RC24318 (W0707-145) July 23, 2007

Computer Science

IBM Research Report

Map Feature Language (MFL) Specification, V1.0

Jonathan P. Munson

IBM Research Division

Thomas J. Watson Research Center

P.O. Box 704

Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

IBM UCL and IBM WRC

TOPAZ

Map Feature Language (MFL) Specification
Version 1.0

Copyright Notice

TABLE OF CONTENTS

1 Introduction

The Map Feature Language, MFL, is an XML language that enables content providers to provide map-based content without supplying the entire map. It supports a model where content from different providers can be overlaid on the same map. MFL was defined as part of the *MapWeb* project at IBM's T. J. Watson Research Center.

MFL was designed to enable commonly portrayed map features to be defined textually. Currently, MFL supports the following kinds of features: icons, point-shapes, text, linestrings, polygons, multi-linestrings, multi-polygons, and image tiles. The geometry of each feature is described in geographic coordinates. (MFL allows the geometries to be described in any well-known geographic coordinate system, but MFL viewers are only required to support the common latitude and longitude of WGS-84 coordinates.) The presentation of each feature is described using presentation attributes such as color, line-width, font and font-size. This integration of geographic coordinates with presentation information is what gives MFL its unique role as presentation language for map features.

Feature geometry in MFL is specified with elements from the Basic Features specification of the Geography Markup Language (GML), defined by the Open Geospatial Consortium. Implementors of MFL interpreters should consult the GML specification directly [GML]. For convenience, some elements of GML are described here. Readers should note that the descriptions here are based on GML 2, whereas the latest version is GML 3. Some elements of GML 2 are deprecated in GML 3.

1.1 MFL Basic and MFL Tiny

Following the example of SVG Mobile [SVGM], two subsets of MFL are defined: MFL Basic and MFL Tiny. MFL Basic is a proper subset of MFL and is designed to accommodate the limitations of devices such as PDAs. MFL Tiny is a proper subset of MFL Basic and is designed to accommodate the further limitations of devices such as cell phones.

Features that are not supported in MFL Basic or MFL Tiny are noted as such in their description.

2 Structural Elements

2.1 The FeatureSet Element

`FeatureSet` is the root element of an MFL document.

Attributes

`id`	(Optional) Uniquely identifies this feature-set among all others returned by a MapWeb Feature Server from the same URL. Useful for distinguishing between zoom levels for a given feature-set.

