Autodesk
MapGuide® Enterprise

# Migrating from Autodesk MapGuide 6.5 to the New MapGuide Technology

This paper provides an overview of some of the changes from Autodesk MapGuide® 6.5 software and earlier releases to the new MapGuide technology. Furthermore, it provides a guide on how to move from existing Autodesk MapGuide 6.5 applications to the new MapGuide technology.

For simplicity, the new MapGuide technology, both the open source version and the commercial version, is referred to in this paper as "Autodesk MapGuide® Enterprise.” Autodesk MapGuide 6.5 and earlier versions are referred to as “Autodesk MapGuide.” All concepts discussed here apply to both the open source and commercial versions of the new MapGuide technology. Autodesk MapGuide® Studio is available separately.

## Contents

## Rationale

Autodesk MapGuide® Enterprise software has been developed in response to several customer and market drivers. Web development tools and technologies have come a long way in the last 10 years. MapGuide technology has evolved successfully but has not changed with them all. Modern web applications use the browser for presentation, while “business logic” runs in the web or middle tier. Complex MapGuide applications require significant logic in the client tier. Further, although PHP, ASP.NET, and JSP are the technologies of choice for web development, the core MapGuide platform integrates with none of them. Finally operations on the data, such as buffering, should operate on the geometry data, not the graphics that have been transformed and clipped.

Autodesk MapGuide Enterprise also serves a larger market. Currently, penetration of the Windows Server® operating system is only around 48 percent while adoption of Linux® operating system is growing rapidly. Not having Linux® support can be a barrier in some environments. Support for Apache is also important because Apache is the number one web server by far.

Increasingly, web mapping solutions require a raster-based viewing technology to support the broadest range of Internet users. Users need a raster-based viewing technology that performs well, is scalable, and fits into the core platform. There is a need to bring the capabilities of the raster based platform at par with the plug-in versions of MapGuide.

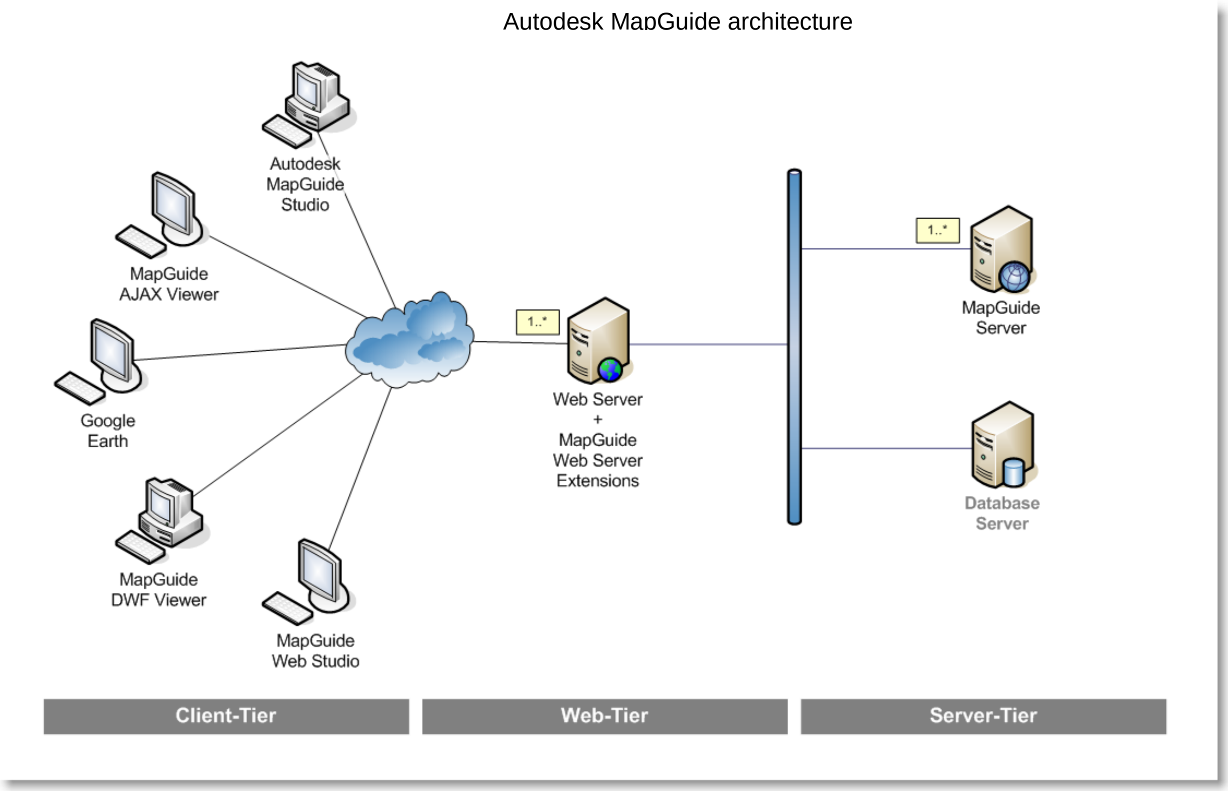
Autodesk MapGuide Enterprise also uses and enhances DWF™ technology. In particular it takes advantage of the strong printing and plotting capabilities of DWF and provides excellent visual fidelity with AutoCAD® software drawings. It also enables many companies to move to a single viewer technology base.

The MapGuide architecture is more than 10 years old. There is a need to modernize the platform in order to provide future support for 2.5D data and data analysis with 3D presentation; additional Open Geospatial Consortium (OGC) initiatives like WFS (Web Feature Service); and Web Services and support for rich metadata.

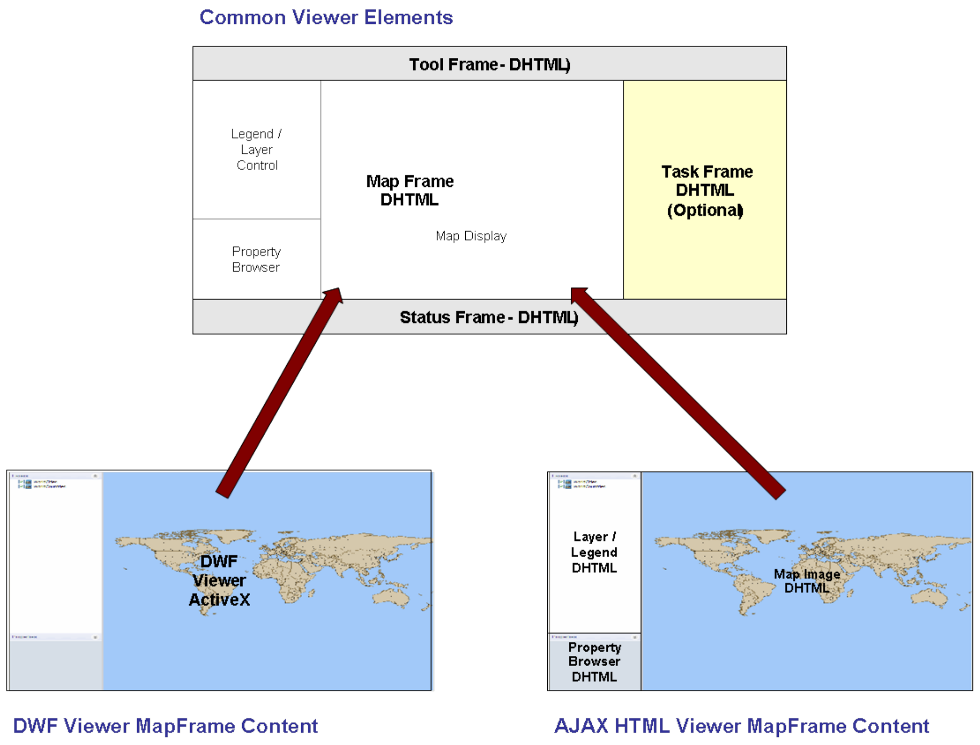
## Autodesk MapGuide Enterprise—A New Architecture

The new web mapping platform, Autodesk MapGuide Enterprise, goes far beyond Autodesk MapGuide 6.5 software, providing new architecture, programming language support, data access methods, viewing options, and authoring environment.

The new platform is available as open source software (MapGuide Open Source) and as a commercial version (Autodesk MapGuide Enterprise). The new MapGuide technology can be installed, customized, and developed on multiple platforms, including the Linux operating system.

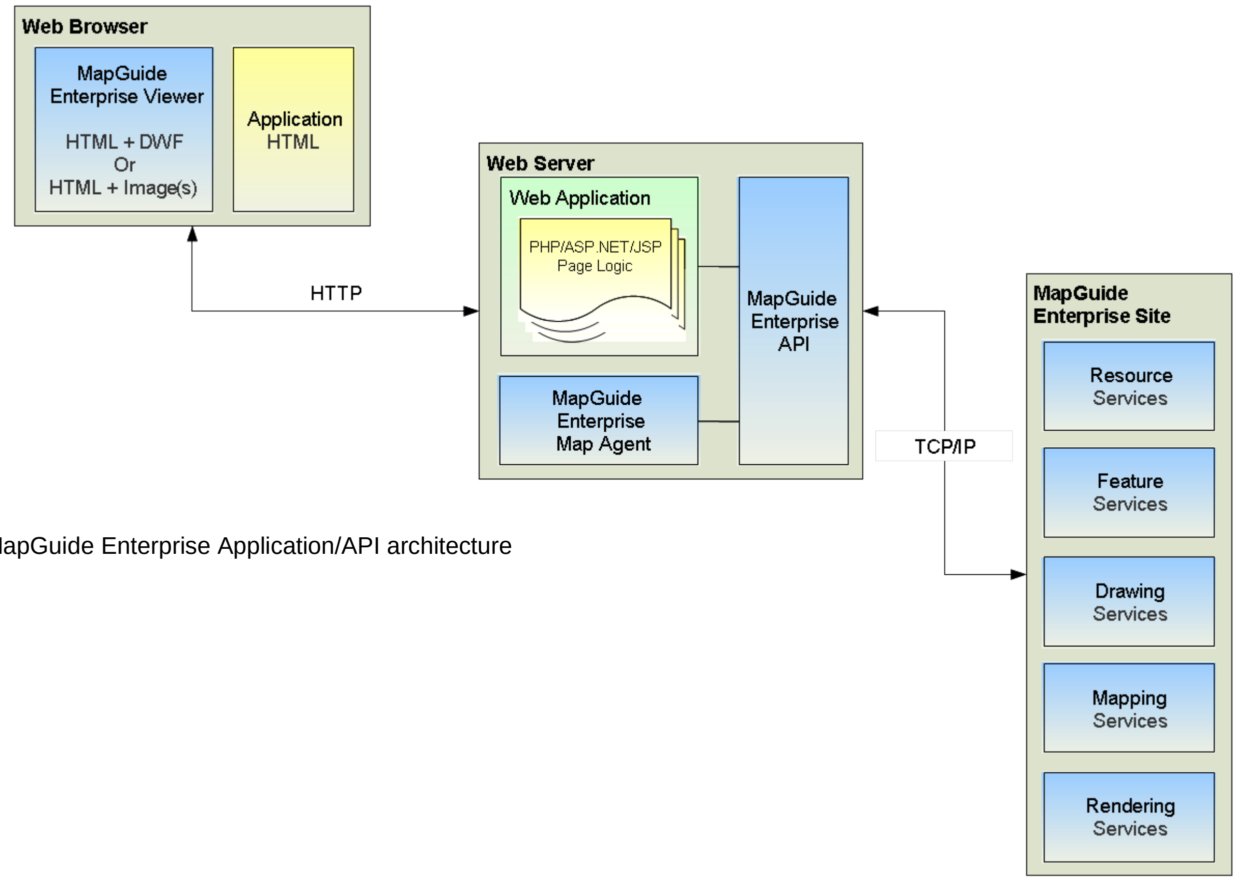


Autodesk MapGuide Enterprise has architecture different from that of Autodesk MapGuide 6.5, moving to server-side functionality whereby much of the mapping and geospatial functions are executed on the server. The API (application programming interface) is exposed primarily through the web server extensions, which are available in the .NET, Java®, and PHP development environments. The choice of viewer technologies (an ActiveX®-based viewer using DWF technology as well as a non plug-in viewer based on AJAX technology) has been made possible as a result of this shift in architecture. The application functionality available to the DWF and AJAX viewers is consistent. Perhaps most important, the shift to server-side application functionality has enabled Autodesk MapGuide Enterprise to serve as a web-based geospatial engine capable of handling geospatial queries that can be executed on several data sources such as SHP, SDF, Oracle®, and ArcSDE®™, among others, via a single Feature Data



Objects (FDO) API. This capability enables the developer to use just one API on disparate data sets, focusing on the application rather than the data format.

