precisely

MapXtreme

Developer Guide

Version 9.4

Contact and support information is located at: http://support.precisely.com.

November 2020

Table of Contents

1

1 – Introduction to MapXtreme

Welcome Developers to Precisely's latest offering in the world of .NET programming. In support of Microsoft's .NET Framework for Windows, MapXtreme reflects a single object model for developing or extending mapping applications for the desktop, traditional client/server environments or the Web.

MapXtreme is an application development tool for organizations who recognize that data visualization and mapping can help you make better business decisions and manage assets and operations more effectively. MapXtreme is for organizations that need to incorporate location analysis or definition into desktop, client/server and web-based products. MapXtreme can be used as a powerful analysis toolkit to make critical business decisions such as optimal locations for sales offices, how to transport products most efficiently, and how to manage and protect assets. Developers can use MapXtreme to shorten their development time and improve performance, reliability, and security.

In this chapter:

Overview of MapXtreme

MapXtreme is Precisely's premier Windows software development toolkit that allows .NET-experienced developers to create powerful location-enhanced desktop and client/server applications.

From this single SDK, you can develop applications using your favorite .NET programming language, share and reuse code between desktop and web deployments, access data from a wide variety of sources using standard protocols, and more.

This is all possible through MapXtreme's object model, an API of 100 percent managed code that was developed on Microsoft's .NET Framework. The Framework's Common Language Runtime (CLR) provides the foundation that makes simplified development a reality.

The following components and features are included in MapXtreme:

- **Product framework**: The MapXtreme Object Model is built using the Microsoft .NET Framework 4.7.2. See Object Model Overview for more information.
- **Development Environment Tools**: A variety of templates, controls, sample code and tools help you develop Windows Forms and ASP.NET applications within Visual Studio. You can extend some of these components to provide more advanced functionality available through the object model. Two data management utilities are included for managing tables that you will use in your application (Geodictionary Manager) and manage workspaces for ease of use and portability (Workspace Manager). See Chapter 5 Web Applications, Controls, and Tools and Chapter 7 Desktop Applications, Controls, Dialogs, and Tools.
- **Full Mapping and Analytical capabilities**: Map creation and display, data access, thematic mapping, raster and grid handling, object processing and presentation, and more.
- **Scalable Infrastructure**: Session object pooling and caching capabilities offer big performance gains for web applications. Maintain session and user information by saving information to MapXtreme's XML-based workspace format. See Chapter 9 Working with Core MapXtreme Classes.
- **Runtime Deployment**: MapXtreme uses Windows Installer technology (merge modules) that developers can use to install or redistribute runtime components used in deployed applications. See Deploying Your Application.
- **Extensive Documentation**: Product documentation is at your fingertips as integrated components in the Visual Studio development environment. The MapXtreme *Learning Resources* page keeps you connected to all product resources, including what's new and changed in this release. It's available from the Start menu.

Key Features

MapXtreme is packed with features and conveniences to help you build your Windows Forms or ASP.NET Web applications efficiently. Regardless if you are making the map the cornerstone of your application or are adding some basic mapping functionality to support your existing application, the framework and tools you use are the same. Here is an overview of MapXtreme's capabilities:

If you are new to Precisely's mapping products, be sure to also see Chapter 3 Mapping Concepts for additional help on the basics.

For developers upgrading to MapXtreme, see the Release Notes for what's new and changed in the product. See also Migrating to MapXtreme for a mapping of features in MapX to features in MapXtreme .NET.

Feature *	Purpose
Tables, Layers, Features	Maps in MapXtreme consist of geographic features like point locations, boundaries and street networks. The feature information is stored in tables and display in the map as layers.
Data access	MapXtreme supports data from a wide variety of sources, including spatial and non-spatial RDBMS, MS Access, dBase and ASCII, as well as its own native type the MapInfo Table (.TAB). All data operations are carried out via the MapInfo.Data namespace. Operations include adding and removing tables, and inserting, updating and deleting records from a variety of data sources. .NET Dataset Provider support: allows any ADO.NET dataset provider to be treated as a table in MapInfo.Data. This will allow you to use external non-mappable data.
Web Services	MapXtreme provides clients and APIs for you to access several popular web services: geocoding, routing, WMS and WFS.
Selections and Searches	This common mapping operation allows you to find just the data that matches your criteria, by using attributes or spatial queries.

Feature *	Purpose
Thematic mapping	One of the most widely used ways of analyzing your data is to show the visual relationship and underlying data through theme maps. MapXtreme supports the creation and use of six themes: ranged, individual value, graduated symbol, dot density, and pie and bar charts.
Labeling	MapXtreme provides a sophisticated labeling capability for you to label features with names or other information (column data or expressions), and create ranged and individual value themes on the labels themselves to impart your message without relying on text alone.
Map Styling	Labeling is one form of map styling that you can control on your map in nearly infinite ways. Style also refers to the color, patterns, fonts, line styles and symbols of map features that is used in many areas of MapXtreme, including features, adornments (map titles), dialog boxes and text.
Geographic processing and analysis	This refers to making new features from existing ones, such as combining postal code boundaries to create sales territories. It also refers to using the feature's location coordinates to learn more about its relationship to other features. For example, create a buffer around a point that measures five miles in radius to find out what other points fall within the buffer zone.
Projections and Coordinate Systems	These are ways of representing locations on a two-dimensional map. Knowing the coordinate system of your data allows you to line up features properly for accurate display and measurement. MapXtreme supports a wide variety of projections and coordinate systems and provides information for creating your own.

* Some features or tools in this product may not be supported for some languages or in some regions. Please consult your local customer service representative for details.

Migrating to MapXtreme

The following is a table that compares features and functionality from MapX with that of MapXtreme .NET. As with any new architecture of a product, note that the equivalent may not be exact. Use the MapXtreme topics in the right column of this list to find further details elsewhere in this Developer Guide and in the online help and object model. A full list of the MapX object model and the equivalent functionality in MapXtreme is covered in Appendix J: Migrating to MapXtreme.

MapX5.0	MapXtreme
Map object	Map class: holds collection of Layers. MapControl: way to view a map on a form. MapInfo.Mapping namespace
MapXBroker	Session class: starting point for all MapXtreme-based applications. Related topics: MICommand, Catalog, Pooling MapInfo.Engine namespace
Layer-centric model	Tablecentric model Related topics: Table metadata (TableInfo class), Feature class, Column class (MI_Geometry, MI_Style, MI_Key), MapInfo ADO.NET data provider. MapInfo.Data namespace
Datasets, data binding	Add a temporary column to an Table using the Table.AddColumns() method. Related topic: Geodictionary Manager MapInfo.Data namespace
Geosets. Geoset Manager	Workspaces (.MWS): XML format. Geosets are supported. Related topics: Workspace Manager. MapInfo.Persistence namespace

MapX5.0	MapXtreme
Annotations	Adornments: a Legend, a Title, a Scalebar, or some other similar user-defined object in a single map. MapInfo.Mapping namespace
Thematic mapping	Same thematic map types. Themes are no longer layers. Related topics: ModifierThemes (graduated symbol, pie and bar themes), ObjectThemes (ranged, individual value, dot density themes. MapInfo.Mapping.Thematic namespace
Feature layer and Feature Collections	FeatureGeometry: all geometries are now objects. Includes point, multipoint, curve, multicurve, polygon, multipolygon ring. Geometries include rectangle, rounded rectangle, ellipse, legacy arcs, legacy text. Related topics: CoordSysFactory (registered coordinate systems), object processing (see FeatureProcessor, below) MapInfo.Geometry namespace
FeatureFactory	FeatureProcessor class: object processing Buffer, Combine, Intersection, ConvexHull. Related topic: Difference (formerly Erase) in FeatureGeometry class. MapInfo.Geometry namespace
Tools	Desktop tool SelectRegion can assign to mouse buttons and configure mouse wheel for zooming. MapInfo.Tools namespace

MapX5.0	MapXtreme
Raster and Grid Images	New table structure, RasterImageInfo, GridImageInfo. Related topics: controllable styles: brightness, contrast, color/grayscale, translucency, one-color transparency. MapInfo.Raster namespace
Label objects and Label collections	LabelLayer: allows the separate ordering of labels and layers. LabelSource: information from the data source that is used for labeling. MapInfo.Mapping namespace
Selection object	Selection class: a multi-feature collection of lists of features that are tied to a table. MapInfo.Engine namespace
Spatial Server connectivity	MI ADO.NET data providers, MapInfo SQL language. MapInfo.Data namespace
Style Object	Style class: new object model. Styles are now objects, not properties of other objects; information is stored in column MI_Style. Related topics: FeatureStyleModifiers, FeatureOverrideStyleModifiers in the MapInfo.Mapping namespace. MapInfo.Styles namespace

Learning to Use MapXtreme

MapXtreme has a lot to offer beginning and experienced .NET developers alike. This section describes several support mechanisms we have created to get you up and running quickly and with minimal disruption in your development timetable.

Support Resources

MapInfo is committed to supporting new MapInfo developers as well as our long time customers. We provide a wide variety of tools to help you make the transition or get up and running quickly in the Visual Studio environment with the following resources.

MapXtreme Learning Resources Page

Accessible from the Start Menu after you install MapXtreme, the Learning Resources Page brings together a wide variety of information about MapXtreme, including best development practices, code samples, tutorial web applications, links to all documentation and online resources on the MapInfo website, and much more. Refer to this page often to get comfortable with MapXtreme and as you continue to develop mapping applications that match your business needs.

Documents and Help Systems

This MapXtreme Developer Guide provides an overview of the MapXtreme development environment and namespaces. The Visual Studio-integrated Help System provides the more specific API-level information you need to use these tools to develop integrated Windows desktop and web-based applications with the MapInfo powerful mapping components. If you have comments regarding the documentation, you can share them at to http://support.precisely.com

---

ⓘ The above-mentioned email address should not be used for questions specific to the software or clarification about subjects contained in the documentation. Please send those questions to Technical Support (see below).

If you are new to using or developing with MapInfo mapping products, be sure to see Chapter 3 Mapping Concepts.

To get the latest release information, be sure to download a copy of the MapXtreme Release Notes from the Precisely website.

Technical Support

Precisely offers unparalleled technical support for users of MapInfo software products. Our Technical Support department provides technical assistance to registered users of MapInfo software – so you don't need to be an expert in all aspects of our products in order to get results. See the Precisely Web site at http://support.precisely.com for information on the tech support offerings.

2 – Getting Started

This chapter provides all the information you need to install, configure, and deploy your first MapXtreme application.

In this chapter:

Installation Requirements

Precisely has tested and supports MapXtreme on the following.

Architecture	• 64-bit • 32-bit
Operating Systems	• Windows 10 (x86, x64) • Windows 8 and 8.1 (x86, x64) • Windows 7 (x86, x64) • Windows Server 2016 • Windows Server 2012 R2 (x64) • Windows Server 2012 (x64) • Windows Server 2008 R2 (x64) • Windows Server 2008 with SP2 (x86, x64)
Development Framework and IDE Support *	• Microsoft .NET Framework 4.7.2 • Visual Studio 2017 • Visual Studio 2015
Browsers	• Internet Explorer 10 and higher** • Firefox 3.5 and higher • Chrome 20 and higher
For web application and deployment:	• IIS 10 (Windows 10, Windows Server 2012 R2) • IIS 8 and higher (Windows 8 and 8.1) • IIS 7 (Windows Server 2008 R2 and Windows 7)

Supported databases	• Microsoft Access 2007 and Excel 2007 • Microsoft Access 2003 • Oracle 12C R2 • Oracle 11G (11.1.0.6.0 and 11.1.0.7.0) • Oracle 10G, 10GR2 • Microsoft SQL Server 2014 • Microsoft SQL Server 2012 (with SQL Native Client 11) • Microsoft SQL Server 2008 (with SQL Native Client 10)
For data access:	• MDAC 2.8

* Recommended development environments (IDE). Others can be used, however, the MapXtreme installer will not integrate its templates, samples, and help system.

** The MapXtreme Learning Resources displays in Internet Explorer automatically, regardless of your default browser setting. This will not change your default browser setting.

ⓘ MapXtreme does not support the Express Editions of Microsoft Visual Studio.

Minimum System Requirements

Memory	Windows 10: 1 gigabyte (GB) RAM (32-bit), 2 GB RAM (64-bit) Windows 7 and 8: 1 gigabyte (GB) RAM (32-bit), 2 GB RAM (64-bit) Windows Server 2016: 2 GB RAM Windows Server 2012 R2: 1 GB RAM Windows Server 2008: 512 megabytes (MB) RAM

Processor	Windows 10: 1 gigahertz (GHz) processor Windows 8, Windows 7: 1 GHz processor Windows Server 2016: 1.4 GHz processor Windows Server 2008/2012: 1 GHz processor
Video Card	Graphics card that supports at least 256 colors

Types of Installations

MapXtreme provides two installation types: one for Development (SDK) and one for Deployment (Runtime). Each is selectable from the product CD Browser.

Development (SDK) Installations

The Development Installation installs the MapXtreme Software Development Kit (SDK) on your computer. Choose this installation to develop your desktop and web applications. Upon installation, this SDK is automatically integrated with Microsoft Visual Studio and works in conjunction with the .NET Framework. The SDK provides C# and VB application templates for simplified development.

For instructions on how to install the SDK, see Installing MapXtreme in Your Environment. Instructions are also available from the Help buttons on the installation dialog boxes.

Deployment (Runtime) Installations

The Deployment installation option installs the Location Runtime Environment which lays down the MXTRuntime.exe (or MXTRuntime.exe for software-copy protected versions of MapXtreme). For instructions on installing the Runtime installer see Deploying With the Runtime Installer.

Side-By-Side Installations and Use

You may have more than one version of MapXtreme installed on your system at the same time. Each version of MapXtreme installs into its own directory.

You may also build a desktop and web application against an earlier version of MapXtreme and run it against a later version.

You may run more than one ASP.NET application on the same computer if they are built with different versions of MapXtreme. Create an application pool for each version of MapXtreme and place the appropriate ASP.NET application in it. Restart IIS by issuing an *iisreset* from a command prompt or recycle the application pool that the application is assigned. When an application runs in its own process space, it will load the appropriate version of MapXtreme.

This does not affect desktop applications created with different versions on MapXtreme. Each desktop application always runs in its own process space.

Before You Install

The following are things to be aware of prior to installing MapXtreme.

Administrator Privileges

To install MapXtreme, you must be an Administrator on the machine or the current user must be a member of the group Administrator. Right-click Setup.exe and choose 'Run as administrator'. This applies to both types of installation (SDK and Runtime).

Install .NET Framework and Visual Studio First

Before you install MapXtreme, be sure that you have the .NET Framework and the Visual Studio environment appropriate for the framework installed.

ⓘ You may use a different development environment than Visual Studio, however, the templates, samples, and the online help system will not be integrated.

IIS 7/8.5/10 Support

MapXtreme supports web deployment under Internet Information Services (IIS) IIS 7, IIS 8.5 and IIS 10.

For IIS 7, the operating system requirement is Windows Server 2008 or Windows 7. For IIS 8, the operating system requirement is Windows Server 2012 or Windows 8/8.1. MapXtreme does not support web deployment on Windows XP.

Throughout the Developer Guide references to IIS will refer to IIS 7, IIS 8.5 and IIS 10.

IIS 7 is included (although not necessarily installed) with Windows Server 2008 and Windows 7 Ultimate. MapXtreme supports IIS 7 and IIS 8 in both classic mode and integrated pipeline mode.

Prior to installing MapXtreme, configure IIS 7/8.5/10 following the steps below. These steps apply to Windows Server 2008 and Windows 7.

1. Enable Windows Authentication and Anonymous Authentication.
   a. Go to Control Panel > Administrator Tools, right-click **IIS** and choose to “Run As Administrator”.
   b. Select Default Web Site.
   c. Under the IIS group, double-click Authentication.
   d. Right-click Anonymous Authentication and choose Enable. Do the same for Windows Authentication.
2. Enable the Web Management Tools.
   a. Go to Control Panel > Programs and Features.
   b. Click Turn Windows features on or off. The Windows Features dialog box opens.
   c. Select the Internet Information Services checkbox.
   d. Double-click (or expand) Web Management Tools, and select all checkboxes below it.
3. Enable World Wide Web Services.
   a. In the Windows Features dialog, double-click (or expand) the World Wide Web Services and check the boxes itemized below.
   b. Application Development Features - select all.
   c. Common HTTP Features: Default Document, Directory Browsing, HTTP Errors, Static Content and WebDAV Publishing
   d. Health and Diagnostics: HTTP Logging, Request Monitoring
   e. Performance: Static Content Compression
   f. Security: Request Filtering and Windows Authentication

MapXtreme Web Controls and IIS

MapXtreme's Web Controls have always modified the web.config file of your ASP.NET automatically to include the required modules and handlers. We fully support IIS7 integrated pipeline mode, and will also auto-modify the web.config file to include the necessary code under the system.webServer node. To maintain compatibility with previous ASP.NET applications built to run in IIS6, or IIS7's 'Classic' pipeline mode, the Integrated Pipeline code will only be entered into the web.config file if the MapInfo.Engine.Session.PipelineMode property is set to 'Integrated'. This property is added to the 'appSettings' node of the web.config file when any ASP.NET project is

loaded into Visual Studio with MapXtreme installed on your system. Initially, this property is commented out. Simply uncomment to make the proper edits for an application running in Integrated Pipeline mode in IIS7. Change the value to 'classic', or simply recomment the property, to comment out the system.webServer node for backwards compatibility with Classic Pipeline mode.

Default Install Directories for MapXtreme

MapXtreme is a 64-bit application that installs by default into C:\Program Files on 64-bit computers or in C:\Program Files (x86) on 32-bit computers.

Both default paths are included in the Web.config files of the MapXtreme sample applications. If you have installed MapXtreme to another location, you must edit the Web.config files to point to that location for the samples to run properly.

ⓘ For non-English US (ENU) installations of Windows, the default installation directory C:\Programmer\ is considered a custom install location by MapXtreme. You must edit the samples Web.config files to point to your install directory, as the example below shows.

```
<configuration>
<appSettings>
<add key="MapInfo.Engine.Session.Workspace"
   value="C:\Programmer\MapInfo\MapXtreme\9.x.x\Samples\Data\World.mws" />
</appSettings>
</configuration>
```

Additional Installation Features

MapXtreme provides online installation instructions to follow. You can also access the instructions via the Help button on the install dialogs during installation.

MapXtreme provides free sample data for a variety of world locations. To install the data, choose Install Sample Data from the CD Browser. You can control how much of the data you wish to install by choosing the Custom option. The Complete option (default) will install about 450 MB of world data sets under Program Files\MapInfo\MapXtreme\9.x.x\Samples\Data.

ⓘ You do not need to run this data installer in order to use the sample applications that ship with MapXtreme. Basic sample data is automatically installed to the \Data folder for this purpose.

The MapXtreme DVD Browser also provides a link to the PDF version of this Developer Guide.

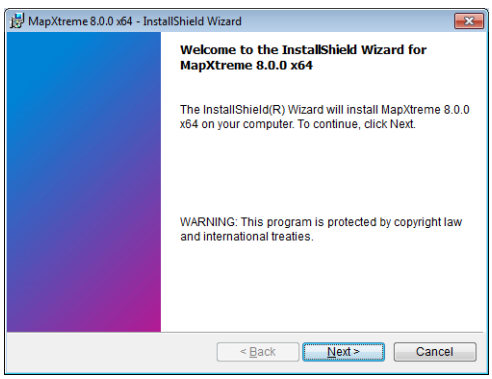
Installing MapXtreme in Your Environment

To install MapXtreme:

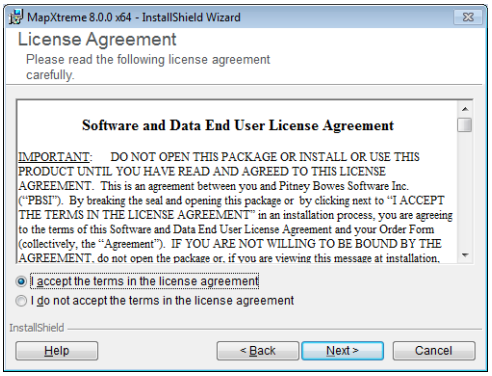
1. Place the MapXtreme product media in the disk drive.
2. At the DVD Browser main page, click **Install**. The Install Description page appears.
3. Choose either Development Install to install the SDK or Deployment Install to install the runtime version.
4. Choose Install SDK or Install Runtime. At the Welcome dialog box, click **Next** to proceed. For deployment installations skip to step 10.

> ⓘ You may also review the installation instructions and install sample data from this page.

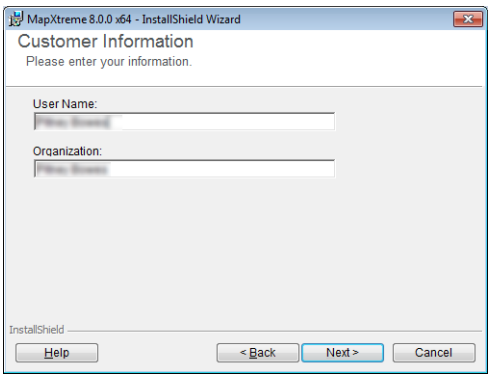
5. At the Installer Welcome dialog box, read the information in the panel and click **Next** to proceed.



6. Choose to accept the License Agreement. Click **Next**. The Customer Information dialog box appears.

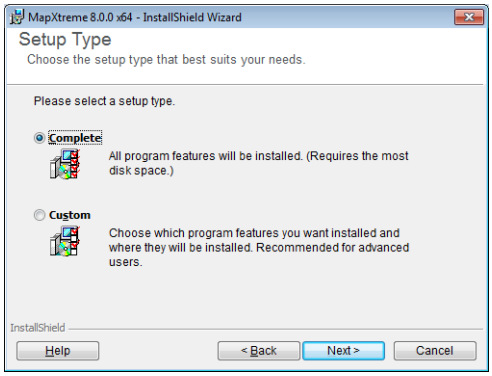


7. At the Customer Information dialog box, enter your user name and company name in the appropriate fields. The Setup Type dialog box appears.

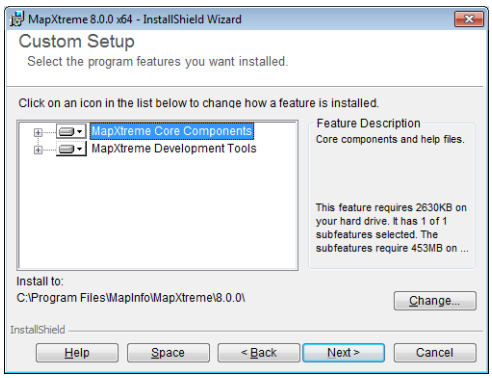


8. At the Setup Type dialog box, select Complete or Custom. Choose Custom if you want the features to be installed or to install to a location other than the default (C:\Program Files\MapInfo\MapXtreme\9.x.x). Click **Next**. If you chose Complete proceed to step 10. If you chose Custom continue to step 9.

