

Setting Up a MapXtreme 2004 WMS Server

This document describes how to set up a WMS server for use with your own MapXtreme 2004-created WMS client or by an application that already has WMS client capabilities. For example, this document demonstrates how to set up a WMS server for MapInfo Professional version 7.5 or later. Refer to the Open Geospatial Consortium (OGC) specifications for more details about the requirements of a WMS or a service compatible client (<http://www.opengeospatial.org/docs/01-068r3.pdf>).

In this document...

◆ Introduction	2
◆ Prerequisites	2
◆ Setting up the WMS Service Under IIS	2
◆ Customizing Your WMS Service	12
◆ Testing your WMS Service Using MapInfo Professional	17
◆ Troubleshooting your WMS Setup	20
◆ Accessing Network Drives	23

Introduction

A Web Map Service (WMS) allows software clients to reference map images over the Internet or a private intranet. MapInfo WMS Server and Client implementations are based on the WMS 1.1.1 OGC specification. Using http requests, a WMS server provides GIS data to a client that displays this data as an image. Each image can act as a separate layer. The images can be provided as GIF, JPG, PNG, and several other image formats. The images rendered can be made of layers that are hierarchical—an image can be served that is made up of a layer that is a collection of other layers. Since WMS renders layers, the characteristics that modify the view of layers (styles, coordinate systems, etc.) can also be included in the layers served, allowing a level of customization of the images provided.

Please refer to the OGC WMS specification (<http://www.opengeospatial.org/docs/01-068r3.pdf>) for more information about developing your own client. For the purposes of this document, we are using MapInfo Professional, version 7.8 as a WMS client.

Prerequisites

As with most modern software, and particularly server-based software, these systems rely on a certain number of operating system components to be in place before they will work. For MapXtreme 2004 to run as a Web Map Service you need the following installed:

- MapXtreme 2004
- Internet Information Services (IIS)
- ASP.NET v1.1.4322

Note: The name of your Windows directory varies depending on the version of the operating system upon which you are running. For the remainder of this document we will refer to this directory as C:\WINNT.

If you have the latest version of MapXtreme 2004 and installed it in the manner described in the installation instructions, the necessary components of ASP.NET should already be installed and configured properly on your computer.

Setting up the WMS Service Under IIS

MapXtreme 2004 ships with a set of configuration files that allow you to set up a WMS server with very few modifications. This section describes how to use these files to get a WMS server up and running quickly. The following section explains this process again using custom data so that you can see how to modify these sample files to match your own server configuration.

Before using the method described in this document, we recommend you at least read Chapter 20 (WMS and WFS) of the *MapXtreme 2004 Developers Guide*. This chapter provides two methods for deploying WMS on the IIS server. The method described in this document is based on the

second method “Configuring the WMS Server and IIS without Visual Studio .NET”. We provide these instructions to assist even the newest GIS Administrator in setting up and running WMS services.

Note: There is a minor flaw in the first method described in Chapter 20. In the “Using Visual Studio” method, point 4 says to add the MapInfo.Wms.Server.dll as a reference to the ASP project. This step fails because Visual Studio cannot locate the .dll. A MapInfo Knowledge Base entry on this point recommends using the manual method described below to resolve this problem.

A high level description of the required steps to set up a Web Map Service is as follows:

1. Verify the MapInfo.Wms.Server software version number.
2. Create the folder where the configuration files will reside.
3. Copy the sample configuration files from your MapXtreme 2004 CD.
4. Modify the configuration files to match your environment.
5. Make this folder available as a Web Share directory.
6. In IIS, change the security properties of the new virtual directory.
7. Test the configuration.

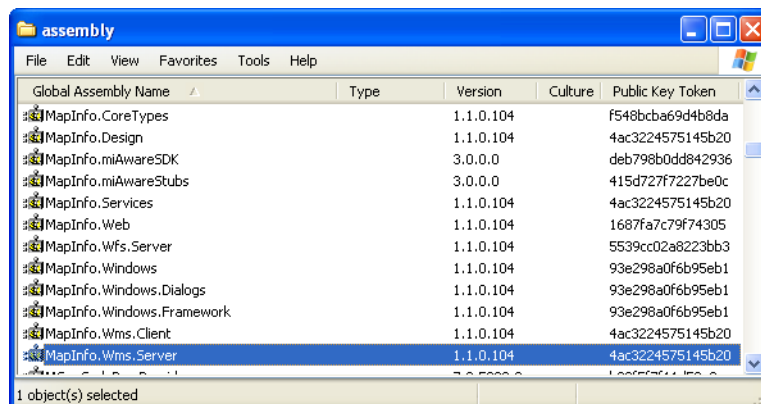
Each of these steps is described in detail in the next sections.

These steps may seem lengthy but in reality it’s all very straightforward. For example, if you only had a few tables to publish on your WMS server, you could have the entire process done in only a few minutes.

Step 1: Verify the Software Version

The version number of the MapInfo software needs to be verified in order to include the correct information in the Web.config configuration file. To check the version number, follow this procedure:

1. Select **RUN...** from the Windows **START** menu and enter the following command:
`c:\WINNT\assembly`
 The **ASSEMBLY** directory is displayed. This window lists all the .NET global assemblies you have installed.
2. Locate the MapInfo.Wms.Server on the list and note the numbers in the Version column. These numbers must be correctly specified in the Web.config file which we will soon be examining. (In the example below, the Version is 1.1.0.104.)



Step 2: Create a Directory to Hold The Configuration Files

Create a directory on the server where you intend to store the WMS configuration files. The actual name does not matter. For this tutorial we'll use c:\WMS as that is what is already specified in the sample configuration files.

Step 3: Copy the Sample Configuration Files From Your MapXtreme 2004 CD

Copy the two sample WMS configuration files from the MapXtreme 2004 CD. The files are located in the folder WMS Configuration Files and are called Web.config and WMSsample.xml. These files are already configured to work using the sample data provided with MapXtreme 2004.

Step 4: Modify the Configuration Files to Match Your Environment

As shown above, there are two configuration files used in setting up your WMS server. The file Web.config requires less modification as its purpose is to define how the ASP process is handled. One of these functions is to reference the WMSsample.xml file. The WMSsample.xml file defines the data sources and layer definitions you wish to serve.

Open the Web.config file in any text editor.

You need to modify this file in two places. The first is where the Web.config file points to the second configuration file. In the sample file provided with MapXtreme 2004, this line reads as:

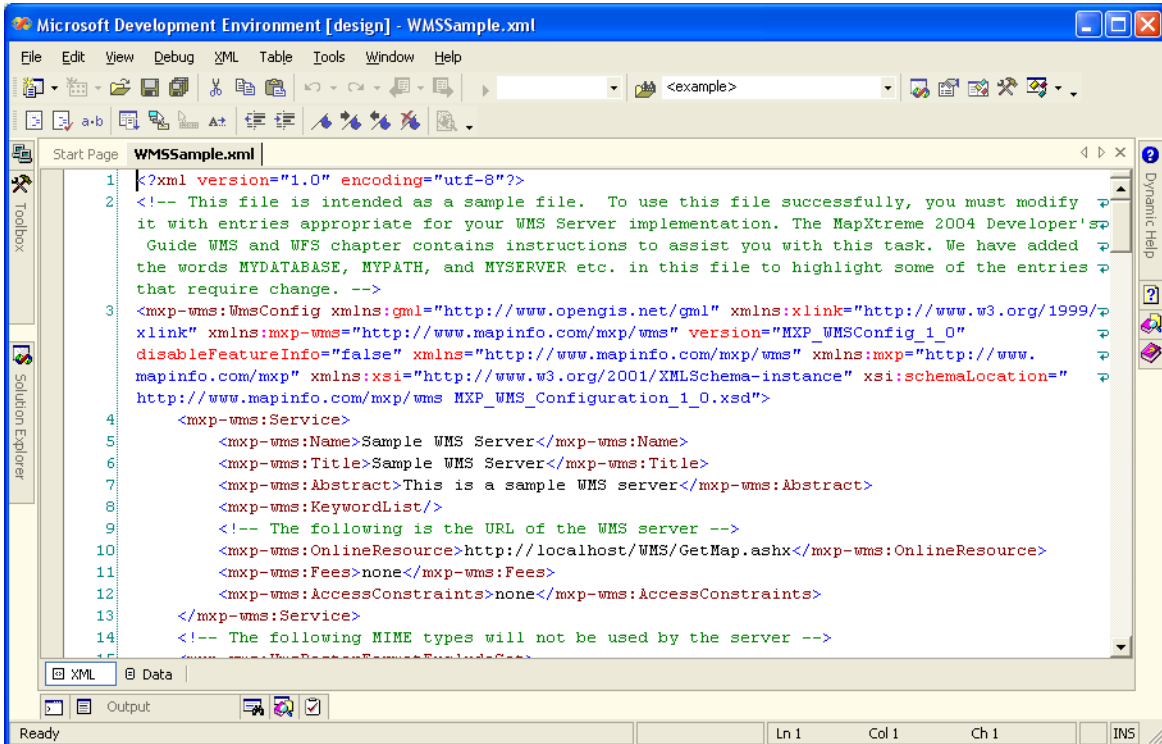
```
<configuration>
  <appSettings>
    <add key="configFile" value="C:\wms\WMSsample.xml" />
```