Sub-elements

`title`	A title for the feature-set.
`scaleRange`	See Section 2.1.1.
`description`	See Section 0.
`style`	See Section 2.1.3.
`Feature`	See Section 2.2.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Thu Jan 20 10:14:42 KST 2005 -->
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>FeatureSet having a number of features</title>
  <Feature id="1">
    <label>LineString feature</label>
    <gml:LineString>
      <gml:coord><x>126.97245025634766</x><y>37.554317474365234</y></gml:coord>
      <gml:coord><x>126.97300720214844</x><y>37.55436325073242</y></gml:coord>
    </gml:LineString>
    <style>
      <fillColor rgb="255,0,51" opacity="50%"/>
      <lineColor rgb="255,51,51" opacity="50%"/>
      <lineWeight>8.0</lineWeight>
    </style>
  </Feature>
  <Feature id="2">
    <label>LineString feature</label>
    <gml:LineString>
      <gml:coord><x>126.97566986083984</x><y>37.559471130371094</y></gml:coord>
      <gml:coord><x>126.97586822509766</x><y>37.55970001220703</y></gml:coord>
    </gml:LineString>
    <style>
      <fillColor rgb="255,0,51" opacity="50%"/>
      <lineColor rgb="255,51,51" opacity="50%"/>
      <lineWeight>8.0</lineWeight>
    </style>
  </Feature>
</FeatureSet>
```

2.1.1 scaleRange

The `scaleRange` element gives the range of view scales (in pixels-per-degree-latitude) that the feature-set creator considers acceptable for this feature set. If not included, the feature set is considered acceptable at any view scale. A MapWeb Feature Server is expected to return, for any given feature set, scale ranges that are contiguous.

Attributes

`gt`	(Mandatory) The view scale should be greater than this value.
`le`	(Mandatory) The view scale should be less than or equal to this value.
`eq`	(Optional—used when the feature-set consists of MFLImage

	elements) The exact scale of the images in the feature set.

The `scaleRange` element enables viewers and feature servers using the MWFS protocol (Section 5) to implement a "distributed zoom" capability. Viewers implement limited scaling of features within the scale ranges of the feature-sets that are loaded, and when the viewer scale goes outside the scale range of any feature-set, the viewer re-requests the feature-set from the feature server it came from, including in the request the new scale factor.

2.1.2 description

The `description` element provides an informational description of the feature or feature-set. If the `href` attribute is included, the MFL viewer is expected to offer the user means of loading the HTML indicated by the URL.

Attributes

`href`	(Optional) An absolute or relative URL that links to extended information or other content related to the feature or feature set.

2.1.3 style

The style element specifies presentation attributes for a feature, or if included in a feature-set, for all features included in the feature set. A feature-set style is overridden, in whole, by a feature style.

Sub-elements

`fillColor`	The fill color of a closed shape. See Section 2.1.4. Default is none.
`lineColor`	The color of a shape. See Section 2.1.4. Default is black.
`textColor`	The color of a shape's text, if a text shape. See Section 2.1.4. Default is black.
`lineWeight`	The width (in pixels, floating-point) of a shape's lines. Default is 1.0.

2.1.4 Color Elements

Attributes

`rgb`	(Mandatory) A triple of the form [0–255],[0–255],[0–255] which represents, in order, the red component value, the green component value, and the blue component value.
`opacity`	(Optional) The opacity of the color, as a percent. 100% is fully opaque; 0% is fully transparent. **Basic/Tiny**: Color transparency is not supported in the MFL Basic or MFL Tiny profiles.

2.2 The Feature Element

The Feature element defines a visual map feature. A Feature element may contain multiple renderable elements, which are described in Section 0. An MFL viewer must render the renderable elements in the order in which they appear in the MFL. The style element applies to all renderable elements.

Sub-elements

`label`	A text-only element containing a label for the feature-set. A viewer MUST offer means to display the label when the feature is selected or otherwise indicated by the user.
`description`	See Section 0.
`style`	See Section 2.1.3.
`attribute`	See Section 2.2.1.
`Icon`	See Section 3.1.
`Image`	See Section 3.2.
`LineString`	See Section 3.3.
`MultiLineString`	See Section 3.4.
`Point`	See Section .
`PointShape`	See Section 3.5.
`Polygon`	See Section 3.6.
`MultiPolygon`	See Section 3.7.
`Text`	See Section 3.8.

2.2.1 attribute Element

The `attribute` element is used to describe a feature through a set of attributes. These can be used by a viewer to filter the features shown on the map. This usage is optional, and is not further described in this document.

Attributes

`name`	The name of the attribute.
`value`	The value of the attribute.
`type`	The type of the attribute, controlling how the `value` attribute is interpreted. `type` may be one of: `Integer`, `Float`, `String`, or `Date`. If not specified, the type is assumed to be `String`.

3 Renderable Elements

Renderable elements are those that are drawn on the map. A Feature element contains one or more renderable elements. Renderable elements in MFL are distinguished from those in vector-based drawing languages such as SVG in their use of geographical coordinates for location. A map viewer uses a coordinate transformation to transform geographical coordinates into screen coordinates. An example of such a transformation is given in Section 4.1.

Some renderable elements are defined by GML [cite], and others are defined by MFL. Elements defined by GML are noted as such in the element descriptions below, and should have the "`gml:`" namespace prefix.

3.1 Icon

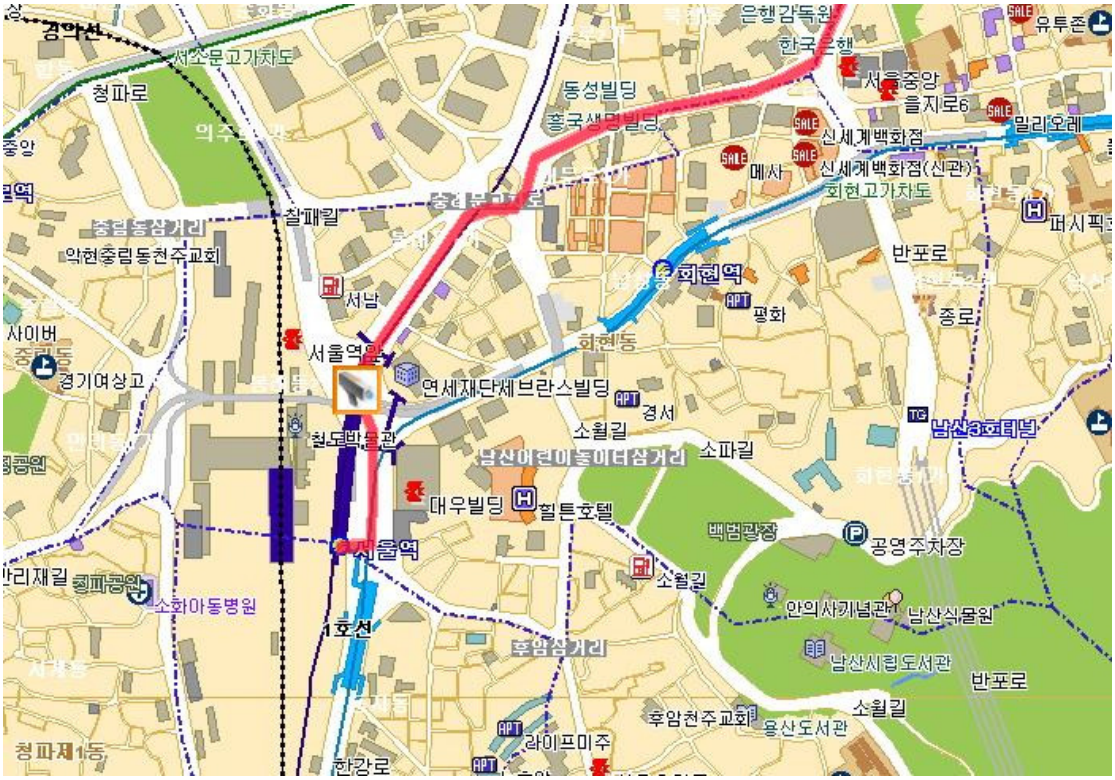
The `Icon` element is used to position an image at a geographic point. The image is drawn so that it is centered on the given point. The image may be supplied via a relative or absolute URL or as Base-64 encoded image data. If supplied as a URL, the viewer must first resolve the URL to an absolute URL, then load the image via the scheme indicated in the URL (`http`, `file`, etc.). Image caching can significantly improve performance with icon images because they may often be repeated in a feature-set.

Sub-elements

`href`	A text-only element containing an absolute or relative URL address from which to load the icon image.
`iconData`	The image data in Base 64 encoding.
`Point`	The location of the icon. See Section 3.1.1.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>Icon feature created on Thu Jan 20 15:44:39 KST 2005</title>
  <Feature id="1">
    <label>Icon feature</label>
    <Icon>
      <href>map_icon_SpeedTrap.GIF</href>
      <gml:Point>
        <gml:coord><x>126.97272491455078</x><y>37.55672073364258</y></gml:coord>
      </gml:Point>
    </Icon>
  </Feature>
</FeatureSet>
```



3.1.1 Point

Point is a GML element for representing a pair of coordinates. It contains a single `coord` element.

