

Copyright Notice

Copyright 2003 *University of Minnesota*

The companies and organizations listed above have granted the Open GIS Consortium, Inc. (OGC) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version.

This document does not represent a commitment to implement any portion of this specification in any company's products.

OGC's Legal, IPR and Copyright Statements are found at <http://www.opengis.org/legal/ipr.htm>.

Permission to use, copy, and distribute this document in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the above list of copyright holders and the entire text of this NOTICE.

We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of OGC documents is granted pursuant to this license. However, if additional requirements (as documented in the Copyright FAQ at http://www.opengis.org/legal/ipr_faq.htm) are satisfied, the right to create modifications or derivatives is sometimes granted by the OGC to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Right in Technical Data and Computer Software Clause at DFARS 252.227.7013

OpenGIS® is a trademark or registered trademark of Open GIS Consortium, Inc. in the United States and in other countries.

Note: This document is not an OGC Standard. Internal and external documents cannot refer to it as such. Drafts are distributed for review and comment and are subject to change without notice.

University of Minnesota (UMN) MapServer

MapServer v3.5 implements WMS features. At the time this document was written, Mapserver supports the following WMS versions: 1.0.0, 1.0.7, and 1.1.0 (a.k.a. 1.0.8).

With MapServer, it is the "mapserv" CGI program that knows how to handle WMS requests. So setting up a WMS server with MapServer involves installing the mapserv CGI program and setting up a mapfile (files which define map object) with appropriate metadata in it.

Recipe A: UMN Mapserver HOWTO for Setting Up a WMS Server with Mapserver

Step 1: Preliminary Requirements

The following preliminary requirements are needed to run a WMS Server using UMN MapServer:

- MapServer should be installed, configured and running. To do so please follow the UMN MapServer online instructions¹.
- If you have a MapServer currently installed, check that your mapserv executable includes WMS support. One way to verify this is to use the "-v" command-line switch and look for "SUPPORTS=WMS".

On Unix:

```
$ ./mapserv -v
MapServer version 3.5 (pre-alpha)
OUTPUT=PNG OUTPUT=JPEG
OUTPUT=WBMP SUPPORTS=PROJ SUPPORTS=TTFF
SUPPORTS=WMS
INPUT=EPPL7 INPUT=JPEG INPUT=OGR
INPUT=GDAL INPUT=SHAPEFILE
```

On Windows:

```
C:\apache\cgi-bin> mapserv -v
MapServer version 3.5 (pre-alpha)
OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP SUPPORTS=PROJ
SUPPORTS=TTFF SUPPORTS=WMS
INPUT=EPPL7 INPUT=JPEG INPUT=OGR
INPUT=GDAL INPUT=SHAPEFILE
```

Step 2: Setup a Mapfile For Your WMS

Each instance of a WMS server that you setup needs to have its own mapfile. This is an ordinary MapServer mapfile in which some parameters and some metadata entries are mandatory. Most of the metadata is required in order to produce a valid GetCapabilities output.

¹ UMN Map Server homepage:<http://mapserver.gis.umn.edu/>

Figure 1, below, below shows an example fragment of a mapfile:

```
NAME DEMO
EXTENT 388013.643812817 5200395.13465842 500802.348432817 5313156.99196842
PROJECTION          # Project definition
  "init=epsg:26915"
END
LAYER              # Layer definitions
  NAME lakespy2
  CONNECTIONTYPE postgis
  CONNECTION "user=hdil2 dbname=mygisdb host=localhost port=5432"
  DATA "the_geom from lakespy2"
  TYPE POLYGON
  STATUS OFF
  CLASS
    COLOR 49 117 185
  END
  DUMP TRUE          # allow GML export
  METADATA
    WMS_TITLE "Lakes and Rivers"
    WMS_ABSTRACT "DLG lake and river polygons for Itasca County."
    WMS_SRS "EPSG:26915"
  END
END                # lakes
.....             # More layers definitions
```

Figure 1: Sample UMN MapServer Mapfile Fragment

Here is the list of parameters and metadata items that usually are optional with MapServer, but are required (or strongly recommended) for a WMS configuration:

At the map level:

- Map NAME & Map PROJECTION
- Map Metadata (in the WEB Object):
 - wms_title
 - wms_onlineresource
 - wms_srs (unless PROJECTION object is defined using "init=epsg:...")