Since we are using the default implementation, we only need to verify that this path matches our set up.

Second, modify the file (shown in bold below) that specifies the software version we verified in **Step 1: Verify the Software Version.**

```
<system.web>
  <httpHandlers>
    <add verb="GET,POST" path="*.ashx" type="MapInfo.Wms.WmsHttpHandler,
      MapInfo.Wms.Server, Version=1.1.0.104, Culture=neutral,
      PublicKeyToken=4ac3224575145b20"/>
  </httpHandlers>
```

Save the changes to this file, then open the second configuration file (WMSsample.xml) in your editor. Note that this file is an XML file and is much larger than the first configuration file.



The first lines to modify are the values for the `<mxp-wms:Name>`, `<mxp-wms:Title>`, and `<mxp-wms:Abstract>` elements defined for the WMS server. The use of the values specified in this section is completely dictated by the client you are using.

Next, change the location of your WMS Virtual Directory (more on this below). For now, we will use the web URL `http://YourMachineName/wms` to match the folder name you created earlier. Later this should be changed to a proper URL definition as required by your Web Administrator.

Note: For the remainder of this paper `MITestServer` is used as the server for any examples. Since there is no machine by that name, the screenshots display `MITestServer` as the server. Make sure to replace your own server name in any place where either `MITestServer` or `MITestServer` may appear.

The lines to change are bolded in the code snippet below.

```

<?xml version="1.0" encoding="utf-8"?>
<!-- This file is intended as a sample file. To use this file successfully,
you must modify it with entries appropriate for your WMS Server
implementation. The MapXtreme 2004 Developer's Guide WMS and WFS chapter
contains instructions to assist you with this task. We have added the words
MYDATABASE, MYPATH, and MYSERVER etc. in this file to highlight some of the
entries that require change. -->
<mxp-wms:WmsConfig xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:mxp-wms="http://
www.mapinfo.com/mxp/wms" version="MXP_WMSConfig_1_0"
disableFeatureInfo="false" xmlns="http://www.mapinfo.com/mxp/wms"
xmlns:mxp="http://www.mapinfo.com/mxp" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.mapinfo.com/mxp/wms
MXP_WMS_Configuration_1_0.xsd">
  <mxp-wms:Service>
    <mxp-wms:Name>Sample WMS Server</mxp-wms:Name>
    <mxp-wms:Title>Sample WMS Server</mxp-wms:Title>

```

```

<mxp-wms:Abstract>This is a sample WMS server</mxp-wms:Abstract>
<mxp-wms:KeywordList/>
<!-- The following is the URL of the WMS server -->
<mxp-wms:OnlineResource>http://MITestServer/WMS/GetMap.ashx
  </mxp-wms:OnlineResource>
<mxp-wms:Fees>none</mxp-wms:Fees>
<mxp-wms:AccessConstraints>none</mxp-wms:AccessConstraints>
</mxp-wms:Service>
<!-- The following MIME types will not be used by the server -->
<mxp-wms:WmsRasterFormatExcludeSet>
  <mxp-wms:Format>image/jpeg2000</mxp-wms:Format>
  <mxp-wms:Format>image/photoshop</mxp-wms:Format>
  <mxp-wms:Format>image/jp2</mxp-wms:Format>
</mxp-wms:WmsRasterFormatExcludeSet>

```

Next you register your mapping layers. The following includes the set of files installed with your version of MapXtreme 2004.

```

<mxp-wms:DataSourceDefinitionSet>
  <!-- The following data sources reference local TAB files -->
  <mxp:TABFileDataSourceDefinition id="USA">
    <mxp:DataSourceName>USA</mxp:DataSourceName>
    <mxp:FileName>C:\Program Files\MapInfo\MapXtreme\6.1\Samples\Data\
      USA.TAB</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="US_HIWAY">
    <mxp:DataSourceName>US_HIWAY</mxp:DataSourceName>
    <mxp:FileName>C:\Program Files\MapInfo\MapXtreme\6.1\Samples\Data\
      US_HIWAY.TAB</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="US_CNTY">
    <mxp:DataSourceName>US_COUNTY</mxp:DataSourceName>
    <mxp:FileName>C:\Program Files\MapInfo\MapXtreme\6.1\Samples\Data\
      US_CNTY.TAB</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="US_CAPS">
    <mxp:DataSourceName>US_CAPS</mxp:DataSourceName>
    <mxp:FileName>C:\Program Files\MapInfo\MapXtreme\6.1\Samples\Data\
      USA_CAPS.TAB</mxp:FileName>
  </mxp:TABFileDataSourceDefinition>
  ...
</mxp-wms:DataSourceDefinitionSet>

```

For each `<mxp:TABFileDataSourceDefinition>` element, specify a value for the `id` attribute and for the `<mxp:DataSourceName>` element. The `id` value must be a simple string beginning with an alpha character. The `<mxp:FileName>` element in each `<mxp:TABFileDataSourceDefinition>` element should contain the path to the actual file. Ensure this value is correct.

Finally, in the WMS Layer List section, define the actual Layer List as used in the client application to show which layers are available. Layers can be nested so that by requesting a parent layer, all the child layers are included in the response. See [WMSConfig.xml on page 13](#) for an example of this kind of configuration.

```

<WmsLayer>
  <Name>US_DATA</Name>
  <Title>US Map</Title>
  <Abstract>General US Geometry Data</Abstract>
  <SRSNameSet/>

