rules defined here may not be optimal for everyone.

Example 46: External ITS rules for XHTML documents

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0"
 xmlns:h="http://www.w3.org/1999/xhtml">

 <!-- special content. (See note 1) -->
 <its:translateRule selector="//h:script" translate="no"/>
 <its:translateRule selector="//h:style" translate="no"/>

 <!-- Normal translatable attributes -->
 <its:translateRule selector="//h:*/@abbr" translate="yes"/>
 <its:translateRule selector="//h:*/@accesskey" translate="yes"/>
 <its:translateRule selector="//h:*/@alt" translate="yes"/>
```

```
  <its:translateRule selector="//h:*/@prompt" translate="yes"/>
  <its:translateRule selector="//h:*/@standby" translate="yes"/>
  <its:translateRule selector="//h:*/@summary" translate="yes"/>
  <its:translateRule selector="//h:*/@title" translate="yes"/>

  <!-- The input element (Important: See note 2) -->
  <its:translateRule selector="//h:input/@value" translate="yes"/>
  <its:translateRule selector="//h:input[@type='hidden']/@value" translate="no"/>

  <!-- Non-translatable element (See note 3) -->

  <its:translateRule selector="//h:del" translate="no"/>
  <its:translateRule selector="//h:del/descendant-or-self::*/@*" translate="no"/>

  <!-- Often-used translatable meta content. -->
  <its:translateRule selector="//h:meta[@name='keywords']/@content"
       translate="yes"/>
  <its:translateRule selector="//h:meta[@name='description']/@content"
       translate="yes"/>

  <!-- Possible term (Important: See note 4) -->
  <its:termRule selector="//h:dt" term="yes"/>

  <!-- Bidirectional information -->
  <its:dirRule selector="//h:*[@dir='ltr']" dir="ltr"/>
  <its:dirRule selector="//h:*[@dir='rtl']" dir="rtl"/>
  <its:dirRule selector="//h:bdo[@dir='ltr']" dir="lro"/>
  <its:dirRule selector="//h:bdo[@dir='rtl']" dir="rlo"/>

  <!-- Elements within text -->
  <its:withinTextRule withinText="yes"
   selector="//h:abbr | //h:acronym | //h:br | //h:cite | //h:code | //h:dfn
   | //h:kbd | //h:q | //h:samp | //h:span | //h:strong | //h:var | //h:b | //h:em
   | //h:big | //h:hr | //h:i | //h:small | //h:sub | //h:sup | //h:tt | //h:del
   | //h:ins | //h:bdo | //h:img | //h:a | //h:font | //h:center | //h:s | //h:strike
   | //h:u | //h:isindex" />

</its:rules>
```

Additional notes on these rules:

1. The `script` and `style` elements may contain text that needs translation, but their content needs to be parsed with, respectively, a script filter and a CSS filter. Depending on the capabilities of your translation tools you may want to leave these elements as needing translation.

2. The `value` attribute of the `input` element may or may not need translation, depending on the way the element is used. The decision as to whether the value of this attribute needs translation or not will depend on its use in a given instance. Note, however, that it can often be undesirable to translate these values, since they are commonly used by scripts as identifiers: change the value of the attribute and the script will often fail. The values of the `value` attribute are not usually seen by a user of a web page.

3. The `del` element indicates removed text and therefore, most often, its content would not be translated. Because ITS rules for elements are not inherited by attributes, and because this element may contain elements with attributes that need translation, such as `img` with an `alt` attribute, you need to: a) define this rule after defining how translation applies to

attribute values, and b) use rules such as `selector="//h:del/descendant-or-self::*/@*"` to override any possibility of translation being applied to an attribute within a `del` element or any of its descendants.

4. The `dt` element is defined by HTML as a "definition term" and can therefore be seen as a candidate to be associated with the ITS Terminology data category. However, for historical reasons, this element has been used for many other purposes. Whether or not `dt` is associated with the ITS term data category will depend on its use in a given instance.

5.2 ITS and TEI

The *Text Encoding Initiative* [TEI] is intended for literary and linguistic material, and is most often used for digital editions of existing printed material. It is also suitable, however, for general purpose writing. The P5 release of TEI consists of 23 modules which can be combined together as needed.

5.2.1 Integrating ITS into TEI

TEI is maintained as a single ODD document, and customizations of it are also written as ODD documents. ODD (*One Document Does it all*) is a literate programming language of the Text Encoding Initiative for writing XML schemas. These documents are processed using XSLT style sheets to make a tailored user-level schema in XML DTD, XML Schema or RELAX NG.

The ITS additions involve two changes to TEI:

1. Allowing `rules` to appear in the TEI metadata section (the `teiHeader`).
2. Adding the ITS local attributes to the TEI global attribute set.

Both of these can be easily achieved using standard techniques in ODD.

The body of a TEI+ITS customization consists of a `schemaSpec` which lists the modules to be included (this example includes six common ones):

Example 47: A `schemaSpec` element with modules to be included

```
<schemaSpec ident="tei-its" start="TEI">
 <moduleRef key="header"/>
 <moduleRef key="core"/>
 <moduleRef key="tei"/>
 <moduleRef key="textstructure"/>
 <moduleRef key="namesdates"/>
 <moduleRef key="msdescription"/>
 <!-- Etc. -->
</schemaSpec>
```

In addition, we load the ITS schema (in its RELAX NG XML format, the language used by TEI for expressing content models), and overload the definition of the TEI content class `model.headerPart` to include the ITS `rules`:

Example 48: Inclusion of ITS `rules` into the TEI schema

