

Scalable Vector Graphics (SVG)

This document describes the procedures for implementing W3C compliant Scalable Vector Graphics (SVG) through the use of MapServer v4.5 (and later).

Author: Jeff McKenna

Contact: jeffmckenna(at)gmail.com

Last Updated: 2005/12/13

Table of Contents

- 1 Introduction
 - ◆ 1.1 Links to SVG-Related Information
- 2 Feature Types and SVG Support Status
 - ◆ 2.1 Annotation Layers
 - ◆ 2.2 Circle Layers
 - ◆ 2.3 Line Layers
 - ◆ 2.4 Point Layers
 - ◆ 2.5 Polygon Layers
 - ◆ 2.6 Raster Layers
 - ◆ 2.7 Text Features
 - ◆ 2.8 WMS Layers
- 3 Testing your SVG Output
- 4 goSVG
 - ◆ 4.1 Definition
 - ◆ 4.2 Support for Specific goSVG Elements
 - ◆ 4.3 Setting up a Mapfile for goSVG Output
 - ◆ 4.4 Testing your goSVG Output
 - ◆ 4.5 Sample goSVG File Produced by MapServer
- 5 About This Document
 - ◆ 5.1 Copyright Information
 - ◆ 5.2 Disclaimer

1 Introduction

SVG (or Scalable Vector Graphics) is a standardized XML language for describing 2D graphics via vector graphics, text and raster graphics. As of version 4.5, MapServer can output SVG v1.1 maps. The following documentation is based on the World Wide Web Consortium's (W3C) Scalable Vector Graphics (SVG) 1.1 Specification.

This document assumes that you are already familiar with certain aspects of MapServer:

- MapServer application development and setting up map files.

1.1 Links to SVG-Related Information

- SVG 1.1 specification
- SVG Discussion Paper
- G-XML Project Page
- SVG Tiny Profile

- [MapFile Reference Doc](#)

2 Feature Types and SVG Support Status

2.1 Annotation Layers

Annotation layers are supported (see the *Text Features* section below for details).

2.2 Circle Layers

Circle layers are not yet supported.

2.3 Line Layers

The following items describe how line layers are handled by MapServer for SVG output:

- Lines are converted to SVG polyline elements.
- The STYLE object's WIDTH parameter is used for SYMBOL 0 for line thickness.
- The STYLE object's SIZE parameter is used for other symbols for line thickness.
- All lines are drawn without symbols - only line thickness changes.
- If a style uses a symbol and this symbol has a dashed style, it will be transformed into an SVG stroke-dasharray element.

2.4 Point Layers

The following items describe how point layers are handled by MapServer for SVG output:

- VECTOR, ELLIPSE, and TRUETYPE symbols are supported.
- PIXMAP symbols are not currently supported.
- Labels attached with the symbols are supported (see the *Text Features* section below for details).

2.5 Polygon Layers

The following items describe how polygon layers are handled by MapServer for SVG output:

- Polygons are converted to SVG polygon elements.
- The STYLE's COLOR is used for the fill.
- The STYLE's OUTLINECOLOR is used for the stroke.
- SVG patterns are not currently supported.

2.6 Raster Layers

The following items describe how raster layers are handled by MapServer for SVG output:

- Temporary image is created through the GD library, and GD functions are used to draw the layer.
- You must have at least PNG or JPEG support compiled in MapServer.
- You must have the WEB object's IMAGEPATH and IMAGEURL set properly in your mapfile.

2.7 Text Features

The following items describe how text features are handled by MapServer for SVG output:

- Text is converted to SVG `text` element.
- Only TRUETYPE fonts are supported.
- Supports labels with ENCODING (output as UTF-8 hexadecimal values).
- The FONT name used in MapServer is parsed to form the SVG `font-family`, `font-style`, and `font-weight`.

2.8 WMS Layers

WMS layers are not yet supported.

Setting up a Mapfile for SVG Output

- You must have valid IMAGEPATH and IMAGEURL parameters set in the WEB object of the mapfile.
- To be able to output a valid SVG file, the user needs to define an OUTPUTFORMAT object in the map file and set the IMAGETYPE parameter to `svg`. Here is an example:

```
MAP
...
IMAGETYPE svg
...
OUTPUTFORMAT
  NAME svg
  MIMETYPE "image/svg+xml"
  DRIVER svg
  FORMATOPTION "COMPRESSED_OUTPUT=TRUE"
  FORMATOPTION "FULL_RESOLUTION=TRUE"
END
...
WEB
  IMAGEPATH "/tmp/ms_tmp/"
  IMAGEURL  "/ms_tmp/"
END
...
LAYER
  ...
END
END
```

Note:

If FORMATOPTION "COMPRESSED_OUTPUT=TRUE" is set MapServer will produce a compressed SVG file (svgz). By default this option is FALSE. Note that to be able to create compressed output, MapServer must be built with the compile flag USE_ZLIB.