```

```

<WmsStyleSet/>
<WmsLayerList>
  <WmsLayer>
    <Name>USALayer</Name>
    <Title>USA</Title>
    <Abstract>USA Regions Layer</Abstract>
    <SRSNameSet>
      <mxp:SRSName>epsg:4326</mxp:SRSName>
    </SRSNameSet>
    <WmsStyleSet>
      <WmsStyle>
        <Name>myName1</Name>
        <Title>myTitle</Title>
        <Abstract>myAbstract</Abstract>
        <mxp:AreaStyle/>
      </WmsStyle>
    </WmsStyleSet>
    <mxp:FeatureLayer id="USALayer" name="USA" alias="USA">
      <mxp:DataSourceRef ref="USA"/>
    </mxp:FeatureLayer>
  </WmsLayer>
  <WmsLayer>
    <Name>HighwayLayer</Name>
    <Title>US Hiways</Title>
    <Abstract>US Line Layer</Abstract>
    <SRSNameSet>
      <mxp:SRSName>epsg:4326</mxp:SRSName>
    </SRSNameSet>
    <WmsStyleSet>
      <WmsStyle>
        <Name>myName4</Name>
        <Title>myTitle</Title>
        <Abstract>myAbstract</Abstract>
        <mxp:AreaStyle/>
      </WmsStyle>
    </WmsStyleSet>
    <mxp:FeatureLayer id="US_HIWAYLayer" name="US_HIWAY"
      alias="US_HIWAY">
      <mxp:DataSourceRef ref="US_HIWAY"/>
    </mxp:FeatureLayer>
  </WmsLayer>
  ...

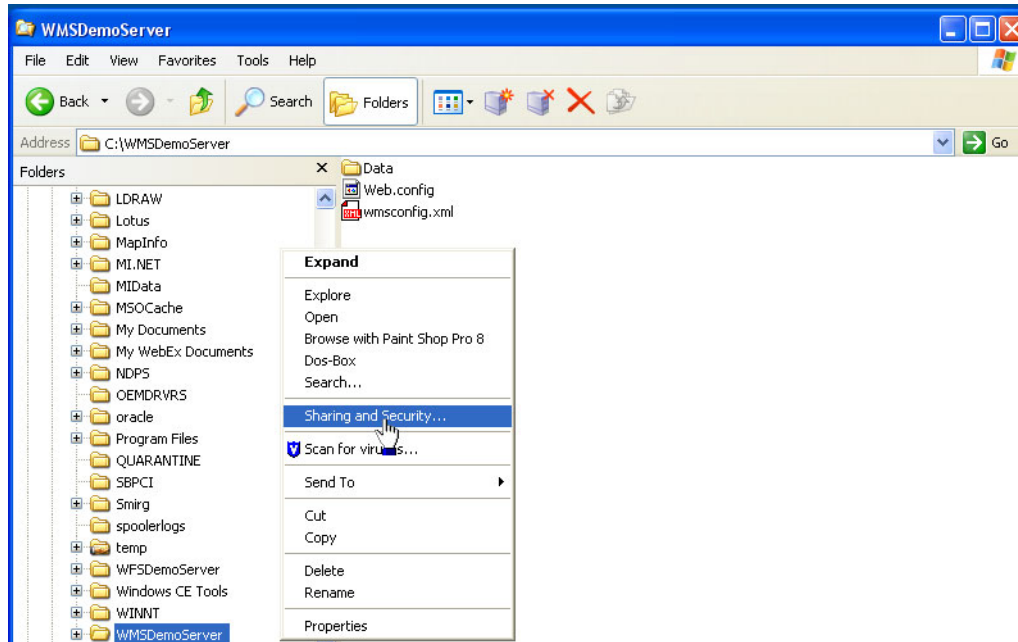
```

Note that the values in the `<mxp:DataSourceRef>` elements (bolded above) refer to the `<mxp:TABFileDataSourceDefinition>` id attribute value supplied in the table definitions section, defined above. Make sure these values match.

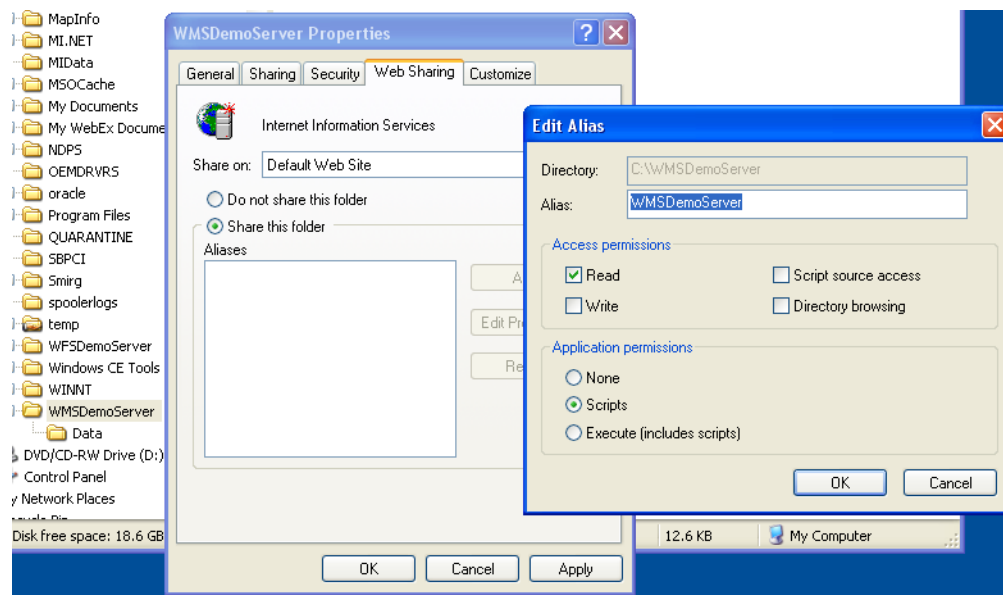
Once you have made all these changes to the two configuration files, save your work and close them.

Step 5: Make this Folder Available as a Web Share Directory

Next you must make the folder in which you have the configuration files available for use from the web. Do this by right-clicking on the folder name and selecting **SHARING AND SECURITY...**



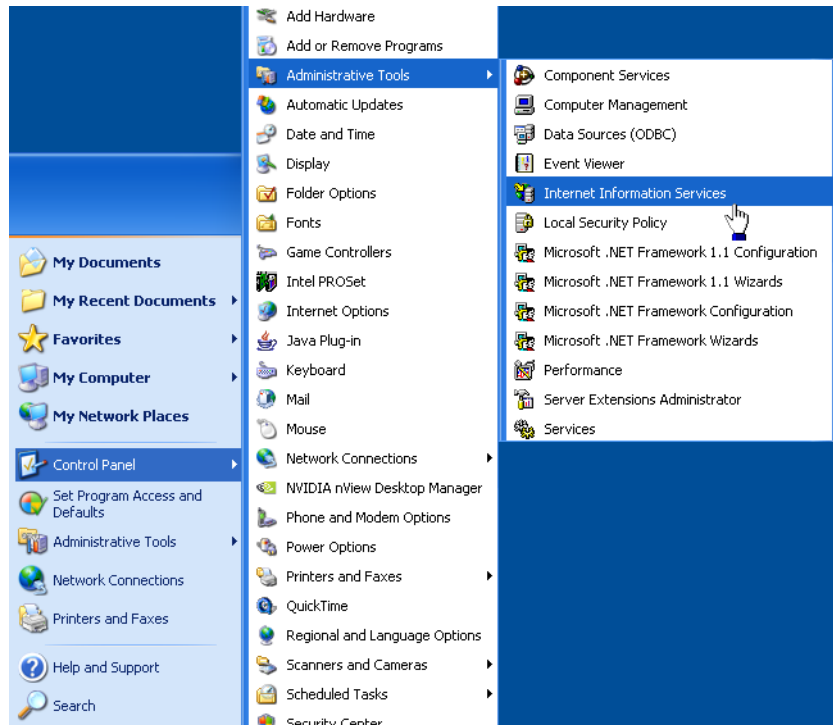
On the **WEB SHARING** tab, select the radio button for **SHARE THIS FOLDER**. The **EDIT ALIAS** dialog box is displayed. Here you can change the Alias name if you choose, then click **OK** to close the dialog box. The **WEB SHARING** tab of the **PROPERTIES** dialog box is updated to show the Alias name to be shared. Click **OK** to close the **PROPERTIES** dialog box.



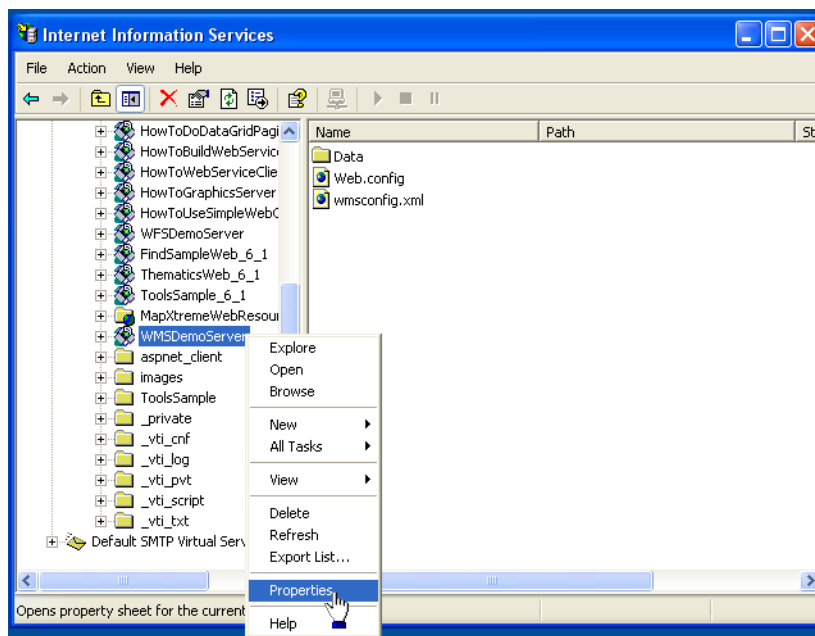
Step 6: Change the IIS Security Properties

The last step of the set up is to set the security properties of your new WMS virtual directory. This task is done from the IIS administration; you change the security properties of your WMS virtual directory. Setting the folder to `Anonymous` means you do not have to worry about user name and passwords.