And for each layer:

- Layer NAME & Layer PROJECTION
- Layer Metadata
 - wms_title
 - wms_srs (optional since the layers inherit the map's SRS value)

Below, each parameter is discussed in more detail:

- **Map NAME and wms_title:**

WMS Capabilities requires a Name and a Title tag for every layer. The Map's NAME and wms_title metadata will be used to set the root layer's name and title in the GetCapabilities XML output. The root layer in the WMS context corresponds to the whole mapfile.

- **Layer NAME and wms_title metadata:**

Every individual layer needs its own unique name and title. Layer names are also used in GetMap and GetFeatureInfo requests to refer to layers that should be included in the map output and in the query.

- **Map PROJECTION and wms_srs metadata:**

WMS servers have to advertise the projection in which they are able to serve data using EPSG projection codes². Recent versions of the PROJ4 library come with a table of EPSG initialization codes and allow users to define a projection like this:

```
PROJECTION
  "init=epsg:4269"
END
```

The above is sufficient for MapServer to recognize the EPSG code and include it in SRS tags in the capabilities output (`wms_srs` metadata is not required in this case). However, it is often impossible to find an EPSG code to match the projection of your data. In those cases, the `"wms_srs"` metadata is used to list one or more EPSG codes that the data can be served in, and the `PROJECTION` object contains the real PROJ4 definition of the data's projection.

Here is an example of a server whose data is in a Lambert Conformal Conic projection (for which there is no EPSG code). It's capabilities output will advertise EPSG:4269 and EPSG:4326 projections (lat/lon), but the `PROJECTION` object is set to the real projection that the data is in:

```
NAME DEMO
...
WEB
...
METADATA
  "wms_title"          "WMS Demo Server"
  "wms_onlineresource" "http://my.host.com/cgi-
bin/mapserv?map=wms.map&"
  "wms_srs"           "EPSG:4269 EPSG:4326"
END
END

PROJECTION
  "proj=lcc"
  "ellps=GRS80"
  "lat_0=49"
  "lon_0=-95"
  "lat_1=49"
  "lat_2=77"
END
...
```

- **Layer PROJECTION and wms_srs metadata:**

By default in the WMS context, layers inherit the SRS or their parent layer (the map's projection in the MapServer case). For this reason, it is not necessary (but still strongly recommended) to provide `PROJECTION` and `wms_srs` for every layer. However, if your server wants to advertise multiple projections, then at least a `PROJECTION` object is required in every layer, otherwise the layers won't be reprojected. This is the way on-the-fly reprojection works in MapServer. Layer `PROJECTION` and `wms_srs` metadata are defined exactly the same way as the map's `PROJECTION` and `wms_srs` metadata.

- **The "wms_onlineresource" metadata:**

² EPSG projection codes: <http://inovagis.dcea.fct.unl.pt/gisserver/epsg.asp>.

The `wms_onlineresource` metadata is set in the map's Web object metadata and specifies the URL that should be used to access your server. This is required for the `GetCapabilities` output. If `wms_onlineresource` is not provided, then MapServer will try to provide a default one using the script name and hostname, but this is not completely reliable. It is strongly recommended that you provide the `wms_onlineresource` metadata.

See the WMS Specification for the whole story about the online resource URL. Basically, what you need is a complete HTTP URL including the `http://` prefix, hostname, script name, potentially a `"map="` parameter, and terminated by `"?"` or `"&"`.

Here is a valid online resource URL:

```
http://my.host.com/cgi-bin/mapserv?map=mywms.map&
```

By creating a wrapper script on the server, it is possible to hide the `"map="` parameter from the URL and then your server's online resource URL could be something like:

```
http://my.host.com/cgi-bin/mywms?
```

This is covered in more detail in Section 0 below, "More About the Online Resource URL."