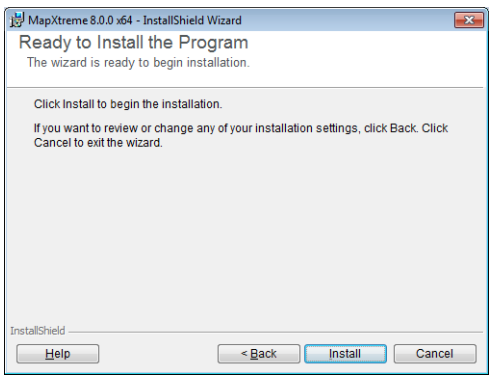
ⓘ If you install MapXtreme to a location other than the default or on a version of Windows XP other than the English US (ENU), you will need to edit the web.config file of any sample web application you intend to run after installation. See the sample web application's ReadMe.rtf file for instructions on editing the web.config file.



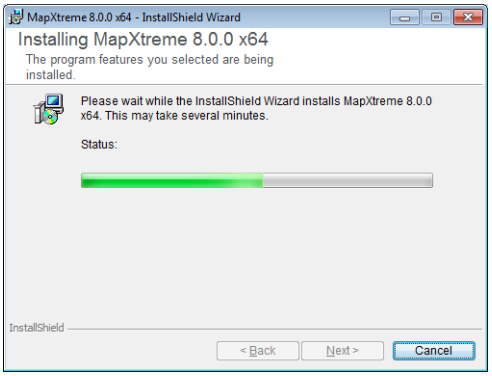
9. If you chose Custom in the previous step, select the components that you want to install, or click the **Change** button to specify a new installation path. Click **Next**.



10. At the Ready to Install the Program dialog box, click **Install**.

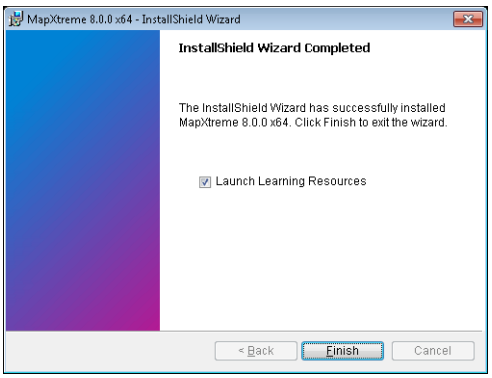


11. The Installing MapXtreme dialog box launches.



12. At the InstallShield Wizard Completed dialog box, uncheck the Launch Learning Resources checkbox if you do not wish to view the MapXtreme Learning Resources page, and then click **Finish** to leave the software installer.

The MapXtreme Learning Resources page is available anytime from the Windows Start menu from an SDK installation It is not available for a runtime installation.



Upgrading MapXtreme

MapXtreme installs into its own directory using the form \<*installdir*>\MapInfo\MapXtreme\x.x.*x*, where *x.x.x* is the current release. It will not overwrite a previous version. This allows you to maintain side-by-side installations of different releases of the product.

Note, when using a side by side install, you must close Visual Studio before opening a project of a different type. For web applications, you will also need to reset Internet Information Services (IIS). Use the **iisreset.exe** command in a console window or from the Start > Run menu option.

Migrating Web Sites to 64-bit Web Applications

MapXtreme supports creating 64-bit web applications. If you have an existing Web site, you will need to migrate them to a Web application to take advantage of 64-bit processing. You may continue to build MapXtreme-based Web sites that are 32-bit only (see Updating Existing Web Sites on page 33).

There is nothing specific to MapXtreme when migrating a Web site to a Web application. For more information, see the following Microsoft topics Web Application Projects versus Web Site Projects and Walkthrough: Converting a Web Site Project to a Web Application Project in Visual Studio.

Updating Existing Web Sites

Follow the instructions in this section if you wish to update a 32-bit MapXtreme Web Site to use the latest assemblies.

In order to run a Web site created with a previous release of MapXtreme, you must edit your application's Web.config file to point to the new versions of the assemblies.

```
<appSettings>
  <!--Use this setting to turn Session pooling on/off (true/false)-->
  <add key="MapInfo.Engine.Session.Pooled" value="true" />
  <!--Use this setting to save Session state automatically (HttpSessionState) or manually (Manual)-->
  <add key="MapInfo.Engine.Session.State" value="Manual" />
  <!--Use this setting to preload a workspace on Session creation-->
  <add key="MapInfo.Engine.Session.Workspace" value="c:\Program Files\MapInfo\MapXtreme\8.0.0\Samples\Data\World.mws;
  c:\Program Files (x86)\MapInfo\MapXtreme\8.0.0\Samples\Data\World.mws" />

</appSettings>
<system.web>
  <!--  DYNAMIC DEBUG COMPILATION
        Set compilation debug="true" to enable ASPX debugging.  Otherwise, setting this value to
        false will improve runtime performance of this application.
        Set compilation debug="true" to insert debugging symbols (.pdb information)
        into the compiled page. Because this creates a larger file that executes
        more slowly, you should set this value to true only when debugging and to
        false at all other times. For more information, refer to the documentation about
        debugging ASP .NET files.
  -->
  <compilation defaultLanguage="c#" debug="true">
    <compilers>
      <compiler language="c#" type="Microsoft.CSharp.CSharpCodeProvider, System, Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=B77A5C561934E089" extension=".cs" compilerOptions="/d:DEBUG;TRACE" />
    </compilers>
    <assemblies>
      <add assembly="MapInfo.WebControls, Version=8.0.0.151, Culture=neutral, PublicKeyToken=0A9556CC66C0AF57" />
      <add assembly="MapInfo.CoreEngine, Version=8.0.0.151, Culture=neutral, PublicKeyToken=93E298A0F6B95EB1" />
      <add assembly="MapInfo.CoreTypes, Version=8.0.0.151, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1" />
      <add assembly="MapInfo.CoreEngine.Wrapper, Version=8.0.0.151, Culture=neutral, PublicKeyToken=93E298A0F6B95EB1" />
      <add assembly="System.Design, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A" />
    </assemblies>
  </compilation>
```

MapXtreme assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 and C:\Windows\Microsoft.NET\assembly\GAC_64.

At a minimum, you must include MapInfo.CoreEngine.dll, MapInfo.CoreEngine.Wrapper.dll, MapInfo.CoreTypes.dll and MapInfo.WebControls.dll.

When you are finished editing, save your Web.config file and rebuild your web site.

Check that the Copy Local property for your web controls is set to False. See Set Copy Local property to False on page 34.

Updating Existing Desktop Applications

Desktop applications created with a previous release of MapXtreme can be recompiled to work in the current release. Controls may have to be re-added to the form.

There are several things you need to do first.

- Set Copy Local property to false
- Add new assemblies to project
- Redirect assemblies to the new assemblies

Set Copy Local property to False

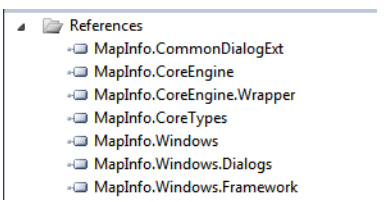
Verify that the Copy Local property for your controls is set to False. There exists a known issue that when you drag and drop a MapXtreme desktop control onto a Windows form, references are loaded that have the Copy Local property set to True. MapXtreme references must point to assemblies residing in the Global Assembly Cache (GAC), not to the local bin path, which is what happens when Copy Local is set to True.

The same behavior exists for ASP.NET Web applications when dragging and dropping MapXtreme web controls onto a form or when building a console application that is not based on a MapXtreme template. The same workaround applies here: Set the Copy Local property to FALSE.

This does not occur when using the MapXtreme web application template or sample applications, which are based on the MapXtreme web application template.

Add New Assemblies to Project

The illustration below highlights the assemblies used for MapXtreme desktop applications.



Redirecting MapXtreme Assemblies To Newer Versions

You must redirect your application to use the current version of the assemblies. Microsoft provides several mechanisms to redirect assemblies. For more detailed information, see Microsoft's .NET Framework Developer Center.

Application Configuration File

It is recommended that you use an application configuration file to accomplish assembly redirection. The configuration file must be located in the same directory as the application and is named after the application. For example, the configuration file for *myApp*.exe must be named *myApp*.exe.config.

The application configuration file overrides settings in the publisher's policy file.

To redirect assemblies, you must identify the version numbers and PublicKey tokens for the current release and add them to your application configuration file. The version number is in the form X.x.x.x, for example, 8.1.0.x.

MapXtreme assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 and C:\Windows\Microsoft.NET\assembly\GAC_64.

Publisher Policy File

A publisher policy file containing redirection settings could be installed in the GAC with the assembly. However, this is not a supported Precisely configuration.

Machine Configuration File

Specifying redirection settings in the machine configuration file will cause *all* applications referencing the assembly to use the upgraded version. Use this method of redirection carefully, since the machine configuration file overrides settings in both the application configuration file *and* the publisher's policy file.

Creating Applications in Visual Studio

With MapXtreme, it's easy to add a map to your application. Visual Basic.NET and Visual C# project templates are provided that allow you to create simple mapping applications without writing any code.

Sample desktop and web applications are also provided for you to review, experiment with and adapt to your own situation. For a step-by-step tutorial on how to use these sample applications, see Appendix A: How to Create and Deploy a MapXtreme Application.

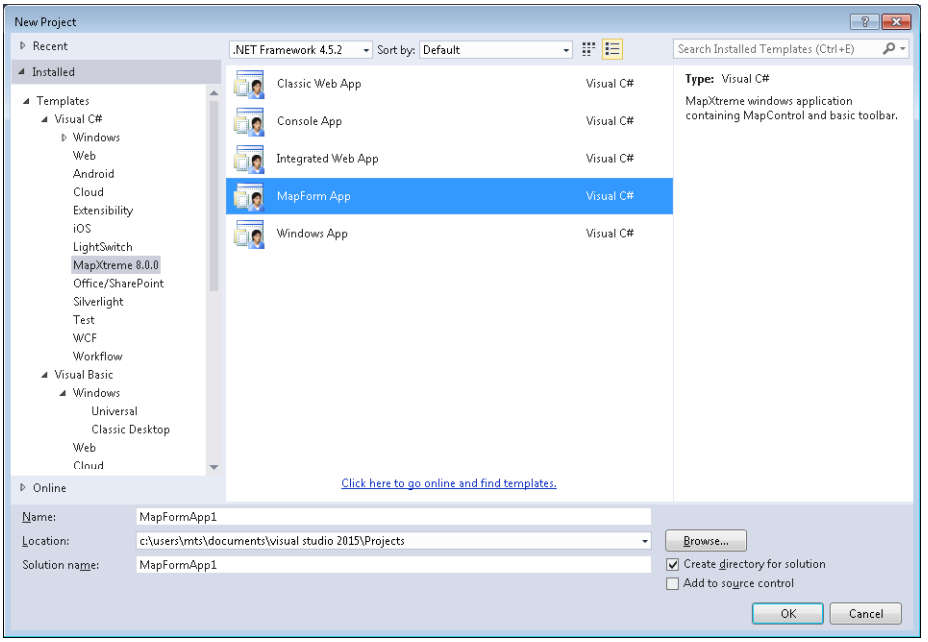
MapXtreme also ships with a collection of tutorial applications to help you understand how to include useful mapping functionality in a web application. Each tutorial application is accompanied by documentation that explains how the application was built. See Learning Resources from the Start > Program menu for the documentation. The tutorial applications are included in a single Visual Studio solution called MapXtremeTutorials.sln, located in the \Tutorials folder of your MapXtreme installation (default location is C:\Program FIles\MapInfo\MapXtreme\9.x.x.\Tutorials, where 9.x.x is the release version).

The following procedure outlines the steps to make a simple desktop mapping application. For steps to create a web application, see ASP.NET Web Applications.

Map Applications

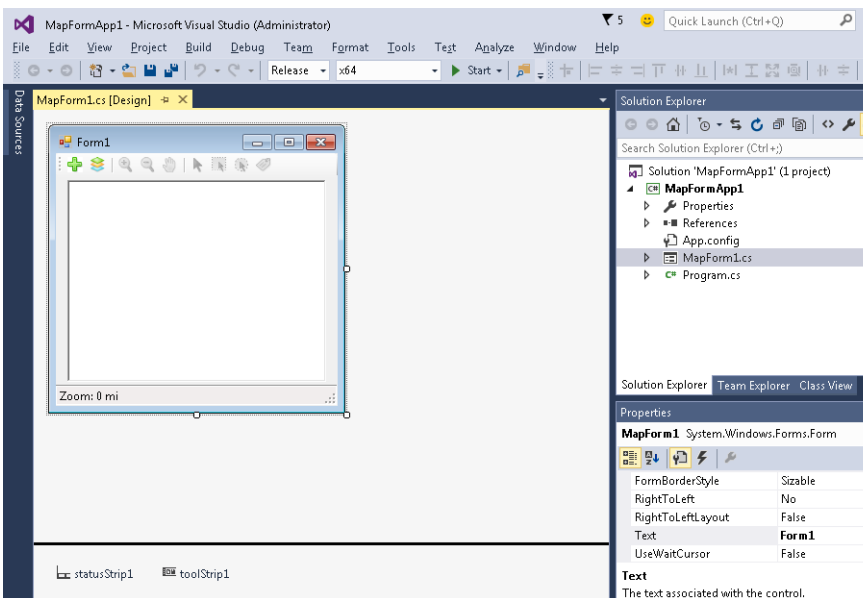
ⓘ This example is specific to Visual Basic.NET. To create a Visual C# Map application substitute Visual C# for Visual Basic in the following steps.

1. From the Visual Studio File menu, select **New Project**. The New Project dialog box appears.
2. In the Installed Templates frame of the New Project dialog box, under the Visual Basic folder, choose Windows.
3. In the Templates frame of the New Project dialog box, select **MapXtreme 9.x.x MapForm Application**.



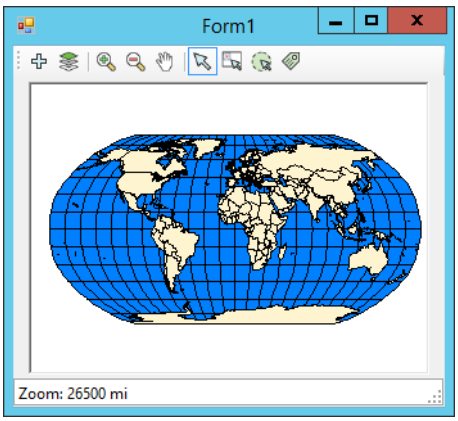
4. Choose an appropriate name and click **OK**. MapXtreme creates the application.

Under Solution Explorer double-click **MapForm.vb** and MapForm.vb [Design] appears.



5. On the Debug menu, click **Start Debugging** to run the application.

6. Click the **Open Table** icon and load your data. The default location for sample data is Program Files\MapInfo\MapXtreme\9.x.x\Samples\Data or, Program Files(x86)\MapInfo\MapXtreme\9.x.x\Samples\Data.



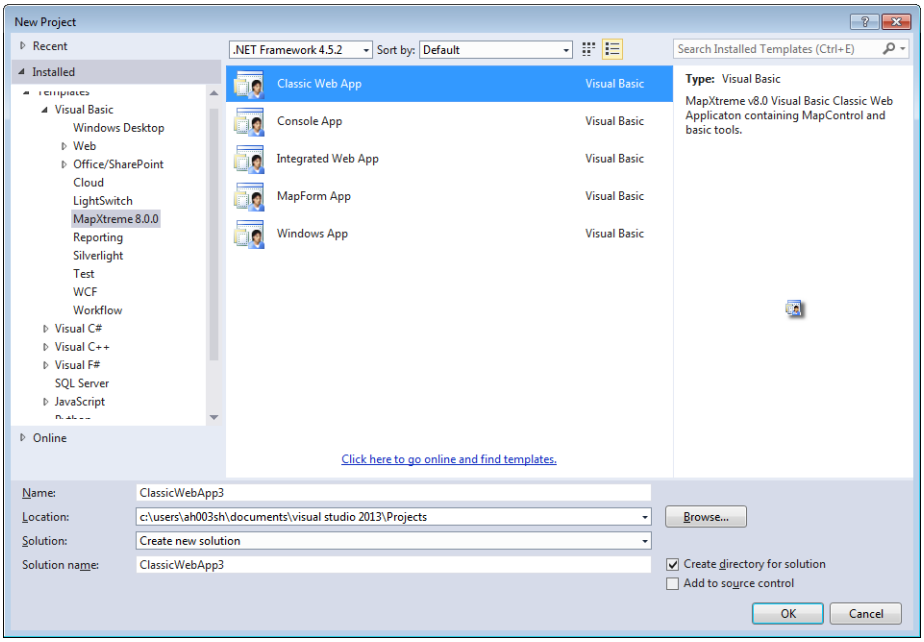
7. Use the controls in the toolbar to manipulate the map as you would with any other Precisely mapping application.

ASP.NET Web Applications

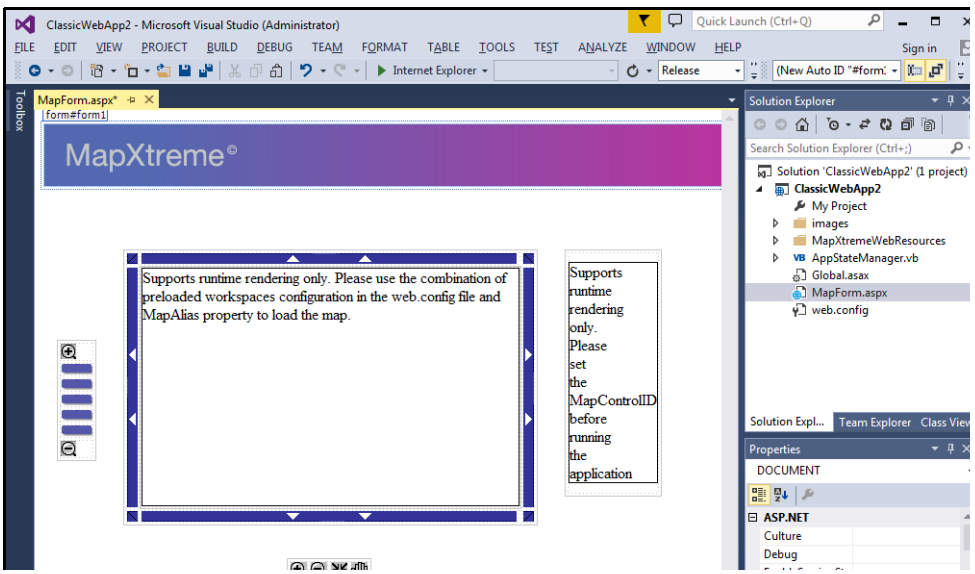
The following procedure outlines the steps to make a simple ASP.NET Web Application using the MapXtreme Web Application template. The template is pre-configured for IIS 7/8.

This example is specific to Visual Basic.NET. To create a Visual C# ASP.NET map application substitute Visual C# for Visual Basic in the following steps.

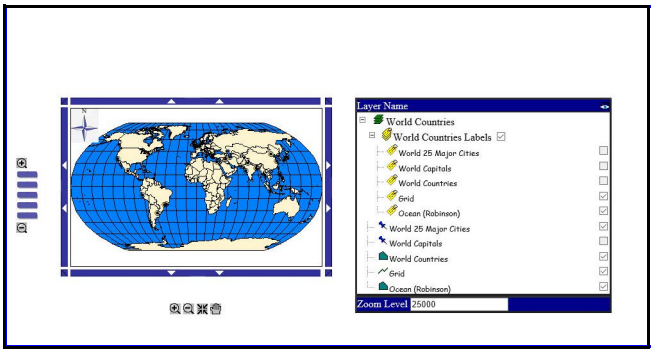
1. From the Visual Studio File menu, click **New Project.** The New Project dialog box appears.
2. From the Installed Templates list, choose Visual Basic and Windows. From the available web templates, choose MapXtreme **Classic Web** Application or MapXtreme **Integrated Web** Application.



3. Set the name of the application and solution and the location for the project. Click OK.
4. Under Solution Explorer double-click **MapForm.aspx** to see the design view of the MapControl and some tools.



5. Build the project.
6. On the Visual Studio Debug menu, click **Start Debugging** to run the application.



7. Use the controls in the toolbar to manipulate the map as you would with any other Precisely mapping application.

If you are unable to run the application, be sure to check that the ASP.NET State Service is running on your system (Control Panel > Administrative Tools > Services > ASP.NET State Service).

MapXtreme Controls

Once you have created a basic application using one of our templates, enhancements are possible using a variety of MapXtreme controls provided in the Toolbox.

For desktop map applications built using Windows Forms, MapXtreme Windows Controls are available. Any of the controls found in the MapXtreme Windows Controls tab of the Visual Studio Toolbox can be added to your form. See Chapter 7 Desktop Applications, Controls, Dialogs, and Tools.

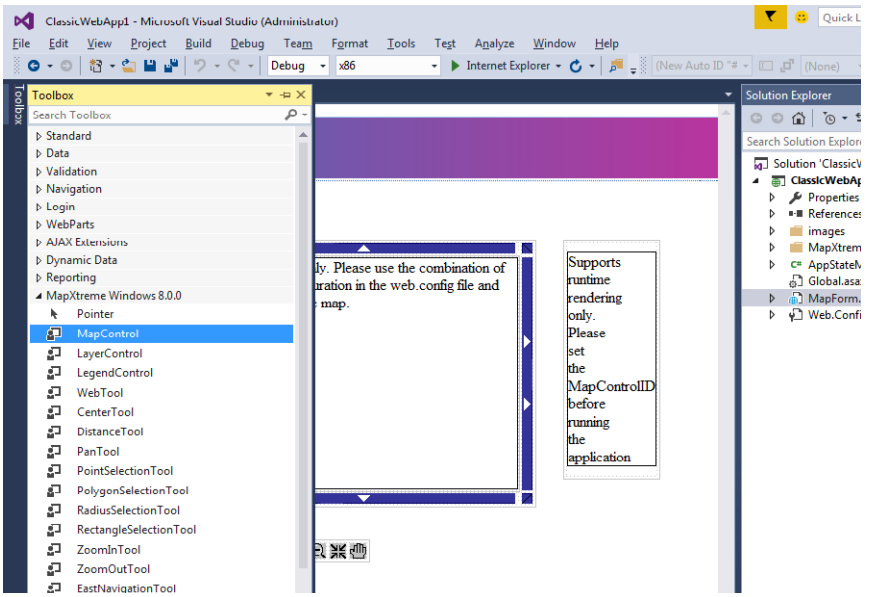
For MapXtreme ASP.NET web applications, MapXtreme Web Controls are available. Any of the controls found in the MapXtreme Web Controls tab of the Toolbox, can be added to your form. See Chapter 5 Web Applications, Controls, and Tools for more information.

