

MapServer User's Manual

Author: Jeff McKenna

Last Updated: April 8th, 2008

Version: 0.1



Table of Contents

1 Introduction.....	4
2 Getting Started.....	4
2.1 Installing.....	4
2.1.1 Windows.....	4
2.1.2 Unix.....	9
2.1.3 Mac OS X.....	9
2.2 Architecture of MapServer.....	10
2.2.1 Supporting Libraries.....	11
3 Preparing Your Data.....	11
3.1 Supported Formats.....	11
3.1.1 Raster Formats.....	12
3.1.2 Vector Formats.....	15
3.2 The .Map File.....	16
3.2.1 Important Notes.....	17
3.2.2 Example Objects.....	18
3.2.3 CLASS Expressions.....	24
3.2.4 Mapfile Includes.....	24
3.2.5 Mapfile Creation Tools.....	26
3.3 MapServer Utilities.....	27
3.3.1 legend.....	28
3.3.2 msencrypt.....	28
3.3.3 scalebar.....	30
3.3.4 shp2img.....	30
3.3.5 shptree.....	32
3.3.6 shptreevis.....	33
3.3.7 sortshp.....	34
3.3.8 tile4ms.....	36
3.4 GDAL/OGR Utilities.....	40
3.4.1 ogrinfo.....	40
3.4.2 ogr2ogr.....	42
3.4.3 ogrtindex.....	43
3.4.4 gdalinfo.....	44
3.4.5 gdaladdo.....	45
3.4.6 gdaltindex.....	46
3.4.7 gdal_translate.....	47
3.4.8 gdalwarp.....	48
3.5 Projections.....	50
3.5.1 PROJ.4.....	50
3.5.2 EPSG Codes.....	50
3.5.3 MapServer and Projections.....	51
3.6 Tips and Tricks.....	53
3.6.1 Rules for Preparing Data for MapServer.....	53
3.6.2 Labelling.....	54

3.6.3	Displaying Highway Shields.....	57
3.6.4	Debugging in MapServer.....	59
3.6.5	Smooth Line Output (Antialiasing).....	60
3.6.6	Color Ramps.....	61
3.6.7	Charting.....	62
3.7	Factors Affecting Performance.....	64
4	Building a MapServer Application.....	66
4.1	Flavors.....	66
4.2	Client Software.....	66
5	Community.....	66
5.1	History of MapServer.....	66
5.1.1	The Past.....	66
5.1.2	The Present.....	66
5.1.3	The Future.....	67
5.2	Project Steering Committee.....	67
5.3	Contributing.....	68
5.4	Getting Help.....	69
5.4.1	Internet Relay Chat (IRC).....	69
5.4.2	Mailing Lists.....	70
5.4.3	Conferences/Workshops.....	71
5.4.4	OSGeo Foundation.....	71
6	Important Links.....	74
7	Glossary.....	75
8	Credits.....	76
9	Appendix.....	77
9.1	Mapfile Reference.....	77
9.1.1	CLASS Object.....	77
9.1.2	FEATURE Object.....	78
9.1.3	GRID Object.....	79
9.1.4	INCLUDE Object.....	80
9.1.5	JOIN Object.....	81
9.1.6	LABEL Object.....	86
9.1.7	LAYER Object.....	87
9.1.8	LEGEND Object.....	92
9.1.9	MAP Object.....	93
9.1.10	OUTPUTFORMAT Object.....	95
9.1.11	PROJECTION Object.....	97
9.1.12	QUERYMAP Object.....	98
9.1.13	REFERENCE Object.....	99
9.1.14	SCALEBAR Object.....	100
9.1.15	STYLE Object.....	101
9.1.16	WEB Object.....	102
9.2	Googlish Mapfile Example.....	103

1 Introduction

MapServer is a popular Open Source project whose purpose is to display dynamic spatial maps over the Internet. Some of its major features include:

- support for display and querying of hundreds of raster, vector, and database formats
- ability to run on various operating systems (Windows, Linux, Mac OS X, etc.)
- support for popular scripting languages (PHP, Python, Perl, Java, C, Ruby)
- on-the-fly projections
- high quality rendering
- fully customizable application output
- many ready-to-use Open Source application environments

2 Getting Started

MapServer has made some great improvements to users in the recent years for new users. The knock against MapServer used to be that the learning curve was so high, but these days through the availability of installers, documentation, tutorials, and conference workshops one can get started with little grief.

2.1 Installing

Since MapServer is Open Source, users have an option of compiling their own build, or using a pre-built version of the software (“binary”). For most users, using a pre-built binary of MapServer will be more than sufficient.

2.1.1 Windows

Pre-built Versions

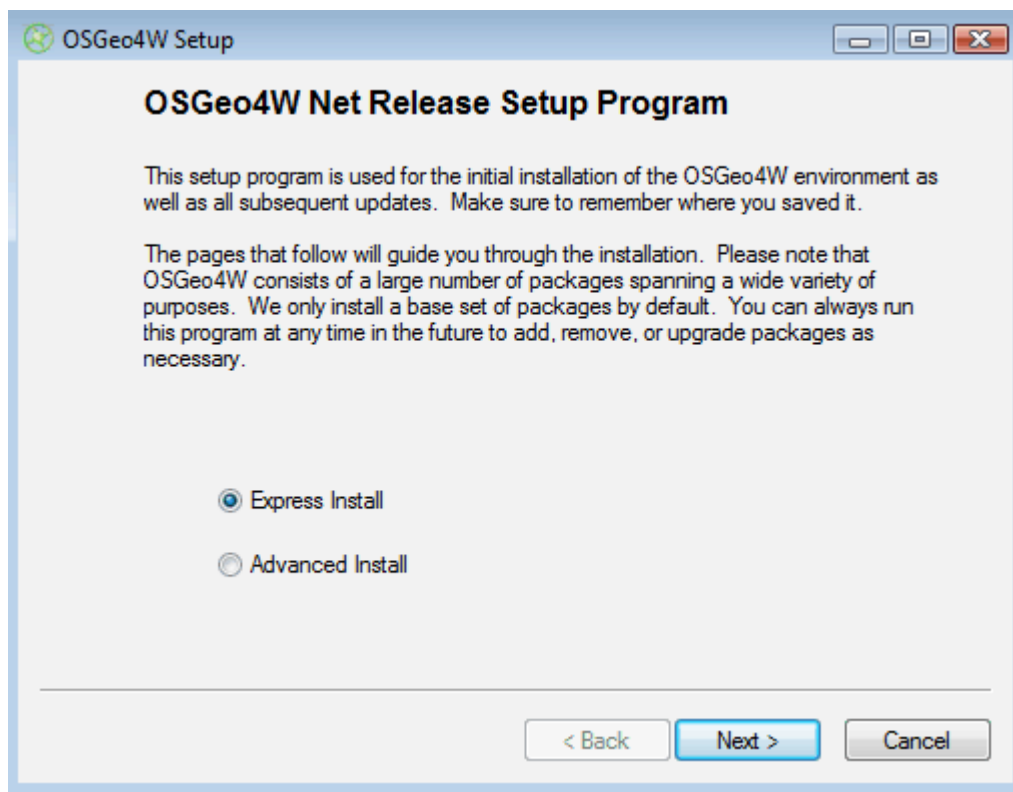
OSGeo4W

Developed in 2008, this new Windows installer can be used to download and/or update MapServer, add-on applications, and also other Open Source geospatial software. The following steps illustrate how to use OSGeo4W:

Step1: Download OSGeo4W

<http://download.osgeo.org/osgeo4w/osgeo4w-setup.exe>

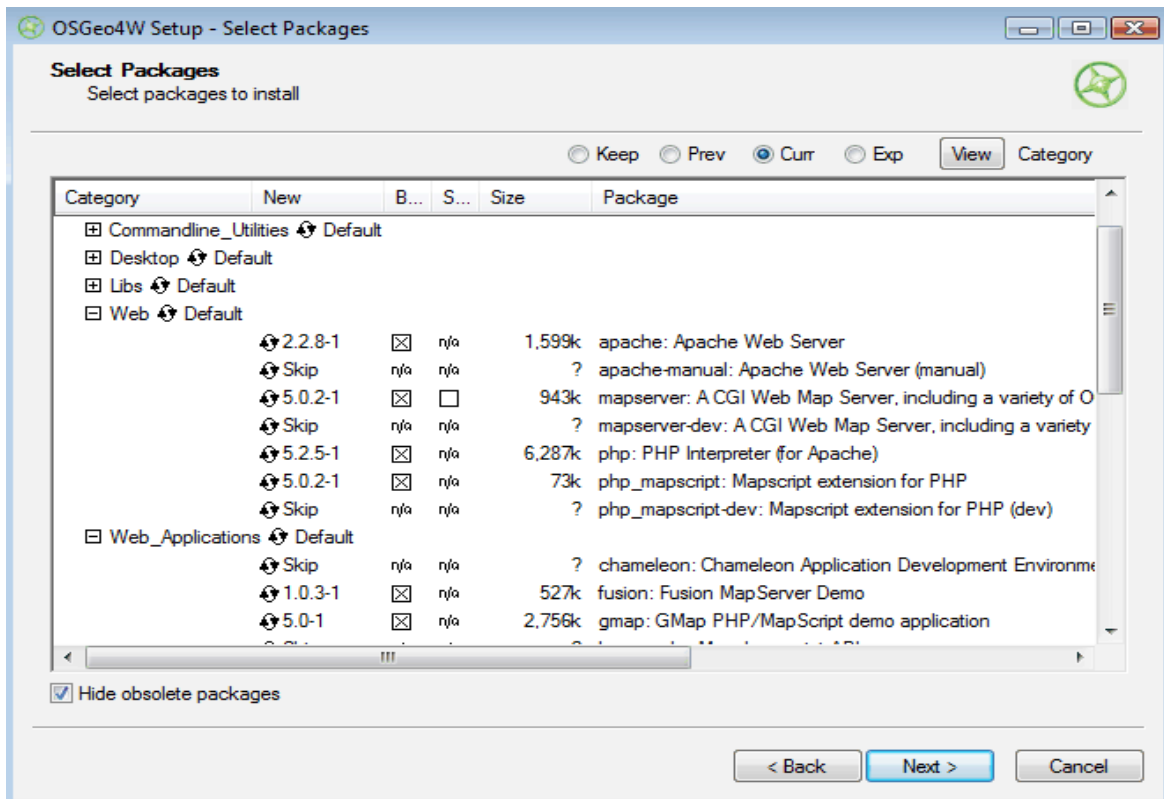
Step2: Double-click the exe



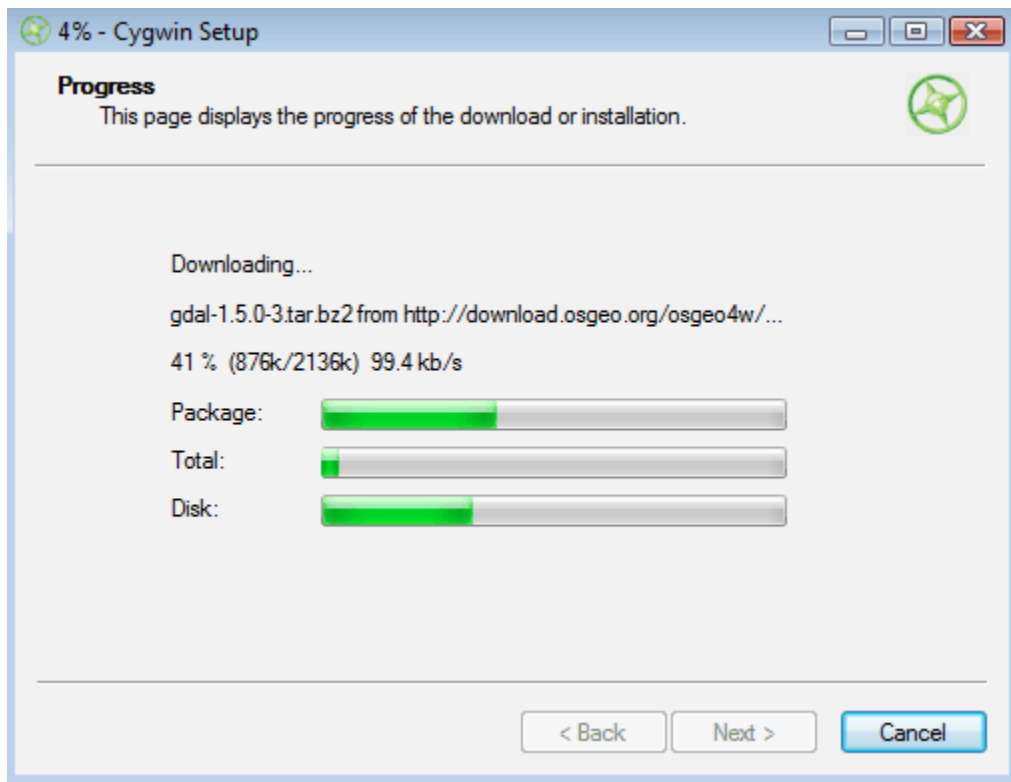
Step3: Choose “Express” or “Advanced” Install Type

Express contains options for higher-level packages such as MapServer, GRASS, and uDig. Advanced gives you full access to choosing commandline tools and applications for MapServer, that are not included in the Express install

Step4: Select Packages to Install



Click on the “Default” text beside the higher-level packages (such as Web) to install all of Web's sub-packages, or click on the “Skip” text beside the sub-package (such as MapServer) to install that package and all of its dependencies. When you have selected the packages you want click the Next button to begin downloading and installing.



Step5: Test Your Installation

If you have installed and started Apache properly, you should be able to see a list of installed MapServer packages on you localhost page (as shown in the following diagram).



MS4W

The father of OSGeo4W, this package is simply a zip archive containing MapServer, Apache, and all their dependencies. Add-on packages can later be downloaded and unzipped over existing MS4W folders.

Location: <http://www.maptools.org/ms4w/index.phtml?page=downloads.html>

Compiling from Source

Compiling MapServer from source on Windows is only recommended for those with experience using compiler environments. Instructions are however well documented in the 'Win32 Compilation and Installation' how-to (http://mapserver.gis.umn.edu/docs/howto/win32_compiling).

2.1.2 Unix

Binaries

FGS

This installer will extract MapServer and of its dependencies on a Linux system.

More information: <http://www.maptools.org/fgs/>

Compiling from Source

Compiling MapServer and its dependencies on Unix is straight forward for an experienced Unix user. Instructions are well documented in the 'Unix Compilation and Installation' how-to (http://mapserver.gis.umn.edu/docs/howto/compiling_on_unix).

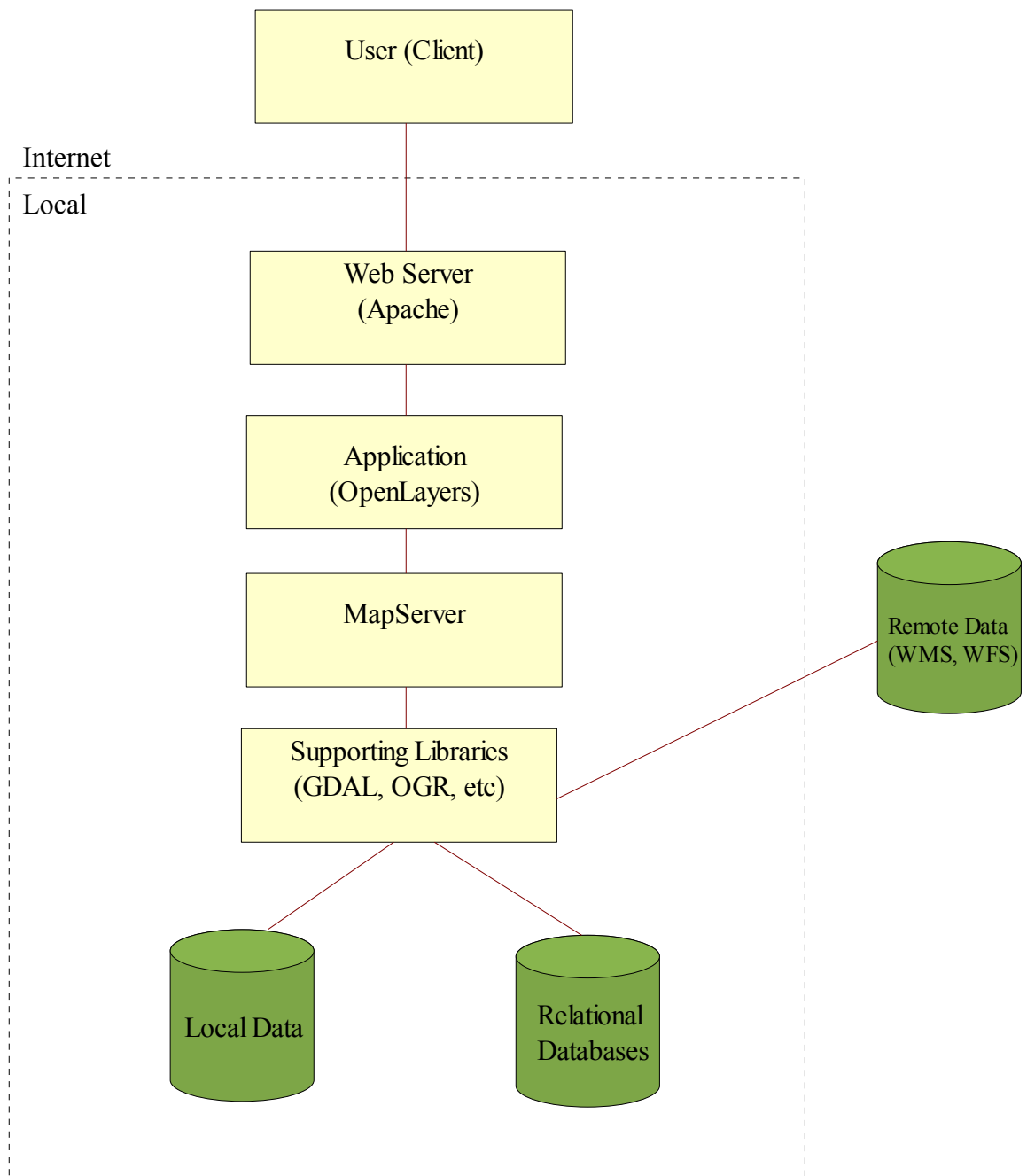
2.1.3 Mac OS X

MapServer packages for Panther, Tiger, and Leopard are available at:

<http://www.kyngchaos.com/wiki/software:mapserver>

2.2 Architecture of MapServer

The following diagram illustrates the relationship between the user (client), MapServer, and the spatial data being served.



2.2.1 Supporting Libraries

As illustrated in the previous diagram, MapServer depends on many other Open Source libraries (projects) for specific functionality. The following list describes several of the important supporting libraries, as you will be hearing of them throughout your MapServer experience:

AGG:	High quality graphics rendering engine
Freetype:	TrueType font library used in labelling
GD:	Graphics rendering engine
GDAL:	Raster data access library
GEOS:	Spatial analysis library
OGR:	Vector data access library
PROJ.4:	Projections library
SWIG:	Wrapper library for accessing scripting languages

3 Preparing Your Data

One of the most important aspects of a MapServer application is data access. You can easily spend as much time configuring data access issues as building your application interface. The goal of this section is to make your data access experience easy.

3.1 Supported Formats

Data access for MapServer occurs through two libraries: GDAL for raster, and OGR for vector, which provide access to many many formats without the need for converting. It should be noted however that the native formats for MapServer are ESRI Shapefiles and GeoTIFFs (we will come back to this point later when we discuss speed of display).

3.1.1 Raster Formats

MapServer uses a library named “GDAL” for raster access. GDAL, standing for “Geospatial Data Abstraction Library”, is an Open Source library that can read and write multiple raster formats. This library is maintained by Frank Warmerdam and a team of developers in the community, and is now used in many other projects such as Google Earth, ArcGIS 9, FME, GRASS, and Quantum GIS.

The following is a list of the supported raster formats, but for an uptodate list you should always check the GDAL website: http://www.gdal.org/formats_list.html

Format	Shortname
Arc/Info ASCII Grid	AAIGrid
ADRG/ARC Digitized Raster Graphics (.gen/.thf)	ADRG
Arc/Info Binary Grid (.adf)	AIG
AIRSAR Polarimetric	AIRSAR
Magellan BLX Topo (.blx, .xlb)	BLX
Microsoft Windows Device Independent Bitmap	BMP
BSB Nautical Chart Format (.kap)	BSB
VTP Binary Terrain Format (.bt)	BT
CEOS (Spot for instance)	CEOS
DRDC COASP SAR Processor Raster	COASP
TerraSAR-X Complex SAR Data Product	COSAR
Spot DIMAP (metadata.dim)	DIMAP
First Generation USGS DOQ (.doq)	DOQ1
DODS / OPeNDAP	DODS
New Labelled USGS DOQ (.doq)	DOQ2
Military Elevation Data (.dt0, .dt1, .dt2)	DTED
ERMapper Compressed Wavelets (.ecw)	ECW
ESRI .hdr Labelled	Ehdr
NASA ELAS	ELAS
ENVI .hdr Labelled Raster	ENVI
Envisat Image Product (.n1)	Envisat

Format	Shortname
EOSAT FAST Format	FAST
FITS (.fits)	FITS
GSat File Format	GFF
Graphics Interchange Format (.gif)	GIF
WMO GRIB1/GRIB2 (.grb)	GRIB
GMT Compatible netCDF	GMT
GRASS Rasters	GRASS
Golden Software ASCII Grid	GSAG
Golden Software Binary Grid	GSBG
Golden Software Surfer 7 Binary Grid	GS7BG
TIFF / GeoTIFF (.tif)	GTIFF
GXF - Grid eXchange File	GXF
Hierarchical Data Format Release 4 (HDF4)	HDF4
Hierarchical Data Format Release 5 (HDF5)	HDF5
Intergraph Raster	INGR
Erdas Imagine (.img)	HFA
Vexcel MFF2	HKV
Idrisi Raster	RST
Image Display and Analysis (WinDisp)	IDA
ILWIS Raster Map (.mpr,.mpl)	ILWIS
JAXA PALSAR Product Reader (Level 1.1/1.5)	JAXAPALSAR
Japanese DEM (.mem)	JDEM
JPEG JFIF (.jpg)	JPEG
JPEG2000 (.jp2, .j2k)	JPEG2000
JPEG2000 (.jp2, .j2k)	JP2KAK
JPEG2000 (.jp2, .j2k)	JP2ECW
JPEG2000 (.jp2, .j2k)	JP2MrSID
NOAA Polar Orbiter Level 1b Data Set (AVHRR)	L1B
Erdas 7.x .LAN and .GIS	LAN

Format	Shortname
Daylon Leveller Heightfield	Leveller
In Memory Raster	MEM
Vexcel MFF	MFF
Multi-resolution Seamless Image Database	MrSID
Meteosat Second Generation	MSG
NDF	NLAPS Data Format
NITF	NITF
NetCDF	netCDF
OGDI Bridge	OGDI
PCI .aux Labelled	PAux
PCI Geomatics Database File	PCIDSK
Portable Network Graphics (.png)	PNG
PCRaster (.map)	PCRaster
Netpbm (.ppm,.pgm)	PNM
Swedish Grid RIK (.rik)	RIK
RadarSat2 XML (product.xml)	RS2
ArcSDE Raster	SDE
Raster Product Format/RPF (a.toc)	RPFTOC
USGS SDTS DEM (*CATD.DDF)	SDTS
Raster Matrix Format (*.rsw, .mtw)	RMF
SAR CEOS	SAR_CEOS
SGI Image Format	SGI
SRTM HGT Format	SRTMHGT
USGS ASCII DEM (.dem)	USGSDEM
Terragen Heightfield (.ter)	TERRAGEN
TerraSAR-X Product	TSX
GDAL Virtual (.VRT)	VRT
OGC Web Coverage Server	WCS
OGC Web Map Server	WMS
X11 Pixmap (.xpm)	XPM

3.1.2 Vector Formats

MapServer uses the “OGR” library for vector data access. OGR is actually part of the GDAL library. The following is a list of the supported vector formats, but for an uptodate list you should always check the OGR website: http://www.gdal.org/ogr/ogr_formats.html

Format	Shortname
Arc/Info Binary Coverage	AVCBin
Atlas BNA	BNA
Comma Separated Value (.csv)	CSV
DODS/OPeNDAP	DODS
ESRI Personal GeoDatabase	PGeo
ESRI ArcSDE	SDE
ESRI Shapefile	ESRI Shapefile
FMEObjects Gateway	FMEObjects Gateway
GeoJSON	GeoJSON
Géoconcept Export	Geoconcept
GML	GML
GMT	GMT
GPX	GPX
GRASS	GRASS
INTERLIS	"Interlis 1" and "Interlis 2"
KML	KML
Mapinfo File	MapInfo File
Microstation DGN	DGN
Memory	Memory
MySQL	MySQL
OGDI Vectors	OGDI
ODBC	ODBC
Oracle Spatial	OCI

Format	Shortname
PostgreSQL	PostgreSQL
S-57 (ENC)	S57
SDTS	SDTS
SQLite	SQLite
UK_NTF	UK. NTF
U.S. Census TIGER/Line	TIGER
VRT - Virtual Datasource	VRT
X-Plane/Flighgear aeronautical data	XPLANE
Informix DataBlade	IDB

3.2 The .Map File

The .map file is the basic configuration file for data access and styling for MapServer. The file is an ASCII text file, and is made up of different objects. Each object has a variety of parameters available for it. All .map file (or “mapfile) parameters are documented at:

<http://mapserver.gis.umn.edu/docs/reference/mapfile>

A simple mapfile example displaying only one layer follows, as well as the map image output:

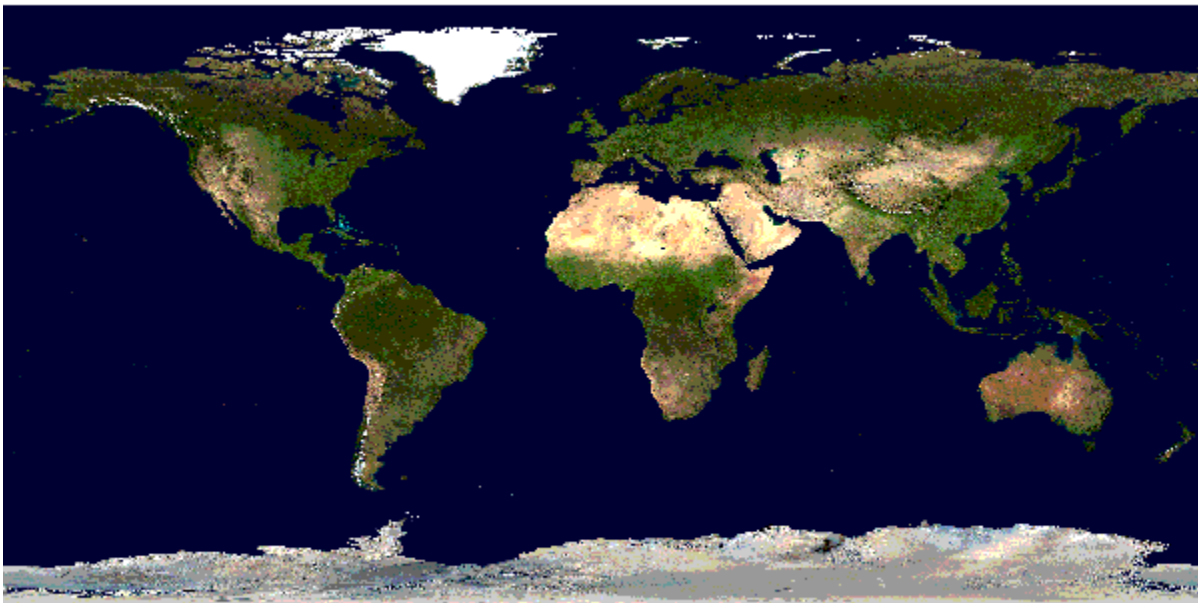
```

NAME "sample"
STATUS ON
SIZE 600 400
SYMBOLSET "../etc/symbols.txt"
EXTENT -180 -90 180 90
UNITS DD
SHAPEPATH "../data"
IMAGECOLOR 255 255 255
FONTSET "../etc/fonts.txt"
#
# Start of web interface definition
#
WEB
  IMAGEPATH "/ms4w/tmp/ms_tmp/"

```



```
IMAGEURL "/ms_tmp/"  
END  
  
#  
# Start of layer definitions  
#  
LAYER  
NAME 'global-raster'  
TYPE RASTER  
STATUS DEFAULT  
DATA bluemarble.gif  
END
```



3.2.1 Important Notes

- Comments in a mapfile are specified with a '#' character
- MapServer parses mapfiles from top to bottom, therefore layers at the end of the mapfile will be drawn last (meaning they will be displayed on top of other layers)

- Using relative paths is always recommended
- Paths should be quoted (single or double quotes are accepted)

3.2.2 Example Objects

Map Object

```
MAP
  NAME      "sample"
  EXTENT    -180 -90 180 90 # Geographic
  SIZE      800 400
  IMAGECOLOR 128 128 255
END
```

- EXTENT is the output extent in the units of the output map
- SIZE is the width and height of the map image in pixels
- IMAGECOLOR is the default image background color

Layer Object

- starting with MapServer 5.0, there is no limit to the number of layers in a mapfile
- DATA parameter is relative to the SHAPEPATH parameter the Map Object
- if no DATA extension is provided in the filename MapServer will assume it is an ESRI shapefile (.shp)

Rasters

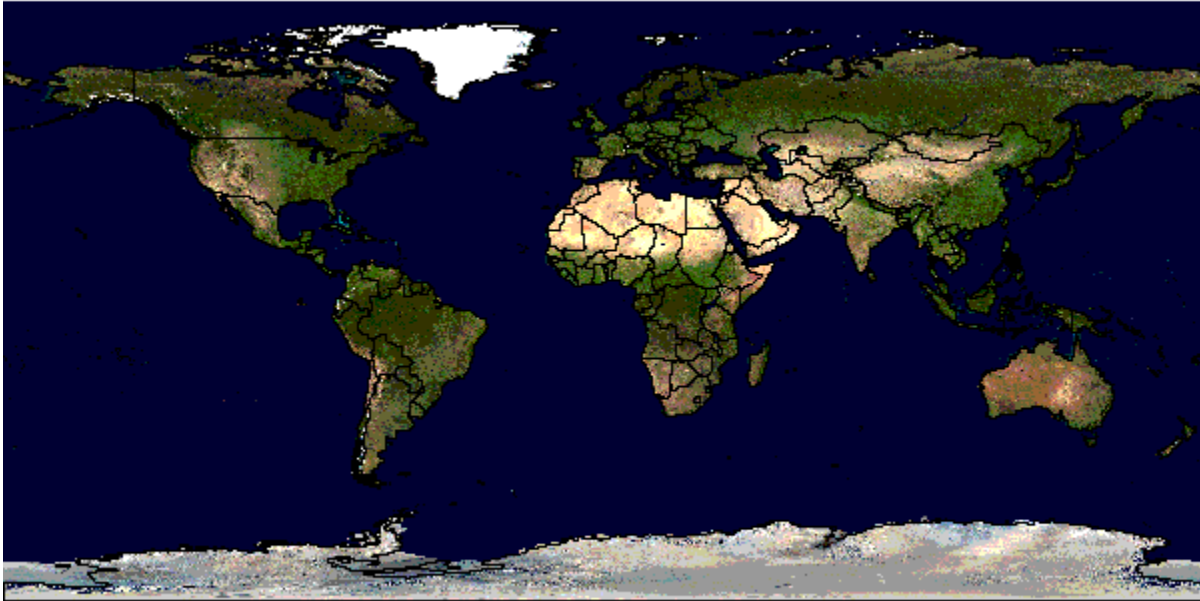
```
LAYER
  NAME bathymetry
  TYPE RASTER
  STATUS DEFAULT
  DATA bath_mapserver.tif
END
```

- See the Raster Data Access guide for more information:
http://mapserver.gis.umn.edu/docs/howto/raster_data

Vectors

- See the Vector Data Access guide for more information on displaying specific vector formats: http://mapserver.gis.umn.edu/docs/reference/vector_data
- vector layers of TYPE point, line, or polygon can be displayed
- the following example shows how to display only lines from a TYPE polygon layer, using the OUTLINECOLOR parameter:

```
LAYER
  NAME "world_poly"
  DATA 'shapefile/countries_area.shp'
  STATUS ON
  TYPE POLYGON
  CLASS
    NAME 'The World'
    STYLE
      OUTLINECOLOR 0 0 0
  END
END
END # layer
```



CLASS and STYLE Objects

- typical styling information is stored within the CLASS and STYLE objects of a LAYER
- starting with MapServer 5.0, there is no limit to the number of classes or styles in a mapfile
- the following example shows how to display a road line with two colors by using overlaid STYLE objects

CLASS

NAME "Primary Roads"

STYLE

SYMBOL "circle"

COLOR 178 114 1

SIZE 15

END #style1

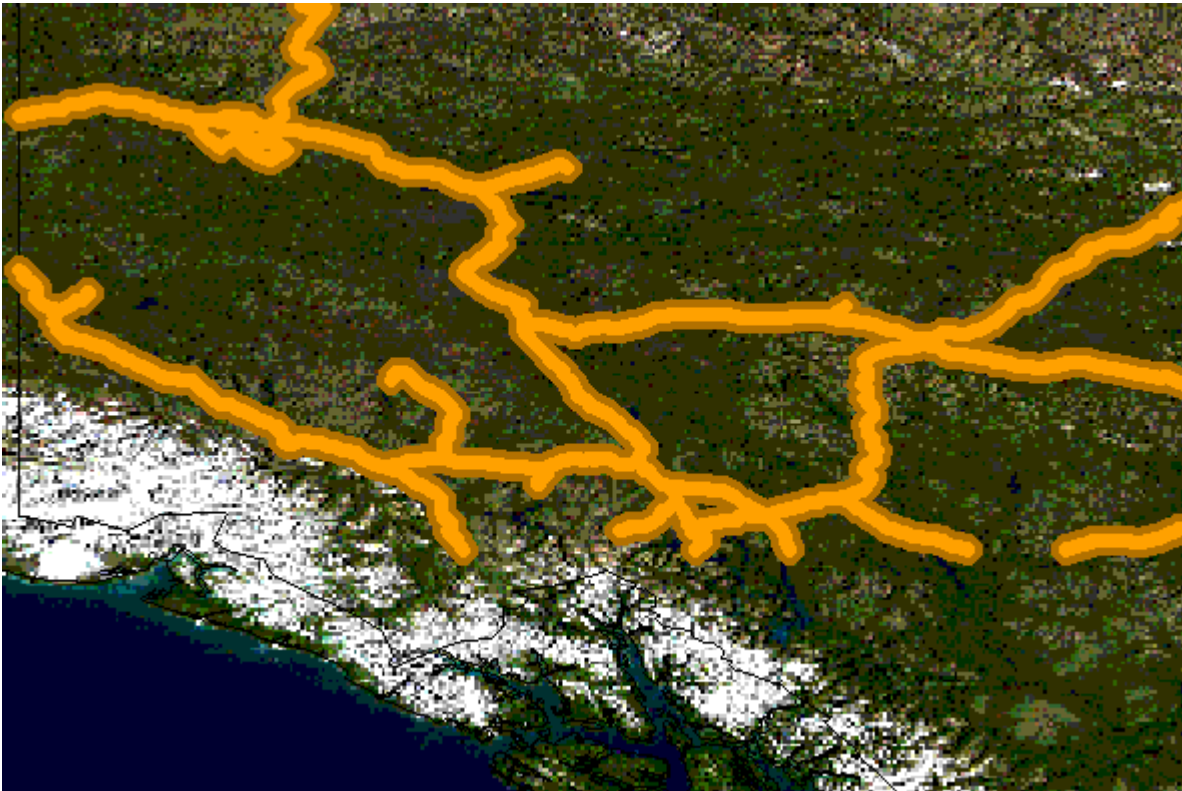
STYLE

SYMBOL "circle"

COLOR 254 161 0

SIZE 7

```
END #style2  
END
```



Symbols

- can be defined directly in the mapfile, or in a separate file
- the separate file method must use the SYMBOLSET parameter in the MAP object:

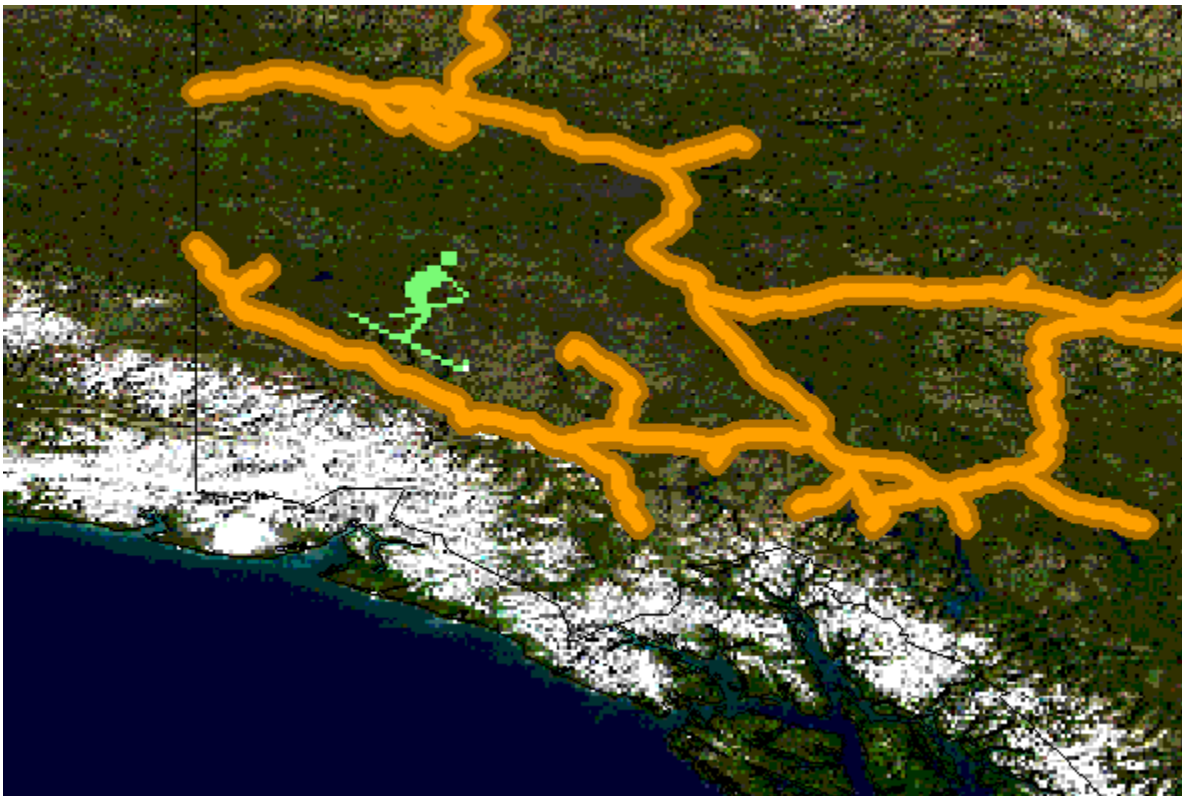
```
MAP  
  NAME      "sample"  
  EXTENT    -180 -90 180 90 # Geographic  
  SIZE      800 400  
  IMAGECOLOR 128 128 255  
  SYMBOLSET "../etc/symbols.txt"  
END
```

where symbols.txt might contain:

```
Symbol
Name 'ski'
Type PIXMAP
IMAGE "ski.gif"
END
```

and the mapfile would contain:

```
LAYER
...
CLASS
NAME "Ski Area"
STYLE
SYMBOL "ski"
END
END
END # layer
```



More information about symbols can be found at:

<http://mapserver.gis.umn.edu/docs/reference/symbology/referencemanual-all-pages>

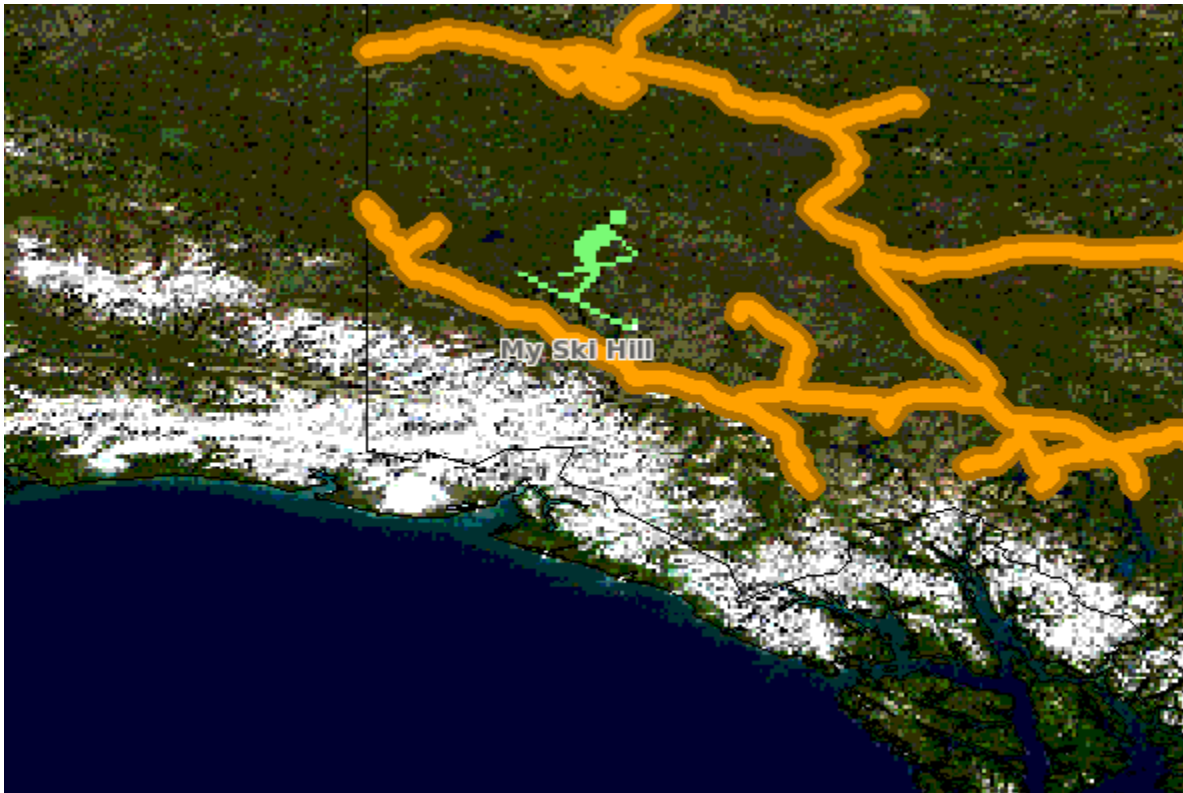
Label Objects

- defined within a LAYER object
- the LABELITEM parameters in the LAYER object can be used to label by a specific column in the data
- refer to a FONSET file, that is set in the MAP object, that contains a reference to the available font names
- structure of a FONTSET file is <lookup name> <filename>, for example a typical fontset file might look like the following:

sans	Vera.ttf
sans-bold	VeraBd.ttf
sans-bold-oblique	VeraBI.ttf
sans-oblique	VeraIt.ttf
serif	VeraSe.ttf
serif-bold	VeraSeBd.ttf

- an example LABEL object that references one of the above fonts might look like:

```
LABEL
FONT "sans-bold"
TYPE truetype
SIZE 10
POSITION LC
PARTIALS FALSE
COLOR 100 100 100
OUTLINECOLOR 242 236 230
END # label
```



3.2.3 CLASS Expressions

MapServer supports 3 types of CLASS expressions in a LAYER:

- 1) String comparisons (EXPRESSION "africa")
- 2) Regular expressions (EXPRESSION /^9|^10/)
- 3) Logical expressions ([POPULATION] > 50000 AND '[LANGUAGE]' eq 'FRENCH')

An important note is that logical expressions should be avoided wherever possible as they are very costly in terms of drawing time.

3.2.4 Mapfile Includes

Added to MapServer 4.10, any part of the mapfile can now be stored in a separate file and added to the main mapfile using the INCLUDE parameter. The filename to be included can have any extension, and it is always relative to the main .map file. How does this impact mapfile creation/management? Huge. Here are some potential uses:

- LAYERS can be stored in files and included to any number of applications

- STYLES can also be stored and included in multiple applications

The following is an example of including one layer:

if 'shadedrelief.lay' contains:

```
LAYER
  NAME      'shadedrelief'
  STATUS    ON
  TYPE      RASTER
  DATA     'GLOBALeb3colshade.jpg'
END
```

therefore the main mapfile would contain:

```
MAP
...
  INCLUDE "shadedrelief.lay"
...
END
```

The following is an example of a mapfile where all LAYERS are in separate .lay files, and all other objects (WEB, REFERENCE, SCALEBAR, etc.) are stored in a “.ref” file:

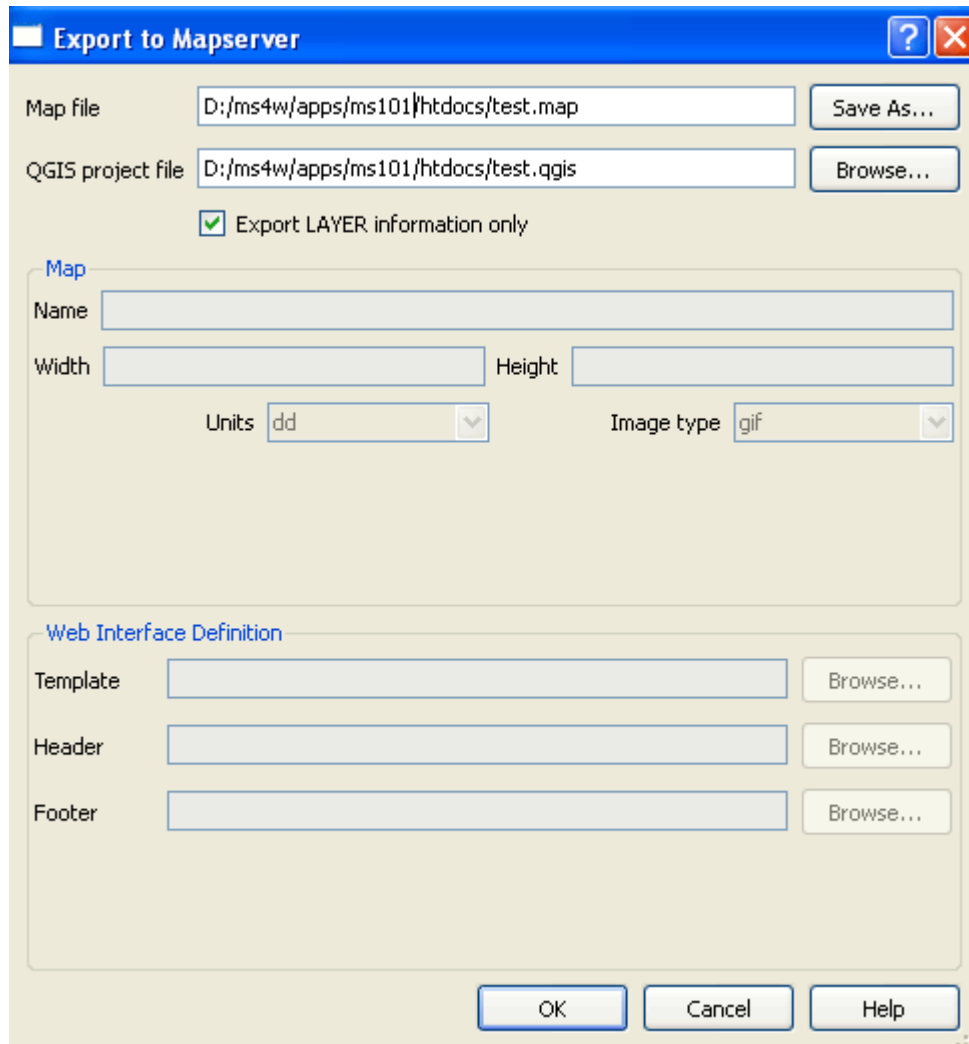
```
NAME "base"
#
# include reference objects
#
INCLUDE "../templates/template.ref"
#
# Start of layer definitions
#
INCLUDE "../layers/usa/usa_outline.lay"
INCLUDE "../layers/canada/base/1m/provinces.lay"
INCLUDE "../layers/canada/base/1m/roads_atlas_of_canada_1m.lay"
INCLUDE "../layers/canada/base/1m/roads_atlas_of_canada_1m_shields.lay"
INCLUDE "../layers/canada/base/1m/populated_places.lay"
END # Map File
```

3.2.5 Mapfile Creation Tools

A variety of stable tools now exist to help you with the creation of thematic mapfiles.

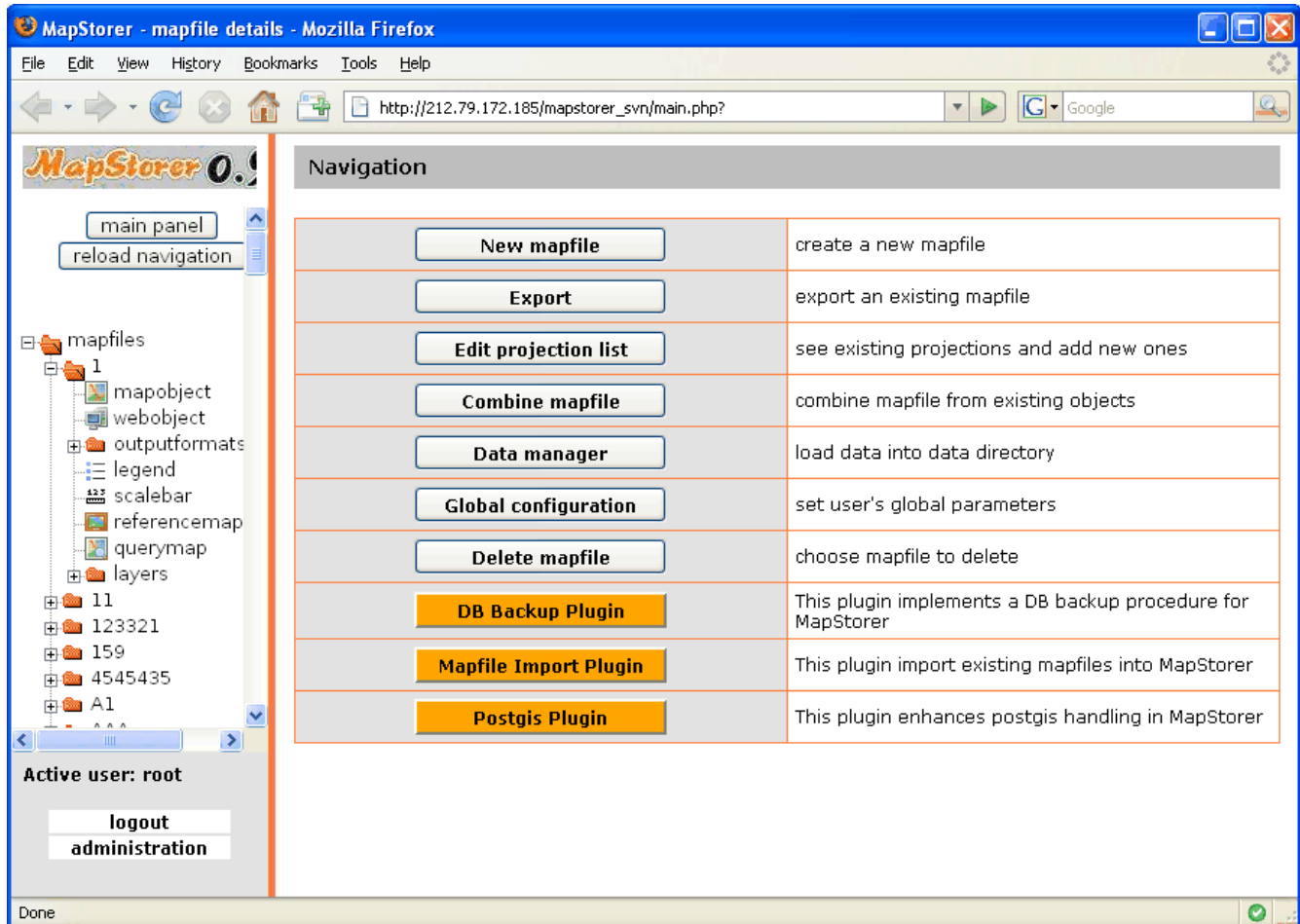
Quantum GIS (QGIS)

The most promising out of the options is QGIS (<http://www.qgis.org/>) and its “File/Export to MapServer Mapfile” utility. QGIS is a desktop Open Source GIS that uses the same underlying data access libraries as MapServer (GDAL/OGR) in a desktop environment, so that means you can classify your data to your heart's content using nice color pickers, and save to a .map file. Give it a try! Note that this utility requires that you have Python 2.5 installed as well. A screengrab of the utility follows:



MapStorer

MapStorer (<http://www.mapstorer.org/>) is an Open Source project that uses a web front-end to manage mapfiles, and store them in a relational database.



