WMS Server

Author: Jeff McKenna
Contact: jmckenna at gatewaygeomatics.com
Last Updated: 2022-09-05

Table of Contents

Introduction

A WMS (or Web Map Server) allows for use of data from several different servers, and enables for the creation of a network of Map Servers from which clients can build customized maps. The following documentation is based on the Open Geospatial Consortium's (OGC) Web Map Server Interfaces Implementation Specification v1.1.1.

MapServer v3.5 or more recent is required to implement WMS features. At the time this document was written, MapServer supports the following WMS versions: 1.0.0, 1.0.7, 1.1.0 (a.k.a. 1.0.8), 1.1.1 and 1.3.0.

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

- MapServer application development and setting up .map files.
- Familiarity with the WMS spec would be an asset. A link to the WMS specification document is included in the "WMS-Related Information" section below.

Links to WMS-Related Information

- MapServer WMS Client Howto
- WMS 1.3.0 specification
- WMS 1.1.1 specification
- OGC's WMS homepage
- WMS Cookbook
- MapServer OGC Web Services Workshop package
- MapServer Styled Layer Descriptor (SLD) Howto
- MapServer WMS Time Support Howto

How does a WMS Work

WMS servers interact with their clients via the HTTP protocol. In most cases, a WMS server is a CGI program. This is also the case with MapServer.

The WMS specification defines a number of request types, and for each of them a set of query parameters and associated behaviors. A WMS-compliant server MUST be able to handle at least the following 2 types of WMS requests:

1. **GetCapabilities:** return an XML document with metadata of the Web Map Server's information
2. **GetMap:** return an image of a map according to the user's needs.

And support for the following types is optional:

1. **GetFeatureInfo:** return info about feature(s) at a query (mouse click) location. MapServer supports 3 types of responses to this request:
    - text/plain output with attribute info.
    - text/html output using MapServer query templates (see Templating) specified in the CLASS TEMPLATE parameter (the filename has to have an .html extension). The MIME type returned by the Class templates defaults to text/html and can be controlled using the metadata "wms_feature_info_mime_type".
    - application/vnd.ogc.gml, GML.1 or GML for GML features.
2. **DescribeLayer:** return an XML description of one or more map layers. To execute this:
    - for vector layers: to have a valid return the user needs to setup wfs_onlineresource (or ows_onlineresource) metadata either at the map level or at the layer level (the layer level metadata is the one which is used if both are defined) - for raster layers: the metadata is wcs_onlineresource with the same logic as above.
3. **GetLegendGraphic:** returns a legend image (icon) for the requested layer, with label(s). More information on this request can be found in the GetLegendGraphic section later in this doc.

With respect to MapServer specifically, 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 a setting up a mapfile with appropriate metadata in it. This is covered in the rest of this document.

Setting Up a WMS Server Using MapServer

Install the Required Software

WMS requests are handled by the mapserv CGI program. Not all versions of the mapserv program do include WMS support (it is included by default when you compile together with the PROJ library), so the first step is to 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_SERVER".

(Unix users should refer to the Compiling on Unix document for any compiling instructions, and Windows users might want to use MS4W, which comes ready with WMS/WFS support)

Example 1. On Unix:

```
$ ./mapserv -v
MapServer version 7.7-dev OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=PROJ
SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=CAIRO SUPPORTS=SVG_SYMBOLS
SUPPORTS=RSVG SUPPORTS=ICONV SUPPORTS=FRIBIDI SUPPORTS=WMS_SERVER
SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT
SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER SUPPORTS=GEOS SUPPORTS=PBF
INPUT=JPEG INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Example 2. On Windows:

```
C:\ms4w> mapserv -v
MapServer version 7.7.0-dev (MS4W 4.0.5) OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML
SUPPORTS=PROJ SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=CAIRO
SUPPORTS=SVG_SYMBOLS SUPPORTS=SVGCAIRO SUPPORTS=ICONV SUPPORTS=FRIBIDI
SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER
SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER
SUPPORTS=FASTCGI SUPPORTS=THREADS SUPPORTS=GEOS SUPPORTS=POINT_Z_M
SUPPORTS=PBF INPUT=JPEG INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Setup a Mapfile For Your WMS

Each instance of WMS server that you setup needs to have its own mapfile. It is just a regular 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.

Here is the list of parameters and metadata items that usually 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:…")
    - wms_enable_request

And for each LAYER:

- Layer NAME
- Layer PROJECTION
- Layer METADATA
  - wms_title
  - wms_srs (optional since the layers inherit the map's SRS value)
- Layer STATUS
  - Layers set to STATUS DEFAULT will always be sent to the client.
  - Layers set to STATUS ON or STATUS OFF can be requested by the client.
- Layer TEMPLATE (required for GetFeatureInfo requests - see Templating)

Let's go through each of these parameters 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. You can suppress the root layer's name by setting `wms_rootlayer_name` to `""`.

- **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. Layer names must start with a letter when setting up a WMS server (layer names should not start with a digit or have spaces in them).

- **Map PROJECTION and wms_srs metadata:**

  WMS servers have to advertise the projection in which they are able to serve data using EPSG projection codes (see The EPSG web page for more back-ground on EPSG codes). The PROJ library come with a table of EPSG initialization codes and allows users to define a projection like this:

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

  (Note that "epsg" has to be in lowercase when used in the PROJ 'init' directive.)

  If the MAP PROJECTION block is provided in the format "init=epsg:xxxx" then MapServer will also use this information to generate a <BoundingBox> tag for the top-level layer in the WMS capabilities document. BoundingBox is a mandatory element of WMS capabilities for WMS 1.3.0 (for WMS 1.1.0 it is optional, but it is good practice to allow MapServer to include it when possible).

  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 PROJ definition of the data's projection.

  Here is an example of a server whose data is in a Lambert Conformal Conic projection (3978). Its capabilities output will advertise EPSG:3857 (web mercator), 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=/path/to/your-mapfile.map" #must change mapfile p
      "wms_srs"             "EPSG:3857 EPSG:4269 EPSG:4326"
    END
  END

  PROJECTION
    "init=epsg:3978"
  END
  ...
  END
  ```

  In addition to EPSG:xxxx projections, a WMS server can advertise projections in the AUTO:xxxx namespace. AUTO projections 42001 to 42005 are internally supported by MapServer. However, AUTO projections are useful only with smart WMS clients, since the client needs to define the projection parameters in the WMS requests to the server. For more information see Annex E of the WMS 1.1.1 specification and section 6.5.5.2 of the same document. See also the FAQ on AUTO projections at the end of this document.

- **Layer PROJECTION and wms_srs metadata:**

  By default layers inherit the SRS of 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. If a layer PROJECTION is not provided then the top-level map projection will be assumed.

  Layer PROJECTION and wms_srs metadata are defined exactly the same way as the map's PROJECTION and wms_srs metadata.

For vector layers, if a PROJECTION block is provided in the format “init=epsg:xxxx” then MapServer will also use this information to generate a <BoundingBox> tag for this layer in the WMS capabilities document. BoundingBox is a mandatory element of WMS capabilities for WMS 1.3.0 (for WMS 1.1.0 it is optional, but it is good practice to allow MapServer to include it when possible).

- **“wms_onlineresource” metadata:**

  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 you shouldn’t count on that too much. It is strongly recommended that you provide the wms_onlineresource metadata.

  See section 6.2.2 of the WMS 1.1.1 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=/path/to/your-mapfile.map" #must change mapfile path
```

  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 the section “More About the Online Resource URL” below.