```
<moduleRef url="its.rng">
 <content xmlns:rng="http://relaxng.org/ns/structure/1.0">
 <rng:define name="model.headerPart" combine="choice">
  <rng:ref name="rules"/>
 </rng:define>
 </content>
</moduleRef>
```

The content class determines which elements are allowed as children of `teiHeader`. Lastly, we change the definition of the global attribute class `att.global` to reference the ITS local attributes (available from the ITS schema we loaded earlier):

Example 49: Addition of the ITS local attributes to the global attributes

```
<classSpec ident="att.global" type="atts" mode="change">
 <attList>
  <attRef name="span.attributes"/>
 </attList>
</classSpec>
```

When processing, this customization produces a schema which permits markup like this:

Example 50: Document which is valid against a schema TEI+ITS

```
<TEI xmlns:its="http://www.w3.org/2005/11/its" xmlns="http://www.tei-c.org/ns/1.0">
 <teiHeader>
  <fileDesc>
   <!-- details of the file -->
  </fileDesc>
  <rules xmlns="http://www.w3.org/2005/11/its" version="1.0"
   xmlns:t="http://www.tei-c.org/ns/1.0">
   <translateRule translate="no" selector="//t:body/t:p/@*"/>
   <translateRule translate="yes" selector="//t:body/t:p"/>
  </rules>
 </teiHeader>
 <text>
  <body>
   <p rend="normal">Hello <hi>world</hi>
   </p>
   <p rend="special">Goodbye</p>
   <p its:translate="no">This must not be translated</p>
  </body>
 </text>
</TEI>
```

In this example, a set of rule elements are provided in the header to provide rules, and the body of the text performs a specific override.

5.2.1.1Conformance statement

This schema conforms to Conformance Type 1 of the ITS specification.

The schema adds the following ITS element to the TEI schema:

- `its:rules`

The schema adds the following local ITS attributes to the TEI schema:

- `its:translate`
- `its:locNote`
- `its:locNoteType`
- `its:locNoteRef`
- `its:term`
- `its:termInfoRef`

5.3 ITS and XML Spec

The XML Spec format [XML Spec] is intended for W3C Working Drafts, Notes, Recommendations, and all other document types that fall under the category of technical reports. XML Spec is available in the following formats: XML DTD, XML Schema and RELAX NG.

5.3.1 Integration of ITS into XML Spec

The text below takes version 2.10 of XML Spec[43] as an example and shows how you would integrate ITS into it. This version is available in DTD[44], XML Schema[45] and RELAX NG[46] formats.

> **Note:** Within the W3C Internationalization Activity[47], a modified version of the XML Spec 2.9 DTD[48] is used for document creation. This version has been updated with ITS markup declarations.

The *integration of ITS into the XML Spec DTD* uses the files xmlspec-its.dtd[49] (the XML Spec schema) and its.dtd[50] (the ITS schema). To achieve the integration, the following modifications to the XML Spec DTD have been made:

1. External ITS definitions are integrated via the new entity `<!ENTITY % its SYSTEM "its.dtd">` and the entity call `%its;`.
2. The existing XML Spec entity `%local.common.att;` has been modified. It now includes the declarations `'%its.att.local.with-ns.attributes;` and `xmlns:its CDATA "http://www.w3.org/2005/11/its"`. The former allows the use of ITS local attributes, the latter is necessary to permit the use of the ITS namespace in the DTD.

43 → http://www.w3.org/2002/xmlspec/#spec210
44 → http://www.w3.org/2002/xmlspec/dtd/2.10/xmlspec.dtd
45 → http://www.w3.org/2002/xmlspec/xsd/2.10/xmlspec.xsd
46 → http://www.w3.org/2002/xmlspec/rng/2.10/xmlspec.rnc
47 → http://www.w3.org/International/
48 → http://www.w3.org/International/xmlspec/documentation/xmlspec-i18n-dtd.html
49 → http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/xmlspec-its.dtd
50 → http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/its.dtd

3. The XML Spec entity `%header.mdl;` contains the content model of the `header` element. The ITS `its:rules` `element` has been added as the last element to this content model. In this way, `its:rules` can be used inside an XML Spec document.

4. The ITS elements `its:ruby` and `its:span` have been added to the XML Spec entity `%p.pcd.mix;`. In this way it is possible to use them as inline elements.

The *integration of ITS into the XML Spec RELAX NG schema* uses the files xmlspec-its.rnc[51] (the XML Spec schema) and its.rnc[52] (the ITS schema). The modifications to the RELAX NG schema have the same motivations like for the DTD described above. The modifications are:

1. External ITS definitions are integrated via the statement `include "its.rnc"`.

2. The pattern `its-att.local.with-ns.attributes` is referenced from the pattern `local.common.att`.

3. The pattern `its-rules` is referenced from the pattern `header.mdl`.

4. The patterns `its-ruby` and `its-span` are referenced from the pattern `p.pcd.mix`.

The *integration of ITS into the XML Spec XML Schema schema* uses the files xmlspec-its.xsd[53] (the XML Spec schema), its.xsd[54] (the ITS schema), xml.xsd[55] (for declarations from the XML namespace[56]) and xlink.xsd[57] (for declarations from the XLink namespace[58]). The modifications to the XML Spec XML Schema schema have the same motivations like for the DTD described above. The modifications are:

1. External ITS definitions are integrated via an `<xs:import>` statement.

2. The attribute group `its-att.local.with-ns.attributes` is added to the attribute group `common.att`.

3. The element declaration `its:rules` is added to the element group `header.mdl`.

4. The element declarations `its:ruby` and `its:span` are added to the element group `p.pcd.mix`.

The following example shows an XML Spec document conforming to the XML Spec+ITS schemas. The its:translateRule element is used to indicate that elements for code, keywords and examples should not be translated. The `w3c-doctype` element is also marked as non-translatable using local ITS markup.

51 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/xmlspec-its.rnc
52 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/its.rnc
53 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/xmlspec-its.xsd
54 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/its.xsd
55 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/xml.xsd
56 ⇢ http://www.w3.org/XML/1998/namespace
57 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/xmlspec/xlink.xsd
58 ⇢ http://www.w3.org/1999/xlink

Example 51: Sample XML Spec document with ITS markup