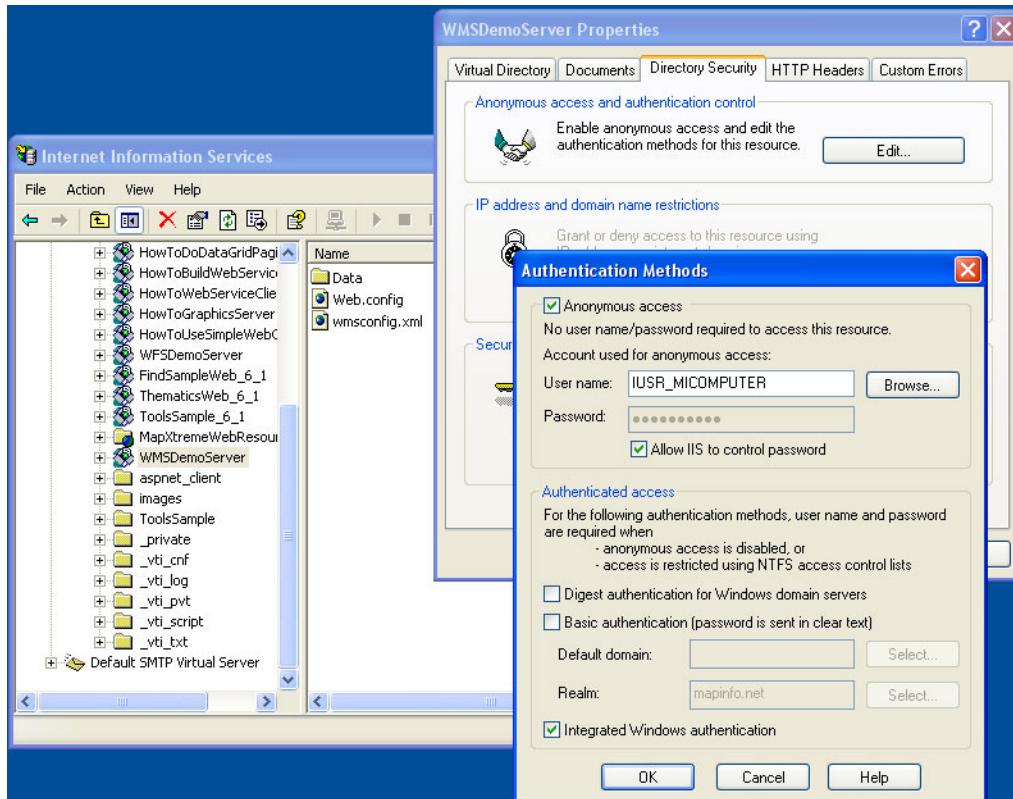
Start Internet Information Services (IIS) from the Windows Start menu (**START > CONTROL PANEL > ADMINISTRATIVE TOOLS > INTERNET INFORMATION SERVICES**).



Expand the Default Web Site item so you can see the WMS virtual directory you just created. Right-click on the WMS directory and click the **PROPERTIES** menu item.



Next select the **DIRECTORY SECURITY** tab then click the **EDIT** button at the top right. In the **AUTHENTICATION METHOD** dialog box, select the **ANONYMOUS ACCESS** check box to allow WMS service users to skip the username/password process.



Step 7: Test the Configuration

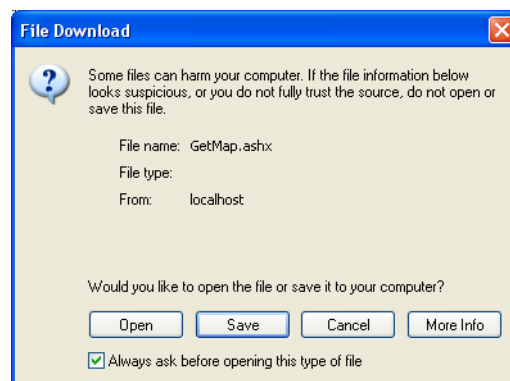
Now that everything is enabled and configured properly, let's do a simple query using Internet Explorer as a client. Internet Explorer will not show us a map, but we will be able to see a good XML response from the server.

Note: You can also use a text editor to display the resulting XML. The result is not as readable, but does display the full XML content.

1. Launch your browser and type in the following URL:

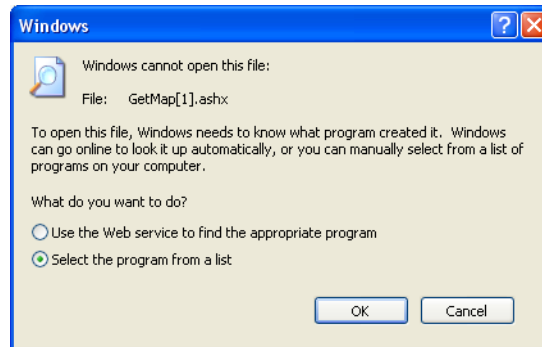
```
http://MITestServer/WMS/
GetMap.ashx?request=GetCapabilities&service=WMS&version=1.1.1
```

A **FILE DOWNLOAD** dialog box is displayed.



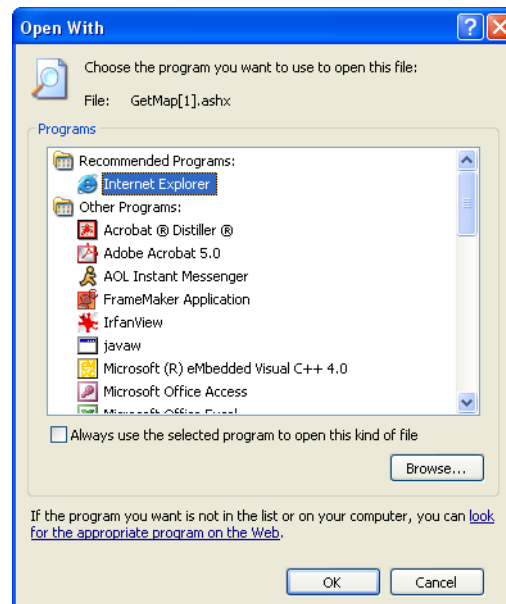
2. In **FILE DOWNLOAD** dialog box, click **OPEN**.

This dialog box appears stating Windows cannot open this file:



3. Choose **SELECT THE PROGRAM FROM A LIST** and click **OK**.

The **OPEN WITH** dialog box is displayed showing a list of available applications which can open the file.



4. Choose Internet Explorer from the top of the list and click **OK**.

Note: You can also use a text editor to display the resulting XML. The result is not as readable, but does display the full XML content.

A new Internet Explorer window opens displaying an XML file that resembles the screen image below.

```

<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
- <WMT_MS_Capabilities version="1.1.1">
- <Service>
  <Name>Sample WMS Server</Name>
  <Title>Sample WMS Server</Title>
  <Abstract>This is a sample WMS server</Abstract>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
    xlink:href="http://localhost/WMS/GetMap.ashx" />
  <Fees>none</Fees>
  <AccessConstraints>none</AccessConstraints>
</Service>
- <Capability>
- <Request>
  - <GetCapabilities>
    <Format>application/vnd.ogc.wms_xml</Format>
    - <DCPtype>
      - <HTTP>
        - <Get>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="simple" xlink:href="http://localhost/WMS/GetMap.ashx" />
        </Get>
      </HTTP>
    </DCPtype>
  </GetCapabilities>
- <GetMap>

```

This demonstrates that the WMS server is configured properly and running well. If you do not see the proper result, double check the steps above and retry it. Then refer to [Troubleshooting your WMS Setup on page 20](#) for a more detailed list of things to check.

