MapGuide Open Source

Building and Configuring on Windows

MapGuide Open Source

March 2006

1 2 3 4 5 6 7 8 9 10

Contents

Building MapGuide on Windows

1

In this chapter

Introduction

This document is for developers who want to build MapGuide Open Source from the source files. It is primarily for those who want to contribute to the development of the MapGuide product itself.

If you want to install a pre-built version of MapGuide and use it for developing mapping applications, see *MapGuide Open Source: Installing and Configuring on Windows* instead.

System Requirements

NOTE The following instructions are for building using Microsoft Visual Studio 2005.

There are two components to MapGuide:

- MapGuide Server
- MapGuide Web Server Extensions

You can build the components separately. They do not have to run on the same computer.

NOTE You will need approximately 2.8 GB of disk space to build MapGuide Server and MapGuide Web Server Extensions.

Building MapGuide Server

The source code for MapGuide is available from *www.osgeo.org* as a gzipped tar file. The file contains source for both MapGuide Server and MapGuide Web Server Extensions. Download the distribution and unzip to a local directory that you will use for building (`InstallDir`).

This creates the following directories:

```
InstallDir
  \mapguide-1.0.0
    \BuildTools
    \Common
    \Oem
    \Server
    \UnitTest
    \Web
```

The rest of this document uses the following conventions for directory names:

- `InstallDir`: The root installation directory
- `WebInstallDir`: The Web installation directory

Make the appropriate substitutions for path names in the instructions that follow.

MapGuide Server requires Feature Data Objects (FDO), also available from *www.osgeo.org* as a gzipped tar file. Unzip the file to the same `InstallDir`. You may also want to download FDO documentation and data for unit tests.

Setting Environment Variables

Set the following system environment variables before building MapGuide Server or any components:

GDAL_DATA

`InstallDir\mapguide-1.0.0\Oem\gdal-1.3.0\data`

PROJ_LIB

`InstallDir\mapguide-1.0.0\Oem\proj-4.4.9\nad`

Building the FDO Components

To build the FDO components, follow the instructions in *InstallDir*\OpenSource_FDO\OpenSourceBuild__README.txt.

NOTE When performing the build, FDO must be installed to `InstallDir\mapguide-1.0.0\Oem`. Do this by using the -o option with build.bat:

```
build -o=InstallDir\mapguide-1.0.0\Oem
```

Building the OEM Components

To build the OEM components, open the oem.sln file located in `InstallDir\mapguide-1.0.0\Oem`. Build the solution. If you plan to create a debug version of the server, then you must build a debug version of the OEM components.

Building the Server

To build the server, open the Server.sln file located in `InstallDir\mapguide-1.0.0\Server`. Build the solution.

If you create a debug version of the server, you must create a debug version of the OEM components, and if you create a release version of the server, you must create a release version of the OEM components.

Building MapGuide Web Server Extensions

If you are installing MapGuide Web Server Extensions separately from MapGuide Server, download the source code from *www.osgeo.org*. Unzip to a directory that will be used for the installation (*`InstallDir`*).

MapGuide Web Server Extensions shares many components with MapGuide Server. If you have already set the proper environment variables and built the OEM components you do not need to repeat those steps.

Setting Environment Variables

Set the following system environment variables before building MapGuide Web Server Extensions or any components:

GDAL_DATA

`InstallDir\mapguide-1.0.0\Oem\gdal-1.3.0\data`

PROJ_LIB

`InstallDir\mapguide-1.0.0\Oem\proj-4.4.9\nad`

Building the OEM Components

To build the OEM components, open the oem.sln file located in `*InstallDir*\mapguide-1.0.0\Oem`. Build the solution. If you plan to create a debug version of the Web Server Extensions, then you must build a debug version of the OEM components.

Installing the Java JDK

MapGuide Web Server Extensions requires the Java JDK, version 1.3 or higher. If you do not have the Java JDK installed, download from *java.sun.com/j2se/1.5.0/download.jsp* and install it.

The JDK tools and include files must be accessible from the Visual Studio build environment. In the following instructions, *JDKDir* represents the directory where you installed the JDK.

In Visual Studio, choose Tools ➤ Options. In Projects and Solutions, VC++ Directories, in the field Show directories for:, select Executable files. Add

```
JDKDir\bin
```

to the list.

Select Include files in the Show directories for: field. Add

```
JDKDir\include
```

and

```
JDKDir\include\Win32
```

to the list.

Building the Web Server Extensions

In Visual Studio, open the `WebInstallDir\src\WebTier.sln` file. Build the solution.

Ensure that the version of the Web Server Extensions matches the version of the OEM components. If you build a debug version of the Web Server Extensions, you must build a debug version of the OEM components. If you build a release version of the Web Server Extensions, you must build a release version of the OEM components.

2 Configuring MapGuide Web Server Extensions on Windows

In this chapter

Possible Configuration Scenarios

To run MapGuide Web Server Extensions, you must install and configure a web server (Apache HTTP Server or Internet Information Services). You must also install and configure PHP if you intend to run the Site Administrator or use PHP as an application development environment.

Download PHP version 5.0.5 from *www.php.net*. Follow the standard installation instructions. This document assumes that you install PHP to the standard directory C:\PHP. If you install to a different location, make the appropriate substitutions in the instructions.

Directory Structure

If you build MapGuide from source, *`InstallDir`* is the top directory of your source tree, containing the `mapguide-1.0.0` subdirectory. The Web Server Extensions directory (*`WebDir`*) is

`InstallDir``\mapguide-1.0.0\Web\src`

Make the appropriate substitutions for path names in the instructions that follow.

Preliminary Settings

MapGuide includes some test forms that you can use for testing API calls via a web interface.

To install the test forms, copy the content of the folder

`InstallDir``\mapguide-1.0.0\UnitTest\WebTier\MapAgent\MapAgentForms`

into the folder

`WebDir``\mapagent`

You must also edit the Web Server Extensions configuration file to set a resources folder and a folder for temporary files. Edit the file

`WebDir``\webconfig.ini`

Change the value of `ResourcesPath` to

`WebDir``\mapagent\Resources`

Change the value of `TempPath` to a folder used for temporary files. You can use an existing folder such as `C:\Windows\Temp`, or you can create a folder in the MapGuide installation, such as *`WebDir`*`\temp`.

Adding the Web Server Extensions Files to the PATH

Add the location of the Web Server Extensions binary files to the PATH system environment variable.

If you built the debug version of the Web Server Extensions, add the following to the PATH variable:

`InstallDir``\mapguide-1.0.0\Web\bin\debug`

If you built the release version of the Web Server Extensions, add the following to the PATH variable:

`InstallDir``\mapguide-1.0.0\Web\bin\release`

NOTE You must reboot the machine to ensure the updated PATH is picked up by the web server service.

Configuration with IIS

If you use IIS as the web server for MapGuide Web Server Extensions, you have the choice of using either PHP or ASP.NET to develop applications.

In either case, if you want to run the Site Administrator from the machine, you must install PHP.

If you build MapGuide from source, you must perform the following steps:

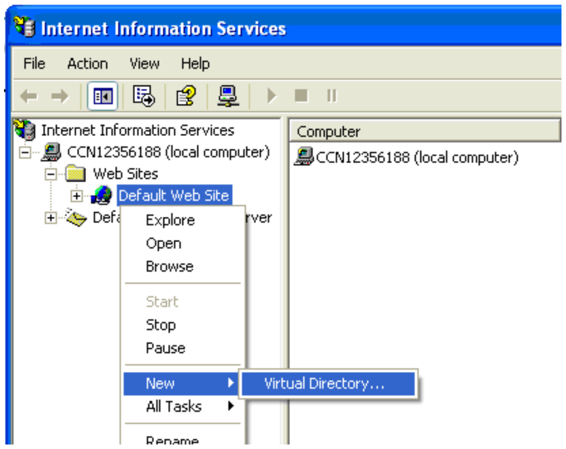
- Set up virtual directories for the Web Server Extensions and the AJAX and DWF Viewers.
- Configure PHP or ASP.NET.

Many of the configuration steps for PHP and ASP.NET are the same.

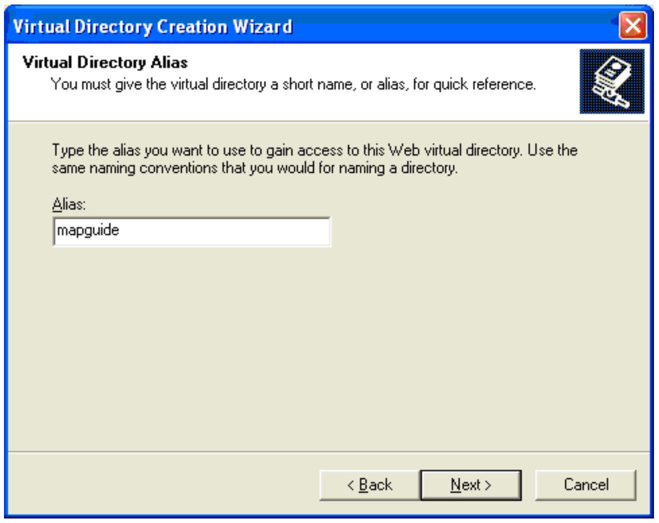
Setting Up Virtual Directories for IIS

NOTE This document contains screenshots showing the IIS Administration Console version 5.1. Dialogs and menus may differ slightly in other versions.