```
<?xml version="1.0" encoding="UTF-8"?>
<spec xmlns:its="http://www.w3.org/2005/11/its">
    <header>
        <title>Best Practices for XML Internationalization</title>
        <w3c-designation>...</w3c-designation>
        <w3c-doctype its:translate="no">W3C Working Draft</w3c-doctype>
        <pubdate>
            <day>5</day>
            <month>December</month>
            <year>2007</year>
        </pubdate>
        <publoc>
            <loc href="..." >...</loc>
        </publoc>
        <latestloc>
            <loc href="...">...</loc>
        </latestloc>
        <prevlocs>
            <loc href="..."
                >...</loc>
        </prevlocs>
        <authlist>
            <author>
                <name>...</name>
                <affiliation>...</affiliation>
            </author>
        </authlist>
        <abstract>
            <p>This document provides a set of guidelines for ...</p>
        </abstract>
        <status>
            <p/>
        </status>
        <langusage>
            <language id="en">en</language>
        </langusage>
        <revisiondesc>
            <p>This is an updated version of this document.</p>
        </revisiondesc>
        <its:rules version="1.0">
            <its:translateRule selector="//code | //kw //eg" translate="no"/>
        </its:rules>
    </header>
    <body>
        <div1 id="idIntro">
            <head>Introduction</head>
            <p>This document is a complement  ...</p>
        </div1>
    </body>
    <back>
        <div1>
            <head>References</head>
        </div1>
        <inform-div1>
            <head>Acknowledgements</head>
        </inform-div1>
    </back>
</spec>
```

5.3.1.1Conformance statement

The three XML Spec schemas described above conform to Conformance Type 1 of the ITS specification.

The following ITS elements are added:

- `its:rules`
- `its:ruby`
- `its:span`

The following local ITS attributes are added:

- `its:translate`
- `its:locNote`
- `its:locNoteType`
- `its:locNoteRef`
- `its:term`
- `its:termInfoRef`

5.3.2 Associating existing XML Spec markup with ITS

A number of XML Spec constructs implement the same semantics as some of the ITS data categories. In addition, some of the XML Spec attribute values need to be translated, which is not the default for XML documents according to the ITS default settings for translatability. These attributes need to be identified as needing translation, and some elements need to be identified as not needing translation.

> **Note:** When you have the choice of using an XML Spec construct or an ITS construct to express the same semantics, make sure you use the XML Spec construct to ensure that XML Spec processing tools work properly. Use ITS local markup only if XML Spec does not provide an equivalent.

An external ITS `its:rules` element can summarize these relationships. The rules defined here are just examples and may need further tailoring for specific use.

Example 52: ITS external rules for XML Spec documents

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its"
 xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0">

 <!-- Translatable attributes -->
 <its:translateRule selector="//graphic/@alt | //table/@summary | //term
  | //termdef | //key-term" translate="yes"/>

 <!-- Non-translatable elements/attributes -->
```

```
    <its:translateRule translate="no"
     selector="//version | // w3c-designation | w3c-doctype | //publoc | //altlocs
     | //prevlocs | //latestloc | //errataloc | //preverrataloc | //translationloc
     | //authlist | //author | //name | //affiliation | //email | //langusage
     | //language | //graphic | //proto | //arg | //scrap | //prodgroup | //prod
     | //lhs | //rhs | //wfc | //vc | //constraint | //bnf | //prodrecap
     | //interface | //module | //reference | //typedef | // struct | //component
     | //union | //case | //enum | //enumerator | //sequence | //constant
     | //exception | //attribute | //method | //parameters | //param | //returns
     | //raises | //typename | //att | //attval | //bibref | //code | //el | //kw
     | //nt | //var | //xnt | //key-term"/>

    <!-- Possible terms -->
    <its:termRule selector="//label | //def" term="yes"/>

    <!-- Elements within text -->
    <its:withinTextRule withinText="yes" selector="//abbr | //att | //attval
     | //code | //eg | //emph | //ins | //kw | //loc | //note | //qterm
     | //phrase | //quote"/>
    <its:withinTextRule withinText="nested" selector="//footnote
     | //termdef | //its:ruby"/>

   </its:rules>
```

5.4 ITS and DITA

The *Darwin Information Typing Architecture* [DITA 1.0] is an XML-based architecture for authoring, producing, and delivering readable information as discrete, typed topics.

5.4.1 Integration of ITS into DITA

DITA offers some of the ITS features by default (See Section 5.4.2: Associating existing DITA markup with ITS on page 87 for more information on that). In some cases, however, you may still want to allow the use of ITS markup directly in your DITA documents. For example, the `its:locNote` attribute, or the `its:rules` element. DITA provides a way to create a domain specialization based on the `foreign` element and attribute extension points.

For example, the DITA Concept DTD can be extended as follows:

First, create two files for the ITS domain specialization. The first one `itsDomain.ent` contains the entity definitions that will be used in the extended DTD.