Sub-element

`coord`	Gives one pair of coordinates, as `<x>…</x><y>…</y>`, where `x` and `y` are WGS-84 longitude and latitude, respectively.

3.2 Image

The `Image` element is used to draw a square image within a given rectangle expressed in geographic coordinates. The image may be supplied via a relative or absolute URL or as Base-64 encoded image data. If supplied as a URL, the viewer must first resolve the URL to an absolute URL, then load the image via the scheme indicated in the URL (`http`, `file`, etc.). Image caching can be used to improve performance.

`Image` elements differ from `Icon` elements in that the image must be scaled to fit within the given geographical coordinates. The geographical coordinates are first transformed to screen coordinates using the viewer's projection transformation, then the image is drawn within the resulting rectangle.

Sub-elements

`href`	A text-only element containing a URL address from which to load the

	icon image.
`imageData`	The image data in Base 64 encoding.
`Box`	A GML element containing the bounding box of the image. See Section 3.2.1.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>Seoul Basemaps</title>
  <Feature id="1">
    <Image>
      <href>northern_seoul2.JPG</href>
      <gml:Box>
        <gml:coord><x>126.03258055555555</x><y>37.556605555555555</y></gml:coord>
        <gml:coord><x>126.99630833333333</x><y>37.576966666666666</y></gml:coord>
      </gml:Box>
    </Image>
    <label>Northern Seoul basemap</label>
  </Feature>
  <Feature id="2">
    <Image>
      <href>southern_seoul.JPG</href>
      <gml:Box>
        <gml:coord><x>127.08558055555555</x><y>37.49928055555556</y></gml:coord>
        <gml:coord><x>127.11639444444444</x><y>37.51956388888889</y></gml:coord>
      </gml:Box>
    </Image>
    <label>Southern Seoul basemap</label>
  </Feature>
</FeatureSet>
```

The above shows the use of the `Image` element in providing a basemap based on tiles. (Basemap tiles would normally be contiguous, although in this example they are not.)

3.2.1 Box

Box is a GML element representing a bounding area. It contains exactly two `coord` elements.

`coord`	Gives one pair of coordinates, as `<x>…</x><y>…</y>`, where `x` and `y` are WGS-84 longitude and latitude, respectively.

3.3 LineString

The `LineString` element is a GML element, and should have the "`gml:`" namespace prefix. A `LineString` may contain either a sequence of `coord` elements, or a single `coordinates` element. See [GML] for the definitive definition.

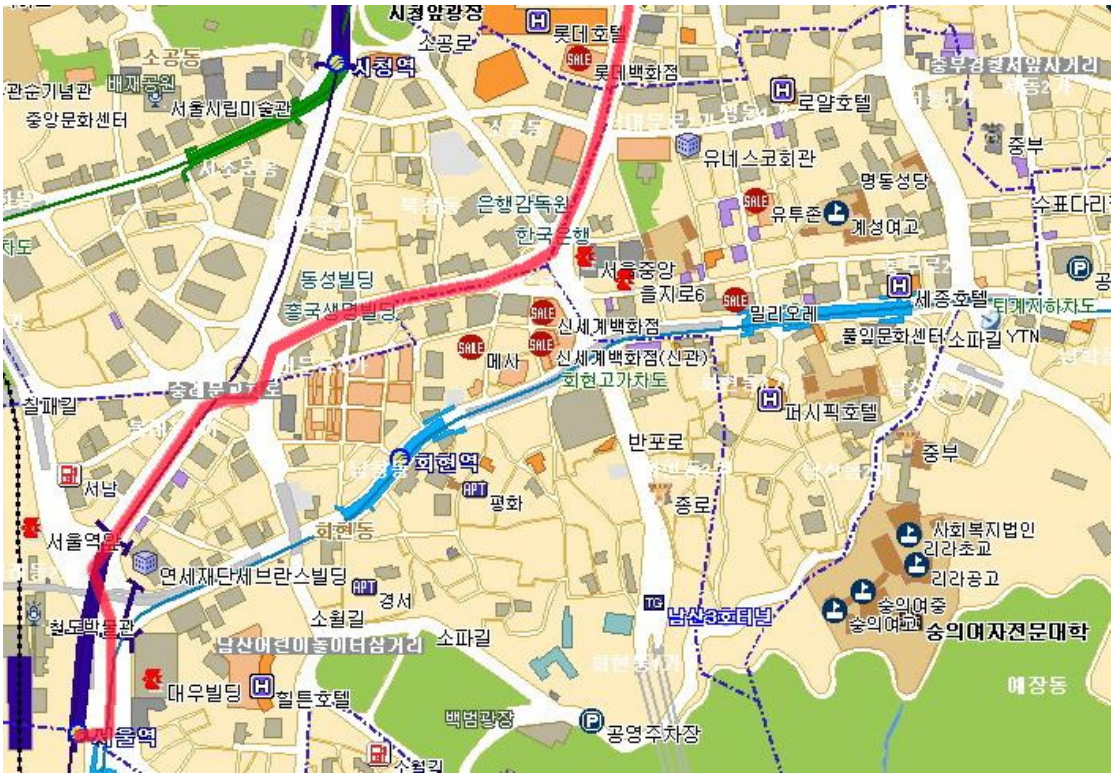
Sub-elements

`coord`	Gives one pair of coordinates, as `<x>…</x><y>…</y>`, where `x` and `y` are WGS-84 longitude and latitude, respectively.

coordinates	An alternative format for coordinates, in which a sequence of coordinates is given as a single text element. See [GML] for a description of this element.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Thu Jan 20 10:14:42 KST 2005 -->
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>LineString feature created on Thu Jan 20 10:12:52 KST 2005</title>
  <Feature id="1">
    <label>LineString feature</label>
    <gml:LineString>
      <gml:coord><x>126.97245025634766</x><y>37.554317474365234</y></gml:coord>
      <gml:coord><x>126.97300720214844</x><y>37.55436325073242</y></gml:coord>
      <gml:coord><x>126.97303771972656</x><y>37.55604934692383</y></gml:coord>
      <gml:coord><x>126.97267150878906</x><y>37.55685043334961</y></gml:coord>
      <gml:coord><x>126.97476959228516</x><y>37.55933380126953</y></gml:coord>
      <gml:coord><x>126.97516632080078</x><y>37.5594482421875</y></gml:coord>
      <gml:coord><x>126.97566986083984</x><y>37.559471130371094</y></gml:coord>
      <gml:coord><x>126.97586822509766</x><y>37.55970001220703</y></gml:coord>
      <gml:coord><x>126.97605895996094</x><y>37.56022644042969</y></gml:coord>
      <gml:coord><x>126.977294921875</x><y>37.560726165771484</y></gml:coord>
      <gml:coord><x>126.97852325439453</x><y>37.56100082397461</y></gml:coord>
    </gml:LineString>
    <style>
      <fillColor rgb="255,0,51" opacity="50%"/>
      <lineColor rgb="255,51,51" opacity="50%"/>
      <lineWeight>8.0</lineWeight>
    </style>
  </Feature>
</FeatureSet>
```