3.3 MapServer Utilities

Several utilities come with MapServer builds, and are very useful. Several packages include utilities:

- OSGeo4W (utils are available through shell window)
- FGS Linux Installer (run setenv.sh first)

3.3.1 legend

Purpose:

Creates a legend from a mapfile. Output is either PNG or GIF depending on what version of the GD library used.

Syntax:

legend [mapfile] [output image]

Example:

legend vector_blank.map legend.png



3.3.2 msencrypt

Purpose:

Used to create an encryption key or to encrypt portions of connection strings for use in mapfiles (added in v4.10). Typically you might want to encrypt portions of the CONNECTION parameter for a database connection. The following CONNECTIONTYPES are supported for using this encryption method:

- OGR
- Oracle Spatial
- PostGIS
- SDE

Syntax:

To create a new encryption key:

msencrypt -keygen [key_filename]

To encrypt a string:

msencrypt -key [key_filename] [string_to_encrypt]

Use in Mapfile:

- The location of the encryption key can be specified by two mechanisms, either by setting the environment variable `MS_ENCRYPTION_KEY` or using a `CONFIG` directive in the `MAP` object of your mapfile. For example:

```
CONFIG MS_ENCRYPTION_KEY "/path/to/mykey.txt"
```

- Use the `{` and `}` characters as delimiters for encrypted strings inside database `CONNECTIONS` in your mapfile. For example:

```
CONNECTIONTYPE ORACLESPATIAL  
CONNECTION "user/{MIIBugIBAAKBgQCP0Yj+Seh8==}@service"
```

Example:

(note: the following PostGIS example requires at least MapServer 5.0.3 or 5.2)

Let's say we have a `LAYER` that uses a `POSTGIS` connection as follows:

```
LAYER  
  NAME "provinces"  
  TYPE POLYGON  
  CONNECTIONTYPE POSTGIS  
  CONNECTION "host=127.0.0.1 dbname=gmap user=postgres password=iluvyou18 port=5432"  
  DATA "the_geom FROM province using SRID=42304"  
  STATUS DEFAULT  
  CLASS  
    NAME "Countries"  
    COLOR 255 0 0  
  END  
END
```

Here are the steps to encrypt the password in the above connection:

1. Generate an encryption key (note that this key should not be stored anywhere within your web server's accessible directories):

```
msencrypt -keygen "E:\temp\mykey.txt"
```

And this generated key file might contain something like:

```
2137FEFDB5611448738D9FBB1DC59055
```

2. Encrypt the connection's password using that generated key:

```
msencrypt -key "E:\temp\mykey.txt" "iluvyou18"
```

Which returns the password encrypted, at the commandline (you'll use it in a second):

```
3656026A23DBAFC04C402EDFAB7CE714
```

3. Edit the mapfile to make sure the 'mykey.txt' can be found, using the "MS_ENCRYPTION_KEY" environment variable. The CONFIG parameter inside the MAP object can be used to set an environment variable inside a mapfile:

```
MAP
...
CONFIG "MS_ENCRYPTION_KEY" "E:/temp/mykey.txt"
...
END #mapfile
```

4. Modify the layer's CONNECTION to use the generated password key, making sure to use the “{}” brackets around the key:

```
CONNECTION "host=127.0.0.1 dbname=gmap user=postgres
password={3656026A23DBAFC04C402EDFAB7CE714} port=5432"
```

5. Done! Give your new encrypted mapfile a try with the shp2img utility!

3.3.3 scalebar

Purpose:

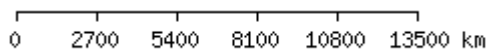
Creates a scalebar from a mapfile. Output is either PNG or GIF depending on what version of the GD library used.

Syntax:

scalebar [mapfile] [output image]

Example:

scalebar vector_blank.map scalebar.png



3.3.4 shp2img

Purpose:

Creates a map image from a mapfile. Output is either PNG or GIF depending on what version of the GD library used. This is a useful utility to test your mapfile. You can simply provide the path to your mapfile and the name of an output image, and an image should be returned. If an image cannot be created an error will be displayed at the command line that should refer to a line number in the mapfile.

Syntax:

```
shp2img -m mapfile [-o image] [-e minx miny maxx maxy] [-s sizex sizey]
        [-l layer1 "[layers2...] "]" [-i format] [-p n]
        [-all_debug n] [-map_debug n] [-layer_debug n]
```

-m mapfile: Map file to operate on - required.

-i format: Override the IMAGETYPE value to pick output format.

-o image: output filename (stdout if not provided)

-e minx miny maxx maxy: extents to render

-s sizex sizey: output image size

-l layers: layers to enable - make sure they are quoted and space seperated if more than one listed.

-all_debug n: Set debug level for map and all layers.

-map_debug n: Set map debug level.

-layer_debug layer_name n: Set layer debug level.

-p n: pause for n seconds after reading the map

Example #1:

```
shp2img -m vector_blank.map -o test.png
```

Result: A file named “test.png” is created

Example #2:

```
shp2img -m vector_blank.map -o test2.png -map_debug 3
```

Result: A file named “test2.png” is created, and layer draw speed are returned:

```
msDrawRasterLayerLow(global-raster): entering.
msDrawMap(): Layer 0 (global-raster), 0.047s
msDrawMap(): Layer 1 (detailed_countries), 0.343s
msDrawMap(): Layer 2 (tab_yukon), 0.000s
msDrawMap(): Layer 3 (tab_yukon_shields), 0.000s
msDrawMap(): Layer 4 (gml_where), 0.000s
msDrawMap(): Drawing Label Cache, 0.016s
msDrawMap() total time: 0.406s
```

3.3.5 shptree

Purpose:

Creates a quadtree-based spatial index for a shapefile. The default tree depth is calculated so that each tree node (quadtree cell) contains 8 shapes. Do not use the default with point files, a value between 6 and 10 seems to work ok. Your millage may vary and you'll need to do some experimenting.

Description: This utility is a must for any MapServer application that uses shapefiles. Shptree creates a spatial index of your shapefile, using a quadtree method. This means that MapServer will use this index to quickly find the appropriate shapes to draw. It creates a file of the same name as your shapefile, with a *.qix* file extension. The quadtree method breaks the file into 4 quadrants, recursively until only a few shapes are contained in each quadrant. This minimum number can be set with the `<depth>` parameter of the command. More information on this command can be found at: <http://mapserver.gis.umn.edu/cgi-bin/wiki.pl?ShpTree>

Syntax:

```
shptree <shpfile> [<depth>] [<index_format>]
```

Where:

`<shpfile>` is the name of the *.shp* file to index.

`<depth>` (optional) is the maximum depth of the index to create, default is 0 meaning that shptree will calculate a reasonable default depth.

`<index_format>` (optional) is one of:

NL: LSB byte order, using new index format

NM: MSB byte order, using new index format

The following old format options are deprecated

N: Native byte order

L: LSB (intel) byte order

M: MSB byte order

The default `index_format` on this system is: NL

Example:

```
shptree us_states.shp
creating index of new LSB format
```

Result: A file named 'us_states.qix' is created in the same location. (note that you can use the shptreevis utility, described next, to view the actual quadtree quadrants that are used by MapServer in this qix file)

Mapfile Notes:

Shapefiles are native to MapServer, and therefore do not require the *.shp* extension in the DATA path of the LAYER. In fact, in order for MapServer to use the *.qix* extension you MUST NOT specify the extension, for example:

```
LAYER
```

```
...
```



```
DATA us_states #MapServer will search for us_states.qix and will use it
...
END

LAYER
...
DATA us_states.shp #MapServer will search for us_states.shp.qix and won't find it
...
END
```

3.3.6 shptreevis

Purpose:

This utility can be used to view the quadtree quadrants that are part of a .qix file (that was created with the shptree utility).

Syntax:

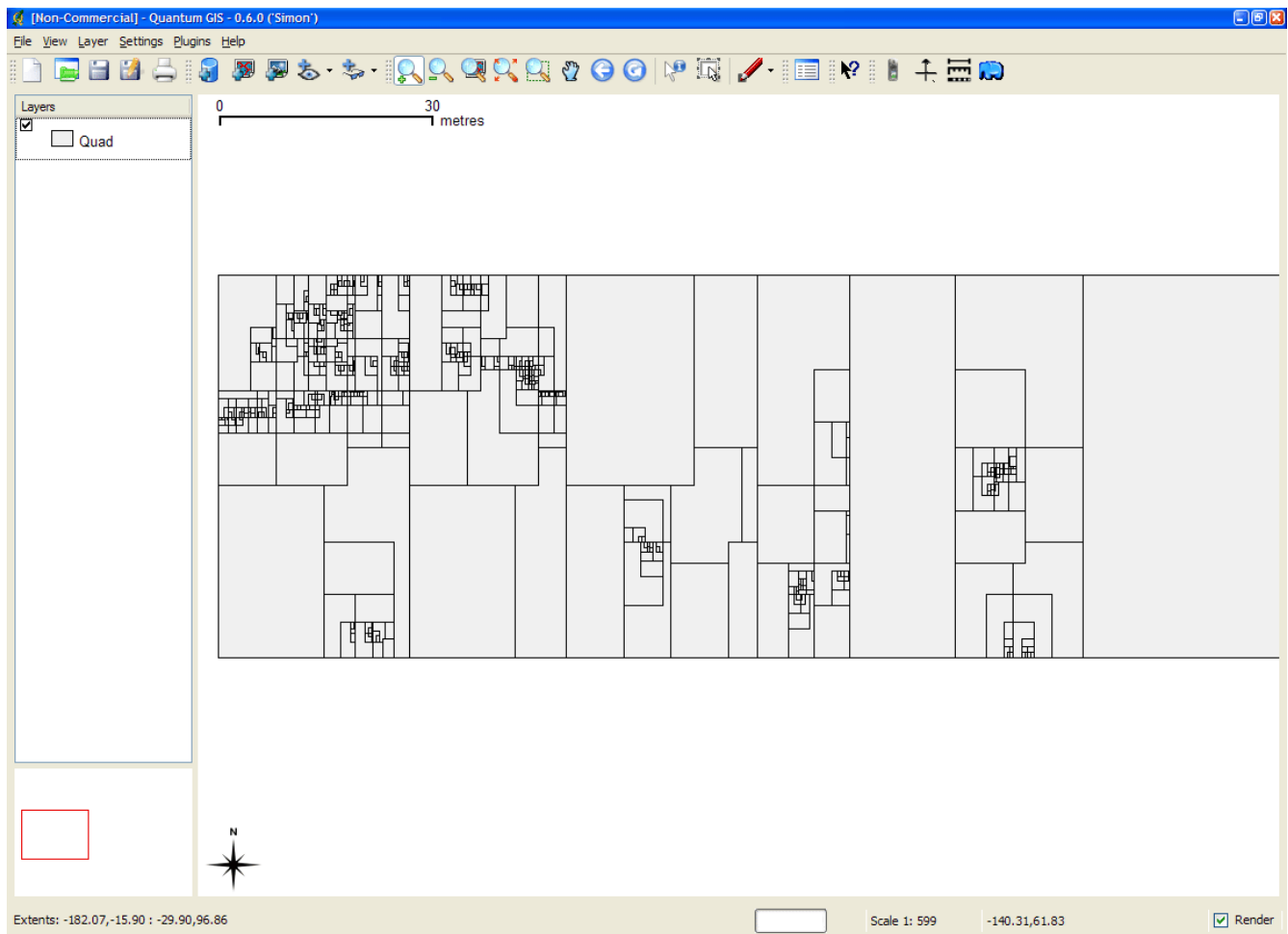
shptreevis shapefile new_shapefile

Example:

shptreevis us_states.shp quad.shp

This new LSB index supports a shapefile with 2895 shapes, 10 depth

Result:A shapefile named 'quad.shp' is created. You can now view this shapefile in a desktop GIS (such as [QGIS](#) for example) to see the quadtrees that were created with the shptree command, as shown in the following image.



3.3.7 sortshp

Purpose:

Sorts a shapefile based on a single column in ascending or descending order. Supports INTEGER, DOUBLE and STRING column types. Useful for prioritizing shapes for rendering and/or labeling. Description: The idea here is that if you know that you need to display a certain attribute classed by a certain value, it will be faster for MapServer to access that value if it is at the beginning of the attribute file.

Syntax:

```
sortshp [infile] [outfile] [item] [ascending|descending]
```

Example:

This example uses a roads file ('roads_ugl') that has a field with road classes in integer format ('class1').

```
sortshp roads_ugl roads-sort class1 ascending
```

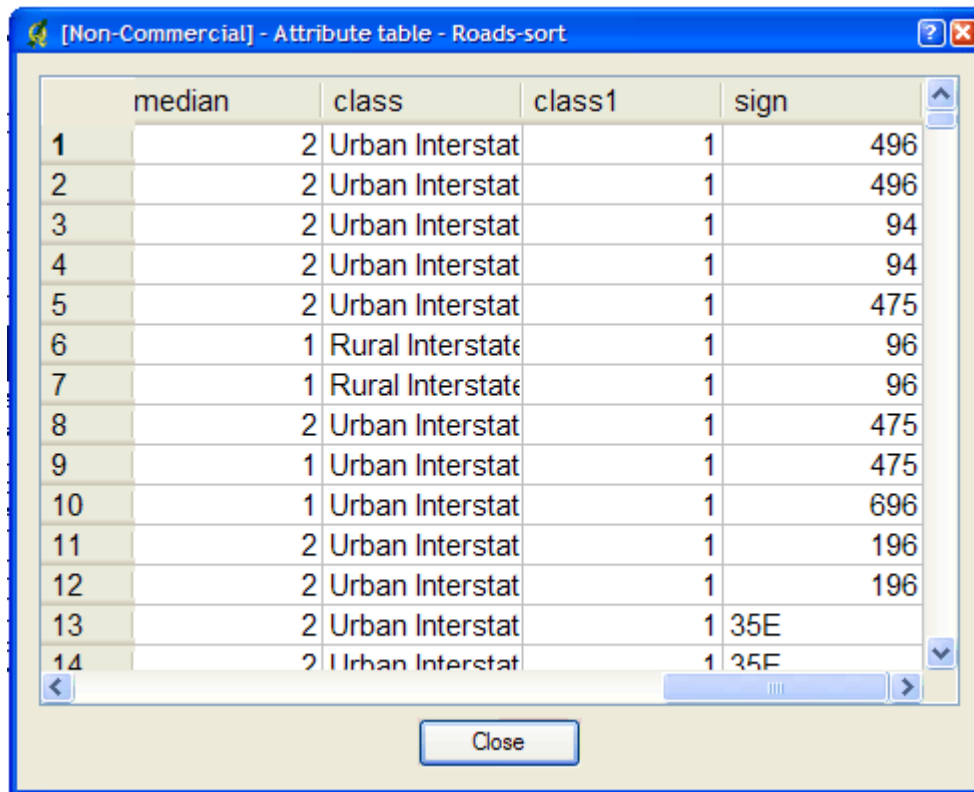
Result:

A new shapefile named 'roads-sort.shp' is created with shapes sorted in ascending order, according to the values in the 'class1' field, as shown below.

Before:

	median	class	class1	sign
1		2 Urban Principa	2	
2		0 Urban Principa	2	
3		2	4	42
4		2	4	42
5		2	4	
6		2 Urban Principa	3	28
7		2 Urban Principa	3	42
8		2	4	14
9		0 Urban Principa	2	
10		2	4	
11		2	4	
12		2 Urban Principa	2	51
13		2 Urban Principa	2	
14		?	?	169

After:



	median	class	class1	sign
1		2 Urban Interstat	1	496
2		2 Urban Interstat	1	496
3		2 Urban Interstat	1	94
4		2 Urban Interstat	1	94
5		2 Urban Interstat	1	475
6		1 Rural Interstate	1	96
7		1 Rural Interstate	1	96
8		2 Urban Interstat	1	475
9		1 Urban Interstat	1	475
10		1 Urban Interstat	1	696
11		2 Urban Interstat	1	196
12		2 Urban Interstat	1	196
13		2 Urban Interstat	1	35E
14		2 Urban Interstat	1	35F

3.3.8 tile4ms

Purpose:

Creates a tile index shapefile for use with MapServer's TILEINDEX feature. The program creates a shapefile of rectangles from extents of all the shapefiles listed in [metafile] (one shapefile name per line) and the associated DBF with the filename for each shape tile in a column called LOCATION as required by mapserv. Description: This utility creates a shapefile containing the MBR (minimum bounding rectangle) of all shapes in the files provided, which can then be used in the LAYER object's TILEINDEX parameter of the mapfile. The new file created with this command is used by MapServer to only load the files associated with that extent (or tile).

Syntax:

tile4ms <meta-file> <tile-file> [-tile-path-only]

<meta-file> INPUT file containing list of shapefile names
(complete paths 255 chars max, no extension)

<tile-file> OUTPUT shape file of extent rectangles and names
of tiles in <tile-file>.dbf

-tile-path-only Optional flag. If specified then only the path to the shape files will be stored in the LOCATION field instead of storing the full filename.

Short Example:

create tileindex.shp for all tiles under the /path/to/data directory:

<on Unix>

```
cd /path/to/data
find . -name "/*.shp" -print > metafile.txt
tile4ms metafile.txt tileindex
```

<on Windows>

```
dir /b /s *.shp > metafile.txt
tile4ms metafile.txt tileindex
```

Long Example:

This example uses TIGER Census data, where the data contains files divided up by county (in fact there are over 3200 counties, a very large dataset indeed). In this example we will show how to display all lakes for the state of Minnesota. (note that here we have already converted the TIGER data into shapefile format, but you could keep the data in TIGER format and use the ogrindex utility instead) The TIGER Census data for Minnesota is made up of 87 different counties, each containing its own lakes file ('wp.shp').

1. We need to create the 'meta-file' for the tile4ms command. This is a text file of the paths to all 'wp.shp' files for the MN state. To create this file we can use a few simple commands:

```
DOS: dir wp.shp /b /s > wp_list.txt
(this includes full paths to the data, you might want to edit the txt
file to remove the full path)
```

```
UNIX: find -name *wp.shp -print > wp_list.txt
```

The newly created file might look like the following (after removing the full path):

```
001\wp.shp
003\wp.shp
005\wp.shp
007\wp.shp
009\wp.shp
011\wp.shp
013\wp.shp
015\wp.shp
017\wp.shp
```

019\wp.shp

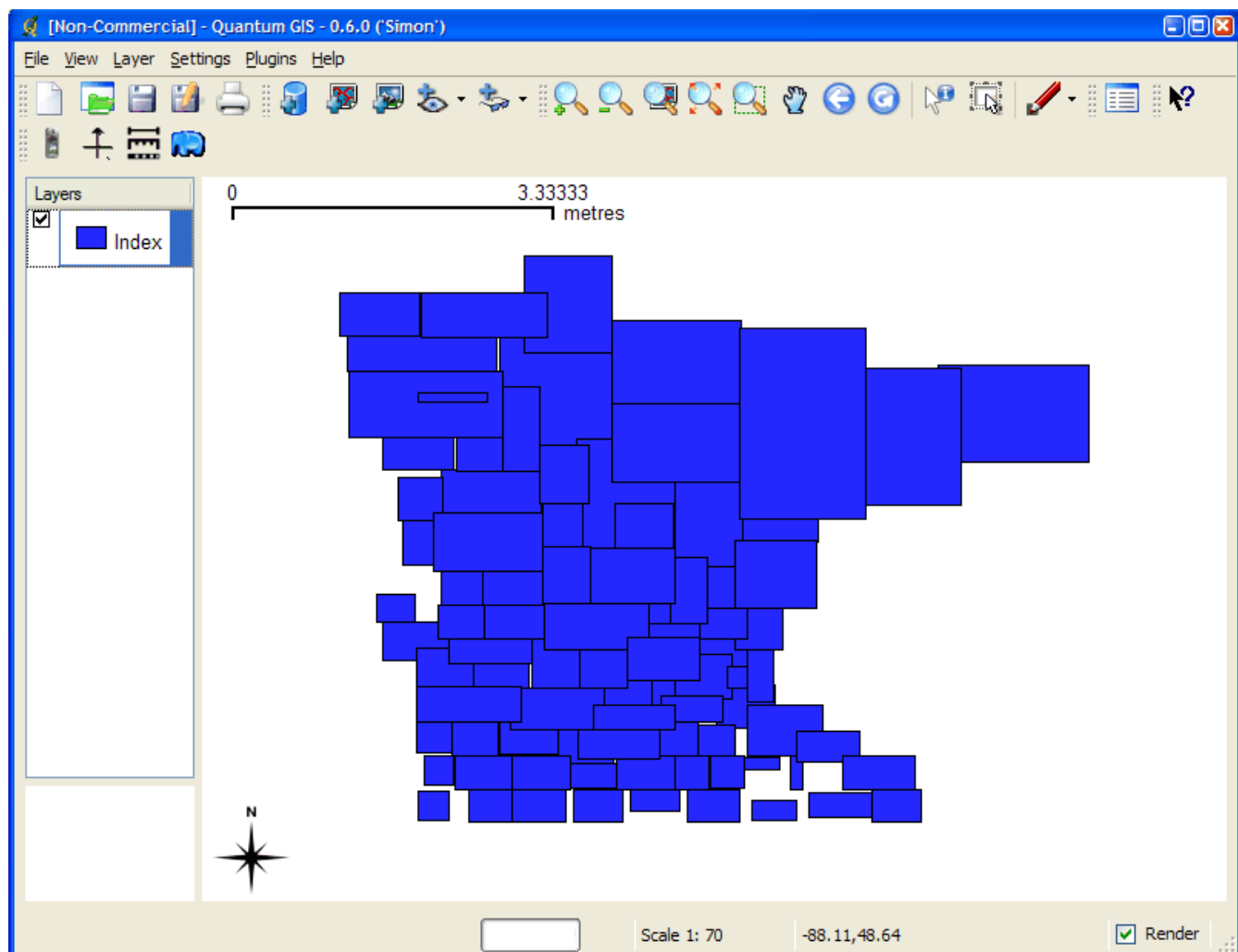
...

2. Execute the tile4ms command with the newly created meta-file to create the index file:

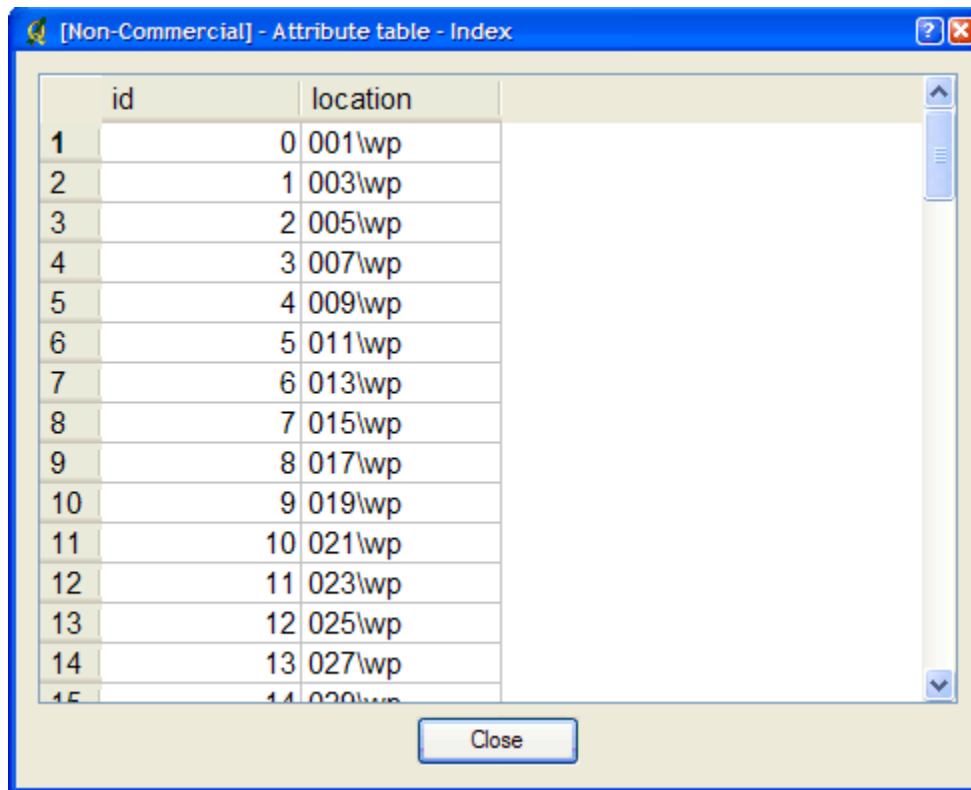
```
tile4ms wp_list.txt index
Processed 87 of 87 files
```

3. A new file named 'index.shp' is created. This is the index file with the MBRs of all 'wp.shp' files for the entire state, as shown in the first image below. The attribute table of this file contains a field named 'LOCATION', that contains the path to each 'wp.shp file', as shown in the second image below.