If FORMATOPTION "FULL_RESOLUTION=TRUE" is set MapServer will not eliminate duplicate points and colinear lines when outputting SVG. By default this option is set to FALSE.

3 Testing your SVG Output

- The easiest way to test your SVG mapfile is to use [MapServer CGI](#). For example, you might enter the following URL in a browser:

```
http://127.0.0.1/cgi-bin/mapserv.exe?map=my/path/to/my-svg.map&mode=map&layers=layer1 layer2
```

- You can also use [PHP/Mapscript](#) to test your SVG mapfile. Your php file might look like the following:

```

<?php

    dl("php_mapscript_45.dll");

    $oMap = ms_newmapObj("my/path/to/my-svg.map");

    $img = $oMap->draw();

    header("Content-type: image/svg+xml");

    $url = $img->saveImage("");

?>

```

An SVG file should be created in your IMAGEPATH directory. If you open the SVG file in a text editor you can see that it is an XML file. Below is a sample SVG file of a point layer with labels:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-flat
<svg version="1.1" width="400" height="300" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http:

<!-- START LAYER popplace -->
<ellipse cx="252" cy="130" rx="3" ry="3" fill="#000000" />
<ellipse cx="37" cy="227" rx="3" ry="3" fill="#000000" />
<ellipse cx="127" cy="239" rx="3" ry="3" fill="#000000" />
<ellipse cx="255" cy="282" rx="3" ry="3" fill="#000000" />
<polygon fill="#000000" stroke-width="1" points=" 267,263 270,263 271,260 272,263 275,263 273,
<ellipse cx="288" cy="247" rx="3" ry="3" fill="#000000" />
<ellipse cx="313" cy="243" rx="3" ry="3" fill="#000000" />
<ellipse cx="328" cy="233" rx="3" ry="3" fill="#000000" />
<ellipse cx="331" cy="245" rx="3" ry="3" fill="#000000" />
<ellipse cx="366" cy="196" rx="3" ry="3" fill="#000000" />
<ellipse cx="161" cy="246" rx="3" ry="3" fill="#000000" />
<ellipse cx="92" cy="208" rx="3" ry="3" fill="#000000" />
<ellipse cx="40" cy="125" rx="3" ry="3" fill="#000000" />
<ellipse cx="108" cy="146" rx="3" ry="3" fill="#000000" />
<text x="40" y="143" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="43" y="121" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="34" y="205" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="164" y="258" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="316" y="190" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="334" y="258" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="249" y="230" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="241" y="242" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="223" y="260" font-family="fritqat-italic" font-size="8pt" fill="#ff0000" stroke="#fff
<text x="210" y="279" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="82" y="234" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="40" y="223" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="214" y="125" font-family="fritqat" font-size="8pt" fill="#000000" stroke="#ffffff" st
</svg>

```

You can now view the SVG file in a supported browser (see the official [list of SVG implementations](#) for possible SVG viewers). The [Adobe Viewer plugin](#) is very popular.

4 goSVG

goSVG is now supported as a vector output format in MapServer 4.5 (and later).

4.1 Definition

This definition of goSVG was obtained from [here](#).

goSVG is short for "G-XML over SVG" and "g-contents over SVG". This is a subset for mobiles specified within the [G-XML](#) (a Japanese Spatial Information Format which is an XML based protocol with the ability to describe, communicate and exchange Spatial Information and Electric Maps), and is a Spatial Information Exchanging format that determines the method to expand spatial information and connect to the backend system(G-XML standard mark format). goSVG is an expanded [SVG Tiny profile](#) (a Mobile profile of [SVG 1.1](#), suited for cellular phones) that adds functions that are useful for Spatial Information Services (SVG Map Service).

4.2 Support for Specific goSVG Elements

- Name space extension: supported
- Content Area Definition (bounding box): supported
- Geographic Coordinate System: supported
- Map Request Protocol: supported

4.3 Setting up a Mapfile for goSVG Output

4.3.1 Requirements

- A valid MapServer [mapfile](#).
- Valid IMAGEPATH and IMAGEURL parameters set in the WEB object of the mapfile.
- A PROJECTION object defined beneath the MAP object, using an EPSG code. For example:

```
MAP
...
WEB
  IMAGEPATH "/tmp/ms_tmp/"
  IMAGEURL  "/ms_tmp/"
END
...
PROJECTION
  "init=epsg:42304"
END
...
LAYER
...
END
END
```

4.3.2 Setting the OUTPUTFORMAT

To be able to output a valid goSVG file, you must define an OUTPUTFORMAT object in the mapfile and set the IMAGETYPE to svg. Here is an example:

```
MAP
...
IMAGETYPE svg
...
OUTPUTFORMAT
  NAME svg
  MIMETYPE "image/svg+xml"
  DRIVER svg
  FORMATOPTION "GOSVG=TRUE"
```

```

    FORMATOPTION "GOSVG_ZoomInTH=20"
    FORMATOPTION "GOSVG_ZoomOutTH=40"
    FORMATOPTION "GOSVG_ScrollTH=60"
END
...
WEB
    IMAGEPATH "/tmp/ms_tmp/"
    IMAGEURL  "/ms_tmp/"
END
...
PROJECTION
    "init=epsg:42304"
END
...
LAYER
...
END
END

```

4.3.2.1 Specific FORMATOPTIONS Related to goSVG

GOSVG

should be set to TRUE. The default is false.

GOSVG_ZoomInTH

controls the zoomin threshold when outputting the Map Request Protocol. If it is not defined the default value is set to 70.

GOSVG_ZoomOutTH

controls the zoomout threshold when outputting the Map Request Protocol. If it is not defined the default value is set to 100.

GOSVG_ScrollTH

controls the scrolling threshold when outputting the Map Request Protocol. If it is not defined the default value is set to 10.

4.4 Testing your goSVG Output

Refer to the section [Testing your SVG Output](#) to generate and test your goSVG output. goSVG can be read by regular SVG viewers (they will just ignore the goSVG headers).

4.5 Sample goSVG File Produced by MapServer

Below is a sample goSVG file of a point layer with labels:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-flat
<svg version="1.1" width="400" height="300" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http:
<title>DEMO</title>
<metadata>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:crs = "http://www.opengis.org/2000/coordsys" xmlns:svg="http://www.w3.org/2000/svg">
<rdf:Description>
<crs:CoordinateReferenceSystem svg:transform="matrix(0.000066,0.000000,0.000000,-0.000066,171.243
rdf:resource="http://www.opengis.net/gml/srs/epsg.xml#42304"/>
</rdf:Description>
</rdf:RDF>
<au:lbs protocol="maprequest">
<au:zoomin th="20" xlink:href="."/>
<au:zoomout th="40" xlink:href="."/>
<au:scroll th="60" xlink:href="."/>
</au:lbs>

```

```

</metadata>

<!-- START LAYER popplace -->
<ellipse cx="252" cy="130" rx="3" ry="3" fill="#000000" />
<ellipse cx="37" cy="227" rx="3" ry="3" fill="#000000" />
<ellipse cx="127" cy="239" rx="3" ry="3" fill="#000000" />
<ellipse cx="255" cy="282" rx="3" ry="3" fill="#000000" />
<polygon fill="#000000" stroke-width="1" points=" 267,263 270,263 271,260 272,263 275,263 273,
<ellipse cx="288" cy="247" rx="3" ry="3" fill="#000000" />
<ellipse cx="313" cy="243" rx="3" ry="3" fill="#000000" />
<ellipse cx="328" cy="233" rx="3" ry="3" fill="#000000" />
<ellipse cx="331" cy="245" rx="3" ry="3" fill="#000000" />
<ellipse cx="366" cy="196" rx="3" ry="3" fill="#000000" />
<ellipse cx="161" cy="246" rx="3" ry="3" fill="#000000" />
<ellipse cx="92" cy="208" rx="3" ry="3" fill="#000000" />
<ellipse cx="40" cy="125" rx="3" ry="3" fill="#000000" />
<ellipse cx="108" cy="146" rx="3" ry="3" fill="#000000" />
<text x="40" y="143" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="43" y="121" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="34" y="205" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="164" y="258" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="316" y="190" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="334" y="258" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="249" y="230" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="241" y="242" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="223" y="260" font-family="fritgat-italic" font-size="8pt" fill="#ff0000" stroke="#fff
<text x="210" y="279" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
<text x="82" y="234" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="40" y="223" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" str
<text x="214" y="125" font-family="fritgat" font-size="8pt" fill="#000000" stroke="#ffffff" st
</svg>

```

5 About This Document

5.1 Copyright Information

Copyright (c) 2005, Jeff McKenna.

This documentation is covered by the same Open Source license as the MapServer software itself. See MapServer's [License and Credits](#) page for the complete text.

5.2 Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies that may be damaging to your system. Although this is highly unlikely, the author(s) do not take any responsibility for that: proceed with caution.