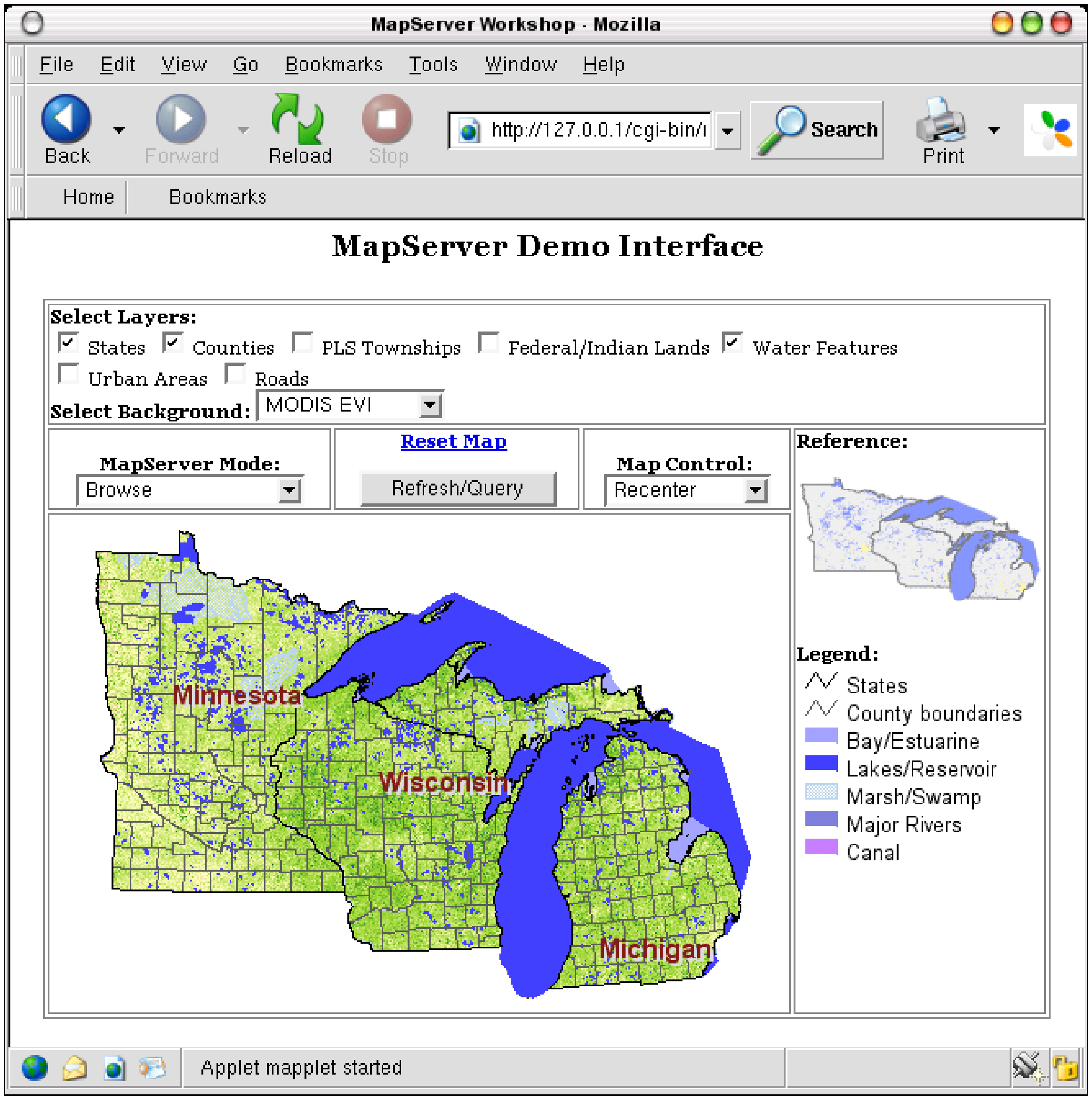
TerraSIP

Introduction to Web Mapping Applications Using MapServer

In this workshop you will learn how to create an application similar to the one



Select Layers: [✓ States] [✓ Counties] ☐ PLS Townships ☐ Federal/Indian Lands [✓ Water Features] ☐ Urban Areas ☐ Roads