Index file created by tile4ms:



Attributes of index file created by tile4ms:



id	location
1	0 001\wp
2	1 003\wp
3	2 005\wp
4	3 007\wp
5	4 009\wp
6	5 011\wp
7	6 013\wp
8	7 015\wp
9	8 017\wp
10	9 019\wp
11	10 021\wp
12	11 023\wp
13	12 025\wp
14	13 027\wp
15	14 029\wp

4. The final step is to use this in your mapfile.
 - LAYER object's TILEINDEX - must point to the location of the index file
 - LAYER object's TILEITEM - specify the name of the field in the index file containing the paths (default is 'location')
 - do not need to use the LAYER's DATA parameter

For example:

```
LAYER
  NAME 'mn-lakes'
  STATUS ON
  TILEINDEX "index"
  TILEITEM "location"
  TYPE POLYGON
  CLASS
    NAME "mn-lakes"
    STYLE
      COLOR 0 0 255
    END
  END
END
```

When you view the layer in a MapServer application, you will notice that when you are zoomed into a

small area of the state only those lakes layers are loaded, which speeds up the application.

3.4 GDAL/OGR Utilities

There are several commandline utilities that come with GDAL/OGR builds. Getting to know the utilities is important for efficiently displaying data in MapServer, since GDAL/OGR is the library that handles data connections in MapServer.

3.4.1 ogrinfo

Description:

The ogrinfo command lists information about an OGR supported vector data source. When working with vector data in MapServer, the ogrinfo command should be the first thing you do with the data. You might use ogrinfo to: 1) see if your format is supported by the MapServer build you are using, 2) get the extents of a dataset, to use for the EXTENT value in the mapfile, 3) see field/column information in your data, or 4) execute SQL to a supported database connection.

Usage:

```
ogrinfo [-ro] [-q] [-where restricted_where]
```

```
    [-spat xmin ymin xmax ymax] [-fid fid]
    [-sql statement] [-al] [-so] [--formats]
    datasource_name [layer [layer ...]]
```

-ro:

Open the data source in read-only mode.

-al:

List all features of all layers (used instead of having to give layer names as arguments).

-so:

Summary Only: suppress listing of features, show only the summary information like projection, schema, feature count and extents.

-q:

Quiet verbose reporting of various information, including coordinate system, layer schema, extents, and feature count.

-where *restricted_where*:

An attribute query in a restricted form of the queries used in the SQL WHERE statement. Only features matching the attribute query will be reported.

-sql *statement*:

Execute the indicated statement and return the result.

-spat *xmin ymin xmax ymax*:

The area of interest. Only features within the rectangle will be reported.

-fid *fid*:

If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries.

--formats:

List the format drivers that are enabled.

datasource_name:

The data source to open. May be a filename, directory or other virtual name. See the [OGR Vector Formats](#) list for supported datasources.

layer:

One or more layer names may be reported.

Example1: Get the List of Supported Formats

```
> ogrinfo --formats
```

Supported Formats:

```
-> "ESRI Shapefile" (read/write)
```

```
-> "MapInfo File" (read/write)
```

```
-> "UK .NTF" (readonly)
```

```
-> "SDTS" (readonly)
```

```
-> "TIGER" (read/write)
```

...

Example2: Get a Summary of the Dataset

```
>ogrinfo WorldCities.shp WorldCities -summary
```

```
INFO: Open of `WorldCities.shp'
```

```
using driver `ESRI Shapefile' successful.
```

```
Layer name: WorldCities
```

```
Geometry: Point
```

```
Feature Count: 12686
```

```
Extent: (-178.166667, -54.800000) - (179.383333, 78.933333)
```

```
Layer SRS WKT:
```

```
(unknown)
```

```
ID: Integer (6.0)
```

```
ATTRIB: Integer (4.0)
```

```
MGCC: Integer (4.0)
```

```
ADMIN_CODE: Integer (4.0)
```

```
UFI: Integer (9.0)
```

```
POPULATION: Integer (9.0)
```

```
NAME: String (250.0)
```

3.4.2 ogr2ogr

Description:

This program can be used to convert between vector file formats, as well as: performing various operations during the process such as spatial or attribute selections, reducing the set of attributes, setting the output coordinate system, reprojecting the features during translation, and merge datasets.

Usage:

```
ogr2ogr [-skipfailures] [-append] [-update] [-f format_name]
        [-select field_list] [-where restricted_where]
        [-sql <sql statement>] [--help-general]
        [-spat xmin ymin xmax ymax] [-preserve_fid] [-fid FID]
        [-a_srs srs_def] [-t_srs srs_def] [-s_srs srs_def]
        [[-dsc NAME=VALUE] ...] dst_datasource_name
        src_datasource_name
        [-lco NAME=VALUE] [-nln name] [-nlt type] [layer [layer ...]]
```

-f *format_name*:

output file format name, some possible values are:

```
-f "ESRI Shapefile"
-f "TIGER"
-f "MapInfo File"
-f "GML"
-f "PostgreSQL"
```

-append:

Append to existing layer instead of creating new

-overwrite:

Delete the output layer and recreate it empty

-update:

Open existing output datasource in update mode rather than trying to create a new one

-select *field_list*:

Comma-delimited list of fields from input layer to copy to the new layer (defaults to all)

-sql *sql_statement*:

SQL statement to execute. The resulting table/layer will be saved to the output.

-where *restricted_where*:

Attribute query (like SQL WHERE)

-spat *xmin ymin xmax ymax*:

spatial query extents

-dsc *NAME=VALUE*:

Dataset creation option (format specific)

-lco *NAME=VALUE*:

Layer creation option (format specific)

-nln *name*:

Assign an alternate name to the new layer

-nlt *type*:

Define the geometry type for the created layer. One of NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTILINE, MULTIPOLYGON or MULTILINESTRING. Add "25D" to the name to get 2.5D versions.

- a_srs** *srs_def*:
Assign an output SRS
- t_srs** *srs_def*:
Reproject/transform to this SRS on output
- s_srs** *srs_def*:
Override source SRS
- fid** *fid*:
If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries.

Example1: Convert from a MapInfo File to an ESRI Shapefile

```
>ogr2ogr -f "ESRI Shapefile" prioadll.shp prioadll.TAB
```

Example2: Reproject a Shapefile from EPSG:4326 to EPSG:42304

```
>ogr2ogr -f "ESRI Shapefile" -t_srs EPSG:42304 -s_srs EPSG:4326 prioadll_lcc.shp prioadll.shp
```

Example3: Merge spatial data from fileB.shp into fileA.shp

```
>ogr2ogr -update -append -f "ESRI Shapefile" fileA.shp fileB.shp -nln fileA
```

3.4.3 ogrtindex

Description:

The ogrtindex program can be used to create a tileindex - a file containing a list of the identities of a bunch of other files along with there spatial extents. This is primarily intended to be used with MapServer for tiled access to layers using the OGR connection type. This is similar to the MapServer utility 'tile4ms', where tile4ms works only on shapefiles, and ogrtindex will work with any OGR supported format.

Usage:

```
ogrindex [-lnum n]... [-lname name]... [-f output_format]
         [-write_absolute_path] [-skip_different_projection]
         output_dataset src_dataset...
```

Example: Create Index File for all MapInfo Files in directory

```
> ogrindex index.shp mapinfo\*.TAB
```

For a thorough example of using a tileindex in MapServer, see the 'tile4ms' section of this document.

3.4.4 gdalinfo

Description:

The `gdalinfo` command lists information about a GDAL supported raster data source. When working with raster data in MapServer, the `gdalinfo` command should be the first thing you do with the data. You might use `gdalinfo` to: 1) see if your format is supported by the MapServer build you are using, 2) get the extents of a raster, to use for the `EXTENT` value in the mapfile.

Usage:

```
gdalinfo [--help-general] [-mm] [-stats] [-nogcp] [-nomd]
        [-noct] [-checksum] [-mdd domain]* datasetname
```

-mm

Force computation of the actual min/max values for each band in the dataset.

-stats

Read and display image statistics. Force computation if no statistics are stored in an image.

-nogcp

Suppress ground control points list printing. It may be useful for datasets with huge amount of GCPs, such as L1B AVHRR or HDF4 MODIS which contain thousands of the ones.

-nomd

Suppress metadata printing. Some datasets may contain a lot of metadata strings.

-noct

Suppress printing of color table.

-checksum

Force computation of the checksum for each band in the dataset.

-mdd domain

Report metadata for the specified domain

Example: Getting Information on a Landsat geotiff

```
>gdalinfo image1.tif
```

```
Driver: GTiff/GeoTIFF
```

```
Files: image1.tif
```

```
Size is 7966, 8274
```

```
Coordinate System is:
```

```
PROJCS["LCC      E008",
```

```
  GEOGCS["NAD83",
```

```
    DATUM["North_American_Datum_1983",
```

```
      SPHEROID["GRS 1980",6378137,298.2572221010002,
```

```
        AUTHORITY["EPSG","7019"]],
```

```
      AUTHORITY["EPSG","6269"]],
```

```
    PRIMEM["Greenwich",0],
```

```
UNIT["degree",0.0174532925199433],  
AUTHORITY["EPSG","4269"]],  
PROJECTION["Lambert_Conformal_Conic_2SP"],  
PARAMETER["standard_parallel_1",49],  
PARAMETER["standard_parallel_2",77],  
PARAMETER["latitude_of_origin",49],  
PARAMETER["central_meridian",-95],  
PARAMETER["false_easting",0],  
PARAMETER["false_northing",0],  
UNIT["metre",1,  
AUTHORITY["EPSG","9001"]]]
```

Origin = (2878735.262000000100000,995462.9749999999980000)

Pixel Size = (30.000000000000000,-30.000000000000000)

Metadata:

AREA_OR_POINT=Area

Image Structure Metadata:

INTERLEAVE=PIXEL

Corner Coordinates:

Upper Left (2878735.262, 995462.975) (52d35'49.00"W, 48d59'18.47"N)

Lower Left (2878735.262, 747242.975) (54d36'32.41"W, 47d13'6.98"N)

Upper Right (3117715.262, 995462.975) (50d 6'40.01"W, 47d37'48.85"N)

Lower Right (3117715.262, 747242.975) (52d 9'36.52"W, 45d55'23.39"N)

Center (2998225.262, 871352.975) (52d22'14.82"W, 47d27'4.74"N)

Band 1 Block=7966x1 Type=Byte, ColorInterp=Red

Overviews: 3983x4137, 1992x2069, 996x1035, 498x518

Band 2 Block=7966x1 Type=Byte, ColorInterp=Green

Overviews: 3983x4137, 1992x2069, 996x1035, 498x518

Band 3 Block=7966x1 Type=Byte, ColorInterp=Blue

Overviews: 3983x4137, 1992x2069, 996x1035, 498x518

3.4.5 gdaladdo

Description:

The gdaladdo utility can be used to build or rebuild overview images for most supported file formats using several downsampling algorithms. Creating overview images are a must for using large raster

images in MapServer.

Usage:

```
gdaladdo [-r {nearest,average,average_mp,average_magphase,mode}]  
        [--help-general] filename levels
```

Example1: Creating Overviews on a GeoTIFF

```
> gdaladdo image1.tif 2 4 8 16
```

Result1:

You will notice the filesize of image1.tif increase, and MapServer will use the embedded overviews in the geotiff with no other changes required.

Example2: Create Overviews on all GeoTIFFs in folder

on windows you would execute at the commandline:

```
> for %i in (*.tif) do gdaladdo %i 2 4 8 16
```

Example3: Create Overviews on all GeoTIFFs in folder and all sub-folders

on windows you would execute at the commandline:

```
> for /R %i in (*.tif) do gdaladdo %i 2 4 8 16
```

3.4.6 gdaltindex

Description:

Similar to tile4ms and ogrtindex, the gdaltindex creates an index of rasters to be used in MapServer. This program builds a shapefile with a record for each input raster file, an attribute containing the filename, and a polygon geometry outlining the raster. In MapServer, using an index file for all of your rasters avoids having to create a single layer for each raster. For a more thorough example of using tileindexes in MapServer see the 'tile4ms' section in this document.

Usage:

```
gdaltindex [-tileindex field_name] [-write_absolute_path] [-skip_different_projection] index_file [gdal_file]*
```

Example: Creating a Tileindex Shapefile of GeoTIFFs

```
> gdaltindex index_raster.shp *.tif
```

3.4.7 gdal_translate

Description:

The `gdal_translate` utility can be used to convert raster data between different formats, potentially performing some operations like reprojecting, subsettings, resampling, and rescaling pixels in the process.

Usage:

```
gdal_translate [--help-general]
  [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/
        CInt16/CInt32/CFloat32/CFloat64}] [-not_strict]
  [-of format] [-b band] [-outsize xsize[%] ysize[%]]
  [-scale [src_min src_max [dst_min dst_max]]]
  [-srewin xoff yoff xsize ysize] [-projwin ulx uly lrx lry]
  [-a_srs srs_def] [-a_ullr ulx uly lrx lry] [-a_nodata value]
  [-gcp pixel line easting northing]*
  [-mo "META-TAG=VALUE"]* [-quiet] [-sds]
  [-co "NAME=VALUE"]*
  src_dataset dst_dataset
```

-ot: *type*

For the output bands to be of the indicated data type.

-not_strict:

Be forgiving of mismatches and lost data when translating to the output format.

-of *format:*

Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-b *band:*

Select an input band *band* for output. Bands are numbered from 1. Multiple **-b** switches may be used to select a set of input bands to write to the output file, or to reorder bands.

-outsize *xsize[%] ysize[%]:*

Set the size of the output file. Outsize is in pixels and lines unless " is attached in which case it is as a fraction of the input image size.

-scale *[src_min src_max [dst_min dst_max]]:*

Rescale the input pixels values from the range *src_min* to *src_max* to the range *dst_min* to *dst_max*. If omitted the output range is 0 to 255. If omitted the input range is automatically computed from the source data.

-srewin *xoff yoff xsize ysize:*

Selects a subwindow from the source image for copying based on pixel/line location.

-projwin *ulx uly lrx lry:*

Selects a subwindow from the source image for copying (like **-srewin**) but with the corners given in georeferenced coordinates.

-a_srs *srs_def:*

Override the projection for the output file. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

-a_ullr *ulx uly lrx lry:*

Assign/override the georeferenced bounds of the output file. This assigns georeferenced bounds to the output file, ignoring what would have been derived from the source file.

-a_nodata *value:*

Assign a specified nodata value to output bands.

-mo *"META-TAG=VALUE":*

- Passes a metadata key and value to set on the output dataset if possible.
- co "NAME=VALUE":**
Passes a creation option to the output format driver. Multiple **-co** options may be listed. See format specific documentation for legal creation options for each format.
- gcp pixel line easting northing:**
Add the indicated ground control point to the output dataset. This option may be provided multiple times to provide a set of GCPs.
- quiet:**
Suppress progress monitor and other non-error output.
- sds:**
Copy all subdatasets of this file to individual output files. Use with formats like HDF or OGDI that have subdatasets.
- src_dataset:**
The source dataset name. It can be either file name, URL of data source or subdataset name for multi-dataset files.
- dst_dataset:**
The destination file name.

Example1: Convert ECW raster to a GeoTIFF

```
> gdal_translate -of GTiff image1.ecw image1.tif
```

3.4.8 gdalwarp

Description:

The `gdalwarp` utility is an image mosaicing, reprojection and warping utility. The program can reproject to any supported projection, and can also apply GCPs stored with the image if the image is "raw" with control information.

- s_srs srs def:**
source spatial reference set. The coordinate systems that can be passed are anything supported by the `OGRSpatialReference.SetFromUserInput()` call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a `.prf` file containing well known text.
- t_srs srs def:**
target spatial reference set. The coordinate systems that can be passed are anything supported by the `OGRSpatialReference.SetFromUserInput()` call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a `.prf` file containing well known text.
- order n:**
order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.
- tps**
Enable use of thin plate spline transformer based on available GCPs. Use this *instead* of the `-order` switch.
- et err_threshold:**
error threshold for transformation approximation (in pixel units - defaults to 0.125).
- te xmin ymin xmax ymax:**
set georeferenced extents of output file to be created.
- tr xres yres:**
set output file resolution (in target georeferenced units)
- ts width height:**
set output file size in pixels and lines
- wo "NAME=VALUE":**

Set a warp options. The [GDALWarpOptions::papszWarpOptions](#) docs show all options. Multiple **-wo** options may be listed.

-ot type:

For the output bands to be of the indicated data type.

-wt type:

Working pixel data type. The data type of pixels in the source image and destination image buffers.

-r resampling_method:

Resampling method to use. Available methods are:

near:

nearest neighbour resampling (default, fastest algorithm, worst interpolation quality).

bilinear:

bilinear resampling.

cubic:

cubic resampling.

cubicspline:

cubic spline resampling.

lanczos:

Lanczos windowed sinc resampling.

-srcnodata value [value...]:

Set nodata masking values for input bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. Masked values will not be used in interpolation. Use a value of `None` to ignore intrinsic nodata settings on the source dataset.

-dstnodata value [value...]:

Set nodata values for output bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. New files will be initialized to this value and if possible the nodata value will be recorded in the output file.

-dstalpha:

Create an output alpha band to identify nodata (unset/transparent) pixels.

-wm memory_in_mb:

Set the amount of memory (in megabytes) that the warp API is allowed to use for caching.

-multi:

Use multithreaded warping implementation. Multiple threads will be used to process chunks of image and perform input/output operation simultaneously.

-q:

Be quiet.

-of format:

Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-co "NAME=VALUE":

passes a creation option to the output format driver. Multiple **-co** options may be listed. See format specific documentation for legal creation options for each format.

srcfile:

The source file name(s).

dstfile:

The destination file name.

Example: Reproject GeoTIFF in EPSG:42304 to EPSG:4326

```
>gdalwarp -s_srs EPSG:42304 -t_srs EPSG:4326 image1.tif image1-ll.tif
```

3.5 Projections

In the simplest sense, projections allow you to represent the round earth on a flat surface, such as your computer monitor. For MapServer, the two important terms you should get to know are PROJ.4 and EPSG, which are both described in the following sections.

3.5.1 PROJ.4

PROJ.4 is the name of the library that MapServer uses for projection transformations. This library is used by many other projects including GDAL/OGR, GRASS, and MapGuide Open Source. This library was developed by Gerald Evenden (then of USGS), and is co-maintained by Frank Warmerdam.

PROJ.4 uses the following syntax to define a projection:

```
+lon_0=<angle> Central Meridian, Longitude of Origin, Center Longitude
+lat_0=<angle> Latitude of Origin, Center Latitude
+k=<scale_factor>
+x_0=<>false_easting>
+y_0=<>false_northing>
+datum=<datum>
```

An example of a transverse mercator projection defined with PROJ.4 parameters is:

```
+proj=tmerc +lon_0=-117 +lat_0=0 +k=0.9996 +x_0=500000 +y_0=0 +datum=WGS84
```

3.5.2 EPSG Codes

MapServer uses the EPSG standard (European Petroleum Survey Group) for storing PROJ.4 projection parameters. Typically an 'epsg' lookup file would contain hundreds of projections, each with its own number. For example here are a few epsg entries in an 'epsg' file, notice the lookup number between the <> symbols:

```
# Anguilla 1957 / British West Indies Grid
```

```
<2000> +proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=400000 +y_0=0 +ellps=clrk80 +units=m +no_defs no_defs
```

<>

Antigua 1943 / British West Indies Grid

```
<2001> +proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=400000 +y_0=0 +ellps=clrk80
+towgs84=-255,-15,71,0,0,0,0 +units=m +no_defs no_defs <>
```

Dominica 1945 / British West Indies Grid

```
<2002> +proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=400000 +y_0=0 +ellps=clrk80
+towgs84=725,685,536,0,0,0,0 +units=m +no_defs no_defs <>
```

3.5.3 MapServer and Projections

MapServer does have the ability to reproject both vector and raster data on-the-fly. A projection can be defined in MapServer in a PROJECTION object by using the full PROJ.4 definition of a projection, or by using the projection's associated EPSG lookup code:

- Using the full PROJ.4 definition:

```
PROJECTION
  "+proj=longlat +ellps=WGS84 +datum=WGS84"
END
```

- Using the associated EPSG code:

```
PROJECTION
  "init=epsg:4326"
END
```

Important rules:

- To have MapServer reproject you must define at least two projection objects: one for the output image (in the MAP object) and one for each layer (in the LAYER objects) to be projected.
- The MAP's EXTENT and UNITS values must always be associated with the output projection (the PROJECTION at the upper MAP level).
- If you specify a MAP-level projection, and then only one other LAYER projection object, MapServer will assume that all of the other layers are in the specified MAP-level projection.
- If all of your data in the mapfile is in the same projection, you DO NOT have to specify any projection objects. MapServer will assume that all of the data is in the same projection.
- The use of the EPSG lookup file has a small performance hit (the time for MapServer to read an external file), so defining the projection with PROJ.4 parameters in the mapfile is optimal for performance.

- Reprojections have a performance hit (anything extra like this takes time) so this should be considered when you are preparing your data beforehand.

Example1: One layer has different projection than the output MAP extents

The key here is that the only layer that requires a PROJECTION object is the layer that is in a different projection:

```
MAP
EXTENT -180 -90 180 90
UNITS DD
...
PROJECTION
  "init=4326"
END

LAYER
...
END

# layer in different projection
LAYER
...
PROJECTION
  "init=42304"
END
...
END

LAYER
...
END

END #mapfile
```

Example2: All layers are in the same projection

No projection objects are required:

```
MAP
EXTENT -180 -90 180 90
UNITS DD
...
```



LAYER

...

END

LAYER

...

END

LAYER

...

END

END #mapfile

3.6 Tips and Tricks

MapServer has a lot of hidden gems, that sometimes might not be documented, or can only be learned through years of use, so the goal of this section is to pass on this knowledge.

3.6.1 Rules for Preparing Data for MapServer

There are some general rules when preparing your data for display over the Internet through MapServer:

1. The fastest formats for display in MapServer is ESRI Shapefile for vector and GeoTIFF for raster. The ogr2ogr (vector) and gdal_translate (raster) utilities can be used to convert your existing data to a faster format.
2. If you are using shapefiles make sure they have quadtree indexes created for each of them (*.qix) with the shptree utility.

e.g. Execute shptree on all shapefiles in the current folder and sub-folders:
for /R %i in (*.shp) do shptree %i

3. For vector files, remove any unnecessary fields/columns with the ogr2ogr utility.
4. Duplicate spatial files covering multiple areas (such as roads.shp that exists in every county folder) should use tileindexes. The tile4ms utility (if they are shapefiles), ogrtindex utility (vectors), or gdaltindex utility (rasters) can be used to create this index file. This index file is then referred to in the mapfile by using the TILEINDEX parameter in the LAYER. (for a thorough example see the 'tile4ms' section of this document)
5. Create overviews for all raster files with the gdaladdo utility, which will downsample the raster

image for the user when zoomed out in a MapServer application. MapServer is smart enough to just use the overviews in the rasters.

6. On-the-fly reprojections should be avoided, so make sure all of your data is in the same projection. The `ogr2ogr` (vector) and `gdalwarp` (raster) utilities can be used to reproject your data.
7. If you have a large single shapefile, you might want to split the file down to smaller pieces, so a user does not have to load the entire shapefile for an area outside the one they are currently viewing. The utility `shp2tile` can be used (which is part of the OSGeo4W package). For example, the following command will split the `drainage.shp` file into several other smaller shapefiles in the `/drainage` folder :

```
shp2tile --quadtree 1000 --square-ext drainage.shp ./out/drainage
```

8. You might also want to create generalized files for vector polygons and linework that have many nodes and vertices. The idea here is that a simplified file would be displayed when the user is zoomed out in a MapServer application, and then the raw file would be displayed when zoomed in.
9. Be careful with using spaces, special characters, or uppercase in filenames and folders containing data. A good rule of thumb is to avoid any issues with those (with MapServer, its underlying libraries, utilities, or different Operating Systems) and not use any spaces, special characters or uppercase at all in files or folders.
10. Use the `INCLUDE` parameter in the mapfile to store each of your layers. It's just easier to manage large mapfiles.

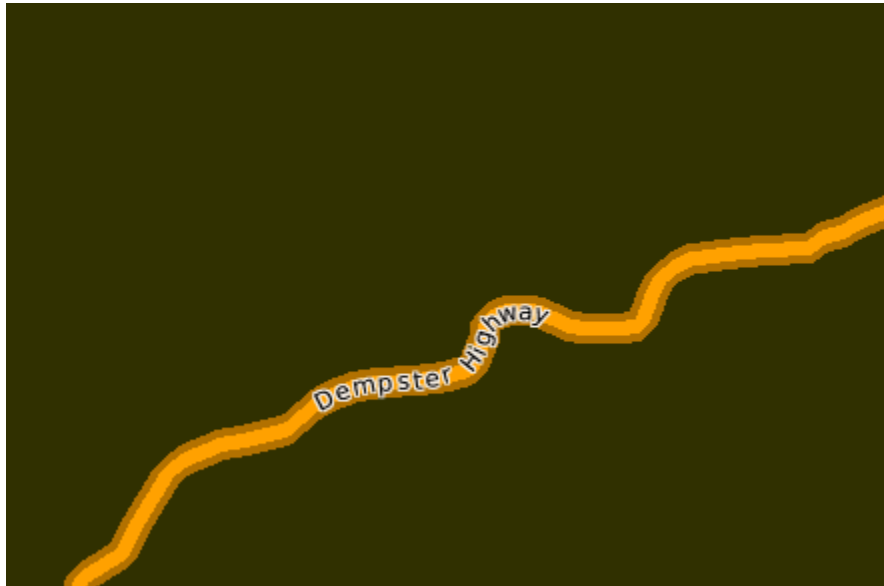
3.6.2 Labelling

Spline Labelling

Added to MapServer 4.10, labels can be configured to follow features, such as a river. The important mapfile parameter to use is the LABEL object's `ANGLE FOLLOW` parameter. The following example shows how to use this parameter to label a curved road feature:

```
CLASS
...
LABELITEM "NAME"
LABEL
  FONT "sans"
  TYPE TRUETYPE
  POSITION CC
  ANGLE FOLLOW
  SIZE 9
```

```
COLOR 0 0 0
OUTLINECOLOR 242 236 230
PARTIALS FALSE
END # label
END #class
```



Setting the Label's Drawing Priority

Added to MapServer 5.0, you can set the drawing priority for a layer using the LABEL object's PRIORITY parameter. The priority can be set from 1 (lowest) to 10 (highest), and the default is 1. The following examples shows how to use label priority for two line layers.

