OGC Testbed-13

MapML ER

Table of Contents

Publication Date: 2018-01-11

Approval Date: 2017-12-07

Posted Date: 2017-11-14

Reference number of this document: OGC 17-019

Reference URL for this document: http://www.opengis.net/doc/PER/t13-NR002

Category: Public Engineering Report

Editor: Joan Maso

Title: OGC Testbed-13: MapML ER

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 Engineering Report discusses the approach of Map Markup Language (MapML) and Map for HyperText Markup Language (Map4HTML) described in: https://github.com/Maps4HTML and supported by the community in https://www.w3.org/community/maps4html/. The objective of MapML is to define a hypermedia type for geospatial maps on the web that can be embedded in HyperText Markup Language (HTML) pages. MapML is needed because while Web browsers implement HTML and Scalable Vector Graphics (SVG), including the <map> element, those implementations do not meet the requirements of the broader Web mapping community. The semantics of the HTML map element are incomplete or insufficient relative to modern Web maps and mapping in general. Currently, robust web maps are implemented by a variety of non-standard technologies. Web maps do not work without script support, making their creation a job beyond the realm of beginners' skill sets. In order to improve collaboration and integration of the mapping and Web communities, it is desirable to enhance or augment the functionality of the <map> element in HTML to include the accessible user interface functions of modern web maps (e.g. panning, zooming, searching for, and zooming to, styling, identifying features' properties, etc.), while maintaining a simple, declarative, accessible interface for HTML authors.

The objective of this Engineering Report is to explore how MapML can be harmonized with the OGC standards mainstream and contribute to the progress of the specification avoiding unnecessary duplication. In particular, the ER proposes Web Map Service (WMS) or Web Map Tile Service (WMTS) as services that can be used to deliver MapML documents with small modifications.

Another consideration on the ER is the inclusion of the time dimension and directions operation in MapML.

1.1. Requirements

The following requirements are being addressed by the MapML Engineering Report:

- Specify how the media type "text/mapml" (pending Internet Assigned Numbers Authority, IANA, registration) can interoperate and encapsulate the semantics of maps to support the stateless client-server requirements of Web browsers. Specific configurations wrapping existing OGC services (WMS, Web Feature Service (WFS), WMTS, etc)
- Collect OGC community feedback about:
  - TileMatrixSet (also called in the CFP Tiled Coordinate Reference System (TCRS)) definitions
  - Image georeferencing markup
  - TCRS/projection negotiation
  - Language negotiation
  - Security considerations
  - Hyperlinking within and between map services
  - Accessibility considerations for map feature markup
  - Microdata / microformat semantic markup recommendations
  - Caching discussion

- Feature styling with Cascading Style Sheets

1.2. Prior-After Comparison

OGC has not explored an approach like this before. This could bring the maps closer to the W3C community.

1.3. What does this ER mean for the Working Group and OGC in general

This work is relevant to the OWS Context standards Working Group. Both OWS Context and MapML share the objective of being able to distribute collections of geospatial resources mainly in the web. An OWS Context file could be the seed for a HTML page that contains a map representing the information in an OWS Context document in the form of MapML documents.

1.4. Document contributor contact points

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

Table 1. Contacts

Name	Organization
Joan Masó	UAB-CREAF

1.5. Future Work

This document contains several recommendations for future testbeds, OGC standards and MapML. A future version of the MapML language needs to be harmonized with OGC language and should use the OGC concepts extracted from WMTS. A future version of MapML should consider changing the encoding for features into a pure HTML encoding complemented by schema.org semantics. The WMTS standard needs to be divided into two documents, one more abstract describing the tiled reference systems, and another one describing the WMTS service. A future WMS or a WMTS extension with a new operation that is able to produce MapML documents could also be produced.

1.6. 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.

1.7. Use of notes in this document

The use of notes in this document is used to emphasize some paragraphs targeting different kind of readers. It follows the following criteria bellow:

CAUTION	This is a note indicating that there is a problem with the aspect discussed that needs attention. E.g. a part of the MapML current document that has not been implemented and that can be deprecated in the next version.
NOTE	A note. If the note starts with *Recommendation to new testbeds:* this means that Testbed 14 sponsors (or other Interoperability experiments) can find aspects of MapML that can be part of a more careful examination and experimentation.
IMPORTANT	An important aspect to be considered in a standards. It will start with the text *Recommendation to xxxx:* where xxxx is the name of the standard affected.

Chapter 2. References

The following normative documents are referenced in this document.

- MicroXML [https://dvcs.w3.org/hg/microxml/raw-file/tip/spec/microxml.html]
- The <web-map> HTML Element proposal [https://maps4html.github.io/HTML-Map-Element/spec/]
- Map Markup Language [https://maps4html.github.io/MapML/spec/]
- IETF RFC 3875, The Common Gateway Interface (CGI) Version 1.1 [https://tools.ietf.org/html/rfc3875]
- W3C HTML 4.01 Specification 17 Forms [http://www.w3.org/TR/html401/interact/forms.html]
- OGC 17-083, OGC Tile Matrix Set Standard (draft) [https://portal.opengeospatial.org/files/?artifact_id=75348] (at requires membership (observer is enough) to WMS.SWG to get access to the document)

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] and the future OGC 17-083 shall apply. In addition, the following terms and definitions apply.

3.1. coordinate system

```
set of mathematical rules for specifying how coordinates are to be assigned to points
(WMTS 1.0)
```

3.2. extent

```
the bounding box of the view in the screen
```

3.3. map

```
portrayal of geographic information as a digital image file suitable for display on a
computer screen (OGC 06-042: WMS 1.3)
```

3.4. tile

```
a rectangular pictorial representation of geographic data, often part of a set of such
elements, covering a spatially contiguous extent and sharing similar information
content and graphical styling covering a cell of the grid defined by a tile matrix on
top of a denser raster grid.
```

3.5. tile coordinates

```
a discrete coordinate system where a position is uniquely defined by a pair of integer
indices for the column and row in the tile matrix, a pair of integer indices inside
the tile, a long with an identifier for the tile matrix.
```

3.6. tile matrix

```
a regular grid defined on top of a more detailed raster regular grid that groups
matrices of cells into tiles. Also, all tiles corresponding to a fixed scale.
```

3.7. tile matrix coordinates

a discrete coordinate system where a position is uniquely defined by a pair of integer indices in the denser raster grid where the tile matrix is defined (starting in the top left corner) a long with an identifier for the tile matrix.

3.8. tile matrix set

a collection of tile matrices defined at different scales

Chapter 4. Abbreviated terms

- API Application Programming Interface
- ER Engineering Report
- HTML HyperText Markup Language
- HTTP HyperText Transport Protocol
- KVP Key and Value Pair
- Map4HTML Map for HyperText Markup Language
- MapML Map Markup Language
- OWS OGC Web Services
- SVG Scalable Vector Graphics
- TCRS Tiled Coordinate Reference System
- WFS Web Feature Service
- WMS Web Map Service
- WMTS Web Map Tile Service
- XML Extendable Markup Language

Chapter 5. Overview

We can classify the standards depending on the level of complexity the developer needs to face. In the modern era of Internet distributed applications, developers tend to prefer standards that hide the complexities of the protocols involved that allow them to concentrate on user interaction. This results in a proliferation of Application Programming Interfaces. But there is still another level of simplification where you can concentrate in describing and tagging the content and assume a default user interaction. This has been the strategy and the success of HTML. In summary, depending on the level of complexity they expose, we can separate standards in three groups:

- OGC Web Services
  - They describe how clients and services exchange messages (and information) based on a distributed architecture and a protocol.
  - e.g. Web Map Tile Service
- Application Programming Interface (API)
  - Describes a set of operations or functions that allow encoding functionality on top of a programming language and programming libraries. APIs hide the protocol and focus on facilitating tools for encoding user interfaces.
  - e.g. Google Maps API
- Document standards
  - Describe a data encoding including raw information and some clues about the preferred styling for presentation. Presenting the information to the user implies that there should be an engine that is able to parse the document and take specific and appropriate actions to build the user interface. The developer can concentrate on the content and assume default behaviors of common user interfaces.
  - e.g. KML, MapML

Traditionally, the OGC has concentrated its efforts on standards that deal with OGC Web services (1st level) and on document standards (3rd level), while companies are offering API solutions (2nd level) that, at the moment, are mainly not interoperable. This Engineering Report focuses on the document level (3rd level) for map visualization. Its objective is to examine the MapML approach as a new MIME type to distribute maps over the Internet that can be embedded in web pages without any immediate need for JavaScript code, provided that web browsers implement support for this document type. It should be mentioned that elimination of JavaScript from mapping is a non-objective. In the short to medium term, JavaScript will be necessary to 'simulate' native support for extended map semantics, through the Custom Elements candidate standard.

5.1. MapML use cases

Hyperlinks between and within services: MapML should allow service-level links such that service providers can link together / federate to provide apparently seamless spatial coverage.

Feature identification: If a MapML document includes features, the MapML author should be able to markup any or all of those features in a way which lends itself to feature identification and

property display.

Attribution: MapML documents encode citation information on a per-request basis, making it available as markup in the response.

The following usage scenarios illustrate some of the ways in which Map Markup Language might be used for various applications.

5.1.1. Embedding a map in a web page

A map can currently be embedded in a web page by means of the HTML <map> element. However, such a map is essentially static: it allows the HTML author to draw hyperlinks over shapes on a static image. Maps should allow us to interact with map content in a way that is more interesting. To be able to do this, a new child element is introduced to the <map> element in the form of the <layer> element. A <layer> element allows to point to a MapML document describing content of a map layer. Multiple layers can be included in a single map, each by means of its own URL. A similar construct is used for the HTML <video> element, which can have one or more source files included as child <source> elements. (Of course, with a video, the most appropriate source file is chosen from the child <source> elements. In contrast, a <map> element's child <layer> elements are stacked and overlayed together as a group.)

5.1.2. Story maps and story telling

MapML could be used as a story maps mechanism where the user navigates progressively through the content and learns by reading a combination of text and map scenes. In story maps the narrative flow is linear but the user can play with the maps presented and verify the textual description that accompanies the map. Commonly, the story is presented as a single page that can be scrolled up and down with maps that are controlled by the text shown or text fragments each one accompanied by its respective map. To make story telling possible, more than one map needs to be supported in a single page.

An example of story maps can be shown here: https://grid-arendal.maps.arcgis.com/apps/Cascade/index.html?appid=54b862ef079445d9a99b8f6249249e4d

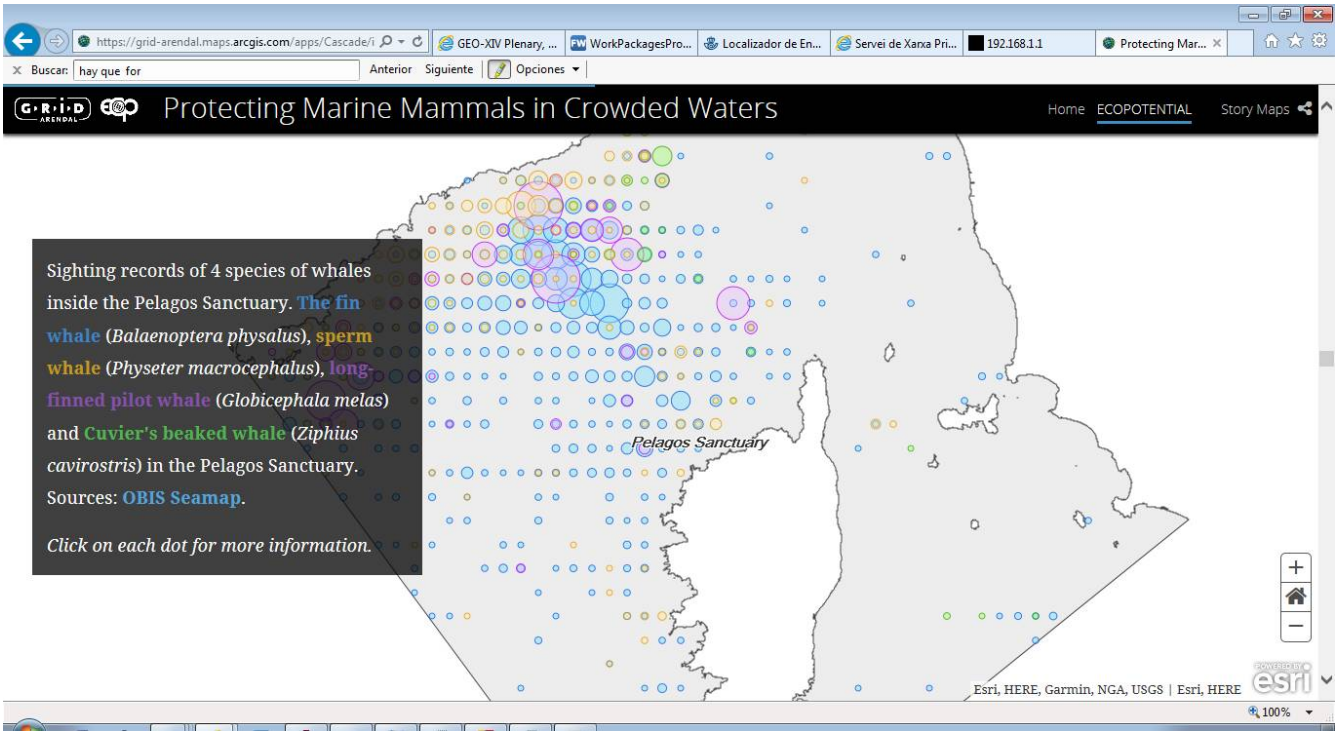


Figure 1. Scene 1 of the ECOPotential Protecting Marine Mammals in Crowded Waters Storymap

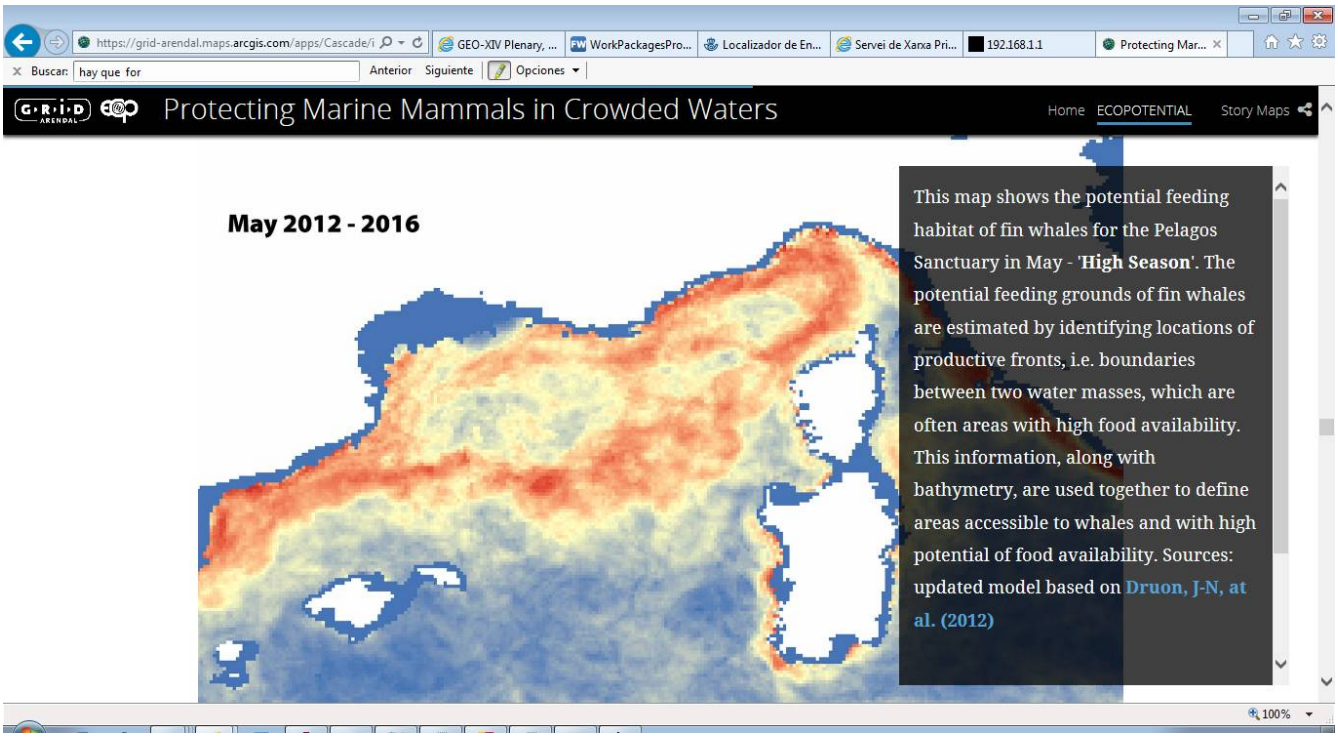


Figure 2. Scene 2 of the ECOPotential Protecting Marine Mammals in Crowded Waters Storymap