Building ASP.NET Web Applications Without a Template

You may also build your ASP.NET map application without using the MapXtreme templates. For example, to create a Visual Basic web application:

1. Choose **File > New Project** from the Visual Studio menu. Under Visual Studio Installed Templates, navigate to the Web section under your preferred programming language. Choose the generic ASP.NET Web Application template and click OK.

2. From the MapXtreme Web Controls group in the Toolbox, choose a control and drag it onto the form. This will add MapXtreme assemblies as references in your project: MapInfo.WebControls. MapInfo.CoreEngine, MapInfo.CoreTypes and MapInfo.CoreEngine.Wrapper. It will also update your web.config file with assembly information.



3. In this situation, you will notice that the MapXtreme web controls and tools will display red X's in the Designer, instead of their icons. To display the icons properly, copy the MapXtremeWebResources folder from one of the MapXtreme sample applications and paste it into your project where your Web.config and default.aspx files are located. Close and re-open the web page to see the icons.

4. To run the web application under IIS 7 or IIS 8 classic mode, copy the following code into the Web.config file. To run in integrated pipeline mode, skip to step 6.

```
<system.web>
    <compilation debug="true" targetFramework="4.7.2">
      <assemblies>
        <add assembly="System.Design, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A" />
        <add assembly="MapInfo.CoreEngine, Version=9.x.x.x, Culture=neutral,
PublicKeyToken=93e298a0f6b95eb1" />
        <add assembly="MapInfo.CoreEngine.Wrapper, Version=9.x.x.x,
Culture=neutral, PublicKeyToken=93e298a0f6b95eb1" />
```

```
        <add assembly="MapInfo.CoreTypes, Version=9.x.x.x, Culture=neutral,
PublicKeyToken=93e298a0f6b95eb1" />
        <add assembly="MapInfo.WebControls, Version=9.x.x.x, Culture=neutral,
PublicKeyToken=0a9556cc66c0af57" />
      </assemblies>
    </compilation>
    <sessionState mode="StateServer"
stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data
source=127.0.0.1;userid=sa;password=" cookieless="false" timeout="20" />
    <httpHandlers>
      <add verb="*" path="MapController.ashx"
type="MapInfo.WebControls.MapController, MapInfo.WebControls,
Version=9.x.x.x, Culture=neutral, PublicKeyToken=0a9556cc66c0af57" />
    </httpHandlers>
    <httpModules>
      <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.x.x.x, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator" />
    </httpModules>
  </system.web>
```

5. Save Web.config. Build the project and run the application.

6. To run the web application under IIS 7 or IIS 8 integrated pipeline mode, add the following key in bold text to the <appSettings> section:

```
<appSettings>
    <!--Use this setting to set config sections for Classic or Integrated
Pipeline Mode-->
    <add key="MapInfo.Engine.Session.PipelineMode" value="Integrated" />
</appSettings>
Copy the following code into the Web.config file below the <appSettings>
section.
<system.webServer>
  <modules>
    <add name="WebSessionActivator" type="MapInfo.Engine.WebSessionActivator,
MapInfo.CoreEngine, Version=9.x.x.x, Culture=neutral,
PublicKeyToken=93e298a0f6b95eb1" />
  </modules>
  <handlers>
    <add name="MapController" verb="*" path="MapController.ashx"
type="MapInfo.WebControls.MapController, MapInfo.WebControls,
Version=9.x.x.x, Culture=neutral, PublicKeyToken=0a9556cc66c0af57" />
  </handlers>
</system.webServer>
```

7. Save Web.config. Build the project and run the application.

Deploying Your Application

There are essentially two strategies for installing the MapXtreme components on the server machine where you wish to host your application: either 1) use the included runtime installer or, 2) create your own installer and add the appropriate merge modules (MSM).

The MapXtreme SDK ships with MXTRunTime.exe.

Deploying With the Runtime Installer

Use the runtime installer (MXTRunTime.exe) as part of your custom installation process. This will install the MapXtreme assemblies and create the needed registry entries. It will also include the required .NET Framework v4.7.2. Runtime assemblies and files are included for both Web and Desktop applications.

For web-based applications, using the runtime installer is the better strategy. All of the runtime components are installed in their default locations. There are no user-configurable options.

One disadvantage of using the runtime installer executable is that, depending on your application, unnecessary files may be installed. Additionally, if you expect to install more than one MapXtreme-enabled application, you will need to maintain your own reference counts when using the runtime installer. By using the MSMs directly with your own installer, references are maintained automatically.

Steps for Deploying an Application Using the Runtime Installer

You must have Administrator permissions on the computer in order to run the installer. You also need IIS permissions to install to the Web server.

1. From the MapXtreme product media, choose **Install**. The Install options page appears.
2. Choose **Deployment Installation** and choose **Install Runtime Environment.** The install wizard opens.
3. Follow the prompts to proceed with the installation.
4. Deploy your Windows or Web application as you wish (for example, create a deployment project in Visual Studio and add your application).

If you wish to run the runtime installer from a command line, follow these instructions. The executable is located in the [DVD root]:\Install\InstallRuntime folder on the product media.

- To run the runtime installer with minimal UI, execute the runtime installer like this:
  `MXTRunTime.exe /v"/qb"`

- To run the runtime installer silently, execute the runtime installer like this:
  `MXTRunTime.exe /v"/qn"`

If you are using a software copy protected version of MapXtreme, the runtime executable is called MXTRuntime.exe.

Deploying With Your Own Installer

Create your own installer using Windows Installer technology (MSI) and include the MapXtreme merge modules. A merge module (MSM file) is a single package that contains all files, resources, registry entries, and setup logic necessary to install a component. Merge modules cannot be installed alone; they must be assimilated into an MSI file. Use this strategy if you want to fine-tune exactly which components are installed, or if you want to create your own MSI-based installer.

One disadvantage of using the MSMs is that you will have to create your own patch or updated installer if Precisely provides updates to this product.

An advantage of using the MSMs is that you control which components get installed and reference counts are maintained automatically. Assemblies of a particular version will be copied into the Global Assembly Cache (GAC) only once, and reference counts will be maintained for each application using those assemblies. If one application is subsequently removed, the reference count (which then decreases by one) will preserve the assemblies in the GAC. When the last application using those assemblies is removed, then the assemblies themselves will be removed.

Several developer tools are available to help you create an MSI installer. InstallShield Developer (Acresso Software Corporation) and Visual Studio (Microsoft) are examples. See the Windows Installer topic in the Microsoft MSDN library.

MapXtreme Merge Modules

The following are the merge modules for MapXtreme. Include them as necessary in your installer MSI. See Packaging Your Desktop Application for more information on using merge modules.

Note the 9.x.x in each MSM filename represents the current version of the product. MSMs are located in \Program Files\Common Files\Merge Modules.

Name of Merge Module	Included Assemblies	Purpose	When Required
Custom Symbols MapInfoCustSymb_9.x.x.msm	none	Contains custom symbols	Required when the application expects to use the stock bitmap symbols
Desktop MapInfoDesktop_9.x.x.msm	MapInfo.Windows MapInfo.Windows.Dialogs MapInfo.Windows.Framework MapInfo.CommonDialogExt	Contains assemblies with .NET controls for use in C# and VB .NET desktop applications	Required when the application uses the 'Desktop' tools
Fonts MIFonts_9.x.x.msm	none	Contains MapInfo fonts such as Symbols, Cartographic, Real Estate, Arrows, Miscellaneous, Oil&Gas, Transportation, Weather, and Shields.	Required when the application expects to use the stock TrueType ® fonts

Name of Merge Module	Included Assemblies	Purpose	When Required
Mapping MapInfoCoreEngine_9.x.x.msm MapinfoMXTConfig_9.x.x.msm MapInfoCoreEngineIntl.msm * MapInfoCoreResJPN_9.x.x.msm † MapInfoCoreResCHN_9.x.x.msm ‡	MapInfo.CoreEngine MapInfo.CoreEngine.Wrapper MapInfo.CoreTypes MapInfo.WMS.Client MapInfo.Windows.Printing MapInfo.Ellis.ExtensibleDataProvider MapInfo.LinearReferencing MapInfo.Ogc MapInfo.WorkspaceManager.Extension	Provides the core mapping functionality. They also install the common configuration and default reference files for these assemblies.	Required

Name of Merge Module	Included Assemblies	Purpose	When Required
Web Controls MapInfoWeb_9.x.x.msm	MapInfo.Web MapInfo.WebControls	Installs the.NET assembly MapInfo.Web, and a number of 'web resources' to be used in web-based applications. It also starts the ASPNetState service, and creates a virtual directory for the 'web resources' (if IIS is installed).	Required for any application that uses the stock web controls
Web Services Clients MapInfoServices_9.x.x.msm	MapInfo.Services	Contains the assemblies for geocoding and routing	Required if your application requires geocoding and routing

Name of Merge Module	Included Assemblies	Purpose	When Required
WFS MapInfoWFS_9.x.x.msm	MapInfo.WFS.Server	Contains the assemblies needed for WFS.	Required if your application uses a Web Feature Service for data transformation
WMS MapInfoWMS_9.x.x.msm	MapInfo.WMS.Server	Contains the assemblies needed for WMS.	Required if your application uses a Web Map Service for retrieving digital images.

* Include if you are deploying applications built with MapXtreme.

† Include if you are deploying applications built with MapXtreme JPN.

‡ Include if you are deploying applications built with MapXtreme CHN.

To Deploying an Application With Your Own Installer

Here are the tasks you must do to get MapXtreme to install with your installer:

1. Include the MSMs you need.

   By default, Visual Studio and InstallShield look in C:\Program Files\Common Files\Merge Modules for merge modules to include. This allows you to build installers immediately, without having to re-configure your development environment.

Deploying a Web Application

If you create your own deployment for a Web application, you will need to add the MSMs manually. The web setup project does not detect the assemblies that are referenced within the Web.config file; thus the assemblies (and the corresponding MSMs) are not detected as dependencies. To add the MSMs manually, right-click on your Web setup project and choose Add>Merge Module. Select MapInfoCoreEngine_9.x.x, MapInfoMXTConfig_9.x.x, MapInfoWeb_9.x.x and any other MSMs you need.

Deploying Applications that Access Data

Any Visual Basic or Visual C# application that includes data access has a dependency on Microsoft Data Access Components (MDAC) version 2.7 or later. MDAC must be installed on a target computer prior to installing your application or the application will fail.

MapXtreme Web Applications Behind Proxy Servers

If you are deploying your web application behind a proxy server, be sure to add the MapXtreme Server URL to the proxy server's bypass cache list. This will allow MapXtreme to deliver dynamic maps for every request, which it is designed to do.

Proxy servers rely on cached images for display. MapXtreme's images, however, are built for every request, so in the case of the web application located behind the proxy server, no images are sent to the cache. The web application displays a red X in place of the map image.

Permissions to Temp Directory for Deployed Web Applications

For deployed web applications, if you are using integrated security, ensure all users who will access the site have permissions on the temp directory and any other resources. MapXtreme executes in the ASP.NET process space and this process executes with the security token passed from the IIS process. You must grant access to any user who will log in access to this directory. If you are using anonymous access then you must grant access to the temp directory to the IUSR_*LocalMachineName* system account. MapXtreme gets the temp directory from the current TEMP environment setting.

Application Data Files

Application data files are nonexecutable files used by an application. MapXtreme installs and uses the following set of application data files. For information about customizing these file locations, see Location of Application Data Files:

File Type	Filename
Abbreviation file	MAPINFOW.ABB
Pen file	MAPINFOW.PEN
Projection file	MapInfoCoordinateSystemSet.xml
Vector symbol file	MapInfow.fnt
Custom symbol directory	CustSymb
Nadcon files	*.las, *.los
jgd2000 files	jgd2000.*

By default, MapXtreme applications look in the following directories for data files:

- Program Files\Common Files\MapInfo\MapXtreme\9.x.x—This is the directory the MapXtreme installer places these files.
- The directory where your application is located. For a Windows application, this is the directory where the .exe file is located. For web applications, this is the directory where the Web.config file is located.

Deployment Installation Troubleshooting

Consider the following questions when troubleshooting a deployment installation:

- What was used to deploy the application (for example runtime, SDK, Setup.exe?)
- Is the customer on the deployment machine logged on as the administrator?
- Is the 4.7.2 Framework installed on the deployment machine?
- Is the CoreEngine registered as a COM+ application on the deployment machine?
- Is the Visual C++ runtime component (CRT) for Visual Studio 2015 and Visual Studio 2017 installed on the machine?

3

3 – Mapping Concepts

Before you create a mapping application, it's helpful to understand basic mapping concepts and how these concepts are implemented in MapXtreme. This chapter discusses the common concepts you will come across as you learn MapXtreme.

At the end of this Developer Guide, we have provided Appendix M: Glossary containing mapping and programming terms that you will also find useful.

In this chapter:

Mapping and MapXtreme

The central element to a mapping application is the map. This chapter presents a short overview of the most important mapping terms that you will likely encounter while building your application with MapXtreme. The introductions also point you to the appropriate namespace in the MapXtreme object model so that you can quickly get the technical information you need. The topics include:

- Maps
- Tables
- Layers
- Features
- Labels and Legends
- Themes
- Tools
- Workspaces
- Coordinate Systems and Projections

Maps

A map displays the spatial relationship among map features, such as town boundaries, customer locations, or power lines. The map visually orients you to where those features are and what they represent. In addition to features, elements on the map can include labels, titles, legends, and themes. Themes are created based on some action taken involving the features and information on the map.

The map is contained in a MapControl. The MapControl also provides basic tools for viewing the map (pan, zoom in, zoom out, center).

In MapXtreme you can create a map in a variety of ways:

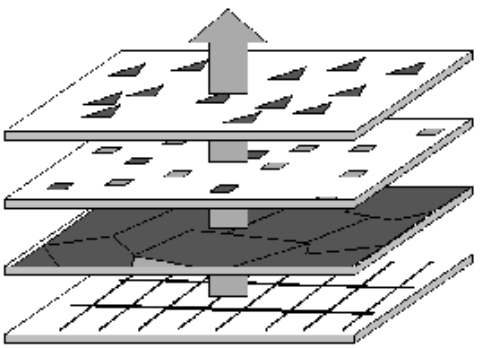
- Use the MapXtreme Workspace Manager to build and save a map workspace. (See Features of the Workspace Manager).
- Use a MapXtreme template that provides a MapControl that you can drag and drop onto a Visual Studio form (See Chapter 7 Desktop Applications, Controls, Dialogs, and Tools for desktop applications and Chapter 5 Web Applications, Controls, and Tools for web applications.
- Use the MapXtreme Object Model to programmatically build mapping into your application (See MapFactory and the MapInfo.Mapping namespace in the Developer Reference (online help).

Tables

Tables contain the data you wish to display on the map. Tables hold rows and columns of information that describe the features, including their geometry, style, and attributes. MapXtreme supports tables from a wide variety of sources including, native tables (MapInfo .TAB), relational database management systems (RDBMS), dBase, MS Access, ASCII files, NativeX, Geopackage, and ESRI ShapeFiles. Speciality tables include raster, grid, seamless, views, WMS, and ADO.NET. The type of table is available through the TableInfo class. Tables are opened and closed via the Catalog in the MapInfo.Data namespace. See Chapter 8 Working with Data.

Layers

Maps are made up of layers. Layers contain map features, such as postal code boundaries, schools, or streets networks. It is important to understand the order of the layers. The bottommost layer is drawn first and the topmost layer drawn last. Layers containing features that would obscure the features of other layers should be placed lower, respectively. For example, a layer of boundary regions should be placed beneath a layer of points.



Layers in MapXtreme can represent more than map features. Layers can be raster or grid images, seamless maps (joined maps). They can contain labels or user-drawn features, or contain an object theme, such as a pie theme. Layers can be grouped for easier positioning and to facilitate animation of their features. The main interface is IMapLayer. For more information see Layers.

Features

Features are described by their geometry, style, data source, key and attributes. Typically a feature is a row in a table. Supported geometries include closed objects that cover a given area (Polygons, MultiPolygons, Rings, Rectangle, RoundedRectangles, and Ellipses); point objects that represent single locations of data (Points, MultiPoints); and line objects that cover a given distance (Curves, MultiCurves and LegacyArcs).

One of the main uses of computerized maps is to gather information about the features. In MapXtreme features are returned in FeatureCollections by any of several methods, either created from scratch using a schema, selected using selection tools or methods or by searching the Catalog for those that meet a specific set of criteria.

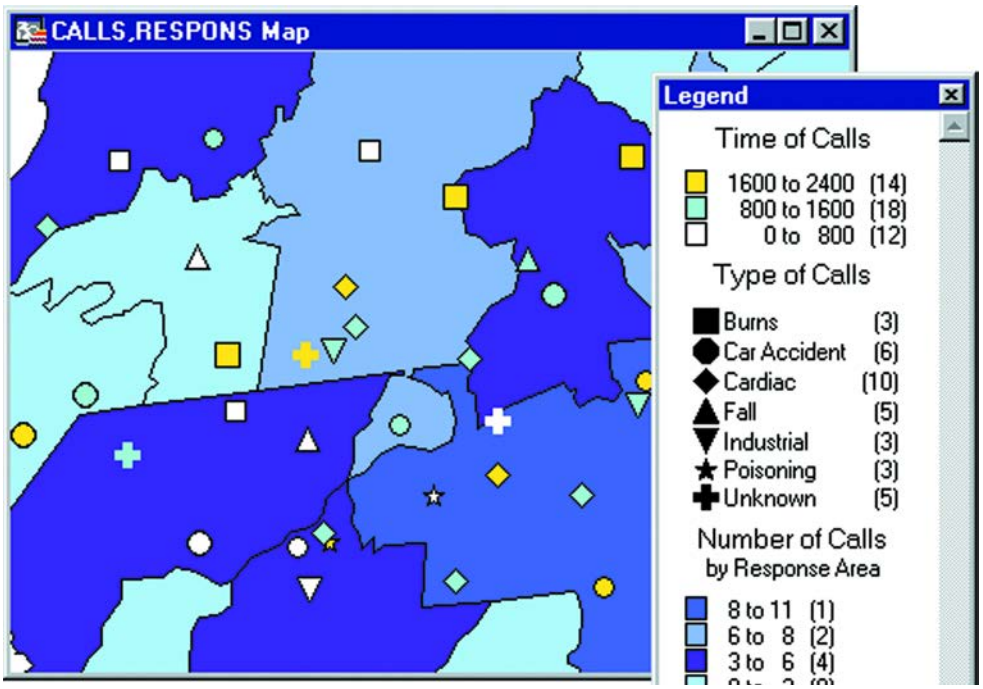
Feature classes are located in the MapInfo.Data namespace.

Labels and Legends

Maps without elements to describe what is displayed are not very useful. Maps need text such as labels and legends. Labels have been mentioned above as belonging to a type of layer called a LabelLayer. This allows you to control every aspect of a label's visibility, position, style, and content. MapXtreme classes for working with labels include LabelSource, LabelProperties, and LabelModifiers. See Introduction to the MapInfo.Mapping Namespace.

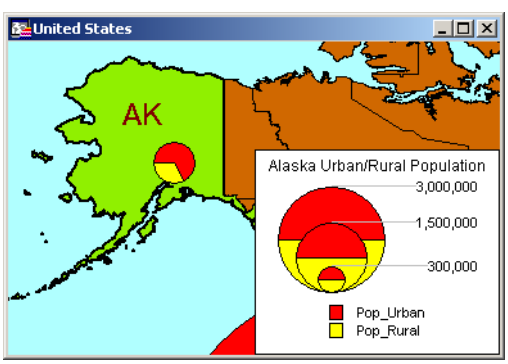
Other text elements can also be used in a map to help deliver its message properly. Legends are cartographic elements that describe the features in a coded manner. For example, the legend may describe the boundaries as school districts, the lines as a power line network, or points as corporate office locations. Legends also contain a title to describe collectively what the map represents.

In MapXtreme, legends are part of the Adornments class, along with map titles and scalebars. Adornments reside in the MapInfo.Mapping namespace.



Themes

Computer maps are not only useful for visibly showing spatial relationships among the map features, but you can analyze the underlying data that is associated with the features to learn more about what you see. A common analytical technique is to create a theme based on a feature layer in which the data is ranked in specific ways. For example, a ranged theme shows color blocks where each color represents features on the map that meet the same criteria. A graduated symbol theme is useful for showing distributions of populations for example, with the largest symbol representing the largest population.



Themes can also be created for labels. For example, use a ranged label theme to show the relative population size among cities. The largest labels represent the cities with the largest populations.

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as style overrides on Feature layers and as Object themes. Modifier themes change the style, while object themes add a new layer. All themes implement the ITheme interface.

Tools

Most mapping applications provide an assortment of toolbar buttons (tools) to aid with common drawing tasks (such as drawing a line on the map) and navigation tasks (such as zooming in). MapXtreme provides a number of mapping tools, plus you can also create your own custom tools.

The tools are divided into desktop tools and web tools, the API for each contained in their own namespace (MapInfo.Tools for desktop and MapInfo.WebControls for web tools.)

For more information on desktop tools in MapXtreme see MapXtreme Desktop Tools API. For more information on web tools see Chapter 5 Web Applications, Controls, and Tools.

Workspaces

While not strictly a mapping concept, workspaces are included here because they will make working with all the mapping elements easier. MapXtreme supports an XML-based workspace format that uses the .MWS extension. In it are all the settings for your maps. The format for the workspace is explained in Appendix C: Understanding the MapInfo Workspace. MapXtreme provides a utility called Workspace Manager to help you construct workspaces and save them for later use. See Chapter 27 Workspace Manager.

Coordinate Systems and Projections

Coordinate systems and projections are two important mapping concepts about which you should have a basic understanding. Projection refers to how a map is displayed on a flat surface such as a paper map or computer screen, while a coordinate system describes how map features are spatially arranged. Both are important considerations when developing applications, especially those where spatial precision and accuracy are important.