Example1: 2 layers with LABEL objects

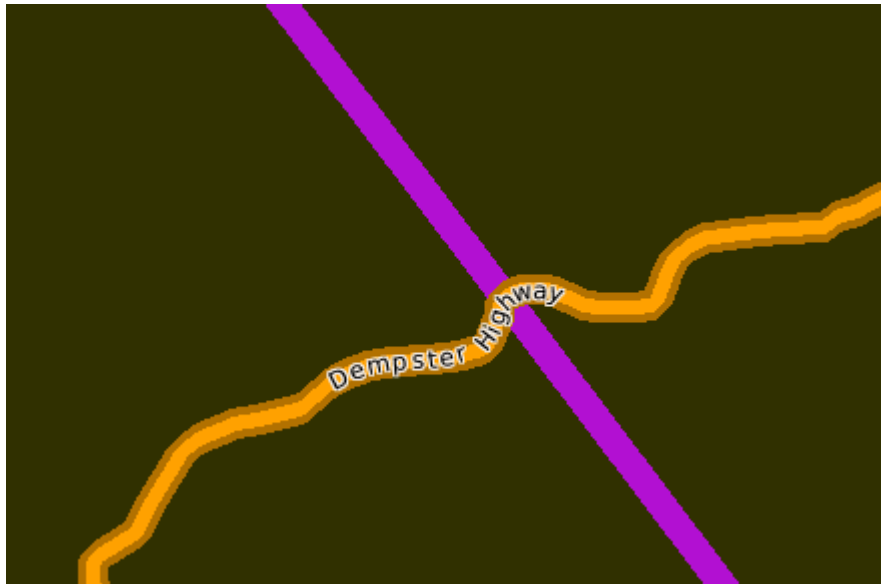
```
...
LAYER
  NAME "transmission lines"
  ...
  LABEL
  ...
  END
END
```

```
LAYER
  NAME "highways"
```

```
LABEL
...
END
END
```

Result:

Since the highways layer is drawn last (lower down in mapfile) its label will win over the transmission label, as shown in this image (where the purple line is the transmission line layer):



Example1: 2 layers with LABEL objects using PRIORITY

```
...
LAYER
  NAME "transmission lines"
  ...
  LABEL
  ...
  PRIORITY 10
  END
END
```

```
LAYER
  NAME "highways"
  LABEL
  ...
```



```
PRIORITY 9  
END  
END
```


Result:

Since the transmission layer's label has a higher priority set, its label will win over the highway's label, as shown in the image below:



3.6.3 Displaying Highway Shields

There are several tricks to assist you in adding a labelled highway shield to your road feature:

1. use a separate layer for highway shields
2. set the layer type to ANNOTATION
3. use a blank shield SYMBOL in the layer's CLASS, such as:

4. make sure your data contains a field containing highway numbers (if your data does not have highway numbers, you may have to do some pre-processing to extract the highways numbers from the name attribute)
5. use a LABEL object with the LABELITEM set to the field containing the highway numbers

Here is an example of a highway shield layer:

```
LAYER
```

```
NAME "shields"  
TYPE ANNOTATION  
STATUS DEFAULT  
DATA "roads"  
LABELITEM "ID"  
CLASS  
  NAME "Expressways"  
  SYMBOL 'Highway'  
  COLOR 0 0 0  
  # LABEL object  
  LABEL  
    POSITION CC  
    SIZE TINY  
    MINDISTANCE 300  
    MINFEATURESIZE 5  
    COLOR 0 0 0  
    PARTIALS FALSE  
    BUFFER 10  
  END #label  
END #class  
END #layer
```

Here is the resulting image:



3.6.4 Debugging in MapServer

Error File

MapServer can output errors to a text file. Some applications hide errors, so this error file can be useful. Here is an example to create an error file in a MapServer 5.0 mapfile:

```
MAP
...
CONFIG "MS_ERRORFILE" "/ms4w/tmp/ms_error.txt"
DEBUG 2
...
END
```

A file named 'ms_error.txt' will be created at that location when an error is thrown.

You can also enable this error file on your system by setting the environment variable named "MS_ERRORFILE" with a path and filename (e.g. you can set environment variables in Apache's httpd.conf file using SetEnv, such as:

```
SetEnv MS_ERRORFILE "/ms4w/tmp/ms_error.txt"
```

Debug Levels

Since MapServer 5.0, you can set the debugging level to get more details from MapServer:

- Level 0 – Errors only
- Level 1 – Errors and notices
- Level 2 – Map Tuning (includes draw speeds)
- Level 3 – Verbose (includes WMS URLs)
- Level 4 – Very verbose

The shp2img utility has a '-map_debug' switch to set the debug level at the commandline:

```
shp2img -m gmap75.map -o test.png -map_debug 2
```

```
msDrawRasterLayerLow(bathymetry): entering.
msDrawMap(): Layer 0 (bathymetry), 0.047s
```

```
msDrawMap(): Layer 1 (land_fn), 0.032s
msDrawMap(): Layer 3 (drain_fn), 0.000s
msDrawMap(): Layer 4 (drainage), 0.046s
msDrawMap(): Layer 5 (prov_bound), 0.016s
msDrawMap(): Layer 6 (fedlimit), 0.000s
msDrawMap(): Layer 9 (popplace), 0.016s
msDrawMap(): Drawing Label Cache, 0.156s
msDrawMap() total time: 0.313s
```

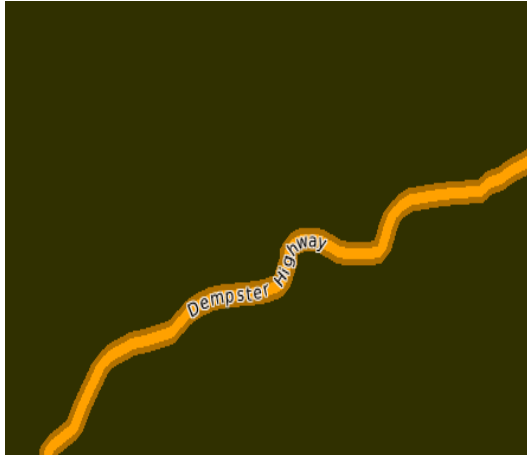
You can also set the Debug level on your system by setting the environment variable named “MS_DEBUGLEVEL” with a value.

3.6.5 Smooth Line Output (Antialiasing)

New in MapServer 5.0 is a high quality rendering engine, named Anti Grain Geometry (AGG). AGG produces smoother lines, making the map image appear much cleaner. However AGG is not enabled by default, and must be configured in the mapfile with the IMAGETYPE and OUTPUTFORMAT parameters, as follows:

```
MAP
...
IMAGETYPE “AGG”
OUTPUTFORMAT
  NAME “AGG”
  DRIVER AGG/PNG
  IMAGEMODE RGB
END
...
LAYER
...
END #layer
END #mapfile
```

default



AGG



(notice the smoother outer lines in the AGG image above)

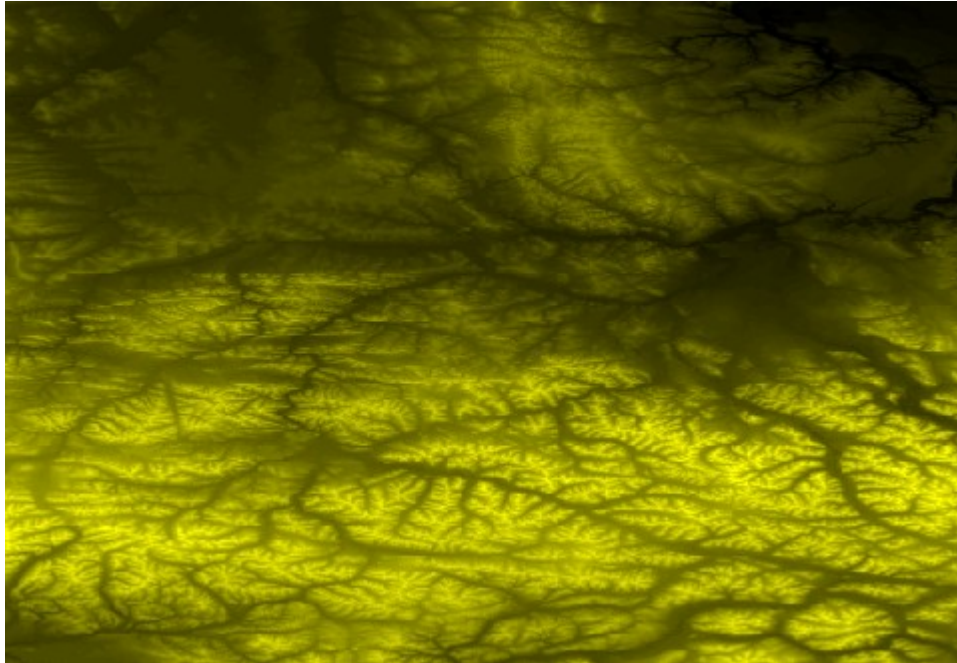
3.6.6 Color Ramps

Added to MapServer 4.6, you can apply a color ramp to an Integer field using a STYLE object and the COLORRANGE, DATARANGE, and RANGEITEM parameters in a mapfile. For example, here is a layer containing a DEM (in ASCII grid format) with a color range set from black to yellow:

```
LAYER
NAME 'dem'
TYPE RASTER
DATA "can3d30"
STATUS DEFAULT
PROJECTION
  "init=epsg:4326"
END
# style the data
CLASS
NAME "DEM"
STYLE
  COLORRANGE 0 0 0 255 255 0 # black to yellow
  DATARANGE 0.0 2000.0
  RANGEITEM "[pixel]"
END
```

END

END



3.6.7 Charting

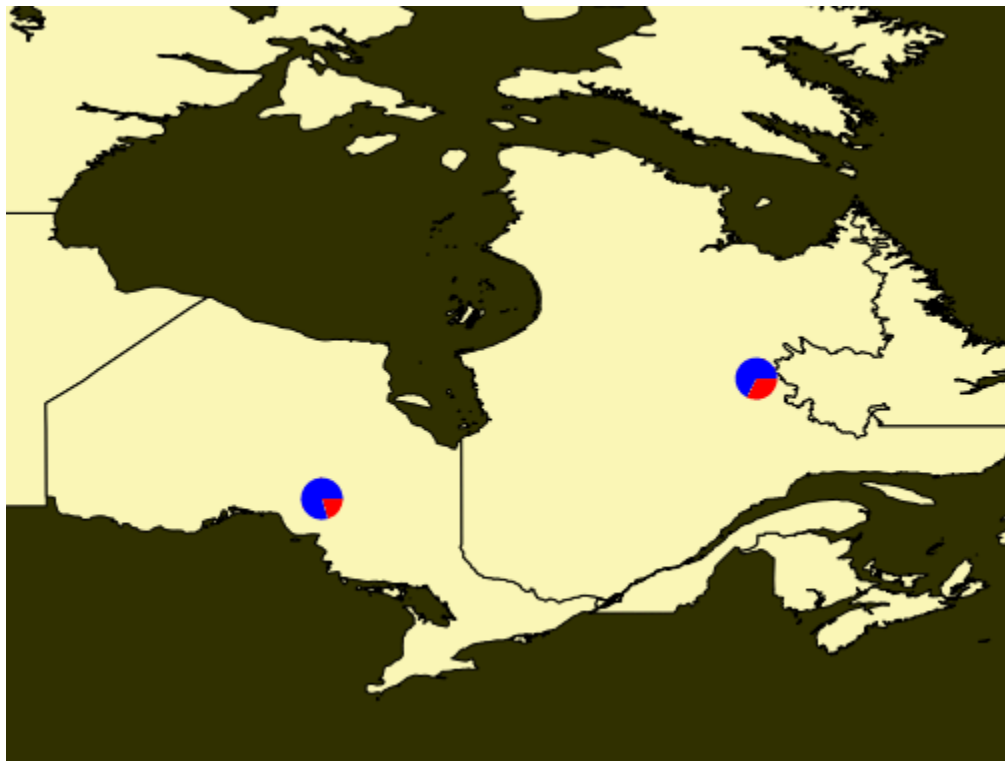
Added in MapServer 5.0, you can create pie or bar graphs on your map. The type and size of the chart is set with a PROCESSING parameter in the mapfile LAYER. Each slice of the chart is a separate CLASS, and has it's own styling. The SIZE parameter in the STYLE object is the value to be displayed in the chart. The following example shows how to create 2 pie charts for Ontario and Quebec, based on population age ranges.

```
LAYER
  NAME "ontario chart"
  TYPE CHART
  PROCESSING "CHART_SIZE=21" # specify size of the chart for pie or bar graphs
  PROCESSING "CHART_TYPE=PIE"
  DATA ./canada/ontario
  STATUS DEFAULT
  CLASS
    NAME "0-14"
    STYLE
      SIZE 1145
      COLOR 255 0 0
    END
  END
END
```

```
CLASS
  NAME "15-64"
  STYLE
    SIZE 4430
    COLOR 0 0 255
  END
END
END #layer
```

LAYER

```
NAME "quebec chart"
TYPE CHART
PROCESSING "CHART_SIZE=21"
PROCESSING "CHART_TYPE=PIE"
DATA ./canada/quebec
STATUS DEFAULT
CLASS
  NAME "0-14"
  STYLE
    SIZE 1300
    COLOR 255 0 0
  END
END
CLASS
  NAME "15-64"
  STYLE
    SIZE 2701
    COLOR 0 0 255
  END
END
END #layer
```



3.7 Factors Affecting Performance

The following is a list of factors that should be kept in mind when displaying spatial data in MapServer:

Format of the data

- some formats are easier to display than others (TIGER is an text/ASCII format that is slow to parse)
- wherever possible use Shapefiles for vectors and GeoTIFF for rasters

Is the data indexed?

- MapServer looks for .qix files when using ESRI Shapefiles (created by the shptree utility)
- Use tile4ms or gdaltindex utilities to create an index shapefile for similar data files

Labelling

- labelling features is considered very 'expensive' in terms of time required to create the map

Size of the data file

- fields/columns not required should be removed from the data if not needed
- large files can also be split with the shp2tile utility

Number of layers to draw

- obviously large mapfiles take longer for map image creation in MapServer, as MapServer must read every line of the mapfile

Mapfile Expressions

- expressions in classes slow down map image creation time
- data should be pre-processed to remove need for expressions wherever possible
- logical expressions (AND/OR) are the slowest and should be avoided if possible

Parsing large symbolset and font lookup files

- you should remove any references to symbols and fonts that you do not need

Projections

- MapServer can reproject data, but for speed of display, you should make sure that all of your data is in the same projection whenever possible
- defining the projection inside the mapfile with the PROJ.4 parameters is faster than referring to an EPSG lookup file

How you are executing MapServer (CGI, MapScript)

- CGI (and especially FastCGI) is fast and lightweight

Server specifications of the MapServer machine

- MapServer is very lightweight, however a faster machine can never hurt
- Linux operating system is considered the best choice for MapServer hosting

4 Building a MapServer Application

4.1 Flavors

4.2 Client Software

5 Community

5.1 History of MapServer

MapServer has come a long way in a very short time, from Steve Lime's thesis project in the '90's, to being one of the most active Open Source projects in the world. The following is a brief timeline of its history.

5.1.1 The Past

- 1994 - C Program Creates ArcPlot AMLs
- 1994 - NASA-sponsored ForNet Project
- 1997 - Shapelib Library (LGPL or “MIT-type”)
- 1997 - NASA-sponsored TerraSIP Project
- 1998 - MapServer 3.2 Released as Open Source
- 2001 - MapServer 3.4
- 2003 - MapServer 4.0
- 2004 - MapServer 4.2
- 2006 - MapServer 4.8
- 2006 - MapServer 4.10

5.1.2 The Present

- Version 5.0 Released September, 2007
- 20+ Active Developers Around The World
- 2,100+ Subscribers to MapServer-Users List
- 40,000+ Global Applications Deployed

- Community members active in OSGeo foundation (started Feb 2006)
- Participated in Free and Open Source Software for Geospatial Conference (FOSS4G) in Victoria BC, Canada, Sept 2007

5.1.3 The Future

- MapServer 5.x
- WFS 1.1.0 support
- WMS 1.3 support
- Improved documentation
- OSGeo4W installer
- active in Open Source Geospatial Foundation (osgeo.org)
- participate in FOSS4G2008 conference in Cape Town, South Africa (foss4g2008.org)

Release plans for any upcoming version can be found at:
http://mapserver.gis.umn.edu/development/release_plans/

5.2 Project Steering Committee

The MapServer Project Steering Committee (PSC) essentially makes decisions on all issues (technical and non-technical) regarding the development of MapServer. Before a significant feature is added to the MapServer core, several steps must occur: 1) the person proposing the feature must document the changes in a document, named a Request for Comments (RFC), 2) the PSC must vote on the RFC, and 3) if the changes are accepted, they will be added to the next release.

Here are the MapServer PSC members:

Yewondwossen Assefa

Howard Butler

Tom Kralidis

Steve Lime

Perry Nationales

Umberto Nicoletti

Jeff McKenna

Daniel Morissette



Tamas Szekeres
Frank Warmerdam
Steve Woodbridge

5.3 Contributing

MapServer and all Open Source projects depend on the community to keep it alive and thriving. There are many areas where you can help, and here are a few:

- Helping users via the mailing lists and IRC (Internet Relay Chat)
- Documentation - all documentation can be improved in some way: how-to's, code snippets, tutorials & demos
- Need to update the standard demo to support basic CGI use, but to show other technologies such as OpenLayers, MapBuilder, dBox, PostGIS and so forth
- Programming, we can always use developers experienced with C/C++, Swig, PHP and other languages
- Test development using Python, C/C++ or MapServer configuration files
- Need help building a suite of mapfiles to cover all possible symbolization configurations (e.g. polygon:vector:unfilled, polygon:pixmap:transparency, polygon:pixmap:24-bit)
- Need help developing a set of super-templates that cover all possible substitutions
- Need help developing a set of test data that goes beyond just points, lines and polygons. For example, need z and m shapefiles, ready-to-load versions for PostGIS, attributes sets covering a broad spectrum of possibilities and uses for labeling and filtering...
- Build environment development using configure, makefiles and so on
- Binary distribution development for all platforms
- Technical website maintenance
- Website design
- MapServer logo design
- Fund development of features important to you or your organization
- Translation of documentation to other languages

If any of those items interest you, start by introducing yourself on the 'MapServer-dev' mailing list (discussed in the next section) and mentioning your possible volunteer effort there.

5.4 Getting Help

Thanks to MapServer's thriving community and the efforts of many volunteers, getting help is possible through several methods.

5.4.1 Internet Relay Chat (IRC)

Some of the development of MapServer is coordinated through IRC. This section describes how you log on to chat, ask questions, and hack around with the developers.

Server and Channel Information

Server: irc.freenode.net

Channel: #mapserver

Why IRC?

IRC is a primary medium where Open Source GIS hackers congregate, collaborate, and hack. It makes it easy to communicate things like compilation issues, where immediate, iterative feedback allows folks to make a lot of progress. Something that might take days of heavily-quoted emails through a mailinglist might only take fifteen minutes on IRC.

IRC is a great way to coordinate on-line meetings. Much of the discussion about the development of the new MapServer website was coordinated through IRC.

Some folks find IRC distracting and do not normally participate except for on-line meetings, but it is up to you.

How Do I Join?

Chatzilla (<http://www.mozilla.org/projects/rt-messaging/chatzilla/>) is probably the easiest way to get going. Chatzilla works with Mozilla or Firefox, and once you have it installed, you can log on to the channel by pointing your browser at: irc://irc.freenode.net/#mapserver

5.4.2 Mailing Lists

There are several mailing lists available for you to use. You can either have the messages sent to you as they are submitted, or you can have the messages sent to you once a day in a 'digest'. You can also search past messages to see if your question has already been answered (this is highly recommended).

mapserver-users

Description:

The mapserver-users listserv was created as a means for MapServer users and developers to exchange application ideas, discuss potential software improvements, and ask questions. Since there are no funds to provide direct user support, we hope that the MapServer user community will use the list to support themselves and help new users get started building MapServer applications.

Join:

<http://lists.osgeo.org/mailman/listinfo/mapserver-users>

Search Archives (after 2/29/2008):

<http://www.nabble.com/Mapserver---User-f31321.html>

Search Archives (before 2/29/2008):

<http://www.nabble.com/Mapserver---User-%28old%29-f1215.html>

mapserver-dev

Description:

A separate listserv is available for MapServer developers. It is meant to be used by individuals working on MapServer source code and related libraries to discuss issues that would not be of interest to the entire mapserver-users listserv.

Join:

<http://lists.osgeo.org/mailman/listinfo/mapserver-dev>

Search Archives (after 2/29/2008):

<http://www.nabble.com/Mapserver---Dev-f31322.html>

Search Archives (before 2/29/2008):

<http://www.nabble.com/Mapserver---Dev-%28old%29-f1216.html>

mapserver-announce

Description:

The mapserver-announce listserv is used to announce MapServer software updates or security issues. It is a very low-traffic volume list.

Join:

<http://lists.osgeo.org/mailman/listinfo/mapserver-announce>

5.4.3 Conferences/Workshops

Attending conferences and workshops are an excellent way to learn and grow in the MapServer community. The FOSS4G (Free and Open Source Software for Geospatial) conference is an annual event where the MapServer developers and community members get together to exchange ideas and share with others. Past conferences post workshop materials as well, so you can download the session's data and powerpoints to get started. For a list of future and past conferences and their sites see: <http://mapserver.gis.umn.edu/community/conferences>

5.4.4 OSGeo Foundation

Formed in February 2006, the Open Source Geospatial Foundation, or OSGeo, is a not-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. It also serves as an independent legal entity to which community members can contribute code, funding and other resources, secure in the knowledge that their contributions will be maintained for public benefit. OSGeo also serves as an outreach and advocacy organization for the open source geospatial community, and provides a common forum and shared infrastructure for improving cross-project collaboration.

MapServer is an official OSGeo project, and uses the OSGeo infrastructure for things like mailing lists, servers, etc. MapServer community members are also very involved in the OSGeo community.

Service Providers

The OSGeo website allows users to search by country for individuals and organizations who offer support services for many Open Source Geospatial projects. The search page is located at:

http://www.osgeo.org/search_profile

The providers offer expertise in the following areas:

Chameleon, deegree, FDO, GDAL, GeoNetwork, GeoServer, GeoTools, GRASS, gvSIG, JUMP/OpenJUMP, Ka-map, Mapbender, MapBuilder, MapguideOS, MapServer, MapWindow, OpenLayers, OSSIM, PostGIS, PROJ, QuantumGIS.

Local Chapters

OSGeo local chapters provide a venue to support local users and developers, as well as a mechanism to further OSGeo's mission and goals in a specific linguistic or geographic area. In other words they are a great way to meet others in the Open Source Geospatial community.

Active chapters include the following:

- British Columbia, Canada OSGeo Chapter
- China OSGeo Chapter
- French language OSGeo Chapter
- India OSGeo Chapter
- Italian language OSGeo Chapter
- Japan OSGeo Chapter
- New Mexico, USA OSGeo Chapter
- Ottawa OSGeo Chapter
- Twin Cities (MN) OSGeo Chapter

OSGeo Chapters "In Formation"

- Australia OSGeo Chapter
- Brazilian OSGeo Chapter



- German Language OSGeo Chapter
- Greek Language OSGeo Chapter
- Korean Language OSGeo Chapter
- Taiwan OSGeo Chapter

Sponsor OSGeo

You can also choose to give back to the community by sponsoring the OSGeo Foundation, one of its individual projects, or the annual FOSS4G conference.

6 Important Links

- MapServer Applications Gallery: <http://mapserver.gis.umn.edu/gallery>
- MapServer Documentation: <http://mapserver.gis.umn.edu/docs>
- MapServer Download: <http://mapserver.gis.umn.edu/download/current/>
- MapServer Home: <http://mapserver.gis.umn.edu>
- OSGeo Home: <http://www.osgeo.org/>
- Search OSGeo Service Providers: http://www.osgeo.org/search_profile

7 Glossary

- AGG - Anti-Grain Graphics Library. This is the additional rendering engine available in MapServer 5.0
- EPSG - European Petroleum Survey Group, who maintain a geodetic parameter set. MapServer uses EPSG's lookup codes to access specific projection definitions.
- GD - Graphic library, which MapServer uses by default.
- GDAL - Raster data translator library, used by MapServer to access raster data formats.
- OGC - Open Geospatial Consortium. A group of organizations that develop and support geospatial standards.
- OGR - Vector data translator library, used by MapServer to access vector data formats.
- Open Source - in the simple sense, this means that the software is freely available and the source code is also available. A more proper definition is software that follows these rules: 1) license does not restrict any party from selling or giving away the software, 2) the source code must be distributed, and 3) license must allow modifications of source code.
- Projection - a method to represent areas of the Earth on a flat surface, such as a paper map or a computer monitor.
- SRS - Spatial Reference System. OGC standards often refer to an SRS, which is commonly specified as an EPSG code.
- WCS - Web Coverage Service. This is an OGC standard to share raster data over the Internet.
- WFS - Web Feature Service. This is an OGC standard to share vector data over the Internet.
- WMS - Web Map Service. This is an OGC standard that shares images of data over the Internet.