Select Background: MODIS EVI

MapServer Mode: Browse

Map Control: Recenter

Legend:

⋀	States
⋀	County boundaries
🟪	Bay/Estuarine
🟦	Lakes/Reservoir
▨	Marsh/Swamp
🟪	Major Rivers
🟪	Canal

Applet mapplet started

shown below using MapServer version 3.6 and higher.

You will start with a simple application and add components of increasing complexity to create a WMS compliant application with on-the-fly projection.

Requirements:
Apache httpd server
MapServer 3.6.x or 4.x Beta
Text editor (Word Pad or Notepad)
Web browser (Internet Explorer)

Workshop dataset (will be available on the MapServer web site)
References:
MapServer Home Page:
    http://mapserver.gis.umn.edu/
MapServer Documentation Page
    http://mapserver.gis.umn.edu/doc.html
Template Reference
    http://mapserver.gis.umn.edu/doc/cgi-reference.html
Map file Reference (Bookmark this page now)
    http://mapserver.gis.umn.edu/doc/mapfile-reference.html

Getting Started

Installation

Everything you need to get started has already been installed. Executables and installation instructions for Windows can be downloaded on MapServer's (http://mapserver.gis.umn.edu/win32binaries.html) or DM Solutions' (http://www2.dmsolutions.ca/mapserver/dl) web site.

The system I used is configured as follows (this could change):

Apache Document Root:	C:\Program Files\Apache Group\Apache2\htdocs
Script Alias Directory:	C:\Program Files\Apache Group\Apache2\cgi-bin
MapServer Executable:	C:\Program Files\Apache Group\Apache2\cgi-bin\mapserv.exe
Workshop Files:	C:\Program Files\Apache Group\Apache2\htdocs\ms_workshop
Workshop Web Page:	http://localhost/ms_workshop/
Temp Directory:	C:\Program Files\Apache Group\Apache2\htdocs\tmp
Temp URL:	http://localhost/tmp/

Before we begin with the first exercise, let us take the time to make an index file. Open a text editor (vi, nedit, xedit, whatever you're comfortable with) and type in the following HTML code:

```
<html>
  <head>
    <title>MapServer Examples</title>
  </head>
  <body bgcolor=”#ffffff”>
    <h1 align=”center”>
      <a href=”/ms_workshop/example1.html”>Example 1</a>
    </h1>
  </body>
```

```
</html>
```

Save this as index.html and open it on a web browser by typing the following URL: `http://localhost/ms_workshop/`. We will add links to this page throughout the course of the workshop.

Example 1: Displaying a basic map

Let's open the file "example1.html" (under C:/Program Files/Apache Group/Apache2/htdocs/ms_workshop).   Make sure the code includes the following line:

```
<img border="1" src="/cgi-bin/mapserv.exe?
Map=C:/Program Files/Apache
Group/Apache2/htdocs/ms_workshop/example1.map&amp;mode=map">
```

View this page by clicking on the "Example 1" link from the index page. This shows you how to display a static map on a page.

Now, open the map file "example1.map". This is the configuration file for Example 1; it's referenced in the "src" attribute of the "img" tag. It describes some basic configuration parameters for the application and one "LAYER", based on the shapefile "states_ugl.shp", showing the outline of the Upper Great Lakes states of Minnesota, Wisconsin, and Michigan. The "CLASS" object tells MapServer how to draw the polygons in the shape file. You can draw only the land polygons by using the attribute in the shapefile called "CLASS".

- Add the following line to the "CLASS" object to draw the land only.
  ```
  EXPRESSION ('[CLASS]' = 'land')
  ```
- Try drawing only the 'water'.
- Try drawing both "classes "using the `EXPRESSION` keyword.
- Try changing the color of the map.

