# OGC Testbed-14

## *MapML Engineering Report*

# Table of Contents

Publication Date: 2019-03-06

Approval Date: 2018-12-13

Submission Date: 2018-12-09

Reference number of this document: OGC 18-023r1

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D012

Category: Public Engineering Report

Editor: Joan Masó

Title: OGC Testbed-14: MapML Engineering Report

---

**OGC Engineering Report**

**COPYRIGHT**



**WARNING**

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

**LICENSE AGREEMENT**

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

# Chapter 1. Summary

This is the second Engineering Report (ER) about the Map Markup Language (MapML) [1] resulting from OGC Testbed initiatives. To find an introduction of MapML and how it works, please, refer to the previous ER OGC 17-019 [2]. MapML is a new media type that can be included in a <layer> element of a <map> section, in a Hypertext Markup Language (HTML) page. This document is mainly focused on the description of the MapML media type and its evolutions. In particular, it considers issues about the Coordinate Reference System (CRS) types in MapML, feature and properties encoding, Cascading Style Sheets (CSS) symbolization, multidimensional data etc.

This document describes two implementations done in OGC Testbed-14: a Cloud-based Proxy (cascade) for MapML done by CubeWerx and a ServiceWorker Proxy for MapML done by George Mason University (GMU).

Finally, this document reviews how the next generation of OGC services can integrate MapML files as part of the designing of use cases and discusses how MapML can be used by social media.

This document proposals increases functionality in MapML and makes proposals for increasing the interoperability of the proposed encoding with the OGC standards baseline and future generations of OGC standards for maps and tiles.

## 1.1. Requirements & Research Motivation

This ER is required to capture all of the results of the MapML task in OGC Testbed-14. In particular, it is required to collect use cases, implementations, best practices, experiences and results.

This ER makes particular emphasis on a MapML Extension Study that is the result of extensive discussion in the group.

This document is required to specify and suggest adaptations or integration of OGC Web Services (OWS) (e.g. Web Map Service (WMS), Web Feature Service (WFS), Web Map Tile Service (WMTS)) serving MapML.

From the set of topics suggested at the beginning of the testbed, the participants have addressed the following:

a. Tiled Coordinate Reference System (TCRS) definitions
b. Image georeferencing markup
c. TCRS/projection negotiation
d. Language negotiation
e. Hyperlinking within and between map services
f. Microdata / microformat semantic markup recommendations
g. Caching discussion
h. Feature styling with Cascading Style Sheets
i. non-spatial content, such as legends, notes, graphics and other standard map meta-information

j. Evaluation of support for vector tiles in the client HTML <map> element

k. Evaluation of support for ‘offline’ MapML maps

l. Definition of layer grouping / styling and animation in MapML

m. Investigation and implementation of semantic markup integration, parsing and display using HTML/Microdata + schema.org + OGC Simple Features.

The following topics initially suggested at the beginning of the Testbed have not been addressed and might be included in future testbeds:

a. Security considerations

b. Accessibility considerations for map feature markup

c. Extent processing discussion / algorithm

d. Feature creation / update / deletion sections via PUT, POST, DELETE considerations

This ER also collects experiences through the Browser Extension implementation of the HTML <map> element Application Programming Interface (API) and event model, to make programming maps on the Web an exercise in the application of browser-supported standards. This ER also collects experiences through the implementation of a Cloud-based Proxy for MapML.

## 1.2. Prior-After Comparison

During OGC Testbed-14, some suggested changes got into a new version of the MapML specification (https://maps4html.github.io/MapML/spec/) including clarifications on the coordinate systems (that completely aligns it to the WMTS concepts), the inclusion of tile support for the World Geodetic System 1984 (WGS84), the use of *selects* as an extra parameter in *extent* (such as elevation, time or band names), the use of alternative styles and the possibility to use external features-based linking to files (or WFS request URLs) instead of having to embed the features directly in the main MapML.

## 1.3. Recommendations for Future Work

The support for tiles and images in the specification is now consolidated, but more work needs to be done in implementing the vector part. In particular, we need to clarify the attribute encoding, the CRS used and the eventual relation with vector tiles.

Another recommendation is to explore how MapML should be integrated in social media as a media type (formerly referred to as a MIME type). In social media there is neither editable HTML nor a <map> section but just the inclusion of a MapML file in a message. This message, could say something about a position ("I’m here" or "please go to this restaurant") but also discuss about more general topics ("this is the impact of Climate Change in global temperature"). The suggested approach of having extra parameters in a MapML URL could be considered to make it possible to recycle and exchange MapML files, while pointing to a restricted area of time in them. For example, the extent of a MapML file could be changed with Bounding Box (BBOX) parameters after the name of the MapML file, as explained in the corresponding section.

The new MapML specifications define how to use WMS and WMTS services in URL templates to dynamically serve tiles or images linked to the MapML file. More work needs to be done on the

integration of OGC services as a source for recovering MapML files. In contrast, WMS and WMTS are potential standards for this but none of them are currently able to create MapML files. The former is not aware of TCRSs (tile matrix sets in the OGC vocabulary) and it is not in a good position to understand the zoom level concept. The latter is not able to return a MapML document that includes more than one tile because the purpose of GetTile is the generation of only one tile. There have been some proposals in previous testbeds to have a WMTS GetTiles operation and some drafts developed in the WMS group. Unfortunately, there has not yet been a demonstration of the real performance of such operations in practice, so the proposals have thus far not been accepted as a WMTS extension.

## 1.4. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

**Contacts**

| Name | Organization |
|---|---|
| Joan Maso | CREAF |
| Keith Pomakis | Cubewerx |
| Gil Heo | George Mason University |

## 1.5. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

# Chapter 2. References

The following normative documents are referenced in this document.

*NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards. All other references are listed in the bibliography.*

- OGC: OGC 06-121r9, OGC® Web Services Common Standard, 2010 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- IETF: RFC 7946, The GeoJSON Format, 2016 [https://tools.ietf.org/html/rfc7946]

# Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- coordinate reference system

  ```
  coordinate system that is related to the real world by a datum term name [ISO
  19111]
  ```

- coordinate system

  ```
  set of mathematical rules for specifying how coordinates are to be assigned to
  points [ISO 19111]
  ```

- map

  ```
  portrayal of geographic information as a digital image file suitable for display on
  a computer screen
  ```

- portrayal

  ```
  presentation of information to humans [ISO 19117]
  ```

## 3.1. Abbreviated terms

*NOTE: The abbreviated terms clause gives a list of the abbreviated terms and the symbols necessary for understanding this document. All symbols should be listed in alphabetical order. Some more frequently used abbreviated terms are provided below as examples.*

- CORS Cross-Origin Resource Sharing
- JPEG Joint Photographic Experts Group
- JSON JavaScript Object Notation
- PNG Portable Network Graphics

# Chapter 4. Overview

MapML is defined in an incremental way, by consensus, in the community. This ER records the regular discussions of the authors of this ER with the rest of the MapML team, including the authors of the MapML specification. Some of the content recorded here describes proposals of the authors that most of the time were not accepted by the group and, as a consequence, never became part of the MapML specification. In case of discrepancy, the MapML specification would be considered the authoritative source.

**NOTE**

*Examples*

Some examples of MapML documents can be found here: http://geogratis.gc.ca/mapml/en/

MapML considers several coordinate reference systems all useful at the same time. Section 5 introduces them and suggests some alignment with the WMTS standard and the Tile Matrix Set draft specification.

Section 6 discusses the introduction of URL templates in the MapML specification and how the introduction of the URL templates has changed the interaction between clients and services.

Section 7 continues the discussions about the encodings for features.

Section 8 presents several discussions about the use of CSS for styling features.

Section 9 suggests the introduction of event handling in Maps4html.

Section 10 presents some discussions and topics that could be extended in further interoperability experiments.

Section 11 records the lessons learned and experiences coming from the components implementing MapML experimented or developed in this testbed.

Section 12 discusses a possible way forward, beyond HTML, to share maps in social media.

Section 13 discusses a web service that combines WMS and WMTS and provides MapML documents as an output.

## 4.1. MapML in relation to other encoding

MapML is unique relative to other encodings but there is some level of duplication. What makes MapML unique is that it takes *Spatial Data on the Web Best Practices* [3] and applies them directly to HTML users instead of favoring the web developers.

The requirements for MapML were originally published on the Web here: http://maps4html.github.io/HTML-Map-Element-UseCases-Requirements. MapML is an extension to HTML, which if implemented, implies that the browser understands map/layer semantics (however those elements are eventually named), as well as feature, property, or geometry semantics. MapML is intended to be user-oriented. It aims at enabling users to create Web pages in all manner of styles, but having the common denominator being HTML and CSS, as well as JavaScript for

progressive enhancement. Today, there is no built-in map/layer behavior in web browsers, much less feature/property/geometry semantics in such browsers. MapML provides the ability to encode map/layer and feature/property/geometry semantics in one (simple) format that will be read and interpreted by web browsers directly. MapML brings geographic knowledge to the web browser, thereby making the web browser the user agent.

There is no other encoding or service that attempts to do what MapML does. GeoJSON, Geography Markup Language (GML), or KML require JavaScript interpretation and are not native on the web browsers. MapML should not be constrained by other encodings should the need for user-oriented features arise (e.g. markup in coordinate strings and possibly other requirements). A major objective of MapML is to make the browser understand not only where the user is (that is already standardized by the GeoLocation API, https://www.w3schools.com/html/html5_geolocation.asp), but also to understand where features are in relation to the user.

## 4.2. MapML as a media type

A MapML document is a new media type for maps on the web. It looks very similar to and it is inspired by HTML, but it is not HTML. Even if it could look like HTML, it is actually written under the rules of MicroXML.

This document uses text/mapml as the media type for MapML documents.

| | |
|---|---|
| **WARNING** | This type has not been negotiated with Internet Assigned Numbers Authority (IANA) yet. |

A MapML document can be included in an HTML document in two ways:

- by a link (src attribute) in the <layer> element in the <map> division:

```
<map zoom="15" lat="45.398043" lon="-75.70683" width="640" height="300" projection=
"CBMTILE">
  <layer label="Canada Base Map" src="http://example.com/mapml/cbmt/" checked></layer>
  <layer label="CanVec+ 031G" src="http://example2.com/mapml/canvec/50k/features/"
></layer>
</map>
```

- embedded directly in the HTML page, in the same way a Scalable Vector Graphics (SVG) document can also be included.

```
<html>
  <body>
    <map projection="CBMTILE" zoom="17" lat="45.4624905" lon="-74.9787676" width="900"
height="400" controls>
      <layer label="CanVec+ Features" checked>
        <extent units="WGS84" method="GET">
          <input ...>
          <...>
          <link rel="features" ...>
        </extent>
      </layer->
    </map>
  </body>
</html>
```

| NOTE | At the time of writing this document, the second option was not part of the Maps4HTML draft specification but the need was discussed in OGC Testbed-14 with the authors of the MapML specification and might be incorporated in the near future. |
|---|---|

There is a fundamental difference between the first and the second approach. If the MapML document is linked to the HTML page, the MapML becomes opaque to the HTML page. In practice this means that the internal content of a MapML document is not part of the Document Object Model (DOM) structure of the HTML and it will not be accessible for scripting (e.g. it will not be accessible to the JavaScript code associated with the HTML page). When the MapML encoding is embedded in the HTML page, it is part of the HTML DOM and it can be read and altered by scripts.

The fact that linked MapML documents are opaque to the scripts suggests the need to have events associated with MapML linked documents, in order to notify scripts about some changes in the map produced by user interactions. This will be discussed later in this document.

# Chapter 5. CRSs in MapML

This chapter discusses about CRS definitions and CRS negotiation in MapML.

## 5.1. Introduction

The MapML specification defines 6 possible CRS types as possible values for "input@units". The following possible values are defined in the version of the MapML specification that was available for OGC Testbed-14 (September 7, 2018):

*Table 1. Original table defining the input@units possible values.*

| Value | Definition |
|---|---|
| tcrs | The location should be serialized in units associated with the TCRS instance, i.e. pixels or in the case of WGS84, decimal degrees |
| pcrs | The location should be serialized in units associated with the projected coordinate system associated to the TCRS instance, e.g. meters for OSMTILE, which has an associated projected coordinate reference system of EPSG:3857. |
| gcrs | The location should be serialized in units associated with the geodetic coordinate system associated to the TCRS instance, e.g. decimal degrees for the OSMTILE TCRS, which has an underlying geodetic coordinate reference system of EPSG:4326. In the case of a TCRS such as WGS84, the gcrs and tcrs location are the same. |
| map | The location should be serialized in units associated with the map coordinate system defined by the extent, i.e. in pixels with the origin at the upper left corner of the extent, with coordinate axes' increasing right and downwards, respectively. |
| tilematrix | The location should be serialized in units associated with the tilematrix at the zoom level of the extent. |
| tile | The location should be serialized in units associated with a tile at the zoom level of the extent. |

MapML is dependent on the concepts of raster tiles and two-dimensional (2D) map projections. Projected CRS are commonly expressed in 'meters' with a few exceptions (such as the equirectangular *plate carréer*). Traditionally we refer to these coordinates as (x, y). Projected spaces are expressed in floating point numbers and can be as precise as the technology is to store coordinates. Many of them are defined to cover large portions of the Earth.

To define raster tiles, we superimpose cells (pixels) to a region of the projected coordinates. The region will start at the so called top-left corner and will be covered by tiles to the right and downwards. To do that, we need to define 3 pairs of regular index axes (for each zoom level) that are parallel to the projected coordinates and quantizes the space in pieces. The first one divides the space in cells (often referred to as pixels). Cell space starts at the top-left corner of the tiled space, and cells (pixels) are counted with integer numbers that increase to the right and downwards. The OGC draft specification for Tile Matrix Sets refers to these coordinates as (i',j'). The second index axis appears when the cells are grouped into tiles. The tiles are numbered from the top-left corner of the tiled space and count (with integer numbers) increases to the right and downwards. WMTS

refers to these coordinates as (TileCol,TileRow). There is also an internal double index inside each tile to point to a cell in a given tile. In WMTS these coordinates are referred to as (i,j).

## 5.2. CRS types in MapML

The authors of this ER proposed the following changes and clarifications in the "input@units" table to align the concepts (not necessarily the names of the concepts) with WMTS. To do that, there was agreement on introducing a new TCRS in the MapML table that would define an equirectangular *plate carréer*. This TCRS has *Base CRS / Projection system*=CRS:84, *Origin (easting,northing)*=-180,90, *Tile row/column size*=265, *Projected Bounds / LatLng Bounds*=LatLng(-180,-90), LatLng(180,90) and has the following zoom level values:

*Table 2. Proposed a new Equirectangular TCRS zoom levels.*

| **Resolution** |
|---|
| 0.703125000000000 |
| 0.351562500000000 |
| 0.175781250000000 |
| 8.78906250000000 10^-2 |
| 4.39453125000000 10^-2 |
| 2.19726562500000 10^-2 |
| 1.09863281250000 10^-2 |
| 5.49316406250000 10^-3 |
| 2.74658203125000 10^-3 |
| 1.37329101562500 10^-3 |
| 6.86645507812500 10^-4 |
| 3.43322753906250 10^-4 |
| 1.71661376953125 10^-4 |
| 8.58306884765625 10^-5 |
| 4.29153442382812 10^-5 |
| 2.14576721191406 10^-5 |
| 1.07288360595703 10^-5 |
| 5.36441802978516 10^-6 |

Then, this ER proposes to change the table of values for input@units in the following direction:

*Table 3. Proposed new table defining the input@units possible values.*

| Value | Definition | Representation |
|---|---|---|
| tcrs | For a given zoom level (that defines a pixel size), the location is expressed in pixel coordinates with the origin at the top-left corner of the tiled space. Not applicable when extent/@units=WGS84 | OGC Tile Matrix Set suggested naming is (i',j'). |
| pcrs | The location is expressed in projected coordinates. Units are meters except for the geodetic CRS of WGS84 that is expressed in longitude-latitude. e.g. meters for OSMTILE, which has an associated projected CRS of EPSG:3857 | Commonly they are represented as (x,y) |
| gcrs | The location is expressed in the geodetic coordinate system associated to the projection, e.g. decimal degrees for the OSMTILE TCRS, which has an underlying geodetic coordinate reference system of EPSG:4326. In the case of extent/@units=WGS84, the gcrs and pcrs location are the same. | Commonly they are represented as (long,lat) |
| map | For a given zoom level and a given extent, the location is expressed in pixel coordinates with the origin starting at the upper-left corner of the extent increasing right and downwards, respectively | They are equivalent to the (i,j) values of the OGC WMS standard. |
| tilematrix | For a given zoom level, the location should be expressed in number of tiles starting to count tiles at the top-left corner of the tiled space increasing right and downwards. Not applicable when extent/@units=WGS84. | In OGC WMTS standard they are names (TileCol, TileRow). |
| tile | For a given zoom level and a given tile, the location is expressed in pixel coordinates with the origin starting at the upper left corner of the tile increasing right and downwards, respectively and ending at 256. The combination of tilematrix and tile coordinates give a location with the precision of a pixel size. Not applicable when extent/@units=WGS84. | They are equivalent to the (i,j) values of the OGC WMTS standard. |

**WARNING** At the time of publishing this document, a new version of the MapML specification has been released (October 17, 2018) adopting most of the changes suggested in this chapter.

## 5.3. CRS negociation

An approach for documenting alternative CRS has been proposed, during OGC Testbed-14, consisting of the combined use of "meta" and similar capabilities. If for some reason a TCRS different for the current one is needed, the map browser can simply replace the current MapML document by an *alternate* one that has the same information in the desired projection.

*Alternative CRSs in MapML.*

```
<meta name="projection" content="OSMTILE"/>
<link rel="alternate" projection="CBMTILE" href=
"https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.inwater
a_1m/cubewerx?projection=CBMTILE"/>
```

**NOTE**

The reason behind the combined use of *meta* and *link* elements derives from the fact that the web browser needs to know the actual projection to be able to decide if the projection for the current MapML (written in the *meta* field; as well as in the extent) is compatible with the other layers already in the view or it needs to try an alternative map projection. Currently, the list of alternatives for projection values is limited by the MapML specification. These differ from the *styles* case where there is no list of *a-priori* alternative name values or even the guarantee that two styles with the same name are actually compatible.

Apart from documenting the current and alternative projections in the MapML document, it would be good to define a parameter in the headers of the GET request to be able to specify the current accepted or supported projections by the web browser in the same way that accept-language works today (see https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html).

*Alternative CRSs in MapML.*

```
Accept-Projection: OSMTILE, WGS84;q=0.5
```

which means: "I prefer OSMTILE, but might accept WGS84".

# Chapter 6. URL templates

The introduction of URL templates in the MapML specification was one of the suggestions of OGC Testbed-13 that was adopted. This chapter explores how the URL templates have been extended in OGC Testbed-14 and how the introduction of the URL templates has provided a new level of interaction between clients and services exchanging MapML documents.

## 6.1. Evolution of the interaction between clients and services exchanging MapML.

This subsection includes some features in MapML that were included in the candidate standard during this Testbed.

### 6.1.1. Features can be linked as well as embedded

In the previous versions of MapML, features could only be embedded in a MapML document as MicroXML encoded (with an XML encoding inspired in the original GeoJSON format). Now it is possible to include a link to external features files that may or may not be encoded in MapML.

### 6.1.2. Static MapML

The OGC Testbed-13 MapML ER (OGC 17-019) presents a description of a dynamic interaction between a client and server that required the exchange of a MapML document for each action the user does in the map (e.g. a pan or a zoom). That was due to, at that time, a MapML document containing references (URLs) to all of the individual tiles that were needed to populate the viewport without making use of possible patterns that tile URLs might follow. An action of the user (e.g. a pan) resulted in a new bounding box and most of the previous tiles were no longer useful so that new tile URLs needed to be obtained resulting in a new MapML exchange. The Figure 1 shows the sequence of events of one of these client-server interactions. This OGC Testbed-14 ER refers to this approach as *dynamic MapML*. In this approach a server is needed to generate MapML documents on-the-fly. The following example illustrates the response of the server to one of these interactions. In it, can be seen several similar tile URLs, one for each tile necessary to cover the viewport.