8 Credits

The entire MapServer community must be thanked as sources for this document. Here are some specific authors whose knowledge were used in this manual:

Author	Source
Perry Nacionales	MapServer Tutorial
Tyler Mitchell	Vector Data Access Guide
Frank Warmerdam	Raster Data Access Guide
Bob Basques	Mapfile Example
David Fawcett	New Users Document
Steve Lime	Mapfile Reference
Steven Woodbridge	Content Recommendations

9 Appendix

9.1 Mapfile Reference

9.1.1 CLASS Object

Defines thematic classes for a given layer and each layer must have at least one class. In cases with more than one class, membership is determined using attribute values and expressions. Starts with the keyword CLASS and terminates with the keyword END.

BACKGROUND_COLOR [r] [g] [b]

Color to use for non-transparent symbols.

COLOR [r] [g] [b]

Color to use for drawing features.

DEBUG [on|off]

Enables debugging of the class object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServer logfile if one is set using the LOG parameter in the [WEB](#) object.

EXPRESSION [string]

Four types of expressions are now supported to define class membership. String comparisons, regular expressions, simple logical expressions, and string functions. If no expression is given, then all features are said to belong to this class.

- String comparisons are case sensitive and are the fastest to evaluate. No special delimiters are necessary although string must be quoted if they contain special characters. (As a matter of good habit, it is recommended you quote all strings).
- Regular expressions function just like previous versions of MapServer. However, you must now delimit a regular expression using /regex/. No quotes should be used.
- Logical expressions allow you to build fairly complex tests based on one or more attributes and therefore are only available with shapefiles. Logical expressions are delimited by parentheses "(expression)". Attribute names are delimited by square brackets "[ATTRIBUTE]". These names are case sensitive and must match the items in the shapefile. For example: EXPRESSION ([POPULATION] > 50000 AND '[LANGUAGE]' eq 'FRENCH') ... The following logical operators are supported: =, >, <, <=, >=, or, and, lt, gt, ge, le, eq, ne. As you might expect this level of complexity is slower to process.
- One string function exists: length(). This obviously computes the length of a string. An example follows:

```
EXPRESSION (length('[NAME_E]') < 8)
```

String comparisons and regular expressions work from the classitem defined at the layer level. You may mix expression types within the different classes of a layer.

JOIN

Signals the start of a [JOIN](#) object.

KEYIMAGE [filename]

Full filename of the legend image for the CLASS. This image is used when building a legend (or requesting a legend icon via MapScript or the CGI application).

LABEL

Signals the start of a [LABEL](#) object.

MAXSCALE [double]

Maximum scale at which this CLASS is drawn.

MAXSIZE [integer]

Maximum size in pixels to draw a symbol. Default is 50.

MINSCALE [double]

Minimum scale at which this CLASS is drawn.

MINSIZE [integer]

Minimum size in pixels to draw a symbol. Default is 0.

NAME [string]

Name to use in legends for this class. If not set class won't show up in legend.

OUTLINECOLOR [r] [g] [b]

Color to use for outlining polygons and certain marker symbols. Line symbols do not support outline colors.

SIZE [integer]

Height, in pixels, of the symbol/pattern to be used. Only useful with scalable symbols. For vector (and ellipse) symbol types the default size is based on the range of Y values in the POINTS defining the symbol. For pixmaps, the default is the vertical size of the image. Default size is 1 for TTF symbols.

STYLE

Signals the start of a [STYLE](#) object. A class can contain multiple styles.

SYMBOL [integer|string|filename]

The symbol name or number to use for all features if attribute tables are not used. The number is the index of the symbol in the symbol file, starting at 1, the 5th symbol in the file is therefore symbol number 5. You can also give your symbols names using the NAME keyword in the symbol definition file, and use those to refer to them. Default is 0, which results in a single pixel, single width line, or solid polygon fill, depending on layer type.

You can also specify a gif or png filename. The path is relative to the location of the mapfile.

TEMPLATE [filename]

Template file or URL to use in presenting query results to the user.

TEXT [string]

Static text to label features in this class with. This overrides values obtained from the LABELTIEM. The string may be given as an expression delimited using the ()'s. This allows you to concatenate multiple attributes into a single label. For example: ([FIRSTNAME],[LASTNAME]).

You can also "stack" 2 symbols to achieve interesting effects. You define the second symbol, which effectively sits "on top" of the symbol normally defined above.

The following parameters allow you to define the symbol, and they are equivalent to their non-overlay counterparts:

- OVERLAYBACKGROUND
- OVERLAYCOLOR
- OVERLAYOUTLINECOLOR
- OVERLAYSIZE
- OVERLAYMINSIZE
- OVERLAYMAXSIZE
- OVERLAYSYMBOL

9.1.2 FEATURE Object

Defines inline features. You can use inline features when it's not possible (or too much trouble) to create a shapefile. Inline features can also be built via urls or forms. Starts with the keyword FEATURE and terminates with the keyword END.

POINTS

A set of xy pairs terminated with an END, for example:

```
POINTS 1 1 50 50 1 50 1 1 END
```

Note that with POLYGON/POLYLINE layers POINTS must start and end with the same point (i.e. close the feature).

TEXT [string]

String to use for labeling this feature.

WKT [string]

A geometry expressed in OpenGIS Well Known Text geometry format. This feature is only supported if MapServer is built with OGR or GEOS support.

```
WKT "POLYGON((500 500, 3500 500, 3500 2500, 500 2500, 500 500))"
WKT "POINT(2000 2500)"
```

9.1.3 GRID Object

The GRID object defines a map graticule as a LAYER. Starts with the keyword GRID and terminates with the keyword END.

LABELFORMAT [DD|DDMM|DDMMSS|C format string]

Format of the label. "DD" for degrees, "DDMM" for degrees minutes, and "DDMMSS" for degrees, minutes, seconds. A C-style formatting string is also allowed, such as "%g°" to show decimal degrees with a degree symbol.

The default is decimal display of whatever SRS you're rendering the GRID with.

MINARCS [double]

The minimum number of arcs to draw. Increase this parameter to get more lines. Optional.

MAXARCS [double]

The maximum number of arcs to draw. Decrease this parameter to get fewer lines. Optional.

MININTERVAL [double]

The minimum number of intervals to try to use. The distance between the grid lines, in the units of the grid's coordinate system. Optional.

MAXINTERVAL [double]

The maximum number of intervals to try to use. The distance between the grid lines, in the units of the grid's coordinate system. Optional.

MINSUBDIVIDE [double]

The minimum number of segments to use when rendering an arc. If the lines should be very curved, use this to smooth the lines by adding more segments. Optional.

MAXSUBDIVIDE [double]

The maximum number of segments to use when rendering an arc. If the graticule should be very straight, use this to minimize the number of points for faster rendering. Optional, default 256.

The following is an example of a GRID object in use:

```
LAYER
NAME "grid"
METADATA
  "DESCRIPTION" "Grid"
END
TYPE LINE
STATUS ON
CLASS
```

```

NAME "Graticule"
COLOR 0 0 0
LABEL
  COLOR 255 0 0
  FONT fritgat
  TYPE truetype
  SIZE 8
  POSITION AUTO
  PARTIALS FALSE
  BUFFER 5
  OUTLINECOLOR 255 255 255
END
END
PROJECTION
  "init=epsg:4326"
END
GRID
  LABELFORMAT DDMM
# LABELFORMAT '%g°' # dec degrees with symbol
  MAXARCS 10
  MAXINTERVAL 10
  MAXSUBDIVIDE 2
# LABELFORMAT '%7.0f m' # nice if a projected SRS used
# MININTERVAL 20000
# MAXSUBDIVIDE 2
END
END # Layer

```

9.1.4 INCLUDE Object

Defines a file to be included in the mapfile parsing.

When this directive is encountered parsing switches to the included file immediately. As a result the included file can be comprised of any valid mapfile syntax. For example:

```
INCLUDE 'myLayer.map'
```

Performance does not seem to be seriously impacted with limited use, however in high performance instances you may want to use includes in a pre-processing step to build a production mapfile. The C pre-processor can also be used (albeit with a different syntax) and is far more powerful.

Notes:

- Supported in versions 4.10 and higher.
- The name of the file to be included **MUST be quoted** (single or double quotes).
- Includes may be nested, up to 5 deep.
- File location can be given as a full path to the file, or (in MapServer >= 4.10.1) as a path relative to the mapfile.
- Debugging can be problematic since: 1) the file an error occurs in does not get output to the user and 2) the line number counter is not reset for each file. Here is one possible error that is thrown when the include file cannot be found:

```
msyylex(): Unable to access file. Error opening included file "parks_include.map"
```


Example:

```
MAP
NAME 'include'
EXTENT 0 0 500 500
SIZE 250 250

INCLUDE "test_include_symbols.map"
INCLUDE "test_include_layer.map"
END
```

where test_include_symbols.map contains:

```
SYMBOL
NAME 'square'
TYPE VECTOR
FILLED TRUE
POINTS 0 0 0 1 1 1 1 0 0 0 END
END
```

and test_include_layer.map contains:

```
LAYER
TYPE POINT
STATUS DEFAULT
FEATURE
POINTS 10 10 40 20 300 300 400 10 10 400 END
END
CLASS
NAME 'Church'
COLOR 0 0 0
SYMBOL 'square'
SIZE 7
STYLE
SYMBOL "square"
SIZE 5
COLOR 255 255 255
END
STYLE
SYMBOL "square"
SIZE 3
COLOR 0 0 255
END
END
END
```

9.1.5 JOIN Object

Defines how a specific join is handled. Starts with the keyword JOIN and terminates with the keyword END.

Description

Joins are defined within a **LAYER** object. It is important to understand that JOINS are *ONLY* available once a query has been processed. You cannot use joins to affect the look of a map. The primary purpose is to enable lookup tables for coded data (e.g. 1 => Forest) but there are other possible uses.

Supported Formats:

- DBF/XBase files
- CSV (comma delimited text file)
- PostgreSQL and PostGIS tables
- MySQL tables

MapFile Parameters:

CONNECTION [string]

Parameters required for the join table's database connection (not required for DBF or CSV joins). The following is an example for PostgreSQL:

```
CONNECTION "host=127.0.0.1 port=5432 user=postgres password=postgres dbname=somename"
```

CONNECTIONTYPE [string]

Type of connection (not required for DBF or CSV joins). The following is an example for PostgreSQL:

```
CONNECTIONTYPE ogr
```

FROM [item]

Join item in the dataset. This is case sensitive.

NAME [string]

Unique name for this join. Required.

TABLE [filename|tablename]

For file-based joins this is the name of XBase or comma delimited file (relative to the location of the mapfile) to join TO. For PostgreSQL and MySQL support this is the name of the PostgreSQL/MySQL table to join TO.

TEMPLATE [filename]

Template to use with one-to-many joins. The template is processed once for each record and can only contain substitutions for items in the joined table. Refer to the column in the joined table in your template like [joinname_columname], where joinname is the NAME specified for the JOIN object.

TO [item]

Join item in the table to be joined. This is case sensitive.

TYPE [ONE-TO-ONE|ONE-TO-MANY]

The type of join. Default is one-to-one.

Example 1: Join from SHP file to DBF File

Mapfile Layer

LAYER

NAME prov_bound

TYPE POLYGON

STATUS DEFAULT

DATA prov.shp

CLASS

NAME "Province"

STYLE

OUTLINECOLOR 120 120 120

COLOR 255 255 0

```

END
END
TEMPLATE "../htdocs/cgi-query-templates/prov.html"
HEADER "../htdocs/cgi-query-templates/prov-header.html"
FOOTER "../htdocs/cgi-query-templates/footer.html"
JOIN
  NAME "test"
  TABLE "../data/lookup.dbf"
  FROM "ID"
  TO "IDENT"
  TYPE ONE-TO-ONE
END
END # layer

```

Ogrinfo

```

>ogrinfo lookup.dbf lookup -summary
INFO: Open of `lookup.dbf'
using driver `ESRI Shapefile' successful.

```

```

Layer name: lookup
Geometry: None
Feature Count: 12
Layer SRS WKT:
(unknown)
IDENT: Integer (2.0)
VAL: Integer (2.0)

```

```

>ogrinfo prov.shp prov -summary
INFO: Open of `prov.shp'
using driver `ESRI Shapefile' successful.

```

```

Layer name: prov
Geometry: Polygon
Feature Count: 12
Extent: (-2340603.750000, -719746.062500) - (3009430.500000, 3836605.250000)
Layer SRS WKT:
(unknown)
NAME: String (30.0)
ID: Integer (2.0)

```

Template

```

<tr bgcolor="#EFEFEF"><td align="left">[NAME]</td><td align="left">[test_VAL]</td></tr>

```

Example 2: Join from SHP file to PostgreSQL Table

Mapfile Layer

```

LAYER
  NAME prov_bound
  TYPE POLYGON
  STATUS DEFAULT

```

```
DATA prov.shp
CLASS
  NAME "Province"
  STYLE
    OUTLINECOLOR 120 120 120
    COLOR 255 255 0
  END
END
TOLERANCE 20
TEMPLATE "../htdocs/cgi-query-templates/prov.html"
HEADER "../htdocs/cgi-query-templates/prov-header.html"
FOOTER "../htdocs/cgi-query-templates/footer.html"
JOIN
  NAME "test"
  CONNECTION "host=127.0.0.1 port=5432 user=postgres password=postgres dbname=join"
  CONNECTIONTYPE ogr
  TABLE "lookup"
  FROM "ID"
  TO "ident"
  TYPE ONE-TO-ONE
END
END # layer
```

Ogrinfo

```
>ogrinfo -ro PG:"host=127.0.0.1 port=5432 user=postgres password=postgres dbname=join" lookup -summary
INFO: Open of `PG:host=127.0.0.1 port=5432 user=postgres password=postgres dbname=join'
using driver `PostgreSQL' successful.
```

```
Layer name: lookup
Geometry: Unknown (any)
Feature Count: 12
Layer SRS WKT:
(unknown)
ident: Integer (0.0)
val: Integer (0.0)
```

Template

```
<tr bgcolor="#EFEFEF"><td align="left">[NAME]</td><td align="left">[test_val]</td></tr>
```

Note

When testing with MapServer 4.10.0 on Windows this postgresql join caused a mapserv.exe crash. However when testing this with a MapServer build > 4.10.0 the crash did not occur.

Example 3: Join from SHP file to CSV file

Mapfile Layer

```
LAYER
  NAME prov_bound
  TYPE POLYGON
  STATUS DEFAULT
```

```
DATA prov.shp
CLASS
  NAME "Province"
  STYLE
    OUTLINECOLOR 120 120 120
    COLOR 255 255 0
  END
END
TOLERANCE 20
TEMPLATE "../htdocs/cgi-query-templates/prov.html"
HEADER "../htdocs/cgi-query-templates/prov-header.html"
FOOTER "../htdocs/cgi-query-templates/footer.html"
JOIN
  NAME "test"
  TABLE "../data/lookup.csv"
  FROM "ID"
  TO "IDENT"
  TYPE ONE-TO-ONE
END
END # layer
```

CSV File Structure

```
"IDENT","VAL"
1,12
2,11
3,10
4,9
5,8
6,7
7,6
8,5
9,4
10,3
11,2
12,1
```

Ogrinfo

```
>ogrinfo lookup.csv lookup -summary
INFO: Open of `lookup.csv'
using driver `CSV' successful.
```

```
Layer name: lookup
Geometry: None
Feature Count: 12
Layer SRS WKT:
(unknown)
IDENT: String (0.0)
VAL: String (0.0)
```

Template

```
<tr bgcolor="#EFEFEF"><td align="left">[NAME]</td><td align="left">[test_VAL]</td></tr>
```

9.1.6 LABEL Object

This object is used to define a label, which is in turn usually used to annotate a feature with a piece of text. Labels can however also be used as symbols through the use of various TrueType fonts.

ANGLE [double|auto|follow]

Angle, given in degrees, to draw the label or AUTO to allow the software to compute the angle, AUTO is valid for LINE layers only. FOLLOW was introduced in version 4.10 and tells map server to compute a curved label for appropriate linear features (see RFC 11 for specifics).

ANTI_ALIAS [true|false]

Should text be antialiased? Note that this requires more available colors, decreased drawing performance, and results in slightly larger output images.

BACKGROUND_COLOR [r] [g] [b]

Color to draw a background rectangle (i.e. billboard). Off by default.

BACKGROUND_SHADOW_COLOR [r] [g] [b]

Color to draw a background rectangle (i.e. billboard) shadow. Off by default.

BACKGROUND_SHADOW_SIZE [x][y]

How far should the background rectangle be offset? Default is 1.

BUFFER [integer]

Padding, in pixels, around labels. Useful for maintaining spacing around text to enhance readability. Available only for cached labels. Default is 0.

COLOR [r] [g] [b]

Color to draw text with.

ENCODING [string]

Supported encoding format to be used for labels. If the format is not supported, the label will not be drawn. Requires the iconv library (present on most systems). The library is always detected if present on the system, but if not the label will not be drawn.

Required for displaying international characters in MapServer. More information can be found at:

<http://www.foss4g.org/FOSS4G/MAPSERVER/mpsnf-i18n-en.html>.

FONT [name]

Font alias (as defined in the FONTSET) to use for labeling.

FORCE [true|false]

Forces labels for a particular class on, regardless of collisions. Available only for cached labels. Default is false.

MAX_SIZE [integer]

Maximum font size to use when scaling text (pixels). Default is 256.

MINDISTANCE [integer]

Minimum distance between duplicate labels. Given in pixels.

MINFEATURESIZE [integer|auto]

Minimum size a feature must be to be labeled. Given in pixels. For line data the overall length of the displayed line is used, for polygons features the smallest dimension of the bounding box is used. "Auto" keyword tells MapServer to only label features that are larger than their corresponding label. Available for cached labels only.

MINSIZE [integer]

Minimum font size to use when scaling text (pixels). Default is 4.

OFFSET [x][y]

Offset values for labels, relative to the lower left hand corner of the label and the label point. Given in pixels. In the case of rotated text specify the values as if all labels are horizontal and any rotation will be compensated for.

OUTLINE_COLOR [r] [g] [b]

Color to draw a one pixel outline around the text.

PARTIALS [true|false]

Can text run off the edge of the map? Default is true.

POSITION [ul|uc|ur|cl|cc|cr|ll|lc|lr|auto]

Position of the label relative to the labeling point (layers only). First letter is "Y" position, second letter is "X" position. "Auto" tells MapServer to calculate a label position that will not interfere with other labels. With points and polygons, MapServer selects from the 8 outer positions (i.e. excluding cc). With lines, it only uses lc or uc, until it finds a position that doesn't collide with labels that have already been drawn. If all positions cause a conflict, then the label is not drawn (Unless the label's [FORCE](#) a parameter is set to "true"). "Auto" placement is only available with cached labels.

PRIORITY [integer][[item_name]

The priority parameter (added in v5.0) takes an integer value between 1 (lowest) and 10 (highest). The default value is 1. It is also possible to bind the priority to an attribute (item_name) using square brackets around the [item_name]. e.g. "PRIORITY [someattribute]"

Labels are stored in the label cache and rendered in order of priority, with the highest priority levels rendered first. Specifying an out of range PRIORITY value inside a map file will result in a parsing error. An out of range value set via MapScript or coming from a shape attribute will be clamped to the min/max values at rendering time. There is no expected impact on performance for using label priorities.

SHADOWCOLOR [r] [g] [b]

Color of drop shadow.

SHADOWSIZE [x][y]

Shadow offset in pixels.

SIZE [integer][[tiny|small|medium|large|giant]

Text size. Use "integer" to give the size in pixels of your TrueType font based label, or any of the other 5 listed keywords to bitmap fonts.

TYPE [bitmap|truetype]

Type of font to use. Generally bitmap fonts are faster to draw than TrueType fonts. However, TrueType fonts are scalable and available in a variety of faces. Be sure to set the FONT parameter if you select TrueType.

WRAP [character]

Character that represents an end-of-line condition in label text, thus resulting in a multi-line label.

9.1.7 LAYER Object

The most used object in a MapFile, this one describes layers used to make up a map. Layers are drawn in their order of appearance in the MapFile (first layer is at the bottom, last in on top).

CLASS

Signals the start of a [CLASS](#) object.

CLASSITEM [attribute]

Item name in attribute table to use for class lookups.

CONNECTION [string]

Database connection string to retrieve remote data.

An SDE connection string consists of a hostname, instance name, database name, username and password separated by commas.

A PostGIS connection string is basically a regular PostgreSQL connection string, it takes the form of "user=nobody password=***** dbname=dbname host=localhost port=5432"

An Oracle connection string: user/pass[@db]

CONNECTIONTYPE [local|sde|ogr|postgis|oraclespatial|wms]

Type of connection. Default is local. See additional documentation for any other type.

DATA [filename][[sde parameters]][postgis table/column][oracle table/column]

Full filename of the spatial data to process. No file extension is necessary for shapefiles. Can be specified relative to the SHAPEPATH option from the Map Object.

If this is an SDE layer, the parameter should include the name of the layer as well as the geometry column, i.e. "mylayer,shape,myversion".

If this is a PostGIS layer, the parameter should be in the form of "<columnname> from <tablename>", where "columnname" is the name of the column containing the geometry objects and "tablename" is the name of the table from which the geometry data will be read.

For Oracle, use "shape FROM table" or "shape FROM (SELECT statement)" or even more complex Oracle compliant queries! Note that there are important performance impacts when using spatial subqueries however. Try using MapServer's [FILTER](#) whenever possible instead. You can also see the SQL submitted by forcing an error, for instance by submitting a DATA parameter you know won't work, using for example a bad column name.

DEBUG [on|off]

Enables debugging of the layer object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServer logfile if one is set using the LOG parameter in the [WEB](#) object.

DUMP [true|false]

Switch to allow mapserver to return data in GML format. Usefull when used with WMS GetFeatureInfo operations. "false" by default.

FEATURE

Signals the start of a [FEATURE](#) object.

FILTER [string]

This parameter allows for data specific attribute filtering that is done at the same time spatial filtering is done, but before any CLASS expressions are evaluated. For OGR and shapefiles the string is simply a mapserver regular expression. For spatial databases the string is a SQL WHERE clause that is valid with respect to the underlying database.

For example: FILTER "type='road' and size <2"

FILTERITEM [attribute]

Item to use with simple [FILTER](#) expressions. OGR and shapefiles only.

FOOTER [filename]

Template to use *after* a layer's set of results have been sent. Multiresult query modes only.

GRID

Signals the start of a [GRID](#) object.

GROUP [name]

Name of a group that this layer belongs to. The group name can then be reference as a regular layer name in the template files, allowing to do things like turning on and off a group of layers at once.

HEADER [filename]

Template to use *before* a layer's set of results have been sent. Multiresult query modes only.

LABELANGLEITEM [attribute]

Item name in attribute table to use for class annotation angles. Values should be in degrees.

LABELCACHE [on|off]

Specifies whether labels should be drawn as the features for this layer are drawn, or whether they should be cached and drawn after all layers have been drawn. Default is on. Label overlap removal, auto placement etc... are only available when the label cache is active.

LABELITEM [attribute]

Item name in attribute table to use for class annotation (i.e. labeling).

LABELMAXSCALE [double]

Maximum scale at which the layer is labeled.

LABELMINSCALE [double]

Minimum scale at which the layer is labeled.

LABELREQUIRES [expression]

Sets context for labeling this layer, for example:

LABELREQUIRES "![orthoquads]"

means that this layer would NOT be labeled if a layer named "orthoquads" is on. The expression consists of a boolean expression based on the status of other layers, each [layer name] substring is replaced by a 0 or a 1 depending on that layer's [STATUS](#) and then evaluated as normal. Logical operators AND and OR can be used.

LABELSIZEITEM [attribute]

Item name in attribute table to use for class annotation sizes. Values should be in pixels.

MAXFEATURES [integer]

Specifies the number of features that should be drawn for this layer in the CURRENT window. Has some interesting uses with annotation and with sorted data (i.e. lakes by area).

MAXSCALE [double]

Maximum scale at which this layer is drawn.

METADATA

This keyword allows for arbitrary data to be stored as name value pairs. This is used with OGC WMS to define things such as layer title. It can also allow more flexibility in creating templates, as anything you put in here will be accessible via template tags.

Example:

METADATA

title "My layer title"

author "Me!"

END

MINSCALE [double]

Minimum scale at which this layer is drawn.

NAME [string]

Short name for this layer. Limit is 20 characters. This name is the link between the mapfile and web interfaces that refer to

this name. They must be identical. The name should be unique, unless one layer replaces another at different scales. Use the GROUP option to associate layers with each other.

OFFSITE [r] [g] [b]

Sets the color index to treat as transparent for raster layers.

POSTLABELCACHE [true|false]

Tells MapServer to render this layer after all labels in the cache have been drawn. Useful for adding neatlines and similar elements. Default is false.

PROCESSING [string]

Passes a processing directive to be used with this layer. The supported processing directives vary by layer type, and the underlying driver that processes them. Here we see the SCALE and BANDS directives used to autoscale raster data and alter the band mapping. All raster processing options are described in the Raster Data Access HOWTO.

PROCESSING "SCALE=AUTO"

PROCESSING "BANDS=3,2,1"

This is also where you can enable connection pooling for certain layer types. Connection pooling will allow MapServer to share the handle to an open database or layer connection throughout a single map draw process. Additionally, if you have FastCGI enabled, the connection handle will stay open indefinitely, or according to the options specified in the FastCGI configuration. Oracle, ArcSDE, OGR and PostGIS currently support this approach.