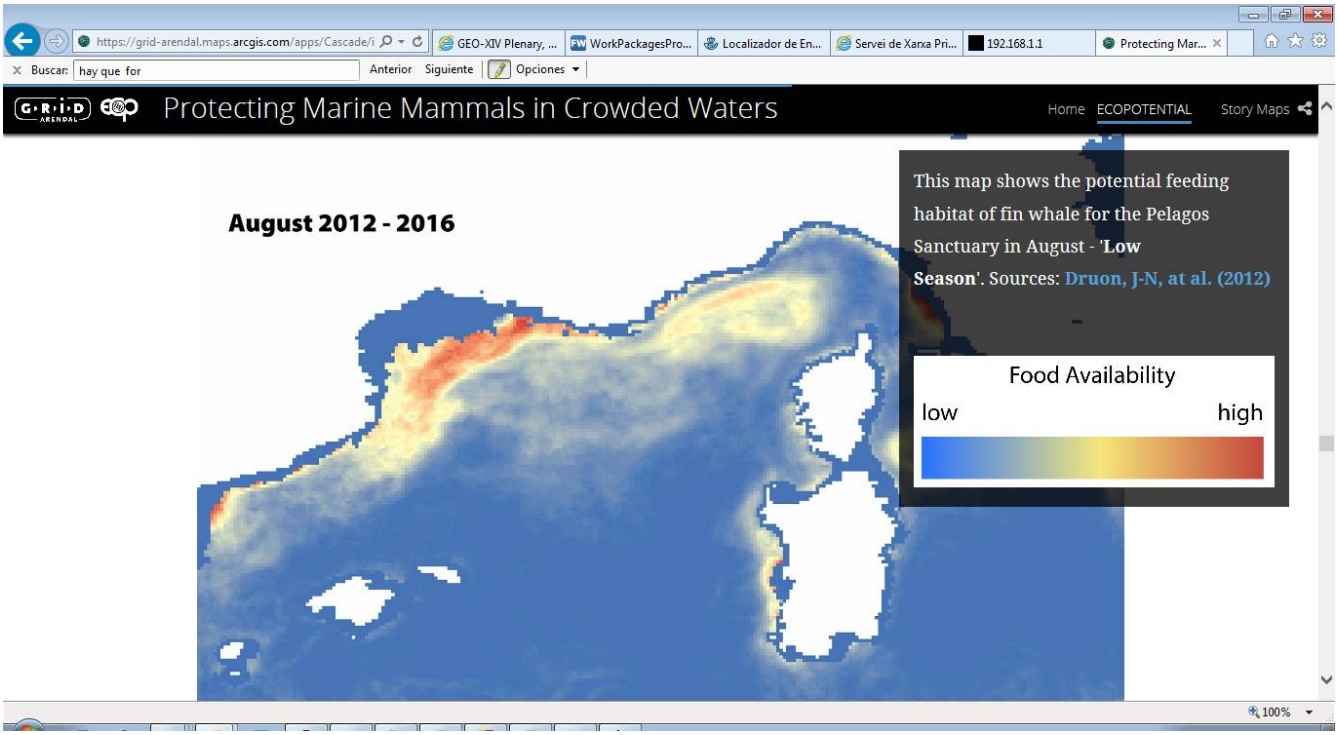


Figure 3. Scene 3 of the ECOPotential Protecting Marine Mammals in Crowded Waters Storymap

5.2. MapML data model

A MapML document is similar to an HTML document in that it consists of 2 main blocks: the <head> and <body> elements. Within the <body>, there are four spatial content types: the extent form, tiles or images, and features.

- **Extent** defines the area the MapML layer shows and the area it eventually can cover. It also contains a mechanism to describe the action needed to send to the server to get a new MapML in response to a user action.
- **Tile** describes the tiles (rendered raster fragments) necessary to cover the area that extent defines.
- **Image** describes the location and orientation of the upper left corner of a geo-referenced image.
- **Feature** describes a collection of features (vector) that cover the area that extent defines.

The following UML diagram describes the structure of a MapML document in the left-hand side. The right-hand side describes how to include a MapML document in an HTML file, and will be explained below.

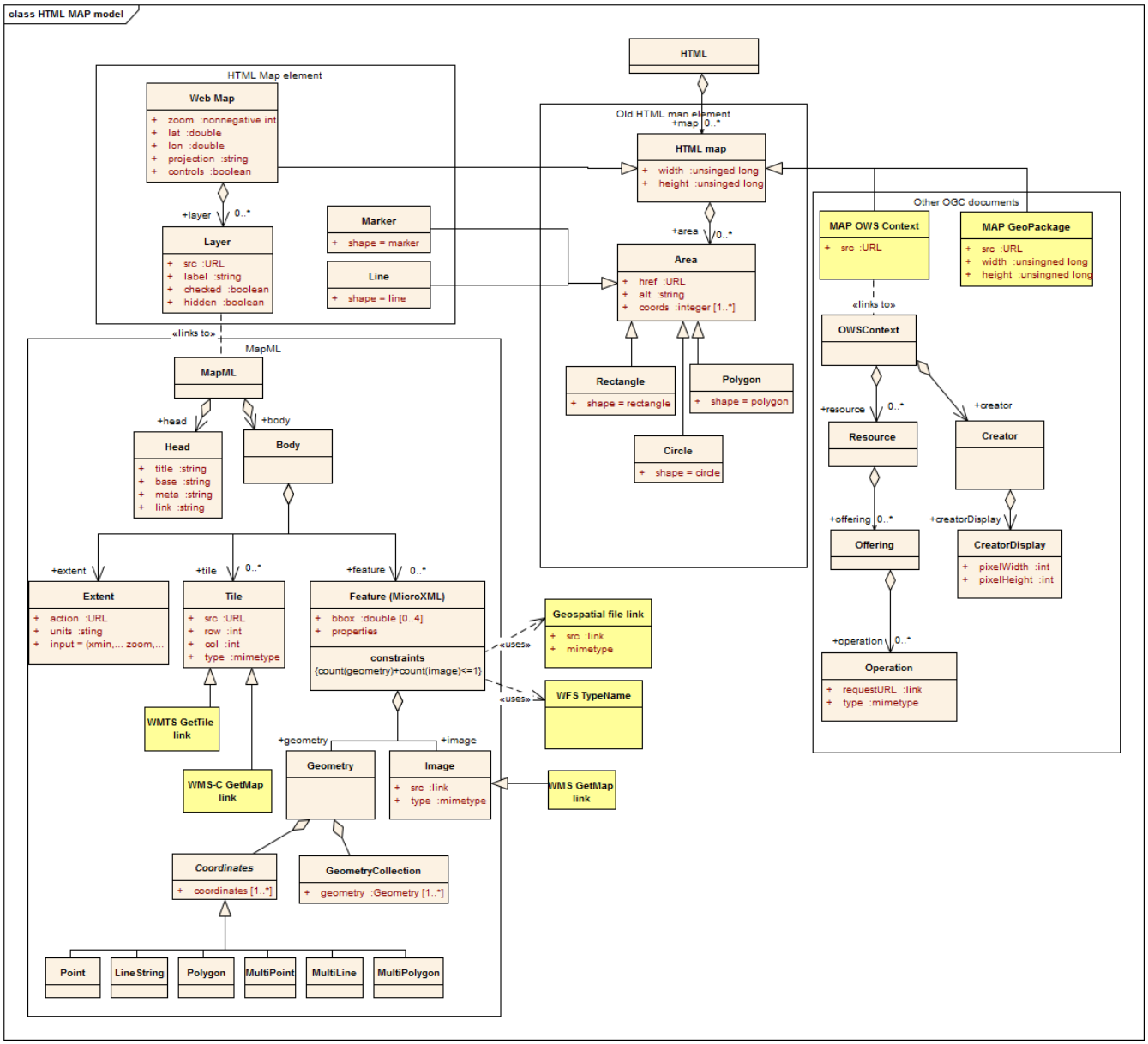


Figure 4. HTML and MapML data model UML class diagram

In order for a map to be 'dynamic', it requires a server that creates MapML documents in response to the user actions such as panning, zooming or textual searches. This MapML server is typically implemented as a facade to OGC and non-OGC services (see MapML services and navigation modes in this document for more details). This way MapML hides the complexity of the underlying implementations and translates their content to a set of elements that the web browser client can easily draw and presents them to the user.

The next clause MapML encoding presents the details of the elements of MapML, how it is composed, and discusses possible alternatives and improvements. MapML services and navigation modes presents the particularities of MapML services. TileMatrixSet specification clause presents a summary of a new standard candidate (OGC 17-083) that was extracted from WMTS 1.0. This document describes TileMatrixSets and that can be used as the bases for other standards such as GeoPackage, WMTS and MapML.

5.3. MapML in practice

MapML is a document type for HTML ingestion of maps. This means that a MapML document has to

be included in a HTML page in order to be visualized.

5.3.1. Including a MapML in a HTML page

This standard starts by proposing an extension to the old <map> element that represented a map browser that starts in a *lat,lon* position in a *projection* (that is actually a tiled CRS: a tile matrix set) at a given *zoom* level. Apart from the old *area* elements for descended compatibility to old browsers, it has a new element type that is called *layer* that references one or more MapML files.

```
<!DOCTYPE html>
   <head>
     <title>Web Map Template - default style</title>
     <script src=
"http://geogratis.gc.ca/mapml/client/bower_components/webcomponentsjs/webcomponents-
lite.min.js"></script>
     <link rel="import" href=
"http://geogratis.gc.ca/mapml/client/bower_components/web-map/web-map.html">
   </head>
   <body>
      <map is="web-map" projection="OSMTILE" zoom="3" lat="60.5005254" lon="
-98.2617190" width="900" height="400" controls>
          <layer- label="Canada Base Map - Transportation (CBMT)" src=
"http://geogratis.gc.ca/mapml/en/osmtile/cbmt/" checked />
      </map>
   </body>
</html>
```

The source of the example can be found here: http://geogratis.gc.ca/mapml/en/osmtile/cbmt/

This is all you have to do as a web designer to include a map in your web page.

5.3.2. A MapML file example

Behind an easy to do link "http://geogratis.gc.ca/mapml/en/osmtile/cbmt/" in the previous example there is the hidden complexity of the MapML implementation.

Let us start to say that in the previous *layer* section there is a *src* link to the http://geogratis.gc.ca/mapml/en/osmtile/cbmt/ URL that is actually a MapML MIME type (text/mapml) that looks like this:

```
<?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="256.0" min="0.0" name="xmin" type="xmin"/>
      <input max="256.0" min="0.0" name="ymin" type="ymin"/>
      <input max="256.0" min="0.0" name="xmax" type="xmax"/>
      <input max="256.0" min="0.0" name="ymax" type="ymax"/>
      <input max="15" min="0" name="zoom" type="zoom" value="0"/>
      <input name="projection" type="projection" value="OSMTILE"/>
    </extent>
  </body>
</mapml>
```

Once the MapML engine gets a file, it determines the properties of the extent of the layer and the minimum, maximum zoom and the CRS projection. Then it goes back to the <map> section in the HTML page to calculate the real extent (*xmin*, *xmax*, *ymin*, *ymax*) that corresponds to the zoom="3" lat="60.5005254" lon="-98.2617190" that covers the width="900" and height="400". The result of this calculation is not done in CRS coordinates but in the tilematrix *zoom* 3 pixel coordinates (the so called "tilematrix" coordinates) that starts at the origin of the time matrix (Top-left corner). It virtually inputs it in the *xmin*, *xmax*, *ymin*, *ymax* and *zoom* values and submits the *extent* in the same way that a *form* is submitted. Since the method is *GET* and the enctype is x-www-form-urlencoded, the engine builds a URL with the form element name and values pairs and sends it to the server. In this case the request is: http://geogratis.gc.ca/mapml/en/osmtile/cbmt/?xmin=-104&ymin=389&xmax=1034&ymax=789&zoom=3&projection=OSMTILE

Please note that coordinates provided in the URL and in the responding MapML are also in tile matrix coordinates.

The return of this request contains a list of the URLs of all tiles that need to be requested to compose the map visualization area:

```
<?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
Â© Natural Resources Canada"/>
  </head>
  <body>
    <extent action="/mapml/en/osmtile/cbmt/" enctype="application/x-www-form-
urlencoded" method="get" units="OSMTILE">
      <input max="2048.0" min="0.0" name="xmin" type="xmin" value="-104.0"/>
      <input max="2048.0" min="0.0" name="ymin" type="ymin" value="389.0"/>
      <input max="2048.0" min="0.0" name="xmax" type="xmax" value="1034.0"/>
      <input max="2048.0" min="0.0" name="ymax" type="ymax" value="789.0"/>
      <input max="15" min="0" name="zoom" type="zoom" value="3"/>
      <input name="projection" type="projection" value="OSMTILE"/>
    </extent>
    <tile col="1" row="2" src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/2/1?m4h=t"/>
    <tile col="2" row="2" src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/2/2?m4h=t"/>
    <tile col="1" row="1" src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapSe
rver/tile/3/1/1?m4h=t"/>
...
    <tile col="4" row="1" src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_TXT_3857/MapServer/t
ile/3/1/4?m4h=t"/>
    <tile col="4" row="3" src=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_TXT_3857/MapServer/t
ile/3/3/4?m4h=t"/>
  </body>
</mapml>
```

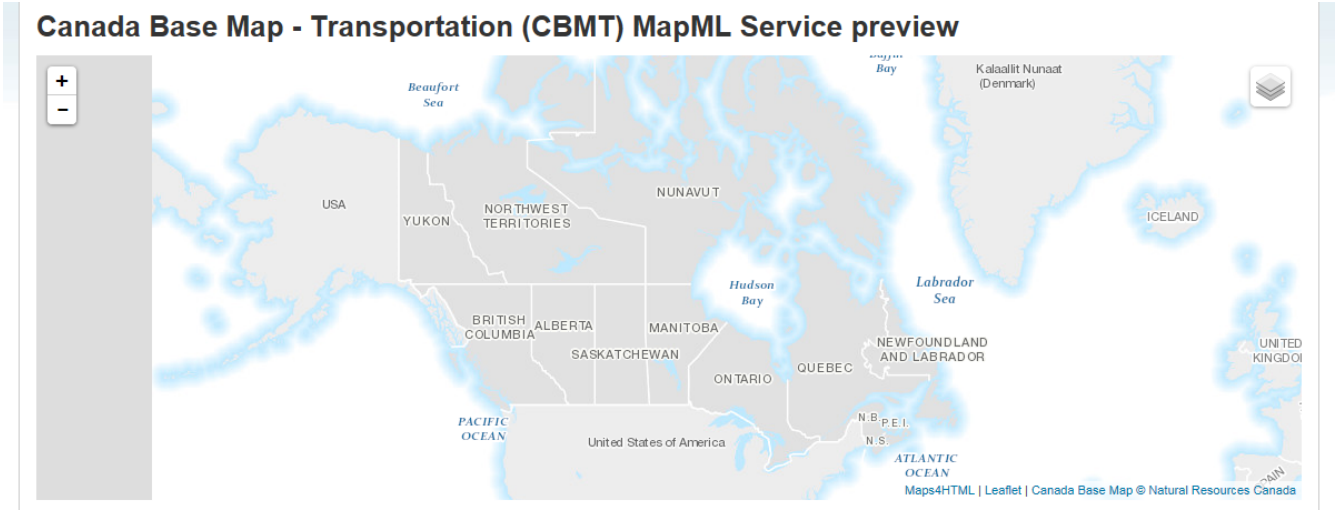


Figure 5. Visual result of the HTML page with the map

On the server side, there is an application that is producing these MapML responses. The application identifies the CBMT layer and knows about the ArcGIS web tile service that is able to provide the needed tiles. In this case we discover that the layer is actually associated with two datasets in an ArcGIS rest service: CBMT_TXT_3857 and CBMT_CBCT_GEOM_3857 so that, they will be presented as overlayer datasets in the map browser. Two URL examples are: https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_CBCT_GEOM_3857/MapServer/tile/3/2/2?m4h=t and https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT_TXT_3857/MapServer/tile/3/2/2?m4h=t

Figure 6. One of the tiles corresponding to the CBMT_CBCT_GEOM_3857 dataset:

Figure 7. The same tile indices but corresponding to the CBMT_TXT_3857 dataset:

Next time the user makes an action on the map, a new Key-Value Pair (KVP) request will be sent to the server with the new bounding box coordinates. For example a pan of only a few kilometers to the left hand side will result in the following request: http://geogratis.gc.ca/mapml/en/osmtile/cbmt/?xmin=-310&ymin=389&xmax=828&ymax=789&zoom=3&projection=OSMTILE that results in a very similar MapML response that is different for the previous on in the extent section and in the list of tiles URL it contains.

```
    <extent action="/mapml/en/osmtile/cbmt/" enctype="application/x-www-form-
urlencoded" method="get" units="OSMTILE">
      <input max="2048.0" min="0.0" name="xmin" type="xmin" value="-310.0"/>
      <input max="2048.0" min="0.0" name="ymin" type="ymin" value="389.0"/>
      <input max="2048.0" min="0.0" name="xmax" type="xmax" value="828.0"/>
      <input max="2048.0" min="0.0" name="ymax" type="ymax" value="789.0"/>
      <input max="15" min="0" name="zoom" type="zoom" value="3"/>
      <input name="projection" type="projection" value="OSMTILE"/>
    </extent>
```

Since the references to the tiles (not reproduced in the last source code sample) correspond to the same URLs, the map browser takes advantage of the cache and does not request any additional tile. The map browser only portrays the same tiles in a different position in the screen.