A projection is a method of reducing the distortion that occurs when objects from a spherical surface are displayed on a flat surface. There are two main trade-offs to be considered: the preservation of equal area, and the preservation of the true shape of a

feature. There are many different types of projections, each designed to reduce the amount of distortion for a given area. Some projections preserve shape; others preserve accuracy of area, distance, or direction.

A coordinate system is a set of parameters that tells you how to interpret the locational coordinates for objects. One of those parameters is projection. Coordinates can be of two types: Spherical or Cartesian. Spherical relates to locations on the curved surface of the Earth, while Cartesian describes flat surface locations in two dimensions. Both are represented by x and y coordinates. The difference comes when calculating distance or area of features that represent real Earth locations such as streets or rivers (Spherical), or relative locations, such as a map of brain anatomy or a chess board (Cartesian).

Knowing which coordinate system your map uses is an important consideration when developing applications. Analytical operations involving distance and area calculations, such as buffering, routing, and querying use the coordinate system and projection to yield the correct results.

Coordinate system and projection classes are part of the MapInfo.Geometry namespace. For more information see Chapter 16 Spatial Objects and Coordinate Systems.

Geocoding with MapXtreme

All of the maps discussed above use data that provide additional information beyond what you can see on the map. For example, a table of store locations not only includes geographic coordinates to place the stores in the correct map location, it may contain data about the locations, such as store hours, customer service phone numbers and manager name. This gives the application the power to analyze and yield information that would otherwise be lost in rows and columns of tables.

Typically a table of custom data is included on a map along with reference layers, such as streets, town boundaries and water features that represent the true environment of the area. These references layers are usually purchased ready to display on a map. Precisely sells a wide variety of reference data for locations around the world. Additionally, MapXtreme provides more than 400 MB of sample data for world locations. To install, from the MapXtreme product CD browser, choose Install Sample Data. Use the Custom installation option to install as much of this data set that you need.

But the custom data, like your store locations or call center regions, may not be ready to display on a map. The table must contain geographic coordinates so the mapping engine knows where to draw the objects. The process of assigning coordinates to data is called geocoding. Any table of data that contains locational information, such as address or postal code, can be geocoded. The process involves matching the custom table against

an already geocoded table covering the same location. If an address match is made, the coordinates from the geocoded table are assigned to the custom data. Then the custom data is ready to be viewed on a map.

Geocoding is a typical early step in the process of creating a map. As a developer of mapping applications, you will need to consider the type of data you wish to display on the map and its need to be geocoded.

The MapXtreme framework provides classes for using a geocoding client that can access Precisely's server geocoding products. For more information about geocoding see Chapter 20 Geocoding and the MapInfo.Geocoding namespace in the online Help (accessible via Visual Studio).

Routing with MapXtreme

Another component available to developers of MapXtreme is routing. Driving direction applications and those involved in planning routes for deliveries or laying cable, for example, utilize routing. Typically the goal is to locate the route by shortest distance or shortest travel time.

MapXtreme provides four types of routing: point-to-point routing, multi-point routing, matrix routing, and isogram routing. Each type offers numerous options for creating the appropriate routing network for your needs.

Like the provision for geocoding, MapXtreme allows developers to use a pre-built routing client in their application that interacts with Precisely's routing server products. See Chapter 21 Routing and the MapInfo.Routing namespace in the MapXtreme Online Help.

4 – Understanding the MapXtreme Architecture

This chapter focuses on the design of the MapXtreme architecture so you can make informed choices for your development needs. Understanding the architecture of the product will help you to create applications that efficiently use the features and capabilities of the product.

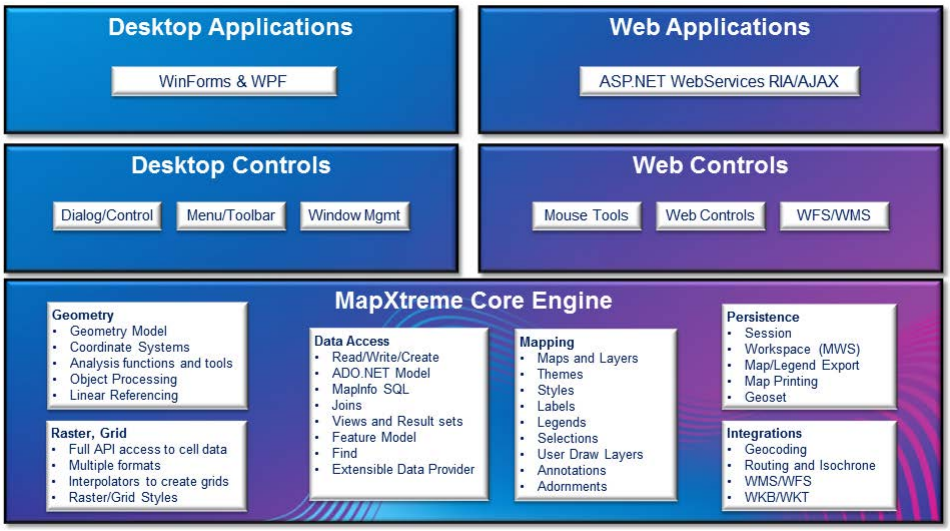
In this chapter:

MapXtreme Architecture

MapXtreme is built on top of Microsoft's .NET framework and utilizes the functionality that its infrastructure includes. This Precisely initiative enables you to leverage the power and adaptability of developing your applications on the .NET framework. We have also worked hard to combine the power and ease of our Windows products into one comprehensive object model. The object model is the basis for Precisely's partners and customers, as well as Precisely itself, for developing Windows-based products for the foreseeable future.

With similar code, you can develop an application that can be deployed on either a desktop machine or the Web. If you develop an application for the desktop, you can then adjust the application for subsequent web deployment with only minimal code changes.

The following figure illustrates the MapXtreme architecture. The MapInfo.CoreEngine.dll assembly and MapInfo.CoreTypes.dll assembly contain most of the core mapping and data access functionality. On top of the Core Engine are the MapInfo.Windows and MapInfo.Web namespaces that contain controls, tools, and other functionality specific to each deployment environment. An application developed using the MapXtreme object model is built atop of the MapInfo.Windows or MapInfo.Web namespace.



MapXtreme *Architecture*

Object Model Overview

The MapXtreme Object Model is made up of many namespaces. A .NET namespace is a type classification system that differentiates specific classes, methods, and properties from others with the same name. By utilizing namespaces, .NET developers can avoid collisions between names of objects and their methods and properties.

MapXtreme ships with a full-color poster of the key interfaces and classes and shows how they relate to each other via namespace segregation. A PDF version of the poster is viewable from the Learning Resources page, which is accessible from the Windows Start menu. Specifically, the Learning Resources page can be accessed from: Start > All Programs > MapInfo > MapXtreme > Learning Resources.

The list below contains several of the namespaces implemented in the MapXtreme Object Model. A broad overview of each namespace is included in the following sections. Each description contains a reference to the part of this manual that contains detailed information about it and its use.

- MapInfo.Data Namespace
- MapInfo.Data.Find Namespace
- MapInfo.Engine Namespace
- MapInfo.Geometry Namespace
- MapInfo.Mapping Namespace
- MapInfo.Persistence Namespace
- MapInfo.Raster Namespace
- MapInfo.Styles Namespace
- MapInfo.WebControls Namespace
- MapInfo.Tools Namespace
- MapInfo.Geocoding Namespace
- MapInfo.Routing Namespace

The complete object model is organized by namespace in the MapXtreme Programmer's Reference, which is integrated into Visual Studio.

If you have used MapX or the non-.NET version of MapXtreme (MapXtreme for Windows v3), be sure to review Appendix J: Migrating to MapXtreme for a comparison of the two product's object models.

MapInfo.Data Namespace

The MapInfo.Data namespace contains the classes and interfaces that implement the MapInfo Data Provider. The object model has several different classes to access data. Depending on the format in which your data is stored, there are specific classes to use to access it. Additionally, we now implement ADO.NET access to any data that is contained in formats not covered by any other class. For details on the MapInfo.Data namespace, see Chapter 8 Working with Data and Chapter 11 Accessing Data from a DBMS.

MapInfo.Data.Find Namespace

The MapInfo.Data.Find namespace contains the classes used for searching through data. The namespace facilitates the search for an object by specifying a mappable table and column (it must be indexed) on which to perform the search. For details on the MapInfo.Data.Find namespace, see Chapter 13 Finding Locations.

MapInfo.Engine Namespace

The MapInfo.Engine namespace contains all classes directly related to the core functionality that drives all applications based on MapXtreme. This includes the core Session class which is the starting point for all MapXtreme applications. For details on the MapInfo.Engine namespace, see Chapter 9 Working with Core MapXtreme Classes.

MapInfo.Geometry Namespace

The MapInfo.Geometry namespace is an extensible hierarchy based on OGC (Open GIS Consortium) standards, coordinate system interoperability, and object processing. The MapInfo.Geometry namespace contains classes, interfaces, and enumerations for creating and editing Geometry objects. For details on the MapInfo.Geometry namespace, see Chapter 16 Spatial Objects and Coordinate Systems.

MapInfo.Mapping Namespace

The MapInfo.Mapping namespace contains classes, interfaces, and enumerations for creating, displaying, and exporting maps, layers, modifiers, and labels. For details on the MapInfo.Mapping namespace, see Chapter 12 Adding Mapping Capability to Your Applications.

MapInfo.Mapping.Legends Namespace

The MapInfo.Mapping.Legends namespace contains classes, interfaces, and enumerations for creating and displaying Cartographic and Thematic Legends. For more information, see Legends and Using Themes and Legends.

MapInfo.Mapping.Thematics Namespace

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as styles of layers and as layers themselves. Themes can be applied to change the style. For examples, the Modifier theme changes the style of the layer and the Object theme adds a new layer. All themes implement the ITheme interface. For details on the MapInfo.Mapping.Thematics namespace, see Chapter 14 Using Themes and Legends.

MapInfo.Persistence Namespace

The MapInfo.Persistence namespace contains classes that support the reading and writing of XML-based workspaces to enable the saving and retrieval of mapping workspaces. See Appendix C: Understanding the MapInfo Workspace.

MapInfo.Raster Namespace

The MapInfo.Raster namespace exposes the full functionality of Precisely's C/C++ Raster and Grid APIs. Raster images can be opened for querying using MapInfo.Raster.RasterRead. Grid images can be opened for querying using MapInfo.Raster.GridRead. Hillshading can be added to existing grids using MapInfo.Raster.HillshadeWrite. Related classes include MapInfo.Raster.RasterInfo and MapInfo.Raster.GridInfo. For details on the MapInfo.Raster namespace, see Chapter 17 Working with Rasters and Grids.

MapInfo.Styles Namespace

The MapInfo.Styles namespace highlights the Styles object model. The Styles class is the base class of all styles. For details on the MapInfo.Styles namespace, see Chapter 15 Stylizing Your Maps.

MapInfo.WebControls Namespace

The MapInfo.WebControls namespace provides support for using Visual Studio templates for a MapXtreme ASP.NET application. There are MapControl and LayerControl design-time enhancements available from this namespace, as well as web tools. For details on the MapInfo.WebControls namespaces, see Chapter 5 Web Applications, Controls, and Tools.

MapInfo.Windows Namespace

The MapInfo.Windows namespace contains classes that implement various windows controls and their requisite components for use with developing forms in Windows applications. The Windows.Dialogs namespace contains classes that implement various dialog boxes and dialog box components to be used in Windows applications. For details on the MapInfo.Windows namespace, see Chapter 7 Desktop Applications, Controls, Dialogs, and Tools.

MapInfo.Tools Namespace

The MapInfo.Tools namespace contains classes for creating and implementing many types of tools to use in your desktop map application. For details on the MapInfo.Tools namespace, see Overview of the MapInfo.Tools Namespace.

MapInfo.Geocoding Namespace

The MapInfo.Geocoding namespace contains the classes, interfaces and enumerations that define the MapXtreme client for geocoding. Geocoding using either the MapInfo geocoding server or the MapInfo Location Utility service is supported. The URL of a running geocoding server or Location Utility service must be available in order to perform geocoding. The interface of the geocoding server and Location Utility service are similar, since they both use the same classes for geocode requests, constraints, responses, result codes, and input and candidate addresses. See Chapter 20 Geocoding.

MapInfo.Routing Namespace

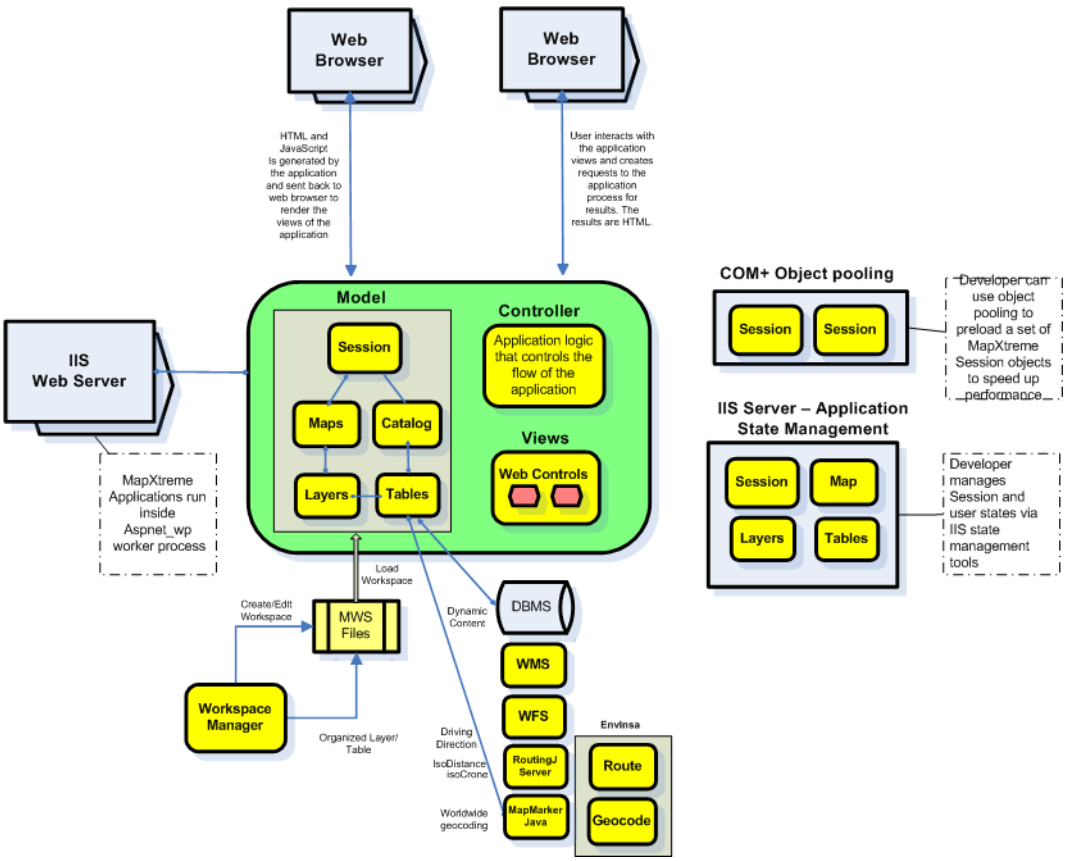
The MapInfo.Routing namespace contains classes, interfaces, and enumerations which comprise the .NET client for Routing. The MapInfo.Routing namespace contains classes that support point-to-point, multi-point, matrix and isogram routing. optimized for shortest time or shortest distance. It interacts with other MapInfo routing server products. The ability to avoid certain points is also available. Routing results can include step by step directions and/or a route geometry which can be displayed on a map. See Chapter 21 Routing.

Application Architectures

Now that you have had an overview of the MapXtreme namespaces, the next step is to consider the architecture of the application you plan to build.

Using MapXtreme, you can build both web and desktop applications. The follow sections illustrate possible architectures for web and desktop applications. The designs are based on the Model-View-Controller paradigm that separates an application's data model, user interface, and control logic into three distinct components. This allows for modifications to one component with minimal impact to the others. Chapter 5 Web Applications, Controls, and Tools and Chapter 7 Desktop Applications, Controls, Dialogs, and Tools provides important additional information on key design elements and decisions you need to consider when planning your MapXtreme application.

Web Application Architecture



Architecture Description

The Web application illustrated above takes into account the following components and capabilities:

- MapXtreme Web Application
- Microsoft .NET Infrastructure
- Map-building Tools
- Data Access

MapXtreme Web Application

A typical MapXtreme web application contains views (presentation layer), a model (to interact with data source and application internal data model), and controller (the business logic that controls the flow of the application).

MapXtreme provides web templates that are integrated into Microsoft Visual Studio to help you create your initial web application. For a tutorial on building a web application, see Building a Web Application.

Build views by dragging and dropping MapXtreme web controls onto a Visual Studio web form. Build your internal data structures and interact with external data sources, base maps and dynamic content by using objects under MapInfo.Engine and MapInfo.Data namespaces. Use the controller code to tie the views and data together and provide the user with an engagement sequence to effectively use the application to resolve a business need or problem.

Microsoft .NET Infrastructure

MapXtreme runs under Microsoft .NET 4.7.2 Framework. An application built using MapXtreme runs as an ASP.NET application under the worker process of IIS.

The Microsoft ASP.NET framework provides COM+ object pooling for developers of high performance enterprise applications so that objects such as workspaces can be preloaded. MapXtreme's object model operates very efficiently under this framework. The framework also provides application state management tools such as StateServer and SQL Server, as well as automatic and manual state management control. Chapter 6 Understanding State Management presents important information on these topics.

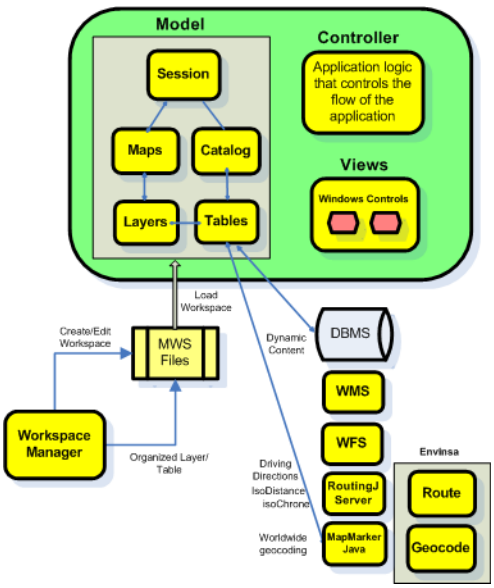
Map-building Tools

Use the MapXtreme Workspace Manager to create your application's base maps. Here you can manage each layer of a map and control its zoom level, labeling, styles, themes and adornments to give you exactly the presentation you need for your application. The information is saved to an XML-based workspace for easy retrieval at a later date. See Chapter 27 Workspace Manager.

Data Access

A key element of this architecture is the ability to access to dynamic data content. Objects that exist within the MapInfo.Data namespace provide this data access. Data content can come from a number of sources, such as WMS, WFS, remote database management systems, live feeds from GPS or driving directions from the MapInfo Routing service. To make the most of disparate data, you can concurrently use information from different data sources. See Chapter 8 Working with Data.

Desktop Application Architecture



The desktop application architecture is similar to the web application architecture in its Model-View-Controller design.

Separate components are used for the presentation layer, application model and business logic. Use the Workspace Manager to build any base maps that you need. Use Windows controls and dialogs to give your application a rich user experience. You can also concurrently use dynamic data content from a variety of sources, and control the flow and logic of the application.

See Chapter 7 Desktop Applications, Controls, Dialogs, and Tools for more information.

5

5 – Web Applications, Controls, and Tools

This chapter brings together a host of information related to building ASP.NET web applications using the web controls, tools, and conveniences provided with MapXtreme.

In this chapter:

Web Application Request/Response Lifecycle

In order to plan and build an effective web application, you need a solid understanding of the behind the scenes interactions between the client (browser) and the server (web application). In its simplest form, a web application is a software application that is accessed through a web browser over an Internet or intranet connection. The capabilities of the application are presented to a user as an HTML page, and through user interaction with the elements on the web page, HTTP requests are sent to a web server for processing. The web server sends back a response that satisfies the user's request.

A MapXtreme web application typically presents the user with an image of a map and some tools to interact with the map. A single request/response cycle could be as simple as the user clicks a Zoom-In tool to display a different view of the map. Behind the scene, the request to zoom in is sent to the server. The server processes the request and responds with a refreshed image of the map showing the new view.

For information on the architecture of a MapXtreme web application, see Chapter 4 Understanding the MapXtreme Architecture.

For a tutorial on building a MapXtreme application, see Appendix A: How to Create and Deploy a MapXtreme Application.

For a discussion of creating ASP.NET web applications, see ASP.NET Web Application Projects in the MSDN library.

Components of a MapXtreme Web Application

The following sections cover the major components that make up a typical MapXtreme web application, including:

- MapXtreme Session
- Background Map
- MapControl
- Map Tools

MapXtreme Session

The MapXtreme Session is the starting point for all MapXtreme applications. It manages the initialization of resources needed for a MapXtreme application. The MapXtreme Session also provides access to other important objects such as Catalog, MapFactory, CoordSysFactory, Selections, and others.

To access the MapXtreme Session, call the MapInfo.Engine.Session.Current() method. Each thread in your process has a MapXtreme Session object available. There can be only one MapXtreme Session per thread and this session cannot be shared by multiple threads. See Session Interface.

Web applications can have one MapXtreme Session object per user, or pooled MapXtreme Sessions available to many users. Both development models are discussed in Chapter 6 Understanding State Management.

Background Map

The background map, in the form of an image, is the most visual component of a MapXtreme web application. It provides the user of the application with information about map features and shows their relationship to other map features. Typically, a background map is made up of reference layers, such as administrative boundaries, street networks, and points of interest. Custom data related to the application, such as points representing office locations, cell towers, or ATM machines, are additional layers. The reference layers and the custom layers do not change based on the user's interaction with the application. What typically changes is the display of the map. A user may zoom into a particular location or create a thematic shading that overlays the map based on criteria the user submitted in a web request.

The background map is presented to the user in its initial, or "clean" state. This state is an important consideration when you design a pooled application since your application must handle changes in user state.

The background map is pre-loaded into the application from an XML-based workspace (.mws). The workspace is identified in the Web.config file of the application. See What Should the Initial Map Look Like?. The MapXtreme web template and sample applications provide Web.config files that include hard-coded paths to sample data. If you base your web application on one of these, be sure to adjust the path to your own data.

The Workspace Manager utility provided with MapXtreme is an example of a desktop application that assists you with building a background map. See Chapter 27 Workspace Manager.