PROCESSING "CLOSE_CONNECTION=DEFER"

PROJECTION

Signals the start of a [PROJECTION](#) object.

REQUIRES [expression]

Sets context for displaying this layer (see [LABELREQUIRES](#)).

SIZEUNITS [pixels|feet|inches|kilometers|meters|miles]

Sets the unit of [CLASS](#) object SIZE values (default is pixels). Useful for simulating buffering.

STATUS [on|off|default]

Sets the current status of the layer. Often modified by MapServer itself. Default turns the layer on permanently.

Some notes regarding the STATUS values:

- In CGI mode, layers with STATUS DEFAULT cannot be turned off using normal mechanisms. It is recommended to set layers to STATUS DEFAULT while debugging a problem, but set them back to ON/OFF in normal use.
- For WMS, layers in the server mapfile with STATUS DEFAULT are always sent to the client.

STYLEITEM [attribute]

Item to use for feature specific styling. This is *very* experimental and OGR only at the moment.

SYMBOLSCALE [double]

The scale at which symbols and/or text appear full size. This allows for dynamic scaling of objects based on the scale of the map. If not set then this layer will always appear at the same size. Scaling only takes place within the limits of MINSIZE and MAXSIZE as described above.

TEMPLATE [file|url]

Used as a global alternative to CLASS [TEMPLATE](#).

TILEINDEX [filename|layername]

Name of the tileindex file or layer. A tileindex is similar to an ArcInfo library index. The tileindex contains polygon features for each tile. The item that contains the location of the tiled data is given using the TILEITEM parameter. When a file is used as the tileindex for shapefile or raster layers, the tileindex should be a shapefile. For CONNECTIONTYPE OGR layers, any OGR supported datasource can be a tileindex. Normally the location should contain the path to the tile file relative to the shapepath, not relative to the tileindex itself. If the DATA parameter contains a value then it is added to the end of the location. When a tileindex layer is used, it works similarly to directly referring to a file, but any supported feature source can be used (ie. postgres, oracle).

NOTE: All files in the tileindex should have the same coordinate system, and for vector files the same set of attributes in the same order.

TILEITEM [attribute]

Item that contains the location of an individual tile, default is "location".

TOLERANCE [double]

Sensitivity for point based queries (i.e. via mouse and/or map coordinates). Given in TOLERANCEUNITS. If the layer is a POINT or a LINE, the default is 3. For all other layer types, the default is 0. To restrict polygon searches so that the point must occur in the polygon set the tolerance to zero.

TOLERANCEUNITS [pixels|feet|inches|kilometers|meters|miles|dd]

Units of the [TOLERANCE](#) value. Default is pixels.

TRANSPARENCY [integer|alpha]

Sets the transparency level of all classed pixels for a given layer. The value can either be an integer in the range (0-100) or the named symbol "ALPHA". Although this parameter is named "transparency", the integer values actually parameterize layer opacity. A value of 100 is opaque and 0 is fully transparent.

The "ALPHA" symbol directs the mapserver rendering code to honor the indexed or alpha transparency of pixmap symbols used to style a layer. This is only needed in the case of RGB output formats, and should be used only when necessary as it is expensive to render transparent pixmap symbols onto an RGB map image.

TRANSFORM [true|false ul|uc|ur|lc|cc|lr|ll|lc|lr]

Tells MapServer whether or not a particular layer needs to be transformed from some coordinate system to image coordinates. Default is true. This allows you to create shapefiles in image/graphics coordinates and therefore have features that will always be displayed in the same location on every map. Ideal for placing logos or text in maps. Remember that the graphics coordinate system has an origin in the upper left hand corner of the image, contrary to most map coordinate systems.

Version 4.10 introduces the ability to define features with coordinates given in pixels (or percentages, see UNITS), most often inline features, relative to something other than the UL corner of an image. That is what 'TRANSFORM FALSE' means. By setting an alternative origin it allows you to anchor something like a copyright statement to another portion of the image in a way that is independent of image size.

TYPE [point|line|polygon|circle|annotation|raster|query|chart]

Specifies how the data should be drawn. Need not be the same as the shapefile type. For example, a polygon shapefile may be drawn as a point layer, but a point shapefile may not be drawn as a polygon layer. Common sense rules. Annotation means that a label point will be calculated for the features, but the feature itself will not be drawn although a marker symbol can be optionally drawn. this allows for advanced labeling like numbered highway shields. Points are labeled at that point. Polygons are labeled first using a centroid, and if that doesn't fall in the polygon a scanline approach is used to guarantee the label falls within the feature. Lines are labeled at the middle of the longest arc in the visible portion of the line. Query only means the layer can be queried but not drawn.

In order to differentiate between POLYGONS and POLYLINES (which do not exist as a type), simply respectively use or omit the COLOR keyword when classifying. If you use it, it's a polygon with a fill color, otherwise it's a polyline with only an OUTLINECOLOR.

For CHART layers, see the [Dynamic Charting howto](#).

A circle must be defined by a a minimum bounding rectangle. That is, 2 points that define the smallest square that can contain it. These 2 points are the two opposite corners of said box.

The following is an example using inline points to draw a circle:

```
LAYER
  NAME 'inline_circles'
  TYPE CIRCLE
  STATUS ON
  FEATURE
  POINTS
    74.01 -53.8
    110.7 -22.16
  END
END
CLASS
  STYLE
    COLOR 0 0 255
  END
END
END
```

9.1.8 LEGEND Object

Defines how a legend is to be built. Legend components are built automatically from class objects from individual layers. Starts with the keyword LEGEND and terminates with the keyword END.

The size of the legend image is NOT known prior to creation so be careful not to hard-code width and height in the tag in the template file.

IMAGECOLOR [r] [g] [b]

Color to initialize the legend with (i.e. the background).

INTERLACE [on|off]

Deprecated Default is [on]. This keyword is now deprecated in favour of using the FORMATOPTION "INTERLACE=ON" line in the [OUTPUTFORMAT](#) declaration.

LABEL

Signals the start of a [LABEL](#) object

OUTLINECOLOR [r] [g] [b]

Color to use for outlining symbol key boxes.

POSITION [ul|uc|ur|ll|lc|lr]

Where to place an embedded legend in the map. Default is lr.

KEYSIZE [x][y]

Size of symbol key boxes in pixels. Default is 20 by 10.

KEYSPACING [x][y]

Spacing between symbol key boxes ([y]) and labels ([x]) in pixels. Default is 5 by 5.

POSTLABELCACHE [true|false]

Tells MapServer to render this legend after all labels in the cache have been drawn. Useful for adding neatlines and similar elements. Default is false.

STATUS [on|off|embed]

Is the legend image to be created.

TEMPLATE [filename]

HTML legend template file. Refer to the [HTML Legend howto](#).

TRANSPARENT [on|off]

Deprecated Should the background color for the legend be transparent. This flag is now deprecated in favour of declaring transparency within [OUTPUTFORMAT](#) declarations. Default is off.

9.1.9 MAP Object

A MAP object defines the master object of the mapfile. It defines application/map wide parameters.

ANGLE [double]

Angle, given in degrees, to rotate the map. Default is 0. The rendered map will rotate in a clockwise direction. The following are important notes:

- Requires a PROJECTION object specified at the MAP level and for each LAYER object (even if all layers are in the same projection).
- Requires MapScript (SWIG, PHPMapscript). Does not work with CGI mode.
- If using the LABEL object's ANGLE or the LAYER object's LABELANGLEITEM parameters as well, these parameters are relative to the map's orientation (i.e. they are computed after the MAP object's ANGLE). For example, if you have specified an ANGLE for the map of 45, and then have a layer LABELANGLEITEM value of 45, the resulting label will not appear rotated (because the resulting map is rotated clockwise 45 degrees and the label is rotated counter-clockwise 45 degrees).
- More information can be found on the MapRotation [Wiki Page](#).

CONFIG [key] [value]

This can be used to define the location of your EPSG files for the PROJ.4 library. Setting the [key] to PROJ_LIB and the [value] to the location of your EPSG files will force PROJ.4 to use this value. Using CONFIG allows you to avoid setting environment variables to point to your PROJ_LIB directory. Here are some examples:

1. Unix

```
CONFIG "PROJ_LIB" "/usr/local/share/proj/"
```

2. Windows

```
CONFIG "PROJ_LIB" "C:/somedir/proj/nad/"
```

Any other value will be passed on to `CPLSetConfigOption()`. This provides customized control of some GDAL and OGR driver behaviours. Details on such options would be found in specific GDAL driver documentation.

DATAPATTERN [regular expression]

This defines a regular expression to be applied to requests to change DATA parameters via URL requests (i.e. `map_layername_data=...`). If a pattern doesn't exist then web users can't monkey with support files via URLs. This allows you to isolate one application from another if you desire, with the default operation being very conservative. See also [TEMPLATEPATTERN](#).

DEBUG [on|off]

Enables debugging of the map object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServer logfile if one is set using the LOG parameter in the WEB object. Apache users will see timing details for drawing in Apache's `error_log` file.

Requires MapServer to be built with the `DEBUG=MSDEBUG` option (`--with-debug` configure option).

EXTENT [minx] [miny] [maxx] [maxy]

The spatial extent of the map to be created. In most cases you will need to specify this, although mapserver can sometimes (expensively) calculate one if it is not specified.

FONTSET [filename]

Filename of fontset file to use. Can be a path relative to the mapfile, or a full path.

IMAGECOLOR [r] [g] [b]

Color to initialize the map with (i.e. background color). When transparency is enabled (`TRANSPARENT ON`) for the typical case of 8-bit pseudocolored map generation, this color will be marked as transparent in the output file palette. Any other map components drawn in this color will also be transparent, so for map generation with transparency it is best to use an otherwise unused color as the background color.

IMAGEQUALITY [int]

Deprecated Use `FORMATOPTION "QUALITY=n"` in the `OUTPUTFORMAT` declaration to specify compression quality for JPEG output.

IMAGETYPE [gif|png|jpeg|wbmp|gtiff|swf|userdefined]

Output format to generate. See details in the [OUTPUTFORMAT](#) section for available formats. The name here must match the 'NAME' of a user defined or internally generated `OUTPUTFORMAT` section.

INTERLACE [on|off]

Deprecated Use `FORMATOPTION "INTERLACE=ON"` in the [OUTPUTFORMAT](#) declaration to specify if the output images should be interlaced.

LAYER

Signals the start of a [LAYER](#) object.

LEGEND

Signals the start of a [LEGEND](#) object.

MAXSIZE [integer]

Sets the maximum size of the map image. This will override the default value. For example, setting this to 2048 means that you can have up to 2048 pixels in both dimensions (i.e. max of 2048x2048).

NAME [name]

Prefix attached to map, scalebar and legend GIF filenames created using this MapFile. It should be kept short.

PROJECTION

Signals the start of a [PROJECTION](#) object.

QUERYMAP

Signals the start of a [QUERYMAP](#) object.

REFERENCE

Signals the start of a [REFERENCE MAP](#) object.

RESOLUTION [int]

Sets the pixels per inch for output, only affects scale computations and nothing else, default is 72.

SCALE [double]

Computed scale of the map. Set most often by the application.

SCALEBAR

Signals the start of a [SCALEBAR](#) object.

SHAPEPATH [filename]

Path to the directory holding the shapefiles or tiles. There can be further subdirectories under SHAPEPATH.

SIZE [x][y]

Size in pixels of the output image (i.e. the map).

STATUS [on|off]

Is the map active? Sometimes you may wish to turn this off to use only the reference map or scale bar.

SYMBOLSET [filename]

Filename of the symbolset to use. Can be a path relative to the mapfile, or a full path.

SYMBOL

Signals the start of a [SYMBOL](#) object.

TEMPLATEPATTERN [regular expression]

This defines a regular expression to be applied to requests to change TEMPLATE parameters via URL requests (i.e. map_layername_template=...). If a pattern doesn't exist then web users can't monkey with support files via URLs.

This allows you to isolate one application from another if you desire, with the default operation being very conservative. See also [DATAPATTERN](#).

TRANSPARENT [on|off]

Deprecated Use FORMATOPTION "TRANSPARENT=ON" in the OUTPUTFORMAT declaration to specify if the output images should be transparent.

UNITS [feet|inches|kilometers|meters|miles|dd]

Units of the map coordinates. Used for scalebar and scale computations.

WEB

Signals the start of a [WEB](#) object.

9.1.10 OUTPUTFORMAT Object

This section discusses how output formats are defined and selected.

A map file may have zero, one or more OUTPUTFORMAT object declarations, defining available output formats supported including formats like PNG, GIF, JPEG, GeoTIFF and Flash (SWF).

If OUTPUTFORMAT sections declarations are not found in the map file, the following implicit declarations will be made. Only those for which support is compiled in will actually be available. The GeoTIFF depends on building with GDAL support, and the Flash (SWF) depends on compiling with support for the MING library.

```
OUTPUTFORMAT
NAME gif
DRIVER "GD/GIF"
MIMETYPE "image/gif"
IMAGEMODE PC256
EXTENSION "gif"
END
OUTPUTFORMAT
NAME png
DRIVER "GD/PNG"
MIMETYPE "image/png"
IMAGEMODE PC256
EXTENSION "png"
END
OUTPUTFORMAT
NAME jpeg
DRIVER "GD/JPEG"
MIMETYPE "image/jpeg"
IMAGEMODE RGB
EXTENSION "jpg"
END
OUTPUTFORMAT
NAME wbmp
DRIVER "GD/WBMP"
MIMETYPE "image/wbmp"
IMAGEMODE PC256
EXTENSION "wbmp"
END
OUTPUTFORMAT
NAME swf
DRIVER "SWF"
MIMETYPE "application/x-shockwave-flash"
EXTENSION "swf"
IMAGEMODE PC256
FORMATOPTION "OUTPUT_MOVIE=SINGLE"
END
OUTPUTFORMAT
NAME GTiff
DRIVER "GDAL/GTiff"
MIMETYPE "image/tiff"
IMAGEMODE RGB
EXTENSION "tif"
END
```

NAME [name]

The name to use in the IMAGETYPE keyword of the map file to select this output format.(optional)

DRIVER [name]

The name of the driver to use to generate this output format. Some driver names include the definition of the format if the driver supports multiple formats. For GD the possible driver names are "GD/Gif", "GD/PNG", "GD/WBMP" and "GD/JPEG". For flash the driver is just called "SWF". For output through GDAL the GDAL shortname for the format is appended, such as "GDAL/GTiff". Note that PNG, JPEG and GIF output can be generated with either GDAL or GD (GD is generally more efficient).(mandatory)

IMAGEMODE [PC256/RGB/RGBA/INT16/FLOAT32]

Selects the imaging mode in which the output is generated. Does matter for non-raster formats like Flash. Not all formats support all combinations. For instance GD/GIF supports only PC256. (optional)

- PC256: Produced a pseudocolored result with up to 256 colors in the palette (traditional MapServer mode)
- RGB: Render in 24bit Red/Green/Blue mode. Supports all colors but does not support transparency.
- RGBA: Render in 32bit Red/Green/Blue/Alpha mode. Supports all colors, and alpha based transparency.
- BYTE: Render raw 8bit pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.
- INT16: Render raw 16bit signed pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.
- FLOAT32: Render raw 32bit floating point pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.

MIMETYPE [type]

Provide the mime type to be used when returning results over the web. (optional)

EXTENSION [type]

Provide the extension to use when creating files of this type. (optional)

TRANSPARENT [ON/OFF]

Indicates whether transparency should be enabled for this format. Note that transparency does not work for IMAGEMODE RGB output. Not all formats support transparency (optional). When transparency is enabled for the typical case of 8-bit pseudocolored map generation, the IMAGECOLOR color will be marked as transparent in the output file palette. Any other map components drawn in this color will also be transparent, so for map generation with transparency it is best to use an otherwise unused color as the background color.

FORMATOPTION [option]

Provides a driver or format specific option. Zero or more FORMATOPTION statement may be present within a OUTPUTFORMAT declaration. (optional)

- GD/JPEG: The "QUALITY=n" option may be used to set the quality of jpeg produced (value from 0-100).
- GD/PNG: The "INTERLACE=[ON/OFF]" option may be used to turn interlacing on or off.
- GD/GIF: The "INTERLACE=[ON/OFF]" option may be used to turn interlacing on or off.
- GDAL/GTiff: Supports the TILED=YES, BLOCKXSIZE=n, BLOCKYSIZE=n, INTERLEAVE=[PIXEL/BAND] and COMPRESS=[NONE,PACKBITS,JPEG,LZW,DEFLATE] format specific options.
- GDAL/*: All FORMATOPTIONs are passed onto the GDAL create function. Options supported by GDAL are described in the detailed documentation for each GDAL format

9.1.11 PROJECTION Object

This object defines the coordinate system to display your data.

To set up projections you must define two projection objects: one for the output image (in the MAP object) and one for each layer (in the LAYER objects) to be projected. MapServer relies on the [PROJ.4](#) library for projections. Projection objects therefore consist of a series of PROJ.4 keywords, which are either specified within the object directly or referred to in an *epsg* file. An *epsg* file is a lookup file containing projection parameters, and is part of the PROJ.4 library.

The following two examples both define the same projection (UTM zone 15, NAD83), but use 2 different methods:

Example1: Inline Projection Parameters

PROJECTION

"proj=utm"

"ellps=GRS80"

"datum=NAD83"

"zone=15"

```
"units=m"  
"north"  
"no_defs"  
END
```

Example2: epsg Projection Use

```
PROJECTION  
"init=epsg:26915"  
END
```

(this refers to an epsg lookup file that contains a '26915' code with the full projection parameters)

The next two examples both display how to possibly define unprojected lat/longs ("geographic"):

Example3: Inline Projection Parameters

```
PROJECTION  
"proj=latlong"  
"ellps=WGS84"  
"datum=WGS84"  
END
```

Example4: epsg Projection Use

```
PROJECTION  
"init=epsg:4326"  
END
```

Important Notes:

- If all of your data in the mapfile is in the same projection, you DO NOT have to specify any projection objects. MapServer will assume that all of the data is in the same projection.
- If you specify a MAP-level projection, and then only one other LAYER projection object, MapServer will assume that all of the other layers are in the specified MAP-level projection.
- Always refer to the epsg file in lowercase, because it is a lowercase filename and on Linux/Unix systems this parameter is case sensitive.

For More Information:

- If you get projection errors, refer to the [MapServer Error page](#) to check if your exact error has been discussed.
 - Search the MapServer-users [email list archives](#), odds are that someone has faced your exact issue before.
- See the [PROJ.4](#) user guides for complete descriptions of supported projections and coordinate systems.
- Refer to the [Cartographical Map Projections](#) page for background information on projections.

9.1.12 QUERYMAP Object

Defines a mechanism to map the results of a query. Starts with the keyword QUERYMAP and terminates with the keyword END.

COLOR [r] [g] [b]

Color in which features are highlighted. Default is yellow.

SIZE [x][y]

Size of the map in pixels. Defaults to the size defined in the map object.

STATUS [on|off]

Is the query map to be drawn?

STYLE [normal|hilite|selected]

Sets how selected features are to be handled. Layers not queried are drawn as usual.

- Normal: Draws all features according to the settings for that layer.
- Hilite: Draws selected features using COLOR. Non-selected features are drawn normally.
- Selected: draws only the selected features normally.

9.1.13 REFERENCE Object

Defines how reference maps are to be created. Starts with the keyword REFERENCE and terminates with the keyword END.

Three types of reference maps are supported. The most common would be one showing the extent of a map in an interactive interface. It is also possible to request reference maps as part of a query. Point queries will generate an image with a marker (see below) placed at the query point. Region based queries will depict the extent of the area of interest. Finally, feature based queries will display the selection feature(s) used.

COLOR [r] [g] [b]

Color in which the reference box is drawn. Set any component to -1 for no fill. Default is red.

EXTENT [minx][miny][maxx][maxy]

The spatial extent of the base reference image.

IMAGE [filename]

Full filename of the base reference image. Must be a GIF image.

MARKER [integer|string]

Defines a symbol (from the symbol file) to use when the box becomes too small (see MINBOXSIZE and MAXBOXSIZE below). Uses a crosshair by default.

MARKERSIZE [integer]

Defines the size of the symbol to use instead of a box (see MARKER above).

MINBOXSIZE [integer]

If box is smaller than MINBOXSIZE (use box width or height) then use the symbol defined by MARKER and MARKERSIZE.

MAXBOXSIZE [integer]

If box is greater than MAXBOXSIZE (use box width or height) then draw nothing (Often the whole map gets covered when zoomed way out and it's perfectly obvious where you are).

OUTLINECOLOR [r] [g] [b]

Color to use for outlining the reference box. Set any component to -1 for no outline.

SIZE [x][y]

Size, in pixels, of the base reference image.

STATUS [on|off]

Is the reference map to be created? Default it off.

9.1.14 SCALEBAR Object

Defines how a scalebar should be built. Starts with the keyword SCALEBAR and terminates with the keyword END. Scalebars currently do not make use of TrueType fonts. The size of the scalebar image is NOT known prior to rendering, so be careful not to hard-code width and height in the tag in the template file. Future versions will make the image size available.

ALIGN [left|center|right]

Defines how the scalebar is aligned within the scalebar image. Default is center. Available in versions 5.2 and higher.

BACKGROUNDCOLOR [r] [g] [b]

Color to use for scalebar background, not the image background.

COLOR [r] [g] [b]

Color to use for drawing all features if attribute tables are not used.

IMAGECOLOR [r] [g] [b]

Color to initialize the scalebar with (i.e. background).

INTERLACE [true|false]

Should output images be interlaced? Default is [on]. This keyword is now deprecated in favour of using the FORMATOPTION "INTERLACE=ON" line in the [OUTPUTFORMAT](#) declaration.

INTERVALS [integer]

Number of intervals to break the scalebar into. Default is 4.

LABEL

Signals the start of a [LABEL](#) object

OUTLINECOLOR [r] [g] [b]

Color to use for outlining individual intervals. Set any component to -1 for no outline which is the default.

POSITION [ul|uc|ur|ll|lc|lr]

Where to place an embedded scalebar in the image. Default is lr.

POSTLABELCACHE [true|false]

For use with embedded scalebars only. Tells the MapServer to embed the scalebar after all labels in the cache have been drawn. Default is false.

SIZE [x][y]

Size in pixels of the scalebar. Labeling is not taken into account.

STATUS [on|off|embed]

Is the scalebar image to be created, and if so should it be embedded into the image? Default is off. (Please note that embedding scalebars require that you define a markerset. In essence the scalebar becomes a custom marker that is handled just like any other annotation.)

STYLE [integer]

Chooses the scalebar style. Valid styles are 0 and 1.

TRANSPARENT [on|off]

Should the background color for the scalebar be transparent. This flag is now deprecated in favour of declaring transparency within [OUTPUTFORMAT](#) declarations. Default is off.

UNITS [feet|inches|kilometers|meters|miles]

Output scalebar units, default is miles. Used in conjunction with the map's units to develop the actual graphic. Note that decimal degrees are not valid scalebar units.

9.1.15 STYLE Object

This object holds parameters for symbolization. Multiple styles may be applied within a class.

This object is new in 4.0 and is intended to separate logic from looks. The final intent is to have named styles (Not yet supported) that will be re-usable through the mapfile. This is the new, preferred way of defining the appearance of an object, notably a class.

ANGLE [double]

Angle, given in degrees, to draw the line work. Default is 0. For symbols of Type HATCH, this is the angle of the hatched lines. For its use with hatched lines, see Example#8 in the [SYMBOL examples](#).

ANGLEITEM [string]

Attribute/field that stores the angle to be used in rendering. Angle is given in degrees with 0 meaning no rotation.

ANTIALIAS [true|false]

Should TrueType fonts and Cartoline symbols be antialiased.

BACKGROUND_COLOR [r] [g] [b]

Color to use for non-transparent symbols.

COLOR [r] [g] [b]

Color to use for drawing features.

MAXSIZE [integer]

Maximum size in pixels to draw a symbol. Default is 50.

MINSIZE [integer]

Minimum size in pixels to draw a symbol. Default is 0.

MINWIDTH [integer]

Minimum width in pixels to draw the line work.

OFFSET [x][y]

Offset values for shadows, hollow symbols, etc ...

OUTLINE_COLOR [r] [g] [b]

Color to use for outlining polygons and certain marker symbols. Line symbols do not support outline colors.

SIZE [integer]

Height, in pixels, of the symbol/pattern to be used. Only useful with scalable symbols. Default is 1. For symbols of Type HATCH, the SIZE is the distance between hatched lines. For its use with hatched lines, see Example#8 in the [SYMBOL examples](#).

SIZEITEM [string]

Attribute/field that stores the size to be used in rendering. Value is given in pixels.

SYMBOL [integer|string|filename]

The symbol name or number to use for all features if attribute tables are not used. The number is the index of the symbol in the symbol file, starting at 1, the 5th symbol in the file is therefore symbol number 5. You can also give your symbols names using the NAME keyword in the symbol definition file, and use those to refer to them. Default is 0, which results in a single pixel, single width line, or solid polygon fill, depending on layer type.

You can also specify a gif or png filename. The path is relative to the location of the mapfile.

WIDTH [integer]

Width refers to the thickness of line work drawn, in pixels. Default is 1. For symbols of Type HATCH, the WIDTH is how thick the hatched lines are. For its use with hatched lines, see Example#8 in the [SYMBOL examples](#).

9.1.16 WEB Object

Defines how a web interface will operate. Starts with the keyword WEB and terminates with the keyword END.

EMPTY [url]

URL to forward users to if a query fails. If not defined the value for ERROR is used.

ERROR [url]

URL to forward users to if an error occurs. Ugly old MapServer error messages will appear if this is not defined

FOOTER [filename]

Template to use AFTER anything else is sent. Multiresult query modes only.

HEADER [filename]

Template to use BEFORE everything else has been sent. Multiresult query modes only.

IMAGEPATH [path]

Path to the temporary directory for writing temporary files and images. Must be writable by the user the web server is running as. Must end with a / or depending on your platform.