The presented example only contains tile references but it could eventually contain features encoded in an XML encoding that has the same restrictions and capabilities that GeoJSON has. This capacity is defined in the MapML current specification but its implementation is very limited.

5.4. MapML and HATEOAS

The key aspect to understand how MapML works is to realize that MapML is designed to exchange the state of the map browser application. In that sense it respects the HTTP design and implements the **Hypermedia as the Engine of Application State** (**HATEOAS**) for maps. MapML transports the initial status and uses the HTML KVP <form> mechanism to exchange the status of the map window (consisting in layer name, bounding box, zoom level and projection) to a service that returns the status back to the client with the additional information to update the map. MapML becomes the

Hypermedia for maps.

In the next section we will discuss about the MapML format in more detail and we will formulate some suggestions. After that, we will present some details on the MapML services and its relation with OGC services.

Chapter 6. MapML encoding

For the moment, and if we take a look at the current MapML specification, there are four big classes that are specified:

- The meta section: provides static metadata
- The extent section: controls the view
- The tiles section: provides a mechanism for raster or rendered data to be delivered
- The features section: provides a mechanism for vector data or annotations to be delivered

The current specification provides a solution for encoding all of them, even if only the first three have been fully tested in real implementations so far (including the activities in Testbed-13). In other words, at this point, to propose a complete redefinition of the extent and tiles sections would not be well received by the MapML community due to the number of implemented instances already existing. There is still room for proposing radical alternatives for the features section.

All positions in a MapML file are provided in tile matrix coordinates and refer to a single tile matrix (zoom level) specified in the extent section.

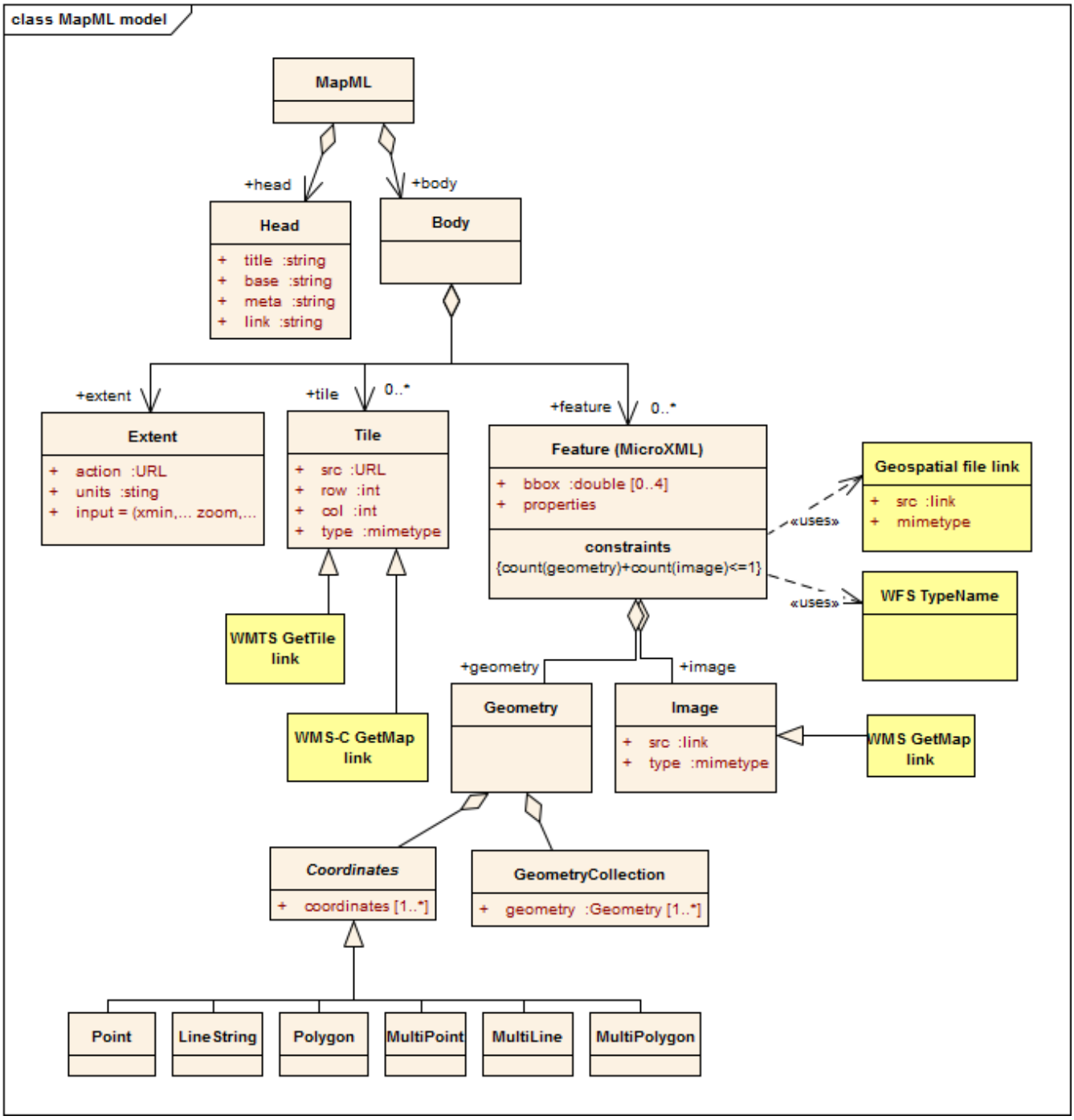


Figure 8. MapML data model UML class diagram

6.1. Properties of a MapML document

The <head> element of a MapML document is used to communicate properties of the document to the client. As such, it is important to include the required <title> element with a meaningful value describing the map resource in hand. As well, it is possible and desirable to include certain <meta> element values which describe the intended scope of usage of the content of the document, for example the zoom range in which the representation should be displayed, or the geographic extent in which the content exists. The user agent can then decide what the appropriate user interface clues are that may be communicated to the user (for example, disabling the layer representation in the layer control is one such possible clue).

6.2. meta content document properties

The model of the <meta> element in MapML is borrowed from HTML; it should have meta@name or meta@http-equiv and meta@content attributes containing the name of the document property, and the value of that property, respectively. The HTML <meta> element specifies extensive functionality that is not currently considered in-scope for MapML. What follows here is recommended or required usage of the MapML <meta> element.

The meta@http-equiv=content-type value identifies the meta@content value of the document, which is proposed to be text/mapml, pending IANA registration. As such, optional media type parameters "projection" and "zoom" may be specified to better describe the content of the document, for example: <meta http-equiv="Content-Type" content="text/mapml;projection=CBMTILE;zoom=2"/>.

The meta@name=projection property identifies the MapML projection (tile matrix set) of the current document. A MapML document can have zero or one instances of such a meta property, and the set of possible meta@content values is given by the MapML specification. When the document has content, but does not carry a meta[@name=projection] element or otherwise identify the projection, the default meta[@name=projection]/@content value is "OSMTILE".

The meta@name=zoom property identifies the range of zoom levels at which the document should be displayed. Typically, when a document is generated by a service, the meta[@name=zoom]/@content value will be a single integer value of 0 or greater, within the range defined by the MapML projection. When the document is static or otherwise applicable across a range of zoom levels, the meta[@name=zoom]/@content value can contain a range of values in the form of a pair of comma-separated term of the form "min=n,max=m" where -1<n<m and n and m are within the range of zooms specified by the MapML specification for the document's projection property.

The meta@name=extent property identifies the bounding box within which the document content should or can be displayed. When interacting with a MapML service over a datastore, typically the service will emit an explicit <extent> form element in the body of the document, which specifies the nominal bounding box of the document via input@type=xmin|ymin|xmax|ymax values. In such a case it is recommended that no meta@name=extent element is included in the document. However, where the document is a static resource, it is recommended to include a meta[@name=extent]/@content=xmin=number,ymin=number,xmax=number,ymax=number bounding box so that the user agent can decide what the bounding box of the content should be without having to compute it.

6.3. Map content licensing

Perhaps the key strength of the Web is the hyperlink. To this end, MapML documents are recommended to contain <link> elements in the <head> content, with specific @rel attribute values.

Most importantly for publicly consumable map content is the link@rel=license as explained in the section 4.2.1.1.2.4 of the current version of MapML. It is recommended that the user agent dedicates a specific and reliable user interface for such links, such as the "Attribution control" in the bottom right-hand corner of the map. It is also recommended that such a control be present in all maps,

such that it cannot be turned off by the HTML author (i.e. is presented regardless of the map@controls state).

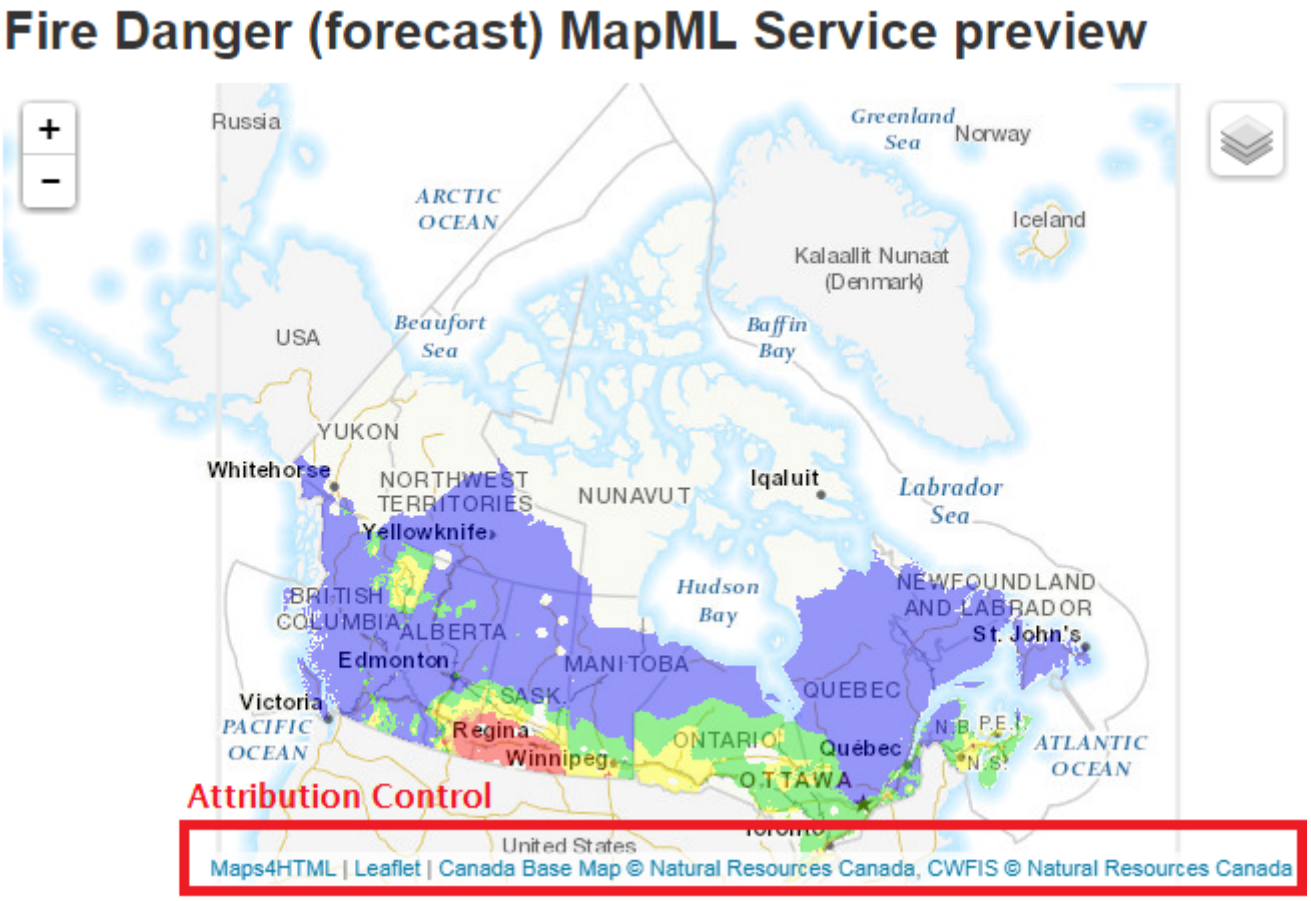


Figure 9. <map> element attribution control

If the MapML content is provided in conjunction with a WMS, it can contain the tag "Attribution" for this layer in the GetCapabilities document.

Do not get confused by the HTML4MAP <layer label=""> element that it actually used to populate the text in the "layer control" panel of the map combination.

6.4. Legends

A map layer can have one or more possible legend representations. It is recommended that a MapML document include one or more links to legends for the information contained in the map. Such links should be represented by link elements with the link@rel=legend link relation. Such links can be afforded by the client in a variety of ways. It is recommended that the user agent provide a default mechanism such as hyperlink anchors from the standard layer control.

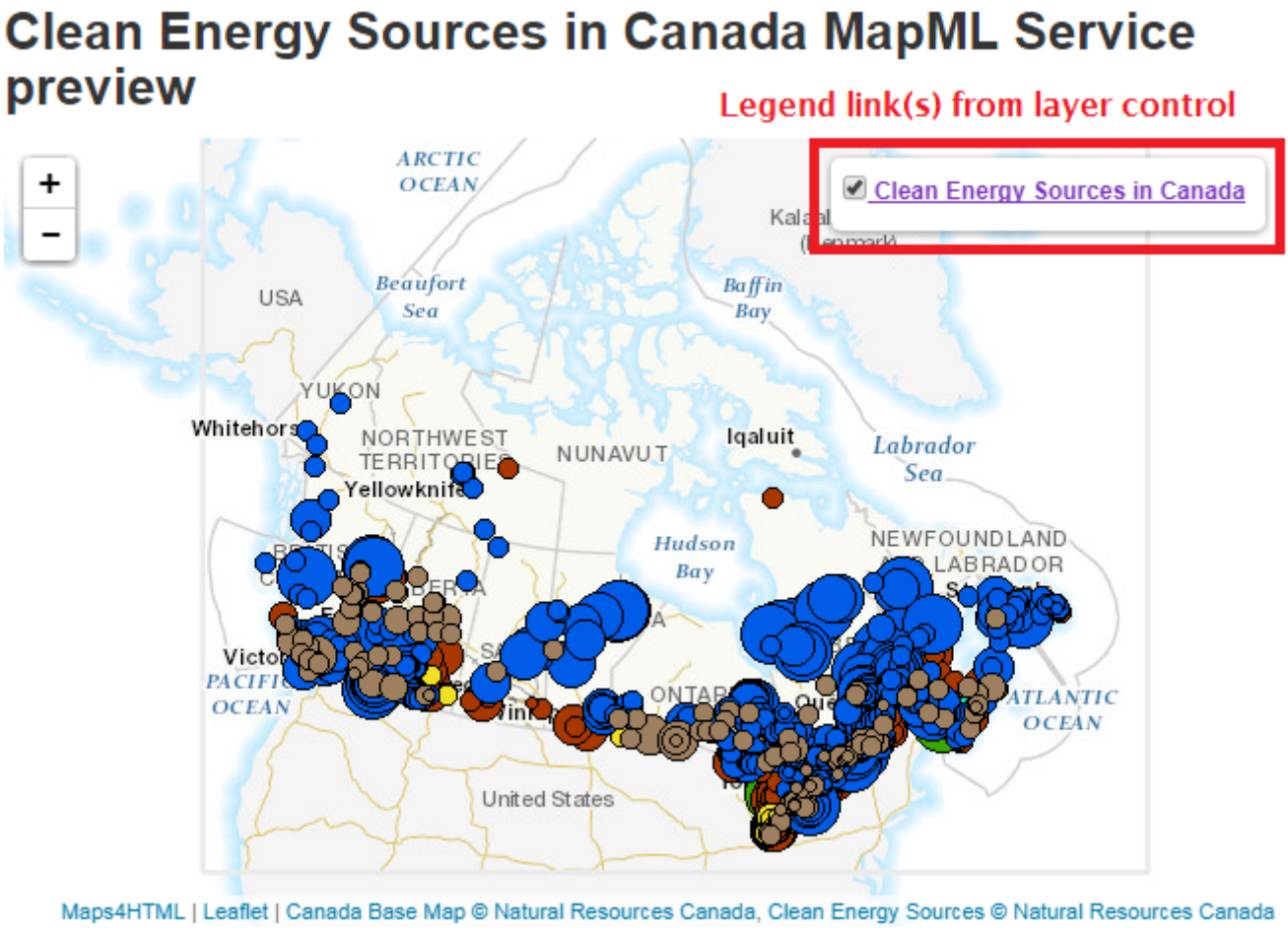


Figure 10. <map> element layer control legend hyperlinks

If the MapML is provided in conjunction with a WMS, it can contain:

- the tag "LegendURL" for this layer in the GetCapabilities document
- the dynamic link to a GetLegendGraphics operation provided by the SLD extension of WMS.