- **“wms_enable_request” metadata:**

  Specify which requests to enable. If not specified, no requests will be enabled! See the explanation below.

- **Configuring for GetFeatureInfo Requests:**

  You must set the layer TEMPLATE parameter for the layer to be queryable by GetFeatureInfo requests (see Templating), with a value such as *“empty”* or the name of your html template file. For requests of type “text/html” you should also set the layer HEADER and FOOTER parameters.

```
MAP
  ...
  LAYER
    ...
    TEMPLATE "empty"
    ...
  END #layer
END #map
```

  As of MapServer 4.6 you must set the *gml_** metadata for the layer attributes to be served (see the Layer Object metadata in the Reference Section later in this document). To include geometry, gml_geometries and gml_[geometry name]_type has to be specified.

  Here are working examples of GetFeatureInfo requests: text/plain / text/html / gml (for gml, your browser might ask you to save the file, if so save it locally as a .gml file and view it in a text editor)

Test Your WMS Server

Validate the Capabilities Metadata

OK, now that we’ve got a mapfile, we have to check the XML capabilities returned by our server to make sure nothing is missing.

Using a web browser, access your server’s online resource URL to which you add the parameters “SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities” to the end, e.g.

```
http://my.host.com/cgi-bin/mapserv?map=/path/to/your-mapfile.map&SERVICE=WMS&VERSION=1.1.1
                                          &REQUEST=GetCapabilities
```

Here is a working GetCapabilities request (note that the SERVICE parameter is required for all GetCapabilities requests):

https://demo.mapserver.org/cgi-bin/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities

This should return a document of MIME type application/vnd.ogc.wms_xml, so your browser is likely going to prompt you to save the file. Save it and open it in a text editor (Emacs, Notepad, etc.), and you will see the returned XML from the WMS server.

If you get an error message in the XML output then take necessary actions. Common problems and solutions are listed in the FAQ at the end of this document.

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 things are likely not going to work.

Note that when a request happens, it is passed through WMS, WFS, and WCS in MapServer (in that order) until one of the services respond to it.

Test With a GetMap Request

OK, now that we know that our server can produce a valid XML GetCapabilities response we should test the GetMap request. MapServer only checks for a few of the required GetMap parameters, so both of the minimum MapServer parameters and a valid GetMap request will be explained below.

The following is a list of the required GetMap parameters according to the WMS spec:

VERSION=version: Request version

REQUEST=GetMap: Request name

LAYERS=layer_list: Comma-separated list of one or more map layers. Optional if SLD parameter is present.

STYLES=style_list: Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present. Set "STYLES=" with an empty value to use default style(s). Named styles are also supported and are controlled by CLASS GROUP names in the mapfile.

SRS=namespace:identifier: Spatial Reference System.

BBOX=minx,miny,maxx,maxy: Bounding box corners (lower left, upper right) in SRS units.

WIDTH=output_width: Width in pixels of map picture.

HEIGHT=output_height: Height in pixels of map picture.

FORMAT=output_format: Output format of map.

Note: WMS Servers only advertise supported formats that are part of the gd / gdal libraries.

A valid example would therefore be:

```
http://my.host.com/cgi-bin/mapserv?
     map=/path/to/your-mapfile.map
     &SERVICE=WMS&VERSION=1.1.1
     &REQUEST=GetMap&LAYERS=prov_bound&STYLES=&SRS=EPSG:4326
     &BBOX=-173.537,35.8775,-11.9603,83.8009&WIDTH=400&HEIGHT=300
     &FORMAT=image/png
```

Here is a working valid request.

Tip: Since MapServer 8.0, the *tile_map_edge_buffer* metadata can also be used in WMS GetMap requests, by adding the vendor-specific parameter *TILED=TRUE* to your GetMap request.

See also: Tile Mode

Test with a Real Client

If you have access to a WMS client, then register your new server's online resource with it and you should be off and running.

If you don't have your own WMS client installed already, here are a few pointers:

- MapServer itself can be used as a WMS client, see the MapServer WMS Client Howto.
- QGIS is a full GIS package which includes WMS client support. (recommended)

  Tip: You can/should use the QGIS Network Logger to get the exact GetMap or GetCapabilities requests sent by QGIS.

- OpenJUMP is a desktop GIS package which includes WMS client support.
- uDig is a desktop package that allows users to add WMS layers.
- Deegree provides a WMS client.

This list is not exhaustive, there are several Open Source or proprietary packages that offer WMS support and could be used to interact with your new MapServer WMS server instance.

GetLegendGraphic Request

This request returns a legend image (icon) for the specified layer. The request will draw an icon and a label for all classes defined on the layer. If the requested layer-name is a GROUP-name, all included layers will be returned in the legend-icon.

Requirements

The following are required in the WMS server mapfile to enable this GetLegendGraphic request:

- a LEGEND object.
- a CLASS object for each layer.
- a NAME in the CLASS object.
- the STATUS of each LAYER must be set to ON.

Parameters

The following are valid parameters for this GetLegendGraphic request:

- **LAYER** - (Required) Name of the WMS layer to return the legend image of. Note that this is the <Name> parameter of the Layer in the GetCapabilities.
- **FORMAT** - (Required) Format of the legend image (e.g. “image/png”).
- **WIDTH** - (Optional) Width of the legend image. Note that the Width parameter is only used when the Rule parameter is also used in the request.
- **HEIGHT** - (Optional) Height of the legend image. Note that the Height parameter is only used when the Rule parameter is also used in the request.
- **SLD** - (Optional) The URL to the SLD. Applies the SLD on the layer and the legend is drawn after the SLD is applied (using the classes specified by the SLD). Note here that you need to put a <Name>class1</Name> inside the Rule element so that a class name is created from the SLD and therefore a correct legend image.
- **SLD_BODY** - (Optional) The body (code) of the SLD, instead of specifying a URL (as in the ‘SLD’ parameter).
- **SLD_VERSION** - (Optional) The SLD version.
- **SCALE** - (Optional) Specify a scale so that only layers that fall into that scale will have a legend.
- **STYLE** - (Optional) The style.
- **RULE** - (Optional) Specify the name of the CLASS to generate the legend image for (as opposed to generating an icon and label for ALL classes for the layer).