*Dynamic MapML example for describing a tile service*

```
<?xml version="1.0" encoding="UTF-8"?>
<mapml>
  <head>
    <title>Canada Base Map - Transportation (CBMT)</title>
    <base href="/mapml/en/osmtile/cbmt/"/>
    <link href="https://www.nrcan.gc.ca/earth-sciences/geography/topographic-
information/free-data-geogratis/licence/17285" rel="license" title="Canada Base Map ©
Natural Resources Canada"/>
  </head>
  <body>
    <extent action="/mapml/en/osmtile/cbmt/" enctype="application/x-www-form-
urlencoded" method="get" units="OSMTILE">
      <input max="2048" min="0" name="xmin" type="xmin" value="104"/>
      <input max="2048" min="0" name="ymin" type="ymin" value="389"/>
      <input max="2048" min="0" name="xmax" type="xmax" value="1034"/>
      <input max="2048" min="0" name="ymax" type="ymax" value="789"/>
      <input max="15" min="0" name="zoom" type="zoom" value="3"/>
      <input name="projection" type="projection" value="OSMTILE"/>
    </extent>
    <tile  src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/2/1?m4h=t"/>
    <tile  src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/2/2?m4h=t"/>
    <tile  src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/1/1?m4h=t"/>
...
    <tile  src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_TXT_3857/MapServer/t
ile/3/1/4?m4h=t"/>
    <tile  src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_TXT_3857/MapServer/t
ile/3/3/4?m4h=t"/>
  </body>
</mapml>
```

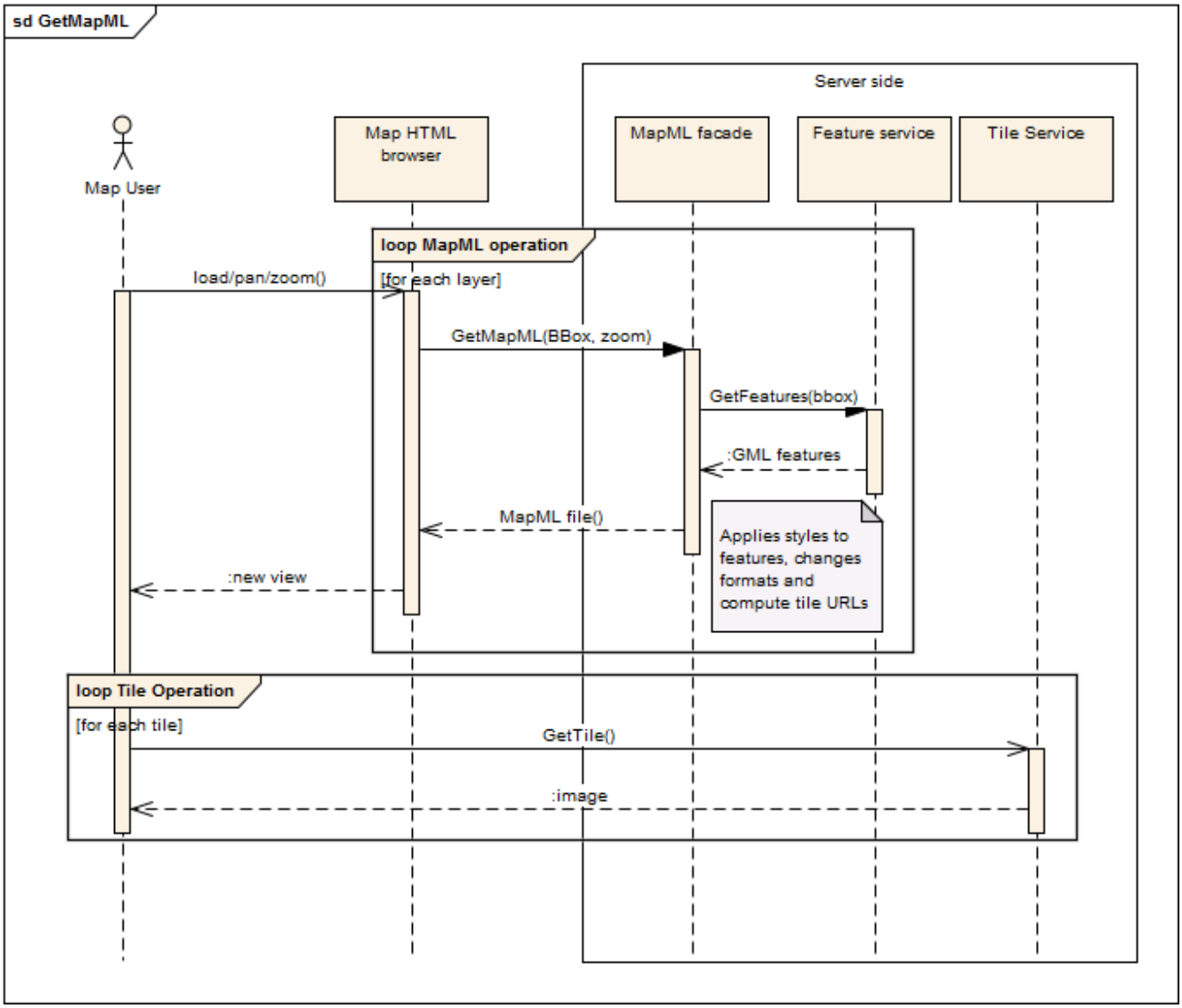


*Figure 1. Interaction between client and server in a dynamic MapML*

In OGC Testbed-14, participants experimented with the use of URI templates to simplify the iteration. In this case, a MapML document no longer offers full URLs pointing to individual resources. Instead, it includes a generic URI template and the client should be able to figure out the necessary URLs to request tiles, images of features. Since the URL template is valid for any zoom and pan, this approach removes the need for having a server responding to each and every user basic interaction (e.g. zoom and pan) and generation MapML documents on the fly. In the extreme, it removes the need for MapML generation on the server-side because MapML documents can be statically produced for the resources and stored in a web server as files. In this document, the approach is referred to as static MapML. The following example illustrates a MapML document that describes how to access a tile service. In it, can be seen a single URI template for tiles; the client should be able to derive from the URI template all the necessary tiles to cover the viewport. In addition to the URL template, the MapML document includes the definition of each variable in the URI template as an <input> of the <extent> section. In the case of the zoom level it also includes the minimum and maximum value this variable can adopt. Note that this approach allows for dialoging with different services using different variable naming conventions. For example, in OSM it is common to express the URL template using the *z*, *y* and *x* variables while WMTS uses *TileMatrix*, *TileRow* and *TileCol* respectively. Figure 3 shows the inspection of the network interaction of the web browser with the tile server, demonstrating the conversion of the URL template into 10 different tile requests in a 2x5 matrix (including the request for 2 non-existing tiles that results in

an error).

*Static MapML example for describing a tile service (from: https://geogratis.gc.ca/mapml/en/cbmtile/cbmt/?alt=xml simplified)*

```
<mapml>
  <head>
    <title>Canada Base Map - Transportation (CBMT)</title>
    <base href="/mapml/en/cbmtile/cbmt/"/>
    <link rel="license" href="https://www.nrcan.gc.ca/earth-
sciences/geography/topographic-information/free-data-geogratis/licence/17285" title=
"Canada Base Map © Natural Resources Canada"/>
  </head>
  <body>
    <extent units="CBMTILE" method="GET">
      <input name="xmn" type="xmin"  min="768" max="1024"/>
      <input name="ymn" type="ymin"  min="768" max="1280"/>
      <input name="xmx" type="xmax"  min="768" max="1024"/>
      <input name="ymx" type="ymax"  min="768" max="1280"/>
      <input name="z" type="zoom"  value="0" min="0" max="17"/>
      <input name="projection" type="projection" value="CBMTILE"/>
      <input name="y" type="location" units="tilematrix" axis="row"/>
      <input name="x" type="location" units="tilematrix" axis="column"/>
      <template type="tile" tref=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT3978/MapServer/tile/{
z}/{y}/{x}?m4h=t" />
    </extent>
  </body>
</mapml>
```

Please note that individual tiles are defined outside the <extent> section while the tile template has been moved inside the <extent> section.

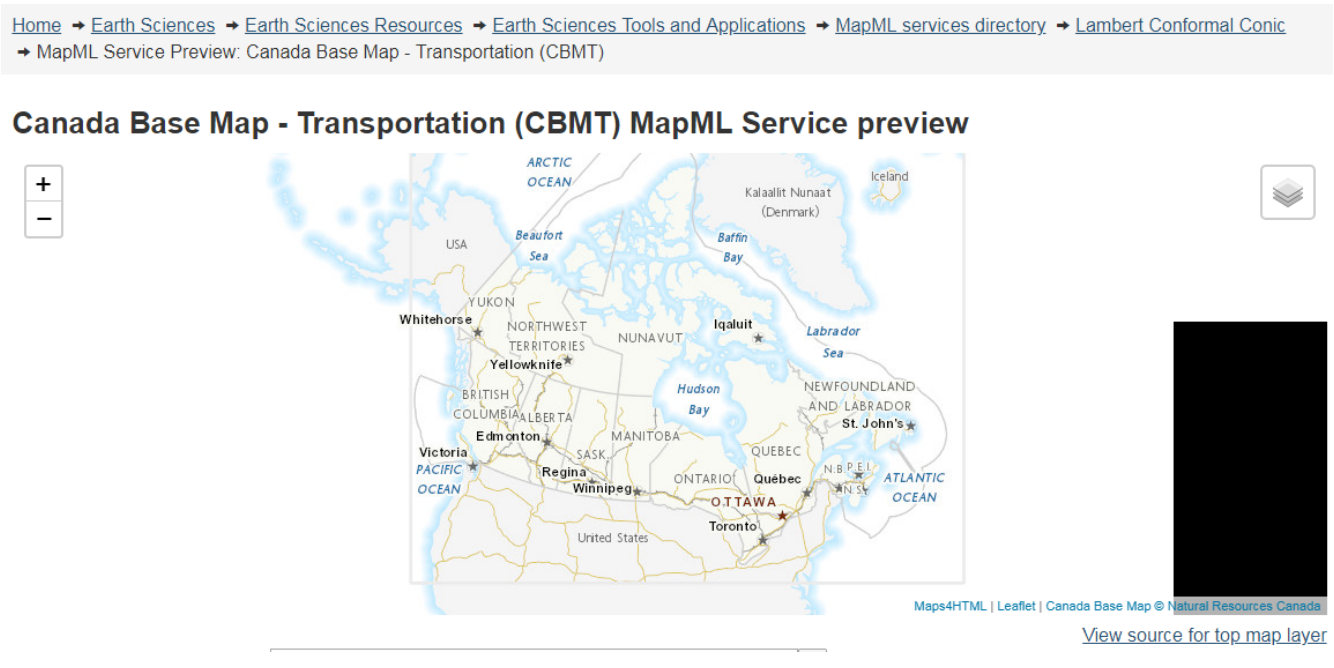


*Figure 2. Visualization of the static MapML Transportation map, usign tile URI templates.*

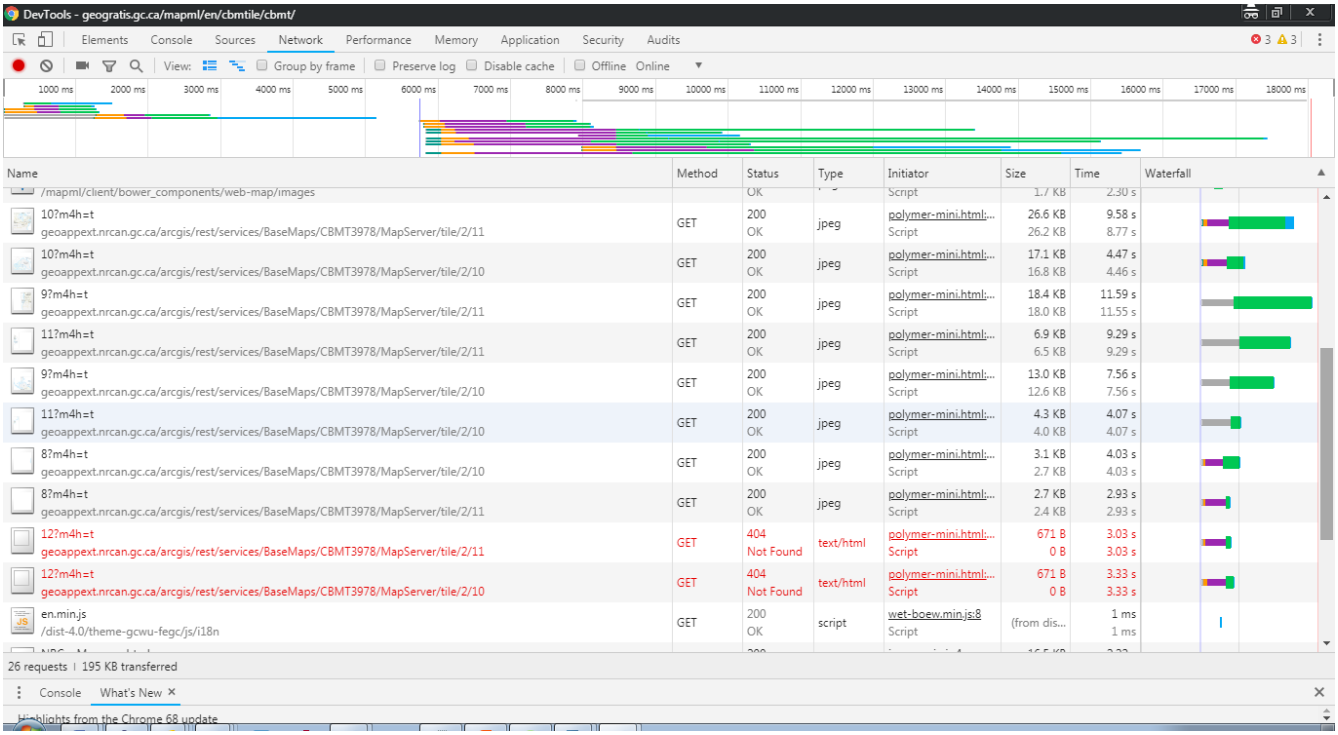


*Figure 3. Visualization of the static MapML Transportation map, using tile URI templates.*

Most tile services are commonly based on the definition of URL templates favoring the transition to static MapML documents. Other services (e.g. WMS, WFS...) do not explicitly expose URL templates for accessing its resources. For that reason, it was not so obvious that the same approach could be extended to other kinds of information services.

The first case that is examined is the case of a map server that provides an image that covers exactly the size of the viewport. In this case, an OGC WMS instance is used.

*Static MapML example for describing a OGC Web Map Service (from: http://geogratis.gc.ca/api/beta/mapml/en/osmtile/toporama?alt=xml simplified)*

```
<mapml>
  <head>
    <title>Toporama</title>
    <base href="/api/beta/mapml/en/osmtile/toporama/"/>
  </head>
  <body>
    <extent units="OSMTILE" >
      <input name="z" type="zoom"  value="18" min="4" max="18"/>
      <input name="w" type="width"/>
      <input name="h" type="height"/>
      <input name="xmin" type="location" units="pcrs" position="top-left" axis=
"easting" min="-2.003750834E7"  max="2.003750834E7" />
      <input name="ymin" type="location" units="pcrs" position="bottom-left" axis=
"northing" min="-2.003750834E7"  max="2.003750834E7" />
      <input name="xmax" type="location" units="pcrs" position="top-right" axis=
"easting" min="-2.003750834E7"  max="2.003750834E7" />
      <input name="ymax" type="location" units="pcrs" position="top-left" axis=
"northing" min="-2.003750834E7"  max="2.003750834E7" />
      <link rel="image" tref="http://wms.ess-
ws.nrcan.gc.ca/wms/toporama_en?SERVICE=WMS&amp;REQUEST=GetMap&amp;FORMAT=image/jpeg&am
p;TRANSPARENT=FALSE&amp;STYLES=&amp;VERSION=1.3.0&amp;LAYERS=WMS-Toporama
&amp;WIDTH={w}&amp;HEIGHT={h}&amp;CRS=EPSG:3857&amp;BBOX={xmin},{ymin},{xmax},{ymax}&a
mp;m4h=t" />
    </extent>
  </body>
</mapml>
```

Even if the WMS standard never mentions URI templates, the standardized Key-Value Pairs (KVP) notation can easily be expressed as a URI template. When a WMS URI template is used to include an image in MapML, all variables in the URI template should be stated as input parameters. Width and height are special cases because MapML does not imposes a map width or height; actually, they are only present if the MapML document has been included in a <map> section of a <html> as reference in a <layer> element.

OGC Testbed-14 also introduced the capability to link to external sources of feature data. If this capacity is combined with the possibility to express interactions with WFS as URI templates, the result is an ability to get the features covering the bounding box of the map for any user action. In the following example, the feature service does not adopt any particular OGC standard, but it expresses requests that will result in feature collections. These requests can be summarized in a URI template.

*Static MapML example for describing a feature service (from: geogratis.gc.ca/api/beta/mapml/static/features.html modified)*

```
<mapml>
  <body>
    <extent units="WGS84" method="GET">
      <input name="z" type="zoom" min="0" max="18"/>
      <input name="xmin" type="location" units="gcrs" axis="longitude" position="top-
left" min="-76" max="-74"/>
      <input name="ymin" type="location" units="gcrs" axis="latitude" position=
"bottom-right" min="45" max="46"/>
      <input name="xmax" type="location" units="gcrs" axis="longitude" position=
"bottom-right" min="-76" max="-74"/>
      <input name="ymax" type="location" units="gcrs" axis="latitude" position="top-
left" min="45" max="46"/>
      <link rel="features" tref=
"https://geogratis.gc.ca/api/beta/vectors/canvec/50k/features/?zoom={z}&amp;xmin={xmin
}&amp;ymin={ymin}&amp;xmax={xmax}&amp;ymax={ymax}&amp;projection=WGS84&amp;entry-
type=full&amp;max-results=100"/>
    </extent>
  </body>
</mapml>
```

If the web page is loaded into a web browser and the developer tools activated, the inspection of the page reveals that a request is done automatically to this URL: https://geogratis.gc.ca/api/beta/vectors/canvec/50k/features/?start-index=964807&max-results=100&entry-type=full&xmin=-75.0198075989449&ymin=45.446105109662014&xmax=-74.93772892700747&ymax=45.47886441546374&projection=WGS84&zoom=17, and the response is in a fully flagged MapML format that contains the needed features. In practice, if it contains other formats such as GeoJSON the other formats may also be supported. Support for other formats will make possible a connection with the new WFS 3.0 web services that the OGC is drafting.

**Connection with the next generation of OGC web services**

The OGC has recently been working on the next generation of WFS services (WFS 3.0) that has been completely remastered. Other OGC web services are being considered for a similar approach, that is based on OpenAPI. The adoption of an approach consistent with the current draft of WFS 3.0 can favor integration with the static style of MapML. The reason is that both (MapML and OpenAPI) have adopted the same URI template standard to define what is to be retrieved. WFS 3.0 lists resources that the service can offer in a OpenAPI document. Resources can be retrieved individually or as part of a collection. In the OpenAPI document, for each resource type a URI template is specified as well as all variables that the URI template contains, indicating the data type and the restrictions imposed on the values (including limits, enumerations, etc)

There are some differences: OpenAPI separates the variables that may appear before or after the '?' in the URI. The variable before the '?' are assumed to have order and are part of the URL template definition. Variables that appear after the '?' are assumed to be in KVP notation, do not have any order so they do not appear in the URL template (but are enumerated as variables needed by the resource path). To understand this better, consider how the WFS 3.0 OpenAPI defines the way a

feature collection should be retrieved.

*WFS 3.0 OpenAPI document fragment*

```
'/collections/{collectionId}/items':
  get:
    summary: 'retrieve features of feature collection {collectionId}'
    description: >-
      Every feature in a dataset belongs to a collection. A dataset may
      consist of multiple feature collections. A feature collection is often a
      collection of features of a similar type, based on a common schema.\

      Use content negotiation to request HTML or GeoJSON.
    operationId: getFeatures
    tags:
      - Features
    parameters:
    - $ref: '#/components/parameters/collectionId'
    - $ref: '#/components/parameters/bbox'
components:
  parameters:
    collectionId:
      name: collectionId
      in: path
      required: true
      schema:
        type: string
    bbox:
      name: bbox
      in: query
      required: false
      schema:
        type: array
        minItems: 4
        maxItems: 6
        items:
          type: number
      style: form
      explode: false
```

Even if the authors of this document consider mentioning the difference, they do not perceive this as a problem. MapML can document a more restrictive URI template containing the query variables provided that the proposed template is compatible with the requirements of the WFS 3.0 (provided that the KVP syntax for the query variables is followed).

Interestingly, WFS 3.0 uses GeoJSON as a default encoding and WGS84 for the coordinates, which is compatible with the way MapML is defined today.

### Implications for OWS Context

In OGC Testbed-13 there was a discussion on the similarities to other standards and in particular to OWS Context. The evolution of MapML into a more static client-server interaction moved MapML a step closer to OWS Context documents. OWS Context provides a way to create a document describing the viewport on an integrated client. OWS Context documents can encode the presence of one or more geospatial services that will be opened by a user interface when the document is loaded. There are still significant differences. In OWS Context, a sample of a server GET request is included instead of a URI template. There is an assumption that the client knows about the logic of the service requests and there is no need to include a URI template or any reference to variables on the URI template pattern. When writing a context document, developers can take the logical decision of at least providing a URL sample that that covers the viewport, but this is not mandatory in OWS Context. In other words, OWS Context assumes that the client is fully aware of the mechanics of the GET URLs used in OGC services and will be able to read the capabilities document, extract all the information it needs about the service and its layers and formulate GET requests when it needs it. In contrast, MapML provides the client with enough information to formulate GET requests for sending to the server by using URI templates and input variables, and does not require knowledge of the geospatial standard used in the request.

# Chapter 7. Feature Encoding

This section discusses approaches for encoding feature information in MapML. MapML features do not consider any form of vector tiles yet. Instead, features are provided as points, lines, polygons (and aggregations of them), the geometry of which is encoded as coordinates in one of the defined coordinate systems.