Example 53: Content of the `itsDomain.ent` file

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!ENTITY % its-d-foreign "its"           >
   <!ENTITY   its-d-att     "(topic its-d)" >
```

The second file, `itsDomain.mod`, contains the definition of the element where the ITS markup will be placed.

Example 54: Content of the `itsDomain.mod` file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- declaration for the specialized wrapper and alternate element -->
<!ENTITY % its "its">

<!-- definition for the specialized wrapper and alternate element -->
<!ELEMENT its ((%its-rules;) | (%its-ruby;)) >
<!ATTLIST its %global-atts;
          class CDATA "+ topic/foreign its-d/its ">
```

Then you can adapt the `concept.dtd` file to take into account the new domain.

Example 55: The DITA Concept DTD modified for ITS

Include the ITS domain entities at the end of the Domain Entity Declarations section:

```
<!ENTITY % its-d-dec SYSTEM "itsDomain.ent" >
  %its-d-dec;
```

Include the ITS document type and namespace:

```
<!ENTITY % its-def SYSTEM  "its.dtd" >
  %its-def;
<!ENTITY % its-d-namespace "xmlns:its CDATA #FIXED 'http://www.w3.org/2005/11/its'">
<!ENTITY % props-attribute-extensions  ""                           >
<!ENTITY % base-attribute-extensions   "%its-d-namespace;
  %att.version.attributes;
  %att.locNote.attributes;" >
```

Define the extension element at the end of the Domain Extension section:

```
<!ENTITY % foreign "foreign | %its-d-foreign;" >
```

Modify the list of included domains in the included-domains entity:

```
<!ENTITY included-domains
  "&ui-d-att; &hi-d-att; &pr-d-att; &sw-d-att;
  &ut-d-att; &indexing-d-att; &its-d-att;" >
```

Include the ITS domain module at the end of the Domain Element Integration section:

```
<!ENTITY % its-d-def SYSTEM "itsDomain.mod" >
  %its-d-def;
```

All these changes allow you to include a new `its` element in different parts of the DITA document and use ITS-defined constructs where DITA may be missing support, such as for ruby text. This also allows you to use a selection of ITS-defined attributes to complement what DITA already provides.

Example 56: DITA document with ITS

This DITA document includes the following ITS constructs:

- An `its:rules` element is added to the `prolog` element to specify that, in the scope of this document, the content of `uicontrol` elements is not to be translated.
- The second `p` element includes a `its:locNote` attribute that applies to its content.
- The last paragraph includes an `its:ruby` element.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
 "dtd/itsConcept.dtd">
<concept id="DITAwithITS" xml:lang="en"
 xmlns:its="http://www.w3.org/2005/11/its" its:version="1.0">
 <title>Using ITS in DITA</title>
 <prolog>
  <its>
   <its:rules>
    <its:translateRule selector="ui" translate="no"/>
   </its:rules>
  </its>
 </prolog>
 <conbody xml:lang="ja">
  <p>Example of applying global rules: Normal text with
   <uicontrol>non-translatable text</uicontrol>.</p>
  <p its:locNote="Localization note">Text where the note applies.</p>
  <p>Example of ruby text:
   <ph xml:lang="ja">この本は <its><its:ruby>
    <its:rb>慶応義塾大学</its:rb>
    <its:rp>(</its:rp>
    <its:rt>けいおうぎじゅくだいがく</its:rt>
    <its:rp>)</its:rp>
   </its:ruby></its>/の歴史を説明するものです。</ph>
  </p>
 </conbody>
</concept>
```

5.4.1.1Conformance statement

This schema conforms to Conformance Type 1 of the ITS specification.

The schema adds the following ITS element to the DITA DTD:

- `its:rules`
- `its:ruby`

The schema adds the following local ITS attributes to the DITA DTD:

- `its:locNote`
- `its:locNoteRef`
- `its:locNoteType`
- `its:version`

5.4.2 Associating existing DITA markup with ITS

There are several ITS data categories that are already implemented in DITA. For example, DITA offers a `translate` attribute that provides the same functionality as `its:translate`.

In the same way as for other formats, these existing features can be associated with ITS data categories, so ITS-enabled tools can process seamlessly DITA source documents.

> **Note:** When you have the choice of using a DITA construct or an ITS construct to express the same thing, use the DITA construct to ensure that DITA processors work properly. Use ITS local markup only if DITA does not provide an equivalent.

Example 57: Associating DITA markup with ITS

```
<?xml version="1.0"?>
<!-- Possible default ITS rules for DITA -->
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">

 <!-- Translatable attribute (some are deprecated) -->
 <its:translateRule selector="//image/@alt" translate="yes"/>
 <its:translateRule selector="//lq/@reftitle" translate="yes"/>
 <its:translateRule selector="//note/@othertype" translate="yes"/>
 <its:translateRule selector="//object/@standby" translate="yes"/>
 <its:translateRule selector="//othermeta/@content" translate="yes"/>
 <its:translateRule selector="//state/@value" translate="yes"/>
 <its:translateRule selector="//map/@title" translate="yes"/>
 <its:translateRule selector="//topicref/@navref" translate="yes"/>
 <its:translateRule selector="//topicgroup/@navtitle" translate="yes"/>
 <its:translateRule selector="//topichead/@navtitle" translate="yes"/>
 <its:translateRule selector="//data/@label" translate="yes"/>

 <!-- Non-translatable elements -->
 <its:translateRule selector="//draft-comment//*" translate="no"/>
 <its:translateRule selector="//draft-comment/descendant-or-self::*/@*"
  translate="no"/>
 <its:translateRule selector="//required-cleanup//*" translate="no"/>
 <its:translateRule selector="//required-cleanup/descendant-or-self::*/@*"
  translate="no"/>
 <its:translateRule selector="//coords" translate="no"/>
 <its:translateRule selector="//shape" translate="no"/>

 <!-- Translatability flags -->
 <its:translateRule selector="//*[@translate='no']" translate="no"/>
 <its:translateRule selector="//*[@translate='no']/descendant-or-self::*/@*"
  translate="no"/>
 <its:translateRule selector="//*[@translate='yes']" translate="yes"/>