This shift in architecture requires that steps be taken to migrate an application built using the various components of Autodesk MapGuide 6.5. With the new MapGuide technology, tedious processes, such as setting up the server environment, configuring the data sources, loading data, and programming the application, have been greatly reduced. Although it may seem that migration requires extra effort, the benefits of the new technology outweigh the burden as Autodesk MapGuide Enterprise reduces the development time for both basic and advanced functions due to its powerful API. Autodesk MapGuide, on the other hand, requires significant client-side scripting to develop custom functions because of the limitations of the Autodesk MapGuide 6.5 client-side oriented API.



MapGuide Enterprise Application/API architecture

## Comparing Autodesk MapGuide 6.5 Components with Autodesk MapGuide Enterprise

The following table lists the Autodesk MapGuide 6.5 components with the equivalent component of Autodesk MapGuide Enterprise.

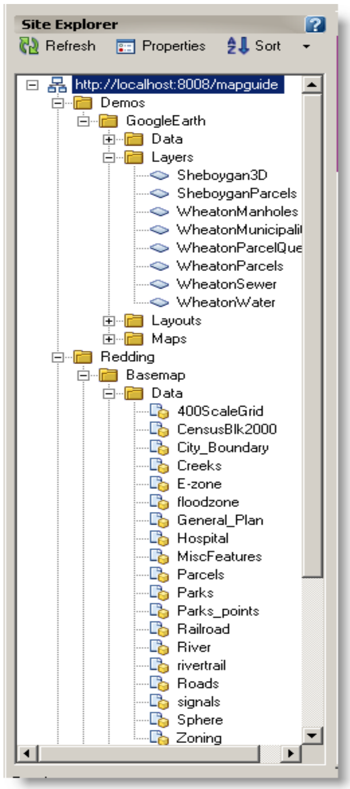
MapGuide Component Comparison

| **Autodesk MapGuide 6.5 Component** | **Autodesk MapGuide Enterprise Component** |
|---|---|
| Server | Server/Web Server Extensions |
| Author | Autodesk MapGuide Studio |
| ActiveX Plug-in | Autodesk® DWF™ Viewer |
| Dynamic Authoring Toolkit | Web Server Extensions API |
| SDF COM Toolkit | Web Server Extensions API |
| MapGuide Raster Workshop | Autodesk MapGuide Studio |
| Autodesk MapGuide API | Web Server Extensions API, Client-Side Wrapper API |
| SDF Loader | Autodesk MapGuide Studio |
| LiteView | AJAX Viewer |
| Server Administrator | Web-Based Server Site Administrator |

Autodesk MapGuide 6.5, the MapGuide Server provided functionality to create data sources to geospatial and database sources; create groups and users to access data sources; start and stop the MapGuide agent; and display usage and error logs. These functions are now available through the web-based Site Administrator and using Autodesk MapGuide Studio.

In Autodesk MapGuide 6.5, MapGuide Author enabled users to create Map Window Files (MWF), which are embedded into applications. With Autodesk MapGuide Enterprise, these MWF files are replaced with *resources* that consist of data sources, layers definitions, and map definitions, all of which reside on the server.

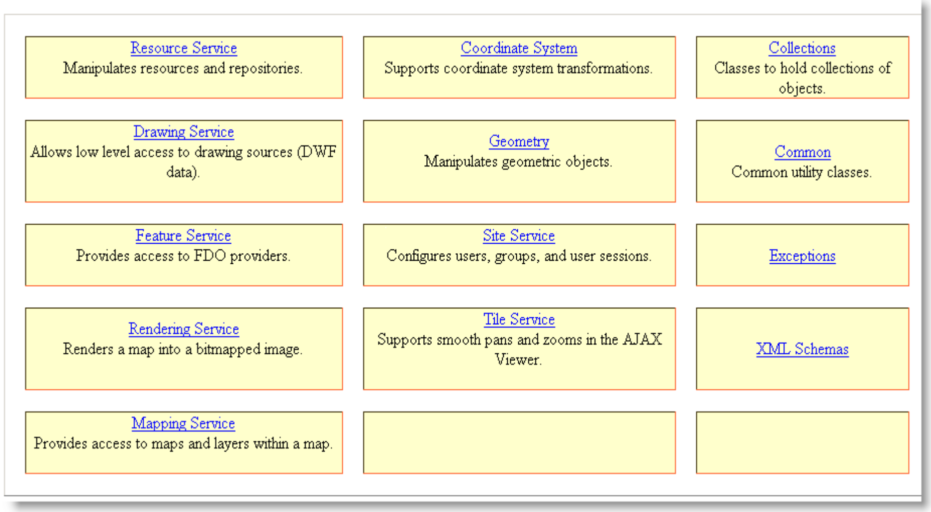
The Autodesk MapGuide 6.5 plug-in, which is available in ActiveX or Java, is replaced by Autodesk's commonly used DWF Viewer. The DWF Viewer has the advantage of being the standard viewer technology for all Autodesk product-generated DWF files, including mechanical and architectural files. It is a single, unified viewer for all digital design content.



Site Resources

The Autodesk MapGuide 6.5 Dynamic Authoring Toolkit (DAT) enables users to manipulate MWF files by converting them to XML format, modifying the XML data, and then converting the XML back to an MWF file, which is then delivered to the browser. The DAT has been replaced with an even more robust Web Tier API that gives users access to all elements of a map. Users can add, edit, or delete any portion of a map on the fly.

The Autodesk MapGuide 6.5 SDF COM Toolkit is a server-side COM object that enables users to query and manipulate SDF data sources in Autodesk MapGuide 6.5. The feature service API in the Web Extensions builds on the FDO read-write provider capabilities and enables users to query and manipulate geospatial data, including SDF and other supported formats. It is now possible through the feature service API in the Web Extensions to dynamically create data stores and manipulate them as necessary.

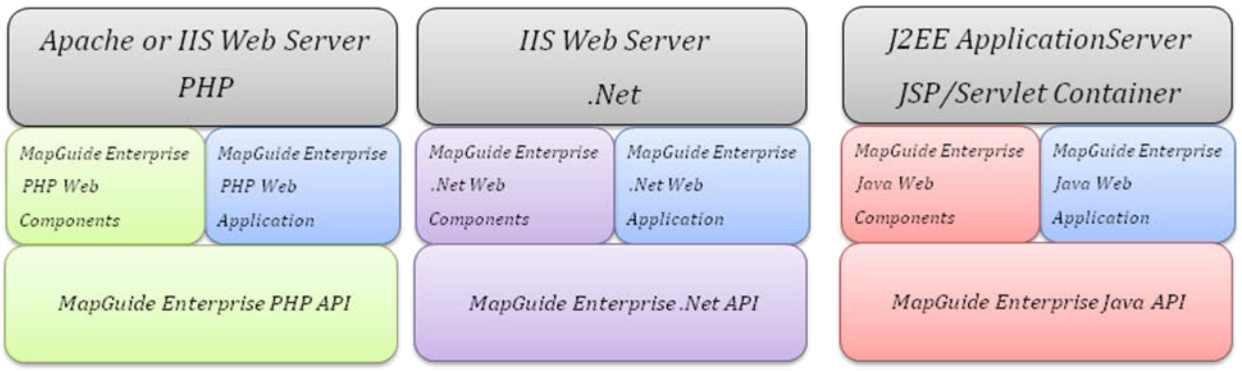


| | | |
|---|---|---|
| Resource Service<br>Manipulates resources and repositories. | Coordinate System<br>Supports coordinate system transformations. | Collections<br>Classes to hold collections of objects. |
| Drawing Service<br>Allows low level access to drawing sources (DWF data). | Geometry<br>Manipulates geometric objects. | Common<br>Common utility classes. |
| Feature Service<br>Provides access to FDO providers. | Site Service<br>Configures users, groups, and user sessions. | Exceptions |
| Rendering Service<br>Renders a map into a bitmapped image. | Tile Service<br>Supports smooth pans and zooms in the AJAX Viewer. | XML Schemas |
| Mapping Service<br>Provides access to maps and layers within a map. | | |

Autodesk MapGuide Enterprise Web Tier API Model

The Autodesk MapGuide 6.5 Raster Workshop enables users to set up raster image catalogs and manipulate raster files to create a subsampled version that can then be used to create scale-dependent layering of raster images. Autodesk MapGuide Studio has built-in functionality that enables users to perform the same tasks.

The Autodesk MapGuide 6.5 API is a client-side API that allows the manipulation of MWF files. The API is browser dependent, making it difficult to create Autodesk MapGuide 6.5 applications that are compatible across browsers. The API is also limited in its functionality and requires sophisticated programming to create special functionality.


Apache or IIS Web Server
PHP
MapGuide Enterprise PHP Web Components
MapGuide Enterprise PHP Web Application
MapGuide Enterprise PHP API
IIS Web Server
.Net
MapGuide Enterprise .Net Web Components
MapGuide Enterprise .Net Web Application
MapGuide Enterprise .Net API
J2EE ApplicationServer
JSP/Servlet Container
MapGuide Enterprise Java Web Components
MapGuide Enterprise Java Web Application
MapGuide Enterprise Java API

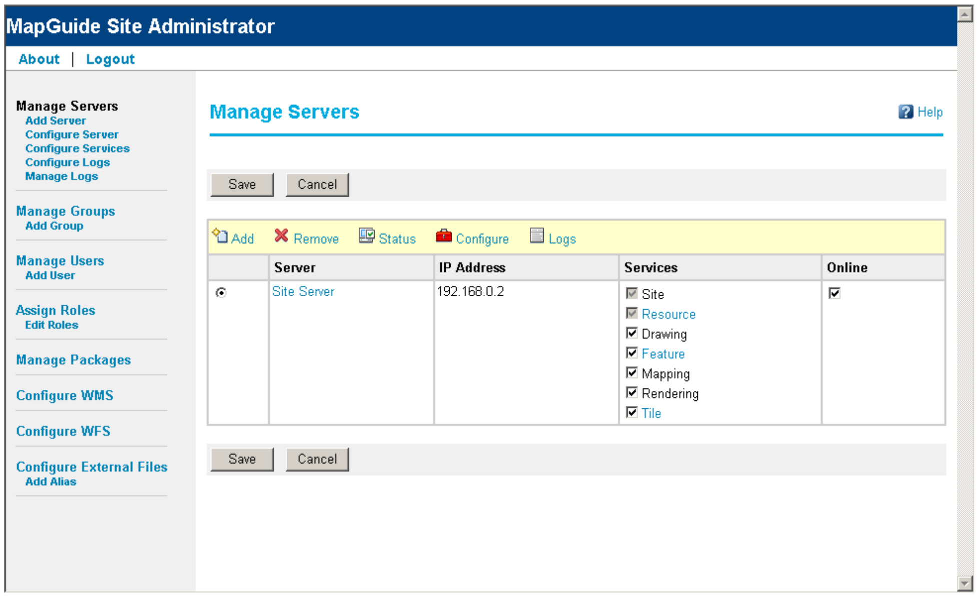
Autodesk MapGuide Enterprise Development Options

The Autodesk MapGuide Enterprise Web Tier API has been re-engineered as a server-side API. As a result, users can develop much more robust and complex applications. Because the API is server side, it is not browser dependent. Users can therefore write cross-browser compatible applications with any of the three supported development environments (.NET, Java, PHP). Users can easily port between languages with the assurance that all functions are available regardless of language. Autodesk MapGuide Enterprise continues to have a client-side API for manipulating user interaction between the frames, but most mapping operations are executed in the business logic tier on the web server.