## 7.1. Current version encoding

The current proposed encoding is a direct translation of the GeoJSON Encoding into a MicroXML. The following code is an example of how a MultiPolygon looks like in the current MapML encoding.

*Code Example of a MapML containing embedded feature information*

```
<feature>
   <properties>
     <h1>Alaska</h1>
   </properties>
   <geometry>
     <multipolygon>
       <polygon>
         <coordinates>-131.602021 55.117982 -131.569159 55.28229 -131.355558 55.183705
-131.38842 55.01392 -131.645836 55.035827 -131.602021 55.117982</coordinates>
       </polygon>
       <polygon>
         <coordinates>-131.832052 55.42469 -131.645836 55.304197 -131.749898 55.128935
-131.832052 55.189182 -131.832052 55.42469</coordinates>
       </polygon>
       <polygon>
         <coordinates>-132.976733 56.437924 -132.735747 56.459832 -132.631685
56.421493 -132.664547 56.273616 -132.878148 56.240754 -133.069841 56.333862
-132.976733 56.437924</coordinates>
       </polygon>
       <polygon>...</polygon>
    </multipolygon>
  </geometry>
</feature>
```

## 7.2. Encoding geometries

This subsection discusses the encoding of geometrical characteristics of a feature.

### 7.2.1. How different is GeoJSON in MicroXML from GML?

This section demonstrates that the geometrical part of GeoJSON in MicroXML is not so different from a conveniently simplified GML encoding. It is true that GML encoding may appear intimidating but the main complexity of GML originates in the need for preparing an application schema that consequently inherits the complexities introduced by XML validation. If an application

is only interested in the geometrical classes and is thus limited to a concrete version of GML (as they are predefined in the GML standard, e.g. v3.2) the need for an application schema disappears, and most of the complications dissipate! In addition, the GML subsections used in MapML can be totally included in MapML avoiding the need to check a long GML document.

Given the self-imposed restrictions in the GeoJSON specification, the authors of this ER are still not convinced of the advantages of GeoJSON in MicroXML and the need to specify a new encoding for XML, instead of using what GML has defined for more than a decade. The following subsections compare both encodings.

| | |
|---|---|
| **NOTE** | Despite the arguments exposed on this subsection, the authors of the MapML specification still prefer the use of GeoJSON in MicroXML encoding because they believe that it could be better accepted by the Web community. |

#### Points

Surprisingly, the GML notation for a point is even more compact than the GeoJSON equivalent.

*Point example in GeoJSON in MicroXML*

```
<Point>
    <coordinates>123 321</coordinates>
</Point>
```

*The same point example in schema-less GML*

```
<Point>
    <pos>123 321</pos>
</Point>
```

#### Lines

The GML notation for a line looks surprisingly similar to the GeoJSON equivalent.

*Line example in GeoJSON in MicroXML*

```
<LineString>
    <coordinates>123 321 122 322</coordinates>
</LineString>
```

*The same line example in schema-less GML*

```
<LineString>
   <posList>123 321 122 322</pos>
</LineString>
```

### Polygons

Things start to get a bit more complicated for polygons in GML but the encoding is much more readable and makes a clearer distinction between external and internal rings.

*Polygon example in GeoJSON in MicroXML*

```
<Polygon>
      <coordinates>103 421 104 422 106 342 103 321</coordinates>
      <coordinates>123 321 112 132 112 132 123 321</coordinates>
</Polygon>
```

In GeoJSON there is an assumption that the first ring is the exterior one, but this can be too restrictive. Polygons might need more than one exterior ring.

*The same polygon example in schema-less GML*

```
    <Polygon>
      <exterior>
        <LinearRing>
          <posList>103 421 104 422 106 342 103 321
          </posList>
        </LinearRing>
      </exterior>
      <interior>
        <LinearRing>
          <posList>123 321 124 322 126 322 123 321
          </posList>
        </LinearRing>
      </interior>
    </Polygon>
```

### Multipoint

Multi-feature objects are a bit more verbose in GML but not much.

*Multipoint example in GeoJSON in MicroXML*

```
    <MultiPoint>
       <point>
         <coordinates>1 1</coordinates>
       </point>
       <point>
         <coordinates>2  2</coordinates>
       </point>
    </MultiPoint>
```

*The same multipoint example in schema-less GML*

```
<MultiPoint>
  <pointMember>
    <Point>
      <pos>1 1</pos>
    </Point>
  </pointMember>
  <pointMember>
    <Point>
      <pos>2 2</pos>
    </Point>
  </pointMember>
<MultiPoint>
```

| NOTE | During the discussion on this topic, a mistake in the definition of MultiPolygons was spotted and corrected by the authors of the MapML specification. |
|---|---|

### 7.2.2. Adding CRS

During the discussions in this testbed, the need for alternative CRSs was identified. On one hand, if provided in the same coordinate reference of the tiles (for example in TCRS), the web browser can render them directly. In this case the web browser only needs to apply an offset to move the origin from the top-left corner of the tiled space to the top-left corner of the viewport to have coordinates that SVG or Canvas can use. On the other hand, GCRS coordinates can be easily supported due to the existence of open source JavaScript libraries (e.g. proj4js) that are available to transform latitude/longitude coordinates (GCRS) into projected coordinates (PCRC) that can later be transformed into TCRS and then to the viewport coordinates by applying a linear (scaling and offsetting) transformation.

If the GeoJSON abstract rules are strictly followed as stated in the IETF specification, the only possible option is to use GCRS coordinates in WGS84. This is the text of the specification forcing this rule:

> The coordinate reference system for all GeoJSON coordinates is a geographic coordinate reference system, using the World Geodetic System 1984 (WGS 84) [WGS84] datum, with longitude and latitude units of decimal degrees. This is equivalent to the coordinate reference system identified by the Open Geospatial Consortium (OGC) URN urn:ogc:def:crs:OGC::CRS84. Note: the use of alternative coordinate reference systems was specified in [GJ2008], but it has been removed from this version of the specification because the use of different coordinate reference systems has proven to have interoperability issues.
>
> — GeoJSON standard

| | |
|---|---|
| **NOTE** | MapML should decide if it strictly follows the GeoJSON rules or it supports other CRS. |

Can MapML be produced directly in a coordinate system that escapes the need of performing complicate floating point operations used in projection formulas? Obvious alternatives are "tcrs" "tilematrix", "map" and "tile" coordinate systems. The "tile" coordinate system can be disregarded because it is local to a single tile and more than one tile are needed to fill the map. The "tilematrix" can also be disregarded because of insufficient accuracy. A MapML document can define an extent that indirectly defines a "map" origin. Nevertheless, the map extent might be changed by the context in which the MapML is loaded; e.g. the HTML "map" element. This leaves us with the "tcrs" coordinates that are pixel-based coordinates with a fix origin in the origin of the tiled space.

This ER suggests inclusion of an attribute in the geometrical part of each feature that indicates the type of Coordinate System with all possible values of the current input@units (even if "map", "tilematrix" and "tile" are not recommended).

*Possible MapML encoding of a feature expresed in TCRS coordinates*

```
<LineString units="tcrs">
    <coordinates>123 321 122 322</coordinates>
</LineString>
```

## 7.3. Encoding attributes in features

After examining the geometrical part of the features, this ER now concentrates on the attribute part. There is some consensus on the convenience of using HTML to express attributes ready to visualize. This has the issue of machine readability of the information contained in the HTML fragment. The suggestion is to use schema.org encodings to enable that.

### 7.3.1. Encoding feature properties with Microdata.

Three alternatives are presented below. All three rely on the use of microdata to specify the non-geometrical properties of the feature. The way they encode geometry is different.

| | |
|---|---|
| **NOTE** | Please note that in this chapter there is no suggestion to use the GeoShape elements included in schema.org. The authors of this ER consider that the geoshape element has been oversimplified. It can be useful to report the approximate position of a resource to a web crawler but it cannot substitute an accurate description of the coordinates of the features. |

### 7.3.2. Alternative 1: Compact geometry encoding with microdata

This encoding is departing from the general feature model where a feature can have geometrical and non-geometrical properties at the same level. Instead, it assumes that the content of a feature is characterized by its geometrical properties first and non-geometrical properties can be attached to it as sub-properties, in the same way that GeoJSON does. The advantage of this approach is that we can attach non-geometrical properties to the complete geometry but also to any part of it.

In this encoding, root feature elements are named <polygon>, <polyline>.... Inside these elements, HTML text is included describing the feature. Schema.org encoded in microdata is introduced to tag the HTML text with semantics. In addition, the linearRing element, that includes the coordinates of the polygon, is also introduced.

The style of the polygon applies to all linearRings but can be overwritten by them if a style is applied to a linearRing directly. It is assumed that the style does not apply to the non-geometrical properties that should define their one styles.

*Example of a polygon encoding*

```
<mapml>
  <polygon id="myrestaurant" style="fill:red;stroke:black;stroke-width:3;opacity:0.5"
itemscope itemtype="http://schema.org/Restaurant">
    <linearRing boundary="exterior">123,321 124,322 126,322 123,321</linearRing>
    <h1 itemprop="name">Fondue for Fun and Fantasy</h1>
    <p itemprop="description">Fantastic and fun for all your cheesy occasions.</p>
    <p>Open: <span itemprop="openingHours" content="Mo,Tu,We,Th,Fr,Sa,Su 11:30-23:00"
>Daily from 11:30am till 11pm</span></p>
    <p>Phone: <span itemprop="telephone" content="+155501003333">555-0100-3333
</span></p>
    <p>View <a itemprop="menu" href="http://example.com/menu">our menu</a>.</p>
  </polygon>
</mapml>
```

The following example shows how non-geometrical properties can be added to an inner ring.

*Example of a polygon with a ring of a different color and additional non-geometrical properties.*

```
<mapml>
  <polygon id="myrestaurant" style="fill:red;stroke:black;stroke-width:3;opacity:0.5"
itemscope itemtype="http://schema.org/Restaurant">
    <h1 itemprop="name">Fondue for Fun and Fantasy</h1>
    <p itemprop="description">Fantastic and fun for all your cheesy occasions.</p>
    <p>Open: <span itemprop="openingHours" content="Mo,Tu,We,Th,Fr,Sa,Su 11:30-23:00"
>Daily from 11:30am till 11pm</span></p>
    <p>Phone: <span itemprop="telephone" content="+155501003333">555-0100-3333
</span></p>
    <p>View <a itemprop="menu" href="http://example.com/menu">our menu</a>.</p>
    <linearRing boundary="exterior">123,321 124,322 126,322 123,321</linearRing>
    <linearRing boundary="interior" style="stroke:orange;stroke-width:5">123.5,321.5
124.5,322.5 125.5,321.5 123.5,321.5
      <p itemprop="description">Hole in the restaurant land.</p>
    </linearRing>
  </polygon>
</mapml>
```

### 7.3.3. Alternative 2: Compact geometry encoding with microdata

This encoding is more similar to the GeoJSON alternative and separates, in a clearer way, the geometric and non-geometric characteristics. An example of a polygon encoding is shown below.

*Example of a polygon encoding*

```
<mapml>
  <feature id="myrestaurant" itemscope itemtype="http://schema.org/Restaurant">
    <h1 itemprop="name">Fondue for Fun and Fantasy</h1>
    <p itemprop="description">Fantastic and fun for all your cheesy occasions.</p>
    <p>Open: <span itemprop="openingHours" content="Mo,Tu,We,Th,Fr,Sa,Su 11:30-23:00"
>Daily from 11:30am till 11pm</span></p>
    <p>Phone: <span itemprop="telephone" content="+155501003333">555-0100-3333
</span></p>
    <p>View <a itemprop="menu" href="http://example.com/menu">our menu</a>.</p>
    <polygon id="MyRestaurantGeometry" style="fill:red;stroke:black;stroke-width:5
;opacity:0.5" itemscope itemtype="http://pending.schema.org/GeospatialGeometry">
      <linearRing boundary="exterior">123,321 124,322 126,322 123,321</linearRing>
    </polygon>
  </feature>
</mapml>
```

### 7.3.4. Alternative 3: GML geometry encoding with microdata

The following encoding reuses GML geometrical classes to encode the geometrical properties. In this case, there is neither namespace nor GML schema but only the reuse of the GML encoding of geometrical features. The need for defining feature types *a priori* is eliminated. The resulting notation is comparable to other alternatives in terms of complexity and size.

Since GML validation has been relaxed, GML elements can be considered extensible and attributes and elements added when needed. One of the additions is the inclusion of styles to represent how the objects need to be portrait in the screen. In the following examples, the style property is used. Other approaches can also be used to associate styles to elements in HTML such as the use of "class" names or the association of styles to element id's.

| | |
|---|---|
| **NOTE** | The use of SVG (css) styles in GML is not new and was introduced in GML 3.0.0 and is still present in the informative annex H in GML 3.2.1 (OGC 07-036). Nevertheless, the encoding suggested here is different and based on how HTML links elements with css styles. |

*Example of a polygon encoding*

```
<mapml>
  <feature id="myrestaurant" style="stroke:black;stroke-width:5;opacity:0.5" itemscope
itemtype="http://schema.org/Restaurant">
    <h1 itemprop="name">Fondue for Fun and Fantasy</h1>
    <p itemprop="description">Fantastic and fun for all your cheesy occasions.</p>
    <p>Open: <span itemprop="openingHours" content="Mo,Tu,We,Th,Fr,Sa,Su 11:30-23:00"
>Daily from 11:30am till 11pm</span></p>
    <p>Phone: <span itemprop="telephone" content="+155501003333">555-0100-3333
</span></p>
    <p>View <a itemprop="menu" href="http://example.com/menu">our menu</a>.</p>
    <Polygon id="MyRestaurantGeometry" style="fill:lime;stroke:black;stroke-width:5
;opacity:0.5" itemscope itemtype="http://www.opengis.net/gml/3.2">
      <exterior>
        <LinearRing>
          <posList>123 321 124 322 126 322 123 321
          </posList>
        </LinearRing>
      </exterior>
    </Polygon>
  </feature>
</mapml>
```

In this example, the relaxation of GML validation is used to include new style attributes to an inner ring and to add non-geometrical properties to it.

*Example of a polygon encoding with holes*

```
<mapml>
  <feature id="myrestaurant" style="stroke:black;stroke-width:5;opacity:0.5" itemscope
itemtype="http://schema.org/Restaurant">
    <h1 itemprop="name">Fondue for Fun and Fantasy</h1>
    <p itemprop="description">Fantastic and fun for all your cheesy occasions.</p>
    <p>Open: <span itemprop="openingHours" content="Mo,Tu,We,Th,Fr,Sa,Su 11:30-23:00"
>Daily from 11:30am till 11pm</span></p>
    <p>Phone: <span itemprop="telephone" content="+155501003333">555-0100-3333
</span></p>
    <p>View <a itemprop="menu" href="http://example.com/menu">our menu</a>.</p>
    <Polygon id="MyRestaurantGeometry" style="fill:lime;stroke:black;stroke-width:3
;opacity:0.5" itemscope itemtype="http://www.opengis.net/gml/3.2">
      <exterior>
        <LinearRing>
          <posList>123 321 124 322 126 322 123 321
          </posList>
        </LinearRing>
      </exterior>
      <interior style="stroke:orange;stroke-width:5">
        <LinearRing>
          <posList>123.5 321.5 124.5 322.5 125.5 321.5 123.5 321.5
          </posList>
        </LinearRing>
        <span itemscope itemtype="http://schema.org/Thing">
          <p itemprop="description">Hole in the restaurant land.</p>
        </span>
      </interior>
    </Polygon>
  </feature>
</mapml>
```

If this approach is taken, extensions that are allowed should be detailed in the specification. Moreover, some addition to the set of GML geometrical objects is needed, such as the addition of ellipses.

## 7.4. MapML in relation to vector tiles

Recently, some vendors (such as Mapbox, Google, etc) have produced encodings for vector tiles. There is no consensus of a transversal or interoperable encoding yet, what currently makes vector tiles an 'unknown media type' as far as the browser is concerned. As such, they are handled by the JavaScript layer (e.g. a leaflet plug-in), not by the web browser engine, except e.g. for the canvas API calls that might be done by the JavaScript library. In this respect, vector tiles are even less portable than PNG tiles, since for the latter the web browser engine 'understands' how to layout and paint picture formats. The main advantage of vector tiles is bandwidth conservation, which is important, but it is not the first main goal in standardization of geospatial concepts in the web browser.

MapML does not overlap or duplicate the role of vector or raster tiles. However, a MapML client

engine, whether it was implemented in JavaScript or Web Assembly, or preferably by the web browser, could use vector data, (instead of, or in addition to raster tiles) to paint a map layer. To adopt vector tiles, the main barrier is the standardization of the vector tile format to the point where it is widely understood and implemented as a PNG. Finally, despite the canvas element, vector tiles styling is done by scripting, because vector tiles are not included in the DOM, hence they are not susceptible to styling via CSS.

# Chapter 8. CCS Symbolization

Cascading Style Sheets (CSS) describes how HTML elements are to be displayed on screen.

A CSS document comprises a sequence of rule-sets that consist of two parts:

- The selector, that points to the HTML element to be symbolized. This element should be part of the DOM of the HTML. This excludes styling geometrical features rendered in a canvas element.
- The declaration block that contains one or more symbol declarations

*Example of a ruleset in CSS*

```
p {
    color: red;
    text-align: center;
}
```

In the example above, the *selector points* to all <p> elements and declares that letters will be in *red* and sentences will be *centered* for all paragraphs in the HTML page. CSS rule-sets can be provided in an HTML page by:

- including in a <style> section in the HTML (or MapML) document
- including a link to an external CSS document

In addition, *declaration blocks* can be used directly in the HTML element they apply to (in-line) using the *style* attribute. This way, there is no need to use a selector.

*Example of a ruleset in CSS*

```
<p style="color: red; text-align: center;">
```

This section discusses how the CSS practices in HTML can be reused for a MapML document. CSS considerations mainly apply to features included in MapML. Maps included in MapML via links to tiles or images cannot be easily manipulated to change aspect: in fact, the common way of changing tile or image styles is to change the MapML document for another one that links to the same objects rendered in different style from the server side.

## 8.1. How to apply css styles to geometries

The more direct way to apply CCS symbolization is to do it in-line by using a style attribute in the same way that it can be done in other HTML tags.

*Example of a polygon encoding*

```
<polygon style="fill:red;stroke:black;stroke-width:5;opacity:0.5">
...
</polygon>
```

Another direct way to set styles is to use a class or and element identifier attribute to assign a CSS ruleset by its name.

```
<style>
.nice
{
  fill:red;stroke:black;stroke-width:5;opacity:0.5
}
</style>

<polygon class="nice">
...
</polygon>
```

Things start to be more interesting when we use a characteristic in CSS that allows for selecting elements to be symbolized depending on the values of some attributes of the element.

```
<style>
polygon[type="road"]
{
  fill:red;stroke:black;stroke-width:5;opacity:0.5
}
</style>

<polygon type="road">
...
</polygon>
```

CSS has selectors that can be used to select element tag names, class attributes or elements id's. An interesting capability is that selectors can be used to select whatever element name that has an attribute (using the '*' character).

```
<style>
*[type="road"]{
    background:red;
}
</style>
<feature>
  <properties>
    <span type="road">road</span>
  </properties>
</feature>
```

As suggested by https://www.brmwebdev.com/dev/css/schema-based-styling, this can be used for semantic tagging in schema.org microdata to define the symbolization of elements that has a particular semantic annotation. This means it can be used to select features that have a particular property. In practice, if features are tagged with microdata, then elements of a particular itemtype

can be selected. Going further in this approach, itemtype's that are in a particular scope can be selected. The authors of this ER have not been able to find anyone suggesting that, but it is what you should do if you want to be precise in your selectors. This approach was already mentioned in OGC 16-053r1 [4].

```
<style>
*[itemtype="http://www.opengis.net/road"] [itemprop="type"]{
    background:red;
}
</style>
<feature>
  <properties>
    <div itemscope itemtype="http://www.opengis.net/road">
      <span itemprop="type">road</span>
    </div>
  </properties>
</feature>
```

Exploring all these possible combinations together, and analyzing what is possible in CSS, some important limitations in CSS were found that prevent developers from doing things that are common in GIS such us conditioning the style of a geometry to some values of the properties or specifying a style declaration value as a function of a property value.

## 8.2. Limitations in CSS

In the experimentation done to apply CCS to features with properties and geometry, the following limitations were detected:

- You cannot select an element of the HTML and apply the symbol to another element
- You cannot set a selector based on the value of the element (you can do it based on an attribute of the element)
- You cannot set a selector based on the value of two properties at the same level.
- You cannot set a symbol declaration value (e.g. *width*) as a function of a value of an element or attribute in the HTML

Some experts suggest that the limitations in the selectors syntax were imposed on purpose to limit complexity of the parses and to allow a faster parsing of the HTML-CSS styler.