6.5. Styles

A map representation can have different styles associated to it. It is recommended that a MapML document for which such alternative representations exist contain a link element for each such representation, with the link@rel=style link relation.

<link rel="style" title="RADARURPREFLECTR" href="../RADARURPREFLECTR" type="text/mapml"/>

6.6. Map content metadata

It is recommended that MapML documents be able to communicate full and appropriate content metadata. As such, the link element could be used to accomplish this task in the same manner as is used for other document metadata: using the link@rel=via to link to ISO 19115-1 compliant metadata. Such links should carry advisory link@type attributes with appropriate MIME media type values (for ISO 19115-1 metadata in xml, application/xml is recommended). It is recommended that at least one such have the value link@type=text/html, so that human users of the map may read such descriptive metadata and understand the use constraints of the map content, as communicated by the map content author. Once again, it would be up to the user agent as to how

best to provide access or present such links. One possibility would be as a right-click submenu choice in the standard layer control.

NOTE	The use of rel="via" follows the same recommendation as the OGC 12-084r2 OWS Context Atom Encoding Standard

Table 2. ISO 19115-1 metadata for discovery

Metadata name	Metadata element
Metadata reference information	MD_Metadata.metadataIdentifier
Resource title	MD_Metadata.identificationInfo > MD_DataIdentification.citation > CI_Citation.title
Resource reference date	MD_Metadata.idenitificationInfo > MD_DataIdentification.citation > CI_Citation.date)
Resource identifier	MD_Metadata.identificationInfo>MD_DataIdentification.citation > CI_Citation.identifier>MD_Identifier
Resource point of contact	MD_Metadata.identificationInfo > MD_DataIdentification.pointOfContact > CI_Responsibility
Geographic location	MD_Metadata.identificationInfo > MD_DataIdentification.extent > EX_Extent.geographicElement > EX_GeographicExtent > EX_GeographicBoundingBox or EX_GeographicDescription
Resource language	MD_Metadata.identificationInfo> MD_DataIdentification.defaultLocale > PT_Locale
Resource topic category	MD_Metadata.identificationInfo > MD_DataIdentification.topicCategory > MD_TopicCategoryCode
Spatial resolution	MD_Metadata.identificationInfo > MD_Identification.spatialResolution > MD_Resolution.equivalentScale MD_Resolution.distance, MD_Resolution.vertical, or MD_Resolution.angularDistance, or MD_Resolution.levelOfDetail
Resource type	MD_Metadata.metadataScope >MD_Scope.resourceScope
Resource abstract	MD_Metadata.identificationInfo > MD_DataIdentification.abstract
Temporal or vertical extent information	MD_Metadata.identificationInfo > MD_Identification.extent > EX_Extent > EX_TemporalExtent or EX_VerticalExtent
Resource lineage	MD_Metadata >resourceLineage> LI_Lineage

Metadata name	Metadata element
Resource on-line Link	MD_Metadata.identificationInfo >MD_DataIdentification.citation>CI_Citation.onlineResource>CI_OnlineResource)
Keywords	MD_Metadata.identificationInfo > MD_DataIdentification> descriptiveKeywords >MD_Keywords
Constraints on resource access and use	MD_Metadata.identificationInfo > MD_DataIdentification > MD_Constraints.useLimitations and/or MD_LegalConstraints and/or MD_SecurityConstraints
Metadata date stamp	MD_Metadata.dateInfo
Metadata point of contact	MD_Metadata.contact > CI_Responsibility

IMPORTANT Recommendation to MapML: Specify the link@rel=via to link to external metadata documents.

6.7. Extent

Extent has two main purposes. When a MapML document is received, it is used by the map browser to know the extent of the data that the MapML document is describing (in the form of *tile*, *image* and/or *feature* elements), as well as to communicate the global (server) extent where data exists for this layer. At the same time, and since the extent element is defined as an actionable part that can be processed for submission to the server as a HTTP request, it is a search mechanism that is submitted to the web server each time new data is needed (i.e. more tiles or features are needed).

If we assimilate *extent* to a search mechanism, we should compare this with what we already have in OGC: OpenSearch Geo and Filter encoding.

6.7.1. Extent element encoding assimilated to GeoOpeoSearch

OpenSearch-Geo provides a mechanism to search for data. It incorporates a way to communicate a bounding box, a time window and a search by keyword. The parallelism with the current mechanism of MapML is apparent. But there are a couple of fundamentals differences that could justify two encodings: OpenSearch-Geo does not define an HTML form but a URL template based on KVP (that is almost the same as what enctype x-www-form-urlencoded intents to do) and the expected response of an action of submitting the form is an atom file, instead of a MapML document. Assuming that the two presented obstacles can be relaxed, it is worth continuing to explore the commonalities between both encodings and consider completely replacing the current extent section with an OpenSearch-Geo compatible one.

Let us compare the names of the parameters between the two standards in a tabular form:

Table 3. Comparision of OpenSearch parameters and MapML

Description	OpenSearch Geo (Table 3)	MapML	limitations or differences
bounding box	geo:box	xmin, xmax, ymin, ymax	OpenSearch asks for coordinates of longitude, latitude, in a EPSG:4326 decimal degrees while MapML uses tilematrix coordinates in pixels
time window	time:start, time:end	-	
point query	geo:lat, geo:lon, geo:radius	-	Latitude and longitude in decimal degrees in EPSG:4326
place name	geo:name or searchTerms	search	{searchTerms} comes from OpenSearch while {geo:name} is proposed by OpenSearch Geo and it is only a geocoded name.

The OpenSearch Geo section "9.2.1 Search request parameters" mentions, which refers to a bounding box (bbox):

```
Note that for the given key-value pairs, the key can be an arbitrary string, specified
by one given instance of an OpenSearch repository. For example, one Description may
provide a URL template asking for box={geo:box}, another specifying bbox={geo:box}.
```

This means that in order to use this mechanism, MapML should define a URL template that will fix the names in the *extent* form. For example the current MapML request template: xmin={mapml:xmin}&xmax={mapml:xmax}&ymin={mapml:ymin}&ymax={mapml:ymax}&zoom={mapml:zoom}&projection={mapml:projection}&search={mapml:search} could be assimilated to something like bbox={geo:box}&zoom={mapml:zoom}&projection={mapml:projection}&q={searchTerms}

IMPORTANT	Recommendation to MapML: consider to include a temporal extent, as a part of the extent definition to be able to request information in a time window. We recommend to use the OpenSearch Geo approach and notation to it.

IMPORTANT	Recommendation to MapML: describe the search field as OpenSearch {searchTerms} or to OpenSearch Geo {geo:name} depending on the exact aim of the textural search term. If it corresponds to a *place name* can be parsed and geocoded use the {geo:name} concept. We recommend to use the OpenSearch Geo approach and notation to it.

6.7.2. Extent as it is today with small modifications

Assuming that we keep the general mechanism as proposed, there are a couple of modifications that we can suggest.

Input types

HTML5 defines several *types* of inputs. In HTML, input@type conveys the semantics of the control and its representation on screen. For example, input@type=date implies that the value that is input should conform to a date data type. Often, it is possible for the user agent to represent such an input with a calendar-like representation that simplifies entering a valid value for the input.

In MapML, the bounds of the map on screen represent a viewport onto a potentially large amount of data at a range of scales. Typically, in modern Web maps, the user will use a pointing device to drag or otherwise interact with a map. When this happens, it is necessary to coordinate what happens to the map with the gestures made by the user. In principle, this is similar to how users interact with regular HTML Web documents, except in the case of maps, the expectation for what should happen is different, and specific to maps. The calculated edges of the viewport before, during and after the user interaction event can be used by the user agent to request server content based on information already received in the form of <input> elements. In order to semantically group and operate these input elements similar to how an HTML <form> works, we create a parent <extent> element. The extent element identifies its edges using four child input[@type=xmin|ymin|xmax|ymax] elements, which allows the variables corresponding to those edges to be independently calculated and transmitted to the server without regard to variable ordering (in contrast to for example, a single bbox={xmin},{ymin},{xmax},{ymax} with its anonymous parameters and required ordering).

In order to avoid forcing servers to implement specific variable names, MapML creates the input[@type=xmin|ymin|xmax|ymax] values, and allows the server to specify what the corresponding query parameter name should be via the input@name value. In this way, the server and the user agent share the semantic meaning of input[@type=xmin|ymin|xmax|ymax] by virtue of both 'understanding' (implementing) the MapML specification, while leaving the server's namespace completely up to the server.

List of available projections.

The current MapML specification has an input[@type=projection] with the projection name. The map browser could prefer to change projection to another value but there is no information on the *extent* section about the possible projection values. If a number of possible choices of projection were available for a particular resource, it might be reasonable to offer a menu of such choices so that the user agent could 'negotiate' what projection it needed for a particular map layer. Given that any input[@type=xmin,ymin,xmax,ymax] attributes present would necessarily contain coordinate

values in the units of the MapML projection identified by the input[@type=projection]/@value string that was returned or selected by the server for the resource being represented, it is reasonable to note that the extent@units value describes the units of such input[@type=xmin,ymin,xmax,ymax] property values and so is not necessarily redundant or non-'DRY'; the user agent should be careful that when submitting such an extent, should it select an alternative projection value from the list of values, it SHOULD NOT submit corresponding input[@type=xmin,ymin,xmax,ymax] query parameters since their units would not correspond to that projection selected.

HTML5 offers the *datalist* element as a mechanism to enumerate suggested options, for an input, that we also can apply here. This is the proposed change:

```
    <extent action="/mapml/cbmt/" enctype="application/x-www-form-urlencoded" method=
"get">
      <input name="projection" type="projection" list="projection" value="CBMTILE"/>
      <datalist id="projection">
        <option value="OSMTILE"/>
        <option value="CBMTILE"/>
        <option value="APSTILE"/>
      </datalist>
    </extent>
```

NOTE	Recommendation to new Testbeds: Enumerate the possible projections supported by the server resource in a *datalist* element with an id that is linked to an input *list* attribute.

IMPORTANT	Recommendation to MapML: Remove the erroneous enctype="application/x-www-form-urlencoded" value when the method="GET", since there is no entity body being transmitted. When it becomes possible or necessary to allow method="POST" or method="PUT" or method="PATCH", it will be necessary to include an appropriate extent@enctype value for MIME media type of the entity body being transmitted.

Projection names need to be substituted by tile matrix set identifiers

The tile matrix set standard candidate (OGC 17-083) defines a list of tile matrix sets with a predefined identifier. It should be good that the next version of MapML refers to this list and removes the current list of internally defined "projections".

IMPORTANT	Recommendation to MapML: replace the list of internally defined *projections* by a reference to the near future tile matrix set standard candidate OGC 17-083.

Tile matrix names instead of zoom levels

In OGC 17-083 a tile matrix set has a list of tile matrices available that has associated an identifier and a scale denominator but not a number. It is still possible to refer to the tile matrix by a number

(considering the order in which they have been defined) and call this *zoom level* but this is not a practice recommended by OGC 17-083.

If we replace the *zoom* level by a tile matrix identifier, *zoom* is no longer a range of numbers. To mitigate this, we recommend to use the same *datalist* solution that was suggested for extending *projection*

```
    <extent action="/mapml/en/osmtile/cbmt/" enctype="application/x-www-form-
urlencoded" method="get">
      <input name="zoom" list="zoom" type="zoom" value="0">
      <datalist id="zoom">
        <option value="0"/>
        <option value="1"/>
        <option value="2"/>
      </datalist>
    </extent>
```

IMPORTANT	Recommendation to MapML: replace the zoom level number by a tile matrix identifier (text). Include a datalist enumerating the possible values of *zoom*

Projection and zoom

The word *projection* is an unfortunate election for the concept relevant in MapML. The *projection* is only a part of the coordinate reference system definition that is also a part of the tile matrix set definition. The use of the word "projection" is misleading and not conformant with the OGC abstract specification.

The word *zoom* is commonly used in the map browsers using tiles. Unfortunately, this is not the name selected by the WMTS specification and it could be appropriate to change it accordingly.

IMPORTANT	Recommendation to MapML: consider to change *projection* and *zoom* by *tilematrixset* and *tilematrix* for consistency with OGC 17-083.

6.7.3. Time and animation

In the new era of Big Data, there are many sources of datasets that are systematically being automatically acquired. It is important to know the time of the actual phenomena measured and to be able to filter by the time the user is interested in. In this document, we propose three alternatives to deal with time.

Alternative 1: A MapML has a single static time

The MapML document represents a single snapshot of an extent at a moment in time. A <meta> element could indicate what time the tiles, images and features represent. The HTML author is able to add layers that represent individual snapshots to allow the user to select among them using the standard user interface layer control, or to allow the map browser to animate them.

A meta element for such a MapML document could be like <meta name="time" content="<ISO 8601

value>"/>

Alternative 2: MapML has control the time of the snapshot

The MapML represents a single snapshot but offers a range of possible times and the actual time in the <extent> control. The HTML composer adds a single layer and offers the user to select among the possible times. The tiles and features in the MapML represents a single time.

This approach can be implemented with a syntax like this:

```
<extent action=" https://example.org/mapml/layer/" ...>
  ....
  <input type=time name="t" min="some time value" max="a time value greater than min"
value="a value between max and min">
</extent>
```

The <extent> action will add a "t" key like

```
https://example.org/mapml/layer/?t={value}
```

which would imply a map at time={value}

In such a scenario, in addition to displaying the layer in the layer control, each layer could represent the time parameter as a "time control" (e.g. a time slider +/- date picker) so that the user could control the time (t) value transmitted in the request(s). A simplified approach could be to have a single time control for the map and assume that all layers in the html map are represented synchronized.

Alternative 3: A MapML service can represent an interval of times

The MapML document represents an interval of times. The map browser might offer a one or two time-sliders to the user to control the queries issued as the user pans or zooms, and potentially the response could be animated using Cascading Style Sheets (CSS) processing emitted by the server.

```
<extent action=" https://example.org/mapml/layer/" ...>
  ...
  <input type=tmin name="tmn" min="2017-01-01T00:00:00Z" max="2017-12-31T23:59:59Z"
value="2017-10-01T00:00:00Z">
  <input type=tmax name="tmx" min="2017-01-01T00:00:00Z" max="2017-12-31T23:59:59Z"
value="2017-10-31T23:59:59Z">
</extent>
```

```
https://example.org/mapml/layer/?tmn={tmin-value}&tmx={tmax-value}
```

which would imply a map over a time range. In the response to the above query, <feature> elements in the response might have a time property:

```
<feature>
    <properties>
        <name>Thing1</name>
        <time>2017-10-20T13:15:08Z</time>
    </properties>
    <geometry>
        <Point>
            <coordinates>-75.6749127 45.3840208</coordinates>
        </Point>
    </geometry>
</feature>
```

NOTE

Recommendation to new Testbeds: Experiment with linking one or two user interface controls (sliders) for time to <input> parameters in the <extent> to allow query of MapML services by time interval. Determine if indeed input@type=tmin|tmax are required or if a single input@type=time with multiple instances (different input@name values), linked to user interface controls would suffice. Verify what a "map of a time interval" response would look like and how it might behave. For example, a MapML response could include a <layers> element with a set of child <layer> elements, each with its own layer@src value. Such a response could include CSS code to animate the display of the child layers/layer elements so that the overall response would represent a map of a time interval.

6.7.4. Tile templates

Some tile service implementations follow a pattern in the tile URL naming, making a tile URL predicable if the pattern is known by the client. TileCache uses the pattern {base_url}/{z}/{y}/{x}. OGC WMTS REST interface also defines URL patterns that are not fixed but declared in the service metadata. These patterns are declared using the URI templates specification (actually, the specification does not cite the approved version or RFC6570 https://tools.ietf.org/html/rfc6570 explicitly, but a draft of it, because the approved one was not available at the time WMTS standard was approved). In addition, the WMTS KVP syntax can also be expressed as a URI template like this: {base_url}?service=WMTS&request=GetTile&version=1.0.0&layer=etopo2&style=default&format=image/png&TileMatrixSet=WholeWorld_CRS_84&TileMatrix={TileMatrix}&TileRow={TileRow}&TileCol={TileCol}.

If a URI template describing the URL of all possible tiles is available, it could be included in a MapML file. In this case, the client could retrieve any tile if it needs to react to the actions of the user that changes the view bounding box (e.g. pan or zoom), without any need to request a new MapML file that contains the tiles URLs of the affected new bounding box. Instead, the client can transform the URL template into as many tile requests as it needs directly.