Step 3: Test Your WMS Server

Validate the Capabilities Metadata

With the mapfile in place, you have to check the XML capabilities returned by your server to make sure nothing is missing. Using a Web browser, access your server's online resource URL to which you add the parameter `"REQUEST=GetCapabilities"` to the end, e.g.:

```
http://my.host.com/cgi-bin/mapserv?map=mywms.map&
REQUEST=GetCapabilities
```

This should return a document of MIME type `application/vnd.ogc.wms_xml`, so your Web browser is likely to prompt you to save the file. Save it and open it in a text editor (Emacs, Notepad, etc.)

If you get an error message in the XML output, then take necessary actions. Common problems and solutions are the following:

Q: How can I find the EPSG code for my data's projection?

A: If you know the parameters of your data's projection, then you can browse the `"epsg"` file that comes with PROJ4 and look for a projection definition that matches your data's projection. It's a simple text file and the EPSG code is inside brackets (`< . . . >`) at the beginning of every line.

Q: My WMS server produces the error `"msProcessProjection(): no system list, errno: .."`

A: That's likely PROJ4 complaining that it cannot find the `"epsg"` projection definition file. Make sure you have installed PROJ 4.4.3 or more recent and that the `"epsg"` file is installed at the right location. On Unix it should be under `/usr/local/share/proj/`, and on Windows PROJ looks for it under `C:\PROJ\`.

If everything went well, you should have a complete XML capabilities document. Search it for the word `"WARNING"`... MapServer inserts XML comments starting with `"<!--WARNING: "` in the XML output if it detects missing mapfile parameters or metadata items. If you notice any warning in your XML output, then you have to fix all of them before you can register your server with a WMS client, otherwise you will probably run into problems.

Test With a GetMap Request

Knowing that the server can produce a valid XML `GetCapabilities` response, you can now test the `GetMap` request.

Simply adding "`VERSION=1.1.0&REQUEST=GetMap`" to your server's URL should generate a map with the default map size and all the layers with `STATUS ON` or `DEFAULT` displayed, e.g.

```
http://my.host.com/cgi-bin/mapserv?map=mywms.map&VERSION=1.1.0&
REQUEST=GetMap
```

Note: this works with MapServer's WMS interface even if it would be an incomplete `GetMap` request according to the WMS spec. It lacks several mandatory parameters such as `SRS`, `BBOX`, etc, but it is good enough for testing at this point..

Test With a Real Web Client

If you have access to a WMS client, then register your new server's online resource with it and you're running. If you don't have your own WMS client, then CubeWerx's WMS interface (`cubeview`) can be useful to test a new server. Access the following page:

```
http://www.cubewerx.com/demo/cubeview/cubeview.cgi
```

and enter your server's URL at the bottom of the page and click "GO". The default set of layers in the interface should be replaced with your server's layers.

More About the Online Resource URL

As mentioned in Section 0 "Setup a Mapfile for Your Web Map Service" above, the following Online Resource URL is perfectly valid for a MapServer WMS according to section 6.2.1 of the WMS 1.1.0 specification:

```
http://my.host.com/cgi-bin/mapserv?map=mywms.map&
```

Some people will argue that the above URL contains mandatory vendor-specific parameters and that this is illegal. First we would like to point out that "`map=...`" is not considered a vendor-specific parameter in this case since it is part of the Online Resource URL which is defined as an opaque string terminated by "?" or "&".

However, even if it's valid, the above URL is still ugly and you might want to use a nicer URL for your WMS Online Resource URL. Here are some suggestions; some of them will work only on Unix, but at least #2 will work for Windows/Apache users as well.

On Unix servers, you can setup a wrapper shell script that sets the `MS_MAPFILE` environment variable and then passes control to the `mapserv` executable that results in a cleaner OnlineResource URL:

```
#!/bin/sh
MS_MAPFILE=/path/to/demo.map
export MS_MAPFILE
/path/to/mapserv
```

Another option is to use the "`setenvif`" feature of Apache: use symbolic links that all point to a same `mapserv` binary, and then for each symbolic link, test the URL, and set the `MAP` environment accordingly. For Windows and Apache users, the steps are as follows (this requires Apache 1.3 or newer):

- Copy `mapserv.exe` to a new name for your WMS, such as "`mywms.exe`".
- In `httpd.conf`, add:

```
SetEnvIf Request_URI "/cgi-bin/mywms"  
MS_MAPFILE=/path/to/mymap.map
```

Recipe B: UMN MapServer HOW TO for Setting Up a WMS Client with Mapserver

Step 1: Compilation / Installation

The WMS connection type is enabled by the `--with-wmsclient` configure switch. It requires PROJ4 and W3C's libwww (libwww v5.3.2 or newer. This is required because there was a bug that caused an infinite loop in older versions), and GDAL is optional (see below).