MapControl

MapControl is a MapXtreme web server control containing an instance of a Map object. The MapControl is the main web control in a MapXtreme application; it displays the background map and responds to tool interactions on the map. Behind the scenes, the

Map object is obtained from the MapFactory using the MapAlias property of the MapControl. The map is rendered and exported as an image and returned to the browser in the control's <img> tag.

The sample applications that ship with MapXtreme have a MapControl built in, as does the MapXtreme Web application template. You will see, however, that in design mode, the map is not rendered. That is because to get a map from a web MapControl, there must be a running web application on the server to serve up the map image along with some dynamically generated javascript. That is only available at run/debug time.

ⓘ If you start from a Visual Studio Visual Basic or C# ASP.NET template, you must manually add the MapControl and tools to your web form in order for the MapXtreme resources to be included in your project.

For more information, see MapXtreme Web Controls and Tools.

Figure 1: a MapXtreme Web Application View at Design Time

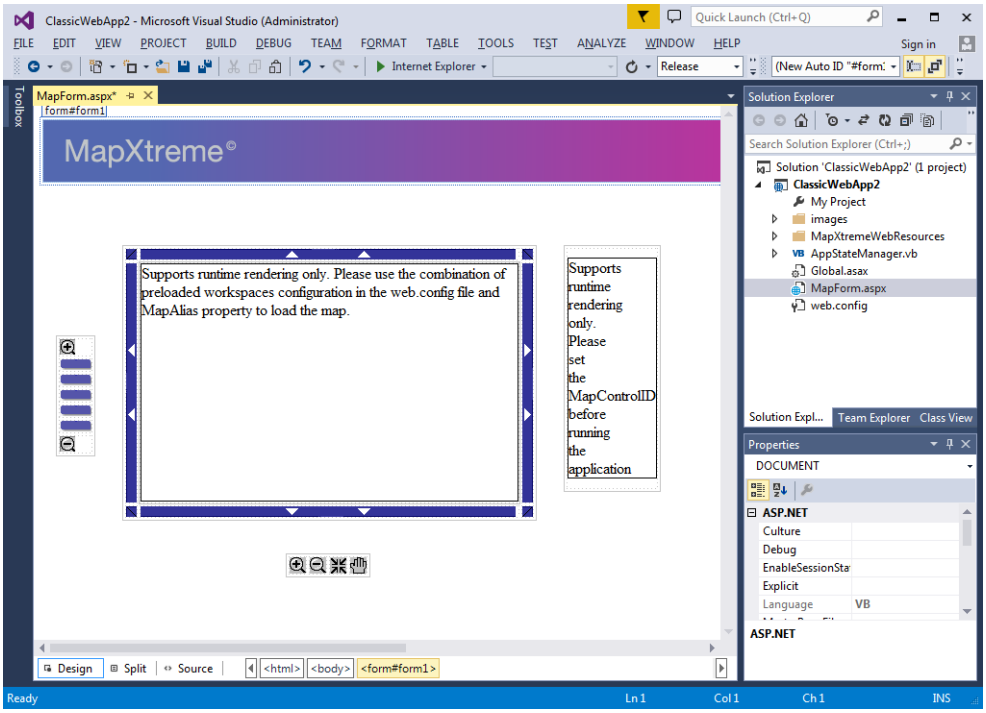
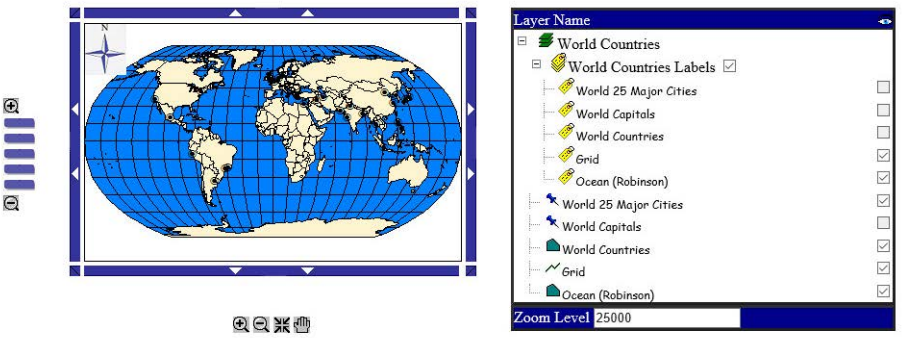


Figure 1: a MapXtreme Web Application View at Run Time



Map Tools

MapXtreme provides a number of map tools to assist you in navigating and interacting with the background map. These tools are contained in the Visual Studio toolbox. Use them by dragging and dropping the tool onto a web form.

Many of these tools are built into the MapXtreme web application template and sample applications, including:

- Basic tools for Center, Pan, Zoom-in, and Zoom-out
- ZoomBarTool with pre-set zoom levels between 500 and12,500 map units
- Directional tools (N, NE, E, SE, S, SW, W, NW) for direct line panning in 45-degree increments

The LegendControl, Distance tool and selection tools are located in the MapXtreme portion of the Visual Studio toolbox.

Tools are discussed in Description of Web Controls and Tools.

A generic WebTool, the base tool for all MapXtreme map tools is also located in the toolbox. Use this tool if you wish to add custom behaviors to a tool. See Creating a Custom Tool.

State Management and Pooling Capabilities

One of your goals in building a MapXtreme web application is to make it stable, scalable and offer a satisfying experience for your users. MapXtreme provides configurations to help you manage the state of your application, and allow it to grow with the numbers of users.

The MapXtreme web application template and the sample applications are configured to manage state manually. This means that only the changes your users make during their interaction with your application are saved and restored. You accomplish this by writing code that is specific to your application and save/restoration needs.

To more efficiently serve the potentially growing number of users of your application, the template and sample applications are configured to use the Microsoft COM+ pooling service in which MapXtreme Session objects are available and shared across multiple requests.

State management and pooling need careful consideration when you are planning your web application. See Chapter 6 Understanding State Management.

MapXtreme Web Controls and Tools

MapXtreme's web controls and tools are embedded in a web page. Web controls respond to interactions by web tools. Information that is captured using a tool is processed on the server by the web page and returned to the client, typically as a new map image.

MapXtreme provides three web controls (MapControl, LayerControl, and LegendControl) and a number of map tools. The web controls show content, such as a map, a list of map and label layers and their properties, or a legend to identify what the layers represent. The tools interact with the MapControl to change the view of the map, select features on the map for further analysis, and more. For a description of the web controls and tools see Description of Web Controls and Tools.

The Web controls and tools are available from several places within Visual Studio:

- **Visual Studio toolbox** under the MapXtreme heading. Drag and drop these controls onto your Web Form to add mapping functionality to your project.
- **Web Application template**: Start with the MapXtreme web application template (Visual Studio File > New Project menu) to build a mapping prototype that requires no coding.
- **Sample Applications**: Task-oriented samples are located In the \Samples\Visual Studio 20xx\Web\Features folder. Source code is also provided for you to learn from or extend for your own needs.

The API for Web controls and tools is contained in the MapInfo.WebControls namespace. See the Developer Reference for more information.

JavaScript-Enabled for Partial-Page Updates

MapXtreme web controls and tools use a combination of a web page and JavaScript to tell the MapXtreme server what needs to be done. Each map tool specifies a JavaScript ClientInteraction that defines what must happen (for example, click, draw a rectangle, line, or polygon) and a JavaScript ClientCommand that sends a URL request to the server to process the command (for example, pan, zoom, or select an object).

These JavaScript-enabled tools do not trigger a full page postback with each use. Typically, only the image of the map is refreshed after each tool use. See Map Tools Architecture for an understanding of this development model. See Managing Your Web Controls for information on event handling, error management and state management.

Previous MapXtreme controls (pre-v6.5 releases) required a postback that called Page_Load and Page_Unload every time a tool was used. These tools are provided for backward compatibility and are not recommended for new development projects. See Migrating Post-back Web Controls to JavaScript Web Controls.

Customizable

If the provided tools do not offer you the functionality to meet your needs, consider modifying them. This can be as simple as changing a built-in tool property or as complicated as writing your own JavaScript and server-side class to extend its functionality. Source code for the web controls and tools is provided in the Samples\MapXtremeWebControlsSourceCode folder. See Creating a Custom Tool.

MapXtreme provides an ASP.NET AJAX sample application that demonstrates how to use Microsoft's ASP.NET AJAX controls in a MapXtreme web mapping application. See ASP.NET AJAX and MapXtreme Web Applications.

Description of Web Controls and Tools

The following are the available MapXtreme Web controls and tools.

Web Controls	Description
MapControl	Allows you to display an instance of a Map object. Each Map object is referred to by a MapAlias, such as Map1. At runtime, the MapControl displays the map which is obtained from a MapFactory by using the map's MapAlias property. The map is drawn by exporting the map image, and referencing this image in the HTML <img> tag. If the MapAlias property is not specified or is invalid, the first map from the MapFactory is chosen. Set the MapControl MapAlias property at design time.
LayerControl	Allows you to display map feature layers and label layers in a tree view structure. This control can turn the visibility on or off for a particular layer and displays a read-only current zoom value. The visibility changes are persistent only for the active life of the application using the LayerControl.

LegendControl	The LegendControl allows you to display a legend for a given MapControl. The legend that is returned is a non-interactive image. The default export format is .GIF. The legend to display can be specified at design time using its LegendAlias or its index in the map's legend list. This can also be set using JavaScript on the page. JavaScript can also be used to show and hide the legend. You can arrange for the legend to be scrollable. Both thematic and cartographic legends are supported in the LegendControl. The Legend Control web sample application supports this control. It demonstrates how to create and use a customized LegendControl based on our current web control architecture, and how to create a theme and display a legend by sending requests to the server using JavaScript without needing to refresh the whole page. For more information, see the MapXtreme Learning Resources browser on your Program menu or from the MapInfo website under Support & Training. Expand the Learning Resources link and click on Sample Applications.

Map Tools	**Description**
CenterTool	Allows you to recenter the map by clicking on the map.
DistanceTool	Allows you to get the distance between two or more points by clicking on the map.
NavigationTools	Allow you to pan the map by fixed directions: North, South, East, West, NorthEast, NorthWest, SouthEast, and SouthWest.
PanTool	Allows you to reposition the map within its window by dragging in any direction.
PointSelectionTool	Allows you to select a feature (nearest to the point) when clicked on the map.

PolygonSelectionTool	Allows you to select all features whose centroids lie within the polygon. The selection area is drawn on the map using mouse clicks representing the nodes of the polygon. A double click closes/ends the polygon.
RadiusSelectionTool	Allows you to select all features whose centroids lie within the radius. The radius is drawn on the map using mouse clicks representing the center and boundary of the circle. The selection radius is drawn on the map using a click and drag operation.
RectangleSelectionTool	Allows you to select all features whose centroids lie within the rectangle. The rectangle selection area is drawn on the map using a click and drag operation.
ZoomBarTool	Allows you to zoom a map to a series of preset levels between 500 and 12,500 map units.
ZoomInTool	Allows you to zoom a map by either a single click or by selecting a rectangular area
ZoomOutTool	Allows you to draw a rectangle representing the view to zoom out of the map.
InfoTool Sample Web Application	An example of an InfoTool is included in the sample Web applications. It demonstrates how to create and use a customized map tool based on the MapXtreme Web control architecture.

Web Control Architecture

The MapXtreme Web control architecture follows the ASP.NET model for creating Web applications. The general architecture is a Model-View-Controller (MVC) design pattern, in which the web application represents the Model, the web page (HTML, JavaScript) represents the View, and the MapXtreme Server that responds to requests for information represents the Controller.

The user interacts with the web application through web controls and tools that capture data and send instructions and commands to the server which processes and returns the information.

In MapXtreme, controls and tools are rendered when the web page is rendered at initialization time. After initialization time, when a tool is used, only the map image is rendered. These JavaScript-enabled tools are an improvement over the pre-v6.5 web controls that performed a full page postback with every tool operation.

MapXtreme controls provide the following behaviors and functionality:

- A background map is loaded via a pre-defined workspace. At design time, set the MapControl MapAlias property to the map alias of the map defined in the preloaded workspace. At runtime, the corresponding map is loaded into the MapControl, ready for users to interact with it using map tools.
- The first time a page is rendered, Page_Load and Page_Unload are called. Page_Load initializes the state of the application, either to a default state for a new session, or restores state if the session is not new. Page_Unload stores the state of any changes, in anticipation of another request.
- The StateManager is implemented in an application as a class, and an instance of that class is put in the MapXtreme Session. SaveState and RestoreState methods are called from this object. The SaveState and RestoreState methods are called every time a tool is used. If Manual mode is used for state management, a StateManager class instance must be in the session. (MapInfo.Engine.Session.State is set to Manual in your Web.config file.) For more on state management, see State Management.
- Error processing is done in the global_asax.cs/.vb file in the application_error event handler. See Error Management.
- Events are handled through client-side JavaScript commands that send a request to the server. A server-side command class does server-side processing. See Event Handling.

The MapXtremeWebResources folder included with the Visual Studio MapXtreme Web templates contains dependencies for the web controls and map tools. Make sure to include these files when deploying an application.

Map Tools Architecture

MapXtreme map tools are used to interact in some way with the map, such as panning to a different view or selecting an area of the map to collect data for further analysis. A basic set of navigational tools are provided in the MapXtreme Web application template. These tools, along with others for selecting map features and creating a legend, are located in the Visual Studio toolbox.

The map tools are made up of client-side and server-side components. On the client side, the tools have a JavaScript interaction component and a JavaScript client command component. On the server side, the tools have server command class component.

The client-side map tool components are responsible for:

- Drawing and mouse operations (for example, rubber band rectangle, mouse click)
- Collecting data from mouse operations (for example, getting the screen coordinates for a zoom-in operation)
- Sending the url request to the server

The server-side map tool components are responsible for:

- Carrying out the business logic of the tool (for example, calculating the distance between two points)

Client Side Map Interaction

The client side map tool interaction is implemented with JavaScript classes. These are generic classes that can be used by any user interface to perform an interaction on any HTML element. The base Interaction class, located in Interaction.js, is extended to create all of the individual interactions such as ClickInteraction and RectInteraction. The constructor for Interaction is:

```
Interaction(elementID, onComplete)
```

where `elementID` is the IMG tag of the map and `onComplete` is the function which is called when the interaction is complete.

For example, the Interaction class can a draw a rubber band rectangle over the map and collect all the point features that are contained within it.

Client Side Command Execution

A map tool has a client side JavaScript command object that is responsible for performing a specific task. The base Command class, located in Command.js, is extended to create all of the individual commands such as PanCommand and ZoomCommand. After the map tool interaction is complete, the tool executes the client command The Command class constructor is:

```
Command(name, interaction)
```

where `name` is the server side Command class name and `interaction` is the data gathered during the client interaction.

The client Command generates a URL request that is sent to the server, which then processes the response to display the new map.

The interaction object can be null. This means there is no client side interaction, such as pan, zoom, point select, that will fire off the command automatically. The command can still be fired, but it would have to be done programmatically.

Server Side Command Architecture

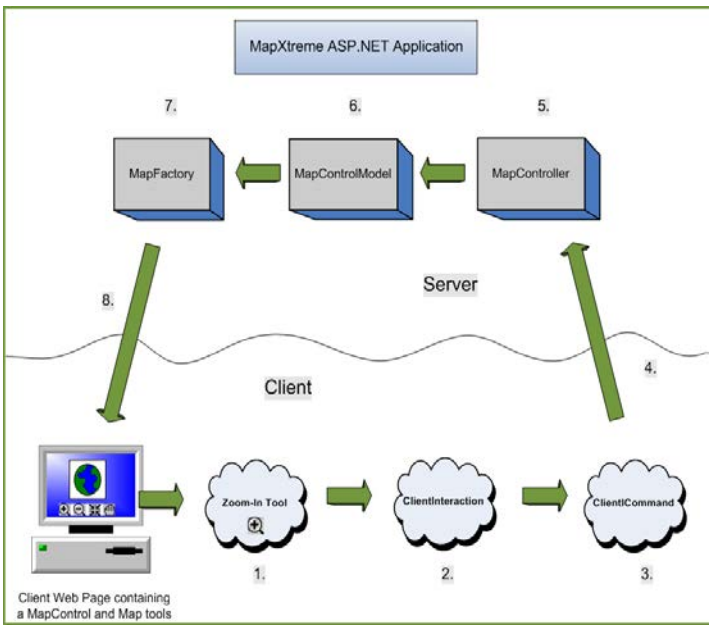
On the server, the MapInfo.WebControls.MapControlModel class handles the URL request sent from the client. This class contains methods for carrying out basic map navigation commands such as pan and zoom, as well as selection commands for selecting points, polygons, and features within drawn rectangles and radii. The InvokeCommand method locates the specified tool command and calls its Execute method. Execute calls the RestoreState, then Process, and then finally SaveState.

To perform commands other than those provided in this class, you must write a Command class that extends the WebControls.MapBaseCommand class.

How a Map Tool Works

The following describes a typical request/response cycle for a web map tool, in this case, a Zoom-In tool. You can create custom map tools using this same architecture. See Creating a Custom Tool.

The numbers in the diagram refer to the stages described below.



1. A user draws a rectangle around the general area of Europe displayed in a MapControl using a Zoom-In tool.
2. The tool's ClientInteraction property called RectInteraction collects the screen coordinates that define the rectangle. RectInteraction is defined in Interaction.js.

3. The tool's ClientCommand property MapCommand creates a URL request and assigns it to the MapControl's image.src property. ClientCommand is defined in Command.js. The URL request looks like:

   ```
   MapController.ashx?Command=Zoom&Width=300&Height=300&ZoomLevel=1200&Points=1,
   50,100&MapAlias=Map1
   ```

4. The URL request is sent to the server.
5. The MapController receives the request and invokes the MapControlModel. The MapController derives from System.Web.IHttpHandler.
6. The MapControlModel parses the URL request and invokes the ZoomInCommand class.
7. The ZoomInCommand gets the map from the MapFactory and carries out the operation to zoom in on the map. The map image is updated to reflect the new view.
8. The map image is exported to an image, and written to the HTTP response as a stream, and returned to the client.

Using the MapXtreme Web Controls

To use the MapXtreme Web controls in your web application:

1. Do one of the following:
   a. Create a MapXtreme web application from Visual Studio File > New Project and choose the MapXtreme Web Application template. The MapControl, LayerControl and map navigation tools are built in.
   b. Open one of the sample web applications and modify it for your needs.
   c. Drag and drop what you need onto the web form from the MapXtreme Web 9.x.x toolbox in Visual Studio.
   d. Create a MapXtreme web application from Visual Studio File > New Project and choose the generic ASP.NET Web Application template [1].
2. In your application's Web.config file, specify the path and name of a workspace that contains map layers for pre-loading into the MapControl. This is necessary regardless of which choice you made in step1.

1. In this situation, you will notice that the MapXtreme web controls and tools display red X's in the Designer instead of their icons. To display the icons properly, copy the MapXtremeWebResources folder from one of the MapXtreme sample applications and paste it into your project where your Web.config and default.aspx files are located. Close and re-open the web page to see the icons. To avoid this manual step. choose option "a" instead.

The Web.config file for the template and samples contains a path to the default location of the installed sample data, as shown below:

```
<configuration>
   <appSettings>
       <add key="MapInfo.Engine.Session.Workspace" value="c:\Program
           Files\MapInfo\MapXtreme\9.x.x\Samples\Data\World.mws" />
   </appSettings>
</configuration>
```

If your application requires multiple maps, they can exist in one or more workspaces, each with a unique MapAlias property. To preload more than one workspace in the Web.config file, use a semicolon between the full path of each workspace.

3. Make sure that MapAlias property of the MapControl is set. The default MapAlias for the MapControl in the template is Map1. If this property is not set, MapXtreme will render the first map from the session which may not be the map you expect.

   To determine the MapAlias, open the .mws file in a text editor and look for the MapDefinition element. The MapAlias is stored as an alias attribute.

   You can also find the MapAlias from Workspace Manager. When you hover the mouse pointer over the map node (top node in the layer list), the alias appears in a ToolTip.

4. Set the MapControlID property for all map tools, LayerControl, LegendControl and custom controls (if any) to point to the appropriate MapControl.

Managing Your Web Controls

An important part to using Web controls is managing them effectively. This section explains how to perform:

- Event Handling
- Error Management
- State Management

Event Handling

Map tool events are handled through client-side JavaScript commands that send a request to the server. On the server-side a command class derived from MapBaseCommand does processing for the tool.

In many command cases, the result of the server-side processing is sent back to the client. This is an image in the case of the MapControl or XML in the case of the LayerControl. Then, only a portion of the web page is updated with the command result via client-side JavaScript (for example, a new map image is displayed after panning).

Error Management

Error handling in the Web controls can be handled in many ways, and it is very specific to the application. Therefore, this section only explains one of the ways of handling errors. Since the response expected by the client side is an image, we can catch the exception using a detailed message, and send the response back with both an image and an error message. As a result, the MapControl will now contain the error message.

The Global.asax files provided with our sample applications demonstrate an example of handling application errors. If the application encounters an error while processing a request for a map image, the Application_Error method creates an image containing an error message, and returns that to the client.

For more information on Error Management, refer to the MSDN site on error handling with ASP.NET.

State Management

The MapInfo.WebControls.StateManager is an abstract class that includes methods SaveState and RestoreState. MapXtreme's map tools call RestoreState and SaveState before and after the processing, respectively. Since state management is application specific, it is your responsibility to implement these methods in a concrete class in the application. This allows you to control what gets restored and saved and how things are restored and saved.

For more information on state management, see Chapter 6 Understanding State Management.

Creating a Custom Tool

To create a custom tool, you can modify or add behavior to a built-in tool or write your own custom commands and tools.

For example, you may wish to modify a Zoom-In tool to zoom-in *and* select a feature with one click. This tool requires a server command class that contains code to zoom-in and perform the selection. Since we provide the source code for all the server command classes, you can simply modify the ZoomIn command class to add the selection code.

If our source code does not provide a starting point for your customization, you must write your own commands and tool classes. The source code is located in your MapXtreme installation under \Samples\MapXtremeWebControlsSourceCode.

As you plan your customization, keep the following MapXtreme tool architecture in mind. A MapXtreme map tool consists of:

- A client-side tool class that inherits from MapInfo.WebControls.WebTool.
- Properties on the tool class that control the behavior of the tool, including:
  - JavaScript that describes the interaction of the tool with the MapControl (click, draw rectangle, etc.)
  - JavaScript that creates the url request for the tool.
- A server-side command class that is derived from MapInfo.WebControls.MapBaseCommand that carries out the desired tool behavior.