Start the IIS administration console and create a new virtual directory.



Name the virtual directory `mapguide`.



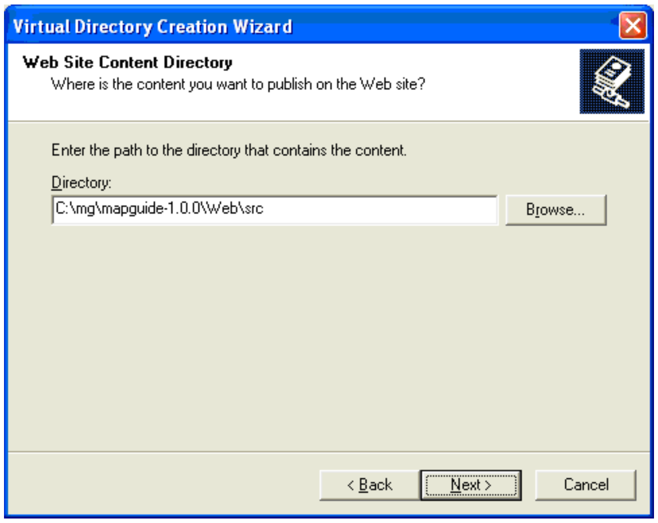
Virtual Directory Alias

You must give the virtual directory a short name, or alias, for quick reference.

Type the alias you want to use to gain access to this Web virtual directory. Use the same naming conventions that you would for naming a directory.

Alias: mapguide

Set the physical path to *`WebDir`*. In this example *`InstallDir`* is `C:\mg` and *`WebDir`* is `C:\mg\mapguide-1.0.0\Web\src`.


Virtual Directory Creation Wizard

Web Site Content Directory

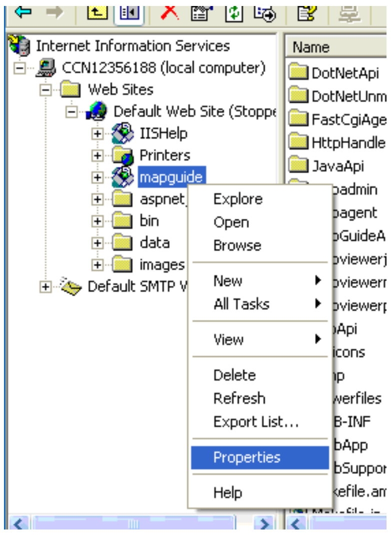
Where is the content you want to publish on the Web site?

Enter the path to the directory that contains the content.

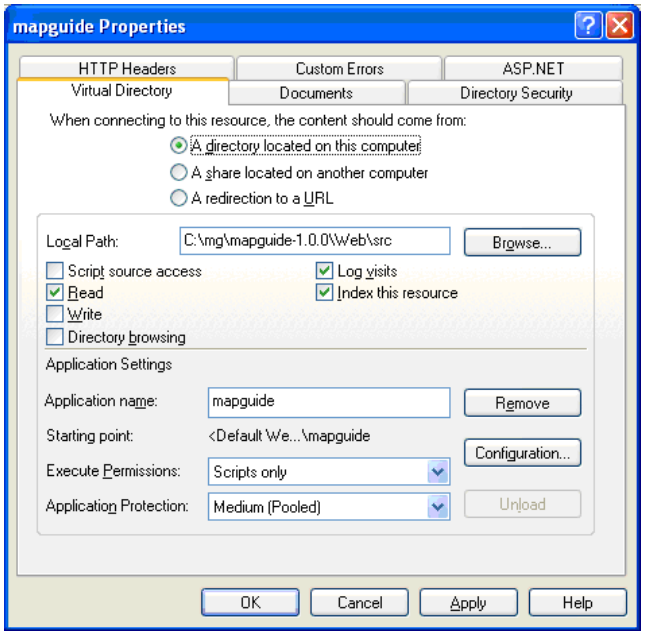
Directory:

C:\mg\mapguide-1.0.0\Web\src

Open the properties dialog for the virtual directory:



Set the properties for the virtual directory, as shown below:



When connecting to this resource, the content should come from:

- [✓ A directory located on this computer]
- A share located on another computer
- A redirection to a URL

Local Path: C:\mg\mapguide-1.0.0\Web\src

- [ ] Script source access
- [✓] Read
- [ ] Write
- [ ] Directory browsing
- [✓] Log visits
- [✓] Index this resource

Application Settings

Application name:	mapguide
Starting point:	<Default We...\mapguide
Execute Permissions:	Scripts only
Application Protection:	Medium (Pooled)

Setting Up the PHP Association

This is required if you will be using PHP as a development environment or if you will be running the Site Administrator on this machine.

You must set up an application mapping so IIS handles PHP files properly. You can choose to have PHP run as a CGI process or as an ISAPI extension.

Setting up the application mapping is the same in both cases, except that the executable path is different.

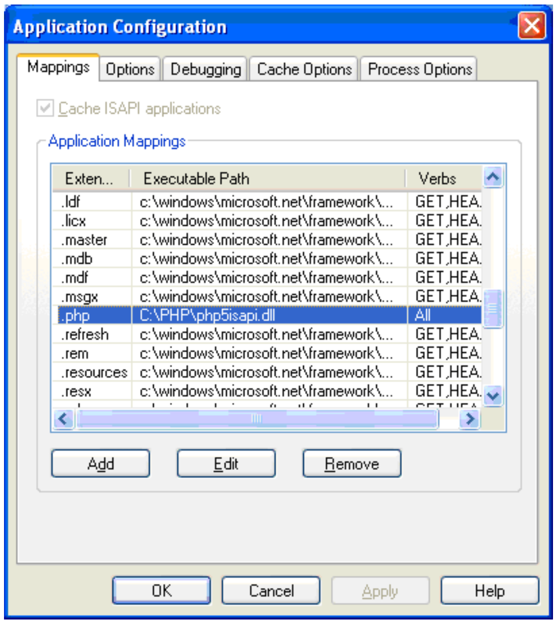
For CGI, the executable is

```
C:\PHP\php-cgi.exe
```

For ISAPI, the executable is

```
C:\PHP\php5isapi.dll
```

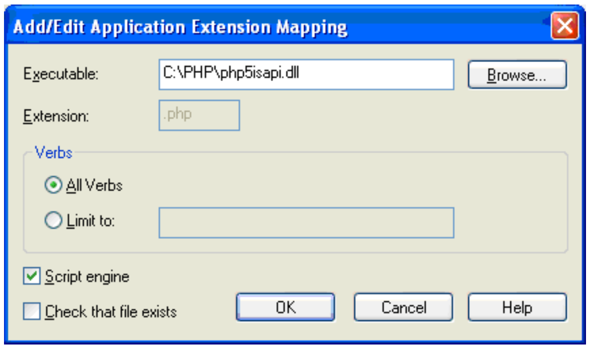
From the Properties dialog, click Configuration.



[✓] Cache ISAPI applications

Exten...	Executable Path	Verbs
.ldf	c:\windows\microsoft.net\framework\...	GET,HEA.
.licx	c:\windows\microsoft.net\framework\...	GET,HEA.
.master	c:\windows\microsoft.net\framework\...	GET,HEA.
.mdb	c:\windows\microsoft.net\framework\...	GET,HEA.
.mdf	c:\windows\microsoft.net\framework\...	GET,HEA.
.msgx	c:\windows\microsoft.net\framework\...	GET,HEA.
.php	C:\PHP\php5isapi.dll	All
.refresh	c:\windows\microsoft.net\framework\...	GET,HEA.
.rem	c:\windows\microsoft.net\framework\...	GET,HEA.
.resources	c:\windows\microsoft.net\framework\...	GET,HEA.
.resx	c:\windows\microsoft.net\framework\...	GET,HEA.

If a mapping for PHP has not been set up or is not set up properly, click Add or Edit to set up the mapping.



Executable: C:\PHP\php5isapi.dll

Extension: .php

[✓ All Verbs] Limit to:

[✓] Script engine

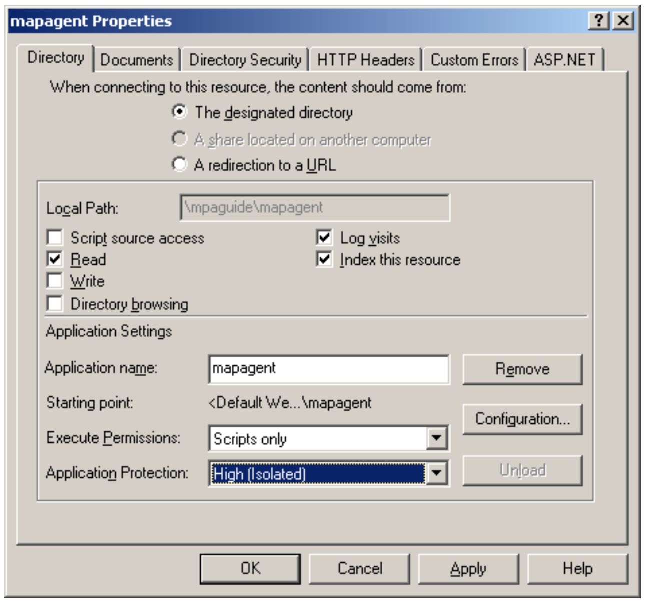
[ ] Check that file exists

Setting Up the FastCGI Association

MapGuide uses a FastCGI agent to improve performance. To configure FastCGI, perform the following steps.