The Autodesk MapGuide 6.5 SDF Loader is a command-line utility that enables users to create and convert geospatial data into SDF format. It also enables users to create batch files to perform bulk conversion of data on regular basis. Autodesk MapGuide Enterprise Studio replaces the SDF Loader and enables users to create load procedures that can be scheduled.

Autodesk MapGuide 6.5 LiteView software is a servlet-based software that enables users to build image-based applications that don't require a plug-in. In Autodesk MapGuide Enterprise, LiteView has been replaced with the AJAX viewer. Developers can now build one application that can be viewed using either the DWF plug-in or the AJAX viewer. Developers no longer need to write separate code to publish non-plug-in based applications.

The Autodesk MapGuide 6.5 Server Administrator tool enables users to create data sources to geospatial and database sources; create groups and users to access data sources; start and stop the MapGuide server; and display usage and error logs. These functions are now available in the Autodesk MapGuide Enterprise Site Administrator and Autodesk MapGuide Enterprise Studio.



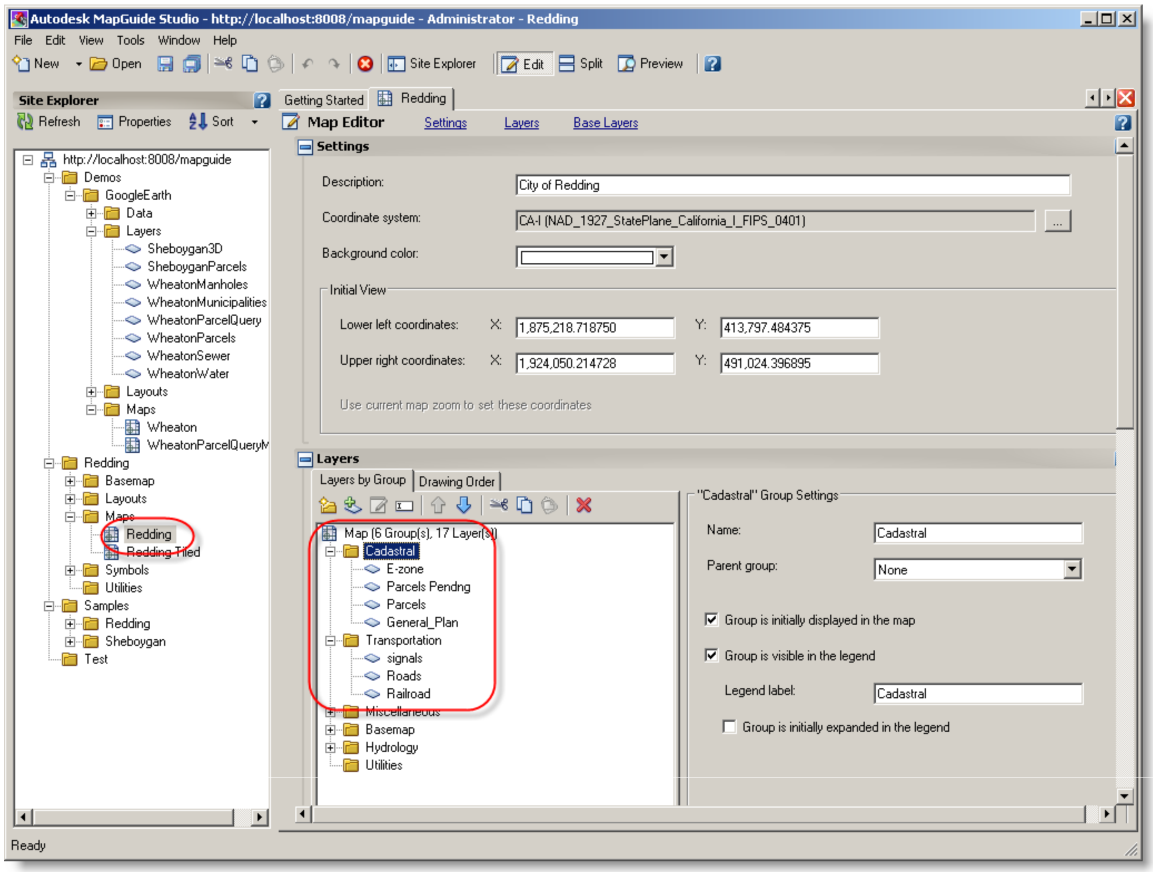
| | Server | IP Address | Services | Online |
|---|---|---|---|---|
| ● | Site Server | 192.168.0.2 | ✓ Site<br>✓ Resource<br>✓ Drawing<br>✓ Feature<br>✓ Mapping<br>✓ Rendering<br>✓ Tile | ✓ |

Autodesk MapGuide Enterprise Web based server administrator

## MWX/MWF Requirements

Autodesk MapGuide Enterprise performs stylization on the server and hence no longer requires a MWF or MWX file. The definitions and rules stored in the MWF/MWX files are now stored on the server as resources. Maps no longer contain layers. Instead, a map definition references layer definitions, allowing a layer to be defined once and used in many maps.

Resources define things such as maps, layers, and data connections. Each resource is defined by an XML document that is stored in a hierarchical resource repository. Each repository can be thought of as a virtual file system that is managed within a resource database. The resource database lives on the Autodesk MapGuide Enterprise Server and provides centralized storage for a collection of resource repositories.



Settings

Description: City of Redding

Coordinate system: CA-I (NAD_1927_StatePlane_California_I_FIPS_0401)

Background color:

Initial View

Lower left coordinates: X: 1,875,218.718750 Y: 413,797.484375

Upper right coordinates: X: 1,924,050.214728 Y: 491,024.396895

Use current map zoom to set these coordinates

Layers

"Cadastral" Group Settings

Name: Cadastral

Parent group: None

[✓] Group is initially displayed in the map

[✓] Group is visible in the legend

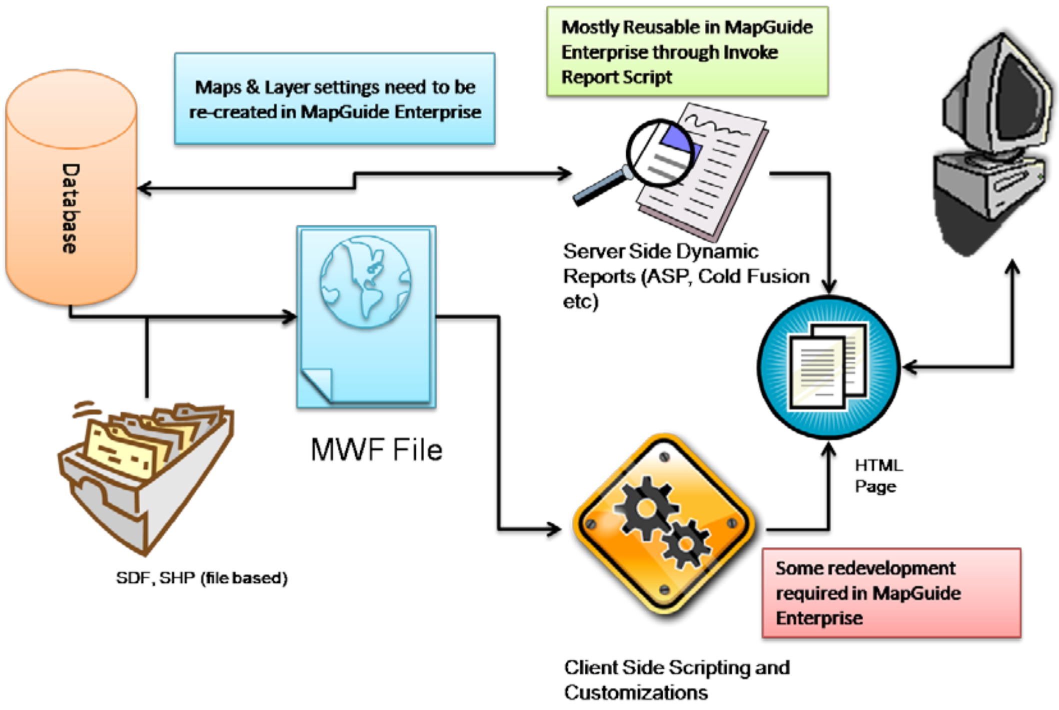
Legend label: Cadastral

[ ] Group is initially expanded in the legend

Map Definition created using Autodesk MapGuide Studio

## Migration Strategy

The following diagram shows a typical Autodesk MapGuide 6.5 application. The components highlighted in blue can be re-created using more advanced capabilities such as true color support and transparency in MapGuide Enterprise. The section in green is mostly reusable through a script (see Appendix for source code) that redirects report requests from Autodesk MapGuide Enterprise to the existing Autodesk MapGuide 6.5 report. The section in red requires redevelopment because of the shift to a server-side API in Autodesk MapGuide Enterprise.

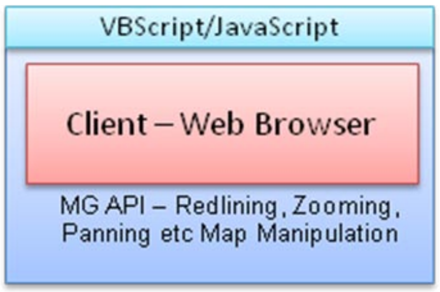

Database
Maps & Layer settings need to be re-created in MapGuide Enterprise
Mostly Reusable in MapGuide Enterprise through Invoke Report Script
Server Side Dynamic Reports (ASP, Cold Fusion etc)
MWF File
HTML Page
SDF, SHP (file based)
Some redevelopment required in MapGuide Enterprise
Client Side Scripting and Customizations

A Typical MapGuide 6.5 application

The transition from Autodesk MapGuide 6.5 to Autodesk MapGuide Enterprise requires some planning and redevelopment of the customizations that were done on Autodesk MapGuide 6.5. Use the following sections as a step-by-step guide to migrating an existing application from Autodesk MapGuide 6.5 or earlier to Autodesk MapGuide Enterprise.

## Phase 0—Planning and Setup

The initial phase of the process should be to plan the migration of the application. In this phase, a clear plan that establishes goals for the application, a list of custom functions that need to be created, development of the web interface design, selection of development platform, and identification of the server hardware configuration or architecture should be written. Responsibilities, timelines, and milestones can be included for project management purposes. Developing a written plan enables developers to successfully migrate the existing application without encountering major hurdles.