3.4 MultiLineString

`MultiLineString` is a GML element that is a container for a set of `LineString` elements. See [GML] for the definition of this element.

3.5 PointShape

The `PointShape` element is used to draw geometric shapes at geographic locations. Ellipse and rectangle are the currently defined shapes.

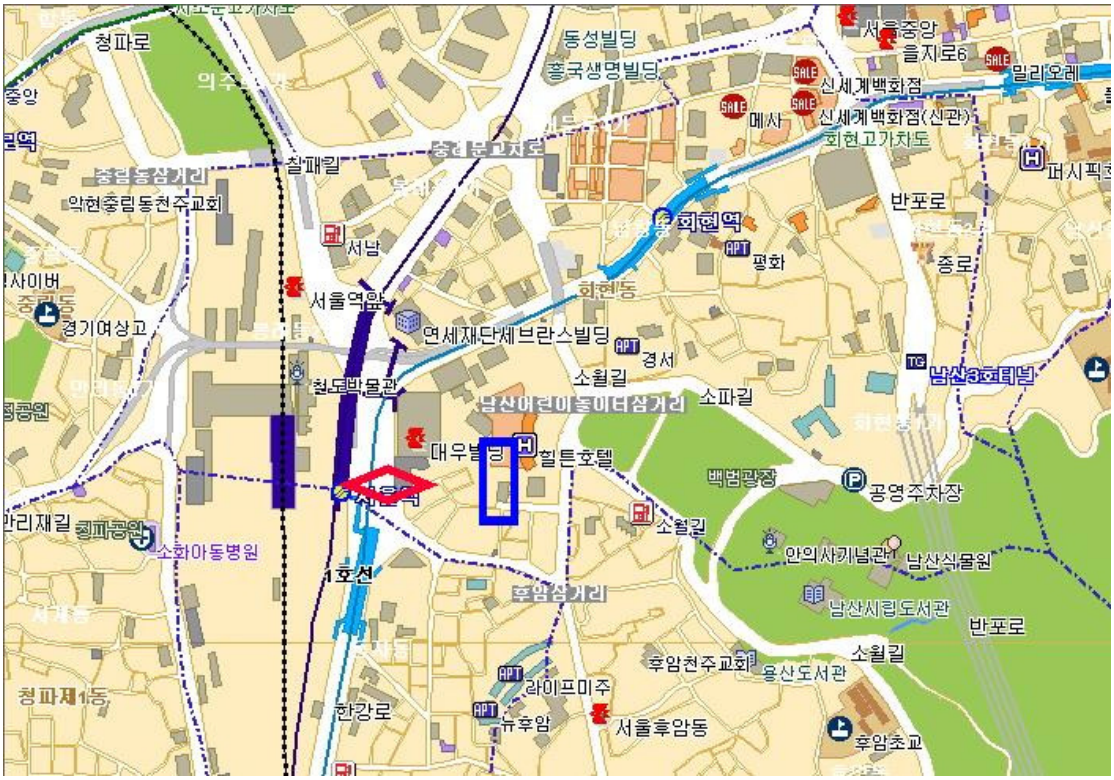
Sub-elements

`shape`	An attributes-only element that specifies the shape to use. Attributes: `type`: "`ellipse`" or "`rectangle`" `width`: the width of the shape, in pixels `height`: the height of the shape, in pixels
`Point`	The location of the icon. A GML element, see

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Fri Aug 19 13:30:25 KST 2005 -->
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>PointShape feature created on Fri Aug 19 13:29:34 KST 2005</title>
  <Feature id="1">
    <label>PointShape feature</label>
    <PointShape>
```