 <!-- Directionality flags -->
 <its:dirRule selector="//*[dir='ltr']" dir="ltr"/>
 <its:dirRule selector="//*[dir='rtl']" dir="rtl"/>
 <its:dirRule selector="//*[dir='lro']" dir="lro"/>
 <its:dirRule selector="//*[dir='rlo']" dir="rlo"/>

 <!-- Elements within text (inline) -->
 <its:withinTextRule withinText="yes"
  selector="//boolean | //cite | //itemgroup | //keyword | //ph | //q |
   //state | //term | //tm | //xref | //b | //i | //sub | //sup | //tt |
   //u | //apiname | //codeph | //delim | //fragref | //kwd | //oper |
   //option | //parmname | //repsep | //sep | //synnoteref | //synph |
   //var | //cmdname | //filepath | //msgnum | //msgph | //systemoutput |
   //userinput | //varname | //menucascade | //shortcut | //uicontrol |
   //wintitle | //coords | //shape" />

 <!-- The keyword elements within keywords are sub-flow, no in-line -->
 <its:withinTextRule withinText="nested" selector="//keywords/keyword" />

 <!-- Elements within text (subflow) -->
 <its:withinTextRule withinText="nested"
  selector="//draft-comments | //required-cleanup | //alt | //fn |
  //indexterm" />

 <!-- Terminology -->
 <its:termRule selector="//term | //dt | //termindex" term="yes" />

</its:rules>
```

The declarations above cover different versions of DITA.

5.5 ITS and GladeXML

Glade [Glade] is a user interface builder system for GTK+ and Gnome. It uses XML files (GladeXML) to store the user-interface components. The library has been ported to different platforms and offers bindings in different programming languages.

Example 58: Example of a GladeXML document

```
<?xml version="1.0" standalone="no"?><!--*- mode: xml -*-->
<glade-interface>
 <widget class="GtkWindow" id="main_window">
  <property name="visible">True</property>
  <property name="title" translatable="yes">Glade Text Editor</property>
  <property name="type">GTK_WINDOW_TOPLEVEL</property>
  <property name="window_position">GTK_WIN_POS_NONE</property>
  <property name="modal">False</property>
  <property name="default_width">600</property>
  <property name="default_height">450</property>
  <property name="resizable">True</property>
  <property name="destroy_with_parent">False</property>
  <property name="decorated">True</property>
  <property name="skip_taskbar_hint">False</property>
  <property name="skip_pager_hint">False</property>
  <property name="type_hint">GDK_WINDOW_TYPE_HINT_NORMAL</property>
  <property name="gravity">GDK_GRAVITY_NORTH_WEST</property>
  <property name="focus_on_map">True</property>
  <property name="urgency_hint">False</property>
  <signal name="delete_event" handler="on_main_window_delete_event"/>
  <child>
   <widget class="GtkVBox" id="vbox1">
    <property name="visible">True</property>
    <property name="homogeneous">False</property>
    <property name="spacing">0</property>
    <child>
     <widget class="GtkHandleBox" id="handlebox2">
      <property name="visible">True</property>
      <property name="shadow_type">GTK_SHADOW_OUT</property>
      <property name="handle_position">GTK_POS_LEFT</property>
      <property name="snap_edge">GTK_POS_TOP</property>
      <child>
```

```
          <widget class="GtkMenuBar" id="menubar1">
           <property name="visible">True</property>
           <property name="pack_direction">GTK_PACK_DIRECTION_LTR</property>
           <property name="child_pack_direction">GTK_PACK_DIRECTION_LTR</property>
           <child>
            <widget class="GtkMenuItem" id="File">
             <property name="visible">True</property>
             <property name="label" translatable="yes">_File</property>
             <property name="use_underline">True</property>
             <child>
              <widget class="GtkMenu" id="File_menu">
               <child>
                <widget class="GtkImageMenuItem" id="Open">
                 <property name="visible">True</property>
                 <property name="label">gtk-open</property>
                 <property name="use_stock">True</property>
                 <signal name="activate" handler="on_Open_activate"/>
                </widget>
               </child>
               <child>
                <widget class="GtkImageMenuItem" id="Exit">
                 <property name="visible">True</property>
                 <property name="label">gtk-quit</property>
                 <property name="use_stock">True</property>
                 <signal name="activate" handler="on_Exit_activate"/>
                </widget>
               </child>
              </widget>
             </child>
            </widget>
           </child>
          </widget>
         </child>
        </widget>
        <packing>
         <property name="padding">0</property>
         <property name="expand">False</property>
         <property name="fill">True</property>
        </packing>
       </child>
      </widget>
     </child>
    </widget>
   </glade-interface>
```

5.5.1 Integration of ITS into GladeXML

The content of the GladeXML files are mostly composed of data that should not be translated: user-interface widgets properties. Text content is limited to titles, labels and a few various other types of strings.

GladeXML does offer support for some of the ITS features, but not all of them. While it would be technically feasible to allow the use of additional ITS markup directly in your GladeXML resources, there is little point doing it here because these resources are closely tied to the Glade's editors and compilers which would have to be modified as well.

5.5.2 Associating Existing GladeXML Markup with ITS

GladeXML offers a `translatable` attribute that provides the same functionality as `its:translate`. The `comments` attribute can also be associated with localization notes.

Like for other formats, existing features of GladeXML can be associated with ITS data categories using global rules, so ITS-enabled tools can seamlessly process GladeXML source documents.

Example 59: Associating GladeXML markup with ITS

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its" version="1.0">
 <!-- ITS rules for Glade 2.0, based on http://glade.gnome.org/glade-2.0.dtd -->
 <its:translateRule selector="/glade-interface" translate="no"/>
 <its:translateRule selector="//*[@translatable='yes']" translate="yes"/>
 <its:translateRule selector="//atkaction/@description" translate="yes"/>
 <its:locNoteRule selector="//*[@translatable='yes']"
  locNoteType="description" locNotePointer="@comments"/>
</its:rules>
```

5.6 ITS and DocBook

DocBook is a general purpose XML schema particularly well suited to books and papers about computer hardware and software (though it is by no means limited to these applications). DocBook is maintained by the DocBook Technical Committee[59] of OASIS[60].