Note: All rules that are used to draw the legend in normal CGI mode apply here. See the CGI Reference doc if necessary.

The CLASS object’s KEYIMAGE parameter can also be used to specify a legend image for a CLASS. See the MapFile Reference doc if necessary. Example Request

An example request might look like:

```
http://127.0.0.1/cgi-bin/mapserv.exe?map=/path/to/your-mapfile.map&SERVICE=WMS&VERSION=1.1.1&layer=park&
REQUEST=getlegendgraphic&FORMAT=image/png
```

Content Dependent Legends

Support for content dependent legends is supported (since the *MapServer 6.4* release) for WMS GetLegendGraphic calls on POINT, LINE and POLYGON layers, when the client includes the (non-standard) BBOX, SRS/CRS and WIDTH/HEIGHT parameters in the request URL (such as a GetMap request). If these parameters are not present, the normal legend rendering will occur. For more information, please see MS RFC 101: Add Support for Content Dependent Legend Responses.

Changing the Online Resource URL

As mentioned in the section “Setup a Mapfile / wms_onlineresource metadata” above, the following Online Resource URL is perfectly valid for a MapServer WMS according to section 6.2.2 or the WMS 1.1.1 specification:

```
http://my.host.com/cgi-bin/mapserv?map=/path/to/your-mapfile.map
```

However, some people will argue that the above URL contains mandatory vendor-specific parameters and that this is illegal. First we would like to point 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 “&” (See WMS 1.1.1 section 6.2.2).

Note: It is strongly recommended to review the security steps for the *MAP*= call to the MapServer executable, by setting *MS_MAP_PATTERN* or *MS_MAP_NO_PATH* or hiding the *MAP*= parameter on public servers, as recommended in the document Limit Mapfile Access. All possible environment variables to secure your server are listed in Environment Variables.

But anyway, 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:

Apache ReWrite rules (using Apache mod_rewrite)

One can use Apache’s mod_rewrite to avoid specifying the map, or any other default parameter in the mapserver URL. This task consist of three steps, specifying the mod_rewrite module to be loaded, enabling the mod_rewrite module for the selected directories and at last to write a .htaccess file to do the rewriting.

In the httpd.conf file, the mod_rewrite module is per default disabled. To enable it, remove the opening # in the line

```
#LoadModule rewrite_module modules/mod_rewrite.so
```

To be able to use the module, it must be enabled, using the directive AllowOverride. This can be done per server or per directory. If you just have one server, add an “AllowOverride All” line in the httpd.conf file (see the Apache documentation to be sure about the security implications of this). Per directory is the easiest way to make it work on virtual hosts. Within the <virtualHost> section of the httpd.conf insert:

```
<Directory myhtdocsdir>
  AllowOverride All
</Directory>
```

Where myhtdocsdir is the directory defined as documentroot for the actual virtual server.

When the directives are set to load and enable the mod_rewrite module, Apache has to be restarted.

In a web-accessible directory make a .htaccess file like the following:

```
RewriteEngine on
RewriteRule wmsmap?(.*) /cgi-bin/mapserv?map=/home/www/mapserverstuff/mymapfile.map&$1
```

The rewriteRule says: given a webpage starting with wmsmap, pick out the query parameters, make a new page request starting with /cgi-bin/mapserv?map=(…)? and add on whatever was the query parameters in the original page request.

e.g, the URL wmsmap?mode=map will be rewritten as

::

    /cgi-bin/mapserv?map=/home/www/mapserverstuff/mymapfile.map&mode=map

If just the URL wmsmap is given (without any parameters) a page not found error will show up as that does not match the wmsmap? expression.

Apache environment variables - MS_MAPFILE

A default mapfile can be specified using the MS_MAPFILE environment variable:

```
Alias /mywms /usr/lib/cgi-bin/mapserver
<Location /mywms>
   SetHandler cgi-script
   Options ExecCGI
   SetEnv MS_MAPFILE /path/to/mymapfile.map
</Location>
```

Apache SetEnvIf

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
```

ASP script (IIS - Microsoft Windows)

On IIS servers (Windows), you can use the following ASP script:

Note: The script below, while functional, is intended only as an example of using ASP to filter MapServer requests. Using ASP in a production WMS server will likely require additional ASP especially in the area of error handling and setting timeouts.*

```
<%
  Server.ScriptTimeout = 360

  Select Case Request.ServerVariables("REQUEST_METHOD")
    Case "GET"  strRequest = Request.QueryString
    Case "POST" strRequest = Request.Form
  End Select

  strURL = "http://myserver/cgi-bin/mapserv.exe?
            map=C:\Inetpub\wwwroot\workshop\itasca.map&" & strRequest

  Dim objHTTP
  Set objHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP")
  objHTTP.open "GET", strURL, false
  objHTTP.send ""

  Response.ContentType = objHTTP.getResponseHeader("content-type")
  Response.BinaryWrite objHTTP.responseBody

  Set objHTTP = Nothing
%>
```

Mapscript wrapper

Some OGC services (WFS, SOS) support both GET and POST requests. Here, you can use a minimal MapScript WxS wrapper. Here’s a Python example:

```
#!/usr/bin/python

import mapscript

req = mapscript.OWSRequest()
req.loadParams()
```

```
map = mapscript.mapObj('/path/to/config.map')
map.OWSDispatch(req)
```

Wrapper script (Unix)

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. The script will usually live inside your */cgi-bin/* directory. The result is a cleaner OnlineResource URL:

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

Note: Using a /bin/sh wrapper script causes an overhead on system resources as two processes have to be spawned instead of one, and is therefore not recommended.

WMS 1.3.0 Support

MapServer 5.4 adds support for WMS 1.3.0. Although the general mechanism in MapServer to support this new specification are the same, there are some notable upgrades.

Major features related to the WMS 1.3.0 support

- Support WMS 1.3.0 basic operations: GetCapabilities, GetMap and GetFeatureInfo.
- Implement the Styled Layer Descriptor profile of the Web Map Service Implementation Specification. This specification extends the WMS 1.3.0 and allows to advertise styling capabilities (Styled Layer Descriptor (SLD) support). It also defines two additional operations, GetLegendGraphic and DescribeLayer.
- Implement the Symbology Encoding Implementation Specification, which is the new version of the SLD. Read support was added for Point, Line, Polygon, Raster symbolizers.
- Upgrade the generation of SLD to version 1.1.0 (SLD generated through through the GetStyles operation or through MapScript).

Coordinate Systems and Axis Orientation

The most notable changes introduced in WMS 1.3.0 are the:

- the axis changes
- the introduction of new coordinate reference systems
- the use of CRS parameter (instead of SRS)

The axis order in previous versions of the WMS specifications was to always use easting (x or lon) and northing (y or lat). WMS 1.3.0 specifies that, depending on the particular CRS, the x axis may or may not be oriented West-to-East, and the y axis may or may not be oriented South-to-North. The WMS portrayal operation shall account for axis order. This affects some of the EPSG codes that were commonly used such as ESPG:4326. MapServer 5.x makes sure that coordinates passed to the server (as part of the GetMap BBOX parameter) as well as those advertised in the capabilities document reflect the inverse axe orders for EPSG codes between 4000 and 5000.

MapServer 6.0 and up holds a list of EPSG codes with inverted axis order. It is currently based on EPSG database version 7.6. It is also possible to define the axis order at build time for a specific ESPG code (see issue #3582). This allows for example to use the “normal” axis order for some of EPSG codes between 4000 and 5000.

In addition, the WMS 1.3.0 defines a series of new coordinate system. These are the ones that are currently supported in MapServer:

- CRS:84 (WGS 84 longitude-latitude)
- CRS:83 (NAD83 longitude-latitude)
- CRS:27 (NAD27 longitude-latitude)
- AUTO2:42001 (WGS 84 / Auto UTM)
- AUTO2:42002 (WGS 84 / Auto Tr. Mercator)
- AUTO2:42003 (WGS 84 / Auto Orthographic)
- AUTO2:42004 (WGS 84 / Auto Equirectangular)
- AUTO2:42005 (WGS 84 / Auto Mollweide)

Example of requests

Users can use the CRS:84 coordinate system and order the BBOX coordinates as long/lat:

- …&CRS=CRS:84&BBOX=-180.0,-90.0,180.0,90.0&… (example request)

Users can also use the ESPG:4326 coordinates and use the axis ordering of lat/long:

- …&EPSG:4326&BBOX=-90.0,-180.0,90,180.0&… (example request)

Other notable changes

- valid values for the EXCEPTIONS parameter in a GetMap request are XML, INIMAGE, BLANK

- valid value for the EXCEPTIONS parameter in a GetFeatureInfo request is XML
- LayerLimit is introduced, allowing a server to advertise and limit the number of layers a client is allowed to include in a GetMap request

Some Missing features

- WMS 1.3.0 Post request should be an XML document containing the different operations and parameters.
- SLD documents containing elements from the Feature Encoding 1.1 specification could potentially use ESPG projections with some filters. It is not yet clear nor implemented if the axis ordering should be taken into account in these specific cases.

OCG compliance tests

As of version 5.4, MapServer passes all the basic and query tests of the OGC CITE test suite for WMS 1.3.0.

Reference Section

The following metadata are available in the setup of the mapfile:

(Note that each of the metadata below can also be referred to as 'ows_*' instead of 'wms_*'. MapServer tries the 'wms_*' metadata first, and if not found it tries the corresponding 'ows_*' name. Using this reduces the amount of duplication in mapfiles that support multiple OGC interfaces since "ows_*" metadata can be used almost everywhere for common metadata items shared by multiple OGC interfaces.)

Web Object Metadata

ows_allowed_ip_list (or wms_allowed_ip_list)

- *Description:* (Optional) A list of IP addresses that will be allowed access to the service.

  Example:

```
METADATA
  "ows_allowed_ip_list" "123.45.67.89 11.22.33.44"
END
```

ows_denied_ip_list (or wms_denied_ip_list)

- *Description:* (Optional) A list of IP addresses that will be denied access to the service.

  Example:

```
METADATA
  "ows_denied_ip_list" "123.45.67.89 11.22.33.44"
END
```

ows_http_max_age

- *Description:* (Optional) an integer (in seconds) to specify how long a given map response should be considered new. Setting this directive allows for aware WMS clients to use this resulting HTTP header value as a means to optimize (and minimize) requests to a WMS Server. More info is available at https://www.mnot.net/cache_docs/#CACHE-CONTROL

ows_schemas_location

- *Description:* (Optional) (Note the name ows_schemas_location and not wms_… this is because all OGC Web Services (OWS) use the same metadata) Root of the web tree where the family of OGC WMS XMLSchema files are located. This must be a valid URL where the actual .xsd files are located if you want your WMS output to validate in a validating XML parser. Default is http://schemas.opengis.net.

ows_sld_enabled

- *Description:* (Optional) A value (true or false) which, when set to "false", will ignore SLD and SLD_BODY parameters in order to disable remote styling of WMS layers. Also, SLD is not advertised in WMS Capabilities as a result

ows_updatesequence

- *Description:* (Optional) The updateSequence parameter can be used for maintaining the consistency of a client cache of the contents of a service metadata document. The parameter value can be an integer, a timestamp in [ISO 8601:2000] format, or any other number or string.

wms_abstract

- *WMS TAG Name:* Abstract (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) A blurb of text providing more information about the WMS server.

wms_accessconstraints

- *WMS TAG Name:* AccessConstraints (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) Access constraints information. Use the reserved word "none" if there are no access constraints.

wms_addresstype, wms_address, wms_city, wms_stateorprovince, wms_postcode, wms_country

- *WMS TAG Name:* ContactAddress and family (WMS1.1.1, sect. 7.1.4.2)
- *Description:* Optional contact address information. If provided then all six metadata items are required.

wms_attribution_logourl_format

- *Description:* (Optional) The MIME type of the logo image. (e.g. “image/png”). Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_height

- *Description:* (Optional) Height of the logo image in pixels. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_href

- *Description:* (Optional) URL of the logo image. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_width

- *Description:* (Optional) Width of the logo image in pixels. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_onlineresource

- *Description:* (Optional) The data provider’s URL.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_title

- *Description:* (Optional) Human-readable string naming the data provider.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_bbox_extended:

- *Description:* (Optional) “true” or “false”. If true, bounding boxes are reported for all supported SRS / CRS in the capabilities document. If false, only the bounding box of the first SRS / CRS is reported.
- Introduced in 6.0.

wms_contactelectronicmailaddress

- *WMS TAG Name:* ContactElectronicMailAddress (WMS1.1.1, sect. 7.1.4.2)
- *Description:* Optional contact Email address.

wms_contactfacsimiletelephone

- *WMS TAG Name:* ContactFacsimileTelephone (WMS1.1.1, sect. 7.1.4.2)
- *Description:* Optional contact facsimile telephone number.

wms_contactperson, wms_contactorganization, wms_contactposition

- *WMS TAG Name:* ContactInformation, ContactPerson, ContactOrganization, ContactPosition (WMS1.1.1, sect. 7.1.4.2)
- *Description:* Optional contact information. If provided then all three metadata items are required.

wms_contactvoicetelephone

- *WMS TAG Name:* ContactVoiceTelephone (WMS1.1.1, sect. 7.1.4.2)
- *Description:* Optional contact voice telephone number.

wms_enable_request (or **ows_enable_request**)

- *Description:* Space separated list of requests to enable. The default is none. The following requests can be enabled: *GetCapabilities*, *GetMap*, *GetFeatureInfo* and *GetLegendGraphic*. A “!” in front of a request will disable the request. “*” enables all requests.
- *Examples:*

  To enable only *GetMap* and *GetFeatureInfo*:

  ```
  "wms_enable_request" "GetMap GetFeatureInfo"
  ```

  To enable all requests except *GetFeatureInfo*

  ```
  "wms_enable_request" "* !GetFeatureInfo"
  ```

wms_encoding

- *WMS TAG Name:* Encoding
- *Description:* Optional XML capabilities encoding type. The default is ISO-8859-1.

wms_feature_info_mime_type

- *WMS TAG Name:* Feature_info_mime_type
- *Description:*
    - Used to specify an additional MIME type that can be used when responding to the GetFeature request.

      For example if you want to use the layer's HTML template as a base for its response, you need to add "WMS_FEATURE_INFO_MIME_TYPE" "text/html". Setting this will have the effect of advertising text/html as one of the MIME types supported for a GetFeature request. You also need to make sure that the layer points to a valid html template (see Templating). The client can then call the server with INFO_FORMAT=text/html.
    - If not specified, MapServer by default has text/plain and GML implemented.

wms_fees

- *WMS TAG Name:* Fees (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) Fees information. Use the reserved word "none" if there are no fees.

wms_getcapabilities_version

- *Description:* (Optional) Default version to use for GetCapabilities requests that do not have a version parameter. If not set, the latest supported version will be returned.

wms_getlegendgraphic_formatlist

- *Description:* (Optional) A comma-separated list of valid formats for a WMS GetLegendGraphic request.

wms_getmap_formatlist

- *Description:* (Optional) A comma-separated list of valid formats for a WMS GetMap request.

wms_keywordlist

- *WMS TAG Name:* KeywordList (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) A comma-separated list of keywords or keyword phrases to help catalog searching. As of WMS 1.1.0 no controlled vocabulary has been defined.

wms_keywordlist_vocabulary

- *WMS Attribute Name:* vocabulary of KeywordList -> Keyword
- *Description:* (Optional) Name of vocabulary used in **wms_keywordlist_[vocabulary's name]_items** as described below.

wms_keywordlist_[vocabulary's name]_items

- *WMS TAG Name:* KeywordList -> Keyword
- *Description:* (Optional) A comma-separated list of keywords or keyword phrases to help catalog searching for given vocabulary.

wms_languages

- *Description:* (Optional) A comma-separated list of supported languages. For details please refer to the section Multi-language support for certain capabilities fields in the INSPIRE View Service documentation.

wms_layerlimit

- *WMS TAG Name:* LayerLimit (WMS1.3.0, sect. 7.2.4.3)
- *Description:* (Optional) The maximum number of layers a WMS client can specify in a GetMap request. If not set, then no limit is imposed.

wms_onlineresource

- *WMS TAG Name:* OnlineResource (WMS1.1.1, sect. 6.2.2)
- *Description:* (Recommended) The URL that will be used to access this WMS server. This value is used in the GetCapabilities response.

> **See also:** Sections "Setup a Mapfile / wms_onlineresource metadata" and "More About the Online Resource URL" above.

wms_remote_sld_max_bytes

- *Description:* (Optional) Maximum size in bytes authorized when fetching a remote SLD through http. Defaults to 1 MegaByte (1048596).

wms_resx, wms_resy

- *WMS TAG Name:* BoundingBox (WMS1.1.1, sect. 6.5.6)
- *Description:* (Optional) Used in the BoundingBox tag to provide info about spatial resolution of the data, values are in map projection units.

wms_rootlayer_abstract

- *WMS TAG Name:* Abstract (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) Same as wms_abstract, applied to the root Layer element. If not set, then wms_abstract will be used.

wms_rootlayer_keywordlist

- *WMS TAG Name:* KeywordList (WMS1.1.1, sect. 7.1.4.2)
- *Description:* (Optional) Same as wms_keywordlist, applied to the root Layer element. If not set, then wms_keywordlist will be used.

wms_rootlayer_name

- *WMS TAG Name:* Name (WMS1.1.1, sect. 7.1.4.1)
- *Description:* (Optional) Same as MAP.NAME, applied to the root Layer element. If not set, then MAP.NAME will be used. If set to "", then Name element is suppressed and the root layer name will not be advertised as a GetMap capable layer.

wms_rootlayer_title

- *WMS TAG Name:* Title (WMS1.1.1, sect. 7.1.4.1)
- *Description:* (Optional) Same as wms_title, applied to the root Layer element. If not set, then wms_title will be used.

wms_service_onlineresource

- *Description:* (Optional) Top-level onlineresource URL. MapServer uses the onlineresource metadata (if provided) in the following order:

    1. wms_service_onlineresource
    2. ows_service_onlineresource

    3. wms_onlineresource (or automatically generated URL, see the onlineresource section of this document)

wms_srs

- *WMS TAG Name:* SRS (WMS1.1.1, sect. 6.5.5)
- *Description:* (Recommended) Contains a list of EPSG projection codes that should be advertised as being available for all layers in this server. The value can contain one or more EPSG:<code> pairs separated by spaces (e.g. “EPSG:4269 EPSG:4326”) This value should be upper case (EPSG:3978…..not epsg:3978) to avoid problems with case sensitive platforms.
- See Also: section “Setup a Mapfile / Map PROJECTION and wms_srs metadata” above.

wms_timeformat

- *Description:* The time format to be used when a request is sent. (e.g. “wms_timeformat” “%Y-%m-%d %H, %Y-%m-%d %H:%M”). Please see the WMS Time Support Howto for more information.

wms_title

- *WMS TAG Name:* Title (WMS1.1.1, sect. 7.1.4.1)
- *Description:* (Required) A human-readable name for this Layer.

Layer Object Metadata

gml_exclude_items

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) A comma delimited list of items to exclude. As of MapServer 4.6, you can control how many attributes (fields) you expose for your data layer with metadata. The previous behaviour was simply to expose all attributes all of the time. The default is to expose no attributes at all. An example excluding a specific field would be:

```
"gml_include_items" "all"
"gml_exclude_items" "Phonenum"
```

gml_geometries

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) Provides a name for geometry elements. The value is specified as a string to be used for geometry element names. By default, GML geometries are not written in GML GetFeatureInfo output, unless gml_geometries and gml_[geometry name]_type are both set. By default, only the bounding box is written. If gml_geometries is set to “none”, neither the bounding box nor the geometry are written.

gml_groups

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) A comma delimited list of group names for the layer.

gml_[group name]_group

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) A comma delimited list of attributes in the group. Here is an example:

```
"gml_include_items" "all"
"gml_groups"        "display"
"gml_display_group" "Name_e,Name_f"
```

gml_include_items

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) A comma delimited list of items to include, or keyword “all”. As of MapServer 4.6, you can control how many attributes (fields) you expose for your data layer with this metadata. The previous behaviour was simply to expose all attributes all of the time. You can enable full exposure by using the keyword “all”, such as:

```
"gml_include_items" "all"
```

You can specify a list of attributes (fields) for partial exposure, such as:

```
"gml_include_items" "Name,ID"
```

The new default behaviour is to expose no attributes at all.

gml_[item name]_alias

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) An alias for an attribute's name. The served GML will refer to this attribute by the alias. Here is an example:

```
"gml_province_alias" "prov"
```

gml_[item name]_type

- *Description:* (Optional) Specifies the type of the attribute. Valid values are the OGR data types: Integer|Long|Real|Character|Date|Time|DateTime|Boolean. MapServer translates these to valid GML data types.

  Note: Long is to be used for 64-bit integers, starting with MapServer 7.0.1.

  Note: Time and DateTime have been added in MapServer 8. And since MapServer 8, Date semantics is a date, without time, whereas in previous versions, it was used indifferently for Date, Time or DateTime.

gml_[geometry name]_type

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) When employing gml_geometries, it is also necessary to specify the geometry type of the layer. This is accomplished by providing a value for gml_[geometry name]_type, where [geometry name] is the string value specified for gml_geometries, and a value which is one of:
  - point
  - multipoint
  - line
  - multiline
  - polygon
  - multipolygon

gml_xml_items

- *Description:* (Optional, applies only to GetFeatureInfo GML requests) A comma delimited list of items that should not be XML-encoded.

ows_allowed_ip_list
Same as ows_allowed_ip_list in the Web Object.

ows_denied_ip_list
Same as ows_denied_ip_list in the Web Object.

wms_abstract
Same as wms_abstract in the Web Object.

wms_attribution_logourl_format

- *Description:* (Optional) The MIME type of the logo image. (e.g. "image/png"). Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_height

- *Description:* (Optional) Height of the logo image in pixels. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_href

- *Description:* (Optional) URL of the logo image. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_logourl_width

- *Description:* (Optional) Width of the logo image in pixels. Note that the other wms_attribution_logourl_* metadata must also be specified.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_onlineresource

- *Description:* (Optional) The data provider's URL.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_attribution_title

- *Description:* (Optional) Human-readable string naming the data provider.
- refer to section 7.1.4.5.11 of the WMS 1.1.1 spec.

wms_authorityurl_name, wms_authorityurl_href

- *Description:* (Optional) AuthorityURL is used in tandem with Identifier values to provide a means of linking identifier information back to a web service. The wms_identifier_authority should provide a string that matches a declared wms_authorityurl_name. Both wms_authorityurl_name and wms_authorityurl_href must be present for an AuthorityURL tag to be written to the capabilities.

- refer to section 7.1.4.5.12 of the WMS 1.1.1 spec.

wms_bbox_extended:

- *Description:* (Optional) “true” or “false”. If true, bounding boxes are reported for all supported SRS / CRS in the capabilities document. If false, only the bounding box of the first SRS / CRS is reported.
- Introduced in 6.0.

wms_dataurl_format

- *Description:* (Optional) Non-standardized file format of the metadata. The layer metadata wms_dataurl_href must also be specified.
- refer to section 7.1.4.5.14 of the WMS 1.1.1 spec.

wms_dataurl_href

- *Description:* (Optional) The URL to the layer’s metadata. The layer metadata wms_dataurl_format must also be specified.
- refer to section 7.1.4.5.14 of the WMS 1.1.1 spec.

wms_enable_request (or **ows_enable_request**)

- *Description:* Space separated list of requests to enable. The default is none. The following requests can be enabled: *GetCapabilities*, *GetMap*, *GetFeatureInfo* and *GetLegendGraphic*. A “!” in front of a request will disable the request. “*” enables all requests.
- *Examples:*

  To enable only *GetMap* and *GetFeatureInfo*:

  ```
  "wms_enable_request" "GetMap GetFeatureInfo"
  ```

  To enable all requests except *GetFeatureInfo*

  ```
  "wms_enable_request" "* !GetFeatureInfo"
  ```

wms_exclude_items

- *Description:* (Optional, applies only to GetFeatureInfo text/plain requests) A comma delimited list of items to exclude, or keyword “all”.

  See gml_exclude_items above.

wms_extent

- *WMS TAG Name:* BoundingBox (WMS1.1.1, sect. 6.5.6)
- *Description:* (Optional) Used for the layer’s BoundingBox tag for cases where it is impossible (or very inefficient) for MapServer to probe the data source to figure its extents (such as with databases). The value for this metadata is “minx miny maxx maxy” separated by spaces, with the values in the layer’s projection units. If wms_extent is provided then it has priority and MapServer will NOT try to read the source file’s extents.

  See also: Vector Data Management & Optimization

  For Rasters served through WMS, MapServer can now use the wms_extent metadata parameter to register the image. If a .wld file cannot be found, MapServer will then look for the wms_extent metadata parameter and use the extents of the image and the size of the image for georegistration.

  See also: Raster Management & Optimization

wms_getfeatureinfo_formatlist

- *Description:* (Optional) Comma-separated list of formats that should be valid for a GetFeatureInfo request. If defined, only these formats are advertised through in the Capabilities document.

wms_getlegendgraphic_formatlist

- *Description:* (Optional) Comma-separated list of image formats that should be valid for a GetLegendGraphic request. If defined, only these formats are advertised through in the Capabilities document.

wms_getmap_formatlist

- *Description:* (Optional) Comma-separated list of image formats that should be valid for a GetMap request. If defined, only these formats are advertised through in the Capabilities document.

wms_group_abstract

- *Description:* (Optional) A blurb of text providing more information about the group. Only one layer for the group needs to contain wms_group_abstract, MapServer will find and use the value. The value found for the first layer in the group is used. So if multiple layers have wms_group_abstract set then only the first value is used.

wms_group_title

- *WMS TAG Name:* Group_title (WMS1.1.1, sect. 7.1.4.1)
- *Description:* (Optional) A human-readable name for the group that this layer belongs to. Only one layer for the group needs to contain wms_group_title, MapServer will find and use the value. The value found for the first layer in the group is used. So if multiple layers have wms_group_title set then only the first value is used.

wms_identifier_authority, wms_identifier_value

- *Description:* (Optional) Identifier is used in tandem with AuthorityURL end points to provide a means of linking identifier information back to a web service. The wms_identifier_authority should provide a string that matches a declared wms_authorityurl_name. Both wms_identifier_authority and wms_identifier_value must be present for an Identifier tag to be written to the capabilities.
- refer to section 7.1.4.5.12 of the WMS 1.1.1 spec.

wms_include_items

- *Description:* (Optional, applies only to GetFeatureInfo text/plain requests) A comma delimited list of items to include, or keyword "all".

  See gml_include_items above.

wms_keywordlist
Same as wms_keywordlist in the Web Object.

wms_keywordlist_vocabulary
Same as wms_keywordlist_vocabulary in the Web Object.

wms_keywordlist_[vocabulary's name]_items
Same as wms_keywordlist_[vocabulary's name]_items in the Web Object.

wms_layer_group

- *Description:* (Optional) Can be used to assign a layer to a number of hierarchically nested groups. This grouped hierarchy will be expressed in the capabilities.

  From MapServer 7.2, WMS_LAYER_GROUP is similar to the GROUP keyword in that it always publishes the name and the tile of the group in the capabilities. As a consequence the groups set with WMS_LAYER_GROUP can always be requested with a GetMap or GetFeatureInfo request (see section 7.1.4.5.2 of the WMS implementation specification version 1.1.1. (OGC 01-068r2)).

  A difference with GROUP is that GROUP does not support nested groups. The purpose of this metadata setting is to enable making a WMS client aware of layer grouping.

  All group names should be preceded by a forward slash (/). It is not allowed to use both the WMS_LAYER_GROUP setting and the GROUP keyword for a single layer. GROUP can be seen as deprecated!