Autodesk MapGuide 6.5 custom functions

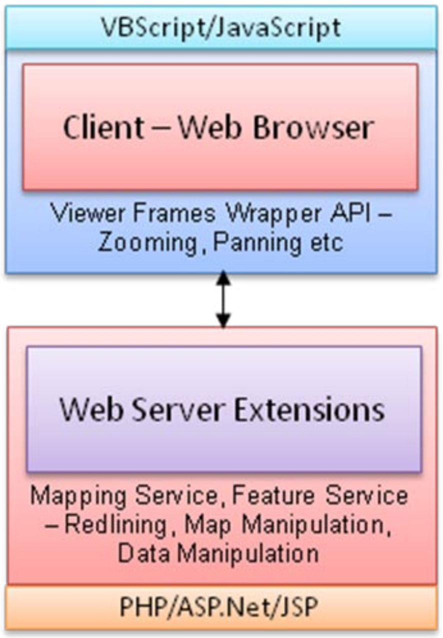
Goals

Possible goals of the migration include the following:

- Enhance functionality
- Eliminate the plug-in viewer
- Streamline spatial data updates
- Reduce custom programming by using built-in Autodesk MapGuide Enterprise functions
- Upgrade server and application software
- Change application development platform

List of Functions

One of the key elements of Autodesk MapGuide 6.5 sites is the custom programming that is performed using the Autodesk MapGuide 6.5 Viewer API to develop advanced applications. The plan should take into account all custom programming functions that existed on the original application and any new functionality that should be integrated. Creating a list of existing functions and then identifying how those functions will be duplicated with Autodesk MapGuide Enterprise helps identify the level of effort required to migrate the application. Furthermore, some functions that needed to be programmed using the API may be standard functions in Autodesk MapGuide Enterprise. For example, many Autodesk MapGuide applications have custom Zoom GoTo functionality that can be replaced with zooming/search functions that can be created using Studio.


VBScript/JavaScript
Client – Web Browser
Viewer Frames Wrapper API – Zooming, Panning etc
Web Server Extensions
Mapping Service, Feature Service – Redlining, Map Manipulation, Data Manipulation
PHP/ASP.Net/JSP

Autodesk MapGuide Enterprise custom functions

Web Interface

The migration of the Autodesk MapGuide application could provide an opportunity to redesign the existing interface. Since Autodesk MapGuide Enterprise has additional functionality, much of the custom interface programming and design that was created for Autodesk MapGuide sites is no longer necessary. For example, with Autodesk MapGuide Enterprise, a built-in Properties Pane enables users to click map objects and obtain attribute information. This functionality was not available out-of-the-box with Autodesk MapGuide 6.5, so many developers created custom functionality to perform these actions. Developers also created custom toolbars to allow for interaction with the map. With Autodesk MapGuide

Enterprise, these custom functions can be incorporated into the Task Pane, eliminating the need for a custom toolbar.

In some cases, developers may choose to leave the interface as is and integrate Autodesk MapGuide Enterprise into the existing web design. This could occur in a situation where users are comfortable with the existing design and changing it would adversely affect the usage of the application. In that case, it would be prudent to maintain the interface and eliminate programming that can be replaced with out-of-the-box functionality.

### Selection of Development Platform

Autodesk MapGuide Enterprise supports the most common programming languages used to perform web development, .NET, PHP, and JSP. Developers can use these languages to access the robust API provided with Autodesk MapGuide Enterprise. Unlike Autodesk MapGuide 6.5, where the client-side APIs are browser and language dependent, Autodesk MapGuide Enterprise provides full compatibility between all three Web API Extensions. This means that application developers can use any of the development platforms or port from one application to another and be confident that the application will perform the same way.

The choice of development platform depends on the preference, experience, and skill of the programmers. Each platform has unique advantages and disadvantages.

### Server Configuration

Migration of the Autodesk MapGuide 6.5 application requires the acquisition of a development server. Developers typically have at least one staging or development server for testing. With Autodesk MapGuide Enterprise, it is recommended that two servers be configured and dedicated to the software. One server is used for staging and the other for the deployment server. It is possible to install Autodesk MapGuide Enterprise on the same server as Autodesk MapGuide, but it’s not recommended.

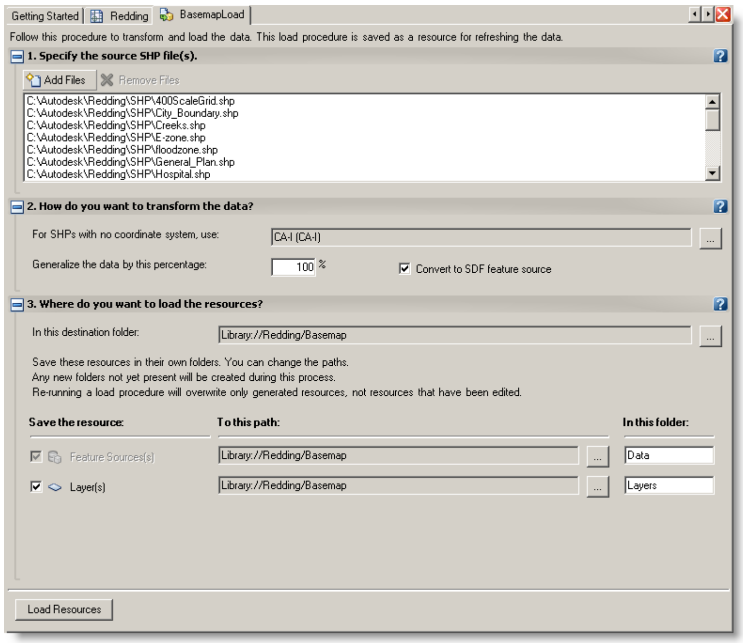
The hardware required to run Autodesk MapGuide Enterprise depends on many factors, including the organization’s networking infrastructure, development platform standards, and developers’ capability. Since Autodesk MapGuide Enterprise can now run on Linux as well as the Microsoft® Windows® operating system, the role of the organization’s IT staff becomes more important in the decision-making process.

## Phase 1—Configuring and Loading Data

The first phase of migration is to configure and load the spatial and tabular data into Autodesk MapGuide Enterprise, and connect to external databases. With Autodesk MapGuide Enterprise, setting up data sources is accomplished by using the Studio software. As with Autodesk MapGuide 6.5, creating an organized data structure is crucial to keeping the data organized. With Autodesk MapGuide Enterprise, rather than creating file folders on the server disk drives, folders are created using Studio. It is recommended that the file folders mimic the existing Autodesk MapGuide structure to maintain consistency. However, if the existing structure is not well organized, the migration process can provide an opportunity to correctly organize file structure and data.

Once the folders have been created, the spatial and tabular data can be imported into Autodesk MapGuide Enterprise using the Studio software. Unlike Autodesk MapGuide 6.5, spatial data sources do not need to be created with the administrator. Studio establishes the data source connection when the data is loaded.

Another major difference between Autodesk MapGuide 6.5 and Autodesk MapGuide Enterprise is that Autodesk MapGuide Enterprise eliminates the need to create elaborate data update routines when loading data. Many administrators of Autodesk MapGuide sites create batch files using the SDF Loader to define a process that updates the spatial data. This process is run either automatically or manually. In many case, administrators do not create update routines but rather execute the entire process manually. With Autodesk MapGuide Enterprise, when data is loaded, a load procedure is created. The load procedure can include several data sets so that when data needs to be updated, the load procedure can be run simply and quickly.



Follow this procedure to transform and load the data. This load procedure is saved as a resource for refreshing the data.

**1. Specify the source SHP file(s).**

```
C:\Autodesk\Redding\SHP\400ScaleGrid.shp
C:\Autodesk\Redding\SHP\City_Boundary.shp
C:\Autodesk\Redding\SHP\Creeks.shp
C:\Autodesk\Redding\SHP\E-zone.shp
C:\Autodesk\Redding\SHP\floodzone.shp
C:\Autodesk\Redding\SHP\General_Plan.shp
C:\Autodesk\Redding\SHP\Hospital.shp
```

**2. How do you want to transform the data?**

For SHPs with no coordinate system, use: CA-I (CA-I)

Generalize the data by this percentage: 100 %

✓ Convert to SDF feature source

**3. Where do you want to load the resources?**

In this destination folder: Library://Redding/Basemap

Save these resources in their own folders. You can change the paths.
Any new folders not yet present will be created during this process.
Re-running a load procedure will overwrite only generated resources, not resources that have been edited.

| Save the resource: | To this path: | In this folder: |
|---|---|---|
| ✓ Feature Sources(s) | Library://Redding/Basemap | Data |
| ✓ Layer(s) | Library://Redding/Basemap | Layers |

Autodesk MapGuide Studio showing a load procedure.

## Phase 2—Building Map Layers/Creating Map

The next phase of the migration is to build layers that can be used in Autodesk MapGuide Enterprise maps. This process is slightly different from the original procedure used with Autodesk MapGuide and Autodesk MapGuide Author. With Autodesk MapGuide, if a user needs to add a layer to a map, the user would open Autodesk MapGuide Author and create the layer in the MWF file. If the application had multiple MWF files, the user would need to create or copy the layer into each MWF file. The user would have to format each layer individually within each MWF file. This process creates a situation where two different layers could be formatted differently even though they were intended to be the same. With Autodesk MapGuide Enterprise, users create layers and define the look of the layer, including themes, tooltips, and more, one time and then use the layer definition within various map definitions. This new functionality within Autodesk MapGuide Enterprise helps eliminate the redundant task of having to create a layer multiple times.

Of course, while migrating the existing application to Autodesk MapGuide Enterprise, users will want to duplicate the style of the layer that existed in the Autodesk MapGuide application. However, since Autodesk MapGuide Enterprise has much better rendering capabilities, users need to decide whether to retain the layer formatting of the previous application.

Once the map layers have been defined, a map can be created. With Studio, the map comprises the layers that are already formatted. Essentially, the map serves as a view of the data and layers that have been defined. The benefit of this approach is that if the format or display of the layers needs to be changed, they can be changed once and reflected wherever they are being used. With Autodesk MapGuide 6.5, in contrast, if the layer is shown on several MWF files, each layer must be changed in each MWF file individually.

The following tables show how Autodesk MapGuide 6.5 files and their elements are migrated to Autodesk MapGuide Enterprise resources.

| MWF/MWX | |
|---|---|
| Web Layout | A default web layout is created where the MWF popup menu is converted to a context menu. |
| Map Definition | Map window properties are converted to map definition properties. |
| Layer Definitions | See MLF.<br>Note: Layers that reference remote MapGuide servers are not migrated. |
| Symbol Libraries | Created from MWF API symbols and point layer symbols. |
| Feature Sources | See MLF and UDL. |
| Drawing Sources | See MLF. |

| MLF | |
|---|---|
| Dynamic Text, Polyline, and Polygon Layers<br><br>Feature Layer Definition | **Feature Layer Definition**<br>• Text layer creates a point stylization<br><br>**Feature Sources**<br>• SHP provider feature source for SHP SDP<br>• SDF3 provider feature source for SDF 2.x SDP<br>• SDF3 provider feature source for DWG™<br>• Oracle provider feature source for Oracle SDP<br>• Microsoft SQL Server provider feature source for Microsoft SQL SDP<br><br>**Attribute Sources**<br>For each secondary table, an attribute source is created and a join is added to the feature source:<br>• ODBC provider feature source for Microsoft Access OLE DB<br>• SHP provider feature source for SHP SDP<br>• SDF3 provider feature source for SDF 2.x SDP<br>• SDF3 provider feature source for DWG<br>• Oracle provider feature source for Oracle SDP<br>• Microsoft SQL Server provider feature source for Microsoft SQL OLE DB |
| Dynamic Point Layer | **Feature Layer Definition**<br>• Same as above, plus<br>• ODBC provider feature source for Microsoft Access SDP<br><br>**Attribute Sources for Secondary Tables**<br>• Same as above<br><br>**Symbol Library** |

| MLF | |
|---|---|
| Dynamic Raster Layer | **Raster Layer Definition**<br><br>**Raster Provider Feature Sources**<br>• JPEG (.*jpg*) converted to TIFF-based feature source<br>• Microsoft Windows Bitmap (.*bmp*) converted to TIFF-based feature source<br>• TrueVision Targa 2.0 format (.*tga*) converted to TIFF-based feature source<br>• Portable Network Graphic (.*png*) converted to TIFF-based feature source<br>• Tagged Image File Format (.*tif*) converted to TIFF-based feature source<br>• CALS MIL-R-28002A Type 1 (.*cal*) converted to TIFF-based feature source<br>• Enhanced Compression Wavelet (.*ecw*) converted to ECW-based feature source<br>• MrSID® (.*sid*) converted to MrSID-based feature source |

| MLF | |
|---|---|
| Dynamic DWF Layer | • Drawing layer definition<br>• Drawing source |
| Dynamic DWG Layer | • Drawing layer definition<br>• Drawing source |
| Static Layers | Migrated as above using the following data source exceptions:<br>• Point converted to SDF 3 provider<br>• Text converted to SDF 3 provider<br>• Polyline converted to SDF 3 provider<br>• Polygon converted to SDF 3 provider<br>• Raster converted to TIFF-based raster provider<br>• DWG converted to drawing source<br>• DWF converted to drawing source |
| Buffer Layer | • Drawing layer definition<br>• Drawing source |
| Redline Layer | • Drawing layer definition<br>• Drawing source |

| UDL | |
|---|---|
| | • SHP provider feature sources for SHP SDP<br>• SDF provider feature sources for SDF SDP<br>• ODBC provider feature sources for Microsoft Access OLE DB |

It should be noted that the DMT allows for basic conversion of the information so that developers are not starting from square one when creating maps. However, some formatting might still be required once the information has been converted. In fact, most developers will want to use the new features of Autodesk MapGuide Enterprise to enhance the format of the map.