Later we will create a map based on data from the WMS Server using MapInfo Professional as a WMS Client to demonstrate the WMS Server/Client relationship.

Customizing Your WMS Service

Now that we have our basic service running it is time to customize it to match your data and setup. For this section we are going to use the same sample configuration files from the MapXtreme 2004 CD that we copied to our WMS directory. This time we are going to configure our files to serve a set of StreetPro data files of New York State.

We will step through the same procedure as in the last section, only using values to match our new server. The steps are as follows:

1. Create a directory to store the configuration files.
2. Copy the sample files from the MapXtreme 2004 CD to the new directory created in [step 1](#).
3. Modify the files to match our new environment.
4. Make this folder available as a Web Share directory.
5. In IIS, change the security properties of the new virtual directory.

6. Test the configuration.

Copy the Sample Files

Copy the sample files to a new directory, C:\WMSDemoServer. Rename WMSSample.xml to WMSConfig.xml to distinguish it from the previous example.

Modify the Files to Match Our New Environment

For this part of the tutorial we are going to point to a data sub-directory of C:\WMSDemoServer which contains a few different data files of New York State.

Web.config

We can modify the Web.config file to have the correct software version number and also point to our WMSConfig.xml file:

```
<configuration>
  <appSettings>
    <add key="configFile" value="C:\WMSDemoServer\WMSConfig.xml" />
    ...
  <system.web>
    <httpHandlers>
      <add verb="GET,POST" path="*.ashx" type="MapInfo.Wms.WmsHttpHandler,
        MapInfo.Wms.Server, Version=1.1.0.104, Culture=neutral,
        PublicKeyToken=4ac3224575145b20"/>
    </httpHandlers>
```

WMSConfig.xml

To begin the file modification we need to match our <mxp-wms:Service> elements of the WMSConfig.xml file to our new configuration. We need to specify a name, title, and abstract for our new site. We also need to update the value of the <mxp-wms:OnlineResource> element for the new location of our server data.

Note: The elements with the namespace (prefix) of `mxp:` are specific to MapInfo's implementation of WMS and do not appear in the OGC specification.

See the code snippet below for details. The bold portions are the values that need to be updated:

```
<mxp-wms:Service>
  <mxp-wms:Name>NYS WMS Server</mxp-wms:Name>
  <mxp-wms:Title>New York State Data WMS Server</mxp-wms:Title>
  <mxp-wms:Abstract>This is WMS server of New York State data
    </mxp-wms:Abstract>
  <mxp-wms:KeywordList/>
  <!-- The following is the URL of the WMS server -->
  <mxp-wms:OnlineResource>http://MITestServer/WMSDemoServer/GetMap.ashx
    </mxp-wms:OnlineResource>
  <mxp-wms:Fees>none</mxp-wms:Fees>
  <mxp-wms:AccessConstraints>none</mxp-wms:AccessConstraints>
</mxp-wms:Service>
```

Note: When taking your system live, be sure that you change `http://MITestServer/WMSDemoServer/` to the URL specified by your Web Administrator.

Next we need to customize the file to see our served layers. For this example we are replacing the default layers with layers containing New York State data from our StreetPro product. We are going to serve five layers, Streets, Water, Parks, Cities, and Counties. The files have been copied into a data subdirectory of WMSDemoServer. The text below shows the relevant portion of our new WMSConfig.xml.

```
<mxp-wms:DataSourceDefinitionSet>
  <!-- The following data sources reference local TAB files -->
  <mxp:TABFileDataSourceDefinition id="Streets">
    <mxp:DataSourceName>Streets</mxp:DataSourceName>
    <mxp:FileName>C:\WMSDemoServer\Data\NYd.tab</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="Water">
    <mxp:DataSourceName>Water</mxp:DataSourceName>
    <mxp:FileName>C:\WMSDemoServer\Data\NYwb.tab</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="Parks">
    <mxp:DataSourceName>Parks</mxp:DataSourceName>
    <mxp:FileName>C:\WMSDemoServer\Data\NYpk.tab</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="Cities">
    <mxp:DataSourceName>Cities</mxp:DataSourceName>
    <mxp:FileName>C:\WMSDemoServer\Data\NYcb.tab</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  <mxp:TABFileDataSourceDefinition id="Counties">
    <mxp:DataSourceName>Counties</mxp:DataSourceName>
    <mxp:FileName>C:\WMSDemoServer\Data\NYcy.tab</mxp:FileName>
  </mxp:TABFileDataSourceDefinition >
  ...
</mxp-wms:DataSourceDefinitionSet>
```

We then need to modify the WMSLayer section of the XML as we did before.

Note: This configuration file segment below includes nested layers allowing the client to display all of the children layers by requesting the parent layer, NYSDData. In order for this to function the parent layer needs to have a coordinate system defined (using the <SRSNameSet> element).

```
<WmsLayer>
  <Name>NYSDData</Name>
  <Title>NY State Data</Title>
  <Abstract>General NY Geometry Data</Abstract>
  <SRSNameSet>
    <mxp:SRSName>epsg:4326</mxp:SRSName>
  </SRSNameSet>
  <WmsStyleSet/>
  <WmsLayerList>
    <WmsLayer>
      <Name>Streets</Name>
      <Title>NY Streets</Title>
      <Abstract>All Streets in NYS</Abstract>
      <SRSNameSet>
        <mxp:SRSName>epsg:4326</mxp:SRSName>
      </SRSNameSet>
      <WmsStyleSet>
        <WmsStyle>
          <Name>myName4</Name>
          <Title>myTitle</Title>
          <Abstract>myAbstract</Abstract>
```

```

        <mxp:AreaStyle/>
    </WmsStyle>
</WmsStyleSet>
<mxp:FeatureLayer id="NYStreets" name="Streets" alias="Streets">
    <mxp:DataSourceRef ref="Streets"/>
</mxp:FeatureLayer>
</WmsLayer>
...
</WmsLayerList>
</WmsLayer>

```

There are several different values for each configured layer to fill in when editing this file. There are no set guidelines about how these values are used in a client application. The specific use of these values is dictated by the client reading the information. Refer to the OGC WMS specification (<http://www.opengeospatial.org/docs/01-068r3.pdf>) for details on how to implement the use of these values.

An example of this is the two elements `<Name>` and `<Title>`. According to the OGC WMS Specification, the Name value is to be used by machine-to-machine communication, while the Title value is used for human interaction. The Name value is what the client would request from the server, while the Title value is what would be displayed to the user. Compare the code above with the screen image displayed after [step 4](#).

The following list describes each of the elements in the WmsLayer portion of the configuration file and how they can be used:

Note: The elements with the namespace (prefix) of `mxp:` are specific to MapInfo's implementation of WMS and do not appear in the OGC specification.

<WmsLayerList>

This element is the root element (the element that holds all the other elements) for the list of layers.

<WmsLayer>

This element wraps a particular layer and its children layers.

<Name>

The value of this element is typically the name used for machine-to-machine communication.

Example: `<Name>NYSStreets</Name>`

<Title>

The value of this element is to provide a user-friendly description of the layer.

Example: `<Title>New York State Streets</Title>`

<Abstract>

The abstract is a longer narrative description of an object.

Example: `<Abstract>All streets and roads in New York State</Abstract>`

<SRSNameSet>

This element groups `<mxp:SRSName>` elements (see below).

<mxp:SRSName>

This element specifies the coordinate system to be used for the particular layer. If this value is empty, the layer inherits the <mxp:SRSName> value from the parent layer. A particular layer can have multiple <mxp:SRSName> elements. Any duplicates are ignored.

Example: <mxp:SRSName>epsg:4326</mxp:SRSName>

<WmsStyleSet>

This element acts as a container for individual <WmsStyle> elements.

<WmsStyle>

Each <WmsStyle> contains a <Name>, <Title>, and <Abstract> element that describe a particular style created in your workspace. Then the specifics of the style are entered (this can be copied from a MapInfo workspace file—*.mws). Make sure to prepend the mxp: namespace prefix for the element.