Copy isapi_fcgi.dll from `*InstallDir*\mapguide-1.0.0\Oem\fcgi\Release` to `*WebDir*\mapagent`.

In the IIS Administration Console, set the properties of the mapguide\mapagent folder. Create an application named `mapagent`. Set the application protection to High (Isolated).



When connecting to this resource, the content should come from:

[✓ The designated directory]
A share located on another computer
A redirection to a URL

Local Path: \mpaguide\mapagent

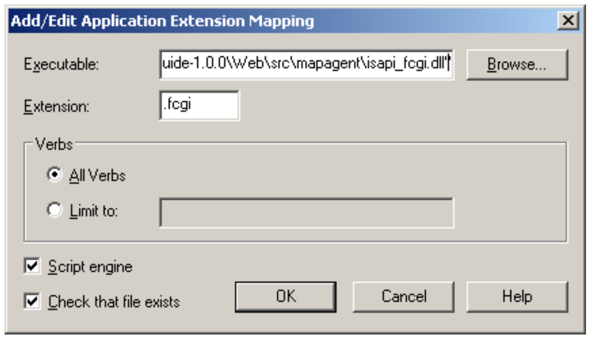
- [ ] Script source access
- [✓] Read
- [ ] Write
- [ ] Directory browsing
- [✓] Log visits
- [✓] Index this resource

Application Settings

Application name: mapagent
Starting point: <Default We...\mapagent
Execute Permissions: Scripts only
Application Protection: High (Isolated)

Click Configuration and set up an application extension mapping to isapi_fcgi.dll for the .fcgi extension.

NOTE You cannot have spaces in the path to isapi_fcgi.dll. For example, you may need to replace `Program Files` with `Progra~1` in the path.



Executable: uide-1.0.0\Web\src\mapagent\isapi_fcgi.dll"
Extension: .fcgi

Verbs
[✓ All Verbs]
Limit to:

- [✓] Script engine
- [✓] Check that file exists

Add the following to the Windows Registry. Change the path to mapagent.exe if necessary.

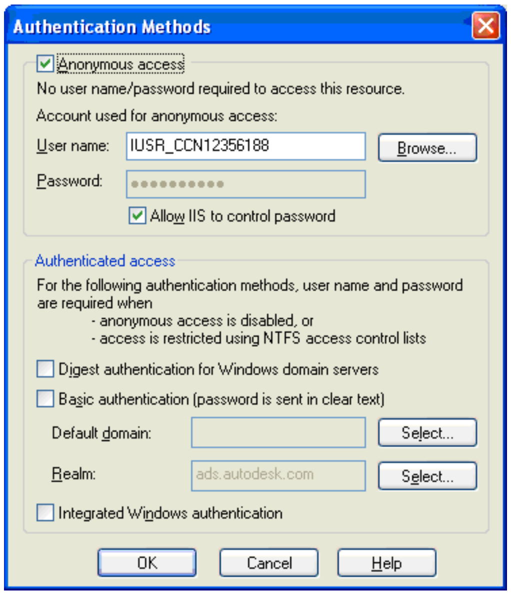
```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\FASTCGI]
"MaxPostData"=dword:00000000
"CustomVars"=dword:00000000
"ThreadPoolSize"=dword:0000000a
"BypassAuth"=dword:00000001
"Impersonate"=dword:00000000
"StartServers"=dword:00000001
"IncrementServers"=dword:00000001
"MaxServers"=dword:00000004
"Timeout"=dword:0000003c

[HKEY_LOCAL_MACHINE\SOFTWARE\FASTCGI\.fcgi]
"BindPath"="app-mapagent"
"AppPath"="C:\\mg\\mapguide-1.0.0\\Web\\src\\mapagent\\mapa
gent.exe"
```

You should increase the number of ephemeral ports available for the FastCGI agent. Otherwise, there is a chance that the pool of available ports will be used up before enough ports have been freed. Refer to the Microsoft support note at *http://support.microsoft.com/default.aspx?scid=kb;en-us;319502* for details.

Another alternative is to disable the FastCGI agent and use a regular CGI agent instead. To do this, set the application mapping in IIS to `mapagent.exe` instead of `isapi_fcgi.dll`.

Setting Up the Security Properties

To set the security properties, go to the Directory Security tab and click Edit. Set the options as shown in the dialog below. Note that the account information is already set for you by IIS and you don't need to change it.


Authentication Methods

[✓] Anonymous access

No user name/password required to access this resource.

Account used for anonymous access:

User name: IUSR_CCN12356188

Password: ••••••••••

[✓] Allow IIS to control password

Authenticated access

For the following authentication methods, user name and password are required when
- anonymous access is disabled, or
- access is restricted using NTFS access control lists

[ ] Digest authentication for Windows domain servers

[ ] Basic authentication (password is sent in clear text)

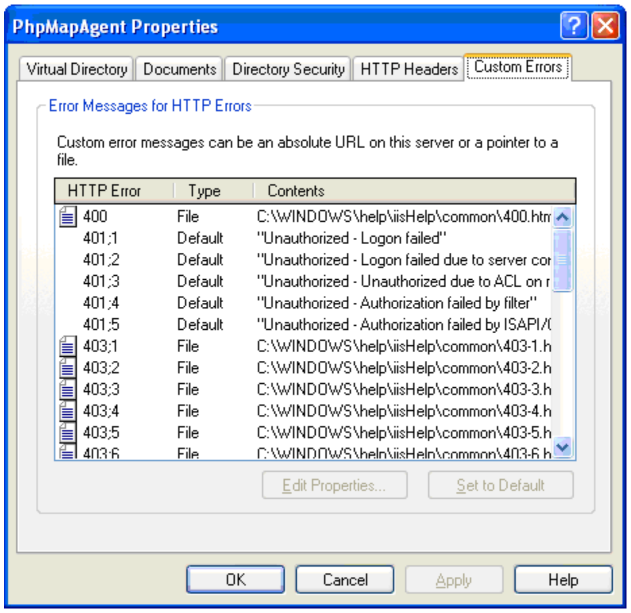
Default domain:

Realm: ads.autodesk.com

[ ] Integrated Windows authentication

Setting Up the Error Pages

On the Custom Errors tab, select all the 401 errors, and click Set to Default. This ensures that MapGuide processes authorization requests properly.



Custom error messages can be an absolute URL on this server or a pointer to a file.

HTTP Error	Type	Contents
400	File	C:\WINDOWS\help\iisHelp\common\400.htm
401;1	Default	"Unauthorized - Logon failed"
401;2	Default	"Unauthorized - Logon failed due to server cor
401;3	Default	"Unauthorized - Unauthorized due to ACL on r
401;4	Default	"Unauthorized - Authorization failed by filter"
401;5	Default	"Unauthorized - Authorization failed by ISAPI/(
403;1	File	C:\WINDOWS\help\iisHelp\common\403-1.h
403;2	File	C:\WINDOWS\help\iisHelp\common\403-2.h
403;3	File	C:\WINDOWS\help\iisHelp\common\403-3.h
403;4	File	C:\WINDOWS\help\iisHelp\common\403-4.h
403;5	File	C:\WINDOWS\help\iisHelp\common\403-5.h
403;6	File	C:\WINDOWS\help\iisHelp\common\403-6.h

Setting Up Virtual Directories for the Viewers

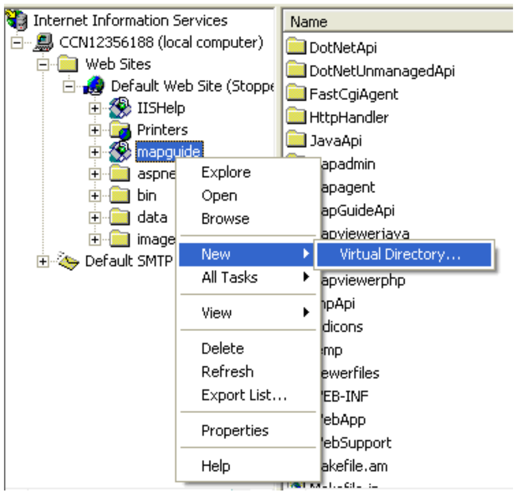
The AJAX Viewer and the DWF Viewer share the same physical path, but you must create separate virtual directories for each. For PHP, the physical path is

WebDir\mapviewerphp

and for ASP.NET, the physical path is

WebDir\mapviewernet

From the IIS administration console, create a new virtual directory for the AJAX viewer.