## Phase 3—Migrating Applications

Once the maps have been created and assuming development platform and server architecture have been selected and set up, the developer can then begin the process of migrating the application. This process is the most time-consuming portion of the migration. The list of functions identified in the plan is important to the development process. The list should be refined to reflect any functionality that can be eliminated by using out-of-the-box Autodesk MapGuide Enterprise functionality.

The most crucial functionality to be migrated is object selection and interaction with the server, Zoom GoTo functionality, and reporting. These functions are the ones most commonly used in Autodesk MapGuide applications.

Developers can take one of two approaches to writing the actual code for the site. Developers can either start from scratch and write the entire code base and integrate either the new design or the old one into the application. Or the developer can duplicate the existing application and modify it by deleting old code and adding new code. It is up to the developer as to which approach works best, but starting from scratch could be more efficient since the API is completely different and none of the old API calls can be reused. Reports that were created for use with Autodesk MapGuide can be reused with Autodesk MapGuide Enterprise via the Invoke Report script (see Appendix for source code).

Further discussion of the Web Tier API and sample code are provided in the code examples section of this document.

## Phase 4—Testing and Feedback

Once the new site is complete, assuming that the developer has tested functionality as it is developed, the application can be rolled out for beta testing. The user group should be familiar the inner workings of the old application and should be trained on how to use the new site and informed about new features or differences in the application. Beta users should be provided with feedback forms so that they can clearly communicate problems to the developer.

## Phase 5—Final Changes

The next step is to collate and understand feedback from beta testers. This may require meeting with the beta testers to help ensure that the feedback is clearly communicated. Once the list of feedback items has been compiled, beta testers should stop using the application so that the developer can update it. Again, it is assumed that the developer will test and fix any problems.

## Phase 6—Deployment

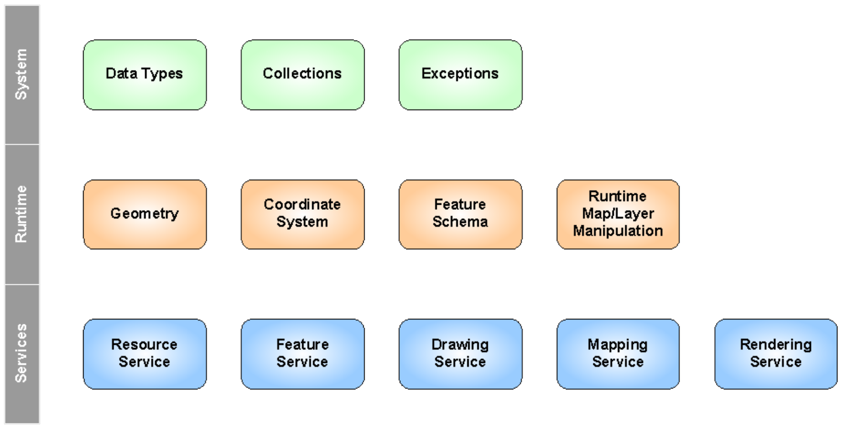
Once the final changes have been made, the site should then be deployed on the live server. It is up to the individual organization as to whether to maintain the old MapGuide site as well as the new one or to replace the old site. Before the new site is launched, it may be necessary to train users on new features of the application. A plan should be developed to gather feedback once the users start working with the new site. Further development can be done on the staging server and deployed to the live server as needed. A time frame for decommissioning the old site should be given to all users.

# Description of Web Tier API

Having the flexibility to choose whether to implement MapGuide with a Microsoft Windows Server or a Red Hat® Linux Server gives organizations the ability to choose the most appropriate operating system and development tools. The Web Tier API is fully consistent across both operating systems.

## APIs for MapGuide

Several API objects are available to developers working with Autodesk MapGuide Enterprise. Autodesk MapGuide Studio has a .NET API that enables the user to build custom applications that use Studio components. If developers want to access any aspect of the feature geometries on the server, there is an FDO API to change, manipulate, re-project, or analyze the features.



Web Server Extension API Model

The following APIs are available:

| Service | Function |
|---|---|
| Resource Service | Enables the manipulation of repositories and resources. Can also be used to manipulate and load data. |
| Feature Service | Provides access to FDO providers. |
| Mapping Service | Provides access to maps and layers within a map. |
| Drawing Service | Allows low-level access to drawing sources. Can manipulate DWF files. |
| Rendering Service | Renders a map into a bitmapped image. This image is typically used for display in the HTML viewer. |
| Coordinate System | Supports coordinate system transformations. |
| Geometry | Manipulates geometric objects. |
| Site Service | Configures users, groups, and user sessions. |

Because much of Autodesk MapGuide and Autodesk MapGuide Studio is driven by XML, including the resource definitions, settings, and layer display configurations, much of the development must be done in applications that can use the Document Object Model (DOM) for XML and that can instantiate and control the APIs.

The following three web development environments represent the three mapagents that are already developed for Autodesk MapGuide and have development examples included in the product:

- PHP 5
- JAVA (JSP)
- ASP.NET

## Developing with PHP 5

PHP is a server-side, web-based scripting language that runs on many platforms, including the following:

- Linux (for example, Red Hat, Fedora™)
- Microsoft Windows
- Unix® (for example, Solaris®)
- Mac OS® X

Since PHP is open source, organizations do not have to purchase it. Since it is constantly being revised and debugged, they can be confident that it is robust. It works on most web servers, including Apache, Microsoft Internet Information Server (IIS), Netscape®, and O'Reilly. The Autodesk MapGuide Enterprise Site Administrator is written in PHP. This administration tool is installed with the Web Extensions.

If you choose to install the PHP mapagent, you can use it to call different aspects of the Autodesk MapGuide Enterprise API using PHP.

Following is an example of accessing a map session from Autodesk MapGuide Enterprise within PHP:

```
<?php

include 'AppConstants.php';

MgInitializeWebTier ($configFilePath); //path to the webconfig.ini file

$userInfo = new MgUserInformation("Administrator", "admin");

$site = new MgSite();

$site->Open($userInfo);

$HTTP_SESSION_VARS['MgSessionId'] = $site->CreateSession();

$mapDefinition = "Library://DUBLIN CA/5. MAPS/Dublin Map.MapDefinition";

$webLayout = "Library://DUBLIN CA/6. WEB LAYOUTS/Dublin Web
Layout.WebLayout";

?>

<frameset rows="110,*" frameborder="NO" border="0" framespacing="0">

<frame src="Title.php?AppName=Sample Application" name="TitleFrame"
scrolling="NO" noresize>
```

```
<frame src="/mapviewerphp/dwfviewer.php?SESSION=<?=
$HTTP_SESSION_VARS['MgSessionId'] ?>&WEBLAYOUT=<?= $webLayout
?>"
name="ViewerFrame">
</frameset>
```

**Note:** The code example uses the library on the server at *Library://DUBLIN CA/6. WEB LAYOUTS/Dublin Web Layout.WebLayout* is used. This is the web layout created in Autodesk MapGuide Studio.

Example of Instantiating Map with PHP:

```
$map = new MgMap();
$map->Open($resourceService, 'Dublin Map');
```

Since the API for Autodesk MapGuide Enterprise is robust, you can use PHP to upload data or link to FDO, create layers with the new information, and then create a map and add that map to the web layouts. The entire process viewed in Autodesk MapGuide Studio could be automated with the API.

## Developing with Java

The mapagent installed with the MapGuide Web Server Extensions is a Java Server Page (JSP) called *mapagent.jsp*. Developed by Sun Microsystems, JSP is a web server scripting agent that communicates requests to the server. JSP is compiled on the server side as a servlet, as opposed to an applet (a java application that runs on the client). JSP can communicate with existing servlets and Java servlets that are on the server. To run JSP pages, you need a web server that is capable of running Java. The most common is the Apache Tomcat Server. Tomcat is another open source application that can be downloaded for free from http://tomcat.apache.org and installed on either Linux or Windows operating system. If you are using Windows or you are running Tomcat in parallel with another web server, you might choose an alternative port when installing it. For example, you may install Autodesk MapGuide Enterprise Web Extensions and use the Java mapagent. It might be found at http://hogwarts:8080/JavaMapAgent/MapAgent.jsp**.**

Example of Instantiating Map with JSP:

```
<%@ page import="org.osgeo.mapguide.*" %>
MgMap map = new MgMap();
map.Open(resourceService, "Dublin Map");
```

All the API calls in Autodesk MapGuide are also available with the Java Web Server Extension.

## Developing with ASP .NET

Active Server Pages (ASP) .NET is based on the Microsoft .NET Framework. This programming framework is installed on a Microsoft Windows operating system. Unlike PHP or Java, .NET works only with Microsoft IIS. Using the Microsoft Web Services that .NET provides, you can create many server-side custom applications in a Microsoft environment. ASP.NET can be programmed in many scripting languages:

- C# (C Sharp)
- Visual Basic (VB)
- JScript

ASP.NET developers usually choose between C# and VB.NET. Using development tools such as Microsoft® Visual Studio® .NET, or Microsoft® Visual Basic® .NET, you can easily put together complex applications using the WYSIWYG (what you see is what you get) features of an Integrated Development Environment (IDE).

The MapGuide Web Server Extensions come with an ASP.NET mapagent as well. The sample files that are bundled with the Web Server Extensions are developed in C# .NET, but any of the .NET scripting languages that are compatible with ASP.NET can be used to develop applications.

Example of Instantiating Map with ASP.NET:

```
<%@ Import Namespace="OSGEO.MapGuide" %>

MgMap map = new MgMap();

map.Open(resourceService, "Dublin Map");
```

## Code Examples

**Note:** The development guide for Autodesk MapGuide Enterprise is a good resource for code samples. Development methodology is available with the product upon installation.

Many applications require custom interfaces and need to extend capabilities beyond those available in Studio. In Autodesk MapGuide Enterprise these are created in the Web Tier of APIs (ASP.NET, PHP, or JSP). The APIs contain and extend all the functions and methods of the MapGuide API. In fact, most seem instantly familiar. The major differences between using the MapGuide API and the Spatial Application Server API are as follows:

- The Spatial Application Server API is organized into server-side Services (see “Description of Web Tier API,” earlier in this document). Whenever you do anything to your map, you must inform the Spatial Application Server web tier in order to save the state of your application. You can accomplish this by creating a ResourceService object that stores state.
- The Viewer must be informed of changes (because the changes actually take place on the server, not in the client). The most common way to do this is with a DHTML OnLoad() event that refreshes the page (and thus an embedded viewer) when an operation takes place.
- In Autodesk MapGuide, the API references an MWF Object embedded in a browser client. In Spatial Application Server the API references Site resources on a server. Thus the methods to access the API class libraries are very different.