Next we can add a label layer. It's easy to add labels, but not so easy to get them to look good automatically. We add labels as another layer in the map file.

- Add the following layer below the states layer in the map file.

```
LAYER # States Labels
  NAME state_label
  DATA states_ugl
  STATUS DEFAULT
  TYPE ANNOTATION

  PROJECTION
    "init=epsg:4269"
  END

  LABELITEM "STATE"
  CLASSITEM "CLASS"
  CLASS # States class
```

```
    EXPRESSION 'land'
    COLOR -1 -1 -1
    LABEL
      COLOR 132 31 31
      SHADOWCOLOR 218 218 218
      SHADOWSIZE 2 2
      TYPE TRUETYPE
      FONT arial-bold
      SIZE 14
      ANTIALIAS TRUE
      POSITION CL
      PARTIALS FALSE
      MINDISTANCE 250
      BUFFER 4
    END # end of label
  END # end of States class object
END # end of layer object
```

- Uncomment the FONTSET and SYMBOLSET parameters (near the top of the map file.

To get the labels to display properly, you can change some of the parameters in the annotation layer.

- Try changing the "position" and "mindistance" parameters.

Now let's try adding a raster layer. There is a tiff file in the raster subdirectory of the data directory. Add it below the state layer so that it will display above that layer. You can use the following code:

```
LAYER # Shaded Relief Raster
  NAME relief
  DATA "raster/shdrlfi020g_ugl.tif"
  STATUS DEFAULT
  TYPE RASTER
  OFFSITE 0

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

To put the state borders on top, you have to add that layer below the raster layer.

- Copy the states layer and the paste it on the line after the raster layer.
- Change the layer TYPE to LINE.
- Move the labels back on top.

Example 2: Adding a User Interface

Controlling the format of MapServer output and creating a user interface is basically an HTML programming task. If you are unfamiliar with HTML you may refer to the NCSA (at UIUC) Beginners Guide to HTML at http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerP1.html.

Two additional documents have been provided for this exercise. You will find a new map file called example2.map and an HTML template called example2.html in the ms_workshop directory. In order to view an application with an HTML template you must:

1. Initialize the application by referencing the map file to be use, i.e.:

```
      <a href="/cgi-
      bin/mapserv.exe?map=C:/Program Files/Apache
Group/Apache2/htdocs/ms_workshop/example2.map">
      Example 2</a>
```

   This is one method to initialize a MapServer application. We can also use a form that allow users to customize certain application parameters (what layers to display, what projection to use).

2. Add a "Web" object in the map file to specify which HTML template to use.
3. Create an HTML template that interacts with the MapServer cgi through a form.

The Web Object in the map file

Open the example2.map file. This map file is based on the file you created in Example 1 with several additional layers. In addition, a "Web" object has been added.

```
WEB
  TEMPLATE example2.html
  IMAGEPATH "C:/Program Files/Apache Group/Apache2/htdocs/tmp/"
  IMAGEURL "/tmp/"
END
```

The "WEB" object specifies three things:

1. the HTML template file,
2. the path to the temp directory that MapServer uses to write temporary files and images,
3. the URL for the temp directory (notice that the temp directory must be in the document root or an otherwise web accessible directory).

HML Template File

Open the example2.html file. This file creates a user interface that allows the users to zoom in on the map and control which layers are displayed.

1. Tables control the layout of the application (cascading style sheets could be used).
2. The user interface is contained within a form that accesses the MapServer CGI.

3.MapServer CGI variables are specified by square brackets (e.g. "[map]"). The MapServer CGI program replaces these values as the user changes the display.
4.The "layer" variables correspond to raster and vector layers in the map file.
5.Certain CGI variables are required by MapServer and are specified by "hidden" input at the beginning of the form.

Try making some changes to the Example 2 application.

- Access the application through a web browser and try changing the display. You must click refresh every time you change a layer. Notice how the URL changes.
- Add one more vector layer check box to the application. There is a layer called "urban" in the Example 2 map file.

Adding a scalebar, legend, and reference map

The column on the right hand side of your application has been reserved for a legend and reference map. These capabilities are built in to MapServer.

- Add the following objects to the Example 2 map file after the WEB object to add a scalebar, legend, and reference map to your application.
- Find where these objects are added in the example2.html template file.