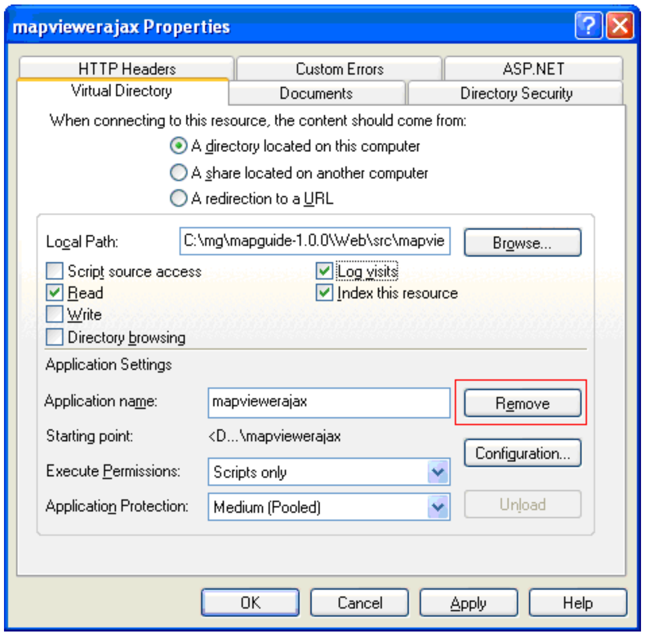


Set the alias name to `mapviewerajax`.

If you use PHP as your development language, set the physical path to *`WebDir`*`\mapviewerphp`. If you use ASP.NET as your development language, set the physical path to *`WebDir`*`\mapviewernet`.

Finally, set up the default document for the new virtual directory. Open the Documents tab of the Properties dialog, and add the appropriate document to the default list:

- For PHP development, add ajaxviewer.php to the default document list.
- For ASP.NET development, add ajaxviewer.aspx to the default document list.



When connecting to this resource, the content should come from:

[✓ A directory located on this computer]
A share located on another computer
A redirection to a URL

Local Path: C:\mg\mapguide-1.0.0\Web\src\mapvie

Script source access [ ]
Read [✓]
Write [ ]
Directory browsing [ ]
Log visits [✓]
Index this resource [✓]

Application Settings

Application name: mapviewerajax
Starting point: <D...\mapviewerajax
Execute Permissions: Scripts only
Application Protection: Medium (Pooled)

Repeat the steps above to create a virtual directory for the DWF viewer. Set the name to `mapviewerdwf` instead of `mapviewerajax`. The physical path is the same for both directories. The default documents are as follows:

- For PHP development, add dwfviewer.php to the default document list.
- For ASP.NET development, add dwfviewer.aspx to the default document list.

Configuring PHP

You must configure PHP if you plan to use it as a development language for MapGuide applications or if you plan to use the Site Administrator. To configure PHP, edit the PHP configuration file, php.ini.

The php.ini file may be in different places depending how PHP was installed on your system. It is often in `C:\Windows` or in the PHP installation folder.

Identifying the Correct php.ini File

PHP checks for php.ini in many different places. You must ensure that the MapGuide version is the one PHP uses. Read the PHP documentation for a complete explanation.

The recommended method is to create a system environment variable named `PHPRC`, and set it to the PHP directory containing php.ini.

With the default PHP installation, this is

```
C:\Php
```

NOTE There may be a Windows registry entry that determines which php.ini will be processed. Check for the key `HKEY_LOCAL_MACHINE\SOFTWARE\PHP\IniFilePath`. Delete it if it exists.

Editing php.ini

If you run PHP as a CGI process, you must make the following change to php.ini. Change

```
; cgi.force_redirect = 1
```

to

```
cgi.force_redirect = 0
```

Make the following additional changes to php.ini:

- Update the extension directory.
  Locate the `extension_dir` entry and set it to `*InstallDir*\mapguide-1.0.0\Web\bin\*config*`, where *`config`* is the configuration you built, either `debug` or `release`.
- Add the MapGuide extensions.
  Locate the list of existing extension keys and add the following extensions. If you built the debug configuration add:

  ```
  extension=php_MapGuideApid.dll
  ```

  ```
  extension=php_mbstring.dll
  ```

  If you built the release configuration add:

  ```
  extension=php_MapGuideApi.dll
  ```

  ```
  extension=php_mbstring.dll
  ```

Copy the file php_mbstring.dll from the PHP ext directory to the `extension_dir` location.

- Increase the resource limits.
  The default PHP installation sets low resource limits for PHP scripts, which limits the amount of information that MapGuide can work with. To facilitate the use of larger data sets, adjust the following resource limits:

  Original values in php.ini:

  ```
  output_buffering = Off
  max_execution_time = 30
  memory_limit = 8M
  post_max_size = 8M
  upload_max_filesize = 2M
  ```

  Recommended values:

  ```
  output_buffering = 4096
  max_execution_time = 60
  memory_limit = 64M
  post_max_size = 64M
  upload_max_filesize = 64M
  ```

Adding the PHP Directory to the PATH

If you run PHP as an ISAPI extension, you must add the PHP directory to the system PATH environment variable. This ensures that the PHP DLLs can be loaded properly.

Add

```
C:\Php
```

to the system PATH environment variable.

Configuring ASP.NET

Verify that you are using the correct version of ASP.NET. MapGuide requires version 2.0.50727 or higher. Older versions will not work. To verify the version, open the IIS administration console. Navigate to the mapguide site and open the site properties. On the ASP.NET tab, check the version number. Upgrade if necessary.

Next, copy the MapGuide server managed and unmanaged API DLLs from the installation folder to the web folder.

- Create a folder named `bin` in the folder *`WebDir`*`\mapviewernet`.
- If you built a debug version of Web Server Extensions, copy the following files to the bin folder:

  `InstallDir``\mapguide-1.0.0\Web\bin\debug\MapGuideDotNetApid.dll`

  `InstallDir``\mapguide-1.0.0\Web\bin\debug\MapGuideUnmanagedApid.dll`
- If you built a release version of Web Server Extensions, copy the following files to the bin folder:

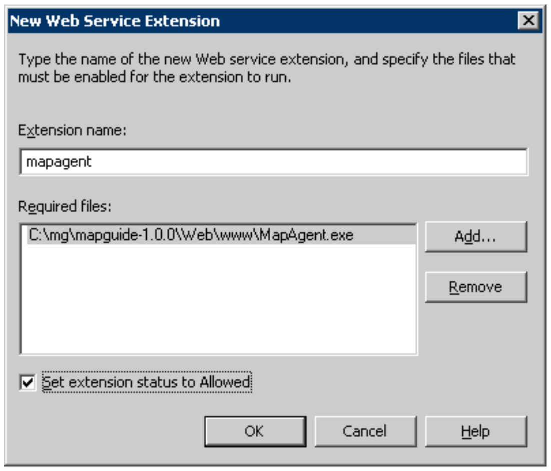
  `InstallDir``\mapguide-1.0.0\Web\bin\release\MapGuideDotNetApi.dll`

  `InstallDir``\mapguide-1.0.0\Web\bin\release\MapGuideUnmanagedApi.dll`

Configuring Web Service Extensions for IIS 6

IIS 6, running on Windows Server 2003, requires some additional configuration to enable PHP and the Map Agent to run.

From the IIS Manager, select the Web Service Extensions folder. Click Add a new Web Service Extension. Enter `mapagent` as the extension name, and *`WebDir`*`\mapagent\isapi_fcgi.dll` as the required file. Set the extension status to Allowed.


New Web Service Extension

Type the name of the new Web service extension, and specify the files that must be enabled for the extension to run.

Extension name: mapagent

Required files: C:\mg\mapguide-1.0.0\Web\www\MapAgent.exe

☑ Set extension status to Allowed

Repeat for the PHP CGI module. Enter `php` as the extension name, and `C:\PHP\php-cgi.exe` as the required file.

NOTE If you use the regular CGI agent instead of the FastCGI agent, replace `isapi_fcgi.dll` with `mapagent.exe` in the instructions above.

Configuration with Apache

If you use Apache as the web server for MapGuide Web Server Extensions, you have the choice of using either PHP or JSP to develop applications.

In either case, if you want to run the Site Administrator from the machine, you must install PHP.

Most of the configuration is done by editing httpd.conf, in the conf directory of the Apache installation.

NOTE Some directives have been split over multiple lines in this document, but they must be written on one line in the configuration file.

Common Configuration Steps

There are some steps you must perform whether you are using PHP or JSP as a development environment.

- If you are using the Site Administrator or plan to use PHP as a development environment, configure Apache to process files with the .php extension. Edit httpd.conf. Look for the configuration of the PHP engine. Ensure it is set as follows, or add the following to the file as necessary.
  ```
  ScriptAlias /php/ "C:/PHP/"
  Action application/x-httpd-php "/php/php-cgi.exe"
  AddType application/x-httpd-php .php
  ```
- In the Dynamic Shared Object Support section, ensure that the rewrite module is enabled. If there is a comment character at the beginning of the line, remove it so it looks like this:
  ```
  LoadModule rewrite_module modules/mod_rewrite.so
  ```

Configuring Apache to Use PHP

To use PHP as the development environment with the Apache HTTP Server, add the following declarations at the end of httpd.conf.