The following examples concentrate on the Site Service and Mapping Service. Within these, examples are given for the following API classes: MgSite, MgSiteConnection, MgWebActions, MgLayer, and MgMap.

- Embedding the viewer into a web page
- Creating user and session information: the Site Service API

- Accessing map information using the MgMap library of classes
- Enumerating, refreshing, and changing the visibility of layers
- Digitizing and redlining
- Invoking existing Autodesk MapGuide 6.5–based reports

Additional code samples are provided with the software.

## Embedding the Viewer into a Web Page

There are several ways to embed the viewer in a web page, but the most common method is to call *dwfviewer.aspx*, installed in the Web Tier's virtual directory, and pass it the name of a Web Layout. This approach enables users to customize the display and add custom commands. It also supports dynamic interaction by the user through the client. As the user changes the display, the DWF Viewer requests updated metadata from the Spatial Application Server and requests updated graphics to display.

The following code calls two ASP.NET pages: one containing a title graphic, the second a pointer to the Viewer Definition *.aspx* page. This page is passed a parameter declaring which web layout to reference. For reference, the old way of performing this function is provided as well

| | |
|---|---|
| ASP.NET (Server) | **<frameset rows="120, *">**<br><br>**<frame src="Title.aspx?AppName=Sample Application">**<br><br>**<frame src="http://localhost/mapguide/mapviewernet/dwfviewer.aspx?WEBLAYOUT=Library://**<br><br>**<YOUR DIRECTORY>/Layouts/<YOUR LAYOUT>.WebLayout" name="ViewerFrame">**<br><br>**</frameset>** |
| MapGuide 6.5 (Client) | **<OBJECT ID="map" WIDTH="100%" HEIGHT="100%"**<br><br>**CLASSID="CLSID:62789780-B744-11D0-986B-00609731A21D">**<br><br>**<PARAM NAME="URL" VALUE="http://localhost/example.mwf">**<br><br>**</OBJECT** |

## Creating User and Session Information: The Site Service API

Using the preceding simple code, developers can view the Spatial Application Server Web Layout and interact with the maps through the default tools. However, to interact with the maps through the API you need to set up user information and session variables. Because Spatial Application Server uses a Web Tier API on the server, any application needs to store session data reflecting the state of the application to pass between the server and the viewer. This is done using a Spatial Application Server session and session variables, accessible through the Site Service API

The following lines of code illustrate the establishment of a Server Site and create a Session ID for later reference.

| | |
|---|---|
| ASP.NET (Server) | **<%@ Page language=”c#” %>**<br>**<%@ Import Namespace=”OSGEO.MapGuide” %>**<br>**<%**<br>**String mapDefinition=”Library://<Path to your map resource>/<your map>.MapDefinition”;**<br>**String webLayout=” Library://<Path to your Layout resource>/<your Web Layout>.WebLayout”;**<br>**String SessionId=””;**<br>**MapGuideApi.MgInitializeWebTier(configPath);**<br>**//configPath is the path to the webconfig.ini file**<br>**MgUserInformation userInfo = new MgUserInformation(“UserName”,“Password”);**<br>**MgSite = new MgSite();**<br>**Site.Open(userInfo);**<br>**sessionId = site.CreateSession()**<br>**%>** |
| MapGuide 6.5 (Client) | No similar methods. |

Some explanation is required here:

- The symbols <% .. %> denote the start and end of blocks of code.
- The first two lines of the code establish the language to be used (in this case, C#) and the libraries of classes to be accessed using the Import method. In this case all the API libraries and associated classes and methods are imported into the application (Library ‘Mg’).
- Next, you create three variables to hold information about the location of the Map and Web Layout resources to be used as well as a Site identification variable.
- Next you initialize the Web Tier.
- Then, you establish the user and create and open a Site using the given user credentials.
- Finally, you create a unique session identifier for the user. The server can now keep track of the session using this variable.

The code to open a viewer and Web Layout earlier stated that although you can interact with the map using the default tools, you could not interact with it through the API. Now, using the Site and session information established here, you can include the viewer in any application simply by adding the session ID variable:

```
<frame src="http://localhost/mapguide/mapviewernet/ajaxiewer.aspx?SESSION=<%=
SessionId %>&WEBLAYOUT=<%= webLayout %>”
```

## Accessing Map Information Using the MgMap Library of Classes

The following example shows how to access the Autodesk MapGuide Enterprise web tier API to access a map and obtain simple information about the map, getting and displaying the name of the map and its extents. The code assumes that the Session ID variable is passed to the script from the calling function or script.

| | |
|---|---|
| ASP.NET (Server) | **<%@ Page language=”c#” %>**<br>**<%@ Import Namespace=”OSGEO.MapGuide” %>**<br>**<%**<br>**String MgSessionId = Request.QueryString[“SESSION”];**<br>**MapGuideApi.MgInitializeWebTier(configPath)**<br>**//configPath is the path to the webconfig.ini file**<br>**MgUserInformation userInfo = new awUserInformation(MgSessionId);**<br>**MgSiteConnection siteConnection = new MgSiteConnection();**<br>**siteConnection.open(userInfo);**<br>**MgResourceService resourceService = siteConnection.CreateService(MgServiceType.ResourceService);**<br>**MgMap map = new MgMap();**<br>**map.Open(resourceService, “<Map Name>”);**<br>**MgEnvelope envelope = map.GetMapExtent();**<br>**Double lowerX = envelope.GetLowerLeftCoordinates().GetX();**<br>**Double lowerY = envelope.GetLowerLeftCoordinates().GetY();**<br>**String mapName = map.GetName();**<br>**%>** |
| MapGuide 6.5 (Client) | <SCRIPT LANGUAGE=”JAVASCRIPT”><br>function getMap()<br>{<br>    if (navigator.appName == "Netscape")<br>        return parent.mapframe.document.map;<br>    else<br>        return parent.mapframe.map;<br>} |

| | |
|---|---|
| | function getCoordinates(){ |
| | var map = getMap(); |
| | var ext = map.getMapExtent(false,true); |
| | var lowerX = ext.getMinX(); |
| | var lowerY = ext.getMinY(); |
| | var mapName = map.getName(); |
| | } |
| | </SCRIPT> |

## Enumerating, Refreshing, and Changing the Visibility of Layers

To get all the layer names in the map, access the MgLayerCollection class and then the MgLayer class for each layer.

| | |
|---|---|
| ASP.NET (Server) | <% |
| | MgLayerCollection LayerCol = map.GetLayers(); |
| | MgLayer layer = null; |
| | String layername = “”; |
| | for (int i = 0; i < LayerCol.GetCount(); i++) |
| | { |
| | layer = LayerCol.GetItem(i); |
| | layername = layer.GetName(); |
| | } |
| | %> |
| MapGuide 6.5 (Client) | <SCRIPT LANGUAGE=”JAVASCRIPT”> |
| | function getMap() |
| | { |
| |   if (navigator.appName == "Netscape") |
| |       return parent.mapframe.document.map; |
| |   else |
| |       return parent.mapframe.map; |
| | } |
| | function getLayerNames(){ |
| | var map = getMap(); |
| | var layers = map.getMapLayersEx(); |
| |   for (var i = 0; i < layers.size(); i++) |

```
    {
      var layer = layers.item(i);
      var layername = layer.getName() ;
    }


}

</SCRIPT>
```

Continuing the example, you can now toggle the visibility of a layer on or off.

ASP.NET (Server)

```
<%

MgLayerCollection LayerCol = map.GetLayers();

MgLayer layer = LayerCol.GetItem("Layer Name");

      if (layer.IsVisible()){

            layer.SetVisible(false);

      }

    else{

            layer.SetVisible(true);

    }


%>
```

MapGuide 6.5 (Client)

```
<SCRIPT LANGUAGE="JAVASCRIPT">

function getMap()

{

   if (navigator.appName == "Netscape")

       return parent.mapframe.document.map;

   else

       return parent.mapframe.map;

}

function layerToggle(name)
{
   var map = getMap();
   var layer = map.getMapLayer(name);
   layer.setVisibility(!layer.getVisibility());
   map.refresh();
}
</SCRIPT>
```

## Digitizing

Autodesk MapGuide Enterprise viewers have wrapper APIs available for user interaction with the application. Digitizing and redlining are functions that typically involve user input and can make use of the APIs provided for this purpose.

The Autodesk MapGuide Enterprise Viewer API has several functions for digitizing user input. In this example, if the user clicks the button to digitize a point

```
<input type="button" value=" Point " onclick="DigitizePoint();">
```

the script calls the JavaScript function:

```
function DigitizePoint() {

parent.parent.mapFrame.DigitizePoint(OnPointDigitized);

}
```

which in turn calls the DigitzePoint() method of the Viewer API in the map frame. It also passes the name of a callback function, OnPointDigitized, which is defined in the current script. DigizitzePoint() calls this function after it has digitized the point and passes it the digitized coordinates of the point. You can use this callback function to process the digitized coordinates as you wish. In this example, the script simply displays them in the task pane.

```
function OnPointDigitized(point) {

ShowResults("X: " + point.X + ", Y: " + point.Y);

}
```

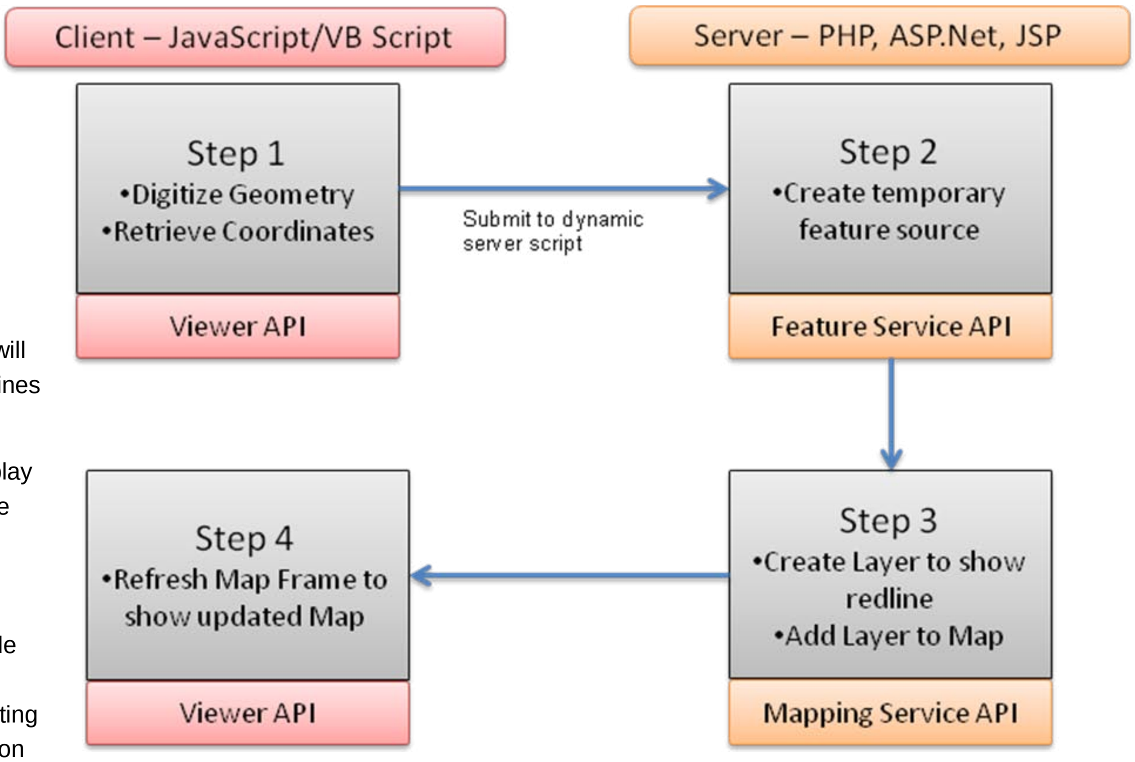
| | |
|---|---|
| MapGuide Enteprise Viewer API (Client) | **<Script Language=”Javascript”><br><br>function digitizeMyLine(myLineDigitizeHandler)<br><br>{<br><br>  parent.MapFrame.digitizeLineString();<br><br>}<br><br><br>Function myLineDigitizeHandler(oLinestring)<br><br>{<br><br> // LineString Object<br><br> Var sPoints;<br><br>  for (i=0;i<oLineString.count();i++)<br><br>     {<br><br>       //generate point list string<br><br>       sPoints = oLineString.point(i).X + “,” + oLineString.point(i).Y + “,”;<br><br>     }** |

| | |
|---|---|
| | `//submit points to server side script to generate redline layer and feature source` `var params = new Array("points",sPoints);` `parent.FormFrame.submit("../CreateRedline.aspx", params, "FormFrame");` `}` `</Script>` |
| MapGuide 6.5 (Client) | `<SCRIPT LANGUAGE="JAVASCRIPT">` `function digitizeMyLine()` `{` `getMap().digitizePolyline();` `}` `</SCRIPT>` `<SCRIPT LANGUAGE="VBSCRIPT">` `Sub onDigitizePolyline(map, oPoints)` `//create Redline Layer on client Map` `End sub` `</SCRIPT>` |
| | |

## Redlining

There are three phases to redlining in Autodesk MapGuide Enterprise.