```
SCALEBAR
  IMAGECOLOR 255 255 255
  LABEL
    COLOR 0 0 0
    SIZE TINY
  END
  STYLE 1
  SIZE 72 3
  COLOR 0 0 0
  UNITS MILES
  INTERVALS 2
  TRANSPARENT TRUE
  STATUS EMBED
  POSITION LR
END

LEGEND
  STATUS ON
  LABEL
    TYPE TRUETYPE
    FONT arial
    COLOR 0 0 0
    SIZE 10
    ANTIALIAS TRUE
  END
END

REFERENCE
  STATUS ON
  IMAGE "C:/Program Files/Apache
Group/Apache2/htdocs/ms_workshop/images/ugl_ref1.gif"
  SIZE 155 105
  EXTENT 201621.496941 -294488.284025 1425518.020722 498254.511514
  COLOR -1 -1 -1
  OUTLINECOLOR 255 0 0
END
```

Making layers scale-dependent
Some layers contain a great amount of detail and will clutter your map unless the user has zoomed in to a certain scale. For such layers, you can add scale-dependency – only turning them on when the map is zoomed in.

- Add the layer below to add labels to the lakes when the scale is smaller that 1:800000.
- Try changing the MAXSCALE and SYMBOLSCALE to change the scale at which the labels turn on and off and how large they appear.

```
LAYER # Lakes Labels
  NAME lake_label
  DATA hydrop_ugl
  STATUS DEFAULT
  TYPE ANNOTATION
  MAXSCALE 800000
  SYMBOLSCALE 800000


  PROJECTION
    "init=epsg:4269"
  END

  LABELITEM "NAME"
  CLASSITEM "FEATURE"
  CLASS
    NAME "Lakes/Reservoir"
    EXPRESSION ('[FEATURE]' eq 'Lake' or '[FEATURE]' = 'Reservoir')
    COLOR -1 -1 -1
    LABEL
      COLOR 132 31 31
      SHADOWCOLOR 218 218 218
      SHADOWSIZE 2 2
      TYPE TRUETYPE
      FONT arial-bold
      SIZE 9
      ANTIALIAS TRUE
      POSITION CL
      PARTIALS FALSE
      MINDISTANCE 250
      BUFFER 4
    END # end of label
  END # end of LAKES class object
END # end of layer object
```

Example 3: Queries

Now that we have covered a little bit of HTML template files, we can start to look at another neat feature of MapServer—queries. MapServer provides several ways to query vector data layers but we will only look at two of these: the QUERY and NQUERY modes. The QUERY mode allows you to query features of a single layer, usually the topmost active layer. NQUERY mode allows you to query features of all active layers in your application.

Let's first open example2.html and save it as example3.html. We'll work with example3.html throughout our query examples.

This is how we set our query modes in the html template file:

```
<select name="mode">
  <option value="browse">Browse</option>
  <option value="query">Single-Layer Query</option>
  <option value="nquery">Multi-Layer Query</option>
</select>
```

And this is how we set query options in the map file, example3.map (I'll use the county layer as example):

```
# Snippet from the web object
WEB
  HEADER header.html
  TEMPLATE example1.html
  FOOTER footer.html
# end of snippet from web object

# County layer snippet
  HEADER "county_header.html"
  FOOTER "county_footer.html"
  CLASSITEM "county"
  CLASS
    NAME "County boundaries"
    EXPRESSION /./
    TEMPLATE "county_query.html"
    COLOR 92 92 92
  END
END # end of county layer snippet
```

Note the keywords HEADER, FOOTER, and TEMPLATE. These are the keywords that control the html query template files. Now, let's look at those files. Here's the web object HEADER template:

```
<html>
<head><title>MapServer Demo Interface</title></head>
<link type="text/css" rel="stylesheet" href="/ms_workshop/ms35.css" />
<body bgcolor=#FFFFFF>
```

Here's the web object FOOTER template:

```
</body>
</html>
```

Here's the county layer HEADER template:

```
<h4><b>Layer: Counties</b></h4><p>
<table cellpadding=5 cellspacing=2 border=0>
<tr bgcolor=#CCCCCC><th>STATE</th><th>COUNTY</th></tr>
```

Here's the county layer FOOTER template:

```
</table><p>
```

Finally, here's the county layer query TEMPLATE file:

```
<tr><td>[STATE]</td><td>[COUNTY]</td></tr>
```

Notice that if you put them together, in the order that MapServer would put them—WEB HEADER, LAYER HEADER, CLASS TEMPLATE, LAYER FOOTER, and WEB FOOTER—they form a proper html file. This is how the QUERY mode works. NQUERY works the same except that the LAYER FOOTER, and the LAYER HEADER templates are repeated for each LAYER, and the CLASS TEMPLATE is repeated for each CLASS within a LAYER.

Now let's work on our example map and HTML template files.
Step 1. Create a file called header.html and type the following lines of html code:

```
<html>
<head><title>MapServer Demo Interface</title></head>
<link type="text/css" rel="stylesheet" href="/ms_workshop/ms35.css" />
<body bgcolor=#FFFFFF>
```

Step 2. Create a file called footer.html and insert the following lines:

```
        <h4>Query Map Examples</h4>
        <p>
        <table cellpadding="3" cellspacing="0" border="0">
        <tr>
          <td>
           <img border="2" name="img" src="[img]">
          </td>
          <td>
           <img border="2"
src="/cgi-bin/mapserv.exe?map=[map]&queryfile=c:/apps/
apache/htdocs/tmp/EXAMPLE3[id].qy[get_layers]
&mode=map&size=200+150">
          </td>
         </tr>
         <tr>
         <th align="center">standard querymap</th>
         <th align="center">cached query</th>
         </tr>
        </table>
        </body>
       </html>
```

Step 3. Create a file called county_header.html. This will be the header template file for the county layer. Insert the following lines in the file and save when finished:

```
<h4><b>Layer: Counties</b></h4>
<p><table cellpadding=5 cellspacing=2 border=0>
<tr bgcolor=#CCCCCC><th>STATE</th><th>COUNTY</th></tr>
```

Step 4. Create a file called county_footer.html. This will be the county layer's footer template. Insert the following line and save when finished:

```
</table><p>
```

Step 5. Create one more file and call it county_query.html. This is where MapServer will dump your layer query results. Insert the following line in the file and save when done:

```
<tr><td>[STATE]</td><td>[COUNTY]</td></tr>
```

Let's have a quick look at this file. The two MapServer CGI variables—[STATE] and [COUNTY]—we supplied here come from the shapefile's attribute table (the DBF file associated with the county shapefile). You can use ArcView (or other data viewer such as ESRI's ArcExplorer, PCI Geomatics' FreeView, etc.) to look at the attribute table and determine which attribute to include in your query template. You can also use Excel or similar spreadsheet software to open the DBF file (open as DBase IV file) but make sure not to make any changes to the file. You could ruin the association between this table and the rest or the shapefile files if you change the wrong attribute.

Step 6. Now, let's edit our map file. Open the file example3.map in a text editor and append the following lines. Find the WEB object and add these two lines within the object:

```
HEADER “header.html”
FOOTER “footer.html”
```

Find the county LAYER and just before the CLASS object (just after the keyword CLASSITEM), insert the following lines:

```
HEADER “county_header.html”
FOOTER “county_footer.html”
```

And within the county CLASS object, insert this line (anywhere but preferably after the keyword EXPRESSION):