```
      <shape type="ellipse" width="50.0" height="20.0"/>
      <gml:Point>
        <gml:coord><x>126.97328186035156</x><y>37.554481506347656</y></gml:coord>
      </gml:Point>
    </PointShape>
    <style>
      <lineColor rgb="255,0,51" opacity="100%"/>
      <lineWeight>5.0</lineWeight>
    </style>
  </Feature>
</FeatureSet>
```



(Due to a bug in Swing, the red ellipse in the example above appears as a diamond.)

3.6 Polygon

`Polygon` is a GML element used for rendering closed regions. A `Polygon` consists of an outer boundary (the `outerBoundaryIs` element) and, optionally, an inner boundary (the `innerBoundaryIs` element) consisting of one or more rings. Inner rings will form holes in the polygon. This can be used for lakes with islands, for example.

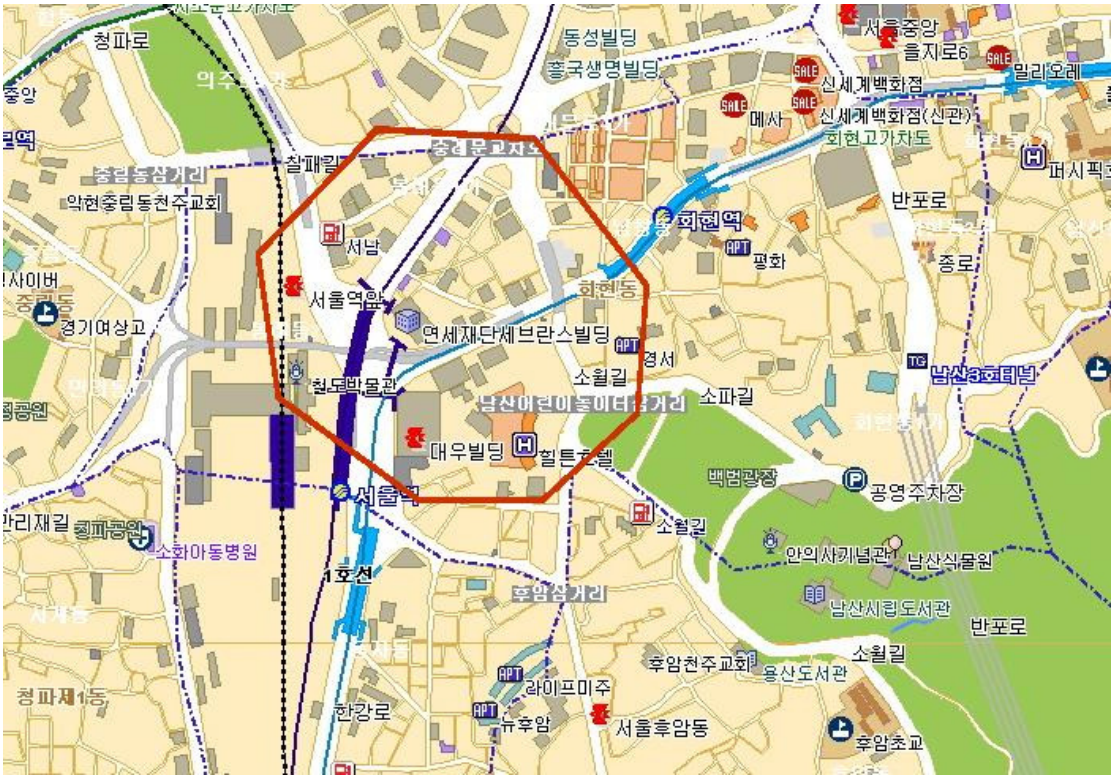
Sub-elements

`outerBoundaryIs`	Contains one `LinearRing` element forming the outer boundary of the polygon.
`innerBoundaryIs`	If present, contains one or more `LinearRing` elements forming the inner boundaries of the polygon.

LinearRing	See Section 3.6.1.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Fri Aug 19 13:45:55 KST 2005 -->
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>Polygon feature created on Fri Aug 19 13:45:41 KST 2005</title>
  <Feature id="1">
    <label>Polygon feature</label>
    <gml:Polygon>
      <outerBoundaryIs>
        <gml:LinearRing>
          <gml:coord><x>126.97310638427734</x><y>37.559898376464844</y></gml:coord>
          <gml:coord><x>126.97085571289062</x><y>37.5579948425293</y></gml:coord>
          <gml:coord><x>126.97124481201172</x><y>37.55580520629883</y></gml:coord>
          <gml:coord><x>126.97386932373047</x><y>37.55424118041992</y></gml:coord>
          <gml:coord><x>126.9761734008789</x><y>37.55424118041992</y></gml:coord>
          <gml:coord><x>126.97795104980469</x><y>37.555564880371094</y></gml:coord>
          <gml:coord><x>126.9781265258789</x><y>37.55751419067383</y></gml:coord>
          <gml:coord><x>126.97602844238281</x><y>37.55975341796875</y></gml:coord>
          <gml:coord><x>126.97307586669922</x><y>37.559898376464844</y></gml:coord>
          <gml:coord><x>126.97266387939453</x><y>37.55953598022461</y></gml:coord>
          <gml:coord><x>126.97266387939453</x><y>37.55953598022461</y></gml:coord>
          <gml:coord><x>126.97310638427734</x><y>37.559898376464844</y></gml:coord>
        </gml:LinearRing>
      </outerBoundaryIs>
    </gml:Polygon>
    <style>
      <lineColor rgb="204,51,0" opacity="100%"/>
      <lineWeight>4.0</lineWeight>
    </style>
  </Feature>
</FeatureSet>
```



3.6.1 The LinearRing element

A LinearRing element contains a sequence of coordinates, either as a sequence of `coord` elements or as a `coordinates` element. See Section 0, “Sub-elements,’ for a description of these elements.

According to the GML specification, the first and last points of a LinearRing element should be coincident. However, we recommend that MFL interpreters allow the closedness of a LinearRing element to be implicit. That is, they should not require the first and last coordinates to be coincident.

3.7 MultiPolygon

`MultiPolygon` is a GML element that is a container for a set of `Polygon` elements. See [GML] for the definition of this element.

3.8 Text

MFL’s `Text` element allows text to be drawn on the map. Text can be drawn in any font, style, and size, and can be drawn at an angle. An “anchor” feature enables text to be placed in different positions relative to the specified location.