Replace *<WebDir>* with your actual path wherever specified.

```
ScriptAlias /mapguide/mapagent/mapagent.fcgi
    "<WebDir>/mapagent/MapAgent.exe"
AliasMatch ^/mapguide/mapviewerajax/([^\?])(.*)$
    "<WebDir>/mapviewerphp/$1$2"
Alias /mapguide/mapviewerajax
    "<WebDir>/mapviewerphp/ajaxviewer.php"
AliasMatch ^/mapguide/mapviewerdwf/([^\?])(.*)$
    "<WebDir>/mapviewerphp/$1$2"
Alias /mapguide/mapviewerdwf "<WebDir>/mapviewerphp/dwfviewer.php"

Alias /mapguide "<WebDir>/"
<Directory "<WebDir>/">
  AllowOverride All
  Options All
  Order allow,deny
  Allow from all
  AddHandler cgi-script .exe
  RewriteEngine on
  RewriteRule .* - [E=REMOTE_USER:%{HTTP:Authorization},L]
</Directory>
```

Save the changes and restart the Apache server.

Configuring Apache to Use JSP

To use JSP as the development environment with the Apache HTTP Server, you must install the following components:

- Java JDK 1.5
- Tomcat
- The Apache/Tomcat connector.

Download the Java JDK from *java.sun.com*. Install using the default installation instructions. Set the environment variable `JAVA_HOME` to the root directory of your Java installation.

Download Tomcat and the Tomcat-Apache connector from *tomcat.apache.org*. Install using the default installation instructions.

You must also perform the following steps:

- Edit the Apache configuration file, httpd.conf.
- Add a workers.properties configuration file for the Tomcat connector.
- Configure and start Tomcat.

Apache Configuration

Add the following declarations at the end of httpd.conf.

Replace *<WebDir>* with your actual path.

```
ScriptAlias /mapguide/mapagent/mapagent.fcgi
    "<WebDir>/mapagent/MapAgent.exe"
AliasMatch ^/mapguide/mapviewerajax/([^\?])(.*)$
    "<WebDir>/mapviewerjava/$1$2"
Alias /mapguide/mapviewerajax
    "<WebDir>/mapviewerjava/ajaxviewer.jsp"
AliasMatch ^/mapguide/mapviewerdwf/([^\?])(.*)$
    "<WebDir>/mapviewerjava/$1$2"
Alias /mapguide/mapviewerdwf
    "<WebDir>/mapviewerjava/dwfviewer.jsp"

Alias /mapguide "<WebDir>/"
<Directory "<WebDir>/">
  AllowOverride All
  Options All
  Order allow,deny
  Allow from all
  AddHandler cgi-script .exe
  RewriteEngine on
  RewriteRule .* - [E=REMOTE_USER:%{HTTP:Authorization},L]
</Directory>
```

Add the following to the configuration file. These declarations are necessary for the Tomcat connector. Replace *<HostName>:<Port>* with the correct name and port. `localhost` will not work for the redirection.

Replace *<ApacheHome>* with the root folder for Apache.

```
LoadModule    jk_module  modules/mod_jk.so

JkWorkersFile "<ApacheHome>/conf/workers.properties"

# Where to put jk logs
JkLogFile     "<ApacheHome>/logs/mod_jk.log"
# Set the jk log level [debug/error/info]
JkLogLevel    info
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions    +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat     "%w %V %T"
# Send everything for context /examples to worker
# named worker1 (ajp13)
JkMount  /mapguide/mapviewerjava/* worker1

Redirect /mapguide/mapviewerajax/
   http://<HostName>:<Port>/mapguide/mapviewerjava/ajaxviewer.jsp
Redirect /mapguide/mapviewerdwf/
    http://<HostName>:<Port>/mapguide/mapviewerjava/dwfviewer.jsp
```

Connector Configuration

Rename the pre-built Apache/Tomcat connector binary to mod_jk.so and place it in the Apache modules directory. The original file includes the version number and should be mod_jk-1.2.14-apache-2.0.54.so.

Add a workers.properties file in the Apache conf folder, containing the following:

```
# Define 1 real worker using ajp13
worker.list=worker1
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=50
worker.worker1.cachesize=10
worker.worker1.cache_timeout=600
worker.worker1.socket_keepalive=1
worker.worker1.recycle_timeout=300
```

Tomcat Configuration

In the Tomcat conf\Catalina\localhost directory, add a context file named mapguide.xml with the following content:

Substitute *<WebDir>* with your actual path.

```
<!--

    Context configuration file for the Tomcat Manager Web App

-->

<Context docBase="<WebDir>" privileged="true"   antiResourceLock
ing="false" antiJARLocking="false">

    <!-- Link to the user database we will get roles from -->
    <ResourceLink name="users" global="UserDatabase"
                  type="org.apache.catalina.UserDatabase"/>

</Context>
```

To start Tomcat, execute

```
startup.bat
```

in the Tomcat bin directory.

Verifying the Configuration

You can run some simple tests to ensure that the Map Agent and Web Server Extensions are configured properly.

Testing the Map Agent

Ensure that the MapGuide Server is running. See Debugging the Server (page 33) for details about starting the server in debug mode.

Open a web browser and go to the following URL. If the Web Server Extensions are not running on your local machine, replace `localhost` with the name or IP address of your server.

*http://localhost/mapguide/mapagent/mapagent.fcgi?
OPERATION=ENUMERATERESOURCES&VERSION=1.0.0&
LOCALE=en&RESOURCEID=Library://&TYPE=&DEPTH=-1*

Note that if your web server is listening on a different port you must include the port number in the HTTP request. For example, if Apache is listening on port 8008, you must go to the following URL:

*http://localhost:8008/mapguide/mapagent/mapagent.fcgi?
OPERATION=ENUMERATERESOURCES&VERSION=1.0.0&
LOCALE=en&RESOURCEID=Library://&TYPE=&DEPTH=-1*

Enter *Administrator* for the user id and *admin* for the password. Both are case-sensitive.

If the Map Agent is running properly, you will get an XML document describing the resources in the repository.

Testing the Site Administrator

Ensure that the MapGuide Server is running. Open a web browser and go to the following URL.

http://localhost/mapguide/mapadmin/login.php

Note that if your web server is listening on a different port you must include the port number in the HTTP request. For example, if Apache is listening on port 8008, you must go to the following URL:

http://localhost:8008/mapguide/mapadmin/login.php

Enter `Administrator` for the user id and `admin` for the password. Both are case-sensitive.

If the Site Administrator is configured properly, you will get a Manage Servers page which lists all the servers configured in the MapGuide Site.

Testing the Web Server Extensions

You can check that the Web Extensions are correctly set up. If your application development language is PHP, go to this URL in a web browser:

*http://localhost/mapguide/mapviewerajax/ajaxviewer.php?
WEBLAYOUT=Library://ATest.WebLayout*

If your application development language is ASP.NET, go to this URL in a web browser:

http://localhost/mapguide/mapviewerajax/ajaxviewer.aspx?WEBLAYOUT=Library://ATest.WebLayout

If your application development language is JSP, go to this URL in a web browser:

http://localhost/mapguide/mapviewerajax/ajaxviewer.jsp?WEBLAYOUT=Library://ATest.WebLayout

Because the web layout ATest does not exist in the MapGuide repository, you should see the following error:

```
Resource was not found: Library://ATest.WebLayout
```

This error is returned by the AJAX Viewer script. This shows that the Web Server Extensions are correctly enabled to receive and process requests.

3 Testing and Debugging

In this chapter

Running the Server Unit Tests

The Server unit tests exercise a large set of functionality, which spans all the services offered by the Server. The tests can be run without having built and configured the Web Server Extensions.

Running All Test Suites

Before running unit tests, ensure that there are no instances of MapGuide Server running. From the Windows Control Panel, select Services. Stop and disable the MapGuide Server service.

To run the tests you must use the `/testmode` command line option. If you are running the server in Visual Studio, select the `ServerCore` project. Choose Project ➤ Properties. Under Configuration Properties ➤ Debugging, in the Command Arguments field, enter

```
/testmode
```

You can also run the tests from a command prompt, using the command

```
mgserver /testmode
```

If you run the tests from a command prompt the test summary will remain visible once the tests complete, but if you run them from Visual Studio the console window automatically closes once the tests complete.

Once the tests complete a summary is displayed. If all tests pass you will get output similar to this:

```
  OK (378 tests)
```

If one or more tests fail, the output will look like this:

```
  !!!FAILURES!!!
  Test Results:
  Run:  378   Failures: 2   Errors: 0
```

followed by any exception messages related to the failures.

Running Individual Test Suites

Running all the test suites can take some time. You also have the option of running just a single test suite. You do this by including the

`/t TestSuiteName`

option after the

`/testmode`

option. The following test suites are currently available

- TestCoordinateSystem
- TestFeatureService
- TestGeometry
- TestLogManager
- TestMappingService
- TestRenderingService
- TestResourceService
- TestTileService
- TestServerAdminService
- TestServerManager
- TestServiceManager
- TestSiteService

For example, to run the geometry test suite, use the following command:

```
mgserver /testmode /t TestGeometry
```

Debugging the Server

Before debugging the server, ensure that there are no instances of MapGuide Server running. From the Windows Control Panel, select Services. Stop and disable the MapGuide Server service.