```
TEMPLATE “county_query.html”
```

Step 7. Check the html template file, example3.html, and look for the lines:

```
      <select name="mode">
        <option value="browse">Browse</option>
        <option value="query">Single-Layer Query</option>
        <option value="nquery">Multi-Layer Query</option>
      </select>
```

If the query and nquery options are not there, make sure to add it. Save the file if you made any changes.

Let's test our query by running the application. Type http://localhost/cgi-bin/mapserv.exe?map=c:/apps/apache/htdocs/ms_workshop/example3.map&layer=state&layer=county&layer=water on your browser's URL box. Select "Single-Layer Query' on the MapServer Mode drop-down selection menu. Click on any county. MapServer should return the results complete with a QUERYMAP. We'll get into the QUERYMAP shortly but first we should add a few more queryable layers. Let's make the "water features", the "federal/indian lands", and "urban" layers available for query.

Step 8. Repeat steps 3 to 5, using the layer name as prefix for each filename. Remember to add or delete table rows to accommodate the number of attributes in your template.

The table below lists the attributes to use in the CLASS TEMPLATE query template:

LAYER	**ATTRIBUTES**
WATER	FEATURE, NAME, STATE
FED	FEATURE1, FEATURE2, NAME1,URL,STATE
URBAN	NAME, STATE

Step 9. Repeat step 6 for each of the layers above.

Test the query again by selecting different layers. Try the NQUERY mode by selecting "Multi-Layer Query" from the MapServer Mode drop-down selection menu. Click anywhere where there's more than one active layer (i.e., lakes on top of urban areas).

QUERYMAP
Querymaps allow users to see what their query results look like in map form. To set a querymap, you need to add the QUERYMAP object in your map file. It's been done for you but do have a look at it—it's a simple but very useful object.

QUERYFILE
The second map generated by your query comes from the QUERYFILE. QUERYFILE is a MapServer CGI variable (not a map file object) and is generated by MapServer using the SAVEQUERY variable. When SAVEQUERY is given a value of "true", MapServer generates a QUERYFILE which can then be used to create a querymap on subsequent queries.

Example 4: Projections
Open a browser window and enter the following URL: http://127.0.0.1/cgi-bin/mapserv.exe?map=C:/Program Files/Apache Group/Apache2/htdocs/ms_workshop/example4.map&layer=states&layer=county&layer=fedland&layer=water&layer=mod_evi&mode=browse

This is what our application would have been like if we didn't use on-the-fly reprojection. For the next 30 minutes or so we will learn about on-the-fly reprojection capabilities of MapServer.

MapServer works with projected and unprojected (geographic) datasets. However, for MapServer to calculate scales and create scalebars properly, the dataset has to be projected in a grid-based coordinate projection system (i.e. UTM, Lambert, Albers). MapServer can reproject geographic data on-the-fly, with the help of the Proj.4 library. In addition, the GDAL library provides support for raster file reprojection. Be aware that on-the-fly reprojection will result in less than optimal performance for MapServer. Reprojection can be a compute-intensive process, and can slow MapServer's output creation substantially.

Having said that, on-the-fly reprojection can still be very useful—you can reproject your data to any projection you want. You might want to create an application where the client chooses the projection and MapServer generates the proper map for that particular client. If you want to create a WMS-compliant application, you must specify your spatial reference system (SRS). You'll need projection support for that SRS to be recognized by MapServer.

We can set our map file PROJECTION object using either "raw" Proj.4 parameters or specifying an EPSG reference number. This "reference system" was created by the European Petroleum Survey Group (EPSG, hence the name) and contains geodetic parameters for several projections. It's become a standard reference system, used in WMS applications. Unfortunately, most of the projections defined in the EPSG file are for smaller areas (larger scale maps) than the upper great lakes. You could use UTM projection for your application but the upper great lakes region cover two UTM zones. What I have done is what ESRI and other companies have done—create custom projection definitions.

For this application, you will use Lambert Azimuthal Equal-Area projection, a common CONUS projection, and UTM Zone 15 using WGS84 as spheroid.

Define custom EPSG projection definition.
Step 1. Edit "/usr/local/ms_lib/share/proj/epsg" using a text editor and append the following two lines at the end of the file:

```
# CLARKE1866/NAD27/Lambert Azimuthal Equal-Area (for Continental US)
<42301> +proj=laea +lat_0=45 +lon_0=-100 +ellps=clrk66 +datum=NAD27 +units=m no_defs <>
```

Save the file when done. That's all you need to do to add a custom projection in the epsg file.

Define your output PROJECTION object in the map file, example4.map.
Step 1. Edit the map file using a text editor and look for the PROJECTION keyword, just after the WEB object block. Check to make sure the line "init=epsg:40001" is enclosed by the keywords PROJECTION and END. The

commented text represents the alternative definition scheme for this custom projection (Lambert Azimuthal Equal-Area). If you don't want to mess with the EPSG file, you have no choice but to define your projection in this way.

Step 2. Add the following lines to each LAYER objects, before the CLASSITEM and CLASS definitions:

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

The reference number 4269 defines the geographic projection (latlon) with the GRS 1980 ellipsoid and NAD83 datum.

You define a different projection for input layers and for the output image to "tell" MapServer to use on-the-fly reprojection.

Step 3. You then have to replace the extents of both the main and reference map. Replace the two EXTENT lines with:

```
EXTENT 201621.496941 -294488.284025 1425518.020722 498254.511514
```

Save your map file and type this on your browser's URL box:
http://127.0.0.1/cgi-bin/mapserv.exe?map=C:/Program Files/Apache Group/Apache2/htdocs/ms_workshop/example4.map&layer=states&layer=county&layer=fedland&layer=water&layer=mod_evi&mode=browse

Step 4. Now edit example4.map once again and this time replace "init=epsg:40001" with "init=epsg:32615" on the main projection object. If you don't want to use the EPSG reference system, you can specify UTM projection like this:

```
PROJECTION
  "proj=utm"
  "zone=15"
  "ellps=WGS84"
  "datum=WGS84"
END
```

Step 5. Comment out the "geographic extent" line and uncomment the "UTM Zone 15 extent" line. Save your map and look at your application once again. This ends the projection section of the workshop.

Example 5: WMS-compliant Server application

For our final example today, you will create a WMS compliant server application and test it against CubeView, Cubewerx's WMS client application. If everything works properly, you should be able to view your data layers in CubeView.

We have already covered projections in the previous example so I won't talk much about it except to say that it is required in WMS application.

MapServer uses the METADATA object for WMS-specific parameters. We have to specify METADATA objects in two levels, one at the main map level and the other at the layer level. The METADATA object in the main level provides parameters for the entire application while the layer level object provides parameters for a particular layer only. In other words, some of the parameters in the main level are inherited by each layer (i.e. projection reference).

Let's work on your application.
Create a METADATA object at the MAP level.
Step 1. Open example5.map in a text editor and add the following two lines just before the PROJECTION object:

```
        METADATA
        END
```

This is how the METADATA object is defined. We will add the parameters between these two lines.

Step 2. Add a title to our application by inserting the following line in our METADATA object:

```
        WMS_TITLE "A WMS Server application"
```

Step 3. Add an abstract to our application by inserting the following line:

```
        WMS_ABSTRACT "This WMS application was created using MapServer 3.5, an open-
source web mapping tool."
```

Step 4. Add an access constraint statement by inserting the following line:

```
        WMS_ACCESSCONSTRAINTS none
```

This isn't a required parameter but you might need it in your applications.

Step 5. Add an online resource (URL) descriptor by inserting the following line:

```
        WMS_ONLINERESOURCE "http://localhost/cgi-bin/mapserv.exe?map=C:/Program
Files/Apache Group/Apache2/htdocs/ms_workshop/example5.map"
```

This provides information on how WMS clients can access this application.

Step 6. Add our projection info by inserting the following line:

```
        WMS_SRS "EPSG:4269 EPSG:32615 EPSG:42103 EPSG:42303"
```

Here, we are providing several projection options for clients.

Step 7. Add some bounding coordinates to our application by inserting the following line:

```
        WMS_LATLONBOUNDINGBOX "-97.238976 41.619778 -82.122902 49.38562"
```

Step 8. Save your map file. You've finished defining your initial WMS parameters and it should look like this:

```
 METADATA
   WMS_TITLE "MapServer 3.6 Tutorial: Section 5, Example 3"
   WMS_ABSTRACT "This WMS client application is provided as an example for beginning
MapServer users."
   WMS_ACCESSCONSTRAINTS none
   WMS_ONLINERESOURCE "http://localhost/cgi-bin/mapserv.exe?map=C:/Program
Files/Apache Group/Apache2/htdocs/ms_workshop/example5.map"
   WMS_SRS "EPSG:4269 EPSG:32615 EPSG:42103 EPSG:42303"
   WMS_LATLONBOUNDINGBOX "-97.238976 41.619778 -82.122902 49.38562"
 END
```

Create a METADATA object at the LAYER level.
I will provide step-by-step instructions for the first layer and will let you do the same thing for the rest of the layers.
Step 1. Open example5.map, if it isn't already, for editing. Insert the METADATA object just before the PROJECTION object:

```
        METADATA
        END
```

Step 2. Add the data layer title to the object by inserting the following line:

```
WMS_TITLE "Upper Great Lakes states"
```

Step 3. Add a data layer abstract by inserting the following line:

```
        WMS_ABSTRACT "This is the state boundary layer for Michigan, Minnesota and
Wisconsin. See http://www.nationalatlas.gov/statesm.html for more information."
```

This is really all you need for each layer but we want to add more details so…
Step 4. Add the projection info for the data layer by inserting the following line:

```
        WMS_SRS "EPSG:4269"
```

Step 5. Create a geographic bound by inserting the following line:

```
        WMS_LATLONBOUNDINGBOX "-97.238976 41.619778 -82.122902 49.38562"
```

We've created the METADATA object for the first layer. Now you can go ahead and add it on each of the other layers, changing the title and abstract as appropriate.

Step 6. Save when done and view the application by typing the following on you browser URL box:
http://127.0.0.1/cgi-bin/mapserv.exe?map=C:/Program Files/Apache

Group/Apache2/htdocs/ms_workshop/example5.map&VERSION=1.1.0&REQUEST=GetCapabilities

Save the resulting file when asked. Call it something like "mywmsapp.xml". Open it using your favorite browser, or your text editor.

Now, let's test your server against one of these clients:
http://www.cubewerx.com/demo/cubeview/cubeview.cgi or
http://mapserver.refractions.net/phpwms/phpwms-rel/ or
http://mapserver.refractions.net/phpwms/phpwms-cvs/

To learn more about WMS-compliant MapServer applications, please visit http://mapserver.gis.umn.edu/doc/wms-server-howto.html and http://mapserver.gis.umn.edu/doc/wms-client-howto.html.

Example 6: The Java mapimage applet.
The HTML template file "example6.html" includes a JavaScript code for a more advanced map interface control . This JavaScript calls a Java applet, called "mapplet", that allow web users to draw rectangular boxes on the map. Visit http://mapserver.gis.umn.edu/doc/mapplet-howto.html for instructions on how to use the mapplet with your MapServer applications.

To see how this mapplet works, point your browser to http://localhost/cgi-bin/mapserv.exe?map=C:/Program Files/Apache Group/Apache2/htdocs/ms_workshop/example6.map&layer=relief&layer=states. You can edit example6.html to see how the mapplet is implemented in this example.

That's all folks!

Prepared by:
Jamie Smedsmo, Ranga Raju Vatsavai and Perry Nacionales

Version 1.0.2
Last updated by Perry, 20030606.