- For PROJ4 and GDAL installation, see the MapServer Compilation HOWTO: UNIX³ or Win32⁴
- For W3C's libwww, download v5.3.2 (or newer) from CVS or in a tarball and compile/install manually⁵.

You might want to also include GDAL support if you want your application to be able to reproject map slides received from remote servers. This is because raster resampling works only when used with the GDAL library in MapServer. If GDAL is not included in your MapServer build, then your application can only serve maps in the subset of the projections supported by all the remote servers (this should be sufficient for most applications). If you compile with GDAL, then make sure that your GDAL includes GIF and/or PNG support, depending on which image format you request from remote servers.

Once the required libraries are installed, then configure MapServer using the `--with-wms client` switch (plus all the other switches you used to use) and recompile.

This will give you a new set of executables (and possibly `php_mapscript` if you requested it). See the *MapServer Compilation HOWTO* for installation details.

Step 2: Check your MapServer executable

To check that your `mapserv` executable includes WMS support, use the `"-v"` command-line switch and look for "SUPPORTS=WMS_CLIENT".

Example 1. On Unix:

```
$ ./mapserv -v  
MapServer version 3.5 (pre-alpha)  
OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP  
SUPPORTS=PROJ SUPPORTS=TTF  
SUPPORTS=WMS_CLIENT INPUT=EPPL7 INPUT=JPEG  
INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Example 2. On Windows:

```
C:\apache\cgi-bin> mapserv -v
```

³ UMN Map Server Compilation HOWTO (for UNIX) <http://mapserver.gis.umn.edu/doc/unix-install-howto.html>

⁴ UMN Map Server Compilation HOWTO (for Win32) <http://mapserver.gis.umn.edu/doc/win32compile-howto.htm>

⁵ W3C's libwww: <http://www.w3.org/Library/>

```
MapServer version 3.5 (pre-alpha)
OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP
SUPPORTS=PROJ SUPPORTS=TTF SUPPORTS=WMS_CLIENT
INPUT=EPPL7 INPUT=JPEG INPUT=OGR
INPUT=GDAL INPUT=SHAPEFILE
```

Step 3: MapFile configuration - CONNECTIONTYPE WMS

A PROJECTION must be set in the mapfile for the MAP unless you are sure that all your WMS layers support only a single projection which is the same as the PROJECTION of the map. The MAP PROJECTION can be set using "init=epsg:xxxx" codes or using regular PROJ4 parameters. Failure to set a MAP PROJECTION may result in blank maps coming from remote WMS servers (because of inconsistent BBOX+SRS combination being used in the WMS connection URL).

WMS layers are accessed via the WMS connection type. Here is an example of a layer using this connection type:

```
LAYER
  NAME bathymetry
  METADATA
    "wms_title" "Elevation/Bathymetry"
    "wms_srs" "EPSG:42304 EPSG:4269 EPSG:4326"
  END
  TYPE RASTER
  STATUS ON
  CONNECTIONTYPE WMS
  CONNECTION "http://www2.dmsolutions.ca:8099/cgi-
bin/mswms_gmap?VERSION=1.1.0&LAYERS=bathymetry&FORMAT=image/png"
  PROJECTION
    "init=epsg:42304"
  END
END
```

The following items are required for the WMS connection type:

- "wms_srs" metadata: a space-delimited list of EPSG projection codes supported by the remote server. You normally get this from the server's capabilities output.
- CONNECTIONTYPE WMS: of course!
- CONNECTION parameter: This is the remote server's online resource URL to which you append some of the getMap/getFeatureInfo request parameters. At this point MapServer sets only the following request parameters:

```
REQUEST
SRS
BBOX
WIDTH
HEIGHT
```

The connection string should contain all other required params, including:

```
VERSION
LAYERS
```

```
FORMAT
TRANSPARENT
```

The following layer parameters are optional:

- `PROJECTION` object: This is optional at this point. MapServer will create one internally if needed.
- `MINSCALE`, `MAXSCALE`: If the remote server's capabilities contain a `ScaleHint` value for this layer, then you might want to set the `MINSCALE` and `MAXSCALE` in the `LAYER` object in the mapfile. This will allow MapServer to request the layer only at scales where it makes sense.
- `"wms_latlonboundingbox"` metadata: The bounding box of this layer in geographic coordinates in the format `"lon_min lat_min lon_max lat_max"`. If it is set, then MapServer will request the layer only when the map view overlaps that bounding box. You normally get this from the server's capabilities output.

e.g.

```
METADATA
"wms_latlonboundingbox" "-124 48 -123 49"
END
```

- `"wms_connectiontimeout"` metadata: The maximum time to load a remote WMS layer, set in seconds (default is 30 seconds). This metadata can be added at the layer level so that it affects only that layer, or it can be added at the map level (in the web object) so that it affects all of the layers. Note that `wms_connectiontimeout` at the layer level has priority over the map level.

...

```
CONNECTIONTYPE WMS
METADATA
"wms_srs" "EPSG:4269 EPSG:32182"
"legend_icon" "Icons/none.gif"
"wms_title" "NF Water (Lakes) (a)"
"wms_boundingbox" "EPSG:32182 42708.6 5.27438e+06
5270155.72117e+06"
"wms_latlonboundingbox" "-59.7807 47.5557 -52.7934
51.6261"
"wms_connectiontimeout" "60"
...
END
...
```

In addition to the above layer parameters, you have to set the `IMAGEPATH` value in the `WEB` object of your mapfile to point to a valid and writable directory. MapServer will use this directory to store temporary files downloaded from the remote servers. The temporary files are automatically deleted by MapServer so you won't notice them... but a valid `IMAGEPATH` is still required.

Limitations/To Do

1. MapServer WMS connections always request exceptions `"INIMAGE"`, but it may be nice to support XML exceptions at some point and return the message via the MapServer error reporting mechanisms.

2. PNG format with `transparent=true` does not work well with all servers. For instance, some servers use the alpha channel for transparency (RGBA images), but this is not well supported by MapServer at the moment, so you may be forced to request GIF format maps in those cases. Transparent PNGs produced by the MapServer WMS work well though, since they are platted images.
3. `GetFeatureInfo` is not fully supported yet since the output of `GetFeatureInfo` is left to the discretion of the remote server. A method `layer.getWMSFeatureInfoURL()` has been added to MapScript for applications that want to access `featureInfo` results and handle them directly.
4. MapServer does not attempt to fetch the layer's capabilities. Doing so at every map draw would be extremely inefficient, and caching that information does not belong in the core of MapServer. This is better done at the application level, in a script, and only the necessary information is passed to the MapServer core via the `CONNECTION` string and metadata.

Note: DM Solutions will soon release the MapBrowser PHP application which demonstrates the use of PHP to fetch and parse remote WMS server capabilities.

Additional Information: Using ArcExplorer 4 as a Client for WMS-compliant UMN Mapserver

ArcExplorer is a freely downloadable GIS viewer offered by ESRI. With version 4, ArcExplorer has an extension that supports the WMS and WFS specifications. Because UMN Mapserver also supports these two standards, you can use ArcExplorer as a client application. This means you need not be limited by the event/response model commonly seen in Web browser based viewers, or worry about writing any client side code (such as Javascript) that may be difficult to support on a variety of browsers. As a plus, ArcExplorer is written in Java and will run on Windows, Linux, Solaris, MacOS X, Irix, HP-UX and AIX. Besides, it's a good example of how the adoption of standards can greatly increase the options available to GIS providers and users. This example also shows how to use PHP to modify responses and make Web clients WMS 1.1.1 compliant. For setup information see the following URL:

<http://mapserver.gis.umn.edu/cgi-bin/wiki.pl?WMSMapserverArcExplorer>