This approach introduces more programming requirements on the client side, since the client needs to transform a bbox in a series of tile requests, while, in the current approach, the server provides MapML files that include them. In contrast, this approach reduces the number of server interactions, in particular if the MapML only contains tiles (and not features).

We propose to extend the <extent> element to include an element called <template> that includes an attribute "type" (for the moment with a value fixed to "tile") and an attribute "uri" that will contain the URI template. The element <template> can be repeated more than once if the MapML includes tiles following more than one URI template (for example, if it includes tiles of two layers overlapped). The following example shows how the current MapML standard is extended to include the new characteristic in the extent element.

```
    <extent units="CBMTILE">
      <input type="xmin" min="768.0" max="1024.0"/>
      <input type="ymin" min="768.0" max="1280.0"/>
      <input type="xmax" min="768.0" max="1024.0"/>
      <input type="ymax" min="768.0" max="1280.0"/>
      <input type="zoom" min="0" max="17" name="z"/>
      <input type="location" position="absolute" property="y" units="tile" name="row
"/>
      <input type="location" position="absolute" property="x" units="tile" name="col
"/>
      <input type="projection" value="CBMTILE"/>
      <template type="tile" uri=
"https://geoappext.nrcan.gc.ca/arcgis/rest/services/BaseMaps/CBMT3978/MapServer/tile/{
TileMatrix}/{TileRow}/{TileCol}">
    </extent>
```

6.8. Other forms

Apart from the *extent* form, a MapML document could contain other *forms* dedicated to deal with specific actions of the user not related to filling the viewport (i.e. not zoom or pan), such as queries or searches. Since a map browser does not have buttons for common map operations, there is need to associate each form to events triggered by the user. A key question would be how to visually or otherwise associate the ability to query a map layer with a user interface expression of that ability. In HTML, there is the <input type="submit"> input type, which is rendered as a button. Such an input type could be used in a map <form> element with a different default expression, for example the pointer could change from its default "hand" display to something more typically associated with visual query, such as a crosshair or other form. The key idea would be to associate a click or touch on the form (which would naturally have a spatial extent due to its presence in a MapML document). Child input elements, as well as form@action properties would have to be configurable so that click/touch coordinate information could be transmitted from the client to the server according to query parameter names identified by the form child input elements. The response from the server could be a MapML or HTML document that would be "geolocated" by the click event.

Our proposal is to introduce forms for use in MapML documents that will be processed and submitted automatically on selected user events, such as click or touch, when an appropriate input@type=submit is present:

```
<form action="http://www.ccc.org/mapml">
<input type="submit">
</form>
```

6.8.1. Query by location

One of the most common actions in a map is clicking on it hoping to get more information about some represented features, such as can be obtained with a WMS GetFeatureInfo request. On a map browser, we can create a form that carries appropriate input@type children which will transmit event specific variables as query parameters (or other potentially also other templated request parts).

```
<form action="http://www.ccc.org/mapml">
<input type="location" name="i" value="location.x">
<input type="location" name="j" value="location.y">
<input type="zoom" name="z">
<input type="submit">
</form>
```

As a result of the click event submitting the above form, the server should return a MapML or HTML document describing the features at the event location that will show in a popup/map balloon (tooltip) or other browsing context. In examining the above (hypothetical) markup, a number of questions naturally arise. For example, in what units are the coordinates of the input@location? Should the units of the location be controlled by other input attributes? Should the input@type=location have a visual expression or should it be hidden by default?

6.8.2. Text search

Textual search is an important characteristic of maps. It is proposed here that such search be implemented by the HTML map element as an aggregating control; if the HTML author enables search for at least one layer that they include on their map, and if the map document for that layer includes a form element(s) configured for search via a child input@type=search element, a search text input control could be presented to the end user in a default manner by the user agent (for example at the top left of the map, as is common in today's map browsers). Since there could be more than one layer on a map which is searchable, the map's search text input would have to aggregate such searches, and convey result sets or search hints appropriately. In general, the user action of selecting from a result set list obtained through search should display the location of the result on the map in a natural manner. The challenge of designing the markup for the map documents is to identify what can be standardized versus what should be left to the provider of the MapML document and again, what should be the responsibility of the user agent. This brings up many questions, including whether a MapML document needs to be consumed by an HTML <map> element, i.e. what if the MapML document is the direct target of the browsing context. What would the map look and behave like?

```
<form action="http://www.ccc.org/mapml">
<input type="search" name="q">
<input type="zoom" name="z">
<input type="submit">
</form>
```

IMPORTANT	Recommendation to HTMLMap: consider the inclusion of a boolean "search" attribute to the layer element, to allow for activating of deactivating the search functionality at the map level.
IMPORTANT	Recommendation to MapML: It is unclear if the search need to be limited to the current bbox. The authors of this document consider that it is better not to do that. Perhaps it is possible to give the spatial constraint ability to the map document author by allowing the map author to identify search bounds through input@type=xmin|ymin|xmax|ymax inclusion in the form.
IMPORTANT	Recommendation to MapML: It this recommendations are adopted, the search mechanisms that the current version of mapml included in the <extent> needs to be removed.

6.8.3. Tooltips

A tooltip is a floating window (commonly with a yellow or orange background) that contains short information about the features in the map and that follows the mouse pointer. The information on the tooltip changes when the mouse changes position without any need for clicking it (hover). This characteristic is losing popularity since mobile phones and tables are not equipped with a mouse but it could reemerge in virtual reality glasses. We propose to define a form that has an *input* element with a *name* = *event* and a *value* defined as *onHover* that could replicate this functionally. The map browser would trigger this event when the mouse moves over the map.

```
<form action="http://www.ccc.org/mapml">
<input type="hidden" name="event" value="onHover">
<input type="location" name="i" value="location.x">
<input type="location" name="j" value="location.y">
</form>
```

6.8.4. Target of a MapML form

In MapML there are two kinds of forms: `<extent>` and `<form>` (the second one is not yet included in the MapML current draft specification). Like HTML5 forms, MapML forms have an `action` attribute that specifies where to submit the form inputs, but there is no actual submit button for the user to press. In the case of the <extent> element, it is expected that the user agent submits the form each time the map needs more map content. Common situations where new information is required are when the user performs a pan or a zoom action, or the window is resized or re-oriented. In the case of the <form> element, this report recommends that when the user clicks the map to know more

about a feature (per the above discussion), the assumed default target of the action is a map balloon. Depending on the *target* attribute of the *form*, the map author might direct what to do with the response to form submission. It is assumed that in the case of a default target, the submission will result in a MapML or HTML document that will display in a map balloon browsing context. If there was a *target*, the response could be redirected to another window or frame with that name and the map view will not be changed in the client.

CAUTION	"target=" has not been specified in the current stable version of MapML.

IMPORTANT	Recommendation to MapML: consider the inclusion of target attribute in the definition of extend of the current version of MapML.

6.8.5. Directions

Another important functionality in a map browser is to be able to provide directions to go from one place to the next. Again, the way the map element is implemented, we expect that search text input(s) should be provided at the map browser level and not at the layer level. Nevertheless a layer in the map browser should be able to provide this functionality, so a form (or some additional elements in the extent form) could be required to allow that. Two input elements of type "from" and "to" is the minimum set required but other attributes (such as the type of transport the user has access to) could be necessary.

NOTE	Recommendation to future Testbeds: Investigate searching for directions on a map layer.

6.9. Tile encoding

Recently, there have been some mass market map browser implementations that use vector tiles. It could be good that MapML is able to support them.

IMPORTANT	Recommendation to MapML: consider allowing vector tiles in MapML.

6.10. Feature encoding

Note the discussion of the 4 formats: GeoRSS, GeoJSON, KML and Schema.org https://www.w3.org/wiki/WebSchemas/GeoShapeExamples

In addition, Well-Known Text (WKT) could be an alternative for the Geometry.

- Feature encoding
- Compare GeoShape with simple features and fix the current issues.
- Symbology (CSS) to segments of lines and links to pieces of geometries.

6.10.1. Feature encoding based on MicroXML

MapML currently proposes an encoding for features that resembles GeoJSON in the sense that it

describes a feature as formed by geometry and properties. Properties are not specified and can have a free list of property elements and values. The geometry is expressed by a set of coordinates. A MapML file can have a list of features but they are not grouped in feature collections.

```
    <feature class="_1240012" id="b71b3e963c7a4792b63e92cd7b54589f">
      <properties>
        <code>1240012</code>
        <accuracy>23</accuracy>
        <valdate>20010428</valdate>
        <id>b71b3e963c7a4792b63e92cd7b54589f</id>
        <theme>VE</theme>
        <type>222</type>
      </properties>
      <geometry>
        <Polygon>
          <coordinates>-75.6003064 45.3998946 -75.6003898 45.4000408 -75.6004666
45.4002181 -75.6005368 45.4004264 -75.6006005 45.4006658 -75.6006576 45.4009362
-75.6007082 45.4012377 -75.60012 45.4013165 -75.5997066 45.4013325 -75.5994679
45.4012857 -75.599404 45.401176 -75.5994095 45.4010429 -75.5993792 45.4009256
-75.5993129 45.4008241 -75.5990932 45.4006444 -75.5989812 45.400518 -75.5988746
45.4003591 -75.5987735 45.4001677 -75.6003064 45.3998946</coordinates>
        </Polygon>
      </geometry>
    </feature>
    <feature class="_1240012" id="e7ccaa32978942518004cf4c06152817">
      <properties>
        <code>1240012</code>
        <accuracy>23</accuracy>
        <valdate>20010428</valdate>
        <id>e7ccaa32978942518004cf4c06152817</id>
        <theme>VE</theme>
        <type>232</type>
      </properties>
      <geometry>
        <Polygon>
          <coordinates>-75.5999067 45.3988121 -75.59997 45.3990799 -75.6000417
45.3993221 -75.6001216 45.3995386 -75.6002099 45.3997294 -75.6003064 45.3998946
-75.5987735 45.4001677 -75.5985081 45.3997915 -75.5983765 45.3995832 -75.5983819
45.3995191 -75.598544 45.3994562 -75.5988659 45.3993708 -75.5992103 45.399268
-75.5994399 45.3991532 -75.5995733 45.3990294 -75.5997227 45.3989147 -75.5999067
45.3988121</coordinates>
        </Polygon>
      </geometry>
    </feature>
```

The authors of this Engineering Report believe that features are an essential part of OGC semantics. As such, the encoding of feature information as currently implemented by MapML may be a starting point for more human-friendly encoding and display of geographic features. What follows are some alternatives or extensions to the current feature model that may make integration of

geographic information more natural on the human web.

6.10.2. Feature encoding alternative 1: schema.org

This encoding explores how to use schema.org to semantically tag html text and provide the necessary information that describes a feature.

There are several entities in schema.org that can be mapped to features depending of their meaning. If none of them apply, we can always use "Place" (http://schema.org/Place) or we can extend schema.org to our purposes. In this example, we are mapping each feature to "Place" and we use a few of the common properties of *Place* to tag our information fields (*identifier*, *url* and *alternateName*). Other existing properties can be useful in many cases (*logo*, *photo*, *review*, *description*, *sameAs*, *potentialAction*) In case none of the properties of Place apply, we can add uncommon properties using the *PropertyValue* mechanism which allows us to include in *additionalProperty* that define and use properties using *name & value* pairs.

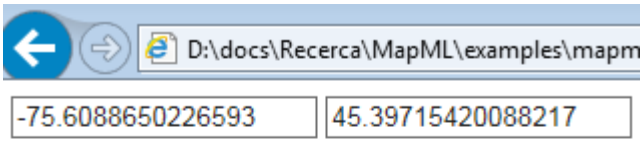
For the geometry property, the standard OGC model could be used, structured into standard element names, with the additional consideration that the coordinates of a feature on a map are analogous to human-readable text in a traditional HTML document. As such, the coordinates content should be subject to further, though possibly constrained markup, such as , <a> and possibly other phrasing content markup [https://www.w3.org/TR/html5/dom.html#phrasing-content-1] (further prototyping/analysis required).

The possibility of using HTML+schema.org for the geometry property has been analyzed but the elements GeoShape and GeoCoordinate do not fit with the requirements needed in terms of coordinate reference system and polygon composed of multiple rings (both possibilities are not supported by schema.org).

```
  <feature>
    <properties>
      <div itemscope="itemscope" itemtype="http://schema.org/Place" class="_1240012" >
        <h3>Code: <span itemprop="alternateName">1240012</span></h3>
        <b>Id:</b><span itemprop="identifier">b71b3e963c7a4792b63e92cd7b54589f
</span><br>
        Id:</b><span itemprop="identifier">b71b3e963c7a4792b63e92cd7b54589f</span>
        <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
            <span itemprop="name" content="accuracy"/>
            <b>Accuracy:</b> <span itemprop="value">22</span>
        </div>
        <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
            <span itemprop="name" content="valdate"/>
            <b>Validation date:</b> <span itemprop="value">20010428</span>
        </div>
        <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
            <span itemprop="name" content="theme"/>
            <b>Theme:</b> <span itemprop="value">VE</span>
```

```
        </div>
        <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
            <span itemprop="name" content="type"/>
            <b>Type:</b> <span itemprop="value">222</span>
        </div>
    </properties>
    <geometry>
        <Polygon>
            <coordinates>
            -75.7371421 45.4251194 -75.7372231 45.4250821 <span class="highlight">
-75.7374238 45.4250815 -75.7376529 45.4251037 -75.7378188 45.425135</span> -75.7379875
45.4252532 -75.7382589 45.4255763 -75.7385533 45.4259813 -75.7386907 45.4261917
-75.7386682 45.426238 -75.7384977 45.4262523 <a id="something" href=
"https://example.org/something">-75.7382147 45.4262377 -75.7379863 45.4261611</a>
-75.7378242 45.426027 -75.7376472 45.4259032 -75.7374442 45.4257981 -75.737286
45.4256386 -75.7371939 45.4254063 -75.7371486 45.4252161 -75.7371421 45.4251194
            </coordinates>
        </Polygon>
    </geometry>
  </feature>
  <feature class="_1240012">
    <properties>
        <div itemscope="itemscope" itemtype="http://schema.org/Place" class="_1240012"
>
            <h3>Code: <span itemprop="alternateName">1240013</span></h3>
            <b>Id:</b><span itemprop="identifier">
e7ccaa32978942518004cf4c06152817</span>
            <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
                <span itemprop="name" content="accuracy"/>
                <b>Accuracy:</b> <span itemprop="value">23</span>
            </div>
            <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
                <span itemprop="name" content="valdate"/>
                <b>Validation date:</b> <span itemprop="value">20010428</span>
            </div>
            <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
                <span itemprop="name" content="theme"/>
                <b>Theme:</b> <span itemprop="value">VE</span>
            </div>
            <div itemprop="additionalProperty" itemscope="itemscope" itemtype=
"http://schema.org/PropertyValue">
                <span itemprop="name" content="type"/>
                <b>Type:</b> <span itemprop="value">232</span>
            </div>
        </div>
    </properties>
    <geometry>
```

```
        <Polygon>
            <coordinates>
                -75.5999067 45.3988121 -75.59997 45.3990799 -75.6000417 45.3993221
-75.6001216 45.3995386 -75.6002099 45.3997294 -75.6003064 45.3998946 -75.5987735
45.4001677 -75.5985081 45.3997915 -75.5983765 45.3995832 -75.5983819 45.3995191
-75.598544 45.3994562 -75.5988659 45.3993708 -75.5992103 45.399268 -75.5994399
45.3991532 -75.5995733 45.3990294 -75.5997227 45.3989147 -75.5999067 45.3988121
            </coordinates>
        </Polygon>
    </geometry>
  </feature>
```



Code: 1240012

Id:b71b3e963c7a4792b63e92cd7b54589f
Accuracy: 22
Validation date: 20010428
Theme: VE
Type: 222
More info...

Code: 1240013

Id:b71b3e963c7a4792b63e92cd7b54589f
Accuracy: 23
Validation date: 20010428
Theme: VE
Type: 232
More info...

Figure 11. HTML interpretation of the features fragment of a MapML example.

This example was validated by the Google Structured Data Testing tool (https://search.google.com/structured-data/testing-tool) resulting in no warnings or errors.