By default the development server configuration file (serverconfig.ini) is set up to use the directories of a server relative to where you built it. The following directories are created when you build a debug version of the server:

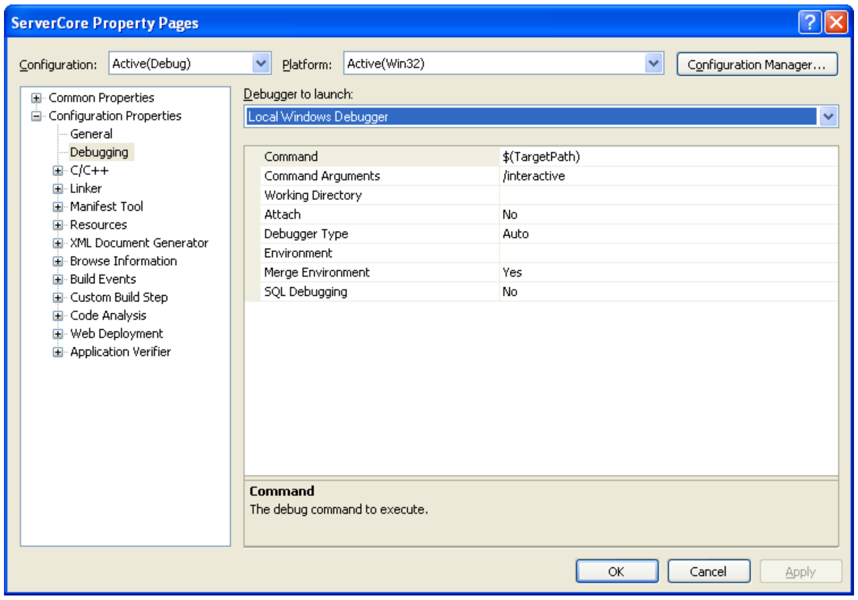
- *InstallDir*\mapguide-1.0.0\Server\bin\debug\FDO
- *InstallDir*\mapguide-1.0.0\Server\bin\debug\Logs

- `InstallDir\mapguide-1.0.0\Server\bin\debug\Packages`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\Repositories`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\Resources`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\Schema`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\Temp`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\wms`
- `InstallDir\mapguide-1.0.0\Server\bin\debug\wfs`

In MS Visual Studio, open the `Server` solution. Ensure that `ServerCore` is selected as the startup project.

Set command line arguments to allow the server to run interactively. This is necessary for debugging. Choose Project ➤ Properties. Under Configuration Properties ➤ Debugging, in the Command Arguments field, enter

`/interactive`.



Command	$(TargetPath)
Command Arguments	/interactive
Working Directory	
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No

Command
The debug command to execute.

Place breakpoints and start the server. This opens a server command window.

To shut down the server, press CTRL+C inside the server command window.

Running Web Server Extensions Unit Tests

If you build MapGuide from source, the source tarball includes unit tests. These are not included if you use the installer program.

The unit tests for the Web Server Extensions exercise both the Web Server Extensions code and the server code. The tests run on the PHP platform so make sure PHP is correctly installed and configured.

For the rest of the document we assume that `UnitTestDir` is equivalent to `InstallDir\mapguide-1.0.0\UnitTest`.

Setup for Unit Tests

Edit the PHP configuration file (php.ini) and add the following extensions. If you built the debug configuration add:

```
extension=php_curl.dll
extension=php_SQLitePHPApid.dll
```

If you built the release configuration add:

```
extension=php_curl.dll
extension=php_SQLitePHPApi.dll
```

The php_curl.dll extension is part of the PHP distribution and can be found in the PHP ext folder . The php_SQLitePHPApi.dll extension is built with the OEM components and can be found in `InstallDir\Oem\SQLite\bin\win32`. Copy both DLLs to your PHP extension_dir location.

Running Unit Tests From a Web Browser

1 Create a virtual directory named UnitTest under the root folder Default Web Sites, pointing to `UnitTestDir\WebTier\Php`.

2 In a web browser, enter the request

   `http://localhost/UnitTest/index.html`

3 Select the test you want to run.

4 Select the test mode:

   Validate - compares results to already existing results in a SQL database. The results can be stored either as a BLOB in the database or in a file.

Dump - writes the result of each operation into a file.

5 The test data is located in `UnitTestDir\TestData\TestSuiteName`. For example, the Drawing Service test suite data is located in `UnitTestDir\TestData\DrawingService`.

If you use dump mode, the dump files are written to `UnitTestDir\TestData\TestSuiteName\DumpFiles`. The DumpFiles folders are not created automatically. You must create the folder before running a test suite in dump mode.

Running Unit Tests From the Command Line

You can run all tests by executing the RunTests.php script located in `UnitTestDir\WebTier\Php`. From the command line type

```
php RunTests.php
```

Tests run from the command line are always run in validate mode.

Once all tests have completed, the script displays how many tests failed out of the total tests run. The following output shows no failure for 922 tests run:

```
Tests failed/run: 0/922
```

Populating Sample Data in a Database

The generate option on the result page allows generating data for specific operations instead of populating sample data for all operations in the test case. To generate data, select the checkboxes corresponding to the operations that you want to populate data for, and click Generate. The Generate option will also write each result to a dump file. The files created during generation are in DumpFiles folders. Examine them to ensure that correct data was populated in the repository. Note that the DumpFiles folders must exist before you execute Generate.

To ensure that each dump file gets correctly updated you should delete it and then recreate by running the Generate option without selecting any check boxes.

Adding New Tests

To add a new test, create a database that will contain the operations to be performed and the required parameters.

Add your test case to one or both of the following files:

`HttpUnitTests.php` (for HTTP tests)

`ApiUnitTests.php` (for API tests)

Use the following syntax in both files:

```
HtmlPrinter::PrintTestCases("YourTestCaseName", "PathToYourDatabase");
```

Note, if your tests are strictly HTTP or API add your tests in the respective file only.

If your tests contain operations that are not defined in the current script you need to add those operations. For example, if you want to add a new operation for the Resource Service that is going to be exposed through both HTTP and API then you have to modify `ResoureServiceApi.php` and `ResourceServiceHttpRequests.php` to include that operation. In addition, you need to add the operation in the long parsing list in `ExecuteOperation.php`.

Database Structure

The database contains five tables:

1 TestCase - high level table that contains the test names and the set of operations to be run. It contains the following fields:
   - ExecuteSequence - field used by the command line script to determine the order of which to execute the tests in a given database
   - TestName - name of the test
   - ParamSets - comma-separated list of all operations to be run (no spaces)
   - Description - explains what this test is doing
   - TestType - the type of the test. It should be either API or HTTP
   - Prerequisite - optional descriptive field that specifies if anything needs to be done before running the test (for example, add resources in the repository).

2 Params - contains parameters that are required to perform the operations. If some of the operations involve uploading a file enter the relative path to that file (the path relative to the PHP script not to the database file itself). It contains the following fields:

   - ParamSet - number that is used to retrieve all parameters for a given operation
   - ParamName - name of the parameter
   - ParamValue - parameter value to be used with the operation

3 ApiTestResults - stores results returned from API calls. It contains the following fields:

   - Description - explains what this operation does
   - ParamSet - references the operation from the Params table
   - ContentType - content type returned after executing the operation (gets set after running the test in generate mode)
   - Result - result returned from the operation. This field may contain either the actual result stored as a BLOB or a relative path to a file containing the sample data. If using a file to store the sample data, prefix the file path with the string “@@”. This ensures that the file will not be overwritten if Generate is performed on the corresponding operation.

4 HttpTestResults - has the same structure as ApiTestResults but stores results returned from HTTP requests.

5 CommonParams - contains parameters that are common for all operations. This table is optional and was created to reduce the size of the Params table. The tests will run even if you include the common parameters in the Params table.

Using the SQLite Console

The binaries for the GUI are built with the OEM components, and are located in: `*InstallDir*\Oem\SQLite\bin\win32`. The console program is named SQLiteGUI.exe.

To Create a New Database

Select Database ➤ New. The UI allows you to create either a blank database or a unit test template database that contains all necessary tables for adding new Web Server Extensions unit tests.

To Add or Edit Tests

Tests can be added/edited by either modifying each affected table in the database or using the separate form which is accessible through Database ➤ Add/Edit Tests.

To Execute SQL Statements

Type the statement in the text box at the bottom and select Execute from the Sql tab. There are some template statements in the dropdown menu. The query Vacuum compacts the database. SQL statements that return a table are not supported by the UI.

File Structure

All files for PHP unit testing are currently located in *`UnitTestDir`*`\WebTier\Php`.