5.6.1 Integration of ITS into DocBook

DocBook V5.0 schema is maintained as a very modular and easy to customize schema written in RELAX NG [RELAX NG 1.0]. General techniques for schema customization are described in [DocBook V5.0 HOWTO].

The ITS additions involve the following changes to the DocBook schema:

1. Adding the ITS local attributes to every existing DocBook element.

   Not all ITS local attributes are added to the schema, as DocBook already provides its own means for specifying directionality of text. Such existing markup should be associated with ITS data categories using `its:rules` element. See Section 5.6.2: Associating existing DocBook markup with ITS on page 93.

2. Allowing the `its:rules` element inside DocBook `info` element which is a general metadata container.

3. Allowing the `its:ruby` as inline element almost everywhere where plain text could be.

Example 60: DocBook schema customization

```
# This schema integrates ITS markup (http://www.w3.org/TR/its/)
# into DocBook schema (http://docbook.org)
#
```

59 ⇢ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook
60 ⇢ http://www.oasis-open.org/

```
# This schema conforms to Conformance Type 1 defined in
# http://www.w3.org/TR/its/#conformance-product-schema
#
# Schema adds the following ITS elements into DocBook schema:
#   * rules
#   * ruby
#
# Schema adds the following local ITS attributes into DocBook schema:
#   * translate
#   * locNote
#   * locNoteType
#   * locNoteRef
#   * term
#   * termInfoRef

# Namespace declarations for DocBook, ITS and HTML
# (HTML is used internally in DocBook schema)
namespace db = "http://docbook.org/ns/docbook"
namespace its = "http://www.w3.org/2005/11/its"
namespace html = "http://www.w3.org/1999/xhtml"

# Include base DocBook schema
include "docbook.rnc"
{
   # Exclude ITS markup from "wildcard" element
   db._any =
      element * - (db:* | html:* | its:*) {
         (attribute * { text }
          | text
          | db._any)*
      }
}

# Include base ITS schema
include "its.rnc"

# Define pattern for local ITS attributes
db.its.attributes =
   its-att.translate.attributes?
   & its-att.locNote.attributes?
   & its-att.term.attributes?
   & its-att.version.attributes?

# Add local ITS attributes to all DocBook elements
db.common.base.attributes &= db.its.attributes

# Allow its:rules inside info element
db.info.extension |= its-rules

# Allow Ruby markup almost everywhere
db.ubiq.inlines |= its-ruby
```

For your convenience there is also available a “flattened” schema stored inside one file.

- dbits.rnc[61] (RELAX NG compact syntax schema in one file)
- dbits.rng[62] (RELAX NG schema in one file)

61 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/docbook/dbits.rnc
62 ⇢ http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/docbook/dbits.rng

There is no need to add the `its:span` element as DocBook provides similar element called `phrase` which can be used for attaching ITS local attributes to an arbitrary piece of text.

The following example shows a sample DocBook article conforming to the DocBook+ITS schema. The `its:translateRule` element is used to indicate that function names (marked up using the `function` element) should not be translated. The first paragraph is also marked as not to be translated using local ITS markup.

Example 61: Sample DocBook document with ITS markup

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook"
         xmlns:its="http://www.w3.org/2005/11/its"
         xmlns:db="http://docbook.org/ns/docbook"
         version="5.0" xml:lang="en">
  <info>
    <title>Sample article</title>
    <its:rules version="1.0">
      <its:translateRule translate="no" selector="//db:function"/>
    </its:rules>
  </info>
  <para its:translate="no">Non-translatable content</para>
  <section>
    <title>Sample section</title>
    <para>You can delete file using <function>unlink()</function>
     function.</para>
  </section>
</article>
```

5.6.1.1Conformance statement

This schema conforms to Conformance Type 1 of the ITS specification.

The schema adds the following ITS elements to the DocBook schema:

- `its:rules`
- `its:ruby`

The schema adds the following local ITS attributes to the DocBook schema:

- `its:translate`
- `its:locNote`
- `its:locNoteType`
- `its:locNoteRef`
- `its:term`
- `its:termInfoRef`

5.6.2 Associating existing DocBook markup with ITS

A number of DocBook constructs implement the same semantics as some of the ITS data categories. In addition, some of the DocBook attributes have values which should be translated, which is not the default for XML documents according to the ITS default settings. These attributes need to be identified as needing translation.

> **Note:** When you have the choice of using a DocBook construct or an ITS construct to express the same thing, make sure you use the DocBook construct to ensure DocBook processing tools work properly. Use ITS local markup only if DocBook does not provide an equivalent.

An external ITS `its:rules` element can summarize these relationships. Because DocBook use is widespread and diverse, the rules defined here are just examples which may need further tailoring for specific use.

Example 62: ITS external rules for DocBook documents

```
<its:rules xmlns:its="http://www.w3.org/2005/11/its"
    xmlns:db="http://docbook.org/ns/docbook"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    version="1.0">