Example:

```
<WmsStyleSet>
  <WmsStyle>
    <Name>StreetStyle</Name>
    <Title>Street Style</Title>
    <Abstract>This is the line style for displaying streets.</Abstract>
    <mxp:LineStyle stroke="black" width="1"
      width-unit="mapinfo:image size pixel">
      <Pen>mapinfo:pen 2</Pen>
    </mxp:LineStyle>
  </WmsStyle>
</WmsStyleSet>
```

<mxp:FeatureLayer>

This element identifies the layers that are rendered. Each <mxp:FeatureLayer> element must have a unique value for its id attribute. The other two attributes in this element, name and alias, must have values, but what those values are have no impact on the WMS.

Example: <mxp:Featurelayer id="NYStreets" name="Streets" alias="streets">

mxp:DataSourceRef

This empty element has a single attribute, ref, which must match a layer (<mxp:TABFileDataSourceDefinition> element) specified in the <mxp:DataSourceDefinitionSet> section of the configuration file.

Example: <mxp:DataSourceRef ref="Streets"/>

Make This Folder Available as a Web Share Directory

This step follows the identical procedure as already described in [Step 5: Make this Folder Available as a Web Share Directory on page 7](#).

In IIS, Change the Security Properties of the New Virtual Directory

This step follows the identical procedure as already described in [Step 6: Change the IIS Security Properties on page 8](#).

Test the Configuration

To ensure our configuration is working properly, let's send a sample request using Internet Explorer. Type the following URL into your browser and then follow the rest of the steps specified in **Step 7: Test the Configuration on page 10**.

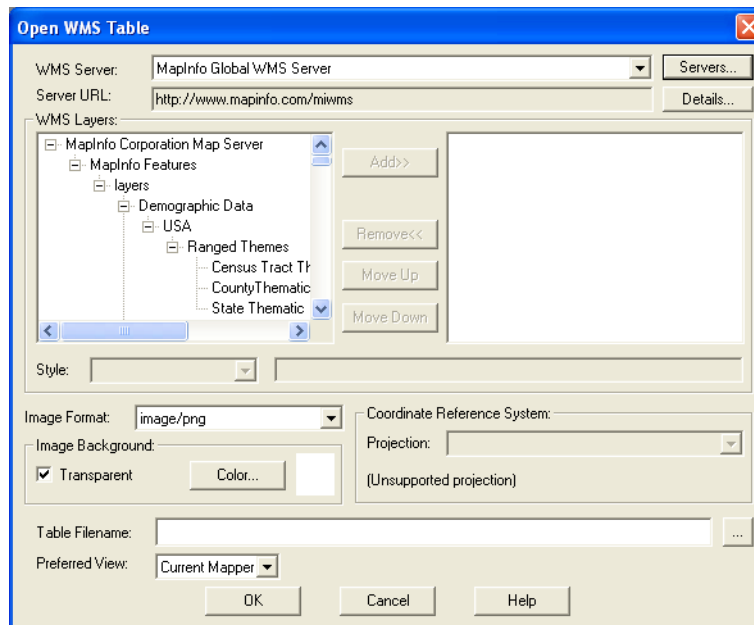
```
http://MITestServer/WMSDemoServer/
GetMap.ashx?request=GetCapabilities&service=WMS&version=1.1.1
```

You should see a similar result to the image in the above-referenced section. In the next section we use this same WMS setup using MapInfo Professional as the client.

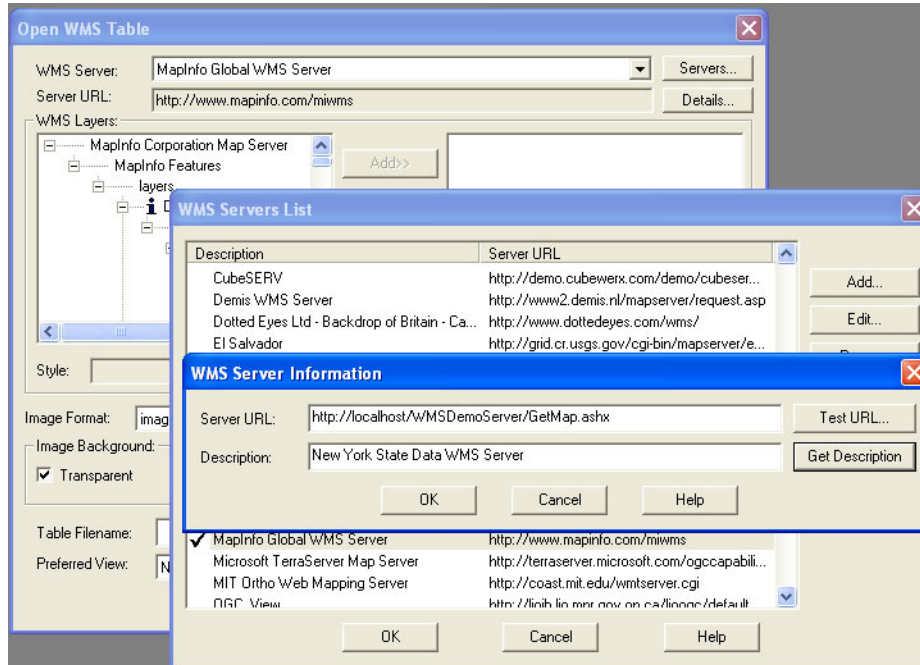
Testing your WMS Service Using MapInfo Professional

Now that the WMS Server is running, we can use MapInfo Professional as a WMS client to create the graphic files provided by the server.

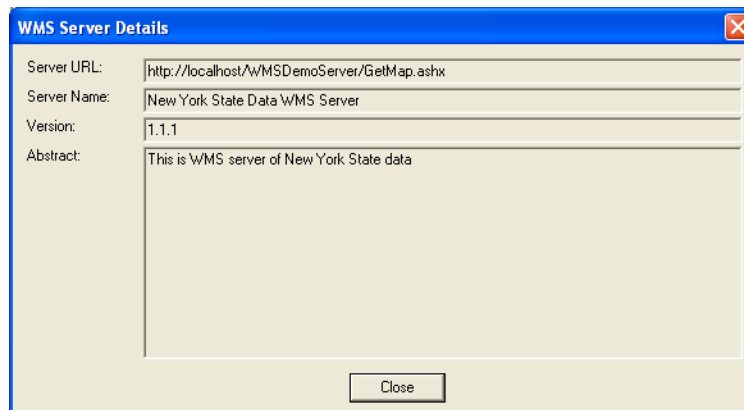
1. Start MapInfo Professional (7.5 or later) then use the **FILE > OPEN WEB SERVICE > OPEN WMS** command. This displays the **OPEN WMS TABLE** dialog box. It may take a few seconds to display the entire list as the application needs to download the default WMS server details from the web (in this case the MapInfo Global WMS Server).



2. Click the **SERVERS** button to edit the list of available servers and add your own service. Next click the **ADD** button and enter the address of your own WMS Service.



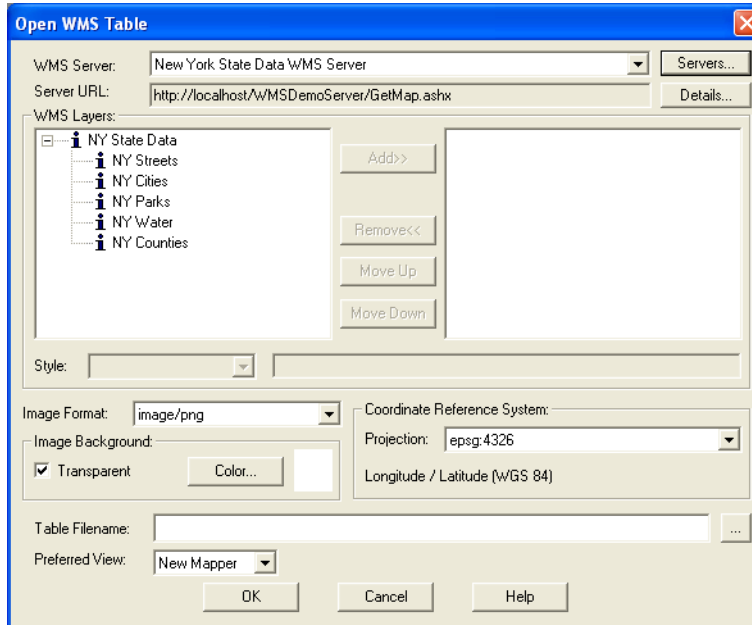
3. To ensure that you entered the URL address correctly, click the **TEST URL** button. If the test is successful you should see something similar to the following dialog box. Note that the Server Name and Abstract match the names you supplied in the configuration file.