- ApiUnitTests.php, HttpUnitTests.php - generate the HTML forms to allow running unit tests from a browser
- DrawingServiceAPI.php, DrawingServiceHttpRequests.php - implement functionality to execute API calls and send HTTP requests for the Drawing Service
- ExecuteOperation.php - parses the operation name and executes it
- FeatureServiceAPI.php, FeatureServiceHttpRequests.php - implement functionality to execute API calls and send HTTP requests for the Feature Service
- HtmlPrinter.php - outputs HTML tags to format the output
- HttpRequest.php - sends an HTTP request using the PHP curl extension

- Index.html, Menu.html - HTML for the start page
- ResourceServiceAPI.php, ResourceServiceHttpRequests.php - implement functionality to execute API calls and send HTTP requests for the Resource Service
- Result.php - implements result object that contains data returned from each operation
- Run.php - receives parameters from the HTML forms and executes the test
- RunTests.php - command line script that runs all tests.
- Utils.php - contains common functionality
- Validate.php - validates test results
- ValidatesUtils.php - contains helper functions used for validation

Miscellaneous

For any operation you can define the ALWAYSPASS parameter in the database. That will cause this operation to be flagged as successful regardless of the actual outcome. This parameter should be used with operations that are performing clean up and are not related to the tests performed.

To allow merging of database files, the databases are stored in the vault as SQL text files (dump files). At start up the infrastructure will check the time stamps of the database file and dump file and will create an updated database if needed. A new database file is created if there is no database file for the current test or the current database file has an older time stamp than the dump file. Similarly, a dump file is updated anytime a test suite is run using Generate mode. A dump file will not be updated if it is read-only or it has a newer time stamp than the current database file. If you add new data to a database, ensure that you run the tests in Generate mode at least once to update the dump file with the new data. Generate can be run (and it is strongly recommended to do so before submitting changes) without checking any of the boxes in the test result page. That will update the dump file if necessary.

Debugging Web Server Extensions

The Web Server Extensions are made of several layers. At the lower level is the MapGuide API. This C++ code mostly marshals arguments and remotely calls the actual API implementation in the MapGuide server. Among the small portion of the API that is executed in the Web Server Extensions is the coordinate system API, the web layout API and the map/layer API. The MapGuide API is used either by web applications - the AJAX viewer is an example of such an application - or by the MapGuide agent which receives and executes HTTP requests.

To debug your web application, whether it is implemented in PHP, Java or ASP.NET, please refer to your favorite tool's documentation.

Debugging the C++ API code requires attaching the debugger to the process that loads the MapGuide API DLLs. The actual process depends on the platform you are running on and the type of web server you are using.

Debugging from a PHP platform with IIS

The PHP engine as configured is loaded by a process named dllhost. The dllhost process will be created the first time you execute any PHP code. Unfortunately there might be more than one dllhost process running on your system at a time. A good way to determine which one is associated with the PHP engine is to inspect the user name with Windows task manager:

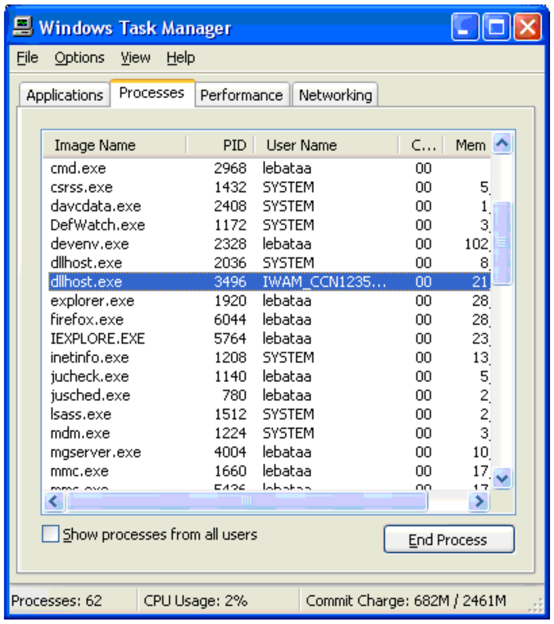
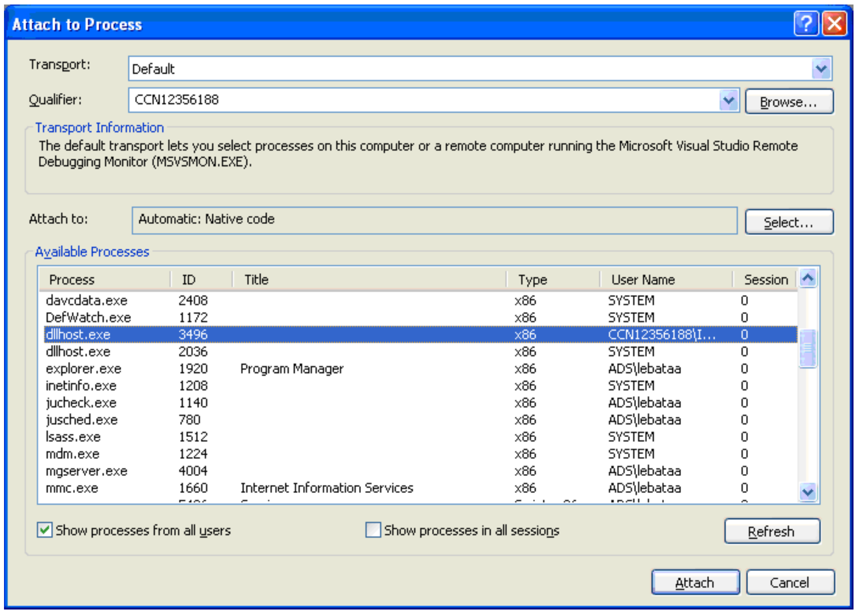

Windows Task Manager
File Options View Help
Applications Processes Performance Networking
Show processes from all users
End Process

Image Name	PID	User Name	C...	Mem
cmd.exe	2968	lebataa	00	
csrss.exe	1432	SYSTEM	00	5
davcdata.exe	2408	SYSTEM	00	1
DefWatch.exe	1172	SYSTEM	00	3
devenv.exe	2328	lebataa	00	102
dllhost.exe	2036	SYSTEM	00	8
dllhost.exe	3496	IWAM_CCN1235...	00	21
explorer.exe	1920	lebataa	00	28
firefox.exe	6044	lebataa	00	28
IEXPLORE.EXE	5764	lebataa	00	23
inetinfo.exe	1208	SYSTEM	00	13
jucheck.exe	1140	lebataa	00	5
jusched.exe	780	lebataa	00	2
lsass.exe	1512	SYSTEM	00	2
mdm.exe	1224	SYSTEM	00	3
mgserver.exe	4004	lebataa	00	10
mmc.exe	1660	lebataa	00	17

Processes: 62 CPU Usage: 2% Commit Charge: 682M / 2461M

The highlighted process with the user name IWAM_machinename was spawned by IIS and this is the one you will need to attach the debugger to. Note the process ID (3496 in this case). In Visual Studio choose Tools ➤ Attach to Process. In the dialog, select Show processes from all users. Locate the dllhost.exe instance in the list that has the process ID you noted, select it, and click Attach.



Transport: Default

Qualifier: CCN12356188

Transport Information

The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).

Attach to: Automatic: Native code

Available Processes

Process	ID	Title	Type	User Name	Session
davcdata.exe	2408		x86	SYSTEM	0
DefWatch.exe	1172		x86	SYSTEM	0
dllhost.exe	3496		x86	CCN12356188\I...	0
dllhost.exe	2036		x86	SYSTEM	0
explorer.exe	1920	Program Manager	x86	ADS\lebataa	0
inetinfo.exe	1208		x86	SYSTEM	0
jucheck.exe	1140		x86	ADS\lebataa	0
jusched.exe	780		x86	ADS\lebataa	0
lsass.exe	1512		x86	SYSTEM	0
mdm.exe	1224		x86	SYSTEM	0
mgserver.exe	4004		x86	ADS\lebataa	0
mmc.exe	1660	Internet Information Services	x86	ADS\lebataa	0

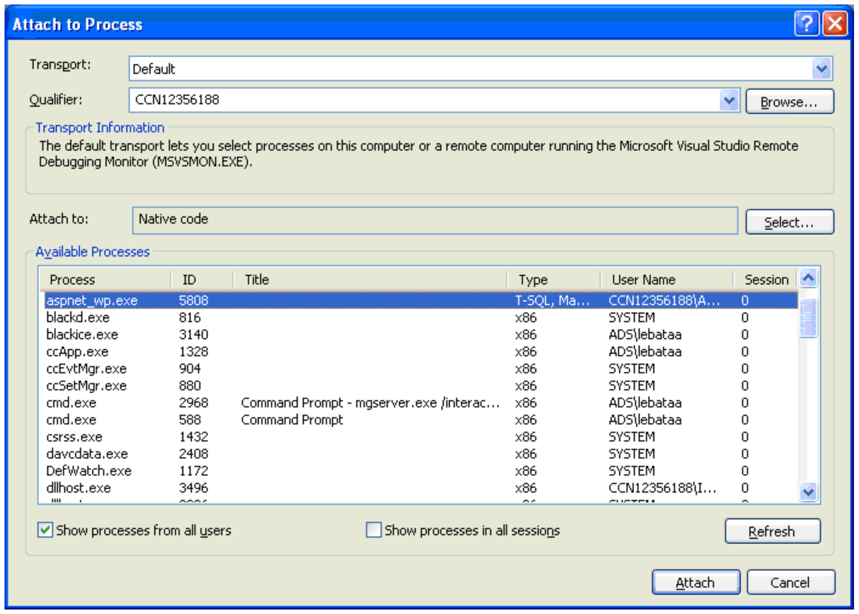
[✓] Show processes from all users [ ] Show processes in all sessions

At this point the breakpoints that you have set in the C++ code will be hit if this code is called by a PHP web application.