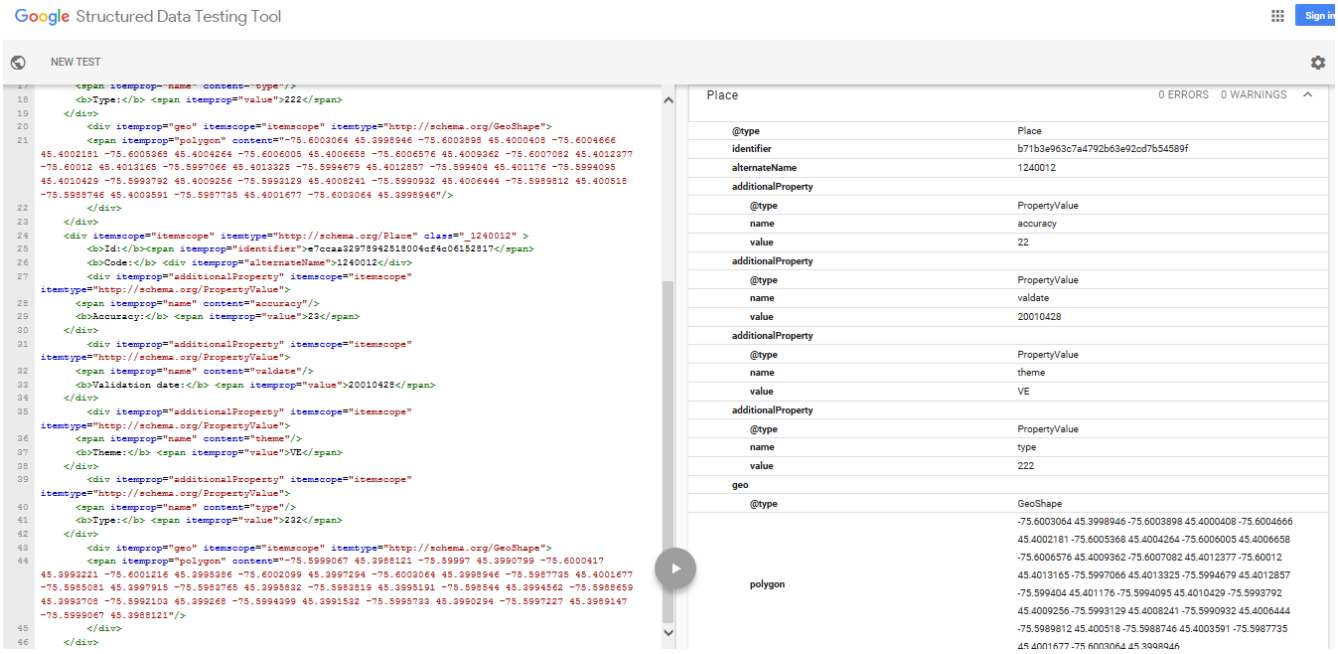


Figure 12. Result of the test of the example in the Google Structured Data Testing tool

6.10.3. Why current version of schema.org is not enough

- GeoCoordinates only allow for a point in lat/long wgs84. (https://github.com/schemaorg/schemaorg/issues/802)
- GeoShape does not specify the CRS and does NOT allow for a point.
- It is not clear how to do a multipolygon. Can we assume that the repetition of a coordinate is the end of a ring? a negative area ring a hole in a polygon?
- Can we do geometry collections?
- Do we need a bbox?

The editors of this document are now members of the discussion list and can submit a question on the best way to extend schema.org. OGC should review http://schema.org/docs/extension.html.

In fact, Schema.org is not very consistent in its own definitions. For example they are proposing a http://pending.schema.org/GeospatialGeometry that is actually talking about topology. It mentions an informal collaboration with the W3C Spatial Data on the Web Working Group. (http://schema.org/docs/releases.html)

Chapter 7. MapML versus other formats

This section compares MapML with OWS Context and KML.

7.1. OWS Context

MapML shares some similarities with OWS Context [http://www.opengeospatial.org/standards/owc]. The most obvious is that OWS Context can transport references to WMS, WMTS services and embedded content in the form of features and MapML also does that.

In fact, a OWS Context document can be assimilated to a <map> section that uses the HTML4MAP extension. OWS Context document contains a list of resources (called entries in the Atom encoding) while a <map> section contains a list of <layer>s (that are actually a list of MapML document links). Each entry in a OWS Content can contain one or more offerings with references to tiles, embedded features or other content, while a MapML document is able to transport also references to tiles and embedded features. Technically, the parallelism of both encodings is apparent. Nevertheless, there are some significant differences:

- OWS Context can include other kinds of OGC services (e.g. WPS) and other kind of embedded content (e.g. GML) that MapML does not support.
- OWS Context does not enumerate the URLs to resources but, for each service, provides only a URL "example" for getting resources (e.g. for a WMTS, a URL to a tile that usually does not cover the entire view port) and assumes that the client would be able to understand the logic behind the OGC service and will be able to generate the necessary requests (e.g in the case of WMTS, the URLs for all needed tiles to cover the view port). In contrast, MapML does not require that the client has any knowledge on how to build OGC web service requests and provide the exact requests need for each situation.
- The current OWS Context standard includes a set of extensions that define how to reference a limited set of OGC standards and formats in the offerings sections. For tiles, only a OGC WMTS service extension is provided, leaving the possibility of using non OGC tile services undefined. In contrast, MapML can support non OGC tile services as far as it is possible to point to tiles using URLs.
- OWS Context can also transport symbology that is associated with features to provide default presentation. MapML is not doing this at this moment and should rely on a separated css document. This document will be included in the web page where the MapML document is also included.

Perhaps the most significant aspect is that among the use cases that OWS Context considers, there is the capacity to save and share the status of a map browser but it is not intended to be a media type to exchange the status of a map between a map browser and a map server. We could say that OWS Context was not designed to cover the same use cases that MapML does. The OWS Context use case assumes that the map browser is intelligent enough to dialog directly with a variety of OGC services without the need to exchange the status of the map browser to a OWS Context service when a user event occurs.

We have started discussions with the OWS Context SWG to see if the use cases covered by MapML

could be covered by OWS Context and MapML could be considered a encoding extension for OWS Context.

IMPORTANT	Recommendation to OWS Context.SWG: Consider including the use cases covered by MapML in the list of OWS Context use cases and the combination of HTML4MAP and MapML as an HTML encoding extension for OWS Context.

7.2. KML

MapML shares some similarities with KML [http://www.opengeospatial.org/standards/kml]. KML was designed as a format for exchanging geospatial data and symbology among virtual globes. Even if it has characteristics that are specific to 3D virtual globes (e.g. point of view), it can also be used in 2D map browsers. It has both the capacity to exchange features in KML format and tiles.

KML was designed to share vector or images at once when it is loaded and does not communicate the status to a server for incremental data. Actually, the KML superoverlays allow for some degree of incremental load of information but in a very different that MapML does.

Chapter 8. MapML services and navigation modes

In practice, MapML can be implemented in four possible modes of navigation. These modes are complementary and some or all of them can be made available in an implementation of a MapML engine. The four navigation modes are:

- Map package: A MapML document contains a large amount of information (tiles, features and image) in a way that allows the user to navigate through the map view without exiting a pre-defined limits. In other words, the MapML file contains much more information than the one needed for the current view and includes adjacent areas and information valid at different resolutions.
- Map links: The MapML document contains detailed information in the <meta> section about how to move from one zoom to the next and to the current view to lateral views using the <link> element. The client offers these limited possibilities more prominently to the user.
- Map service: A MapML document contains only the requested extent bbox and will submit a new extent form each time the user makes an action and waits for the next MapML file to be returned.
- Map search: An alphanumeric string can be sent to the server that will look for places that corresponds to the searched string (place names, geoparsing etc).

8.1. Determining the center of the map to start with

The MapML <extent> section defines the zoom level that a MapML engine should use to start the visualization in the <input type=zoom> key. In the usage of MapML, there is some ambiguity in the definition of the exact area that will be presented to the user as starting point, because the bounding box in the <extent> might not be compatible with the width and height of the screen defined in the HTML <map> and because a <map> could include MapML files (<layer>s) with different <extent>s . This section describes how the MapML engine will determine the center of the map and the initial zoom level.

Initially the MapML engine needs to determine the projection and zoom level it wants to work with. That can be selected among the list of MapML files included in a map in several ways. We suggest to adopt the combination of CRS and tile matrices that is mentioned more times and initially ignore the MapML documents that do not use this combination. From the selected extents, compute the minimum bounding box that includes all the individual bounding boxes and determine the central point of it. Then, using the width and height of the <map> section, and the selected tile matrix zoom level calculate the actual initial bounding box that will be shown in the map window. The following figure shows a UML sequence diagram representing this sequence of decisions.

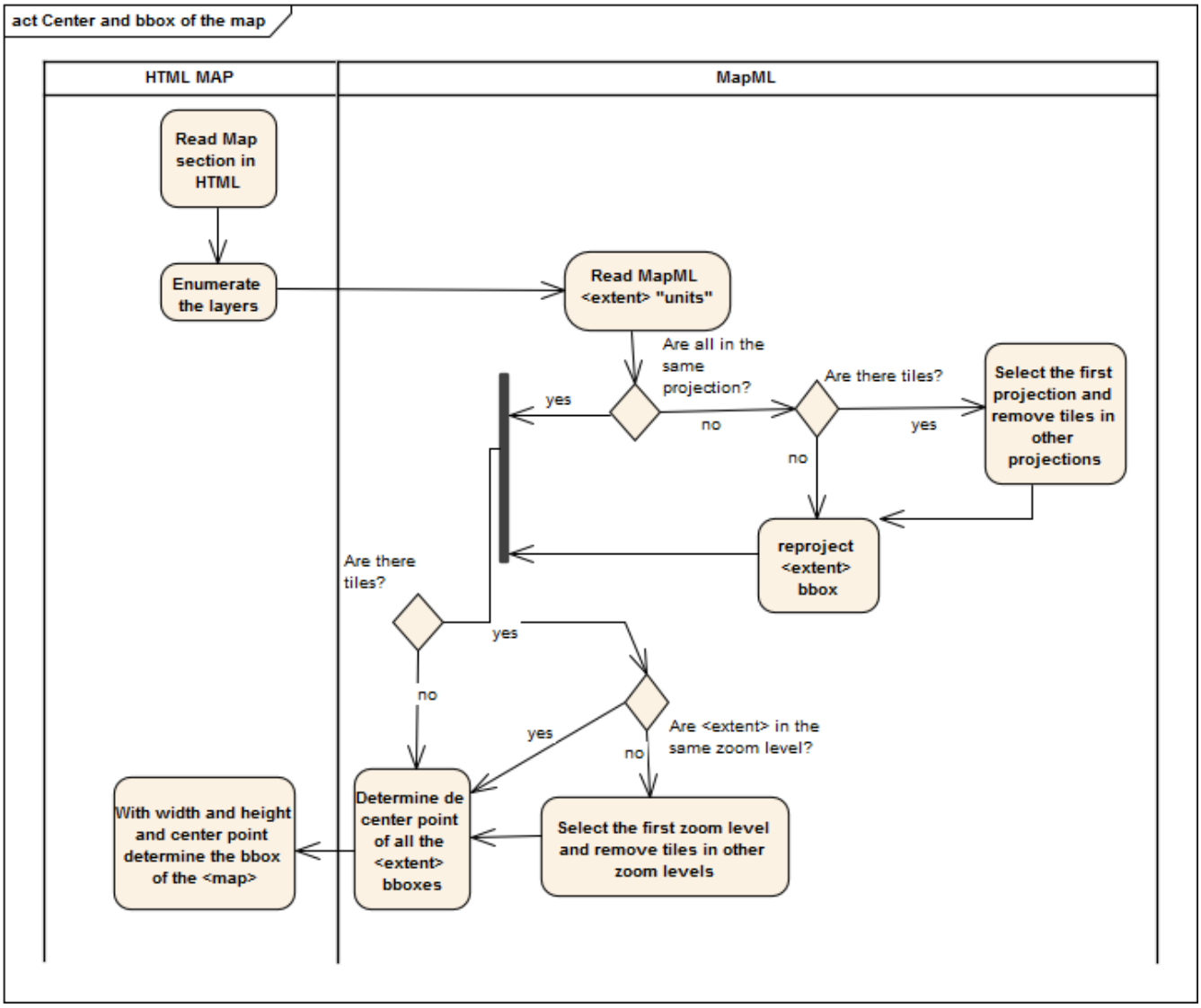


Figure 13. MapML interaction UML sequence diagram

8.2. MapML package

This is one of the possible navigation modes. A MapML document has no *a priori* limitation in the number of tiles and features it can include. It is perfectly possible to generate a single MapML file that contains all the necessary information that is needed for the initial user interactions. The <extent> section is used as an indication of the bbox and zoom level that should be presented to the user, but the MapML file could have links to tiles and feature representations that goes far beyond that bbox. This way, the user will be able to interact with the map but the client will not need to request any additional information to the server.

If changes in the zoom level of the map are required, this approach cannot be fully implemented in the current version of MapML due to an explicit limitation in the specification that forces all tiles (actually all data) in the MapML to be of the same the zoom level. This is also a consequence of the decision of having all MapML coordinates as tilematrix coordinates. A clarification in the <tile> section of the specification confirms this diagnostic:

```
The "zoom" value is a global integer property of a MapML document whose coordinate
system is defined by this specification. All MapML documents have a defined zoom
value. The "zoom" value is equal to the "zoom/@value" child of the "extent" element.
Hence the zoom value is not a direct attribute of the "tile" element.
```

CAUTION	For big datasets this approach can result in very large files an could be impractical. The next sections describe more flexible approaches to client-server dialog for MapML exchange that limits the size of a MapML file.

IMPORTANT	Recommendation to MapML: consider relaxing the limitation that all tiles needs to be in the same zoom level stated in the sub section 4.2.1.1.6 of the current version of the MapML allowing other zoom levels to be described in a single MapML.

A partial alternative to this approach (only applicable for tiles) could be based on URI templates for tiles and the new element <template> introduced in <extent> the previous section.

8.3. MapML links

In this navigation mode, the MapML contains detailed information about how to request a new MapML that moves from one zoom to the next and to the current view to lateral views. This was proposed to allow for implementations that does not require to be spatially aware and should rely on the HATEOAS constraint of the REST application architecture (sometimes referred as "follow the links!"). MapML files can be pre-calculated on the server side and delivered to the client as files. To do that, <link> elements are included in the <meta> section. The current MapML version defines a set of 'rel' values that inform the map browser on how to navigate from one MapML to the next.

- `west` Indicates a resource that may be used to the west of the maximum extent of the current resource, at the current zoom level.
- `southwest` Indicates a resource that may be used to the southwest of the maximum extent of the current resource, at the current zoom level.
- `south` Indicates a resource that may be used to the south of the maximum extent of the current resource, at the current zoom level.
- `southeast` Indicates a resource that may be used to the southeast of the maximum extent of the current resource, at the current zoom level.
- `east` Indicates a resource that may be used to the east of the maximum extent of the current resource, at the current zoom level.
- `northeast` Indicates a resource that may be used to the northeast of the maximum extent of the current resource, at the current zoom level.
- `north` Indicates a resource that may be used to the north of the maximum extent of the current resource, at the current zoom level.
- `northwest` Indicates a resource that may be used to the northwest of the maximum extent of the current resource, at the current zoom level.

- `zoomin` Indicates a resource that may be used to the zoom at the current zoom level plus 1.
- `zoomout` Indicates a resource that may be used to the zoom at the current zoom level minus 1.

CAUTION	For the moment, this approach has not been implemented and it is unclear if it is longer necessary. The next section describes a more flexible approach to client-server dialog for MapML exchange that is, in some sense, redundant with this one.

We propose to add two next 'rel' values to include the time dimension in the considerations.

- `before` Indicates a resource that may be used to present a scene that represents a moment immediately before the current one.
- `after` Indicates a resource that may be used to present a scene that represents a moment immediately after the current one.

IMPORTANT	Recommendation to MapML: consider the inclusion of `before` and `after` in the section 4.2.1.1.2.4 of the current version of MapML.

NOTE	Recommendation to new testbeds: The HATEOAS approach to MapML based on <link> could be experimented in a next Testbed.

8.4. MapML service

In this navigation mode, there is a service implementation that is able to dynamically generate a MapML file that responds to each action of the user. The action of the user is translated to changes in the <input> elements of the <extent> element of the MapML and this acts as an HTML form, submitting the new values to the service that will generate a new MapML in response containing the relevant information to regenerate the user map view.

This clause describes how to generate a MapML file for a service. Let us review the concept of "tilematrix coordinates". Tilematrix coordinates are the pixel based coordinates that result from defining a grid that has a top-left corner in some far place and a size of the "scale denominator" (zoom level). They are long integer values. They are similar to "screen coordinates" (sometimes referenced a CRS0 or CRS1) (they share the pixel size) with the origin far away from the scene.

This subsection analyzes how WMS and WMTS can serve MapML. The authors of this document prefer the first alternative but in Testbed-13 a WMTS extension was implemented and demonstrated by CubeWerx into practice. WMS can perfectly respond with a MapML file with features and tiles as a response of a GetMap request. WMS could respond with a MapML file that includes features and tiles as a response to a GetMap (or a GetMapML) request; WMTS can do this only with tiles but it works. Both, WMS and WMTS services have the concept of 'scale'. WMTS has the scale denominator directly in the tilematrix definition while WMS uses it implicitly in the GetMap request by providing BBOX, WIDTH and HEIGHT (scale will be WIDTH/xmax-xmin*0.28/1000). In other words, WMS and WMTS are dealing with two CRSs at the same time and are able to translate between map coordinates (in a system commonly identified by a EPSG CRS) and screen coordinates (commonly known as CRS0 or CRS1) by applying scale and displacement. WMS can be easily modified or extended to replace screen coordinates by tilematrix coordinates

and use them in the BBOX and in the returned extent, tiles and features of a MapML file. WMS is supposed to serve a format that you can render on the screen and MapML is a format that is designed to be rendered in a HTML equivalent of the screen. On the contrary, WMTS has the tilematrix concept already in place but it was designed to serve a single tile while WMS is designed receive a BBOX and provide a format that includes the elements to render, and MapML could be an alternative for this. WMTS has the advantage that it is already aware of a tilematrixset structure while it has to be part of the extension in WMS.