IMAGEURL [path]

Base URL for IMAGEPATH. This is the URL that will take the web browser to IMAGEPATH to get the images.

LOG [filename]

File to log MapServer activity in. Must be writable by the user the web server is running as.

MAXSCALE [double]

Maximum scale at which this interface is valid. When a user requests a map at a bigger scale, MapServer automatically returns the map at this scale. This effectively prevents user from zooming too far out.

MAXTEMPLATE [file|url]

Template to be used if above the maximum scale for the app, useful for nesting apps.

METADATA

This keyword allows for arbitrary data to be stored as name value pairs. This is used with OGC WMS to define things such as layer title. It can also allow more flexibility in creating templates, as anything you put in here will be accessible via template tags. Example:

```
METADATA
  title "My layer title"
  author "Me!"
END
```

MINSCALE [double]

Minimum scale at which this interface is valid. When a user requests a map at a smaller scale, MapServer automatically returns the map at this scale. This effectively prevents the user from zooming in too far.

MINTEMPLATE

Template to be used if above the minimum scale for the app, useful for nesting apps.

OUTPUTFORMAT [mime-type]

Format of the query output. Default is "text/html". This is experimental, the use of the [OUTPUTFORMAT](#) object is recommended instead.

TEMPLATE [filename|url]

Template file or URL to use in presenting the results to the user in an interactive mode (i.e. map generates map and so on ...)

9.2 Googlish Mapfile Example

```
#####  
#####  
##Author: Bob Basques - bob.basques (at) ci.stpaul.mn.us  
##Description: Attempt at a "googlish" look and feel applied to the TLG data layer  
##Source: http://www.geomoose.org/moose/download/mapfiles-for-mapserver  
#####  
  
#####  
## Road classifications  
#####  
  
## Feature class codes (f_class) are generalized versions of the U.S. census bureau standards. Namely:  
##  
## Interstates A10  
## Interstate HOV lanes A15  
## US highways A20  
## State highways A25  
## County Roads A30  
## City Streets A40  
## Ramps and Loops A63  
## Service Drives A64  
## Private Roads A66  
## Dormitory Walkways A71  
## Restricted Access A98  
  
#####  
## Code Index  
#####  
  
## FG = Foreground color  
## BG = Background color  
  
#####
```

Scaling

#####

< 10000 = Bottom of the Stack for Google
10000 - 15000 = local roads (A40) shrink
15001 - 20000 = local roads (A40) shrink again and labels turn off
collectors and arterials (A30, A25 and A20) shrink
20001 - 25000 = Local roads (A40) color overlay turn off.
##

MAP

NAME 'label_shields'
SIZE 800 650
STATUS ON
EXTENT 332000 -18000 678000 328000
UNITS FEET
FONTSET ./interstate.fonts
SYMBOLSET ./interstate.symbols
TRANSPARENT TRUE
#IMAGECOLOR 239 235 231 #GOOGLE BG COLOR

#IMAGETYPE PNG ## This uses the MapServer GD image builder
IMAGETYPE AGGA ## This uses the AGG image builder if compiled into MapServer. It's defined below.

LEGEND

STATUS ON
KEYSPACING 5 10
KEYSIZE 20 12
IMAGECOLOR 255 255 255
LABEL
TYPE TRUETYPE
FONT bluehigh
COLOR 0 0 0
OUTLINECOLOR 254 254 254


```
SIZE 12
    END
END
```

```
OUTPUTFORMAT
NAME 'AGGA'
DRIVER AGG/PNG
IMAGEMODE RGBA
#FORMATOPTION "TRANSPARENT=ON"
#TRANSPARENT ON
END
```

```
OUTPUTFORMAT
NAME 'AGG_Q'
DRIVER AGG/PNG
IMAGEMODE RGBA
FORMATOPTION "QUANTIZE_FORCE=ON"
FORMATOPTION "QUANTIZE_DITHER=OFF"
FORMATOPTION "QUANTIZE_COLORS=255"
TRANSPARENT ON
END
```

```
SYMBOL
NAME 'dashed'
TYPE ELLIPSE
POINTS 1 1 END
FILLED TRUE
STYLE 10 8 10 8 END
END
```

```
SYMBOL
NAME 'dotted'
TYPE ELLIPSE
POINTS 1 1 END
FILLED TRUE
```

```
STYLE 1 5 1 5 END  
END
```

```
SYMBOL  
NAME 'circle'  
TYPE ELLIPSE  
POINTS 1 1 END  
FILLED TRUE  
END
```

```
WEB  
  HEADER "tlg_header.xml"  
  FOOTER "tlg_footer.xml"  
END
```

```
#####  
## Text Labels - Only included for the Local and collector Roads  
#####
```

```
LAYER  
  METADATA  
    "ows_title" "local_label"  
    "gml_include_items" "all"  
  END  
  NAME 'local_label'  
  GROUP 'interstate_poly'  
  TILEINDEX 'TILEINDEX'  
  ##TILEINDEX 'tgr_interstates_line_idx'  
  STATUS DEFAULT  
  TYPE LINE  
  CLASSITEM 'F_CLASS'  
  LABELITEM 'STREETALL'  
  CLASS  
    EXPRESSION 'A40' # Local Road  
    LABEL
```

```
TYPE TRUETYPE
FONT bluehigh
ANGLE FOLLOW
SIZE 10
BACKGROUNDCOLOR 255 255 254
COLOR 28 28 28
MINDISTANCE 125
MINFEATURESIZE 25
OFFSET 0 0
PARTIALS FALSE
END
MAXSCALE 3000
END
CLASS
MINSCALE 3001
EXPRESSION 'A40' # Local Road
LABEL
    TYPE TRUETYPE
    FONT bluehigh
    ANGLE FOLLOW
    SIZE 9
    BACKGROUNDCOLOR 255 255 254
    COLOR 28 28 28
    MINDISTANCE 250
    MINFEATURESIZE 50
    OFFSET 0 0
    PARTIALS FALSE
END
MAXSCALE 10000
END
CLASS
EXPRESSION 'A30' # Local Road
LABEL
    TYPE TRUETYPE
```

```
    FONT bluehigh
    ANGLE FOLLOW
SIZE 11
BACKGROUNDCOLOR 255 251 115 # Same as line work color
COLOR 0 0 0
MINDISTANCE 175
MINFEATURESIZE 5
OFFSET -1 0
PARTIALS FALSE
END
MAXSCALE 3000
END
CLASS
  MINSCALE 3001
  EXPRESSION 'A30' # Local Road
  LABEL
    TYPE TRUETYPE
    FONT action
    ANGLE FOLLOW
    SIZE 8
    BACKGROUNDCOLOR 255 251 115 # Same as line work color
    COLOR 0 0 0
    MINDISTANCE 300
    MINFEATURESIZE 10
    OFFSET -1 0
    PARTIALS FALSE
  END
  MAXSCALE 8000
END
END
```

```
#####
## Symbols (Shields, etc)
#####
```

```
LAYER
METADATA
  "ows_title" "road_sym"
  "gml_include_items" "all"
END
NAME 'road_sym'
GROUP 'interstate_poly'
TILEINDEX 'TILEINDEX'
STATUS DEFAULT
TYPE ANNOTATION
CLASSITEM 'F_CLASS'
LABELITEM 'HIGHWAY_NU'
TOLERANCE 1000
CLASS
  EXPRESSION 'A10' # interstates
  STYLE
    SYMBOL 'interstate1_wide_back'
    SIZE 22
    COLOR 255 0 0
  END
  STYLE
    SYMBOL 'interstate1_wide_front'
    SIZE 22
    COLOR 115 113 206
    OUTLINECOLOR 254 254 254
  END
LABEL
  TYPE TRUETYPE
  FONT bluehigh
  SIZE 12
  COLOR 254 254 254
  MINDISTANCE 200
  MINFEATURESIZE 10
  OFFSET 0 0
  PARTIALS FALSE
```

```
END
END

CLASS
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'us_highway_back'
    SIZE 23
    COLOR 254 254 254
  END
  STYLE
    SYMBOL 'us_highway_front'
    SIZE 23
    COLOR 0 0 0
  END
  LABEL
    TYPE TRUETYPE
    FONT bluehigh
    SIZE 11
    COLOR 0 0 0
    MINDISTANCE 250
    MINFEATURESIZE 15
    OFFSET 0 -1
    PARTIALS FALSE
  END
  MAXSCALE 400000
END
```

```
CLASS
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'mn_highway_back'
    SIZE 23
    COLOR 254 254 254
  END
```

```
STYLE
  SYMBOL 'mn_highway_front'
  SIZE 23
  COLOR 0 0 0
END
LABEL
  TYPE TRUETYPE
  FONT bluehigh
  SIZE 11
  COLOR 0 0 0
  MINDISTANCE 250
  MINFEATURESIZE 15
  OFFSET 0 -1
  PARTIALS FALSE
END
MAXSCALE 400000
END
```

```
CLASS
  EXPRESSION 'A30' # State Highway
  STYLE
    SYMBOL 'state_highway_back'
    SIZE 23
    COLOR 254 254 254
  END
  STYLE
    SYMBOL 'state_highway_front'
    SIZE 23
    COLOR 0 0 0
  END
  LABEL
    TYPE TRUETYPE
    FONT bluehigh
    SIZE 11
    COLOR 0 0 0
```

```
MINDISTANCE 250
MINFEATURESIZE 15
OFFSET 0 -1
PARTIALS FALSE
END
MAXSCALE 4000000
END
END
```

```
#####
#### Begin Linework Section
#####
```

```
#####
#### Linework for background color shadow (gray) for all but Interstates (A10), which
#### is combined with the Interstate ForeGround definition further down.
#####
```

```
LAYER
METADATA
  "ows_title" "shadow_comb_bg"
  "gml_include_items" "all"
END
NAME 'shadow'
GROUP 'interstate_poly'
TILEINDEX 'TILEINDEX'
STATUS DEFAULT
TYPE LINE
CLASSITEM 'F_CLASS'
CLASS
  EXPRESSION 'A63' # Access Ramp
  STYLE
    SYMBOL 'circle'
    SIZE 6
```



```
    COLOR 175 175 175
END
MAXSCALE 40000
END
#####
CLASS
  EXPRESSION 'A40' # Local Road
  STYLE
    SYMBOL 'circle'
    SIZE 16
    COLOR 175 175 175
  END
  MAXSCALE 3000
END
CLASS
  MINSCALE 3001
  EXPRESSION 'A40' # Local Road
  STYLE
    SYMBOL 'circle'
    SIZE 14
    COLOR 175 175 175
  END
  MAXSCALE 10000
END
CLASS
  MINSCALE 10001
  EXPRESSION 'A40' # Local Road
  STYLE
    SYMBOL 'circle'
    SIZE 7
    COLOR 175 175 175
  END
  MAXSCALE 15000
END
CLASS
```

```
MINSCALE 15001
EXPRESSION 'A40' # Local Road
STYLE
  SYMBOL 'circle'
  SIZE 3
  COLOR 175 175 175
END
MAXSCALE 25000
END
CLASS
MINSCALE 25001
EXPRESSION 'A40' # Local Road, Gray only level, no colored overlay
STYLE
  SYMBOL 'circle'
  SIZE 3
  COLOR 175 175 175
END
MAXSCALE 50000
END
CLASS
MINSCALE 50001
EXPRESSION 'A40' # Local Road, Gray only level, no colored overlay
STYLE
  SYMBOL 'circle'
  SIZE 1
  COLOR 200 200 200
END
MAXSCALE 160000
END

#####
CLASS
EXPRESSION 'A30' # Secondary Connecting Road
STYLE
  SYMBOL 'circle'
```

```
SIZE 15
COLOR 175 175 175
END
MAXSCALE 3000
END
CLASS
MINSCALE 3001
EXPRESSION 'A30' # Secondary Connecting Road
STYLE
  SYMBOL 'circle'
  SIZE 14
  COLOR 175 175 175
END
MAXSCALE 15000
END
CLASS
MINSCALE 15001
EXPRESSION 'A30' # Secondary Connecting Road
STYLE
  SYMBOL 'circle'
  SIZE 7
  COLOR 175 175 175
END
MAXSCALE 50000
END
CLASS
MINSCALE 50001
EXPRESSION 'A30' # Secondary Connecting Road
STYLE
  SYMBOL 'circle'
  SIZE 4
  COLOR 175 175 175
END
MAXSCALE 200000
END
```

```
CLASS
  MINSCALE 200001
  EXPRESSION 'A30' # Secondary Connecting Road
  STYLE
    SYMBOL 'circle'
    SIZE 3
    COLOR 175 175 175
  END
  MAXSCALE 600000
END
```

```
CLASS
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 15
    COLOR 175 175 175
  END
  MAXSCALE 3000
END
```

```
CLASS
  MINSCALE 3001
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 14
    COLOR 175 175 175
  END
  MAXSCALE 15000
END
```

```
CLASS
  MINSCALE 15001
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'circle'
```

```
    SIZE 7
    COLOR 175 175 175
  END
  MAXSCALE 50000
END
CLASS
  MINSCALE 50001
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 4
    COLOR 175 175 175
  END
  MAXSCALE 200000
END
CLASS
  MINSCALE 200001
  EXPRESSION 'A25' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 3
    COLOR 175 175 175
  END
  MAXSCALE 2000000
END
CLASS
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'circle'
    SIZE 15
    COLOR 175 175 175
  END
  MAXSCALE 3000
END
```

```
CLASS
  MINSCALE 3001
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'circle'
    SIZE 14
    COLOR 175 175 175
  END
  MAXSCALE 15000
END
CLASS
  MINSCALE 15001
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'circle'
    SIZE 7
    COLOR 175 175 175
  END
  MAXSCALE 50000
END
CLASS
  MINSCALE 50001
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'circle'
    SIZE 4
    COLOR 175 175 175
  END
  MAXSCALE 200000
END
CLASS
  MINSCALE 200001
  EXPRESSION 'A20' # US Highway
  STYLE
    SYMBOL 'circle'
```

```
SIZE 3
COLOR 175 175 175
END
#MAXSCALE 450000
END

END

#####
## Linework for Access Ramp A63
#####

LAYER
METADATA
  "ows_title" "ramp_fg"
  "gml_include_items" "all"
END
NAME 'ramp'
GROUP 'interstate_poly'
TILEINDEX 'TILEINDEX'
STATUS DEFAULT
TYPE LINE
CLASSITEM 'F_CLASS'
CLASS
  #NAME 'Access Ramp'
  EXPRESSION 'A63' # Access Ramp
  STYLE
    SYMBOL 'circle'
    SIZE 4
    COLOR 239 211 66
  END
  MAXSCALE 400000
END
END
```

```
#####  
## Linework for Local Road A40  
#####
```

```
LAYER  
METADATA  
  "ows_title" "local_fg"  
  "gml_include_items" "all"  
END  
NAME 'local'  
GROUP 'interstate_poly'  
TILEINDEX 'TILEINDEX'  
STATUS DEFAULT  
TYPE LINE  
CLASSITEM 'F_CLASS'  
CLASS  
  EXPRESSION 'A40' # Local, Rural Road  
  #NAME 'Local Roads'  
  STYLE  
    SYMBOL 'circle'  
    SIZE 12  
    COLOR 254 254 254  
  END  
  MAXSCALE 3000  
END  
CLASS  
  MINSCALE 3001  
  EXPRESSION 'A40' # Local, Rural Road  
  #NAME 'Local Roads'  
  STYLE  
    SYMBOL 'circle'  
    SIZE 10  
    COLOR 254 254 254  
  END  
  MAXSCALE 10000
```



```
END
CLASS
  MINSCALE 10001
  EXPRESSION 'A40' # Local, Rural Road
  #NAME 'Local Roads'
  STYLE
    SYMBOL 'circle'
    SIZE 3
    COLOR 254 254 254
  END
  MAXSCALE 15000
END
CLASS
  MINSCALE 15001
  EXPRESSION 'A40' # Local, Rural Road
  #NAME 'Local Roads'
  STYLE
    SYMBOL 'circle'
    SIZE 1
    COLOR 254 254 254
  END
  MAXSCALE 50000
END
END
```

```
#####
## Linework for Secondary Road A30
#####
```

```
LAYER
  METADATA
    "ows_title" "secondary_fg"
    "gml_include_items" "all"
  END
  NAME 'secondary'
```

```
GROUP 'interstate_poly'  
TILEINDEX 'TILEINDEX'  
STATUS DEFAULT  
TYPE LINE  
CLASSITEM 'F_CLASS'  
CLASS  
  #NAME 'County Road'  
  EXPRESSION 'A30' # Secondary Connecting Road  
  STYLE  
    SYMBOL 'circle'  
    SIZE 11  
    COLOR 255 251 115  
  END  
  MAXSCALE 3000  
END  
CLASS  
  MINSCALE 3001  
  #NAME 'County Road'  
  EXPRESSION 'A30' # Secondary Connecting Road  
  STYLE  
    SYMBOL 'circle'  
    SIZE 10  
    COLOR 255 251 115  
  END  
  MAXSCALE 15000  
END  
CLASS  
  MINSCALE 15001  
  #NAME 'County Road'  
  EXPRESSION 'A30' # Secondary Connecting Road  
  STYLE  
    SYMBOL 'circle'  
    SIZE 5  
    COLOR 255 251 115  
  END
```

```
MAXSCALE 50000
END
CLASS
  MINSCALE 50001
  #NAME 'County Road'
  EXPRESSION 'A30' # Secondary Connecting Road
  STYLE
    SYMBOL 'circle'
    SIZE 2
    COLOR 255 251 115
  END
  MAXSCALE 200000
END
CLASS
  MINSCALE 200001
  #NAME 'County Road'
  EXPRESSION 'A30' # Secondary Connecting Road
  STYLE
    SYMBOL 'circle'
    SIZE 1
    COLOR 255 251 115
  END
  MAXSCALE 600000
END
END
```

```
#####
## Linework for Minnesota Highway A25
#####
```

```
LAYER
METADATA
  "ows_title" "mn_highway_fg"
  "gml_include_items" "all"
END
```

```
NAME 'mnhighway'  
GROUP 'interstate_poly'  
TILEINDEX 'TILEINDEX'  
STATUS DEFAULT  
TYPE LINE  
CLASSITEM 'F_CLASS'  
CLASS  
  #NAME 'MN Highway'  
  EXPRESSION 'A25' # MN Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 11  
    COLOR 255 251 115  
  END  
  MAXSCALE 3000  
END  
CLASS  
  MINSCALE 3001  
  #NAME 'MN Highway'  
  EXPRESSION 'A25' # MN Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 10  
    COLOR 255 251 115  
  END  
  MAXSCALE 15000  
END  
CLASS  
  MINSCALE 15001  
  #NAME 'MN Highway'  
  EXPRESSION 'A25' # MN Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 5  
    COLOR 255 251 115
```

```
END
MAXSCALE 50000
END
CLASS
MINSCALE 50001
#NAME 'MN Highway'
EXPRESSION 'A25' # MN Highway
STYLE
  SYMBOL 'circle'
  SIZE 2
  COLOR 255 251 115
END
MAXSCALE 200000
END
CLASS
MINSCALE 200001
#NAME 'MN Highway'
EXPRESSION 'A25' # MN Highway
STYLE
  SYMBOL 'circle'
  SIZE 1
  COLOR 255 251 115
END
MAXSCALE 2000000
END
END
```

```
#####
## Linework for US Highways A20
#####
```

```
LAYER
METADATA
"ows_title" "us_highway_fg"
```

```
"gml_include_items" "all"  
END  
NAME 'ushighway'  
GROUP 'interstate_poly'  
TILEINDEX 'TILEINDEX'  
STATUS DEFAULT  
TYPE LINE  
CLASSITEM 'F_CLASS'  
CLASS  
  EXPRESSION 'A20' # US Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 11  
    COLOR 255 251 115  
  END  
  MAXSCALE 3000  
END  
CLASS  
  MINSCALE 3001  
  EXPRESSION 'A20' # US Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 10  
    COLOR 255 251 115  
  END  
  MAXSCALE 15000  
END  
CLASS  
  MINSCALE 15001  
  EXPRESSION 'A20' # US Highway  
  STYLE  
    SYMBOL 'circle'  
    SIZE 5  
    COLOR 255 251 115  
  END  
END
```

```
MAXSCALE 50000
END
CLASS
  MINSCALE 50001
  #NAME 'MN Highway'
  EXPRESSION 'A20' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 2
    COLOR 255 251 115
  END
  MAXSCALE 200000
END
CLASS
  MINSCALE 200001
  #NAME 'MN Highway'
  EXPRESSION 'A20' # MN Highway
  STYLE
    SYMBOL 'circle'
    SIZE 1
    COLOR 255 251 115
  END
END

END
```

```
#####
## Linework for US Interstates A10
#####
```

```
LAYER
METADATA
  "ows_title" "interstate_fg_bg"
  "gml_include_items" "all"
END
```

```
NAME 'interstate'
GROUP 'interstate_poly'
TILEINDEX 'TILEINDEX'
STATUS DEFAULT
TYPE LINE
CLASSITEM 'F_CLASS'
CLASS
  #NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
  EXPRESSION 'A10' # interstates
  STYLE
    SYMBOL 'circle'
    SIZE 14
    COLOR 175 175 175
  END ## BG
  STYLE
    SYMBOL 'circle'
    SIZE 12
    COLOR 247 190 33
  END
  MAXSCALE 20000
END
CLASS
  MINSCALE 20001
  #NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
  EXPRESSION 'A10' # interstates
  STYLE
    SYMBOL 'circle'
    SIZE 12
    COLOR 175 175 175
  END ## BG
  STYLE
    SYMBOL 'circle'
    SIZE 10
    COLOR 247 190 33
  END
```



```
MAXSCALE 25000
END
CLASS
MINSCALE 25001
#NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
EXPRESSION 'A10' # interstates
STYLE
  SYMBOL 'circle'
  SIZE 10
  COLOR 175 175 175
END ## BG
STYLE
  SYMBOL 'circle'
  SIZE 8
  COLOR 247 190 33
END
MAXSCALE 50000
END
CLASS
MINSCALE 50001
#NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
EXPRESSION 'A10' # interstates
STYLE
  SYMBOL 'circle'
  SIZE 7
  COLOR 175 175 175
END ## BG
STYLE
  SYMBOL 'circle'
  SIZE 5
  COLOR 247 190 33
END
MAXSCALE 200000
END
CLASS
```

```
MINSCALE 200001
#NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
EXPRESSION 'A10' # interstates
STYLE
  SYMBOL 'circle'
  SIZE 5
  COLOR 175 175 175
END ## BG
STYLE
  SYMBOL 'circle'
  SIZE 3
  COLOR 247 190 33
END
MAXSCALE 900000
END
CLASS
MINSCALE 900001
#NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
EXPRESSION 'A10' # interstates
STYLE
  SYMBOL 'circle'
  SIZE 4
  COLOR 175 175 175
END ## BG
STYLE
  SYMBOL 'circle'
  SIZE 2
  COLOR 247 190 33
END
MAXSCALE 4500000
END
CLASS
MINSCALE 4500001
#NAME 'US Interstate' ## Using Interstate Shield Symbol Instead
EXPRESSION 'A10' # interstates
```

```
STYLE
  SYMBOL 'circle'
  SIZE 2
  COLOR 247 190 33
END
END
```

END

#####

Legend Chips - This section used exclusively for Legend Generation

Basically these are duplicates of the Style entries
for the separate layers defined above.

##

There is still a potential for adding in the thresholding
to these legend Chip styles so that the Legend image stays
synced with the layers that are visible.

##

This may only be important to GeoMoose operators though,
since the legends are simply displayed and arranged automatically
in GeoMoose and the sizing of the image is compensated
for automatically.

#####

```
LAYER
  NAME 'legend_chips'
  GROUP 'interstate_poly'
  STATUS DEFAULT
  TYPE LINE
  CLASS
  NAME 'Access Ramp'
  STYLE
  SYMBOL 'circle'
  SIZE 4
```

```
    COLOR 175 175 175
  END
  STYLE
    SYMBOL 'circle'
    SIZE 2
    COLOR 239 211 66
  END
END
CLASS
  NAME 'US Highway'
  STYLE
    SYMBOL 'circle'
    SIZE 10
    COLOR 175 175 175
  END
  STYLE
    SYMBOL 'circle'
    SIZE 8
    COLOR 255 251 115
  END
END
CLASS
  NAME 'MN Highway'
  STYLE
    SYMBOL 'circle'
    SIZE 10
    COLOR 175 175 175
  END
  STYLE
    SYMBOL 'circle'
    SIZE 8
    COLOR 255 251 115
  END
END
CLASS
```

```
NAME 'Secondary Road'  
STYLE  
  SYMBOL 'circle'  
  SIZE 10  
  COLOR 175 175 175  
END  
STYLE  
  SYMBOL 'circle'  
  SIZE 8  
  COLOR 255 251 115  
END  
END  
CLASS  
  NAME 'Local Roads'  
  STYLE  
    SYMBOL 'circle'  
    SIZE 10  
    COLOR 175 175 175  
  END  
  STYLE  
    SYMBOL 'circle'  
    SIZE 8  
    COLOR 254 254 254  
  END  
END  
END
```

```
LAYER  
  NAME 'legend_chips'  
  GROUP 'interstate_poly'  
  STATUS DEFAULT  
  TYPE ANNOTATION  
  CLASS  
    NAME 'US Interstate'  
    STYLE
```

```
SYMBOL 'interstate1_wide_back'  
SIZE 16  
COLOR 255 0 0  
END  
STYLE  
  SYMBOL 'interstate1_wide_front'  
  SIZE 16  
  COLOR 115 113 206  
  OUTLINECOLOR 254 254 254  
END  
LABEL  
  TYPE TRUETYPE  
  FONT bluehigh  
  SIZE 5  
  COLOR 254 254 254  
  MINDISTANCE 200  
  MINFEATURESIZE 10  
  OFFSET 0 0  
  PARTIALS FALSE  
END  
END  
END  
END ## end Map
```