1. Pass the digitized coordinates from the client to the server.
2. Create a temporary feature source. This will be used to draw the lines on.
3. Create a layer to display that temporary feature source.


Client – JavaScript/VB Script
Server – PHP, ASP.Net, JSP
Step 1
•Digitize Geometry
•Retrieve Coordinates
Viewer API
Submit to dynamic server script
Step 2
•Create temporary feature source
Feature Service API
Step 3
•Create Layer to show redline
•Add Layer to Map
Mapping Service API
Step 4
•Refresh Map Frame to show updated Map
Viewer API

Redlining flow chart in Autodesk MapGuide Enterprise

Unlike Autodesk MapGuide 6.5, where the redlining operation consists of creating a temporary redline layer on the client with Autodesk MapGuide Enterprise, redlines are created as temporary feature source on the server and the map is updated to include the newly created redline feature source.

Sample code for redlining is included in the appendix.

## Invoking Existing Autodesk MapGuide 6.5–Based Reports

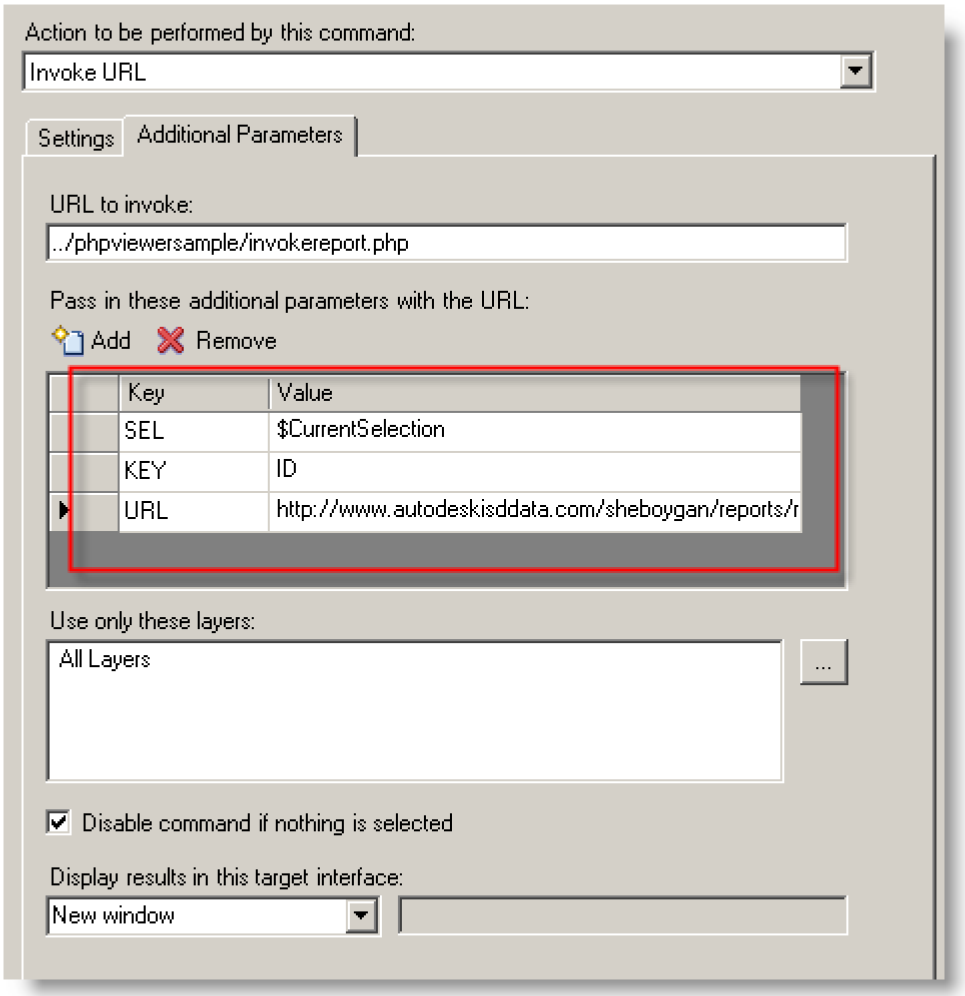
The Invoke Report script enables the reuse of existing Autodesk MapGuide 6.5 reports. The script is incorporated using an Invoke URL command (see Appendix for source code).

SEL - $CurrentSelection (current selection in MapGuide)

KEY—OBJ_KEYS or whatever name of the KEY field in the MG6.5 Report

URL—URL of the MG6.5 Report

METHOD—GET or POST


Action to be performed by this command: Invoke URL

Settings | Additional Parameters

URL to invoke: ../phpviewersample/invokereport.php

Pass in these additional parameters with the URL:

| Key | Value |
|---|---|
| SEL | $CurrentSelection |
| KEY | ID |
| URL | http://www.autodeskisddata.com/sheboygan/reports/r |

Use only these layers: All Layers

☑ Disable command if nothing is selected

Display results in this target interface: New window

Adding the Invoke Report script allowing the redirection
of report requests to a MapGuide 6.5 report

Note that you can add an optional fourth parameter (not shown) called METHOD, which can be set to either GET or POST.

This sample reuses an existing MG6.5 application report written using ASP.Net www.autodeskisddata.com/sheboygan/reports/rptParcel.aspx

## Resources

- **Autodesk MapGuide Enterprise product center**—Get the latest information on all versions of MapGuide at the official Autodesk website at **www.autodesk.com/mapguideenterprise**
- **Autodesk MapGuide discussion group**—Here you will find the MapGuide discussion group, which is an excellent resource for sharing and obtaining information from your peers. Visit **www.autodesk.com/discussiongroup-mapguide**.
- **Autodesk MapGuide developer discussion group**—The MapGuide developer discussion group is another great resources for development relation issues. Visit **www.autodesk.com/discussiongroup-mapguide-developer**.
- **MapGuide Open Source website (OSGeo.org)**—This site is the destination for information about MapGuide Open Source. Download the latest release from this site and join the mailing lists at **mapguide.osgeo.org**.
- **Autodesk Design Review product center**—Download for the Autodesk Design Review Viewer which includes the DWF viewer, which can be used with MapGuide Open Source, at **http://www.autodesk.com/designreview**.
- **Autodesk DWF developer center**—Find developer tools for the DWF Viewer, including the DWF Viewer API, which can be used with MapGuide Open Source, at **www.autodesk.com/developdwf**.

# Appendix: Invoke Report Source Code (PHP Sample)

The following PHP script redirects report requests from Autodesk MapGuide Enterprise to the existing Autodesk MapGuide 6.5 report.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>Invoke Report</title>

</head>

<body onload="OnPageLoad()">


<?php

include '../mapviewerphp/constants.php';

$configFilePath = "C:\Program
Files\MapGuideOpenSource\WebServerExtensions\www\webconfig.ini";


try

{

    // Parse the Parameters

    // - SESSION - MapGuide Session Identifier

    // - MAPNAME - Name of the Map

    // - SEL     - Current Selection

    // - URL     - URL of the MapGuide 6.5 Report

    // - METHOD  - GET or POST (default = POST)


    $session = urldecode(($_SERVER['REQUEST_METHOD'] == "POST")?
$_POST['SESSION']: $_GET['SESSION']);

  $mapName = urldecode(($_SERVER['REQUEST_METHOD'] == "POST")?
$_POST['MAPNAME']: $_GET['MAPNAME']);

    $selText = urldecode(($_SERVER['REQUEST_METHOD'] == "POST")?
$_POST['SEL']: $_GET['SEL']);
```

```
    $reportUrl = urldecode(($_SERVER['REQUEST_METHOD'] == "POST")?
$_POST['URL']: $_GET['URL']);
    $keyName = urldecode(($_SERVER['REQUEST_METHOD'] == "POST")?
$_POST['KEY']: $_GET['KEY']);

    $method = "POST";
    if ($_SERVER['REQUEST_METHOD'] == "POST")
    {
        if (array_key_exists('METHOD', $_POST))
            $method = $_POST['METHOD'];
    }
    else
    {
        if (array_key_exists('METHOD', $_GET))
            $method = $_GET['METHOD'];
    }
    $objkeys = '';
    $errorMsg = null;

  // Initialize the Web Extensions and connect to the MapGuide Site using
  // session identifier supplied by the caller.

  MgInitializeWebTier ($configFilePath);

  $userInfo = new MgUserInformation($session);
  $siteConnection = new MgSiteConnection();
  $siteConnection->Open($userInfo);

    // Create an instance of the Resource and Mapping Services and use them to open
    // the map and initialize a selection object.