 <!-- Translatable attributes -->
 <its:translateRule selector="//db:table/@summary" translate="yes"/>
 <its:translateRule selector="//db:*/@xlink:title" translate="yes"/>
 <its:translateRule selector="//db:*/@xreflabel" translate="yes"/>
 <its:translateRule selector="//db:*/@label" translate="yes"/>

 <!-- Non-translatable elements/attributes -->
 <its:translateRule translate="no" selector="//db:*[@revisionflag = 'deleted']"/>
 <its:translateRule translate="no" selector="//db:*[@revisionflag = 'deleted']//@*"/>
 <its:translateRule translate="no"
      selector="//db:acronym
         | //db:author
         | //db:classname
         | //db:command
         | //db:constant
         | //db:date
         | //db:editor
         | //db:email
         | //db:envar
         | //db:errorcode
         | //db:exceptionname
         | //db:filename
         | //db:function
         | //db:initializer
         | //db:interfacename
         | //db:markup
         | //db:methodname
         | //db:modifier
         | //db:ooclass
         | //db:ooexception
         | //db:oointerface
         | //db:option
         | //db:parameter
         | //db:person
         | //db:personname
         | //db:productnumber
         | //db:property
         | //db:returnvalue
         | //db:symbol
         | //db:tag
         | //db:type
         | //db:uri
         | //db:varname"/>

 <!-- Possible terms -->
 <its:termRule selector="//db:glossterm" term="yes"/>
 <its:termRule selector="//db:firstterm" term="yes"/>
 <its:termRule selector="//db:abbrev"    term="yes"/>
 <its:termRule selector="//db:acronym"   term="yes"/>

 <!-- Bidirectional information -->
 <its:dirRule selector="//db:*[@dir='ltr']" dir="ltr"/>
 <its:dirRule selector="//db:*[@dir='rtl']" dir="rtl"/>
 <its:dirRule selector="//db:*[@dir='lro']" dir="lro"/>
 <its:dirRule selector="//db:*[@dir='rlo']" dir="rlo"/>

 <!-- Elements within text -->
 <its:withinTextRule withinText="yes"
       selector="//db:abbrev
          | //db:accel
          | //db:acronym
          | //db:application
          | //db:author
          | //db:citation
          | //db:citebiblioid
          | //db:citerefentry
          | //db:citetitle
          | //db:classname
          | //db:code
          | //db:command
          | //db:computeroutput
          | //db:constant
          | //db:database
          | //db:date
          | //db:editor
          | //db:email
          | //db:emphasis
          | //db:envar
          | //db:errorcode
          | //db:errorname
          | //db:errortext
          | //db:errortype
          | //db:exceptionname
          | //db:filename
          | //db:foreignphrase
          | //db:function
          | //db:guibutton
          | //db:guiicon
          | //db:guilabel
          | //db:guimenu
          | //db:guimenuitem
          | //db:guisubmenu
          | //db:hardware
          | //db:initializer
          | //db:interfacename
          | //db:jobtitle
          | //db:keycap
          | //db:keycode
          | //db:keycombo
          | //db:keysym
          | //db:link
          | //db:literal
          | //db:markup
          | //db:menuchoice
          | //db:methodname
          | //db:modifier
          | //db:mousebutton
          | //db:olink
          | //db:ooclass
          | //db:ooexception
          | //db:oointerface
          | //db:option
          | //db:optional
          | //db:org
          | //db:orgname
          | //db:package
          | //db:parameter
          | //db:person
          | //db:personname
          | //db:phrase
          | //db:productname
          | //db:productnumber
          | //db:prompt
          | //db:property
          | //db:quote
          | //db:replaceable
          | //db:returnvalue
          | //db:shortcut
          | //db:subscript
          | //db:superscript
          | //db:symbol
          | //db:systemitem
          | //db:tag
          | //db:token
          | //db:trademark
          | //db:type
          | //db:uri
          | //db:userinput
          | //db:varname
          | //db:wordasword"/>

 <its:withinTextRule withinText="nested"
       selector="//db:alt
          | //db:footnote
          | //db:remark
          | //db:indexterm
          | //db:primary
          | //db:secondary
          | //db:tertiary"/>