```
LAYER
  NAME "mylayer"
  DATA "mylayer"
  TYPE LINE
  CLASS
    STYLE
      COLOR 100 100 255
    END
  END
  METADATA
   "WMS_LAYER_GROUP" "/rootgroup/subgroup"
  END
END
```

wms_metadataurl_format

- *Description:* (Optional) The file format MIME type of the metadata record (e.g. "text/plain"). The layer metadata wms_metadataurl_type and wms_metadataurl_href must also be specified.
- refer to section 7.1.4.5.10 of the WMS 1.1.1 spec.
- To output several MetadataURL elements, use `wms_metadataurl_list`

wms_metadataurl_href

- *Description:* (Optional) The URL to the layer's metadata. The layer metadata wms_metadataurl_format and wms_metadataurl_type must also be specified.
- refer to section 7.1.4.5.10 of the WMS 1.1.1 spec.
- To output several MetadataURL elements, use `wms_metadataurl_list`

wms_metadataurl_list

- *Description:* (Optional) Space-separated list of parts of metadata keys, to be able to specify several MetadataURL. If the value of wms_metadataurl_list is "foo bar", then wms_metadataurl_foo_href, wms_metadataurl_foo_format, wms_metadataurl_foo_type, wms_metadataurl_bar_href, wms_metadataurl_bar_format, wms_metadataurl_bar_type will be queried with the semantics of the wms_metadataurl_href, wms_metadataurl_format and wms_metadataurl_type items.

  Example:

```
"wms_metadataurl_list" "xml html"

"wms_metadataurl_xml_format" "text/xml"
"wms_metadataurl_xml_type" "TC211"
"wms_metadataurl_xml_href" "http://example.com/testXML"

"wms_metadataurl_html_format" "text/html"
"wms_metadataurl_html_type" "TC211"
"wms_metadataurl_html_href" "http://example.com/testHTML"
```

- *New in version 8.0.*

wms_metadataurl_type

- *Description:* (Optional) The standard to which the metadata complies. Currently only two types are valid: “TC211” which refers to [ISO 19115], and “FGDC” which refers to [FGDC-STD-001-1988]. The layer metadata wms_metadataurl_format and wms_metadataurl_href must also be specified.
- refer to section 7.1.4.5.10 of the WMS 1.1.1 spec.
- To output several MetadataURL elements, use `wms_metadataurl_list`

wms_opaque

- *WMS TAG Name:* Opaque (WMS1.1.1, sect. 7.1.4.6.3)
- *Description:* (Optional) Set this metadata to “1” to indicate that the layer represents an area-filling coverage of space (e.g. a bathymetry and elevation layer). This should be taken by the client as a hint that this layer should be placed at the bottom of the stack of layers.

wms_srs

Same as wms_srs in the Web Object .

wms_style

- *Description:* (Optional) The LegendURL style name. Requires the following metadata: wms_style_[style’s_name]_width, wms_style_[style’s_name]_legendurl_height, wms_style_[style’s_name]_legendurl_format, wms_style_[style’s_name]_legendurl_href
- refer to section 7.1.4.5.4 of the WMS 1.1.1 spec.

wms_style_[style’s_name]_legendurl_format

- *Description:* (Optional) The file format MIME type of the legend image. Requires the following metadata: wms_style_[style’s_name]_width, wms_style_[style’s_name]_legendurl_height, wms_style, wms_style_[style’s_name]_legendurl_href.
- refer to section 7.1.4.5.4 of the WMS 1.1.1 spec.

wms_style_[style’s_name]_legendurl_height

- *Description:* (Optional) The height of the legend image in pixels. Requires the following metadata: wms_style_[style’s_name]_width, wms_style, wms_style_[style’s_name]_legendurl_format, wms_style_[style’s_name]_legendurl_href.
- refer to section 7.1.4.5.4 of the WMS 1.1.1 spec.

wms_style_[style’s_name]_legendurl_href

- *Description:* (Optional) The URL to the layer’s legend. Requires the following metadata: wms_style_[style’s_name]_width, wms_style_[style’s_name]_legendurl_height, wms_style_[style’s_name]_legendurl_format, wms_style.
- refer to section 7.1.4.5.4 of the WMS 1.1.1 spec.

wms_style_[style’s_name]_legendurl_width

- *Description:* (Optional) The width of the legend image in pixels. Requires the following metadata: wms_style_[style’s_name]_format, wms_style_[style’s_name]_legendurl_height, wms_style, wms_style_[style’s_name]_legendurl_href.
- refer to section 7.1.4.5.4 of the WMS 1.1.1 spec.

wms_timedefault

- *Description:* (Optional for Time Support) This value is used if it is defined and the Time value is missing in the request. Please see the WMS Time Support Howto for more information.

wms_timeextent

- *Description:* (Mandatory for Time Support) This is used in the capabilities to return the valid time values for the layer. The value defined here should be a valid time range. Please see the WMS Time Support Howto for more information.

wms_timeitem

- *Description:* (Mandatory for Time Support) This is the name of the field in the DB that contains the time values. Please see the WMS Time Support Howto for more information.

wms_title

Same as wms_title in the Web Object.

Layer Metadata API

If `wms_metadataurl_href` is not defined, MapServer will provide a link to the Layer Metadata API for the given layer in the <MetadataURL> element See the Layer Metadata API documentation for more information.

Vendor specific WMS parameters

angle

- Angle (in degrees) to rotate the map.

  Note: The angle value is in degrees clockwise.

radius

- This parameter accepts two types of input:

- An integer that specifies the search radius in pixels.
- The special value *bbox* that will change the query into a bbox query based on the bbox given in the request parameters.

bbox_pixel_is_point

- If this parameter is "TRUE", MapServer will treat the BBOX received in WMS GetMap requests as if it was provided in pixel_is_point mode. Essentially disabling the conversion from pixel_is_area (WMS model) to pixel_is_point that is present in mapwms.c for that specific mapfile.

Cascading WMS Requests

Currently, there are 3 requests that support WMS cascading:

- GetMap
- GetFeatureInfo
- GetLegendGraphic

  Before MapServer 6.2, a GetLegendGraphic request was not cascaded. A legend was returned using the layer classes. To preserve that behavior, a GetLegendGraphic request will be cascaded only **if**:

  1. The GetLegendGraphic request is enabled via the *_enable_request* metadata.
  2. The layer does not contain any class with the name property set. ie:

```
CLASS
  NAME "Parks"
  STYLE
    COLOR 0 255 0
  END
END
```

  This layer won't be cascaded because it contains at least a class with the property NAME set.

Note: If you know that the remote WMS server *does not* support a given WMS request, you should disable this request explicitly for your layer using the (ows/wms)_enable_request metadata. Otherwise, you will simply get the XML exception from the cascaded server.

Sample WMS Server Mapfile

The following is a very basic WMS Server mapfile:

```
MAP
  NAME "WMS-test" ##strongly recommended. containing no special characters or spaces
  STATUS ON
  SIZE 400 300
  EXTENT -2200000 -712631 3072800 3840000
  UNITS METERS
  SHAPEPATH "./data"
  IMAGECOLOR 255 255 255
  FONTSET "../etc/fonts.txt"

  WEB
    IMAGEPATH "/ms4w/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
    METADATA
      "wms_title"          "WMS Demo Server"  ##required
      "wms_abstract"       "Longer description of your service" ##recommended
      "wms_onlineresource" "http://yourpath/cgi-bin/mapserv.exe?map=/path/to/your-mapfile.map"   ##required (must cha
      "wms_srs"            "EPSG:3978 EPSG:3857 EPSG:4269 EPSG:4326"  ##recommended
      "wms_enable_request" "*"   ##required
      "wms_getfeatureinfo_formatlist" "text/plain,text/html,application/vnd.ogc.gml,gml" ##recommended
    END
  END # Web

  PROJECTION
    "init=epsg:3978"   ##required
  END

  SYMBOL
    NAME "circle"
    TYPE ellipse
    POINTS 1 1 END
  END # Symbol

  #
  # Start of layer definitions
  #

  LAYER
    NAME "park" ##no special characters or spaces
    METADATA
      "wms_title"         "Parks"   ##required
      "wms_abstract"      "Longer description of your layer" ##recommended
      "wms_include_items" "all" ##optional
      "gml_include_items" "all" ##optional
```

```
    "gml_featureid"     "ogc_fid" ##optional
    "gml_geometries"    "msgeom" ##optional
    "gml_msgeom_type"   "multipolygon" ##optional
  END
  TYPE POLYGON
  STATUS ON
  DATA "park.shp"
  PROJECTION
    "init=epsg:3978"   ##recommended
  END
  CLASS
    NAME "Parks"
    STYLE
      COLOR 200 255 0
      OUTLINECOLOR 120 120 120
    END # Style
  END # Class
  TEMPLATE "empty" ##recommended (enable GetFeatureInfo / can point to valid .html template)
END # Layer

LAYER
  NAME "cities" ##no special characters or spaces
  METADATA
    "wms_title"         "Cities"  ##required
    "wms_abstract"      "Longer description of your layer" ##recommended
    "wms_include_items" "all" ##optional
    "gml_include_items" "all" ##optional
    "gml_featureid"     "ogc_fid" ##optional
    "gml_geometries"    "msgeom" ##optional
    "gml_msgeom_type"   "point" ##optional
  END
  TYPE POINT
  STATUS ON
  DATA "popplace.shp"
  PROJECTION
    "init=epsg:3978"   ##recommended
  END
  CLASS
    NAME "Cities"
    STYLE
      SYMBOL "circle"
      SIZE 8
      COLOR 0 0 0
    END # Style
  END # Class
  TEMPLATE "empty" ##recommended (enable GetFeatureInfo / can point to valid .html template)
END # Layer

END # mapfile
```

FAQ / Common Problems

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

A: If you know some of the parameters of your data's projection, then you can use the search tool at https://epsg.io/ to find your data's associated EPSG code. The EPSG official home now also allows you to perform a text or map search for a projection: https://epsg.org/

For PROJ version >= 6: the EPSG codes are stored in a SpatiaLite database *proj.db* usually located in /usr/local/share/proj/ on Unix systems and in C:/PROJ/ or C:/PROJ/NAD in Windows systems (depending on the installation), specifically /MS4W/proj/nad/ for MS4W users.

For PROJ version < 6: you can browse the "epsg" file that comes with PROJ 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. The "epsg" file is usually located in /usr/local/share/proj/ on Unix systems and in C:/PROJ/ or C:/PROJ/NAD in Windows systems (depending on the installation). MS4W users will find the epsg file in /MS4W/proj/nad/.

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

A: That's likely PROJ complaining that it cannot find the *epsg* projection definition file (for PROJ version < 6), or the *proj.db* file (for PROJ version >= 6). Make sure you have installed PROJ and that the *proj.db* file (or *epsg* depending on your local PROJ version) 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/ or C:/PROJ/NAD (depending on the installation), MS4W users will find the file in /MS4W/proj/nad/. You should also check the error documentation to see if your exact error is discussed.

If you don't have the *proj.db* file (or *epsg* file) then you can get it as part of the PROJ distribution at https://proj.org/ Alternatively, you can manually download the PROJ < 6 *epsg* file at http://www.maptools.org/dl/proj4-epsg.zip.

Q: **How do AUTO projections work?**

A: When a WMS client calls a WMS server with an auto projection, it has to specify the SRS in the form: AUTO: proj_id,unit_id,lon0,lat0 where:

- proj_id is one of 42001, 42002, 42003, 42004, or 42005 (only five auto projections are currently defined).
- unit_id is always 9001 for meters. (It is uncertain whether anyone supports any other units.)
- lon0 and lat0 are the coordinates to use as the origin for the projection.

When using an AUTO projection in WMS GetCapabilities, you include only the "AUTO:42003" string in your wms_srs metadata, you do not include the projection parameters. Those are added by the application (client) at runtime depending on the map view. For example:

```
NAME "DEMO"
...
```

```
WEB
  ...
  METADATA
    "wms_title"            "WMS Demo Server"
    "wms_onlineresource"   "http://my.host.com/cgi-bin/mapserv?map=/path/to/your-mapfile.map" #must change mapfile path
    "wms_srs"              "AUTO:42001 AUTO:42002"
    "wms_enable_request" "*"   ##necessary
  END # METADATA
END # WEB
```

The above server advertises the first two auto projections.