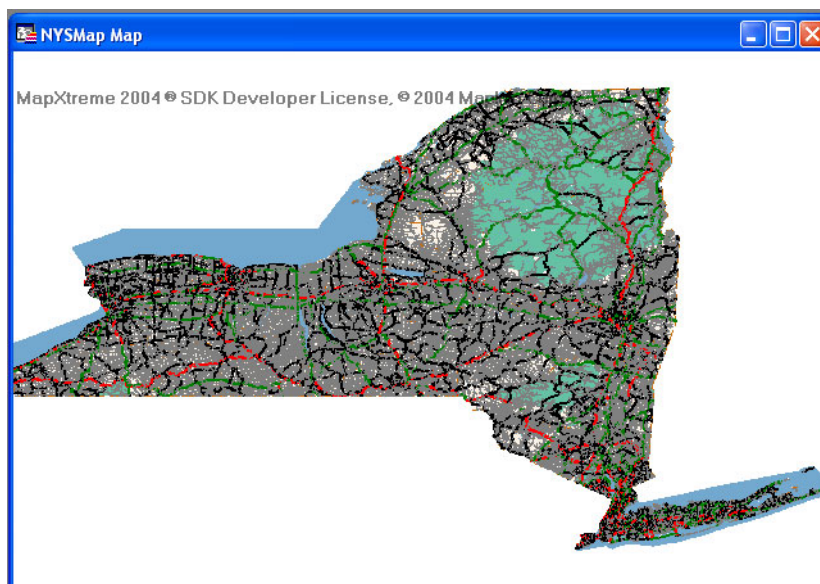
4. Click the **CLOSE** button to return to the **WMS SERVER INFORMATION** dialog box. Click the **GET DESCRIPTION** button to get a value for the **DESCRIPTION** field. Then click **OK**.

Note: If you would like to populate the **DESCRIPTION** field yourself, type in whatever you would like instead of clicking the **GET DESCRIPTION** button.

This returns you to the **WMS SERVERS LIST** dialog box. Select the WMS server you just created from the list of those available. Once you have done this, the list of layers you specified in the WMSConfig.xml file is displayed in the left panel of the dialog box.

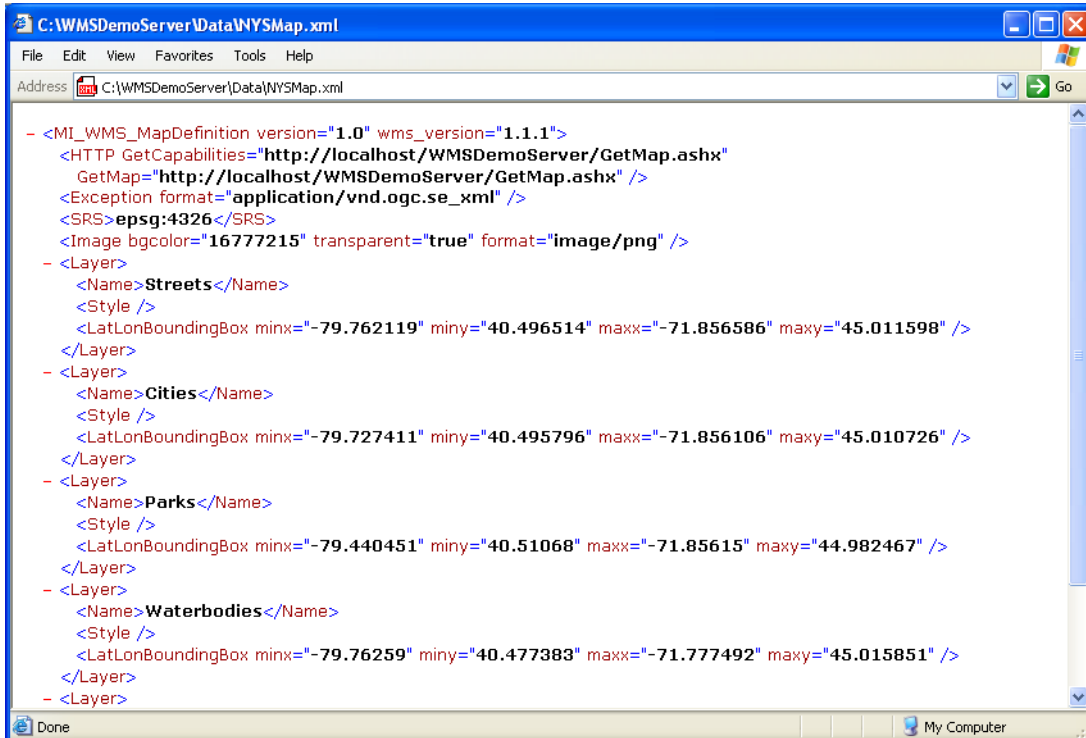


5. Highlight the layers you would like to include and click the **ADD** button. Note that the full Layer alias name appears under the lists as you select each item. Using descriptive names are easier to use than a file name or other abbreviation, such as QLD_OC. You have the option to display the layer(s) as gif, jpeg, png, or tiff images.
6. Finally, type in a name for an output TAB file or select one by browsing using the ellipsis button (...) then click the **OK** button. If all your settings are correct you should see a map window with the contents served from your new WMS server (e.g., not the actual TAB files).



MapInfo Professional now displays the combination of layers that you specified as a single image, which acts as a single layer, within the Map window. You can use the **OPEN WMS TABLE** dialog box multiple times to create a stack of layered images and create separate tables.

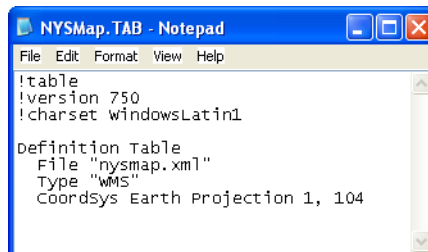
Every time you open a layer from the WMS Server, it is saved locally. The local version of the file includes both a TAB file and an XML file (shown below). These files contain everything needed to connect and display this WMS layer. You do not need to use the **OPEN WMS TABLE** dialog box again if the map has the same contents.



```

- <MI_WMS_MapDefinition version="1.0" wms_version="1.1.1">
  <HTTP_GetCapabilities="http://localhost/WMSDemoServer/GetMap.ashx"
  <GetMap="http://localhost/WMSDemoServer/GetMap.ashx" />
  <Exception format="application/vnd.ogc.se_xml" />
  <SRS>epsg:4326</SRS>
  <Image bgcolor="16777215" transparent="true" format="image/png" />
  - <Layer>
    <Name>Streets</Name>
    <Style />
    <LatLonBoundingBox minx="-79.762119" miny="40.496514" maxx="-71.856586" maxy="45.011598" />
  </Layer>
  - <Layer>
    <Name>Cities</Name>
    <Style />
    <LatLonBoundingBox minx="-79.727411" miny="40.495796" maxx="-71.856106" maxy="45.010726" />
  </Layer>
  - <Layer>
    <Name>Parks</Name>
    <Style />
    <LatLonBoundingBox minx="-79.440451" miny="40.510668" maxx="-71.85615" maxy="44.982467" />
  </Layer>
  - <Layer>
    <Name>Waterbodies</Name>
    <Style />
    <LatLonBoundingBox minx="-79.76259" miny="40.477388" maxx="-71.777492" maxy="45.015851" />
  </Layer>
  - <Layer>