  $resourceSrvc = $siteConnection->CreateService(MgServiceType::ResourceService);
  $featureSrvc = $siteConnection->CreateService(MgServiceType::FeatureService);
  $map = new MgMap();
```

```
$map->Open($resourceSrvc, $mapName);
```

```
  $sel = new MgSelection($map, $selText);
  $selLayers = $sel->GetLayers();

    // For each layer in the selection, query the selected features and append the key
    // values to the string $objkeys.

  for($li = 0; $li < $selLayers->GetCount(); $li++)
  {
        $selLayer = $selLayers->GetItem($li);
     $featureSource = new MgResourceIdentifier($selLayer->GetFeatureSourceId());

     $featureClassName = $selLayer->GetFeatureClassName();
     $filter = $sel->GenerateFilter($selLayer, $featureClassName);
     $query = new MgFeatureQueryOptions();
     $query->SetFilter($filter);

     $featureReader = $featureSrvc->SelectFeatures($featureSource,
$featureClassName, $query);
     while($featureReader->ReadNext())
         {
              if (strlen($objkeys) != 0)
              {
                   $objkeys .= ',';
              }

              $propertyType = $featureReader->GetPropertyType($keyName);
              switch($propertyType)
              {
         case MgPropertyType::Boolean :
              $objkeys .= $featureReader->GetBoolean($keyName);
               break;

              case MgPropertyType::Byte :
              $objkeys .= $featureReader->GetByte($keyName);
```

```
break;

 case MgPropertyType::Single :
$objkeys .= $featureReader->GetSingle($keyName);
break;

case MgPropertyType::Double :
$objkeys .= $featureReader->GetDouble($keyName);
break;

case MgPropertyType::Int16 :
$objkeys .= $featureReader->GetInt16($keyName);
break;

case MgPropertyType::Int32 :
$objkeys .= $featureReader->GetInt32($keyName);
break;

case MgPropertyType::Int64 :
$objkeys .= $featureReader->GetInt64($keyName);
break;

case MgPropertyType::String :
$objkeys .= "'" . $featureReader->GetString($keyName) . "'";
break;

case MgPropertyType::DateTime :
case MgPropertyType::Null :
case MgPropertyType::Blob :
case MgPropertyType::Clob :
case MgPropertyType::Feature :
case MgPropertyType::Geometry :
case MgPropertyType::Raster :
    $errorMsg = 'The data type of the key property is not supported by Invoke Report.';
            break;
            };
        }
     $featureReader->Close();
    }
}
catch (MgException $mge)
{
  $errorMsg = $mge->GetMessage() . "<br><br>" . $mge->GetDetails();
}
catch (Exception $e)
{
    $errorMsg = $e->getMessage();
}

// If no errors were detected output a form with a hidden OBJ_KEYS parameter and
JavaScript
// to submit it. Otherwise output an error message.

if ($errorMsg == null)
{
?>

<form action="<?= $reportUrl ?>" method="<?= $method ?>" enctype="application/x-
www-form-urlencoded" name="invokeReport" target="_self">
<input name="OBJ_KEYS" type="hidden" value="<?= $objkeys ?>" />
</form>
<script language="JavaScript" type="text/javascript">
function OnPageLoad()
{
    document.invokeReport.submit();
}
</script>
```

```
<?php
}
else
{
?>

<h1>Error</h1>
<p><?= $errorMsg ?></p>
<script language="JavaScript" type="text/javascript">
function OnPageLoad()
{
}
</script>

<?php
}
?>

</body>
</html>
```

# Appendix: Redlining Sample Code (PHP Sample)

This code sample is taken from the MapGuide Enterprise Developer Guide that can be found in the installation CD. Please refer to the guide for the complete sample application.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<!--
//  Copyright (C) 2004-2006  Autodesk, Inc.
//
//  This library is free software; you can redistribute it
and/or
//  modify it under the terms of version 2.1 of the GNU
Lesser
//  General Public License as published by the Free
Software Foundation.
//
//  This library is distributed in the hope that it will be
useful,
//  but WITHOUT ANY WARRANTY; without even the implied
warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the GNU
//  Lesser General Public License for more details.
//
//  You should have received a copy of the GNU Lesser
General Public
//  License along with this library; if not, write to the
Free Software
//  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston,
MA  02110-1301  USA
-->
<head>
  <meta content="text/html; charset=utf-8" http-
equiv="Content-Type">
  <title>Draw a Line</title>

  <meta http-equiv="content-style-type" content="text/css">
  <link href="../styles/globalStyles.css" rel="stylesheet"
type="text/css">
  <link href="../styles/otherStyles.css" rel="stylesheet"
type="text/css">
  <link href="/mapgudie/viewerfiles/viewer.css"
rel="stylesheet" type="text/css">

  <meta http-equiv="content-script-type"
content="text/javascript">
  <script language="javascript" type="text/javascript">
    function OnPageLoad()
    {
      parent.mapFrame.Refresh();
    }
  </script>
</head>

<body onLoad="OnPageLoad()">
  <?php
  require_once('../common.php');
  require_once($webExtensionsDirectory .
'www/mapviewerphp/layerdefinitionfactory.php');

require_once('../modifying_maps_and_layers/layer_functions.
php');

  // Get the parameters passed in from the task pane
  $x0 = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['x0']: $_GET['x0'];
  $y0 = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['y0']: $_GET['y0'];
  $x1 = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['x1']: $_GET['x1'];
  $y1 = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['y1']: $_GET['y1'];
  $sessionId = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['SESSION']: $_GET['SESSION'];
  $mapName = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['MAPNAME']: $_GET['MAPNAME'];

  try
  {
    // --------------------------------------------------//
    // Basic initialization needs to be done every time.
    MgInitializeWebTier($webconfigFilePath);

    // Get the session information passed from the viewer.
    $sessionId = ($_SERVER['REQUEST_METHOD'] ==
"POST")?$_POST['SESSION']: $_GET['SESSION'];

    // Get the user information using the session id,
    // and set up a connection to the site server.
    $userInfo = new MgUserInformation($sessionId);
    $siteConnection = new MgSiteConnection();
    $siteConnection->Open($userInfo);

    // Get an instance of the required service(s).
    $resourceService = $siteConnection->
    CreateService(MgServiceType::ResourceService);
    $featureService = $siteConnection-
>CreateService(MgServiceType::FeatureService);

    //---------------------------------------------------//
    // Open the map
    $map = new MgMap();
    $map->Open($resourceService, $mapName);

    $layerName = "Lines";
    $layerLegendLabel = "New Lines";
    $groupName = "Analysis";
    $groupLegendLabel = "Analysis";
    //---------------------------------------------------//
    // Does the temporary feature source already exist?
    // If not, create it
    $featureSourceName =
"Session:$sessionId//TemporaryLines.FeatureSource";
    $resourceIdentifier = new
MgResourceIdentifier($featureSourceName);

    $featureSourceExists =
DoesResourceExist($resourceIdentifier, $resourceService);
    if (! $featureSourceExists)
    {
      // Create a temporary feature source to draw the
lines on

      // Create a feature class definition for the new
feature
      // source
      $classDefinition = new MgClassDefinition();
      $classDefinition->SetName("Lines");
      $classDefinition->SetDescription("Lines to
display.");
      $geometryPropertyName="SHPGEOM";
      $classDefinition->
        SetDefaultGeometryPropertyName(
$geometryPropertyName);

      // Create an identify property
      $identityProperty = new
MgDataPropertyDefinition("KEY");
      $identityProperty-
>SetDataType(MgPropertyType::Int32);
      $identityProperty->SetAutoGeneration(true);
      $identityProperty->SetReadOnly(true);
      // Add the identity property to the class definition
      $classDefinition->GetIdentityProperties()->
        Add($identityProperty);
      $classDefinition->GetProperties()-
>Add($identityProperty);

      // Create a name property
      $nameProperty = new MgDataPropertyDefinition("NAME");
      $nameProperty->SetDataType(MgPropertyType::String);
      // Add the name property to the class definition
      $classDefinition->GetProperties()-
>Add($nameProperty);

      // Create a geometry property
      $geometryProperty = new

MgGeometricPropertyDefinition($geometryPropertyName);
      $geometryProperty->
        SetGeometryTypes(MgFeatureGeometricType::Surface);
      // Add the geometry property to the class definition
      $classDefinition->GetProperties()-
>Add($geometryProperty);
```

```
      // Create a feature schema
      $featureSchema = new MgFeatureSchema("SHP_Schema",
        "Line schema");
      // Add the feature schema to the class definition
      $featureSchema->GetClasses()->Add($classDefinition);

      // Create the feature source
      $wkt = $map->GetMapSRS();
      $sdfParams = new MgCreateSdfParams("ArbitraryXY",
        $wkt, $featureSchema);
      $featureService-
>CreateFeatureSource($resourceIdentifier,
        $sdfParams);
    }

    // Add the line to the feature source
    $batchPropertyCollection = new
MgBatchPropertyCollection();
    $propertyCollection = MakeLine("Line A", $x0, $y0, $x1,
$y1);
    $batchPropertyCollection->Add($propertyCollection);

    // Add the batch property collection to the feature
source
    $cmd = new MgInsertFeatures($layerName,
$batchPropertyCollection);
    $featureCommandCollection = new
MgFeatureCommandCollection();
    $featureCommandCollection->Add($cmd);

    // Execute the "add" commands
    $featureService->UpdateFeatures($resourceIdentifier,
$featureCommandCollection, false);

    //---------------------------------------------------//
    $layerExists = DoesLayerExist($layerName, $map);
    if (! $layerExists )
    {
      // Create a new layer which uses that feature source

      // Create a line rule to stylize the lines
      $ruleLegendLabel = 'Lines Rule';
      $filter = '';
      $color = 'FF0000FF';
      $factory = new LayerDefinitionFactory();
      $lineRule = $factory-
>CreateLineRule($ruleLegendLabel, $filter, $color);

      // Create a line type style
      $lineTypeStyle = $factory-
>CreateLineTypeStyle($lineRule);

      // Create a scale range
      $minScale = '0';
      $maxScale = '1000000000000';
      $lineScaleRange = $factory-
>CreateScaleRange($minScale, $maxScale, $lineTypeStyle);
```

```
      // Create the layer definiton
      $featureName = 'SHP_Schema:Lines';
      $geometry = 'SHPGEOM';
      $layerDefinition = $factory-
>CreateLayerDefinition($featureSourceName, $featureName,
$geometry, $lineScaleRange);

      //---------------------------------------------------
//
      // Add the layer to the map
      $newLayer =
add_layer_definition_to_map($layerDefinition, $layerName,
$layerLegendLabel, $sessionId, $resourceService, $map);
      // Add the layer to a layer group
      add_layer_to_group($newLayer,$groupName,
$groupLegendLabel, $map);
    }

    // --------------------------------------------------//
    // Turn on the visibility of this layer.
    // (If the layer does not already exist in the map, it
will be visible by default when it is added.
    // But if the user has already run this script, he or
she may have set the layer to be invisible.)
    $layerCollection = $map->GetLayers();
    if ($layerCollection->Contains($layerName))
    {
      $linesLayer =$layerCollection->GetItem($layerName);
      $linesLayer->SetVisible(true);
    }

    $groupCollection = $map->GetLayerGroups();
    if ($groupCollection->Contains($groupName))
    {
      $analysisGroup =$groupCollection-
>GetItem($groupName);
      $analysisGroup->SetVisible(true);
    }

    //---------------------------------------------------//
    //  Save the map back to the session repository
    $sessionIdName = "Session:$sessionId//$mapName.Map";
    $sessionResourceID = new
MgResourceIdentifier($sessionIdName);
    $sessionResourceID->Validate();
    $map->Save($resourceService, $sessionResourceID);

    //---------------------------------------------------//

  }
  catch (MgException $e)
  {
    echo "<script language=\"javascript\"
type=\"text/javascript\"> \n";
    echo "    alert(\" " . $e->GetMessage() . " \"); \n";
    echo "</script> \n";
  }
```

```
///////////////////////////////////////////////////////////
////////////////////////
  function MakeLine($name, $x0, $y0 , $x1, $y1)
  {
    $propertyCollection = new MgPropertyCollection();
    $nameProperty = new MgStringProperty("NAME", $name);
    $propertyCollection->Add($nameProperty);

    $wktReaderWriter = new MgWktReaderWriter();
    $agfReaderWriter = new MgAgfReaderWriter();

    $geometry = $wktReaderWriter->Read("LINESTRING XY ($x0
$y0, $x1 $y1)");
    $geometryByteReader = $agfReaderWriter-
>Write($geometry);
    $geometryProperty = new MgGeometryProperty("SHPGEOM",
$geometryByteReader);
    $propertyCollection->Add($geometryProperty);

    return $propertyCollection;
  }


///////////////////////////////////////////////////////////
////////////////////////
  function DoesResourceExist($resourceIdentifier,
$resourceService)
  // Returns true if the resource already exists, or false
otherwise
  {
    try
    {
      $resourceService-
>GetResourceContent($resourceIdentifier);
    }
    catch (MgResourceNotFoundException $e)
    {
      return false;
    }

    return true;
  }


///////////////////////////////////////////////////////////
////////////////////////
  function DoesLayerExist($layerName, $map)
  // Returns true if the layer already exists, or false
otherwise
  {
    $layerCollection = $map->GetLayers();
    return( $layerCollection->Contains($layerName) ? true :
false ) ;
  }
```

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