### 8.2.1. You cannot set a selector based on the value of the element.

CSS selectors can only select elements-based attributes but not on element values (a.k.a. element innerHTML). The use of the attribute "content" of microdata could be a fix to this limitation even if the value has to be repeated.

```
<style>
*[itemtype="http://www.opengis.net/road"] [itemprop="theme"][content="road"]{
    background:red;
}
</style>
<feature>
  <properties>
      <table a="b" class="table-properties" itemscope itemtype=
"http://www.opengis.net/road">
        <tbody>
          <tr>
            <th scope="row">id</th>
            <td itemprop="id">10964418e33d457aabd6f6ab10dc2e4a</td>
          </tr>
          <tr>
            <th scope="row">theme</th>
            <td itemprop="theme" content="road">road</td>
          </tr>
        </tbody>
      </table>
  </properties>
  <geometry>
     <polygon>
     </polygon>
  </geometry>
</feature>
```

To avoid the need to repeat the content as an attribute and as a value, you could use CSS content property as suggested here: https://www.w3schools.com/cssref/pr_gen_content.asp. In the following example, the innerHTML span element is populated with the content of the content attribute using the CSS declaration content. Unfortunately, this solution requires a complex notation that does not favors clarity.

```
<style>
feature *[itemtype="http://www.opengis.net/road"] [itemprop="theme"][
content="highway"] {
    background:red;
}

feature *[itemtype="http://www.opengis.net/road"] [itemprop="theme"]::after {
    content: attr(content);
}
</style>
<feature>
  <properties>
    <div itemscope itemtype="http://www.opengis.net/road">
      <span itemprop="theme" content="highway"></span>
  </properties>
</feature>
```

### 8.2.2. You cannot set select an element of the HTML and apply the symbol to another element

In principle, CSS was not designed to select some elements but to apply the style to another element. In our case, this means that in general it is not possible to define a selector depending on "properties" and apply this to "geometry". The only approximation to this behavior is to select the polygon that is a child of geometry that has a *precedent sibling* (using ~) with an attribute value.

```
<style>
properties[type="road"] ~ geometry polygon {
    background:red;
}
</style>

<feature>
  <properties type="road">
      <table>
      </table>
  </properties>
  <geometry>
     <polygon>
    polygon
     </polygon>
  </geometry>
</feature>
```

The use of JavaScript can help to overcome this limitation. The function querySelectorAll can be used to make use of selector of properties and className to apply the style to geometries.

```
<style>
.road_red {
    background:red;
}
</style>
<script>
function setColorsToGeometries()
{
    var roads=document.querySelectorAll('*[itemtype="http://www.opengis.net/road"]
[itemprop="theme"][content="road"]');

    for (var i=0; i<roads.length; i++)
    {
        var elem=roads[i];
        while (elem && elem.tagName.toLowerCase()!="properties")
            elem=elem.parentElement;
        elem.parentElement.getElementsByTagName("geometry")[0].className="road_red";
    }
}
</script>

<body onLoad="setColorsToGeometries()">
  <feature>
    <properties>
        <table a="b" class="table-properties" itemscope itemtype=
"http://www.opengis.net/road">
          <tbody>
            <tr>
              <th scope="row">theme</th>
              <td itemprop="theme" content="road">road</td>
            </tr>
          </tbody>
        </table>
    </properties>
    <geometry>
       <polygon>
       </polygon>
    </geometry>
  </feature>
</body>
```

## 8.3. Extensions of CSS to support geospatial requirements

One of the extensions needed the capability to apply a selector based on some *properties* values to the *geometry*. The authors of this ER propose to incorporate *condition1* attribute to point another selector that will add extra conditions based on elements that are not directly the ones to symbolize. Both the *selector* and the *condition1* should be of the same father.

A suggested possibility is:

```
<style>
feature polygon; condition1: feature *[itemtype="http://www.opengis.net/road"]
[itemprop="theme"][content="road"]
{
    background:red;
}
</style>
<feature>
  <properties>
      <table a="b" class="table-properties" itemscope itemtype=
"http://www.opengis.net/road">
        <tbody>
          <tr>
            <th scope="row">id</th>
            <td itemprop="id">10964418e33d457aabd6f6ab10dc2e4a</td>
          </tr>
          <tr>
            <th scope="row">theme</th>
            <td itemprop="theme" content="road">road</td>
          </tr>
        </tbody>
      </table>
  </properties>
  <geometry>
     <polygon>
     </polygon>
  </geometry>
</feature>
```

Another extension could be to condition a declaration value (e.g. width) to a property value (e.g. lanes). This could be achieved by using a selector as a value of a symbol declaration:

```
<style>
feature polygon; condition1: feature *[itemtype="http://www.opengis.net/road"]
[itemprop="theme"][content="road"]
{
    background:red;
    label: feature *[itemtype="http://www.opengis.net/road"] [itemprop="name"][
content];
    stroke-width: feature *[itemtype="http://www.opengis.net/road"] [itemprop="lanes"
][content];
}
</style>

<feature>
  <properties>
      <table a="b" class="table-properties" itemscope itemtype=
"http://www.opengis.net/road">
        <tbody>
          <tr>
            <th scope="row">id</th>
            <td itemprop="id">10964418e33d457aabd6f6ab10dc2e4a</td>
          </tr>
          <tr>
            <th scope="row">theme</th>
            <td itemprop="theme" content="road">road</td>
          </tr>
          <tr>
            <th scope="row">theme</th>
            <td itemprop="name" content="route 66">Route 66</td>
          </tr>
          <tr>
            <th scope="row">theme</th>
            <td itemprop="lanes" content="3">3</td>
          </tr>
        </tbody>
      </table>
  </properties>
  <geometry>
     <polygon>
     </polygon>
  </geometry>
</feature>
```

**NOTE**

More work on the use of CSS styling for geospatial objects can be found here:

- A hands-on tutorial from GeoSolutions training materials: https://geoserver.geo-solutions.it/edu/en/pretty_maps/css.html
- The GeoServer documentation getting started tutorial: http://docs.geoserver.org/latest/en/user/styling/css/tutorial.html
- The GeoServer "cookbook": http://docs.geoserver.org/latest/en/user/styling/css/cookbook/index.html
- The GeoServer Full list of supported properties: http://docs.geoserver.org/latest/en/user/styling/css/properties.html
- Full contents with more links here: http://docs.geoserver.org/latest/en/user/styling/css/index.html and also in https://carto.com/docs/carto-engine/cartocss/

## 8.4. Selecting alternative styles for MapML

This subsection assumes that CSS rule-sets are provided in an independent CSS file and linked from the MapML page.

There are two practical ways that emerged from the experiments done during OGC Testbed-14, which support a use case giving the opportunity to the user to select among a list of styles for the "same" map content.

One approach is to use the *link rel* approach to indicate that there are other alternative styles related to the same map available. This can be done by using the link/@rel=style to indicate alternative styles that will have a title and an href to another MapML document. The style currently in use can be tagged as "self style".

*MapML fragment that includes references to two styles (the one with rel=self is the current one), a reference to a legend and a reference to an alternative projection.*

```
<link rel="self style" title="CubeWerx" href=
"http://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.coastl_1m/cubewerx"/>
<link rel="style" title="Red Example" href=
"http://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.coastl_1m/red"/>
<link rel="legend" href=
"http://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/legendGraphics/Foundation.coastl_1m/cubewerx.png"/>
<link rel="alternate" projection="CBMTILE" href=
"http://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.coastl_1m/cubewerx?projection=CBMTILE"/>
```

In a client, alternative styles can be expected to result in alternative presentations available in the legend (e.g. as a drop-down selector or a group of radio buttons)

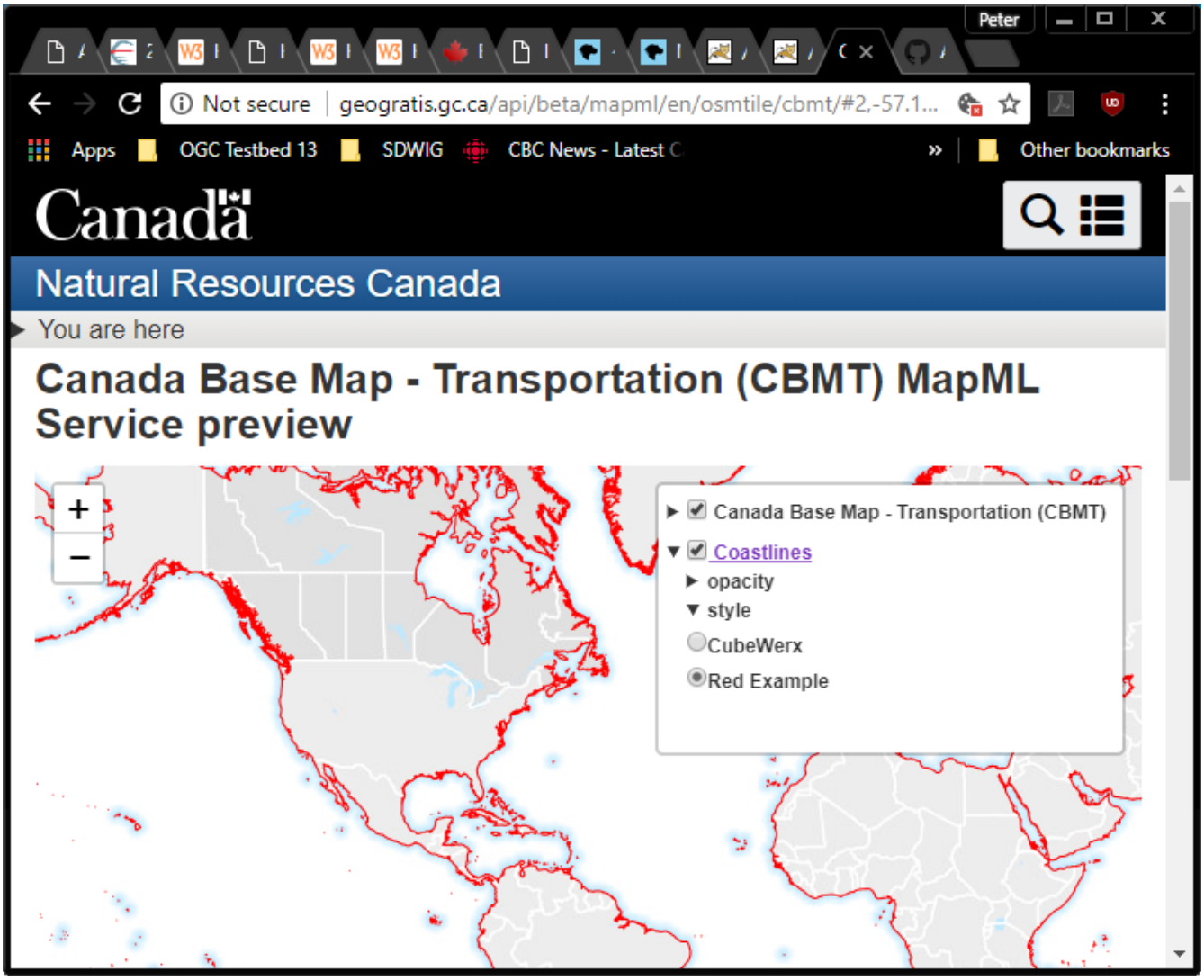


*Figure 4. Visualization of alternative styles a radio buttons in the legend.*

**NOTE**

*Alternative stylesheets*

Do not get confused with a similar but almost forgotten functionality introduced in HTML 4.0 that allows defining alternative stylesheets (alternative CSS) in HTML documents (https://www.w3.org/TR/html40/present/styles.html#h-14.3.1). These alternative styles were available through the web browser menu only in Firefox and Internet Explorer. Since the menu has been hidden in major web browsers it has become completely unknown.

<link rel="stylesheet" title="Gold" type="text/css" href="gold.css">

<link rel="alternate stylesheet" title="Oldstyle" type="text/css" href="Oldstyle.css">

With the recent introduction of selectors and URL templates in MapML, there might be other ways of achieving the exact same functionality. Actually, there are 2 similar ways of implementing a selector.

The first one involves considering the style name as a parameter in the URL template and using a select instead of an input.

*Alternative encoding for alternative styles.*

```
<body>
  <extent units="OSMTILE" method="get">
    <select mane="Style">
      <option value="cubewerx">Cubewerx</option>
      <option value="red">Red</option>
    </select>
    ...
    <link rel="tile" tref=
"https://tb14a.cubewerx.com/cubewerx/wmts/foundation_direct/combo/cubewerx/{Style}/{TileMatrix}/{TileRow}/{TileCol}.jop?INSTANCE=mapML"/>
  </extent>
</body>
```

The second one involves considering the style name as a parameter in the URL template and using a datalist as a selector.

*Alternative encoding for alternative styles.*

```
<body>
  <extent units="OSMTILE" method="get">
    <input name="Style" list="styles"/>
    <datalist id="styles">
      <option value="cubewerx">Cubewerx</option>
      <option value="red">Red</option>
    </datalist>
    ...
    <link rel="tile" tref=
"https://tb14a.cubewerx.com/cubewerx/wmts/foundation_direct/combo/cubewerx/{Style}/{TileMatrix}/{TileRow}/{TileCol}.jop?INSTANCE=mapML"/>
  </extent>
</body>
```

**WARNING**

MapML should decide which is the right way of doing selectors. In the authors opinion, a selector should be implemented as a <select>. The <datalist> tag introduced an "autocomplete" feature on <input> elements (https://www.w3schools.com/tags/tag_datalist.asp) but users of inputs are still free to populate them with other text outside the list of suggestions.

# Chapter 9. Events

Events are an important feature for DOM elements to signal changes generated by the use of the web browser on elements. Events trigger the JavaScript function handler that is executed when the event happens.

## 9.1. Maps4HTML events

In this section, the common situation of an HTML page that links to MapML documents using <layer src=""> is considered. Previously this ER discussed about the convenience that MapML documents are only read and interpreted by the web browser (and not by scripts) to avoid Cross-Origin Resource Sharing (CORS) problems. Still, it is important that the web browser supports scripts that some user actions had appended or are going to happen immediately after the event and report on the consequences on the map visualization. Events will make it possible for the script to trigger some actions on the information shown in the rest of the page that will provide some level of synchronization with what is happening in the map.

Currently, MapML is implemented by a custom element. Actually, custom elements do emit events, as described in web-map.html elaborated, to support MapML. For the moment, not much has been done with them but in some cases the latitude and longitude of the point related to a mouse action is added. This is the list of events that are mentioned in the custom element file:

*Table 4. List of events mentioned in the custom element file*

| event name | variables added |
|---|---|
| 'load' | |
| 'preclick' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'click' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'dblclick' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'mousemove' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'mouseover' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'mouseout' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'mousedown' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'mouseup' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'contextmenu' | {lat: e.latlng.lat, lon: e.latlng.lng, x: e.containerPoint.x, y: e.containerPoint.y} |
| 'movestart' | |

| event name | variables added |
|---|---|
| 'move' | |
| 'moveend' | |
| 'moveend | |
| 'zoomstart' | |
| 'zoomend' | |
| 'zoom' | |

It is the opinion of the authors of this ER that this long list might not be necessary. It might be better to simplify them to a few events that communicate all actions of the user. The authors of this ER propose a minimum set of events, in-line with how the form input controls works in HTML. For example, listening to a single event, the browser would be able to know about the new position and zoom level of an action of the user (whatever the origin of it) instead of having to listen to specific events of the mouse and keyboard. The following changes could be considered.

### 9.1.1. Proposed Maps4HTML events

Two types of events can be proposed: Common preexisting events that are enriched with more parameters or new events related to pan (map move) and zoom.

This is the proposed short list of events:

- The <map> element needs to include an "onChange" event that will be triggered each time that the map is going to be panned (moved) of zoomed in or out. The JavaScript function triggered by the event would receive an event variable that would contain the new extent (bbox) of the map area in PCRS and GCRS, the zoom level as well as the name of the TCRS.
  - A use case where this event could be useful is to overlay a <map> area containing MapML documents with a division containing some of the current or future JavaScript map APIs (e.g. Google Maps API) and being able to synchronize the view on both maps.
  - The event handler function could return true to authorize the change (the default) of false to indicate that the movement should not be executed.
  - A supported use case could be to limit the extent or the zoom levels that the user can navigate to, having an event handler function that will return false if the new extent does not comply with a set of rules.

| Property/Method | Description |
|---|---|
| mapml.topleft-easting | X min value of the bounding box in pcrs coordinates |
| mapml.bottomright-easting | X max value of the bounding box in pcrs coordinates |
| mapml.bottomright-northing | Y min value of the bounding box in pcrs coordinates |
| mapml.topleft-northing | Y max value of the bounding box in pcrs coordinates |
| mapml.topleft-longitude | min longitude value of the bounding box in gcrs coordinates |

| Property/Method | Description |
|---|---|
| mapml.bottomright-longitude | max longitude value of the bounding box in gcrs coordinates |
| mapml.bottomright-latitude | min latitude value of the bounding box in gcrs coordinates |
| mapml.topleft-latitude | max latitude value of the bounding box in gcrs coordinates |
| mapml.z | zoom level |
| mapml.units | name of the TCRS |

An example of a handling function that will prevent going beyond the longitude that is completely outside the interval [-180,180] is shown below.

```
<script>
function limitLongitude(event) {
    if (event.mapml.bottomright-longitude<-180 ||
        event.mapml.topleft-longitude<180)
        return false;
    return true;
}
</script>

<map width="700" height="400" onchange='return limitLongitude();'>
</map>
```

- The <map> element needs to include an "onClick" event that will be triggered each time that the map is clicked. The JavaScript function will receive an event variable that will contain the clicked point in PCRS coordinates and in GCRS coordinates as well as the TCRS name. The event will not contain any information on the features/attributes at that place because that could violate the intention behind CORS rules. The page https://www.w3schools.com/jsref/obj_mouseevent.asp tells us about the event variables that are received by a mouse event. The authors of this ER propose to add the following variables:

| Property/Method | Description |
|---|---|
| mapml.easting | X value of the clicked position of the screen in pcrs coordinates |
| mapml.northing | Y minvalue of the clicked position of the screen in pcrs coordinates |
| mapml.longitude | longitude value of the clicked position of the screen in gcrs coordinates |
| mapml.latitude | latitude value of the clicked position of the screen in gcrs coordinates |
| mapml.units | name of the TCRS |

- The <layer> element need to include "onload" and "onerror" events to allow the HTML page to know if the MapML document has been uploaded correctly and it is being shown.
    - A classical use case is to inform the user that there has been an issue with the requested

MapML that returned an HTTP error.

```
<layer src="mymap.mapml" onload="loadMapML();" onerror="errorMapML(this.src);">

<script>
function loadMapML() {
    alert("MapML document correctly loaded.");
}
function errorMapML(url) {
    alert("Error loading MapML document " +  url + ".");
}
</script>
```

| | |
|---|---|
| **NOTE** | The onLoad event can happen during the load of the page or any time when a new MapML document needs to be updated (something not common for a MapML based on URL templates) |

| | |
|---|---|
| **NOTE** | The onError event can happen because the main MapML has failed to load but it can also happen if a tile URL also fails to load. Failure of loading a tile could be produced by requesting a tile that is out of the limits of tile indexes. This case is relatively common and does not reflect a real issue. It could be useful to have an indication that the error is caused by the main MapML of one of their internal links. |

| | |
|---|---|
| **WARNING** | Note that a new layer can be added programmatically so the handler function needs to be prepared for unexpected layer names |

- The <layer> element also needs to include events that will be triggered just before the change in status of the boolean values such as *hidden*.
    - A common use case is to inform the main page that some extra information about the newly activated layer should also be shown.

# Chapter 10. Other aspects in MapML

This section collects some small discussions and topics that could be extended in further interoperability experiments.

## 10.1. MapML and CORS

In geospatial information, the capacity of being able to overlay maps coming from *different sources* is a basic feature allowed by the co-registration of the positions in a common coordinate reference system (or coordinate reference systems that can be mathematically mapped).

In web browsers, due to security reasons, there is a restriction that prevents content from different domain servers from being mixed in a single page by default. This is known as CORS and means that, by default, a JavaScript programmer cannot access content coming from another server (different that the one that loaded the page). The web browser is allowed to do that in some cases: for example, the browser can render PNG files coming from different domains in the same page with no problem. The same with HTML: the browser is allowed to render HTML in an iframe even if it is not from the same server. Again, JavaScript code cannot access this iframe content by default. It is possible that a server authorizes CORS by adding some lines in the headers but this requires active collaboration of all servers involved and that is not easy to achieve.

This opens up an interesting discussion: MapML is an new media type. The whole purpose of MapML is to empower browsers with geospatial map capabilities. When MapML is adopted by web browsers directly and browsers do all the rendering, then rendering a MapML document can be considered to be equivalent to rendering a PNG. A programmer can be sure that the browser is not going to "mix" content (only overlay it) so there should not be any need for extra CORS protection. When MapML is adopted by web browsers directly such documents will be rendered without any script intervention. If MapML documents are able to escape CORS security protection that will allow overlays of *n* MapML documents coming from different servers to be considered a case similar to having *n* images (included in <img src="">) and should be shown together if there is no script intervention. Indeed, MapML files will be opaque to the script and there will be no risk of mixing content coming from different servers or executing unknown scripts. However, the programmer will not have access to the content of any MapML DOM document outside its server by default.