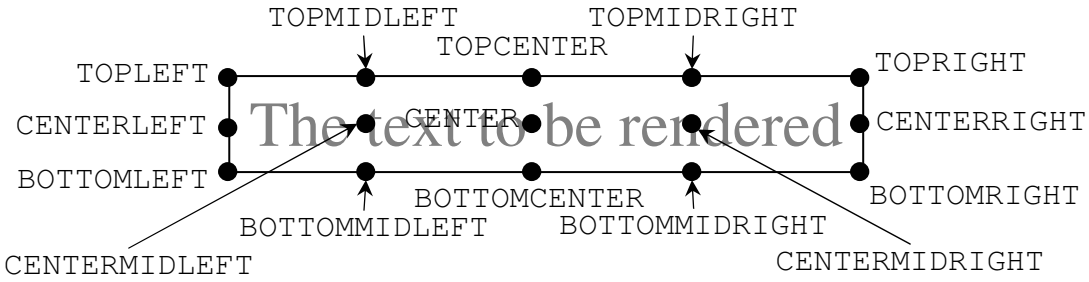
Sub-elements

`text`	The text to be drawn.

font	The font name, style, and size, as attributes. Attributes: name: A font name recognized by the underlying platform. (A future version of MFL will adopt a standard font set.) style: "BOLD", "ITALIC", or "BOLD_ITALIC". Anything else will result in the normal style of text. size: The point size of the font.
baseline	Specifies the location of the text. The baseline is specified as a Point, and angle, and a placement anchor.
Point	The point of the text anchor position.
angle	The angle at which to draw the text. The angle is degrees counter-clockwise from horizontal. **Basic/Tiny**: Text angle is not supported in the MFL Basic or MFL Tiny profiles. Text will be drawn at an angle of 0°.
anchor	One of: BOTTOMLEFT, BOTTOMMIDLEFT, BOTTOMCENTER, BOTTOMMIDRIGHT, BOTTOMRIGHT, TOPLEFT, TOPMIDLEFT, TOPCENTER, TOPMIDRIGHT, TOPRIGHT, CENTERLEFT, CENTERMIDLEFT, CENTER, CENTERMIDRIGHT, CENTERRIGHT. See Section 3.8.1.

3.8.1 The anchor element

The figure below illustrates the meaning of the various anchor element constants.

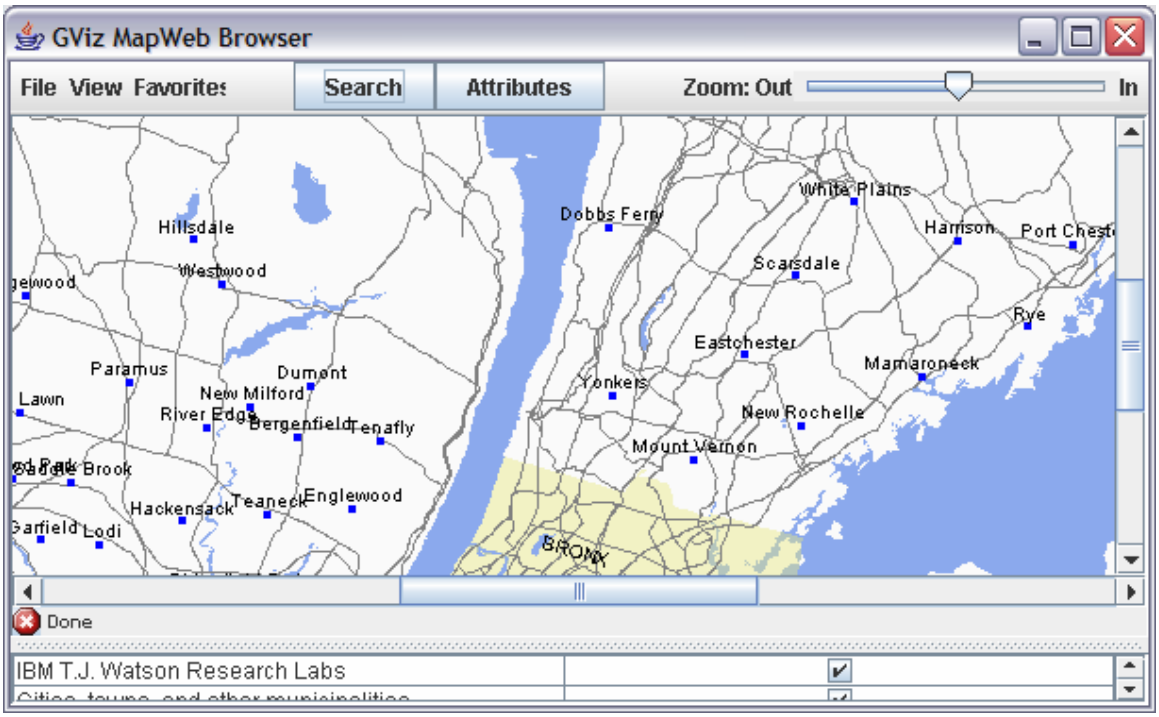


The text should be drawn such that the given Point is at the location—relative to the text—specified by the anchor element. If the anchor location is not specifed, the default is BOTTOMLEFT.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Tue Nov 01 15:38:51 EST 2005 -->
<FeatureSet xmlns:gml="http://www.opengis.net/gml">
  <title>Cities, towns, and other municipalities</title>
  <description>Cities, towns, and other municipalities from the ESRI U.S. PLACES data set</description>
  <Feature id="places_5up:14527">
    <label>Bergenfield</label>
    <PointShape>
      <shape type="rectangle" width="4.0" height="4.0"/>
```