In the end, the solution should be found in a middle ground between WMS and WMTS. We can extend WMS and WMTS to use tilematrix coordinates instead of pure screen coordinates. Conceptually, this could be a big change but will not be difficult to change in implementations. Actually, the fact that MapML is served with a WMS or with a WMTS extension should really be considered an implementation detail, since MapML users and client applications will not consider that and they will simply implement the extent action in the way MapML specifies and will see the MapML layer sources as opaque endpoint URLs. In both cases, the extent action implementation should correctly handle type=hidden inputs and add them to the KVP URL as HTML forms already do. This is consistent with the implementation of forms in HTML4 http://www.w3.org/TR/html401/interact/forms.html: "hidden: controlsAuthors may create controls that are not rendered but whose values are submitted with a form. Authors generally use this control type to store information between client/server exchanges that would otherwise be lost due to the stateless nature of HTTP. The INPUT element is used to create a hidden control.". This behavior is part of the HATEOAS (Hypermedia as the Engine of Application State) where servers and clients are exchanging the application state.

IMPORTANT	Recommendation to MapML: consider the inclusion of a sentence in the <extent> section saying that all types of input values (including hidden) should be submitted to the "action" following what HTML4 form recommends. In addition any MapML document resulting from the submission of an "action" in an "extent" should include the non identified keys as hidden elements of the extent to guarantee that a future "action" submission will still be able to add them.

8.4.1. WMS to serve MapML

The concept of a map in MapML shares a similarity with the definition WMS: "portrayal of geographic information as a digital image file suitable for display on a computer screen" (From WMS 1.3 Section 4.7: map). This section proposes an extension of WMS to support the direct creation of MapML from a WMS service. In addition, the details of the request derived from submitting an <extent> form to the server (that generates a KVP request concatenated to the *action* url) is conceptually very similar to the act of submitting GetMap request in KVP. Both request are aiming at getting a map back. Commonly, a WMS GetMap request returns a static image like a png or a jpeg but nothing prevents the server from returning other content formats, including some sort of markup language or JSON file. The only condition is that the returned format should be easy to render on the client screen. In practice, this means that the server should simplify the data to fit with the screen resolution and that the data can easy be represented as a vertical 2D view (bird view) of the scene. MapML fits perfectly with these requirements (in particular the decision of using 2D tilematrix coordinates in the vector coordinates and in the bbox). In the following table, we compare the parameters of the GetMap request with the MapML extent elements. The main

conclusion is that a GetMap request has many more parameters but they will have fixed values during the MapML interactions and that the common parameters can be assimilated to the ones in WMS but they are not directly compatible.

Table 4. Comparision of WMS GetMap parameters and MapML

WMS GetMap	**MapML extent**
Service=WMS	N/A
Request=GetMap	N/A
Version=1.3	N/A
Layer	N/A
Style	N/A
Format	N/A
BBOX	xmin, ymin, xmax, xmax
width, height	zoom
CRS	projection
Trasparent	N/A
BgColor	N/A
Time	N/A
Elevation	N/A

To avoid radical changes in the MapML standard candidate, we propose to extend WMS with a new request GetMapML and to extend MapML "extent" allowing specific hidden parameters in the section.

Conceptually, a WMS GetMapML request needs the same as GetMap but substituting the *bbox* parameter with the parameter list *xmin, ymin, xmax, xmax* (that are in tilematrix coordinates and not in the CRS coordinate as *bbox* is); *width, height* is substituted with the *zoom* parameter (that is related to the actually a tile matrix identifier that allow to matematically calculate the *width* and *height*) and CRS is substituted by *projection* (even if it actually represents a tile matrix set name). In this case WMS GetMapML will not support *Trasparent* and *BgColor* favoring transparency by default whenever possible.

The GetMapML request will, in addition, incorporate the following extra parameters:

Table 5. MapML proposed hidden parameter names and values

Name	**Value**
Service	"WMS"
Request	"GetMapML"
Version	"2.0"
Layer	a layer name (or coma separated names) in the GetCapabilities of the WMS

Name	Value
Style	a style name supported by the layer(s) in the GetCapabilities of the WMS or blank if it is the default style
Format	"text/mapml"

In addition to them, two additional parameters could be added: *Time* and *Elevation* for extra support to this two variables.

In response of a successful GetMapML request, the WMS service will produce a MapML that will have an extent section with the content similar to the illustration in this example:

```
    <extent action="/mapml/en/osmtile/cbmt?" enctype="application/x-www-form-
urlencoded" method="get" units="OSMTILE">
      <input max="8388608" min="0" name="xmin" type="xmin" value="8000"/>
      <input max="8388608" min="0.0" name="ymin" type="ymin" value="8000"/>
      <input max="8388608" min="0.0" name="xmax" type="xmax" value="9024"/>
      <input max="8388608.0" min="0.0" name="ymax" type="ymax" value="9024"/>
      <input max="15" min="0" name="zoom" type="zoom" value="15"/>
      <input name="projection" type="projection" value="OSMTILE"/>
      <input name="Service" type="hidden" value="WMS">
      <input name="Request" type="hidden" value="GetMapML">
      <input name="Version" type="hidden" value="2.0">
      <input name="Layer" type="hidden" value="layer_name">
      <input name="Style" type="hidden" value="">
      <input name="Format  type="hidden" value="text/mapml"
    </extent>
```

In addition, the MapML document can contain all references to obtain the relevant tiles present in the extent bounding box in <tile> elements. These tile elements can be references to WMTS services, another kind of tile services, or full KVP WMS GetMap requests to get maps that will have equivalent characteristics in terms of bbox, width, height and format of the expected tiles. Secondly, the MapML can contain also features in <feature> elements that are visualized in the map as vector features.

If this proposal is accepted by the MapML community as a service that generates MapML files, it will be necessary to consider what concrete changes are needed to be done in the GetCapabilities response to expose the possibility of having GetMapML requests. The minimum changes in the ServiceMetadata document are:

- Expose the support for the new operation and describe it.
- List the names of the tilematrixsets each layer supports.
- Describe the tilematrixsets in the ServiceMetadata document (this should not be necessary if the tilematrixset are global and described in another standard or in a catalogue).
- Publish the tilematrissetlimits for each layer.

In addition to this, the MapML work done in this testbed and documented in this document to allow

simple queries for more information in a point of a map, should be aligned with the GetFeatureInfo.

The following UML diagram summarizes the approach suggested and gives emphasis to the two internal functionalities that need to be implemented to transform WMS into a MapML service. The most apparent difference is the existence of the GetMapML operation. The "BBOX to WMSC tile URLs" is aware of the tile matrix set concept and is able to transform the BBOX and the zoom level into a sequence of WMS requests that can populate the bounding box, but perfectly fits the requirements of a tile in terms of width and height. To be able to provide the features in the tile matrix coordinates, a "CRS to tile matrix coordinates (internal)" functionality is required. Based on the zoom level, transforms the WFS coordinates into a tile matrix coordinates and transforms a GML file into the right format for the MapML.

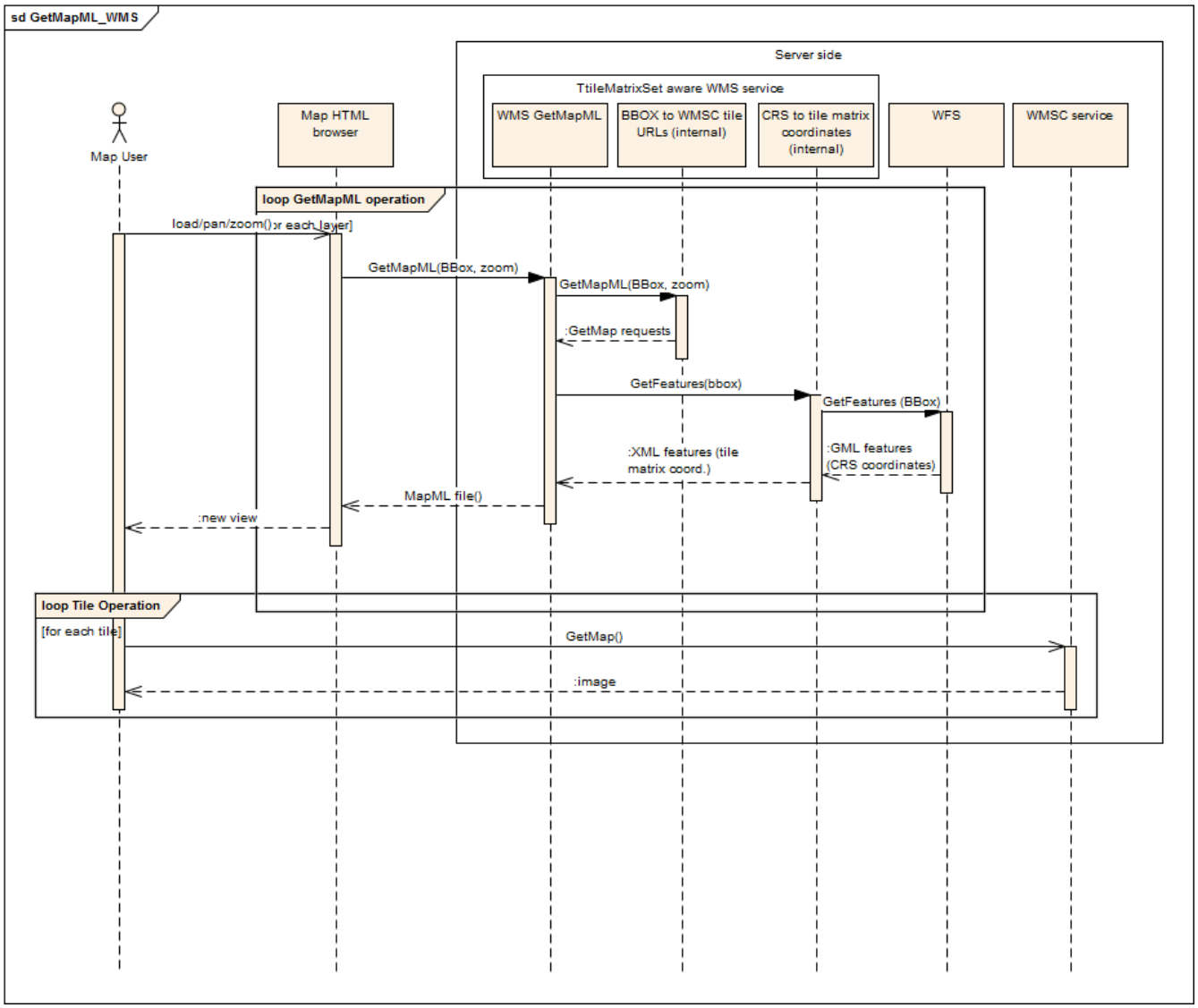


Figure 14. MapML interaction UML sequence diagram based on the adaptation of a WMTS that works with a WFS to provide the feature part.

NOTE

Recommendation to new testbeds: The use of WMS to provide MapML files on demand could be investigated in a next Textbed. The use of WMTS was investigated in this testbed but it is limited to provide tiles (and not features) while WMS will not have this limitation.

8.4.2. WMTS to serve MapML

To satisfy the NR103 deliverable for OGC Testbed-13, CubeWerx implemented a MapML server compliant to the "Map Markup Language" current draft specification at "http://maps4html.github.io/MapML/spec/[http://maps4html.github.io/MapML/spec/]", and compatible with the "<web-map> HTML Element proposal" at "http://maps4html.github.io/HTML-Map-Element/spec/[http://maps4html.github.io/HTML-Map-Element/spec/]".

The implementation was made as an unofficial extension to the WMTS 1.0.0 specification (OGC 07-057r7) by introducing a "GetMapML" operation to the demo WMTS server at: https://tb13.cubewerx.com/cubewerx/cubeserv/default [https://tb13.cubewerx.com/cubewerx/cubeserv/default]

Since GetMapML is not officially a WMTS operation, the WMTS capabilities document is typically bypassed (even though, as a matter of completeness, it does list GetMapML as one of its supported operations). In MapML, each layer has its own endpoint. For this particular server, the endpoint of each layer is (RESTful URL):

https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/<layerName [https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/%3ClayerName]>

An alternative longer form for the endpoints using SOA KVP encoding is: https://tb13.cubewerx.com/cubewerx/cubeserv/default?SERVICE=WMTS&VERSION=1.0.0&REQUEST=GetMapML&LAYER=<layerName [https://tb13.cubewerx.com/cubewerx/cubeserv/default?SERVICE=WMTS&VERSION=1.0.0&REQUEST=GetMapML&LAYER=<layerName]>

This implementation provides three layers:

- "gtopo30.gtopo30" - a low-resolution global elevation coverage
- "NAIP.NAIP" - some high-resolution satellite imagery in Texas and Mexico
- "Ice_Concentration_20150101.Ice_Concentration_20150101" - imagery of ice concentration in the arctic circle

As such, the three MapML endpoints are:

- https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/gtopo30.gtopo30 [https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/gtopo30.gtopo30]
- https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP [https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP]
- https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/Ice_Concentration_20150101.Ice_Concentration_20150101 [https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/Ice_Concentration_20150101.Ice_Concentration_20150101]

All three layers are available in the OSMTILE, CBMTILE and APSTILE projections as defined by the "Map Markup Language" draft specification, with the exception of "NAIP.NAIP" which is not available in APSTILE due to its southern geographic location.

To illustrate the implementation, CubeWerx have set up a fast demonstration map for each of the three projections:

- https://tb13.cubewerx.com/mapmlDemo/osmtile.html [https://tb13.cubewerx.com/mapmlDemo/osmtile.html]
- https://tb13.cubewerx.com/mapmlDemo/cbmtile.html [https://tb13.cubewerx.com/mapmlDemo/cbmtile.html]
- https://tb13.cubewerx.com/mapmlDemo/apstile.html [https://tb13.cubewerx.com/mapmlDemo/apstile.html]

As an example of the inner workings of MapML, one of the MapML accesses that is performed by the CBMTILE demo is:

https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP?xmin=105767&ymin=130523&xmax=107057&ymax=131123&zoom=9&projection=CBMTILE
[https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP?xmin=105767&ymin=130523&xmax=107057&ymax=131123&zoom=9&projection=CBMTILE]

This particular URL returns the following document (with extra formatting added for readability):

```
<mapml>
    <head>
      <title>NAIP</title>
    </head>

      <extent units="CBMTILE"  action=
"https://tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP[https:
//tb13.cubewerx.com/cubewerx/cubeserv/default/wmts/1.0.0/mapML/NAIP.NAIP]" method="
get" enctype="application/x-www-form-urlencoded">
      </extent>
      <tile col="415" row="510"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
510/415.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/510/415.jop"/]>
      <tile col="416" row="510"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
510/416.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/510/416.jop"/]>
      <tile col="415" row="511"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
511/415.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/511/415.jop"/]>
      <tile col="416" row="511"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
511/416.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/511/416.jop"/]>
      <tile col="415" row="512"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
512/415.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/512/415.jop"/]>
      <tile col="416" row="512"  src=
"https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/default/CBMTILE/9/
512/416.jop%22/[https://tb13.cubewerx.com/cubewerx/OpenImageMap/tilesets/NAIP/NAIP/def
ault/CBMTILE/9/512/416.jop"/]>
</mapml>
```

We found that implementing MapML as a WMTS extension was a natural fit, since it allowed us to reuse the tiling infrastructure of our WMTS codebase and to deploy it more easily.

The following UML diagram summarizes the approach taken and gives emphasis to the two internal functionalities that need to be implemented to transform WMTS into a MapML service. The most apparent difference is the existence of the GetMapML operation that allows for a bounding box input. To interpret this the "BBOX (CRS) to tile URLs (tilematrixset coordinates)" internal functionality is required. It transforms the BBOX and the zoom level into a sequence of WMTS requests that can populate the bounding box, considering the tile matrix that corresponds to the right scale denominator. To be able to provide the features in the tile matrix coordinates, a "CRS to tile matrix coordinates (internal)" functionality is required. Based on the zoom level, transforms the WFS coordinates into a tile matrix coordinates and transforms a GML file into the right format for the MapML.