This is extremely important because the current situation with JavaScript map technologies (e.g. Leaflet) is very limiting. By default, you cannot render a GeoJSON file that you found elsewhere on top of your data because of CORS. WMS is the exception of this if you only deal with GetMap and images, but as soon as you try to read a capabilities document, or use GetMap for retrieving vector data you encounter the CORS restriction. If MapML documents can be shown together, ignoring CORS security, it will provide a competitive advantage of MapML to other map technologies in the web that have to face CORS security protection in other more unconventional ways.

| | |
|---|---|
| **NOTE** | In the current implementations of MapML that are necessary based on JavaScript code, it should be possible to overlay data from different services by assimilating a layer element to an iframe and loading an instance of a map browser (e.g. loading the leaflet library) in each iframe. A programmer only needs to make the iframes transparent and draw them one on top of the other. Unfortunately, this is not a complete solution because synchronization among iframe will be a problem anyway. |

Even if the web browser allows for showing MapML files from different sources, there is a ramification of CORS that should not be ignored. If an HTML page includes a MapML file in another domain that has a URL template that will result in including feature information that comes from a 3rd server, then the result is more unpredictable. Currently a similar situation occurs when an HTML page loads an SVG that has links to other files (e.g. an SVG describing an icon). In SVG this is not allowed if the three files come from different services. If this is going to be the final implementation decision, this would limit what MapML will be able to do anyway.

## 10.2. Languages in MapML

Maps4HTML can take advantage of the language negotiation already implemented in the HTTP protocol (see https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html). By implementing language negotiation, when an HTML page is requested by the client, the headers include a line with a list of languages that the user can understand and has specified in the browser options by preference order. The server tries to accommodate the needs of the user depending on the languages available in the server side. This can also be used in HTML documents containing maps.

MapML could also use a meta tag to mark the language of the map as well as a link rel to suggest alternative languages.

## 10.3. Considerations on multidimensional data (and time) in MapML

The capability of adding <select> elements in the <extent> allows for implementing support for selecting time and other dimensions in URI templates

```
  <extent>
     <location />
     <select name="dimension-time">
       <option id="2012-13-10Z12:00:00">12:00</option>
       <option id="2012-13-10Z13:00:00" selected>13:00</option>
     </select>
    <select name="dimension-band">
       <option id="band3">red</option>
       <option id="band2" selected>green</option>
    </select>
    <template type="tile" tref=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/{dimension-band}/{dimension-
time}/CBMT3978/MapServer/tile/{z}/{y}/{x}?m4h=t" />
  </extent>
```

**NOTE** This is good for discrete variables, but it is still unclear how to specify a time interval instead of a list of values. It is also unclear how a time animation could be specified.

# Chapter 11. Testbed-14 Implementations

This chapter describes two implementations elaborated on during OGC Testbed-14.

## 11.1. Cloud Based Proxy (cascade) for MapML

This section describes an implementation of a CubeWerx-delivered cloud-based MapML service to proxy (or cascade) existing Canadian Geospatial Data Infrastructure (CGDI) OWS content services as MapML hosted at https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts.

### 11.1.1. Requirements

The implementation was designed to satisfy the MapML Cloud Proxy requirement as described under the CFP MapML (https://portal.opengeospatial.org/files/?artifact_id=77327#MapML) requirements.

### 11.1.2. Approach

The service can serve MapML by cascading from existing CGDI OWS content services (WMS, WFS, WMTS, ESRI Map Server). The server-side solution also satisfies the chaining requirement. Web browser caching is provided using standard browser mechanisms. The MapML server also implements the ability to serve Mapbox vector tiles as specified by https://github.com/mapbox/vector-tile-spec/tree/master/2.1.

### 11.1.3. Implementation

CubeWerx has implemented and delivered a MapML service that serves MapML by cascading (proxying) from existing CGDI OWS content services (e.g., WMS, WFS, WMTS, ESRI Map Server, etc.). It is currently configured to proxy all of the layers from the WMS server at "http://geo.weather.gc.ca/geomet/" as well as all of the layers from the ArcGISMap server at "https://geoappext.nrcan.gc.ca/arcgis/rest/services/Energy/clean_energy/MapServer". Some local VMAP Level 0 layers are also served to provide appropriate base maps.

The list of the MapML layers that are proxied from the WMS GeoMet server is available at the following URL:

http://geogratis.gc.ca/api/beta/mapml/static/cubewerx_cgdi_proxy_geomet_directory.html

The MapML layers that are proxied from the clean_energy ArcGISMap server are:

Biomass Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.0

Hydro Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.1

Solar Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.2

Tidal Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.3

Wind Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.4

Nuclear Generating Stations (MW):

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Energy.5

The local VMAP Level 0 layers are:

Airport Facility Points:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.aerofacp_1m

Built-Up Areas:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.builtupa_1m

Coastlines:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.coastl_1m

Combination Background:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.combo

Depth Contours:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.depthl_1m

Global 2 Minute Elevations:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.etopo2

Global 30 Second Elevations:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.gtopo30

Inland Water Areas:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.inwatera_1m

Oceans/Seas:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.oceansea_1m

Political Boundaries:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.polbndl_1m

Railroads:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.railrdl_1m

Roads:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.roadl_1m

Trees:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.treesa_1m

Water Courses:

https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/Foundation.watrcrsl_1m

This implementation provides more than a simple proxy, since it can translate from one type of service to another. Furthermore, it has the ability to perform coordinate-system reprojections on the fly. For example, all of the above-listed MapML layers are available in the OSMTILE, CBMTILE and APSTILE projections (as defined by the MapML specification), even though the proxied services do not serve these projections.

As hinted by the above-listed MapML layer URLs, this OGC Testbed-14 MapML service has been implemented as an extension of the OGC Web Map Tile Service (WMTS) 1.0.0 specification. It is not the intention to formalize this extension. The fact that it has been implemented as such an extension is irrelevant to MapML clients and should be considered merely an implementation detail. This choice was made solely to leverage the codebase of an existing service.

A significant portion of the implementation effort was evolving the initial MapML implementation from Testbed-13 [2] to keep up with the many changes that have been made to the in-progress MapML specification. These implementation changes included, but were not limited to:

- adding support for tile-URL templates (to avoid the need for a server to request a new MapML document for every pan and zoom).
- adding support for URL domain "sharding", allowing tile requests to be more parallelized and possibly even split among multiple servers for greater scalability.
- adding support for legend graphics.
- adding support for multiple styles, including the ability for a MapML document to provide links to the alternate styles of a layer.
- adding the ability for a MapML document to provide links to alternate projections of a layer.
- adding the ability to perform feature-info queries on a layer.

Also, another implementation from OGC Testbed-14 is the ability for the MapML server (and by extension, the supporting WMTS server in general) to serve Mapbox Vector Tiles as specified by https://github.com/mapbox/vector-tile-spec/tree/master/2.1. This is not supported through the proxy mechanism at the moment, so is only available for local layers. It is invoked by appending a "format=application/vnd.mapbox-vector-tile" parameter to the MapML request.

Example: https://tb14.cubewerx.com/cubewerx/cubeserv/mapML/wmts/1.0.0/mapML/

Foundation.coastl_1m?format=application/vnd.mapbox-vector-tile

## 11.2. ServiceWorker Proxy for MapML

This section describes the creation by George Mason University (GMU) of the ServerWorker proxy for MapML that uses existing CGDI OWS content services [http://www.nrcan.gc.ca/earth-sciences/geomatics/canadas-spatial-data-infrastructure/19359] as MapML, potentially hosted at https://maps4html.org/cgdi/. The proxy supports vector tiles in the client HTML <map> element and 'offline' MapML maps, using the ServiceWorker browser API (https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API) together with the OGC GeoPackage (https://en.wikipedia.org/wiki/GeoPackage) MapML extension. The application takes a MapML document and injects the web components library call and the web-map.html call in the MapML documents. This gives automatic support to MapML documents for web browsers that do not support MapML natively (all web browsers fall in this category at the time to write this document).

### 11.2.1. Requirements of MapML Browser Extension

```
A Browser Extension shall be defined and implement the HTML <map> element API and event model as a browser extension (progressive web application), to make programming maps on the Web an exercise in the application of browser-supported standards.
```

### 11.2.2. Approach

The implemented deliverable, called a Web-Map-Application, browsed a special type of document which embedded MapML <map> elements but did not import the *Web-Map-Custom-Element* library. The deliverable injected specific sentences which imported the *Web-Map-Custom-Element* library into the end of the <head> element part of the document intentionally. The deliverable is a form of web-based application which works on a web browser, and it consists of two parts: an outer document for the deliverable logic itself, and an inner document for browsing an embedded MapML <map> document. The following Figure 5 shows how the approach works.

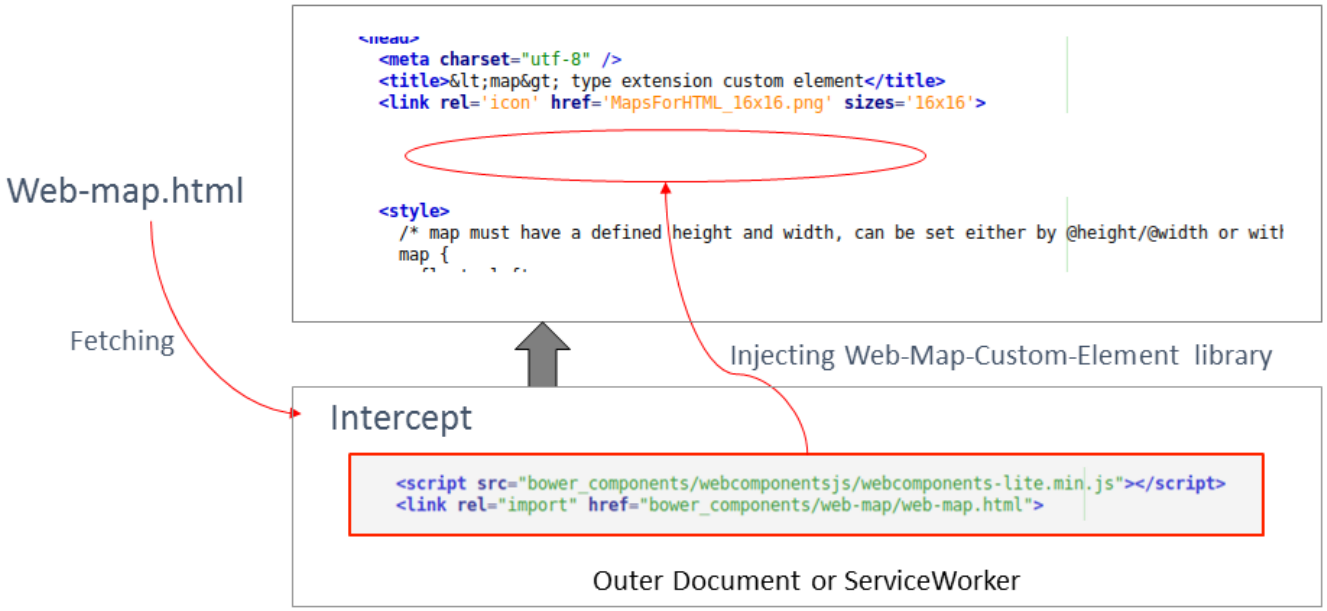


*Figure 5. A key idea of the approach*

### 11.2.3. Restriction

Unlike stand-alone applications, the deliverable is a form of web-based application, and inner documents apply the CORS cross-origin restriction policy for security. There is no way to import an embedded MapML <map> element document as an inner document. To avoid this issue, the deliverable used proxies on the server-side of the web application.

### 11.2.4. Software Design

A deliverable, *Web-Map-Application*, is a web-based application, and it is deployed at a service access endpoint URL. The web server was Apache and did not use any Common Gateway Interface (CGI) processing on the server-side, but because of using proxy servers, the server requires a few *ProxyPass* settings which are supported by Apache HTTP configuration, for bypassing to proxy servers by starting with given prefixes in path of fetched URL.

When a web browser accesses a *Web-Map-Application* endpoint, the application is ready to work. The application has an input text area for acceptance by an end-user, and if the end-user types a URL of embedded MapML <map> HTML document and presses <Enter> key, the internal logic of the application is activated, thereafter the document can be browsed on the <iframe> area in the web browser. The following Figure 6 shows an overview of how the *Web-Map-Application* works.

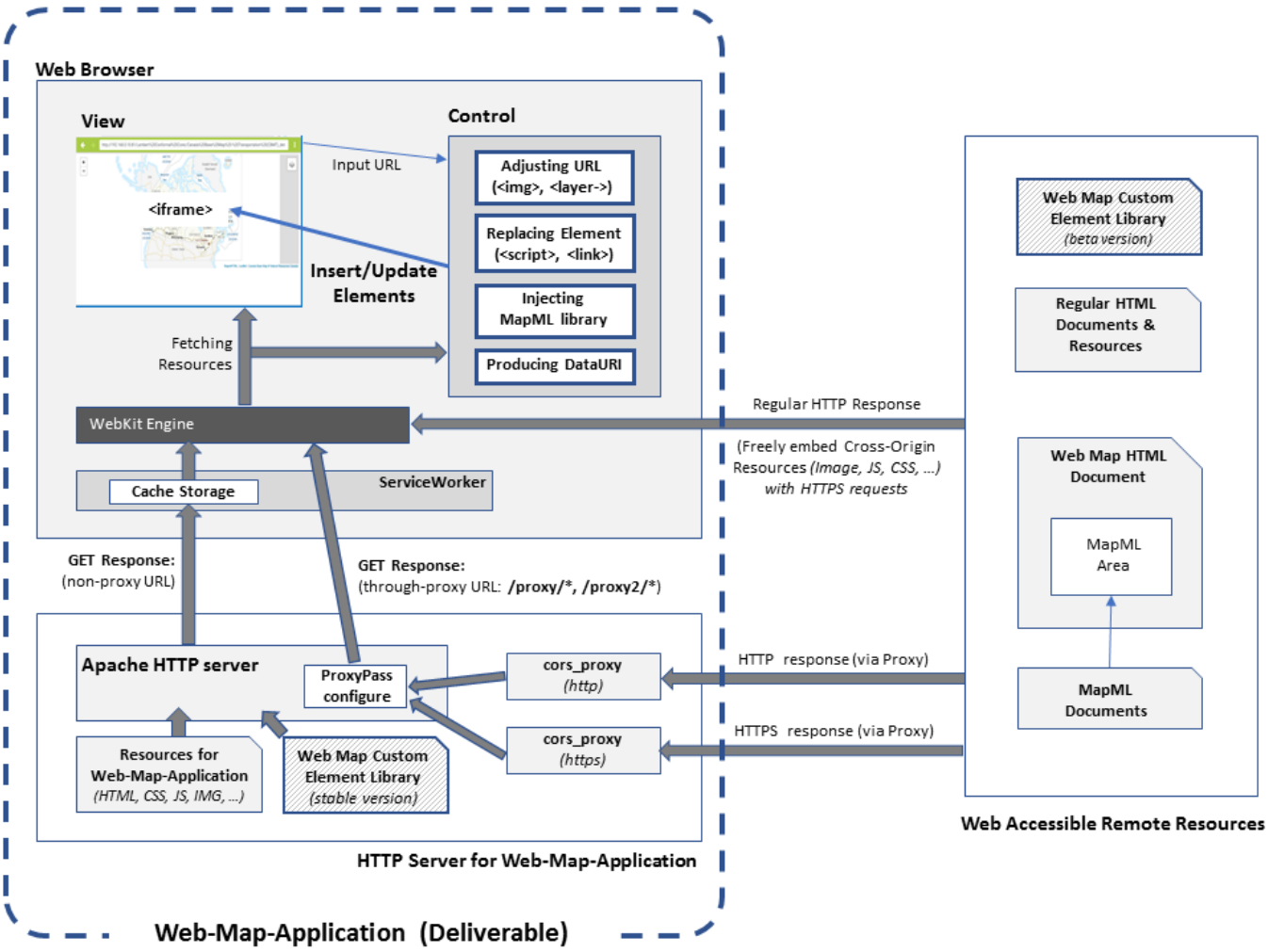


*Figure 6. An overview of the Web-Map-Application*

### The Server-side part (HTTP Server)

The server-side part is the same as a traditional static web application which is based on an Apache HTTP Server. There is no server-side logic processing except supporting client-side requests for acquiring static resources such as web documents, JavaScript codes, stylesheets, images, including *Web-Map-Custom-Element* library as well. All static resources for the *Web-Map-Application* are produced by the Angular command line interface (CLI) from an Angular 6 project.

To avoid cross-origin policy restrictions, the *Web-Map-Application* applies the proxy approach if the required URL meets the cross-origin restriction rule. By proxy configuration of Apache HTTP Server, some specific prefixes of the requested path in URLs are forwarded to another URL. The following table shows which prefix is forwarded by the web server.

*Table 5. URL replacement rule for applying proxy*

| Prefix of URL | Forarding to | Destination |
|---|---|---|
| /proxy/hostname:port/path | http://localhost:1447 | http://hostname:port/path |
| /proxy2/hostname:port/path | http://localhost:1448 | https://hostname:port/path |

### The Client-side part (Web Browser)

To be a web browser for client-side use, a browser should support running *ServiceWorker*. Currently a few (but market dominant) web browsers support the ability to run *ServiceWorker*, even though each of them only supports a different portion of specification. The following Figure 7 shows how a *ServiceWorker* starts, and how to intercept and respond with a response using a URL for a fetching request on the client-side, via local storage instead of fetching an external resource or web browser built-in cache.

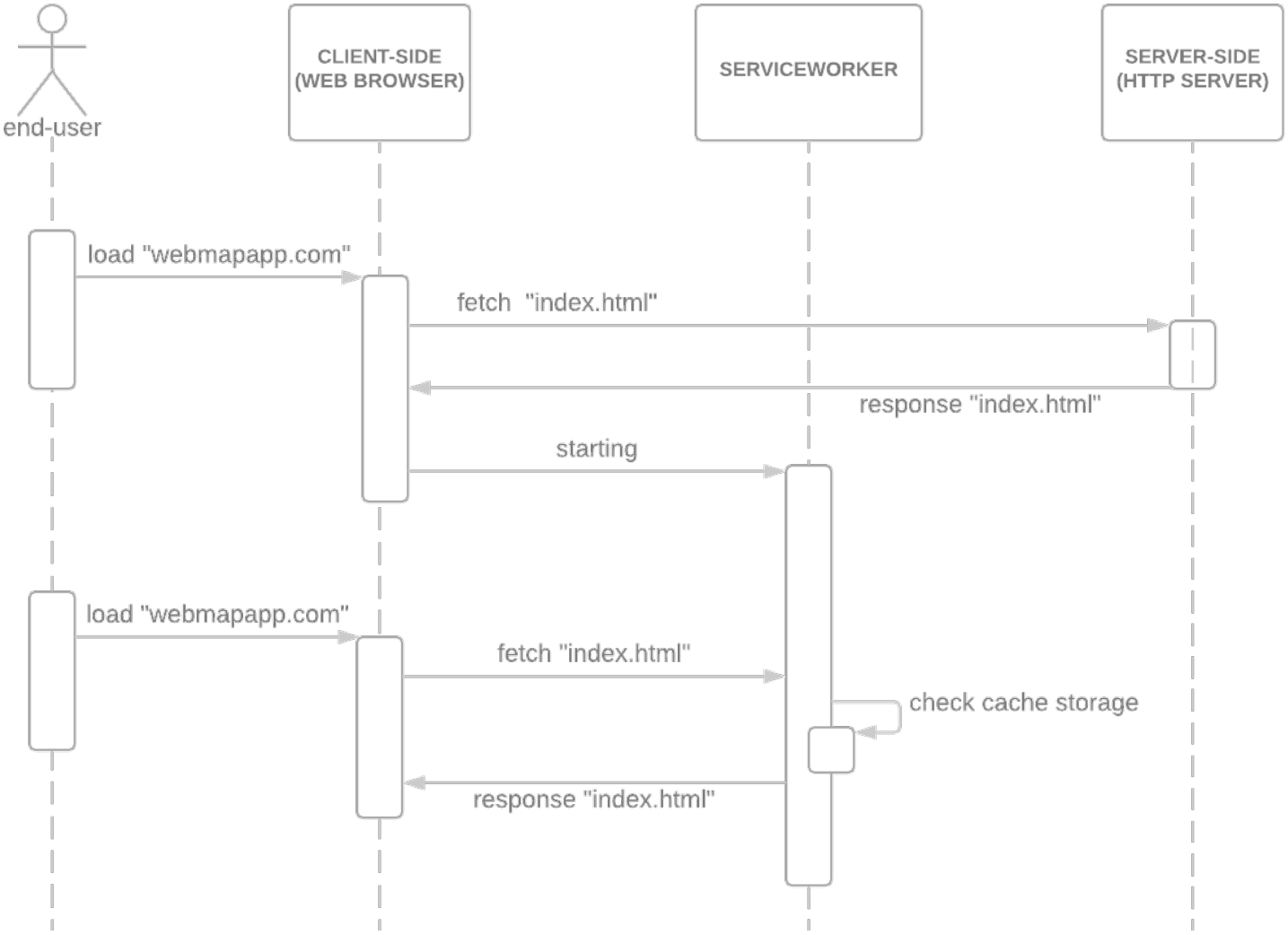


*Figure 7. A sequence diagram of working ServiceWorker*

### Proxy Rules

The inner document area for browsing embedded MapML <map> element document uses an <iframe> area in the outer document. Although an HTML document is free from cross-origin restriction, almost all modern web servers prohibit access from different domains in an <iframe> by setting 'X-Frame-Options' as a 'sameorigin'. To solve this issue, *Web-Map-Application* uses proxy servers. Not all hyperlinks in an HTML document require the use of a proxy, but many of the elements which have “src=” or "href=" attribute should be accessed by using a proxy. In addition, because many modern web browsers displays warnings or block contents if they are a mixed content HTTP request or HTTPS connection of *Web Map Application*, they should be fetched via a proxy to avoid being blocked by the web browser. The following shows which URL-referencing elements are replaced with a proxy URL:

- Candidate elements: <script>, <link>, <img>, <layer->, <a>
- A URL if accessing via HTTP protocol

- A URL if containing hostname
- A URL if using absolute path with no hostname
- In case of <script> and <link>, create new one with proxy URL and replace it with the original
- In case of <img> and <layer->, change "src=" attribute value with proxy URL
- In case of <a>, replace the *onclick* event handler for replacing "href=" with proxy URL

An *onclick* events handler raised by <a> elements should be replaced with a *Web Map Application* defined handler, because the default event handler is controlled by a web browser and it does not request its "src=" URL via proxy.