</its:rules>
```

A References (Non-Normative)

BCP 47
: Addison Phillips and Mark Davis, editors. *Tags for Identifying Languages*. Best Current Practice 47 September 2006. Available at http://www.rfc-editor.org/rfc/bcp/bcp47.txt

Bidi in X/HTML
: Richard Ishida, editor. *Internationalization Best Practices: Handling Right-to-left Scripts in XHTML and HTML Content*. W3C Working Draft 6 June 2007. Available at http://www.w3.org/TR/2007/WD-i18n-html-tech-bidi-20070606/. The latest version of Bidi in X/HTML is available at http://www.w3.org/TR/i18n-html-tech-bidi/ .

CharMod
: Martin J. Dürst, François Yergeau, Richard Ishida, Misha Wolf, Tex Texin, editors. *Character Model for the World Wide Web 1.0: Fundamentals*. W3C Recommendation 15 February 2005. Available at http://www.w3.org/TR/2005/REC-charmod-20050215/. The latest version of CharMod is available at http://www.w3.org/TR/charmod/ .

DITA 1.0
: Michael Priestley, JoAnn Hackos, et. al., editors. *OASIS Darwin Information Typing Architecture (DITA) Language Specification v1.0*. OASIS Standard 9 May 2005. Available at http://www.oasis-open.org/committees/download.php/15316/dita10.zip.

DocBook V5.0 HOWTO
: Jirka Kosek, Norman Walsh and Dick Hamilton. *DocBook V5.0: The Transition Guide*. Available at http://docbook.org/docs/howto/.

Glade
: *Glade - a User Interface Builder for GTK+ and GNOME*. The GNOME Foundation. Available at http://glade.gnome.org/.

ITS
: Christian Lieske and Felix Sasaki, editors. *Internationalization Tag Set (ITS) Version 1.0*. W3C Recommendation 3 April 2007. Available at http://www.w3.org/TR/2007/REC-its-20070403/. The latest version of ITS is available at http://www.w3.org/TR/its/ .

ITS REQ
: Yves Savourel, editor. *Internationalization and Localization Markup Requirements*. W3C Working Draft 18 May 2006. Available at http://www.w3.org/TR/2006/WD-itsreq-20060518/. The latest version of ITS REQ is available at http://www.w3.org/TR/itsreq/ .

NVDL
: *Information technology -- Document Schema Definition Languages (DSDL) -- Part 4: Namespace-based Validation Dispatching Language (NVDL).* International Organization for Standardization (ISO) ISO/IEC 19757-4:2003.

RELAX NG 1.0
: *Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG.*, International Organization for Standardization (ISO) ISO/IEC 19757-2:2003.

Ruby Annotation
: Marcin Sawicki, Michel Suignard, Masayasu Ishikawa, et al. editors. *Ruby Annotation*. W3C Recommendation 31 May 2001 Available at http://www.w3.org/TR/2001/REC-ruby-20010531/. The latest version of Ruby Annotation is available at http://www.w3.org/TR/ruby/ .

TEI
: Lou Burnard and Syd Bauman, editors. *Text Encoding Initiative Guidelines development version (P5)*. TEI Consortium, Charlottesville, Virginia, USA, Text Encoding Initiative.

Unicode in XML
: Martin Dürst and Asmus Freytag. *Unicode in XML and other Markup Languages*. W3C Working Group Note 16 May 2007, Unicode Technical report #20. The latest version of Unicode in XML is available at http://www.w3.org/TR/unicode-xml/ .

XHTML 1.0
: Steven Pemperton et al., editors. *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C Recommendation 26 January 2000, revised 1 August 2002. Available at http://www.w3.org/TR/2002/REC-xhtml1-20020801/. The latest version of XHTML 1.0 is available at http://www.w3.org/TR/xhtml1/ .

XHTMLMod1.1
: Daniel Austin et al., editors. *XHTML™ Modularization 1.1*. W3C Working Draft 5 July 2006. Available at http://www.w3.org/TR/2006/WD-xhtml-modularization-20060705/. The latest version of XHTMLMod1.1 is available at http://www.w3.org/TR/xhtml-modularization/ .

XInclude 1.0
: Jonathan Marsh, David Orchard and Daniel Veillard, editors. - *XML Inclusions (XInclude) Version 1.0 (Second Edition)*. W3C Recommendation 15 November 2006. Available at http://www.w3.org/TR/2006/REC-xinclude-20061115/. The latest version of XInclude 1.0 is available at http://www.w3.org/TR/xinclude/ .

XLIFF 1.2
: Yves Savourel et al., editors. *XLIFF Version 1.2*. OASIS Committee Specification 24 July 2007. Available at http://docs.oasis-open.org/xliff/v1.2/cs02/xliff-core.html. The latest version of *XLIFF 1.2* is available at http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html .

XLink 1.0
: Steve DeRose, Eve Maler and David Orchard, editors. - *XML Linking Language 1.0*. W3C Recommendation 27 June 2001. Available at http://www.w3.org/TR/2001/REC-xlink-20010627/. The latest version of XLink 1.0 is available at http://www.w3.org/TR/xlink/ .

xml:id
: Jonathan Marsh, Daniel Veillard and Norman Walsh, editors. - *xml:id Version 1.0*. W3C Recommendation 9 September 2005. Available at http://www.w3.org/TR/2005/REC-xml-id-20050909/. The latest version of xml:id is available at http://www.w3.org/TR/xml-id/ .

XML Names
: Tim Bray, Dave Hollander and Andrew Layman, editors. *Namespaces in XML*. W3C Recommendation 14 January 1999. Available at http://www.w3.org/TR/1999/REC-xml-names-19990114/. The latest version of XML Names is available at http://www.w3.org/TR/REC-xml-names/ .

XML Spec
: *The XML Spec Schema and Stylesheets*. Available at http://www.w3.org/2002/xmlspec/ .

B Acknowledgements (Non-Normative)

This document has been developed with important contributions from past and present member of the Working Group: Bartosz Bogacki (W3C Invited Expert), Martin Dürst (W3C Invited Expert), Tim Foster, Richard Ishida (W3C/ERCIM), Masaki Itagaki, Jirka Kosek (W3C Invited Expert), Christian Lieske (SAP AG), Sebastian Rahtz (W3C Invited Expert), Felix Sasaki (W3C/Keio), Yves Savourel (ENLASO Corporation), Diane Stoick (The Boeing Company), Najib Tounsi (Ecole Mohammadia d'Ingenieurs Rabat (EMI)), and Andrzej Zydron.

At the date of publication, the members of the Working Group were: Bartosz Bogacki (W3C Invited Expert), Damien Donlon (Sun Microsystems, Inc.), Martin Dürst (W3C Invited Expert), Poonam Gupta (Centre for Development of Advanced Computing (CDAC)), Richard Ishida (W3C/ERCIM), Jirka Kosek (W3C Invited Expert), Christian Lieske (SAP AG), Sebastian Rahtz (W3C Invited Expert), Francois Richard (HP), Goutam Saha (Centre for Development of Advanced Computing (CDAC)), Felix Sasaki (W3C/Keio), Yves Savourel (ENLASO Corporation), Diane Stoick (The Boeing Company), and Najib Tounsi (Ecole Mohammadia d'Ingenieurs Rabat (EMI)).