Debugging From an ASP.NET Platform

The process that executes ASP.NET scripts and loads the MapGuide C++ DLLs that you want to debug is named aspnet_wp.exe. This process will be created the first time you execute any ASP.NET code.

In Visual Studio choose Tools ➤ Attach to Process. In the dialog, select Show processes from all users. Locate aspnet_wp.exe in the list, select it, and click Attach. Make sure that the field Attach to is set to Native code. If not, click Select and select Native code in the list.


Attach to Process

Transport: Default

Qualifier: CCN12356188

Transport Information

The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).

Attach to: Native code

Available Processes

Process	ID	Title	Type	User Name	Session
aspnet_wp.exe	5808		T-SQL, Ma...	CCN12356188\A...	0
blackd.exe	816		x86	SYSTEM	0
blackice.exe	3140		x86	ADS\lebataa	0
ccApp.exe	1328		x86	ADS\lebataa	0
ccEvtMgr.exe	904		x86	SYSTEM	0
ccSetMgr.exe	880		x86	SYSTEM	0
cmd.exe	2968	Command Prompt - mgserver.exe /interac...	x86	ADS\lebataa	0
cmd.exe	588	Command Prompt	x86	ADS\lebataa	0
csrss.exe	1432		x86	SYSTEM	0
davcdata.exe	2408		x86	SYSTEM	0
DefWatch.exe	1172		x86	SYSTEM	0
dllhost.exe	3496		x86	CCN12356188\I...	0

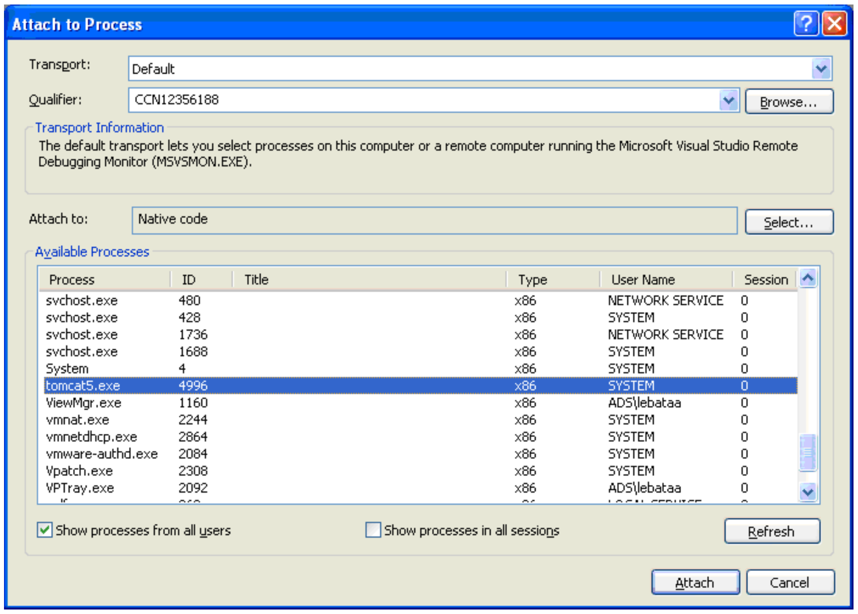
☑ Show processes from all users ☐ Show processes in all sessions

At this point the breakpoints that you have set in the C++ code will be hit if this code is called by an ASP.NET web application.

Debugging From a JSP Platform

The process that executes JSP scripts and loads the MapGuide C++ DLLs that you want to debug is named tomcat5.exe. This process will be created the first time you execute any JSP code.

In Visual Studio choose Tools ➤ Attach to Process. In the dialog, select Show processes from all users. Locate tomcat5.exe in the list, select it, and click Attach.


Attach to Process

Transport: Default

Qualifier: CCN12356188

Transport Information

The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).

Attach to: Native code

Available Processes

Process	ID	Title	Type	User Name	Session
svchost.exe	480		x86	NETWORK SERVICE	0
svchost.exe	428		x86	SYSTEM	0
svchost.exe	1736		x86	NETWORK SERVICE	0
svchost.exe	1688		x86	SYSTEM	0
System	4		x86	SYSTEM	0
tomcat5.exe	4996		x86	SYSTEM	0
ViewMgr.exe	1160		x86	ADS\lebataa	0
vmnat.exe	2244		x86	SYSTEM	0
vmnetdhcp.exe	2864		x86	SYSTEM	0
vmware-authd.exe	2084		x86	SYSTEM	0
Vpatch.exe	2308		x86	SYSTEM	0
VPTray.exe	2092		x86	ADS\lebataa	0

[✓] Show processes from all users   [ ] Show processes in all sessions

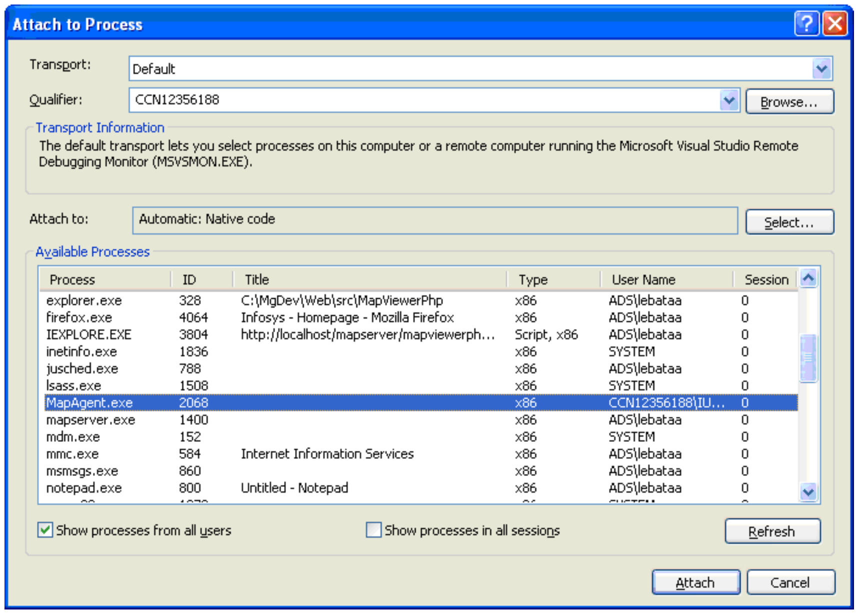
Debugging HTTP Requests

HTTP requests are executed by MapAgent.exe, which runs as a CGI process. This means that a process is created for each request and is destroyed after each request has been served. In order to debug the execution of a request in the agent or some DLL of the MapGuide server, you need to keep MapAgent.exe in memory long enough to be able to attach the map agent process to the debugger. To do this, edit the file `WebDir\webconfig.ini`, and set the `DebugPause` parameter to the number of seconds that gives you sufficient time to attach Visual Studio's debugger to the map agent. Thirty seconds should be a sufficient pause:

```
[AgentProperties]
# *****************************************************************
# M A P   A G E N T
#
# Property Name          Description
# -----------------------------------------------------------------
# DebugPause             Number of seconds the agent pauses before
#                        each request, for debugging purpose
# *****************************************************************
DebugPause=30
```

In Visual Studio, set breakpoints in the code you are interested in debugging, and then use a web browser to execute the HTTP request you want to debug. You can use the test forms for this.

When you execute the request, the browser will wait, as the map agent pauses for the amount of time set in the configuration file. Switch to Visual Studio and choose Tools ➤ Attach to Process. In the dialog, select Show processes from all users. Locate MapAgent.exe in the list, select it, and click Attach.



Transport: Default

Qualifier: CCN12356188

Transport Information

The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).

Attach to: Automatic: Native code

Available Processes

Process	ID	Title	Type	User Name	Session
explorer.exe	328	C:\MgDev\Web\src\MapViewerPhp	x86	ADS\lebataa	0
firefox.exe	4064	Infosys - Homepage - Mozilla Firefox	x86	ADS\lebataa	0
IEXPLORE.EXE	3804	http://localhost/mapserver/mapviewerph...	Script, x86	ADS\lebataa	0
inetinfo.exe	1836		x86	SYSTEM	0
jusched.exe	788		x86	ADS\lebataa	0
lsass.exe	1508		x86	SYSTEM	0
MapAgent.exe	2068		x86	CCN12356188\IU...	0
mapserver.exe	1400		x86	ADS\lebataa	0
mdm.exe	152		x86	SYSTEM	0
mmc.exe	584	Internet Information Services	x86	ADS\lebataa	0
msmsgs.exe	860		x86	ADS\lebataa	0
notepad.exe	800	Untitled - Notepad	x86	ADS\lebataa	0

[✓] Show processes from all users
[ ] Show processes in all sessions

At the end of the pause, the map agent will resume execution and your first breakpoint should be hit.