#### Using DataURI

One easy way to manipulate an inner document is using DataURI as a URL for the "src=" attribute of <iframe>. A DataURI can be created easily in JavaScript as shown below.

*Generating DataURI for HTML string (JavaScript code)*

```
'data:text/html;charset=utf-8,' + encodeURIComponent( HTML_string );
```

To use DataURI, the location of the inner document is treated as a different origin from the outer document in a web browser. The script is blocked from accessing the inner document from the outer document. It means that it is impossible to replace an *onclick* event handler of an <a> element, and an end-user cannot move forward by clicking hyperlinks in the inner document.

| | |
|---|---|
| **NOTE** | In the case of an HTML type, unlike other file types, the type is still supported in all major web browsers, but because of security issues, it is not recommended to be used. |

#### Loading inner document in <iframe>

*Web Map Application* uses <iframe> as the main area for displaying inner HTML documents while avoiding cross-origin restriction. In the most extreme case (except in case of Maps4HTML document), the application displays elements first and then fixes wrong elements after. This approach works fine in many cases, and allows for traversable inner documents without any restriction. The following Figure 8 shows how this loading works.

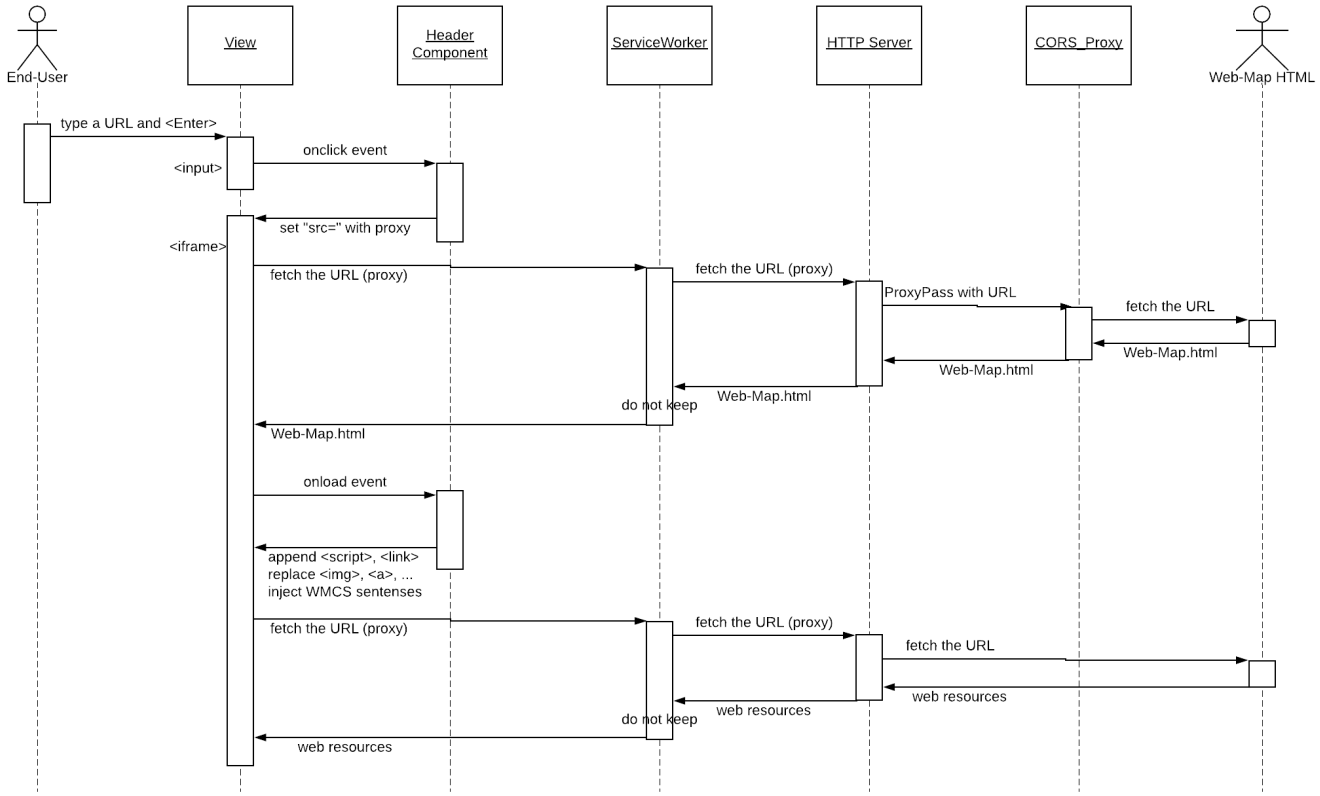


*Figure 8. A sequence diagram of loading Maps4HTML document*

But because the approach of creating and replacing <script> or <link> elements works dynamically, so there is no guarantee of loading order, and it may cause synchronization issues between imported scripts and documents. In the case of the Maps4HTML library in a document causing the same issue, and to avoid this issue, the application uses DataURI as a URL in <iframe> only if the inner document is a Maps4HTML document. Internally, the application checks a document type with the following rule:

- If the inner document **has** one or more <map is="web-map" ... > element(s), it is a Maps4HTML document.
- If the inner document **does not have** any <map is="web-map" ... > element, it is a normal document.

One experimental approach for the hyper-link replacing job was performed by the *ServiceWorker*. This approach was simple and increased document browsing capability, because *ServiceWoker* can catch all fetched links whatever element the link belongs to. However, because of a browser compatibility issue, it is hard to guarantee support on all web browsers. Currently, *ServiceWorker* in *Web-Map-Application* only keeps resources of the application by using cache storage.

### Injecting Maps4HTML library Rule

If an inner document is a Maps4HTML document, but the document does not have Maps4HTML library, the application injects the required library in the end of the <head> area of the document intentionally. The following codes are injected.

*Injecting Beta library*

```
<script src=
"http://geogratis.gc.ca/api/beta/mapml/client/bower_components/webcomponentsjs/webcomponents-
lite.min.js"></script>
<link rel="import" href="http://geogratis.gc.ca/api/beta/mapml/client/bower_components/web-
map/web-map.html">
```

*Injecting GitHub/Web-Map-Custom-Element library*

```
<script src="/assets/Web-Map-Custom-Element/bower_components/webcomponentsjs/webcomponents-
lite.min.js"></script>
<link rel="import" href="/assets/Web-Map-Custom-Element/web-map.html">
```

| **NOTE** | Actually, the URLs described above are replaced with proxy URLs. |
|---|---|

**WMS Tile Image Cacing Issue**

*ServiceWorker* is capable of keeping browsed WMS tile images in its local storage for increasing browsing speed and offline working, but because *Web Map Application* uses DataURI only if a Maps4HTML document and the DataURI is not allowed to access inner documents, WMS tile images are not kept by the *ServiceWorker*. It is just because of the cross-origin restriction rule, so if the URL does not meet the cross-origin restriction criteria in a Maps4HTML document, there is no problem for the *ServiceWorker* to keep WMS tile images.

### 11.2.5. Software Environments

A *ServiceWorker* component works only if a web browser is accessing via an HTTPS connection and programming source code of the *ServiceWorker*'s role is usually located in root path. That's why the endpoint uses a standalone domain name, *wma.webmapapp.com*.

- **Operating System**: Ubuntu 16.04 LTS (or 18.04 LTS)
- **Web Server**: Apache HTTP Server 2.4.33
- **Web Application Platform**: Angular 6
- **Web-Map-Custom-Element library**:
  - **stable**: (GitHub) Maps4HTML/Web-Map-Custom-Element
  - **beta**: (URL prefix) http://geogratis.gc.ca/api/beta/mapml/
- **Proxy Server**:
  - **HTTP**: corsproxy 1.5.0 (modified)
  - **HTTPS**: corsproxy-https 1.5.2 (modified)
- **Web Browser Compatibility**: Chrome, Firefox, Safari, MS Edge

## 11.2.6. User Interface

### Access Endpoint

A *ServiceWorker* component which is a key component of a Progressive Web approach works only if the web browser is accessing a URL via an HTTPS connection and a description of a *ServiceWorker*'s role may be located in the root path. That is why the endpoint uses a separate domain name, *webmapapp.com*.

### Main Screen

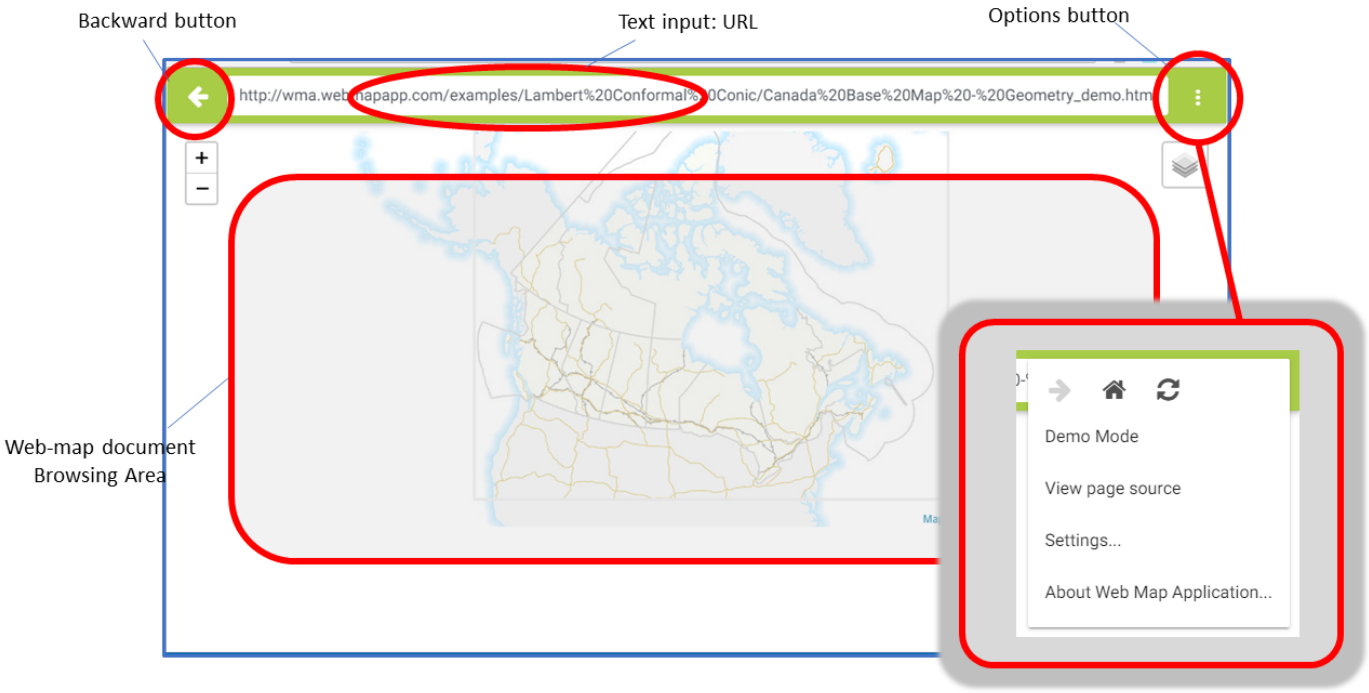


*Figure 9. A Capture Image of Main Screen*

- **Text Input for URL**: an input area for accepting a URL which is pointing to a Web-Map HTML document
- **Backward Button**: a button for moving backward from the current page through the visited history
- **Option Button**: a button for invoking the options panel (under construction)

### Menu Items in Options

- **Forward Button**: a button for moving forward from the current page through the visited history
- **Home Button**: a button for going to the home screen
- **Reload Button**: a button for reloading the current URL
- **Demo Mode Item**: a menu item for setting the view area to demonstration mode
- **Settings Item**:
  - **Web-Map-Custom-Element Library**: a selector of which Maps4HTML library is chosen for injecting into a non-library Maps4HTML document
  - **Maps4HTML Viewing Mode**: a selector of which viewing policy for a Maps4HTML document is chosen

    - **Console Log Level**: a selector of the log level for the development mode
- **About Item**: a menu item for displaying application information and version

# Chapter 12. Including MapML in social media.

This section is inspired by a Peter Rushforth presentation slides that suggests that MapML is simple enough to be used by children trying to learn about how to build web pages. HTML declarative language makes it particularly easy to progressively introduce children to programing languages via JavaScript and onwards to other programming languages at a later stage.

*Figure 10. Developers of the future*

Even if that is a plausible approach, modern children are spending more time in social media and they constantly produce and share content there (and, by the way, use tablets and mobile phones instead of laptops) in very simple ways. Social media provides tools to produce attractive content that combines text, images, videos and *emoticons*. The picture suggests an extra level of simplifications that makes exchanging a map as simple as exchanging a picture in *Whatsapp*.

## 12.1. The use of MapML as the new media for exchanging maps.

To be able to exchange maps, they should exist in some form that can be saved, attached, copied and pasted. Google Maps provides a mechanism to share maps by including a short URL to the Google Maps web page with some parameters to indicate, among others, the zoom level and pan, and being able to reproduce the shared view. This URL can be pasted anywhere to exchange the map view.

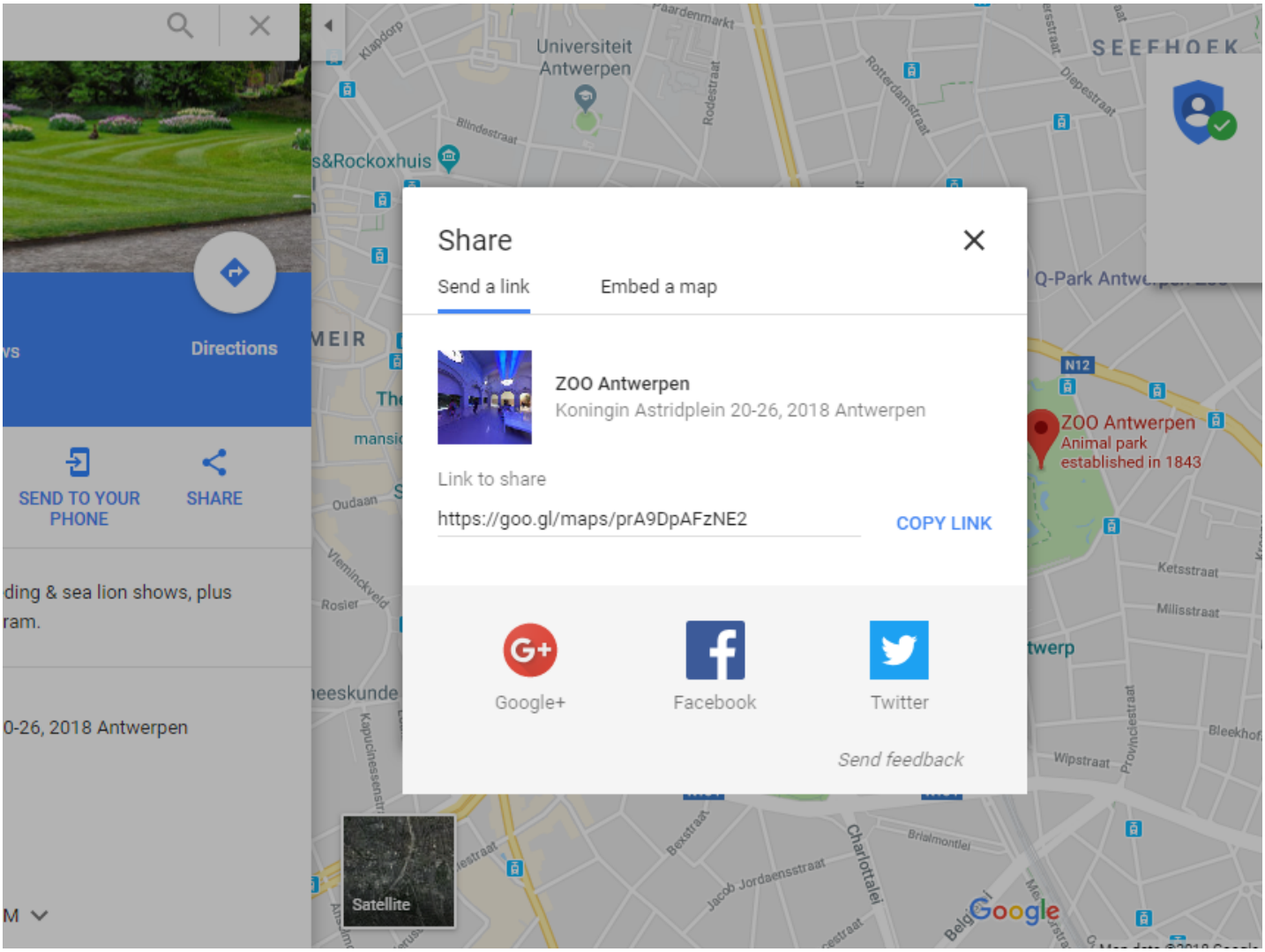


*Figure 11. Google Maps Share functionality.*

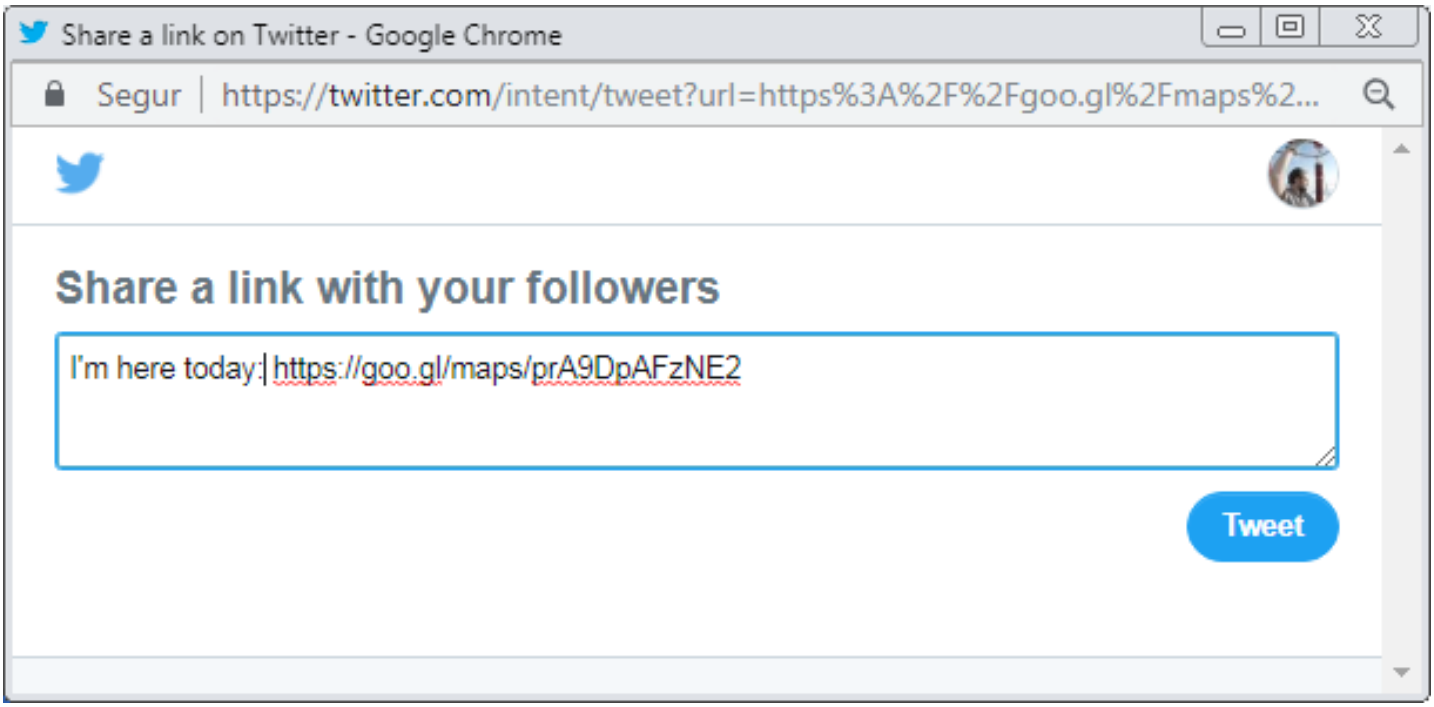


*Figure 12. Google Maps Share using Tweeter*

The main limitation of the Google Maps approach is that it allows only map content already prepared by Google to be shared, restricting users to a nice orthophotography or a road map.