Properties for the tool are defined either in the class or on the web page. The custom tool is referenced on the web page by the tool class name.

The code used in the following procedure can be found under the Samples folder under \Web\Features\CustomTools\CustomToolsCS.

To create a custom tool:

1. Drag the generic WebTool from the MapXtreme toolbox onto your web form. You can also use one of the existing map tools if you want to extend the existing behavior.
2. In the WebTool property page, set the properties for MapControlID, InActive/ActiveImageUrl and CursorImageUrl.
3. Set the appropriate ClientInteraction property by selecting from the drop-down list.

   The built-in interactions include mouse operations for clicking and dragging, drawing lines, polygons, rectangles and radii, which will cover the needs of most web application. See the Interaction.js in the MapXtremeWebResources folder in your project.
4. Set the appropriate ClientCommand property by selecting from the drop-down list.

   The built-in client commands for mapping, panning, zooming, etc., create the URL request that is sent to the server. For a description of these commands, see the Command.js in the MapXtremeWebResources folder in your project.

If one of the built-in commands does not meet your needs, either modify the existing Command.js or write your own. The custom command takes the name of the interaction from step 3 as input. See CustomCommand.js in the CustomToolsCS or CustomToolsVB sample for an example of how to get multiple responses from the server with a single click.

5. Register the JavaScript manually in your .aspx page. Insert the following line within the web page body.

```
<script language="javascript" src="CustomCommand.js"
   type="text/javascript"></script>
```

6. Create a new server command class that derives from MapInfo.WebControls.MapBaseCommand. Include code that carries out the behavior that the client command requested. Alternatively, you can extend an existing server command class.

7. In the server command class, assign the name of the server command in the constructor.

```
namespace ToolsSample
{
public class AddPinPointCommand : MapInfo.WebControls.MapBaseCommand
{
   /// <summary>
   /// Constructor for this command, sets the name of the command
   /// </summary>
   /// <remarks>None</remarks>
   public AddPinPointCommand()
   {
      Name = "AddPinPointCommand";
   }
```

8. In the server command class, override the Process() method by adding code that carries out the business logic for the command.

```
      public override void Process()
   {
      // Your code here.....
   }
```

9. In the Page_Load method of the webform1.aspx, add your server command to the collection of commands in the MapControlModel.

```
MapInfo.WebControls.MapControlModel controlModel =
   MapControlModel.SetDefaultModelInSession();
controlModel.Commands.Add(new AddPinPointCommand());
```

10. Add the tool to the Visual Studio toolbox by creating a new assembly. See Using and Distributing Custom Web Controls.

11. Drag and drop the custom tool onto the web form and set the properties to the names of the ClientCommand and ClientInteraction, as set forth in your JavaScript from steps 3 and 4.

12. Set the property for the server Command, as defined in step 6.

13. Register the tag prefix in the web form by specifying the assembly and namespace in which this control exists.

```
<%@ Register TagPrefix="cc1" Namespace="MapInfo.WebControls"
Assembly="MapInfo.WebControls, Version=9.2.0.x, Culture=neutral,
PublicKeyToken=0a9556cc66c0af57" %>
<%@ Page language="c#" Inherits="ToolsSample.WebForm1"
CodeFile="WebForm1.aspx.cs" %>
<%@ Register TagPrefix="cc2" Namespace="CustomizedWebTools" %>
```

If the existing behaviors of the WebTool do not meet your needs, you can write your own server Command class and Javascript to handle client-side commands and interactions. See Adding an InfoTool to a Web Application for an example.

Using and Distributing Custom Web Controls

Once you have created a custom web tool, you must include it in an assembly so that it is available in the Visual Studio toolbox or to distribute to others.

MapXtreme provides the source code for our Web controls so you can learn from them, modify them, and distribute them as you need. The source code for the Web controls is installed in the \Samples\WebControlsSourceCode folder. In order to use and distribute the modified web controls, you must create a new assembly and register it in Visual Studio.

Whether you modify the MapXtreme source code or create your own tool class from scratch, consider the following important factors regarding the MapXtreme web control assembly.

- The assembly name is MapInfo.WebControls.dll and is installed in the global assembly cache. This assembly has a specific version number which is used by our templates and sample applications.
- The controls in the assembly are installed in the toolbox in Visual Studio.
- The assembly has references to MapInfo.CoreEngine.dll, MapInfo.CoreEngine.Wrapper.dll and MapInfo.CoreTypes.dll.
- Resources such as images and scripts are installed in the C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x\MapXtremeWebResources folder.
- The following files are used by the Web controls:

- *.GIF images that represent tool actions (for example, DistanceToolControlActive, DistanceToolControlInactive).
- *.BMP images that represent tool icons (for example, label, selection arrow)
- JavaScript defining tool interactions and behaviors (Interaction.js, Command.js, LayerControl.js, LegendControl.js and Tool.js)
- *.CUR (cursor) files that display images when the mouse is used.

Creating a Web Assembly

To create a custom web control assembly:

1. Copy the customized web control source files to another directory so the original is preserved.
2. Remove the original assembly from the global assembly cache and from the Visual Studio toolbox.

   The assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 or GAC_64, depending on whether you installed the 32-bit or 64-bit MapXtreme.
3. Create a strongly named key file (.snk) using sn - k MapInfo.WebControls.snk and copy this key file to the main project folder (same level as the project files).
4. Change the AssemblyInfo.cs or AssemblyInfo.vb file to reflect the version number of your web assembly.
5. Open the project in Visual Studio, make any changes you need and build the project. The new assembly should be in the bin\Release directory so you can distribute the release version of the assembly.
6. Register your new assembly in the global assembly cache, and in the Visual Studio toolbox using the following syntax as a model. For more information see Global Assembly Cache Tool.

   ```
   gacutil /i MapInfo.WebControls.dll
   ```
7. Write your application using the new assembly. Drag and drop the new controls onto your form from the Visual Studio toolbox.
8. In the installer for your Web application, make sure the new assembly is installed in the global assembly cache. As long as the Web application points to this version of the assembly, it will use the new controls.

Consider the following scenarios:

- You can insert the Web controls project and resources right into your Web application solution. In this case, change the URLs for your resources (scripts and images) to begin with the project root. This prevents you from having to create virtual directories.
- You do not have to use the global assembly cache and a strongly named assembly. You can set the Copy property for the assembly to true, and have the assembly in the bin folder of your application.

Adding an InfoTool to a Web Application

MapXtreme provides a sample application for an InfoTool that can be adapted and used in a web application. Use this tool to capture information at the point a user clicks on the map and get information returned from the web application.

Follow these steps to modify the InfoTool sample.

1. Locate the InfoToolCS or InfoToolVB Web Application in the samples folder (default location C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio20xx\Web\Features\InfoTool\InfoToolVB.
2. Copy the following files to your project folder and add them to your project:
   - CustomCommand.js from the root of the InfoTool folder
   - CustomizedCommands.cs or CustomizedCommands.vb from the \App_Code folder
   - stylesheet.css from the root of the InfoTool folder.
3. Add a PointSelect tool to your Web Form and add the code below in the form's Page_Load method:

   ```
   MapInfo.WebControls.MapControlModel controlModel =
   MapControlModel.SetDefaultModelInSession();
   controlModel.Commands.Add(new CustomWebTools.Info());
   ```

4. Match the properties of the PointSelect tool to those of the InfoTool from the sample application. Set these in the tool's Properties window.

   These properties include: ClientCommand, ClientInteraction, Command, CursorImageURL, InactiveImageURL. and MapControlID.

5. Switch the form to the HTML view and add the following line after the <form> tag.

   ```
   <script language="javascript" src="CustomCommand.js"
   type="text/javascript"></script>
   ```

6. Add a like the one in the sample application to hold the information retrieved by the tool.

```
<div id="Info" class="infoDiv">
            Div element to display selected feature information in html
table.</div>
```

7. Build the Web Application.

ASP.NET AJAX and MapXtreme Web Applications

MapXtreme's web controls and tools include JavaScript which provides an efficient request/response cycle for web applications. Each time a map tool is used, JavaScript interactions and commands carry out the operation without requiring a full-page postback to the client. Typically, only the map image is refreshed.

Microsoft's ASP.NET AJAX technology takes this behavior further, by integrating scripting libraries with the ASP.NET Framework. The principal controls are the UpdatePanel, a container for server controls that are frequently refreshed, and the ScriptManager, which manages the scripting activities for the web page.

MapXtreme provides a sample application that demonstrates how to use Microsoft's ASP.NET AJAX controls in a MapXtreme web mapping application. The sample is located in ..\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio20xx\Web\Features.

ⓘ The AJAXDemo sample requires that the Microsoft ASP.NET AJAX Extensions 1.0 or later be installed on your system.

The following section provides the steps to add AJAX controls to your MapXtreme application.

Before proceeding, as an exercise in understanding AJAX, it's a good idea to create a web application using the "ASP.NET AJAX-Enabled Web Site" template that is provided with the AJAX extensions. Examine its Web.config file; the httpHandlers section and the httpModules section contain settings that you may need to copy into your application.

You should also familiarize yourself with the AJAXDemo sample web application. The steps that follow refer to JavaScript code and Web.config settings taken from this sample.

Adding ASP.NET AJAX Controls to a MapXtreme Web Application

To add ASP.NET AJAX controls to an existing MapXtreme web application:

1. Open your web form in Visual Studio's Design mode.
2. From the "AJAX Extensions" section of the Visual Studio Toolbox, drag a ScriptManager control onto the form. (It does not matter where you place the ScriptManager, as it will not be visible at run-time.)
3. Drag an AJAX UpdatePanel control onto your form.
4. Move standard controls, such as Button controls, inside the UpdatePanel, to prevent the Button from causing a full-page update.

ⓘ Do not move MapXtreme controls, such as the MapControl or the LayerControl, inside the UpdatePanel. For a detailed example, see the AJAXDemo sample application.

---

5. Open your application's Web.config file, and locate the httpHandlers section. Depending on the contents of your web application, the httpHandlers section might contain just one or two entries -one for MapController.ashx, and, if your application contains the LayerControl, one for LayerController.ashx.

```
<httpHandlers>
    <add verb="*" path="MapController.ashx" . . .
   <add verb="*" path="LayerController.ashx" . . .
</httpHandlers>
```

6. Open the Web.config file from the AJAXDemo sample application, and locate its httpHandlers section, which contains additional entries needed by ASP.NET AJAX:

```
<httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx" . . .
    <add verb="*" path="*_AppService.axd" . . .
    <add verb="GET,HEAD" path="ScriptResource.axd" . . .
    <add verb="*" path="MapController.ashx" . . .
   <add verb="*" path="LayerController.ashx" . . .
</httpHandlers>
```

   If any of those first four httpHandlers entries are missing from your Web.config file, copy the missing entries from the AJAXDemo Web.config file, and paste them into your Web.config file. (It is not necessary to copy the LayerController.ashx entry; if you place the LayerControl on your page in Designer mode, the LayerController.ashx entry will be generated automatically.)

7. Locate the httpModules section of your Web.config file. The httpModules section probably already contains one entry, for MapInfo.Engine.WebSessionActivator. Copy the "ScriptModule" entry from the AJAXDemo Web.config file, so that your httpModules section resembles this:

```
<httpModules>
 <add name="ScriptModule" type="System.Web.Handlers.ScriptModule. . .
 <add type="MapInfo.Engine.WebSessionActivator. . .
</httpModules>
```

If the controls in your UpdatePanel affect the map in some way, you will need to add JavaScript to the page to force the map image to update. The AJAXDemo sample application contains sample JavaScript that demonstrates how you can update the map image whenever the designated UpdatePanels cause a page update.

8. Open the AJAXDemo application's MapForm.aspx page in Source mode. Copy the <script> block and paste it into your aspx page. NOTE: You must paste the <script> block after the ScriptManager tag, because the script makes use of objects provided by the ScriptManager.
9. In the <script> block, delete the DisplayEventInfo function and any calls to it. The DisplayEventInfo function is a debugging tool for the AJAXDemo application; it is not needed in other applications.
10. If you renamed your UpdatePanel, edit the <script> block to use the new UpdatePanel name. (The UpdatePanel name is passed as the second parameter to the TargetPanelWasUpdated function; the default name is "UpdatePanel1".)
11. If you use more than one UpdatePanel in your application, but you do not want all of the UpdatePanels to affect the map, then you should set each UpdatePanel's UpdateMode property to Conditional. For details, see the ReadMe file provided with the AJAXDemo sample application.

For additional details about the Web.config settings required by ASP.NET AJAX extensions, please consult Microsoft's ASP.NET AJAX documentation.

MapXtreme Tile Handler

MapXtreme provides a REST-based tile handler and public URLs that are used to request map tiles and information. With a REST-based handler, you can embed all the arguments of your map request in a single URL.

Map tiles are becoming essential today in web mapping as they can be pre-rendered and stored, awaiting requests from a user. You can design a tile server that stores base maps as static images since these do not need to be updated during the user’s session. For dynamically changing data that contain information exclusive to the user, such as information returned from a query, these maps are generated on the fly. Because they are tiled, only the tiles that would be visible in the map window (based on the tile size, map window size and zoom level) would be returned.

For an example of a web application that takes advantage of the MapXtreme Tile Handler, see the MapXtreme Tile Handler example located on the MapInfo Developers Code Exchange.

MapXtreme provides support for:

- requesting tiles by their row and column positions in the map
- requesting a list of available maps
- requesting a description of a map
- specifying the cache instructions for better tiling performance

The MapXtreme Tile Handler API is included under the MapInfo.WebControls.Tiling namespace. See the Developer Reference for details. Source code for the Tile Handler is provided in the \Samples\WebControlsSourceCode folder.

Using the MapXtreme Tile Handler

In its simplest form, to access the MapXtreme Tile Handler and request map tiles and information from a tile server:

1. Modify your web.config file to point to the handler, as follows.

```
<httpHandlers>
   <add verb="*" path="TileServer/*"
type="MapInfo.WebControls.Tiling.TileHandler" />
      <add verb="*" path="TileServer/*/*"
type="MapInfo.WebControls.Tiling.TileHandler" />
      <add verb="*" path="TileServer/*/*/*"
type="MapInfo.WebControls.Tiling.TileHandler" />
      <add verb="*" path="TileServer/*/*/*/*"
type="MapInfo.WebControls.Tiling.TileHandler" />
</httpHandlers>
```

2. Provide a workspace (.MWS) that contains multiple maps.

   For example, for a coverage locator application where the user wants to know if their location is inside or outside your coverage (cell network, trade area, school districts, etc.) include:

   - a base map layer to server as a background and reference map.
   - an overlay map containing reference points and lines such as point locations and road networks.
   - a coverage layer containing your coverages

3. Request a list of available maps using the following URL format.

```
http://server/TileServer/maplist.{ext}
```

where `ext` is an extension denoting the format of the information returned (current support for JSON only)

This query will return a JSON object (JavaScript Object Notation) with the following format:

```
{
   [
      "Map1Alias",
         "Map2Alias",
         "Map3Alias"
   ]
}
```

4. Using the listed map names from step 3, request more information about a map using the following URL format:

   `http://server/TileServer/{mapname}/description.{ext}`

   where `mapname` is the name of the map on the Tile Server to get metadata about. The value is case-insensitive.

   `ext` is an extension denoting the format of the information returned (current support for JSON only)

   This returns the metadata of a specified map.

```
{
   "numberOfLevels": 20,
"coordSys": "epsg:41001",
"description": "Map of the World",
"name": "World",
"tileWidth": 256,
"tileHeight": 256,
"bounds": {
   "minX": -3.756380109844111E7,
   "minY": -4.497601034176671E7,
   "maxX": 3.773376466605809E7,
   "maxY": 4.49763662218225E7
}
"outputTypes":
[
   "png"
]
}
```

5. With information collected from steps 3 and 4, request the map by providing all the necessary arguments in the URL (explained below):

   `http://server/webapp/TileServer/{mapname}/{level}/{x;y}/tile.{ext}`

   For example:

```
http://<server>/<mywebapp>/TileServer/WorldOverlay/3/0;0/tile.png
```

This will request the upper left tile of a WorldOverlay map consisting of 64 tiles.

The arguments are explained in the table below.

Argument	Description
`server`	Your web server
`webapp`	The name of the web application running on the server.
`TileServer`	The path to your instance of the MapTiling handler. This must match the beginning of the "path" entries in the "httpHandlers" section of your web.config. See above.
`mapname`	The name of the map on the Tile Server.
`level`	The level of requested tiles. Zero-based.
`x;y`	The x and y ordinates of the requested tile (zero-based). For example, at level 3 the map consists of 64 tiles in 8 rows and 8 columns. The x and y arguments for the upper left tile would be 0;0. For the lower left tile, it would be 0;7.
`ext`	An extension denoting the format of the tile (e.g., .gif, png). Must match a supported format.

Caching

The MapXtreme Tile Handler supports caching of frequently used tiles so that application performance does not suffer. Caching instructions are included in the web.config file following Microsoft's.NET Framework HttpCacheability.

Five types of caching (plus a no cache option) are supported. These enumerated values are used to set the Cache-Control HTTP header.

NoCache	Sets the Cache-Control: no-cache header. Without a field name, the directive applies to the entire request and a shared (proxy server) cache must force a successful revalidation with the origin Web server before satisfying the request. With a field name, the directive applies only to the named field; the rest of the response may be supplied from a shared cache.
Private	Default value. Sets Cache-Control: private to specify that the response is cacheable only on the client and not by shared (proxy server) caches.
Public	Sets Cache-Control: public to specify that the response is cacheable by clients and shared (proxy) caches.
Server	Specifies that the response is cached only at the origin server. Similar to the NoCache option. Clients receive a Cache-Control: no-cache directive but the document is cached on the origin server. Equivalent to ServerAndNoCache.
ServerAndNoCache	Applies the settings of both Server and NoCache to indicate that the content is cached at the server but all others are explicitly denied the ability to cache the response.
ServerAndPrivate	Indicates that the response is cached at the server and at the client but nowhere else. Proxy servers are not allowed to cache the response.

To specify a cache option, modify your web.config file to point to the key MapInfo.Engine.Session.Cacheability, as follows.

```
<appSettings>
   <add key="MapInfo.Engine.Session.Cacheability" value="private"></add>
</appSettings>
```

Caching Expiration

You can also set an expiration date for your cached tiles. Set an expiration date when the data in the tile needs to be refreshed. By expiring tiles, users of your application will receive only the most up-to-date information displayed.

To set the cache expiration, add a key to the web.config file, as follows:

```
<appSettings>
    <add key="MapInfo.Engine.Session.CacheExpires" value="1/1/2016"></add>
</appSettings>
```

The value can be any string that can be successfully parsed by Microsoft's DateTime.Parse(String) method.

Note that Microsoft limits the expiration date to one year. Any date beyond one year will not be honored.

HTML/XHTML Validation Issues

If you create a MapXtreme web application and run the resulting HTML through a validator service, you might see the following validation error, depending on what version DOCTYPE tag you use:

```
value of attribute "ID" invalid.  "_" cannot start name
```

This validation error is in reference to a hidden field with the attribute

```
id="__VIEWSTATE"
```

The id attribute that causes this validation error is not output by MapXtreme; it is an attribute that is output by ASP.NET.

To resolve this validation error, you may need to replace the DOCTYPE tag on your .aspx page. Specifically, if you update your DOCTYPE tag to an XHTML DOCTYPE tag, the resulting page will validate, even with the id attribute shown above. (ASP.NET will wrap the offending tag in a DIV tag, all of which will validate against the XHTML DOCTYPE.) As an example, you might use the same DOCTYPE tag that is generated for you when you create a new Web Application from Visual Studio's ASP.NET template:

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

Migrating Post-back Web Controls to JavaScript Web Controls

The process of migrating your pre-v6.5 Web controls[1] to the MapXtreme Web controls in versions 6.6 and higher is not automatic. Every application and migration process is different and depends on how tightly coupled the application is with the functionality and design of the older Web controls.

The recommended migration process is to take a phased approach. In some cases code restructuring might have to be done. The following processes need to be considered while migrating your Web controls:

- Loading Data
- Replacing Controls
- State and Event Management

Loading Data

The only way to load data with MapXtreme Web controls is by using a preloaded workspace. The MapControl points to the MapAlias in the workspace and tries to display the map, not load the map. The preloaded workspace is specified in the Web.config file. See What Should the Initial Map Look Like?.

In the case where the MapControl is used to display multiple map images, set up the data so that all maps are in one location, and change the MapAlias of the MapControl to choose the maps.

Replacing Controls

Replacing controls can be done in various ways, either by deleting the older Web controls from the form and dragging and dropping the new Web controls on the form, or by creating new Web forms, dragging and dropping the new controls and then start adding functionality from the old forms to the new forms. Make sure to set the appropriate MapAlias for the MapControl and set the appropriate MapControlID for all dependent tools and the LayerControl.

1. In MapXtreme releases prior to version 6.5, the web controls required a postback page that called Page_Load and Page_Unload every time a tool was used. These controls have been replaced by more efficient JavaScript-enabled partial-page update controls. We provide the postback controls for backward compatibility, but they have been marked Obsolete. Information on these controls is contained in the MapInfo.Web.UI.WebControls namespace. For the JavaScript controls API, see the MapInfo.WebControls namespace.

State and Event Management

In most cases, the older method of state management is to restore state in Page_Load and save state in Page_Unload. This code must be moved to the new class which derives from the StateManager into the RestoreState and SaveState methods.

If your web application handles state manually, then you must implement the StateManager and put this in the ASP.NET session. It is a best practice with MapXtreme to manually handle state so that you are only restoring the information needed. See Chapter 6 Understanding State Management.

In the case where Server.Transfer or Response.Redirect is used in a web page, the session is no longer new. You must put the StateManager in the ASP.NET session if it previously does not exist.

Specialized Topics for Web Controls

The following sections refer to special usage of MapXtreme Web controls:

- Using Web Controls in Frames
- Using the MapControl in Table Cells
- Web Control Localization

Using Web Controls in Frames

The MapXtreme Web controls work in frames. For information on how to create frames, framesets, and assign pages to frames see the Visual Studio documentation.

When using Web controls in frames, remember that a frame points to a web page and a frameset contains one or more frames. Take the following scenario: A page with a MapControl and other pages with tools or a LayerControl that depend upon the MapControl. All of the pages form a frameset.