```



```

NYMap.TAB - Notepad
File Edit Format View Help
!table
!version 750
!charset windowsLatin1

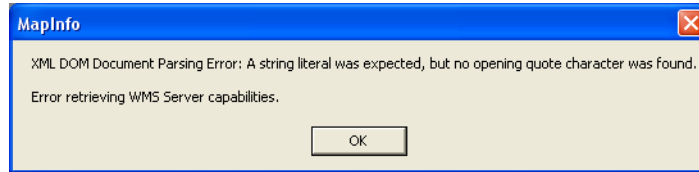
Definition Table
  File "nysmap.xml"
  Type "WMS"
  Coordsys Earth Projection 1, 104

```

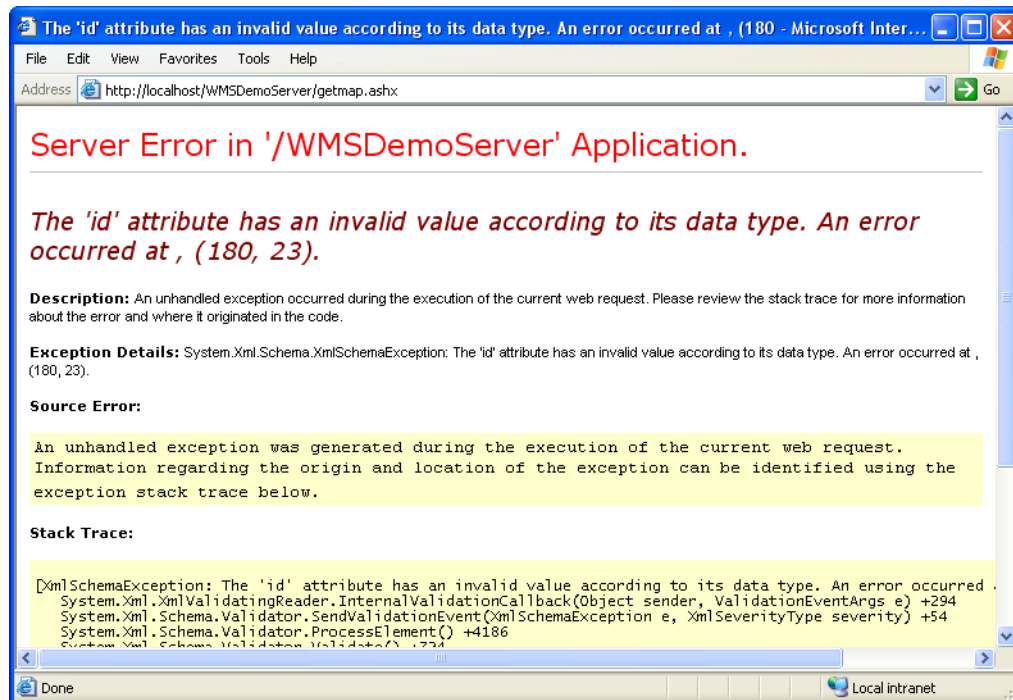
Troubleshooting your WMS Setup

Fixing Your Setup

As with all web service style applications, configuration files are critical. One small spelling mistake can cause the entire system to stop working. Below is shown a typical error message that may occur if you failed to connect to the WMS server you just created.



This, less than helpful, error message tells you there is an error on the WMS server but does not explain what is wrong or what to do next. To get more detailed information about the error, connect to the service from Internet Explorer directly. This method allows you to see more details about the error.



By connecting to the service using Internet Explorer directly, you can see that there is something wrong with the WMSConfig.xml file at line 180. Now you can navigate to that spot to begin troubleshooting.

Every time you adjust the web.config or the WMSConfig.xml files you need to make sure to close the files from your editor. If either of the files are in use by another application while the server is trying to access the file, you'll get an error.

Note: Due to a bug in Windows, you may need to kill the aspnet_wp.exe (or w3wp.exe for Windows 2003 Server) process each time you need to retry the web service.

The process to fix an error is as follows:

1. Edit configuration file (either web.config or wmsconfig.xml).
2. Close the files (and maybe restart the aspnet_wp.exe or w3wp.exe process).
3. Retry the WMS server using IE or your sample client.

Troubleshooting your Setup

There are a several places to verify that your configuration information is correct:

Web.config file

1. Is the value for ConfigFile correct?
The path to the wmsconfig.xml file needs to be completely accurate in the value attribute of the <appSettings>|<add> element in the Web.config for the server to work properly.
2. Are the Public Key Tokens correctly specified?
Make sure that the Public Key Token values (as can be discovered while using Assembly in **Step 1: Verify the Software Version**) are correctly specified in the type(Version) attribute of the <httpHandlers>|<add> element of the Web.config file.
3. Is the version number of the software correctly specified?
Make sure that the Version values discovered in **Step 1: Verify the Software Version** are correctly specified in the type(Version) attribute of the <httpHandlers>|<add> element of the Web.config file.

If any small detail is inaccurate, the system does not operate properly and throws an error. Once you iron out these details, the system works correctly every time.

wmsconfig.xml File

1. Is the value in the <mxp-wms:OnlineResource> element properly specified?
This value needs to specify the correct URL ending with `getmap.ashx`.
2. Are the data sources properly defined in the <mxp-wms:DataSourceDefinitionSet> elements?
Each data source needs to have its own element containing a valid path to each particular file. Make sure you do not duplicate id attribute values or <mxp:DataSourceName> element values in the file. Also each <mxp:FileName> element value must be a valid path pointing to a valid file.
3. Are the elements in the <WmsLayerList> properly specified?
 - a. Are the names and titles unique?
If either a name or title is duplicated the system throws an error.
 - b. Are the <mxp:FeatureLayer> attributes unique?
If any of these values are duplicated, the system throws an error.
 - c. Do the ref attributes in the <mxp:DataSourceRef> elements properly reference their <mxp:TABFileDataSourceDefinition> element counterparts?
Each value for this attribute must perfectly match the id attribute of the corresponding <mxp:TABFileDataSourceDefinition> element in order for the system to work properly.

IIS Setup

1. Is the directory to be served established as a Virtual Directory?
2. Is the IIS server running?
3. Did you kill the aspnet_wp.exe or w3wp.exe process?

It is not immediately apparent that the process has been killed as the process restarts almost instantly, however the memory usage is significantly less.

Accessing Network Drives

The ASP.NET login is a local machine account which does not have network file access privileges. As a result, the ASP.NET process that runs your WMS server must be configured to run as a user that has permissions to access the network folder. Also, the network folder must be fully specified as a Universal Naming Convention (UNC) instead of a mapped drive. For example, use \\server\data rather than a mapped drive such as "Z:\data".

There are a number of scenarios and options considered to configure the ASP.NET process user. The links below access knowledge documents that describe appropriate solutions.

Accessing Network Drives with IIS 5.0/5.1

Simple Password

Choose a network user to run the ASP.NET process and ensure that user has access to the network folder. Where possible limit the permissions of this user to prevent unexpected or malicious access to data.

As an example, assume that the username is "MYDOMAIN\WMSUser", with a password "WMSUserPassword".

1. Make a backup copy of the Web.config file.
2. Add the following line in the <system.web> section of the web.config file:

```
<identity password="WMSUserPassword" userName="MYDOMAIN\WMSUser" impersonate="true" />
```
3. Restart the WMS Server.

Because the password for the named user is stored without any form of encryption, this approach makes it possible for someone to access the web.config file and obtain this password. If you need a more secure approach, use the Encrypted Password section below.

Encrypted Password

Instructions for creating encrypted passwords can be found in the following MSDN articles:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mbnsdkv1/html/mbs_mbn_settheusernameandpasswordforaspnetimperson.asp.

and

<http://support.microsoft.com/default.aspx?scid=kb;en-us;329290>.

It may be necessary to use the hotfix described in the following article to enable the use of encrypted registry passwords:

<http://support.microsoft.com/default.aspx?scid=kb;EN-US;329250>.

Accessing Network Drives with IIS 6.0

The methods described above for IIS5.0/5.1 are also relevant to IIS 6.0. There may also be more convenient methods for the configuration of the ASP.NET process user identity and related security. For example, see the section "userName and password Equivalent Settings in IIS 6.0 Worker Process Isolation Mode" in the following article:

http://www.microsoft.com/resources/documentation/iis/6/all/proddocs/en-us/ca_asp_net.mspx.