To be able to exchange maps as easy as if they were images, what is needed is to make creation easy as well as provision of a way to embed the maps on social media messages.

Imagine that a user is looking at the following forest fire map and the user would like to share it with people that might live in the area at risk:

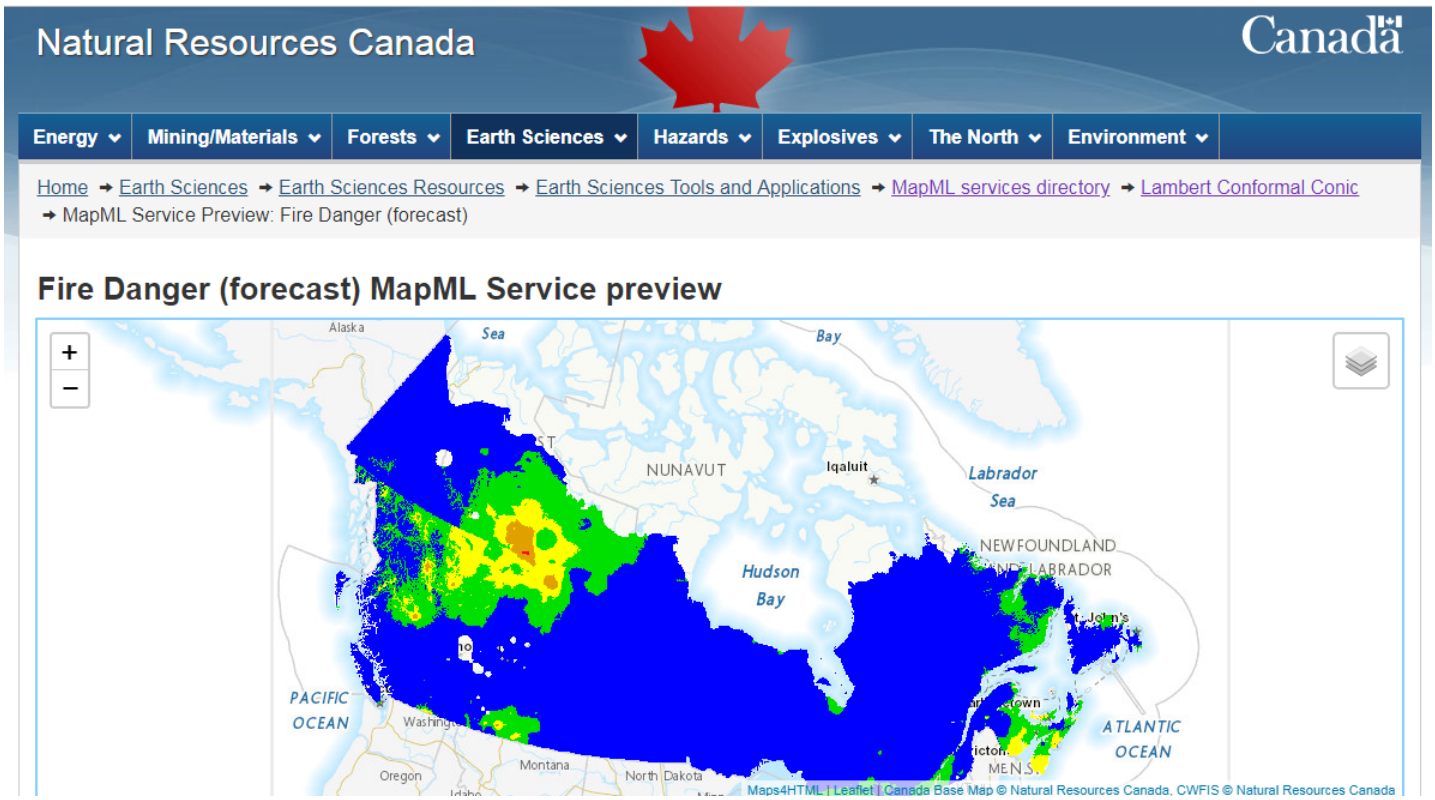


*Figure 13. Fire Danger MapML preview*

The URL to the MapML file is this: https://geogratis.gc.ca/mapml/en/cbmtile/fdi?alt=xml

It could be good to be able to add to the URL the current view configuration. One possible way to do that is to add this to the MapML URL directly using the parameters of the extent. The following listing shows how the MapML document looks like internally.

*Fire Danger MapML fragment*

```
<extent units="CBMTILE" method="GET">
<input name="projection" type="projection" value="CBMTILE"/>
<input name="txmin" type="location" units="tilematrix" position="top-left" axis="easting"/>
<input name="tymin" type="location" units="tilematrix" position="bottom-left" axis="northing
"/>
<input name="txmax" type="location" units="tilematrix" position="top-right" axis="easting"/>
<input name="tymax" type="location" units="tilematrix" position="top-left" axis="northing"/>
<template type="tile" tref="
http://maps.nofc.cfs.nrcan.gc.ca/geoserver/public/wms?service=WMS&version=1.1.0&request=GetMa
p&layers=public:fdr_current&styles=&bbox={txmin},{tymin},{txmax},{tymax}&width=256&height=256
&srs=EPSG:3978&FORMAT=image/png&TRANSPARENT=TRUE&m4h=t"/>
</extent>
```

If the extent is added into the map, the result URL will look like this: https://geogratis.gc.ca/mapml/en/cbmtile/fdi?alt=xml&txmin=-2143734.9&tymin=-1330081.2&txmax=-111730.9&tymax=701922.8 This URL needs to be interpreted by the browser that has to produce a map with the required data.

| | |
|---|---|
| **NOTE** | Currently, the implementation of Natural Resource Canada is already producing similar URLs but with semantics that do not correspond to the MapML extent syntax: https://geogratis.gc.ca/mapml/en/cbmtile/fdi/#3,-88.9421282,61.0858642 |

It could also be possible to consider the exchange of MapML files directly (instead of links). Internet users seem to be used to uploading files in some cases (e.g. JPEG images). There are some reasons for not considering this as a possibility: First, MapML needs to be interpreted by web browsers and no other type of client applications (that can open MapML files) are foreseen so far. Secondly, most MapML fields contain links to external content (e.g. a tile server); meaning that even if you save them in your hard drive, they cannot be used off-line in the same way that you save a JPEG file. Actually, having more tools to create and exchange MapML files should be a priority to make MapML files presence ubiquitous but it is assumed that data producers should be the ones that create them. The production of more MapML documents was one of the objectives in the current Testbed.

## 12.2. Drag and drop maps.

The current implementation of the NRCan MapML reader has the advantage of the drag and drop functionality to allow adding a map to another map and showing both simultaneously. The action is extremely easy. A user just has to drag a link from a list of possible Canadian maps and drop it on another map that the user pre-opened in another window.

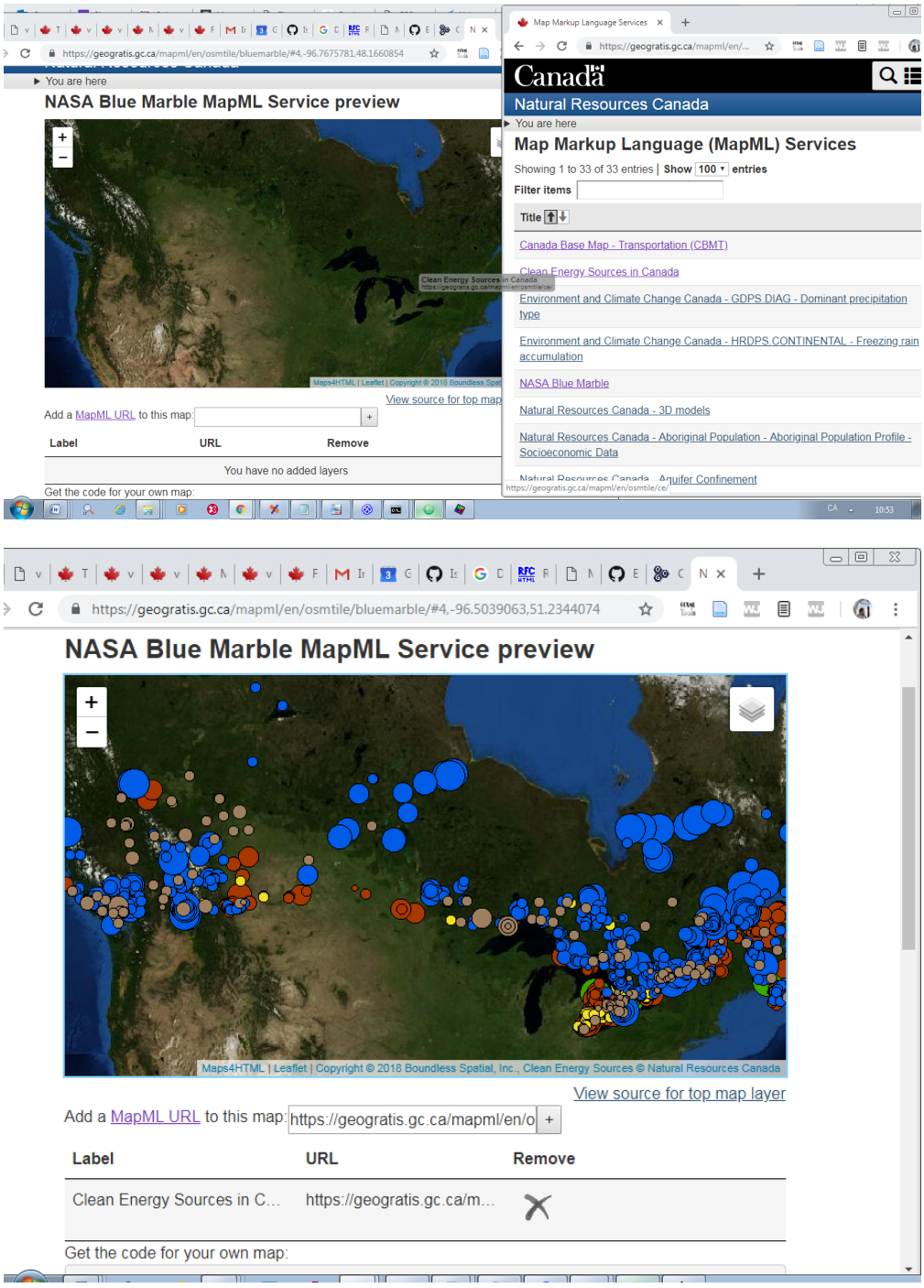





*Figure 14. Result of dropping a Clean Energy Sources in Canada map into a Blue Marble map page.*

Unfortunately, drag and drop is not possible on tablets and mobile phones. The action of copying a URL and pasting it in another page does not work on current Android devices. There is a workaround that involves copying the URL and pasting it in the "add MapML URL" box that is just below the map. This has the same effect that drag and drop but it is not so elegant.

## 12.3. Tests trying to share maps

Sharing maps in an Android phone is possible and elegant. A long press on a link results in a menu that has an option "share link" that opens a list of possible apps. Selecting Twitter will result in a tweet message with the URL already in it. A user only needs to complete the text and send it. If a user is already on the page, looking at the map, the 3 vertical dots icon (the so called "hamburger") offers a "share..." option that results in the same list of apps for sharing information.


*Figure 15. How to share MapML from a mobile phone in twitter.*

**NOTE**

*Request to add a "share" button in the NRCan page template.*

The authors of this ER suggest adding a "share" button in MapML NRCap page template. The results should be a small window to share with Twitter, Facebook etc. as well as a way to copy the current URL of the MapML or MapMLs currently in the view. This will facilitate the sharing of maps in social media.

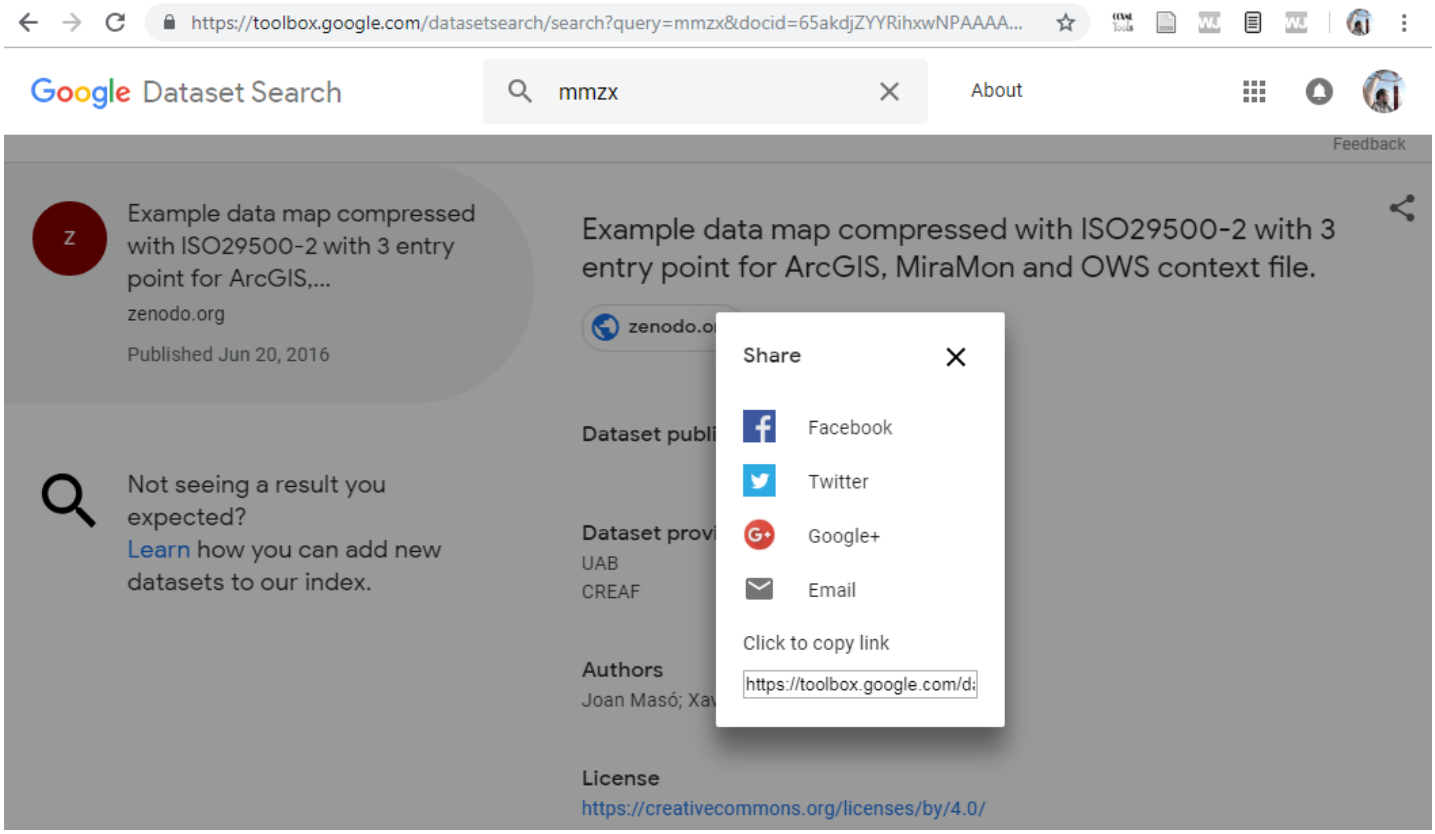


*Figure 16. Example of the "share" mini-window that can be incorporated in the NRCan map page template to facilitate the sharing of maps.*

# Chapter 13. OGC Web service to produce MapML.

Section 8 of the OGC Testbed-13 MapML ER explored the possibility of using current WMS or WMTS to generate MapML documents. The document contains abundant documentation on the details of the study as well as the CubeWerx implementation of a WMTS. The following table summarizes the main results and differences.

*Table 6. Comparison of WMS and WMTS to create MapML documents*

| Characteristic | WMS | WMTS | MapML |
|---|---|---|---|
| TileMatrixSet aware | No | Yes | Yes |
| Support for multitile requests | No | No | Yes |
| Support for extent | Yes | No | Yes |
| Support for discrete zoom | No | Yes | Yes |
| Support for vector features | No | No | Yes |
| Support for map images | Yes | No | Yes |

It is the opinion of the authors of this OGC Testbed-14 ER that neither WMS nor WMTS have all the necessary characteristics to generate MapML files without adding vendor parameters. To be able to generate MapML documents, there is a need for a service that takes characteristics from both WMS and WMTS: it should incorporate tile matrix set support but would be able to provide a map (and not only a single tile) from a bounding box.

In the past, there has been speculation on the possibility of integrating WMS and WMTS. MapML seems to be the perfect use case justifying that integration.

## 13.1. A proposal for new services that integrate maps and tiles

OGC is proposing a new architecture for services based on Web APIs and supporting tools such as OpenAPI. The WFS Standards Working Group (SWG) has already released a draft for WFS that will include an OpenAPI description document for a service. The OpenAPI document will identify two types of resources (collections and items) that can be served as URLs using the following two URL templates:

```
/collections/{collectionId}/items
/collections/{collectionId}/items/{featureId}
```

OpenAPI allows us to define both "path" and "query" parameters. WFS has also defined a query parameter for a BBOX applicable to the first of the two URI templates.

Following the same pattern, a service is possible that is able to retrieve a collection as a portrayal. This portrayal could be retrieved as a complete representation or as a fragment. Two types of fragments can be offered: maps or tiles. Due to the nature of tiles, they could be recovered one by one or as a set of adjacent

tiles. The following are the proposed URI templates that could be used to retrieve the map or the tiles.

```
/collections/{collectionId}/{styleId}/map/{crsId}
/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}
/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{TileMatrixId}
/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{TileMatrixId}/{tilerow}/{tilec
ol}.png
```

/collections/{collectionId}/{styleId}/map/{crsId} will be the equivalent of a GetMap in WMS.

/collections/{collectionId}/{styleId}/tile/{tileMatrixSetId}/{tileMatrixId}/{tilerow}/{tilecol}.png will be the equivalent of a GetTile in WMTS.

/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId} will be the equivalent of GetTiles (a draft operation proposed in a past testbed that never became official) that is able to retrieve several tiles for a BBOX at several zoom levels in a zip file of a similar multi-part document.

/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{tileMatrixId} will be almost identical to the previous one, but restricted to a single zoom level (tileMatrixId). This URI template is the perfect operation for returning a MapML file. Since it receives a BBOX as a *query* parameter and a zoom level (tileMatrixId) as a *path* parameter, it is able to return a MapML file. In the process of creating the MapML, the service that would be able to pass the parameters of the current extent and zoom level of the map. Since the service is aware of the URI templates for a single tile, it could include the URI template to retrieve individual templates directly in the MapML file.

An additional parameter is needed to know if the URLs of each tile covering the map extent needed to be explicitly included in a MapML file or a generic URI template for the tiles is desired. A third alternative could be to provide an image (or an image URI template) instead of a tile. Other parameters could limit the extent of the MapML or the zoom levels to a subset of the original available. The MapML could contain TileMatrixSet and style negotiation using the conventions that have been described in this document.

*Table 7. How characteristics of a service layer are represented in MapML documents*

| **Characteristic of the service layer** | **MapML** |
|---|---|
| Styles | Current style as "link rel=self style", and alternative styles as "link rel=style" |
| CRS | As "extent@units" and as alternative CRS as "link rel=projection" |
| extent of the layer | min and max values for the extent’s input@type=location values |
| other zoom levels | as min and max values for the extent’s input@type=zoom values |

*Table 8. How current request parameters are reflected in MapML documents*

| Parameter of the request | MapML |
|---|---|
| Style | If you want the user to set a parameter, use a select. If you want the style to provide alternate styles via the default user agent mechanism (varies by UA), use alternate links "link rel=style" and "link rel=self style". |
| CRS | CRS as "extent@units" |
| BBOX | current value reflected in input[@type=location]/@value attribute in the extent |
| tilematrix | current value reflected in input[@type=zoom]/@value attribute in the extent |

In the 'Appendix A' includes the complete YAML description of the proposed service. Here, the ER provides the fragment of the document that describes the operation that creates MapML documents.