Given the above scenario, the following rules apply:

- The MapControlID must be manually entered into the properties for the dependent controls. If there is a MapControl on the same page as the dependent control with the same ID, the tools and LayerControl will pick that one.
- Since the frame pages render in a particular order, the MapXtreme session will not be a new session for frames rendered after the first frame. To ensure the StateManager is in place, implement the following code. Ensure there is a StateManager class registered regardless of what page is loaded, The following code is executed before RestoreState is called.

```
' If the StateManager doesn't exist in the session put it in, else get it.
If StateManager.GetStateManagerFromSession() Is Nothing Then
        StateManager.PutStateManagerInSession(New AppStateManager())
End If
```

Using the MapControl in Table Cells

Due to HTML behavior, as soon as an element is dragged out of another element, it is resized to a basic size (mainly 0). If you put a MapControl in a table cell you may have a problem when an element does not have an absolute width and height in the HTML. When the element is dragged out of the cell, it will collapse and therefore the cell size becomes 0.

To solve this issue, set the MapControl height and width, explicitly. The following sample shows how to set the size in the HTML:

```
<table style="Z-INDEX: 101; LEFT: 32px; POSITION: absolute; TOP: 64px"
borderColor="#ff00ff" border="1">
 <TR bordercolor="#ff3366">
  <TD bordercolor="#0066ff">
   <cc1:mapcontrol id="Mapcontrol2" runat="server" Width="300px"
Height="300px"></cc1:mapcontrol>
  </TD>
 </TR>
 <TR>
  <TD>
   <cc1:pantool id="Pantool2" runat="server"
MapControlID="MapControl2"></cc1:pantool>
   <cc1:zoomintool id="ZoomInTool1" runat="server"
MapControlID="Mapcontrol2"></cc1:zoomintool>
   <cc1:zoomouttool id="ZoomOutTool1" runat="server"
MapControlID="Mapcontrol2"></cc1:zoomouttool>
  </TD>
 </TR>
</table>
```

Web Control Localization

MapXtreme provides a Visual Studio solution for developers who wish to translate text strings associated with the web controls. This "localization kit" contains resource projects for all runtime components of MapXtreme. No design-time resources are included.

Included in each project are the English resource strings for translating and a strong named key (.snk) file. that will compile into an assembly that can be incorporated into your MapXtreme application. The MapXtreme web controls are contained in the project called MapInfo.WebControls.resources.

See Appendix K: Localization Kit for instructions on how to build a satellite assembly from localized web control resources.

6

6 – Understanding State Management

This chapter describes concepts and best practices for state management—a subject of great importance when writing web applications.

In this chapter:

Overview

State management is a key consideration in the design and implementation of a MapXtreme web application. How the changes a web site visitor makes when interacting with your web application is central to a successful user experience and to building a scalable, high performing application.

Before getting too far into this subject, the following section is a "must read" for what follows in this chapter. Understanding the difference between MapXtreme Session and HTTP Session, user state and application state, clean and dirty MapXtreme Session objects, and more, will better prepare you for planning and building your web application successfully right from the beginning.

Terminology

MapXtreme Session - The MapInfo.Engine.Session object that holds the Catalog, MapFactory, and CoordSysFactory. The user interacts with an instance of a MapXtreme Session object.

HTTP Session - The System.Web.HttpSessionState object where the user's changes are saved between requests. These changes are saved and restored for each request. The user's changes are known as the user state.

Browser Session -The period of time a unique user interacts with the web application. This is also referred to as the ASP.NET Session.

InProc Development model - A web application development model in which each user has his own instance of the MapXtreme Session object. Any changes the user makes during the browser session do not affect other users. The entire MapXtreme web application is stored in memory with the current application state and is associated with this individual user. This model is useful for small, light-use web applications where the number of users is known, such as a department’s intranet. Contrast this with the more scalable Pooled Development model in which users share the MapXtreme Session and system resources. See InProc Development Model on page 109.

Pooled Development model - A model in which multiple MapXtreme Session instances are available from a COM+ pool associated with the web application and are activated to serve web requests. Each of the pooled MapXtreme Session objects is used to handle requests from multiple web users. This model is more complicated than the InProc model, since your application must manage the state of the MapXtreme Session for each user. However, this is a more efficient use of system resources. Use this model to build scalable applications. See State Management For Pooled Objects on page 113.

State Management - A general term in web application development that deals with saving and restoring information from a browser session.

Background map - The initial map(s) that is pre-loaded with the web application as defined in the Web.config file. This map contains reference layers, such as street networks and postal boundaries, as well as application-specific data such as store locations or cell towers. This base workspace will be pre-loaded in MapXtreme Session instances and will be available unless the application allows the user to change the base maps. See What State Management Options are Available? on page 106.

Application State - An ASP.NET-defined mechanism for storing state information in memory that applies to all users and sessions of a web application. See the ASP.NET Application State Overview at http://msdn2.microsoft.com/en-us/library/ms178594.aspx. See also Handling Initial Requests on page 124

User state - The saved state of the MapXtreme Session and application state between user requests. Any changes the user makes to the MapXtreme Session or application must be saved to the HTTP Session. Changes can be as simple as re-centering the map, or as complicated as a query to create a thematic map.

Beginning state - The condition of a pooled MapXtreme Session instance when the user accesses it. There are four possible beginning states:

- New user to site, the available MapXtreme Session instance from the pool is clean, with the background map in its initial state.
- New user to site, the available MapXtreme Session instance from the pool is dirty (map has changes from another user)
- Returning user, the available MapXtreme Session instance from the pool is clean.
- Returning user, the available MapXtreme Session instance from the pool is dirty (map has changes from this or another user)

MapInfo.Engine.Session.State - A MapXtreme-defined mechanism that determines whether the MapXtreme Session state will be saved automatically or manually. The automatic Session state means the entire MapXtreme Session is saved to the HTTP Session. Automatic Session state is set in the application's Web.config file in the key:

```
<add key="MapInfo.Engine.Session.State" value="HttpSessionState" />
```

Manual Session state means that the developer is responsible for saving state that changes from one user to another. This is the mechanism to use when building scalable, pooled applications.

Manual Session state is set in the application's Web.config file in the key:

```
<add key="MapInfo.Engine.Session.State" value="Manual" />
```

StateManager - A MapXtreme class with methods and properties to help with saving and restoring user state in a pooled web application. When MapInfo.Engine.Session.State is set to Manual in the Web.config file, the application must provide a StateManager class that implements SaveState and RestoreState methods. See Implementing a StateManager on page 121.

sessionState - A standard ASP.NET Web.config element for configuring which storage mechanism is used for saving user state. Three types:

- InProc - user state is stored in memory for the lifetime of the ASP.NET Session.
- StateServer - user state is saved on the server so user changes can be retrieved at a later time.
- SQLServer - user state is saved to an SQL Server database for later access.

What is State Management?

Many types of web applications need to perform state management—the process of saving and restoring information about what state, or condition, each user's session is in. For example, when a retail web application provides a shopping cart, the application must remember the state of each user's shopping cart.

In a web mapping application, if your application allows the user to click the map to zoom in, your application needs to remember that one user might be zoomed in on Europe while another user is zoomed in on Australia. The code that remembers each user's map state is referred to as state management code.

What State Management Options are Available?

MapXtreme provides you with these options for how you manage state:

- You can use InProc state management, which is easy to code; however, it is not appropriate for all applications because it taxes server resources.
- You can use a Pooled architecture with automatic state management. This model is easy to code, but the resulting application may not be fast enough, depending on your needs.
- You can use a Pooled architecture with manual state management. This model requires more coding on your part. You must write code to save and restore the appropriate MapXtreme Session objects that comprise the user's state. However, this model produces the most scalable applications, so it is the best choice for applications that have large numbers of users.

Thus, there are trade-offs associated with each option. You must consider many factors when designing web applications using MapXtreme. Some factors are MapXtreme design decisions and others are Microsoft technology design patterns. This chapter will walk you through some of the decisions you must make and show you how to make the correct choices for your type of application.

It is strongly recommended that you think about state management before you write your web application. The type of state management that you choose will affect how your write your application. If you write a web application, and then later decide to adopt a different type of state management, you may need to rewrite significant parts of your application.

Questions to Ask Before Writing Your Application

If you want to create a simple MapXtreme web application, you can do so quickly and easily - just create a new web application using the Visual Studio template provided with MapXtreme ("MapXtreme Web Application").

However, for a more complicated web application, before you invest significant time and effort coding the business logic for your web application, there are several questions that you should think about, as the answers to these questions will help you decide which type of state management is right for you.

How Many Users Will Access Your Application?

The most important question you can answer is: How many people will access this site?

The earlier you identify the number of users your web application will have, the better off you are. If you develop and test an application with a very small number of users (as a pilot project perhaps), then later on you might discover that your application does not perform well when there are many users. In that situation, you may find that you need to change your application architecture - a change which might require you to rewrite your application. It is far better to plan ahead and anticipate your user load in advance, so that you can use the appropriate state management architecture from the beginning.

Known Number of Users

Perhaps you are creating an intranet site where you have a finite, defined list of potential users. For example, you might be creating a web site to be used by a specific department in your organization. Perhaps you know the names of everyone who will access your site.

Given such a well-defined and finite number of concurrent users, you can consider using the InProc, non-pooled development model. It is the easiest model to code, but not an appropriate choice for all applications, because it places substantial demands on web

server resources. It creates one MapXtreme session instance for each concurrent user. However, if your pool of users is finite and well-defined, these server requirements might not be a problem.

Unknown Number of Users

The other development model is when you do not know the number of users who will want access to the site. Perhaps you are hoping or expecting that over time your web site will attract more and more visitors. In this case, the InProc, non-pooled model would be inappropriate, and you should instead plan on using a pooled development model where you capture each user's state manually. In the pooled development model, you will create a finite number of MapXtreme sessions, and each user request will be serviced by re-using one of the objects from the pool.

The pooled architecture lets you develop your application for scalability. You may later need to add more servers to handle additional load; thus your application must understand how to save the users current state and when and how to apply it when the next request comes in to the server. This may occur on the same server or a different server in your farm. This scalability leads to many options and choices you can make about how your system will store state, access data, and respond to multiple requests. This is why it is important to plan out your application and create a strong architecture that supports distributed applications.

What Should the Initial Map Look Like?

When you develop an application, you must decide what information remains the same for all users. This includes map layers, labels, titles, and color-coding or other types of thematic shading. This is known as the background map.

To set up your background map, run the MapXtreme Workspace Manager (a desktop application installed with MapXtreme), and save your map as a workspace file (.mws file). See Chapter 27 Workspace Manager.

Once you have created an .mws file, edit your web application's Web.config file to include a reference to your .mws file. The following Web.config excerpt shows the syntax:

```
<configuration>
  <appSettings>
    <add key="MapInfo.Engine.Session.Workspace"
value="C:\MIDATA\EvalData\WorldDetail\World_Detail.mws" />
  </appSettings>
</configuration>
```

This tag instructs the MapXtreme Session to load this workspace whenever a new instance of the Session is created.

How Will Users be Allowed to Modify the Map?

Most web applications allow the user to click, drag, or perform other actions that modify the map in some way. In your application, you might consider the following.

Will you allow your users to:

- Zoom in or out or re-center the map?
- Select some features on the map (perhaps by clicking directly on the map)?
- Turn map elements on or off (such as clearing a checkbox to turn off street display)?
- Create and/or modify thematic shading (such as color-coding on the map to show data)?
- Place map annotations, such as symbol markers, on the map?

All of these operations can be supported in a mapping application. However, if you decide to implement a web application using a pooled model, you should be aware that pooled applications need to include code that carefully saves and restores all changes the user has made to the map.

For example, if your web application allows users to place annotations on the map, then you will need to write code to save each user's custom annotations, and then write code to restore that user's custom annotations with each subsequent request.

If your application uses the pooled model with manual state management, then your state management code will grow more complex as you add more and more features that allow the user to modify the map. As you consider features that allow the user to modify the map in various ways, remember to set aside time for the development and testing of your state management code as well.

InProc Development Model

If you use the InProc development model, you code your application much like you would code a desktop application. With the InProc model, there is one MapXtreme Session object for each user; this means you will have the overhead of each user having their own process space, and resources are not shared. (The MapXtreme Session is the holder of the Catalog, MapFactory, CoordSysFactory, and other MapXtreme objects.)

The InProc model is a simple model for development purposes. Because each user has their own MapXtreme Session, changes can be made within a Session without disrupting any other users' Session.

For example, if the user clicks to zoom in on the map, your application code can simply modify the map object's Zoom property, without worrying about whether the change in the zoom level will adversely affect other users. Thus it is easy to code an InProc web application.

You might find it helpful to think of a web application as being analogous to a restaurant. Web applications need to service simultaneous requests from multiple users, just as restaurants need waiters to take food orders from multiple tables.

The InProc model is analogous to a restaurant that hires one waiter per table. It would be expensive for a restaurant to employ one waiter per table. However, if a restaurant was able to afford hiring one waiter per table, then doing so would certainly provide each table with excellent service. Also, having one waiter per table would make each waiter's job simpler. Since each individual waiter would be servicing one table, the waiter would not need to waste any time or energy remembering which order went to which table.

Pros and Cons of the InProc Development Model

If you choose the InProc model, it will be easier for you to write your web application, because you will not need to provide any complex code for saving and restoring each user's state with each subsequent request.

However, the InProc model is not scalable, because it requires having a dedicated MapXtreme Session for each concurrent user. For thousands of users to use your site concurrently, the application would need to maintain thousands of MapXtreme sessions on the server, which taxes your server’s resources. If you anticipate a large number of concurrent users, then the InProc model is not an appropriate choice for your application.

InProc Management: A Walk-Through

An example helps illustrate how MapXtreme Sessions are created and used when an application uses the InProc model. The following describes, in simplified terms, the sequence of events:

1. A user launches a browser and goes to your mapping application web site.
2. A new MapXtreme Session is created on your server. It will be used to service all requests from this user during this ASP.NET session.
3. The workspace file(s) specified in your Web.config file is loaded. As a result, your MapXtreme Session's MapFactory has one or more Map objects. For the sake of this example, let us assume that the workspace contains one map, which initially shows the entire world.

4. The web page's HTML is rendered and returned to the client browser. Part of the page is an HTML img tag, which requests an image of the map from the server. On the server, objects from this user's MapXtreme Session—a Map object, in particular—are used to render the map image, which is returned to the user's browser.
5. The user selects the Zoom In tool, and then draws a marquee box around Australia, to zoom in on Australia.
6. On the server, the application modifies the Map object's Zoom property so that it is zoomed into Australia. The application renders a new map image and streams the image to the client browser.

   The Map object is now in a different state than it was initially-its Center and Zoom properties have changed (to show Australia rather than showing the entire world).
7. The user clicks on the Select tool, and then clicks on the portion of the map that shows Australia.
8. On the server, the application performs a selection: to select the Australia region. A new map image is rendered, to display the new selection.

   There are now two ways the Map's status has changed from its original state: Its center and zoom changed (in step 6), and now there is a Selection.
9. Elsewhere, a second user launches a browser and goes to your mapping application web site. A new MapXtreme Session is created on your server for the second user. This session contains a map, which shows the entire world. This Map object is separate from the first user's Map.

Note that the second user sees a map of the world, rather than a map that is zoomed in on Australia. Only the first user's map is zoomed in on Australia. A second MapXtreme Session (with its own map) was created to service the requests from the second user. Thus the second user will not see the same map as the first user.

Configuring an Application to Use the InProc Development Model

The Web.config file for your web application holds the settings to control the application model. For the InProc model the Web.config file settings are:

```
<!--Use this setting to turn Session pooling on/off (true/false)-->
    <add key="MapInfo.Engine.Session.Pooled" value="false" />

<!--Use this setting to save Session state automatically (HttpSessionState) or
manually (Manual)-->
    <add key="MapInfo.Engine.Session.State" value="HttpSessionState" />
```

```
<sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424"
sqlConnectionString="data source=127.0.0.1;userid=sa;password="
cookieless="false" timeout="20" />
```

These settings specify that the MapInfo.Engine.Session object is not pooled, and the state of the MapXtreme Session object is automatically stored in the user's HTTP Session (key value = HttpSessionState).

In the sessionState element, we have set the mode attribute to InProc to specify that everything is saved in process. InProc is the default mode if sessionState is not specified.

The lifetime of the MapXtreme Session instance depends on the Session state configuration. The MapXtreme Session instance is cached in memory for the lifetime of your ASP.NET session when you use server-based Session State and the ASP.NET sessionState element in your Web.config file is set to InProc.

If you use any other type of Session State configuration (for example, client-based or server-based with sessionState element set to StateServer or SQLServer), the MapXtreme Session instance is created and disposed for each ASP.NET request. However, this has performance implications and should be avoided as a development model. If you follow the InProc model, sessionState should be set to InProc.

Once you decide to follow this option of an InProc session, each user who accesses your web site will be given a copy of the MapXtreme Session object and all it contains. Obviously if the number of users grows, then so will the memory footprint.

Your application will also load the workspace defined in the Web.config each time a MapXtreme Session is created. For the InProc model, this means a new MapXtreme Session is created and the workspace is loaded when the user first visits the site.

Using the MapXtreme Template with the InProc Development Model

If you created a Web Site using the " template, be aware that it does not follow the InProc model.

To create an application with the template that uses InProc:

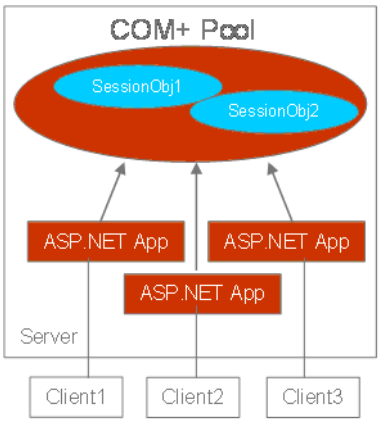
1. Modify the Web.config file as shown in Configuring an Application to Use the InProc Development Model on page 111.
2. Remove the application's Page_Load and Page_Unload methods from MapForm1.aspx.vb or MapForm1.aspx.cs or comment out the manual state-management code that was placed automatically by the MapXtreme template.

State Management For Pooled Objects

State management in a pooled application is a development model that is designed to make applications scalable. If you anticipate that your application will have a large and/or a continuously growing number of users, the pooled model is a better choice for you than the InProc model.

What is Pooling?

In the pooled model, your application creates a finite number of MapXtreme Session objects on the server. Each one of those object instances is then shared within the application and re-used; each MapXtreme Session handles requests from multiple users, as illustrated in the following diagram.



This diagram represents an application that has a COM+ pool of two MapXtreme Session objects, being accessed by three users (web clients). Whenever a client submits a request, the request will be serviced by one of the two MapXtreme Session objects.

In this model, no client has its own dedicated MapXtreme Session. Instead, MapXtreme Session objects are shared and re-used. The following sequence of events is possible:

1. Client1 might request a map zoomed in on Africa. This request might be handled using SessionObj1.
2. Client3 might request a map zoomed in on Australia. This request might also be handled using SessionObj1. In other words, each Session handles requests from multiple clients. No one client "owns" the MapXtreme Session; they are all shared.

3. Client1 might submit a request to pan the map to show Europe. This request might be handled using SessionObj2. In other words, there is no guarantee that every request submitted by a particular client will be serviced by the same MapXtreme Session, or the same Map object.

Pros and Cons of Pooling

Pooling helps reduce the number of concurrent MapXtreme Session instances, and thereby reduces the overhead of creating each object from scratch. When an object is needed, it is pulled from the pool. When the object is not needed anymore, it is placed back into the pool to await the next request. Pooling in MapXtreme means that multiple MapXtreme Session objects can already be created and loaded with a background map and available when a web request is received.

Pooling helps you optimize the server's resources in the following ways:

- Improves overall response time for web applications by running several ASP.NET requests concurrently.
- Conserves resources by reducing the number of concurrent MapXtreme Session instances.
- Maximizes CPU utilization by reducing the number of requests running at the same time (and reducing thread context switching). This is particularly useful when requests are CPU intensive (for example, Map image exports). The general recommendation is to have a pool size of 1-2 session instances per CPU.

However, a pooled application can make an application more complicated. MapXtreme Sessions (and their maps) are shared and re-used; therefore, you need to take steps to ensure that each Map is re-set to an appropriate state each time your application handles a client request.

For example, suppose a client requests a map of Germany. You cannot simply fetch a MapXtreme Session from the pool and then use its Map to render an image, because there is no guarantee that particular map is zoomed in on Germany. The Map that services the request must be set to an appropriate, known state. As the application developer, you must manage the state of the object to ensure that the object is in an appropriate state before you use it to service the current request.

Saving State for Pooled Applications

The pooled development model offers you two options for managing the state of your pooled objects. You can use automatic state management, and have MapXtreme take care of the current user state, or you can use manual state management and control the changes that must be saved and restored for each user.

Automatic State Management

This scenario indicates that the state of the MapXtreme Session object will be saved automatically, to the session mechanism defined in the Web.config (either StateServer or SQLServer). The application programmer does not have to programmatically save any of the current user state information, such as layers, thematics, open tables, or current zoom. The entire state of the MapXtreme Session object will be saved.

This option saves system resources compared to the InProc model, since you do not have a MapXtreme Session for each user and all users will share pooled MapXtreme Session objects. The drawback is that the entire MapXtreme Session will be serialized to the currently defined state mechanism. In general, the MapXtreme Session object retrieves all objects in its factories and serializes them to the state mechanism. This includes all maps and their included layers, all open table definitions, and any projections loaded. If the application is using automatic state management, the MapXtreme Session object does not know about individual user's state and stores all available information. This includes layers that do not change from one user to another, which is the background map. This can be a time consuming process and can cause this type of application to perform more slowly than an InProc model. For information on serialization see Serialization and Persistence in Chapter 9 on page 219.

Manual State Management

This configuration means the MapXtreme Session instance will not automatically save any instance data. It is up to the application programmer to include code in the application to save and restore the state for each user. The MapXtreme Session object will not store any of its state to the currently defined state mechanism.

This configuration provides for the greatest gain in performance and scalability. You are using pooled MapXtreme Session instances that do not save any state, and these pooled objects are being returned to the COM+ pool "dirty". By this we mean the pooled object has any modifications to the map done by the current user and your application is responsible for cleaning up the state and setting it correctly for the next user. You can choose to clean up at the beginning of your web application request or the end. Implementing a StateManager on page 121.

The combination of using pooled MapXtreme Sessions and managing user state manually makes this a complicated development model. You cannot tell which MapXtreme Session will be used for a particular request; however, at the same time you need to satisfy the user's request with a MapXtreme Session that is specific to that user. For this reason, your application must be aware of the beginning state of the pooled MapXtreme Session.

There are four possible beginning states that your application must handle. Each situation involves a user condition and a MapXtreme Session state.