```
        <gml:Point>
          <gml:coord><x>-73.998795</x><y>40.923748</y></gml:coord>
        </gml:Point>
      </PointShape>
      <Text>
        <text>Bergenfield</text>
        <font name="Default" style="PLAIN" size="10"/>
        <baseline>
          <gml:Point>
            <gml:coord><x>-73.998795</x><y>40.923748</y></gml:coord>
          </gml:Point>
          <angle>0</angle>
          <anchor>BOTTOMCENTER</anchor>
        </baseline>
      </Text>
      <style>
         <fillColor rgb="0,0,255" opacity="100%"/>
         <textColor rgb="0,0,255" opacity="100%"/>
      </style>
    </Feature>
</FeatureSet>
```



4 Map Viewer Implementation Suggestions

In this section we offer various suggestions concerning how map viewers may be implemented.

4.1 A Simple Map Projection

Drawing a map implies, either implicitly or explicitly, selection of a map projection. The projection used in an MFL viewer is implementation-dependent. A map viewer may use some algorithm for choosing the projection based on the zoom level of the map and the area of the globe that is being viewed, or it may simply treat the latitude and longitude values as coordinates in a simple Cartesian reference frame.

The map pictures in this document were produced using map viewers that used a simple map projection that is easy to implement and renders suitable maps for small areas, which is typically how interactive maps are used. The projection uses a two-dimensional geometrical transform, which is defined as follows.

We use an object-oriented notation that is based on documentation for the `java.awt.geom.AffineTransform` class, which models a 2-D homogeneous transformation matrix. *I* is the identity matrix, and *scale* and *translate* are methods that apply scaling and translation transforms, respectively.

Let c be the (x, y) coordinate pair to project, and let p be the projected result. s_x and s_y are the scale factors in the x (east/west) and y (north/south) directions, respectively, in units of pixels per degree (to match the units of the coordinates).

$$T = I.scale(s_x, s_y).scale(1, -1).translate(x_{ul}, y_{ul})$$

$$p = T\,c$$

The scale factor s_y is set by the map viewer. The scale factor s_x should be a function of s_y and the latitude of the current view window. A simple formula for s_x is:

$$s_x = s_y \cos(y)$$

where y is the latitude in radians. For convenience, a viewer may simply use y_{ul} as this value, but it must be converted to radians.

4.2 Initial Settings

When a map viewer is initialized, it should display some "home" map view that the user has configured. These "home" settings should include the following:

- The location of the view in geographical coordinates. The upper left corner of the map is convenient for coordinate-transformation purposes (which is x_{ul}, y_{ul} in the section above).
- The view scale. Pixels per degree latitude is convenient for coordinate transformation purposes (this is s_y in the section above).
- The view-pane size (pixel width by pixel height).

- A list of feature-set URLs to load as basemap layers.

Here is a sample XML format that may be used for these purposes.

```
<?xml version="1.0" encoding="UTF-8"?>
<MapViewerConfig>
<!-- view-pane properties set the viewer's initial view -->
  <viewPaneProperties name="Home">
    <upperLeftCoordinates x="126.5" y="37.29999923706055"/>
    <!-- the view scale, in pixels-per-degree latitude  -->
    <pixelsPerDegreeScale value="123900.0"/>
    <!-- the size of the window may be ignored in clients
         where the view-pane size is fixed -->
    <dimension width="1280" height="960"/>
    <backgroundColor value="#8ba9ed"/>
  </viewPaneProperties>
<!-- basemap layers are loaded automatically upon startup -->
  <basemapLayers>
    <FeatureSetDescriptor isBase="true">
      <title>Northern Seoul Basemap</title>
      <!-- the url can point to an MWFS server, or can simply be a
           file or http url for an MFL file containing a basemap -->
      <url>mwfs://kr000802b:9080/TOPAZ-MapServer/MapServerServlet?featureset=basemap</url>
      <source>TOPAZ Demo MapWeb Feature Server</source>
      <!-- this the extent of the geographic region served by
           the feature server given above -->
      <bounds>126.5 37.292152 126.58072 37.2999</bounds>
    </FeatureSetDescriptor>
  </basemapLayers>
</MapViewerConfig>
```

4.3 Miscellaneous Features

The following sections describe certain viewer features that we have found useful in one or more implementations that we have done. Implementations for some contexts, such as viewers for embedded environments such as a telematics client, may not require all features.

4.3.1 Layer controls and pan/zoom controls

It is important for map viewers to give users control of the map view. To prevent overcrowding of the map, viewers should give users the ability to turn the visibility of a feature-set on and off, and to remove feature-sets that are no longer desired. Users should be able pan the map in any direction and zoom in and out in order to focus on the area they desire. An implementation may also offer features that pan and zoom automatically:

- A viewer for a telematics client may pan automatically to follow the path of the vehicle.

- The view may be controlled by an application that wants to show users features that it just pushed to the map, or to show detail of a particular feature (such as a complicated set of turn instructions).

For environments where the user's attention is limited, implementations of these features must focus on ease-of-use.

Viewers for limited-function devices such as cell phones may not be able to support continuous zooming because they may lack the ability to scale images, which may be contained in basemap feature sets. Upon a request from the user to zoom, these viewers can re-request the basemap feature-set from the server, including the desired scale in the request. The server is expected to make a best effort to produce images using this scale. However, the viewer should use the actual scale of the images returned by the feature server to determine its own scale setting.

4.3.2 Showing labels

A map viewer should enable the user to display a feature's label. Cursor mouse-over is a possible implementation, but this may be practical in viewer implementations that rely on touch-screen controls.

4.3.3 Pop-up descriptions

A map viewer should enable users to view feature descriptions. A mechanism for this is to use pop-up windows. In implementations where screen space may not permit this, a viewer can alternatively simply switch to a window where the description has been loaded.

4.3.4 Favorites

Users may wish to bookmark feature-sets that they find particularly useful. Viewers can enable users to store URLs for these feature-sets, in the same manner that Web browsers enable users to store Web page URLs as bookmarks.

4.3.5 Pre-set views and view history

Users may wish to have pre-set view settings that they would like to load in one action. For example, they may wish to frequently view a map that contains their entire commute path. A map viewer may wish to record view settings in named views which, when selected, change the map view settings to the recorded ones.

These named view settings must be persistent. The example file format for initial settings, above, has a distinguished named view, called "Home". Other named views may be added to this file.

Viewers may also want to revert to view settings used recently. A map viewer can record a view-settings history and allow users to go back through the history to return to an earlier view.

4.3.6 Drawing controls

Map viewers for desktop environments may want to offer drawing tools that enable users to create their own map features. Fuller discussion of this is beyond the scope of this document.

5 MapWeb Feature Servers

An MFL feature-set can be as large, in geographic extent, as desired. Feature-sets that represent basemap layers may cover very large areas, up to the entire globe. If these are detailed layers designed for high zoom levels, these feature-sets can be extremely large, in terms of the amount of data, and thus are not practical to send to map viewers in their entirety. Therefore, associated with the MFL specification, there is a simple protocol defined for communicating with Web servers serving these large feature-sets, which enables them to send only that part of the feature-set that will be rendered by the viewer. The protocol consists of predefined query parameters in an ordinary HTTP GET request. We refer to this as the MWFS protocol. Associated with this protocol is a cooperative caching scheme implemented by both the viewer and the server (but is optional for both).

5.1 MWFS Protocol

A request to an MWFS server consists of an HTTP GET request with the following query parameters:

`featureset` Names the feature-set requested. A feature server may serve many different feature-sets, which could be different basemaps for different purposes (e.g., political boundaries or geographical features).

`bounds` The area requested, as a bounding box (see format below). This should typically be the actual bounding box of the current view. The server must return features covering at least this area. A feature server may return features for a larger area, in order to reduce the number of requests on the server; a viewer may also increase the area beyond the current view's bounding box for the same purpose, and as a prefetch mechanism.

The format of the `bounds` parameter is "`lng-min lat-min lng-max lat-max`", where `lng` and `lat` are longitude and latitude, respectively. Note that the spaces separating the coordinate values must be encoded as "+" signs, per the HTTP specification.

`scale` A floating-point parameter that is the current scale used by the viewer, in pixels-per-degree-latitude. This enables the feature server to return a feature-set appropriate for that scale. The feature server should return a feature-set with a `scaleRange` element that indicates the scale range that the feature-set is appropriate for.

`useCache` A boolean parameter that permits the feature server to remember which features it has served a client in a particular client session. It can then elect to not return features that it has already sent a client in the current session. This is designed to reduce the average size of the feature-sets returned to the client. A client will set this to true only if it merges the features received in successive requests for a feature-set into a single feature-set object.

Below is an example of a request to an MWFS server:

```
http://safari.watson.ibm.com:9080/mwfs/MapServerServlet
   ?featureset=basemap
   &bounds=126.9658+37.55883+126.99674+37.5769
   &scale=41451.57
   &useCache=true
```

This particular request

An MWFS server returns an MFL document. If a feature server returns the same features for a given scale range, then each such feature should have a distinct identifier (in the `FeatureSet` element's `id` attribute), and should be contained in a feature-set with a unique identifier (in the `Feature` element's `id` attribute). This enables the client to cache features, as described in the section following. Below is the MFL returned by the request above. The scale range is given as 0.0 to 3.4028235E38, meaning effectively all scale values are in range. The same effect is achieved by not including a `scaleRange` element.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created:Thu Nov 03 13:33:49 EST 2005 -->
<FeatureSet id="basemap-1" xmlns:gml="http://www.opengis.net/gml">
  <scaleRange gt="0.0" le="3.4028235E38"/>
  <title>Basemap from TOPAZ demo map server</title>
  <description>Images created from (source of images)</description>
  <Feature id="11.JPG">
    <label>11.JPG</label>
    <Image>
      <href>tileImages/11.JPG</href>
      <gml:Box>
        <gml:coord><x>126.94383</x><y>37.572506</y></gml:coord>
        <gml:coord><x>126.9673</x><y>37.58948</y></gml:coord>
      </gml:Box>
    </Image>
  </Feature>
  <Feature id="12.JPG">
    <label>12.JPG</label>
    <Image>
      <href>tileImages/12.JPG</href>
      <gml:Box>
        <gml:coord><x>126.9673</x><y>37.572506</y></gml:coord>
        <gml:coord><x>126.99076</x><y>37.58948</y></gml:coord>
      </gml:Box>
    </Image>
```

```
  </Feature>
  <Feature id="21.JPG">
    <label>21.JPG</label>
    <Image>
      <href>tileImages/21.JPG</href>
      <gml:Box>
        <gml:coord><x>126.94383</x><y>37.55553</y></gml:coord>
        <gml:coord><x>126.9673</x><y>37.572506</y></gml:coord>
      </gml:Box>
    </Image>
  </Feature>
  <Feature id="22.JPG">
    <label>22.JPG</label>
    <Image>
      <href>tileImages/22.JPG</href>
      <gml:Box>
        <gml:coord><x>126.9673</x><y>37.55553</y></gml:coord>
        <gml:coord><x>126.99076</x><y>37.572506</y></gml:coord>
      </gml:Box>
    </Image>
  </Feature>
</FeatureSet>
```

5.2 Feature Caching

Map viewers can significantly include viewer response time by caching feature-sets from MapWeb Feature Servers. Feature-sets are cached in two ways. The first is similar to the way Web pages are cached, by URL, but the fact that different zoom levels result in different feature-sets returned means that the cache key should be the URL plus the feature-set id. This technique relies on the feature servers to return the same id for feature-sets within a given scale range.

Map viewers also cache features by merging feature-sets for the same URL and feature-set id. The ability to do this relies on the feature server using unique ids for features belonging to the same feature-set and scale range.

6 References

[GML] OpenGIS® Geography Markup Language (GML) Encoding Specification (Version 3.1.1, Document #03-105r1). Open Geospatial Consortium, www.opengeospatial.org. April, 2004.

[SVGM] Mobile SVG Profiles: SVG Tiny and SVG Basic. http://www.w3.org/TR/SVGMobile/