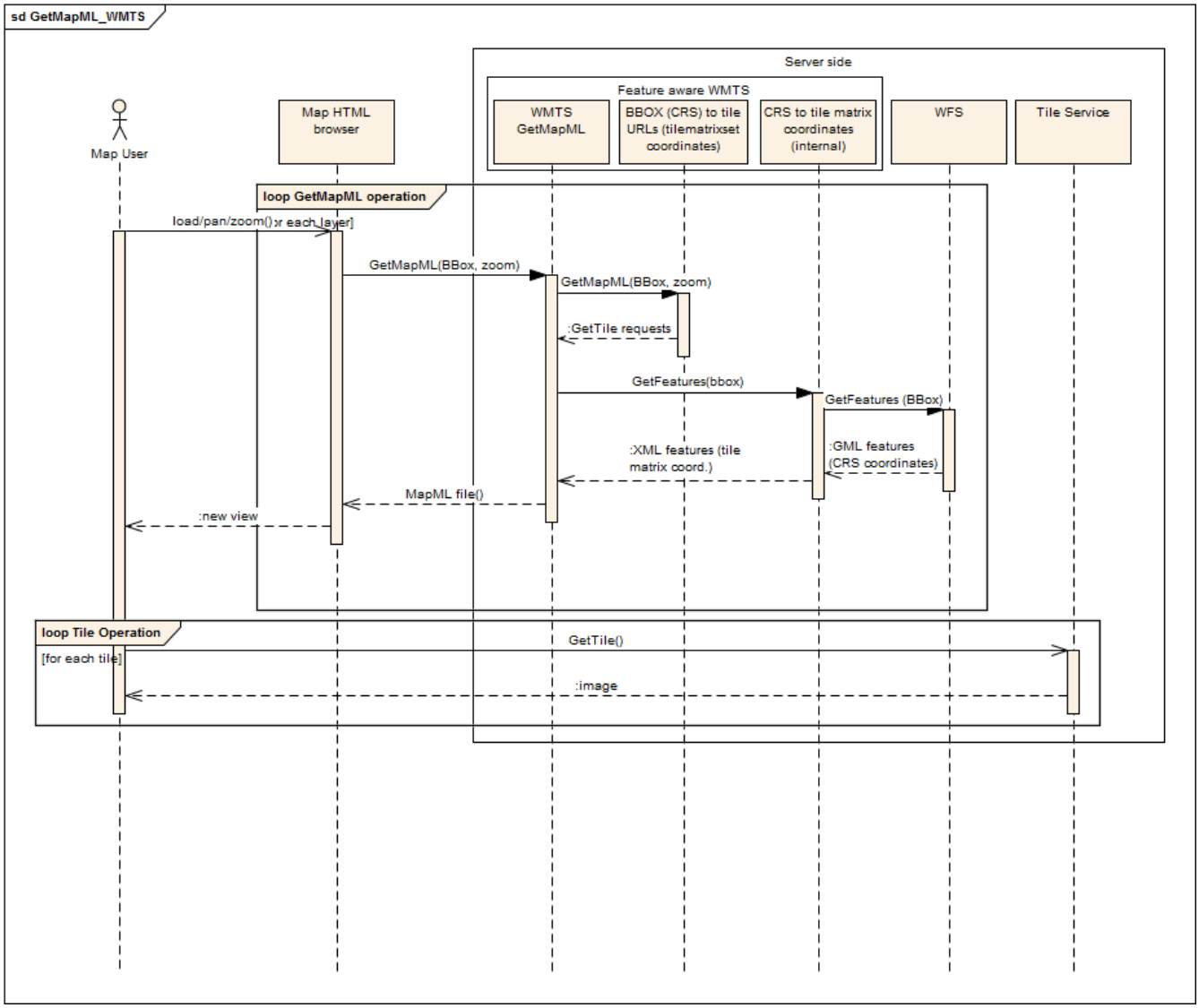


Figure 15. MapML interaction UML sequence diagram based on the adaptation of a WMTS that works with a WFS to provide the feature part.

8.4.3. Integration with other OGC services.

WMS and WMTS are services that are dealing with scale and are able to translate between map coordinates (in a system commonly identified by a EPSG CRS) and screen coordinates (commonly known as CRS0 or CRS1).

In contrast, WCS and WFS are not supposed to associate/translate map coordinates to screen coordinates so they are conceptually different. You could argue that WCS and WFS are able to make CRS transformations on the fly so they could also do map coordinate to tile matrix coordinates transformations. This is true but this forces us to recognize tile matrix coordinates as true CRSs. We could reuse some of WCS.SWG work on parametric CRS's where parameters could "travel" in the actual CRS URL (in our case the zoom level and eventually the top left corner). Parametric CRS are not very popular in CRS because many believe that a URL would be opaque and by definition a parametric URL is not opaque.

Alternatively, we propose to use a WMS or a WMTS as a facade or a broker that will serve the client a dialog with a WCS or WFS for requesting data. This is not a new idea. At the time that SDL was defined as an "extension" of WMS, there was the need to define the data morel behind the layer (it

was not necessary in a pure WMS). You needed the data model to encode the symbology rules. A layer needed to be defined as "coming from feature types" or "coverage types". If you do that you immediately recognize that a WMS can have a WFS at the backend (or as a data repository). This means that when a WMS request comes to the service it translates it to a WFS request that is sent to the backend. When the WFS responses, the WMS takes the features combines the with styles, coordinates are transformed to CRS0/CRS1 rendered and served. The same for a WCS coverage. An illustration of this possibility can be seen here http://slideplayer.com/slide/4632571/15/images/41/Architecture+using+WMS,+WFS,+and+SLD.jpg in an slide of a presentation prepared by the George Percivall talking about the OGC reference model http://slideplayer.com/slide/4632571 and reproduced below for convenience. Actually the illustration also suggests another reason why this architecture could be convenient: that capacity of WMS to incorporate styles using Symbology Encoding rules that are applied to features that are provides "styleless" when coming directly for a WFS. The same architecture can be proposed to serve a MapML if we substitute CRS0/CRS1 for tilematrix coordinates and "render" by "encoding in MapML".

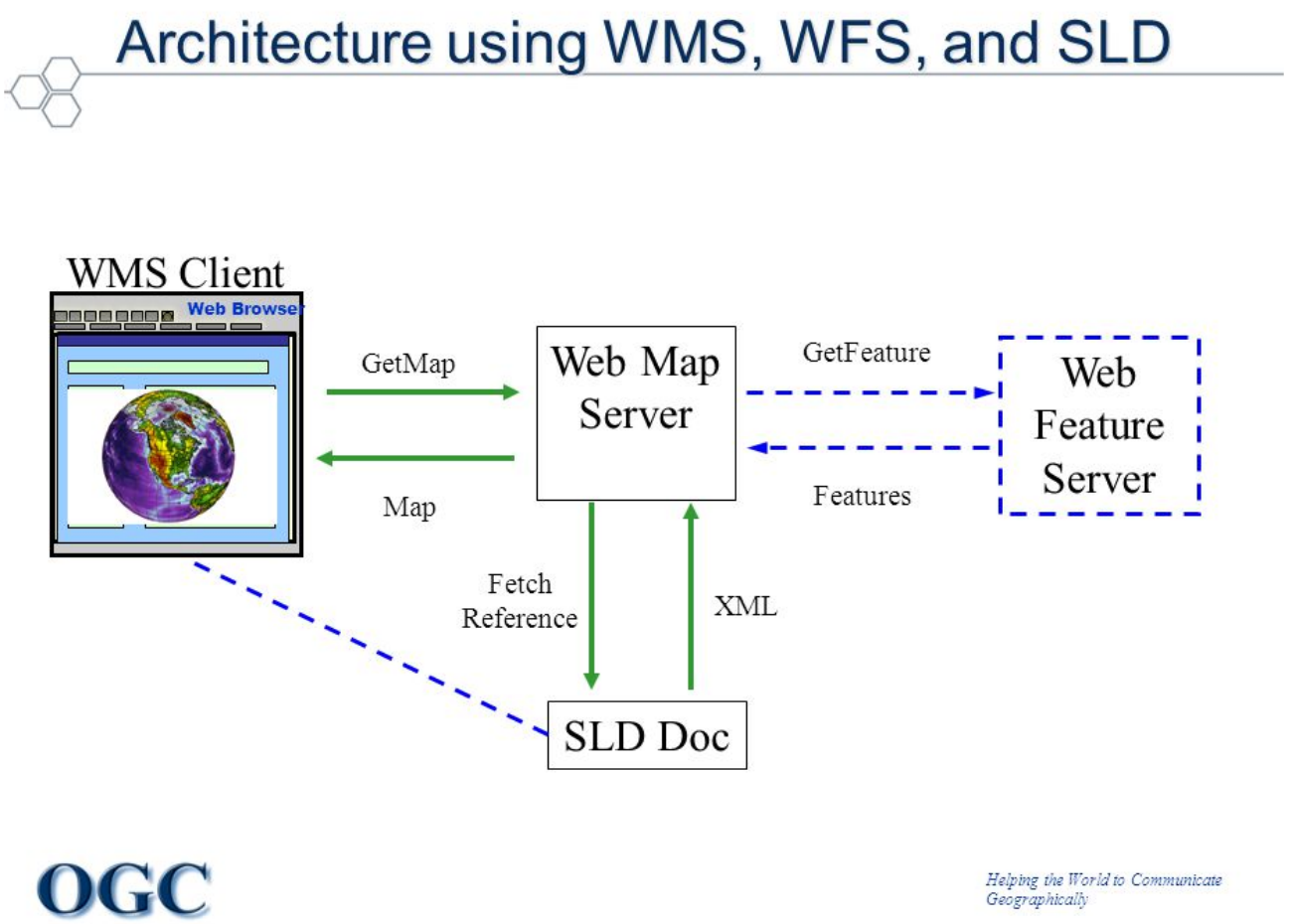


Figure 16. Architecture that uses a WMS as a front end of WFS data that is symbolized by a SLD service (George Percivall)

8.5. Map search

A MapML service might want to allow service consumers to search within the contents of a service for a place name or an address, over which the map is repositioned when a selection is made. MapML should provide (markup) facilities to enable such searches without forcing the client to download large amounts of geospatial data.

We can consider this navigation mode a variant of the previously described "Map service" since it will also depend on a service that is able to respond to an <extent> action. The presence of text in an <input> of the *search* type will generate a location request. As a result, we expect from the server a MapML document with features that match the text *search* and an extent that encompasses all of them. This means that, most probably, the MapML file will not contain *tiles* but *features*. It is up to the client to decide how to present this information to the user. Non-exclusive options are:

- Features will not replace the previously presented features (the previously present MapML files) but will act as if a "results" layer has been added to the map view. In that sense, we expect that the returned MapML includes a title that explains the fact that the content is the results of a search.
- A new window or division can be presented with a list of textual representations of the features that match the request. A click in one of them could move the map to zoom into the selected feature.

8.5.1. Integration with OGC standards

Currently, there are three OGC standards that can specifically address this request: GeoOpenSearch, CSW and WFS+Filter. Following the discussion about the MapML service above, that favors the use of a modified WMS or WMTS to serve tiles, this functionally cannot be implemented by a WMS or a WMTS without having to change the scope of the service in a way that moves it too far away for the original scope. This is the reason why we consider that this functionally should be provided by another service type that will act in tandem with the WMS or WMTS that is providing the main service.

GeoOpenSearch

This standard is easy to use because it does not contemplate complicated queries, leaving some degree of freedom to the server on how to handle the request. In essence, the server receives a textual query and an Atom feed is returned with the results.

Since an OpenSearch request is based on providing a URL template, it seems possible to create this template to describe the current way MapML formulates search requests. We are suggesting that a MapML enabled OpenSearch service could serve Atom files or MapML files as a result of the query. Consideration for OpenSearch geo and temporal extension could also be taken into account.

IMPORTANT	Recommendation: Next testbed could experiment with the use of OpenSearch for textual search that returns a MapML file as a response of the query.

Other OGC standards for search.

CSW is another alternative service that is able to support geospatial textual queries. It could be good for receiving and processing the request, but normally, a CSW record contains a reference to a dataset and not to a feature. For that reason, this alternative is discarded.

WFS with FILTER encoding seem another alternative for implementing the search functionality, at least in the backend. A MapML search query can be easily converted into a WFS filter and the WFS

can return the necessary features that can be then encoded in the right format by a middleware.

Chapter 9. TileMatrixSet specification

The current MapML specification defined some projections with a set of numeric zoom levels. These concepts are almost equivalent to the tilematrixset and tile matrix identifier in WMTS respectively. It could be good that a future version of MapML could use the OGC WMTS naming for these concepts as well as other useful tilematrixset definitions. To make that possible, during Testbed 13 the OGC 17-083 Abstract Tile Matrix Set standard has been drafted and presented to the WMS.SWG. It is foreseen that this candidate will start the standardization process in the near future. It is expected that this future OGC standard can be referenced by future versions of GeoPackage, WMTS, MapML and others.

The current draft version of the document has the following structure:

- 6. Tile Matrix Set concept
  - 6.1 Tile Matrix
    - 6.1.1 Tile matrix in a two dimensional space
  - 6.2 Tile Matrix Set
  - 6.3 Well-known scale sets
  - 6.4 Tile based coordinates in a tile matrix set
  - 6.5 Tile matrix set limits
- 7. TileMatrixSet 2D requirements class
- 8. TileMatrixSetLimits requirements class
- 9. XML encoding of a TileMatrixSet
- 10. XML encoding of a TileMatrixSetLimits
- 11. JSON encoding of a TileMatrixSet
- 12. JSON encoding of a TileMatrixSetLimits
- Annex A Conformance Class Abstract Test Suite (Normative)
- Annex B XML Schema Documents (Normative)
- Annex C Well-known scale sets (Informative)
- Annex D TileMatrixSet definitions (Informative)
  - D.1 World Web Mercator Quad TileMatrixSet definition
  - D.2 World Mercator WGS84 Quad TileMatrixSet definition
  - D.3 World CRS84 Quad TileMatrixSet definition
  - D.4 World Transverse Mercator WGS84 Quad family TileMatrixSet definition
  - D.5 Arctic WGS 84 Polar Stereographic Quad TileMatrixSet definition
  - D.6 Antarctic WGS84 Polar Stereographic Quad definition
  - D.7 European ETRS89 LAEA Quad TileMatrixSet definition
  - D.7 Canadian Lambert Conformal Conic TileMatrixSet definition

- Annex E Example XML documents (Informative)

The most recent document at the time of writing this document can be found here: https://portal.opengeospatial.org/index.php?m=projects&a=view&project_id=274&tab=2&artifact_id=75046

IMPORTANT	Recommendation to MapML: consider adopting the OGC 17-083 Tile Matrix Set standard produced in WMS group as a result of this testbed activity. Remove the description to projections definitions in the MapML document and refer to the OGC 17-083 Tile Matrix Set standard.

Appendix A: Change Requests to MapML

This annex describes some Change Requests(CR) that had been submitted to MapML. Other CR are available throughout this document.

- Please use "Tile Matrix Set" instead of "tiled coordinate reference system" to align with the WMTS specification. use the expression "tilematrix coordinates" instead of "tiled coordinates". If you see new version of the tilematrixset specification I specifically introduced this concept in opposition of "tile coordinates" that, to me, are the ones in WMTS GetFeature info: coordinates inside a tile and not form the origin of the tile matrix, that is what the MapML wants to express.
- In the document there is still a place (describing the <coordinates>) that says: "for example in the tile element, the attributes x and y are used" should say "for example in the tile element, the attributes col and row are used".
- Please clarify the reference system of the <coordinates> element. I believe it is "tilematrix coordinates" but there is no way to be sure with the current text. In addition, I saw one example in WGS84 lat/long somewhere. Actually this is a very important aspect for interoperability with other OGC standards.
- Please clarify the reference system of the <bbox> element. I believe it is "tilematrix coordinates" but there is no way to be sure with the current text.
- Remove WGS84 from this sentence " For WGS84, the exterior should be counterclockwise and holes should be clockwise." It creates uncertainty on the other projections.
- EPSG::4326 and axes order Longitude x, Latitude y in the same raw of the projection table contradicts the EPSG database and will only create confusion. Please refer to the OGC CRS84 as the GeoJSON specification does.
- State the mime type of a MapML document. If there is no common mime type it could be good to know the generic one where it is encompassed

Appendix B: Revision History

Table 6. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
May 21, 2017	J.Maso	.1	all	IER: initial version with the main structure of the document
Oct 17, 2017	J.Maso	.2	all	DER: Draft version for OGC TC WMS.SWG revision

Appendix C: Bibliography

OGC 12-080r2 OWS Context Conceptual Model v1.0 [https://portal.opengeospatial.org/files/?artifact_id=55182]

OGC 12-084r2 OWS Context Atom Encoding v1.0 [https://portal.opengeospatial.org/files/?artifact_id=55183]

OGC 12-007r2 KML v 2.3 [http://docs.opengeospatial.org/is/12-007r2/12-007r2.html]

OGC 07-057r7 Web Map Tile Service Implementation Standard v 1.0 [http://portal.opengeospatial.org/files/?artifact_id=35326]

OGC 06-042 Web Map Service (WMS) Implementation Specification v 1.3 [http://portal.opengeospatial.org/files/?artifact_id=14416]

RFC 6570. URI Template. Internet Engineering Task Force (IETF) Request for Comments [https://tools.ietf.org/html/rfc6570]