*Fragment of the OpenAPI document describing the "operation" that could generate MapML files from a WMS-WMTS service.*

```
'/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{tileMatrixId}':
  get:
    summary: 'retrieve a set of tiles of the {TileMatrixId} of layer {collectionId}'
    description: >-
      Every tile in a dataset belongs to a layer. A dataset may
      consist of multiple layers. A layer is often a renderization of
      collection of features of a similar type, based on a common schema.\

      Use content negotiation to request a ZIP or a multipart file.
    operationId: getTiles
    tags:
      - Tiles
    parameters:
    - $ref: '#/components/parameters/collectionId'
    - $ref: '#/components/parameters/tileMatrixSetId'
    - $ref: '#/components/parameters/tileMatrixId'
    - $ref: '#/components/parameters/styleId'
    - $ref: '#/components/parameters/bbox'
    - $ref: '#/components/parameters/time'
    responses:
      '200':
        description: >-
          Information about the feature collection plus the tiles
          matching the selection parameters.
        content:
          application/x-zip-compressed:
            schema:
              type: string
              format: binary
          text/mapml:
            schema:
              $ref: '#/components/schemas/MapML'
          multipart/related:
            schema:
              type: string
              format: base64
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
          text/html:
            schema:
              type: string
```

In the previous example can be seen the MapML media type as part of the possible outputs of an HTTP 200

response.

*Fragment of the OpenAPI document describing the MapML document. The description is incomplete but reflects the main structure.*

```
  MapML:
    type: object
    description: A MapML document. The description is partial. Refer to
https://maps4html.github.io/MapML/spec for a complete schema.
    required:
      - body
    properties:
      head:
        description: 'header of the document'
        type: object
        properties:
          title:
            type: 'string'
          link:
            type: 'array'
            items:
              $ref: '#/components/schemas/link'
      body:
        description: 'body of the document'
        type: object
        properties:
          extent:
            description: 'extent of the map'
            type: object
            properties:
              input:
                type: 'array'
                items:
                  type: 'string'
              link:
                type: 'array'
                items:
                  $ref: '#/components/schemas/link'
          feature:
            description: 'features in the map'
            type: object
```

# Appendix A: OpenAPI

This is the OpenAPI document that describes a hypothetical hybrid between a WMS and a WMTS that is able to serve MapML document.

To see a better visualization of it, please, paste it in the left-hand side of this editor https://editor.swagger.io/

*WMS_WMTS.yaml*

```
  openapi: 3.0.1
  info:
    title: A sample API conforming to the OGC Web Map (Tile) Service standard
    version: 0.0.1
    description: >-
      This is a sample OpenAPI definition that conforms to the OGC Web Map (Tile) Service
specification (conformance classes: "Core", "GeoJSON", "HTML" and
      "OpenAPI 3.0").
    contact:
      name: Acme Corporation
      email: info@example.org
      url: 'http://example.org/'
    license:
      name: CC-BY 4.0 license
      url: 'https://creativecommons.org/licenses/by/4.0/'
  servers:
    - url: 'https://dev.example.org/'
      description: Development server
    - url: 'https://data.example.org/'
      description: Production server
  paths:
    '/':
      get:
        summary: landing page of this API
        description: >-
          The landing page provides links to the API definition, the Conformance
          statements and the metadata about the feature data in this dataset.
        operationId: getLandingPage
        tags:
          - Capabilities
        responses:
          '200':
            description: links to the API capabilities
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/root'
              text/html:
                schema:
                  type: string
```

```
    '/conformance':
      get:
        summary: information about standards that this API conforms to
        description: >-
          list all requirements classes specified in a standard (e.g., WFS 3.0
          Part 1: Core) that the server conforms to
        operationId: getRequirementsClasses
        tags:
          - Capabilities
        responses:
          '200':
            description: the URIs of all requirements classes supported by the server
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/req-classes'
          default:
            description: An error occured.
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/exception'
    '/collections':
      get:
        summary: describe the feature collections in the dataset that will be used to create
tiles
        operationId: describeCollections
        tags:
          - Capabilities
        responses:
          '200':
            description: Metadata about the feature collections shared by this API.
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/content'
              text/html:
                schema:
                  type: string
          default:
            description: An error occured.
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/exception'
              text/html:
                schema:
                  type: string
    '/collections/{collectionId}':
      get:
        summary: 'describe the {collectionId} feature collection'
```

```
      operationId: describeCollection
      tags:
        - Capabilities
      parameters:
      - $ref: '#/components/parameters/collectionId'
      responses:
        '200':
          description: 'Metadata about the {collectionId} collection shared by this API.'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/collectionInfo'
            text/html:
              schema:
                type: string
        default:
          description: An error occured.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/exception'
            text/html:
              schema:
                type: string
  '/tileMatrixSets':
    get:
      summary: describe the tilematrixsets in the dataset that layers are tiled
      operationId: describeTileMatrixSets
      tags:
        - Capabilities
      responses:
        '200':
          description: Metadata about the feature collections shared by this API.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/content'
            text/html:
              schema:
                type: string
        default:
          description: An error occured.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/exception'
            text/html:
              schema:
                type: string
  '/tileMatrixSets/{tileMatrixSetId}':
    get:
```

```
      summary: 'describe the {TileMatrixSetId} tileMatrixSet'
      operationId: describeTileMatrixSet
      tags:
        - Capabilities
      parameters:
      - $ref: '#/components/parameters/tileMatrixSetId'
      responses:
        '200':
          description: 'Metadata about the {tileMatrixSetId} shared by this API.'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/tileMatrixSetInfo'
            text/html:
              schema:
                type: string
        default:
          description: An error occured.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/exception'
            text/html:
              schema:
                type: string
  '/collections/{collectionId}/{styleId}/map/{crsId}':
    get:
      summary: 'retrieve a set of tiles of layer {collectionId}'
      description: >-
        Every map server belongs to a layer. A dataset may
        consist of multiple layers. A layer is often a renderization of
        collection of features of a similar type, based on a common schema.\

        Use content negotiation to request a png, jpeg...
      operationId: getMap
      tags:
        - Maps
      parameters:
      - $ref: '#/components/parameters/collectionId'
      - $ref: '#/components/parameters/styleId'
      - $ref: '#/components/parameters/bbox'
      - $ref: '#/components/parameters/time'
      - $ref: '#/components/parameters/crsId'
      - $ref: '#/components/parameters/width'
      - $ref: '#/components/parameters/height'
      responses:
        '200':
          description: >-
            The map portraying the feature collection.
          content:
            image/jpeg:
```

```
            schema:
              type: string
              format: binary
          image/png:
            schema:
              type: string
              format: binary
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
          text/html:
            schema:
              type: string
'/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}':
  get:
    summary: 'retrieve a set of tiles of one of more {tileMatrix} {collectionId}'
    description: >-
      Every tile in a dataset belongs to a layer. A dataset may
      consist of multiple layers. A layer is often a renderization of
      collection of features of a similar type, based on a common schema.\

      Use content negotiation to request a ZIP or a multipart file.
    operationId: getTiles
    tags:
      - Tiles
    parameters:
    - $ref: '#/components/parameters/collectionId'
    - $ref: '#/components/parameters/tileMatrixSetId'
    - $ref: '#/components/parameters/styleId'
    - $ref: '#/components/parameters/tileMatrixIds'
    - $ref: '#/components/parameters/bbox'
    - $ref: '#/components/parameters/time'
    responses:
      '200':
        description: >-
          Information about the feature collection plus the tiles
          matching the selection parameters.
        content:
          application/x-zip-compressed:
            schema:
              type: string
              format: binary
          multipart/related:
            schema:
              type: string
              format: base64
      default:
        description: An error occured.
```

```
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
          text/html:
            schema:
              type: string
'/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{tileMatrixId}':
  get:
    summary: 'retrieve a set of tiles of the {TileMatrixId} of layer {collectionId}'
    description: >-
      Every tile in a dataset belongs to a layer. A dataset may
      consist of multiple layers. A layer is often a renderization of
      collection of features of a similar type, based on a common schema.\

      Use content negotiation to request a ZIP or a multipart file.
    operationId: getTiles
    tags:
      - Tiles
    parameters:
    - $ref: '#/components/parameters/collectionId'
    - $ref: '#/components/parameters/tileMatrixSetId'
    - $ref: '#/components/parameters/tileMatrixId'
    - $ref: '#/components/parameters/styleId'
    - $ref: '#/components/parameters/bbox'
    - $ref: '#/components/parameters/time'
    responses:
      '200':
        description: >-
          Information about the feature collection plus the tiles
          matching the selection parameters.
        content:
          application/x-zip-compressed:
            schema:
              type: string
              format: binary
          text/mapml:
            schema:
              $ref: '#/components/schemas/MapML'
          multipart/related:
            schema:
              type: string
              format: base64
      default:
        description: An error occured.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/exception'
          text/html:
            schema:
```

```
                type: string

'/collections/{collectionId}/{styleId}/tiles/{tileMatrixSetId}/{tileMatrixId}/{tileRow}/{tile
Col}':
      get:
        summary: retrieve a tile; use content negotiation to request PNG or a JPEG
        operationId: getTile
        tags:
          - Tiles
        parameters:
        - $ref: '#/components/parameters/collectionId'
        - $ref: '#/components/parameters/styleId'
        - $ref: '#/components/parameters/tileMatrixSetId'
        - $ref: '#/components/parameters/tileMatrixId'
        - $ref: '#/components/parameters/tileRow'
        - $ref: '#/components/parameters/tileCol'
        responses:
          '200':
            description: A tile.
            content:
              image/jpeg:
                schema:
                  type: string
                  format: binary
              image/png:
                schema:
                  type: string
                  format: binary
          default:
            description: An error occured.
            content:
              application/json:
                schema:
                  $ref: '#/components/schemas/exception'
              text/html:
                schema:
                  type: string
  components:
    parameters:
      bbox:
        name: bbox
        in: query
        description: >
          Only features that have a geometry that intersects the bounding box are selected.
          The bounding box is provided as four or six numbers, depending on whether the
          coordinate reference system includes a vertical axis (elevation or depth):

          * Lower left corner, coordinate axis 1
          * Lower left corner, coordinate axis 2
          * Lower left corner, coordinate axis 3 (optional)
          * Upper right corner, coordinate axis 1
```

```
          * Upper right corner, coordinate axis 2
          * Upper right corner, coordinate axis 3 (optional)

          The coordinate reference system of the values is WGS84 longitude/latitude
          (http://www.opengis.net/def/crs/OGC/1.3/CRS84) unless a different coordinate
          reference system is specified in the parameter `bbox-crs`.

          For WGS84 longitude/latitude the values are in most cases the sequence of
          minimum longitude, minimum latitude, maximum longitude and maximum latitude.
          However, in cases where the box spans the antimeridian the first value
          (west-most box edge) is larger than the third value (east-most box edge).

          If a feature has multiple spatial geometry properties, it is the decision of the
          server whether only a single spatial geometry property is used to determine
          the extent or all relevant geometries.
        required: false
        schema:
          type: array
          minItems: 4
          maxItems: 6
          items:
            type: number
        style: form
        explode: false
      time:
        name: time
        in: query
        description: >-
          Either a date-time or a period string that adheres to RFC 3339. Examples:

          * A date-time: "2018-02-12T23:20:50Z"
          * A period: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z" or "2018-02-
12T00:00:00Z/P1M6DT12H31M12S"

          Only features that have a temporal property that intersects the value of
          `time` are selected.

          If a feature has multiple temporal properties, it is the decision of the
          server whether only a single temporal property is used to determine
          the extent or all relevant temporal properties.
        required: false
        schema:
          type: string
        style: form
        explode: false
      collectionId:
        name: collectionId
        in: path
        required: true
        description: Identifier (name) of a specific collection
        schema:
```

```
          type: string
      styleId:
        name: styleId
        in: path
        required: true
        description: Identifier (name) of a specific style
        schema:
          type: string
      crsId:
        name: crsId
        in: path
        description: Local identifier of a specific CRS
        required: true
        schema:
          type: string
      tileMatrixSetId:
        name: tileMatrixSetId
        in: path
        description: Local identifier of a specific TileMatrixSet
        required: true
        schema:
          type: string
      tileMatrixId:
        name: tileMatrixId
        in: path
        description: Local identifier of a specific TileMatrix. The name of the tile matrix
are limited depending on the collectionId.
        required: true
        schema:
          type: string
      tileMatrixIds:
        name: tileMatrixId
        in: query
        description: Local identifier of a specific TileMatrix. The name of the tile matrix
are limited depending on the collectionId.
        required: false
        schema:
          type: array
          minItems: 1
          items:
            type: string
        style: form
        explode: false
      tileRow:
        name: tileRow
        in: path
        description: Row index of tile matrix. Value between 0 and MatrixHeight-1 of this
tile matrix defined in the ServiceMetadata document.
        required: true
        schema:
          type: integer
```

```
          minimum: 0
      tileCol:
        name: tileCol
        in: path
        description: Column index of tile matrix. Non negative integer type value between 0
and MatrixWidth-1 of this tile matrix defined in the ServiceMetadata document.
        required: true
        schema:
          type: integer
          minimum: 0
      width:
        name: width
        in: path
        description: Size of the map image in pixels along the i axis (that is, width-1 is
the maximum value of i)
        required: true
        schema:
          type: integer
          minimum: 1
      height:
        name: height
        in: path
        description: Size of the map image in pixels along the j axis (that is, height-1 is
the maximum value of j)
        required: true
        schema:
          type: integer
          minimum: 1
    schemas:
      exception:
        type: object
        required:
          - code
        properties:
          code:
            type: string
          description:
            type: string
      root:
        type: object
        required:
          - links
        properties:
          links:
            type: array
            items:
              $ref: '#/components/schemas/link'
            example:
              - href: 'http://data.example.org/'
                rel: self
                type: application/json
```

```
            title: this document
          - href: 'http://data.example.org/api'
            rel: service
            type: application/openapi+json;version=3.0
            title: the API definition
          - href: 'http://data.example.org/conformance'
            rel: conformance
            type: application/json
            title: WFS 3.0 conformance classes implemented by this server
          - href: 'http://data.example.org/collections'
            rel: data
            type: application/json
            title: Metadata about the feature collections
    req-classes:
      type: object
      required:
        - conformsTo
      properties:
        conformsTo:
          type: array
          items:
            type: string
          example:
            - 'http://www.opengis.net/spec/wfs-1/3.0/req/core'
            - 'http://www.opengis.net/spec/wfs-1/3.0/req/oas30'
            - 'http://www.opengis.net/spec/wfs-1/3.0/req/html'
            - 'http://www.opengis.net/spec/wfs-1/3.0/req/geojson'
    link:
      type: object
      required:
        - href
      properties:
        href:
          type: string
        rel:
          type: string
          example: prev
        type:
          type: string
          example: application/geo+json
        hreflang:
          type: string
          example: en
    content:
      type: object
      required:
        - links
        - collections
      properties:
        links:
          type: array
```

```
        items:
          $ref: '#/components/schemas/link'
        example:
          - href: 'http://data.example.org/collections.json'
            rel: self
            type: application/json
            title: this document
          - href: 'http://data.example.org/collections.html'
            rel: alternate
            type: text/html
            title: this document as HTML
          - href: 'http://schemas.example.org/1.0/foobar.xsd'
            rel: describedBy
            type: application/xml
            title: XML schema for Acme Corporation data
      collections:
        type: array
        items:
          $ref: '#/components/schemas/collectionInfo'
  collectionInfo:
    type: object
    required:
      - name
      - links
    properties:
      name:
        description: 'identifier of the collection used, for example, in URIs'
        type: string
        example: buildings
      title:
        description: 'human readable title of the collection'
        type: string
        example: Buildings
      description:
        description: 'a description of the features in the collection'
        type: string
        example: Buildings in the city of Bonn.
      links:
        type: array
        items:
          $ref: '#/components/schemas/link'
        example:
          - href: 'http://data.example.org/collections/buildings/items'
            rel: item
            type: application/geo+json
            title: Buildings
          - href: 'http://example.org/concepts/building.html'
            rel: describedBy
            type: text/html
            title: Feature catalogue for buildings
      extent:
```

```
        $ref: '#/components/schemas/extent'
      tileMatrixSetID:
        description: >-
          The coordinate reference systems in which geometries
          may be retrieved. Coordinate reference systems are identified
          by a URI. The first coordinate reference system is the
          coordinate reference system that is used by default. This
          is always "http://www.opengis.net/def/crs/OGC/1.3/CRS84", i.e.
          WGS84 longitude/latitude.
        type: array
        items:
          type: string
  tileMatrixSetInfo:
    type: object
    description: $$$ Needs to be completelly redone. It is a copy of collectionInfo $$$
    required:
      - name
      - links
    properties:
      name:
        description: 'identifier of the collection used, for example, in URIs'
        type: string
        example: buildings
      title:
        description: 'human readable title of the collection'
        type: string
        example: Buildings
      description:
        description: 'a description of the features in the collection'
        type: string
        example: Buildings in the city of Bonn.
      links:
        type: array
        items:
          $ref: '#/components/schemas/link'
        example:
          - href: 'http://data.example.org/collections/buildings/items'
            rel: item
            type: application/geo+json
            title: Buildings
          - href: 'http://example.org/concepts/building.html'
            rel: describedBy
            type: text/html
            title: Feature catalogue for buildings
      extent:
        $ref: '#/components/schemas/extent'
      tileMatrixSetID:
        description: >-
          The coordinate reference systems in which geometries
          may be retrieved. Coordinate reference systems are identified
          by a URI. The first coordinate reference system is the
```

```
              coordinate reference system that is used by default. This
              is always "http://www.opengis.net/def/crs/OGC/1.3/CRS84", i.e.
              WGS84 longitude/latitude.
            type: array
            items:
              type: string
      MapML:
        type: object
        description: A MapML document. The description is partial. Refer to
https://maps4html.github.io/MapML/spec for a complete schema.
        required:
          - body
        properties:
          head:
            description: 'header of the document'
            type: object
            properties:
              title:
                type: 'string'
              link:
                type: 'array'
                items:
                  $ref: '#/components/schemas/link'
          body:
            description: 'body of the document'
            type: object
            properties:
              extent:
                description: 'extent of the map'
                type: object
                properties:
                  input:
                    type: 'array'
                    items:
                      type: 'string'
                  link:
                    type: 'array'
                    items:
                      $ref: '#/components/schemas/link'
              feature:
                description: 'features in the map'
                type: object
      extent:
        type: object
        properties:
          crs:
            description: >-
              Coordinate reference system of the coordinates in the spatial extent (property
`spatial`).
              In the Core, only WGS84 longitude/latitude is supported. Extensions may support
additional
```

```
              coordinate reference systems.
            type: string
            enum:
              - 'http://www.opengis.net/def/crs/OGC/1.3/CRS84'
            default: 'http://www.opengis.net/def/crs/OGC/1.3/CRS84'
          spatial:
            description: >-
              West, north, east, south edges of the spatial extent. The minimum and
              maximum values apply to the coordinate reference system WGS84
longitude/latitude
              that is supported in the Core. If, for example, a projected coordinate
reference
              system is used, the minimum and maximum values need to be adjusted.
            type: array
            minItems: 4
            maxItems: 6
            items:
              type: number
            example:
              - -180
              - -90
              - 180
              - 90
          trs:
            description: >-
              Temporal reference system of the coordinates in the temporal extent (property
`temporal`).
              In the Core, only the Gregorian calendar is supported. Extensions may support
additional
              temporal reference systems.
            type: string
            enum:
              - 'http://www.opengis.net/def/uom/ISO-8601/0/Gregorian'
            default: 'http://www.opengis.net/def/uom/ISO-8601/0/Gregorian'
          temporal:
            description: Begin and end times of the temporal extent.
            type: array
            minItems: 2
            maxItems: 2
            items:
              type: string
              format: dateTime
            example:
              - '2011-11-11T12:22:11Z'
              - '2012-11-24T12:32:43Z'
  tags:
    - name: Capabilities
      description: >-
        Essential characteristics of this API including information about the
        data.
    - name: Maps
```

```
    description: >-
      Access to the renderization of data (maps).
  - name: Tiles
    description: >-
      Access to the renderization of data (tiles).
```

# Appendix B: Revision History

*Table 9. Revision History*

| Date | Editor | Release | Primary clauses modified | Descriptions |
|---|---|---|---|---|
| June 15, 2018 | Joan Masó | .1 | all | initial version |
| September 15, 2018 | Gil Heo | .2 | [implementations] | section about an implementation integrated |
| September 22, 2018 | Keith Pomakis | .3 | [implementations] | section about an implementation integrated |
| October 1, 2018 | Joan Maso | 1.0 | various | clean up |
| October 29, 2018 | Joan Maso | 1.0 | various | last minute section about web services included |

# Appendix C: Bibliography


1. Rushforth, P.: Map Markup Language, http://maps4html.github.io/MapML/spec/, (2018).
2. Maso, J.: OGC Testbed-13: MapML Engineering Report. OGC 17-019,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-019.html (2018).
3. Tandy, J., Brink, L. van den, Barnaghi, P.: Spatial Data on the Web Best Practices. OGC 15-107,World Wide Web Consortium, https://www.w3.org/TR/2017/NOTE-sdw-bp-20170928/ (2017).
4. Maso, J.: Testbed-12 OWS Context: JSON, JSON-LD and HTML5 ER. OGC 16-053r1,Open Geospatial Consortium, http://docs.opengeospatial.org/per/16-053r1.html (2018).