User is...	MapXtreme Session in the Pool is....
New to the site	Clean *
New to the site	Dirty†
Returning to the site	Clean *
Returning to the site	Dirty†

* MapXtreme Session and its background map are in their initial state.
† MapXtreme Session and/or its background map contain another user's changes.

Manual State Management: A Walk-Through

An example helps to illustrate how pooled MapXtreme Sessions are managed when an application uses the manual state management model. The following describes, in simplified terms, a possible sequence of events:

1. A user launches a browser and goes to your mapping application web site.
2. Since this application uses pooling, a MapXtreme Session instance is retrieved from the pool.
3. The RestoreState method is called from Page_Load.

   If the MapXtreme Session is "dirty," with some user's changes, the application sets the map back to the desired beginning state; see the RestoreDefaultState method in the AppStateManager class.

   For this example, let us assume that the beginning state shows a map of the entire world.
4. The web page's HTML is rendered and returned to the client browser. Part of the page is an HTML img tag, which requests an image of the map from the server. On the server, objects from this user's browser session—a Map object, in particular—are used to render the map image, which is returned to the user's browser.
5. In the Page_Unload method, the application saves the user's state, by calling the AppStateManager.SaveState method.

6. The user selects the Zoom In tool and draws a marquee box around Australia. Javascript that is built into the Zoom In tool submits a request to the server to zoom in on Australia.
7. On the MapXtreme server, the application once again must fetch a MapXtreme Session object from the pool. Note that the MapXtreme Session returned from the pool may be a different object than was used to service the previous request. This object could be clean or dirty.
8. On the server the application calls the AppStateManager.RestoreState method, to restore the user state that was saved in step 4. This restores the MapXtreme Session to a known state that is appropriate for this user.
9. The application modifies the Map object's Zoom property so that it is zoomed into Australia. The application renders a new map image and streams the image to the client browser. Again, the application saves the user's state at the end of the request.
10. The user clicks on the Select tool, and then clicks on the Australia region of the map, to select Australia.
11. The application again fetches a MapXtreme Session from the pool, and again calls RestoreState to set the pooled object to a known state.
12. On the server, the application carries out the operation to select Australia. A new map image is rendered, to display the new selection.
13. On the server, the application again calls AppStateManager.SaveState, to save the user state. Since this example allows the user to perform selections, the AppStateManager's SaveState and RestoreState methods will need to include code to save and restore Selections.

    Any aspect of the MapXtreme session that you allow a user to change, such as layers, themes, queries, map views, must be handled in your code to save and restore each item.

Using pooling and managing user state manually is beneficial in that it improves efficiency and allows applications to be scalable. However, this model does require additional work, in terms of saving the user state at the end of each request, and then restoring that state at the beginning of that user's next request.

Configuring a Pooled Application to Use Manual State Management

For examples of a pool application using manual state management, see the sample web applications that ship with MapXtreme. The following are the main points for configuring such an application.

Configure pooled objects in the Web.Config file using the settings:

```
<!--Use this setting to turn Session pooling on/off (true/false)-->
    <add key="MapInfo.Engine.Session.Pooled" value="true" />
<!--Use this setting to save Session state automatically (HttpSessionState) or
manually (Manual)-->
    <add key="MapInfo.Engine.Session.State" value="Manual" />
<sessionState mode="StateServer" stateConnectionString= "tcpip=127.0.0.1:42424"
sqlConnectionString="data source=127.0.0.1;userid=sa;password="
cookieless="false" timeout="20" />
```

These two settings can be combined into the following options:

- For pooled objects and automatic state management, set Pooled to true, and set State to HttpSessionState.
- For pooled objects and manual state management, set Pooled to true, and set State to Manual.

Place the logic that determines the user's current state into the Page_Load method. See Implementing a StateManager on page 121.

User State

Your application must now consider the user's current state and the state of the MapXtreme Session object when it was retrieved from the pool. In this situation you will have to decide:

- Do you return the pooled object clean to the pool in your SaveState method?
- Do you let go of the pooled object and then restore any state in the RestoreState method?

The correct choice here is not always clear since many variables about the difference in state from one user to another may lead you to leave the session object dirty prior to releasing it back to the pool. Another application may save time by waiting for the Page_Load to clean up or to check if the Session needs cleaning.

The example in the section Manual State Management: A Walk-Through on page 116 followed the second option here: to clean up the pooled object in the RestoreState method.

The next section provides more information on how to save and restore state.

A Detailed Look at Manual State Management

This section provides a detailed example of how a pooled application can perform manual state management. We will examine relevant sections of code from the Thematics sample web application, which is installed as part of MapXtreme. If you have not already done so, you might want to run the Thematics sample to familiarize yourself with it, before reading this section.

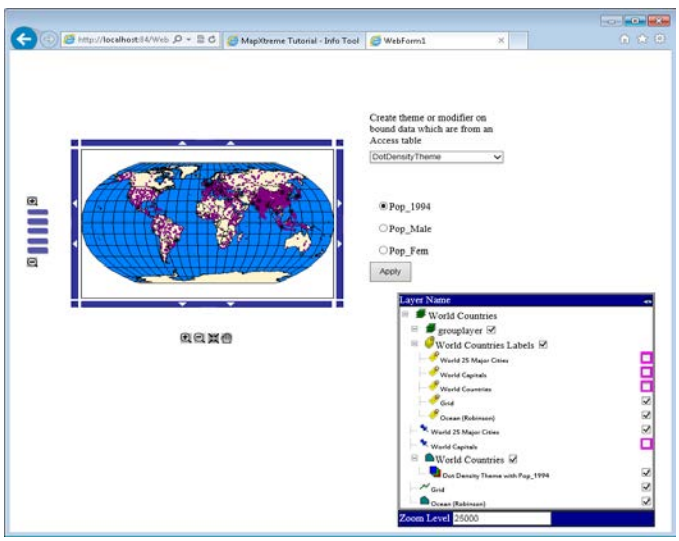
The Thematic sample is one of the projects included in the solutions here:

```
C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio20xx\Web\Features
```

This discussion covers the following topics:

Overview of the Thematics Sample

This application displays a map of the world, loaded from the workspace World.mws (part of the sample data installed with MapXtreme).



Options on the web form allow the user to create various types of thematic shading on the map. These themes modify the appearance of regions (country boundaries) in the table world.tab. The attribute data for the themes comes from a Microsoft Access database, eworld.mdb (included within the Visual Studio project).

In this sample, the world.mws workspace file is pre-loaded when the MapXtreme Session is created; however, the columns from eworld are added later, when a client accesses the application.

ⓘ To get better performance, put attribute column information into a workspace file, so that all needed data will be pre-loaded at the MapInfo Session creation time.

Application Settings

First let's take a look at the parts of the Web.config file that contain relevant application settings. Near the top of the Web.config file you will find these settings:

```
<configuration>
<appSettings>
    <!--Use this setting to turn Session pooling on/off (true/false)-->
    <add key="MapInfo.Engine.Session.Pooled" value="true" />

    <!--Use this setting to save Session state automatically (HttpSessionState)
or manually (Manual)-->
    <add key="MapInfo.Engine.Session.State" value="Manual" />

<!--Use this setting to preload a workspace on Session creation-->
    <add key="MapInfo.Engine.Session.Workspace" value="c:\Program
Files\MapInfo\MapXtreme\9.x.x\Samples\Data\World.mws" />
```

The settings are explained below:

- **MapInfo.Engine.Session.Pooled**: Because this key has a value of "true", MapXtreme Session objects are pooled by the COM+ service. When a new MapXtreme Session is retrieved upon a new client request, the Session could be "clean" - a new object just created - or it could be a "dirty" object which has already been used to handle a previous request. Because of the uncertain state of this object, this application will take steps within each request to restore the MapXtreme Session to a known state before using it.
- **MapInfo.Engine.Session.State**: Because this key has a value of "Manual", the application is responsible for explicitly saving and restoring Session state. The Manual setting aids performance, because the application's StateManager class is written with intelligence about which objects it should save and restore. The StateManager

selectively saves and restores only the minimum number of objects that are needed; that is why it is the most efficient state management option. (See Implementing a StateManager on page 121).

- **MapInfo.Engine.Session.Workspace**: This setting lets you specify a semi-colon separated list of.mws workspace files, which will be pre-loaded at the time the MapXtreme Session is created. To specify multiple workspaces, separate the filenames with semi-colons.

Farther down in the Web.config file, you will find this setting:

```
    <sessionState mode="StateServer"
stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data
source=127.0.0.1;user id=sa;password=" cookieless="false" timeout="20" />
```

This indicates how you wish to save state; in this case, to the StateServer where the information can be retrieved at a later time. You can also set this to SQLServer, where state is saved to a database.

Implementing a StateManager

The Thematic sample application uses manual state management, which means it requires an implementation of a StateManager class to save and restore the appropriate MapXtreme Session changes.

This class inherits from the abstract base class MapInfo.WebControls.StateManager and must implement the following methods:

```
Public Overrides Sub RestoreState()
Public Overrides Sub SaveState()
```

If you use the " template to create a new web site in Visual Studio, the resulting project includes an implementation of the StateManager, called AppStateManager.

The exact contents of the AppStateManager class vary from application to application. The more options your application provides, allowing the user to customize the map, the more code you need to add to the AppStateManager, to save and restore those customizations. This is why the AppStateManager that is provided with the Thematics sample application contains more code than the AppStateManager that you get when you create a new project from the MapXtreme Web Site template.

For example, the thematic sample handles customizations to theme layers, theme tables, attribute tables, and a group layer. The web template, in contrast, has code to handle customizations to layers and selections.

The RestoreState and SaveState methods are used as follows:

1. Each time a client submits a request, the Page_Load method calls RestoreState. RestoreState is also called each time a map tool is used on the client. The RestoreState method ensures that the MapXtreme Session object— which was retrieved from the pool in an unknown, possibly "dirty" state— is restored to a known state, either to the user's state, if it exists, or to the default state from Application state.

   For example, if the user's map was zoomed in on France the last time the user requested a map, then the RestoreState method will ensure that the map retrieved from the pool is returned to a "zoomed in on France" state. This way, the user's session can continue from where it left off.

   The following VB code example is from the Thematic sample application's WebForm1.aspx.vb.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As _
   System.EventArgs) Handles MyBase.Load
      ' The first time in
      If Session.IsNewSession Then

'//************************************************************
'// You need to follow below lines in your own application.
'//************************************************************
   Dim stateManager As New AppStateManager

' tell the state manager which map alias you want to use.
' You could also add your own key/value pairs, the value should be
' serializable.

stateManager.ParamsDictionary.Item(AppStateManager.ActiveMapAliasKey)_
   = Me.MapControl1.MapAlias
' Put state manager into HttpSession, so we could get it later on from
' different class and requests.

MapInfo.WebControls.StateManager.PutStateManagerInSession_
   (stateManager)
Me.InitState()
      End If
MapInfo.WebControls.StateManager.GetStateManagerFromSession.RestoreState()

      PrepareData()
 End Sub
```

2. After the Page_Load method, the application applies whatever business logic is appropriate, given the nature of the client request.

   For example, if the request was generated by the user clicking with the Zoom In tool, the business logic adjusts the map's center and zoom level.

3. At the end of the request processing cycle the Page_Unload method calls SaveState. SaveState is also called after a map tool is used on the client.

The SaveState method saves the map's latest state, so that when and if the user submits yet another request, that request will be able to call RestoreState again. Each call to SaveState is performed in anticipation of a possible subsequent call to RestoreState, the next time a request is received.

```
Private Sub Page_UnLoad(ByVal sender As Object, ByVal e As _
   EventArgs) Handles MyBase.Unload

MapInfo.WebControls.StateManager.GetStateManagerFromSession()_
   .SaveState()
End Sub
```

Serializing MapXtreme Objects in the Proper Order

In the AppStateManager class provided with the Thematics sample, the SaveState method calls the ManualSerializer.SaveMapXtremeObjectIntoHttpSession method several times. The objects are saved (serialized) in the following order:

```
ManualSerializer.SaveMapXtremeObjectIntoHttpSession(MapInfo.Engine.Session.Curr
ent.Catalog(SampleConstants.EWorldAlias), "mdb_table")

ManualSerializer.SaveMapXtremeObjectIntoHttpSession(MapInfo.Engine.Session.Curr
ent.Catalog(SampleConstants.ThemeTableAlias), "theme_table")

ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.
ThemeLayerAlias), "theme_layer")

ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.
GroupLayerAlias), "group_layer")
```

It is important that these SaveMapXtremeObjectIntoHttpSession calls occur in the correct order, because some objects are dependent on other objects.

In this example, first we save tables (which are referenced through the Catalog). Next, we save layers (which are referenced through the map's Layers collection). That is the appropriate order, because most types of layers are dependent on tables.

For example, each FeatureLayer is associated with a table. The table must be open before you can instantiate (or de-serialize) a FeatureLayer that uses that table. Therefore, this AppStateManager class saves the tables first, and then saves the layers.

Similarly, in the RestoreState method, the data objects are restored first, followed by calls to restore the layers:

```
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("mdb_table")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_table")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_layer")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("group_layer")
```

Automatically Deserializing MapXtreme Objects

In the SaveState method, we identify the objects to be saved using specific references, such as a specific layer in the map:

```
ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.
ThemeLayerAlias), "theme_layer")
```

The RestoreState method, however, does not reference the Catalog or the map.Layers collection. In the RestoreState method, we call methods on the ManualSerializer class, but we do not do anything with the results from that call, and there is no reference to a map.Layers collection: They are de-serialized automatically to the appropriate place in the Catalog or Layers collection, or wherever necessary.

```
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_layer")
```

ⓘ Use the ManualSerializer class to carry out saving and restoring objects. Do not store MapXtreme objects directly into the HttpSessionState using syntax such as:

```
HttpContext.Current.Session.Item("myTable") = _
        MapInfo.Engine.Session.Current.Catalog("myTable")
```

Handling Initial Requests

Application state is a server-side state management mechanism that is used to store information that is common, or global, to all user sessions. Application state is initialized when Session.IsNewSession is true. Application state is an important element when handling initial requests.

In the case of MapXtreme, Application state holds the initial state of a background map so that any user's initial request will receive the map in this default state.

The MapXtreme Thematics sample demonstrates the steps to take to handle initial requests.

There are two different types of "first" requests that your application must handle: 1) the first request from the very first user to access the web application, and 2) first requests from all subsequent users.

The very first time any user uses the application, a clean MapXtreme Session is retrieved from the pool with a background map in its initial state. This initial state is stored in Application state and used like a template. This happens in the SaveDefaultState method.

The first request from all subsequent users makes a call to RestoreDefaultState, which retrieves the initial state of the map from Application state.

The following are the steps related to initial requests.

1. In the Page_Load method, check to see if the current session is new. If it is, instantiate your StateManager and add the MapAlias into the state manager's ParamsDictionary property. This is mandatory when using Manual state management.
2. Next in the Page_Load method, since this is a new session, we initialize application data by calling the InitState method. In the InitState method, we prepare the initial state of the map.
3. Next, in the Page_Load method, we make a call to RestoreState to check whether this is the first request for this application.

   If this is the first request for the application, we call SaveDefaultState method to set various properties on the map to their initial state and we call a ManualSerializer method, to store this initial map state in a byte array into HttpApplicationState.

   Conversely, if this a user's first request, but not the very first request for the application, we make a call to RestoreDefaultState to reset the properties to their initial state.

Handling Subsequent Requests

After the user's first request has been handled, he or she will likely submit additional requests, such as clicking the map to zoom in. This time when RestoreState is called (either in the Page_Load method or from the map tools), the initial request logic will be skipped. Instead the code that restores the user's state is executed.

At the end of the user's session, or after using a map tool, the SaveState method is called to save the state of the session.

ⓘ This Page_Unload method will not get called if you use Server.Transfer or Response.Redirect with endResponse flag as true, because those methods will ignore this event handler. You could explicitly invoke MapInfo.WebControls.StateManager.GetStateManagerFromSession().SaveState() before processing Server.Transfer or Response.Redirect with endResponse flag as true to work around this issue.

A Closer Look at the MapXtreme Session

The MapXtreme Session is the starting point for all MapXtreme applications. It manages the initialization of resources needed for a MapXtreme application. The Session also provides access to other important objects such as Data.Catalog, MapFactory, CoordSysFactory, Selections, and others.

To access the MapXtreme Session, call the MapInfo.Engine.Session.Current() method. Each thread in your process has a MapXtreme Session object available. There can be only one MapXtreme Session per thread and this session cannot be shared by multiple threads.

Web applications can have one MapXtreme Session object per user, or pooled MapXtreme Sessions available to many users. The following section describes the COM+ pooling options in more detail.

Configuring Microsoft COM+ Object Pooling

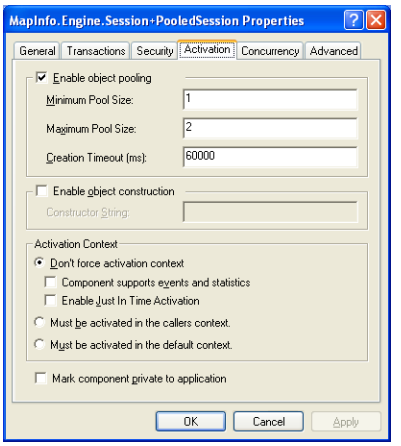
The MapXtreme Session object is registered with the COM+ services on your system. This system handles configuration and activation of any registered pooled objects. The runtime installer has the logic to register and create default settings for the MapInfo Session object. By default the MapXtreme Session object is configured with two pooled objects and a 60 second time-out.

Configuring the Pool Size

The MapXtreme Session is configurable using the system configuration methods for the appropriate section. These settings are available in both the system dialogs in the Control Panel and the .NET configuration files of your application. Web applications must manage these settings when using pooled MapXtreme Sessions.

One important setting is the number of pooled objects created by the COM+ system to service your running application. This setting is accessed using the Control Panel > Administrative Tools > Component Services. Browse through the Component Services > Computers > My Computer > COM+ Applications > MapInfo.CoreEngine > Components. Here you will find the MapInfo.Engine Session object. Right click the icon and select Properties.

The Property dialog allows you to set various properties including the number of pooled objects per application and the activation time-out for session creation. Setting the number of pooled objects correctly directly impacts your applications performance. To set the number of pooled objects correctly you need to understand your application and how it accesses data.



Background Map Affects Performance

Another aspect of the session that can directly impact your application's performance, is the startup workspace that you define in the application Web.config file. This workspace defines which maps, layers, and tables will be loaded when the session object is created. This happens in the COM+ pool thread and outside your web application space.

When the application starts, a request is made to the application pool for a MapXtreme Session object. The COM+ pool then creates the number of pooled objects you specified in the control panel Component Services dialog box. This MapXtreme Session object reads the Web.config file, and the specified workspace is loaded before your application gets an instance from the pool. Therefore, if loading the workspace takes more time than the time-out specified in the Component Services dialog box, you will receive a COM+ activation exception. So it is important to understand what you are loading in this workspace and how long it will take to completely load.

If you specified a Minimum Pool Size of 2, then the COM+ pool will create both copies of the MapXtreme Session instance on the first request for a pooled object. When the COM+ pool senses a time-out waiting for a pooled object, it will create more objects up to the maximum setting. If your workspace is very complex and takes a great deal of time to load, then the activation of subsequent objects may cause time-out errors when the application is active. The web application pool has settings to control the reactivation of pooled applications, including settings for recycling the pooled classes. So at some point your application and the pooled objects may be recycled, and this will cause another workspace loading.

You must decide which maps, tables, and layers are important to have as the background for your application, and load these via the workspace in the Web.config file, But be aware of the implications of a large workspace that takes time to load. To test your load time, simply load the workspace in Workspace Manager on the server machine to determine the actual load time and adjust your time-out setting accordingly.

Initial workspaces are very useful in loading maps, layers, and tables that do not change from user to user, but you have to be aware of the implications of loading complex workspaces.

Further Reading

The MapXtreme Session is highly configurable and is designed to work in conjunction with Microsoft COM+ and web technologies. To better understand how best to use this, you need to understand the Microsoft web delivery architecture. There are many places to get more information on the topics listed above. Here are some links to very good information on web architecture:

COM+ pooling	COM+ Object Pooling Concepts
IIS 7.0	IIS 7.0: Managing Web Applications in IIS 7.0
IIS 8.1	Web Service (IIS) Overview
IIS 8.5	Web Service (IIS) Overview
IIS 10.0	Web Service (IIS) Overview

7

7 – Desktop Applications, Controls, Dialogs, and Tools

This chapter provides information on planning a desktop application, as well as, an overview of the myriad desktop controls, dialogs, and tools available in MapXtreme.

In this chapter:

Planning a Desktop Application

MapXtreme is a .NET-based object model that exposes mapping functionality to any .NET-based development work. There are many emerging technologies that can be used to develop applications using the .NET framework.

Desktop applications provide a rich tool interaction for the user. While web applications are getting more sophisticated, applications designed for the desktop still have the advantage. For example, desktop applications have a “snap to node” capability that can be useful when placing a Point feature exactly at the intersection of the street. Desktop applications also offer InfoTips as you hover the mouse pointer.

If you have an existing WinForms or other .NET-based application framework, you can simply start integrating the MapXtreme classes into your application. The MapXtreme object model is a fully compliant .NET object model, so you have the full power and functionality of the .NET framework to interact with our object model. Simply develop your application in the mode defined by your application framework and reference the MapXtreme objects.

Best Practices for Desktop Applications

The MapXtreme MaForm Application template is designed for rapid prototyping only, since it only references the MapXtreme assemblies. Use this as a starting point for a quick demonstration or proof of concept.

WinForms and the .NET framework provide basic support for single document (SDI) and multiple document interface (MDI) based applications. WinForms does not provide an application framework comparable to Microsoft Foundation Classes (MFC). It is missing document view, data exchange, and other MFC-based user interface concepts.

However, Microsoft does provide its Composite UI Application Block (CAB), as a way of encouraging their recommended patterns and practices. Note that the Composite UI Application Block is available for .NET 2.0 Framework and above. More information is available at the Microsoft Developer Network website at msdn2.microsoft.com/en-us/library/aa480450.aspx.

A very important aspect of the .NET development model is the separation of business logic from presentation. MapXtreme uses this paradigm in its controls to allow the user to open a dialog with a copy of the object to be modified. In this way, the dialog is not directly editing your runtime objects and you can create a new object in another process or machine and pass back the serialized version for use by your application.

MapXtreme and COM

.NET does not mean the end of COM. Many development shops put a great deal of effort into the COM object model and have a large investment that can't simply be discarded. So what is the best method of developing applications that have COM objects?

It is more likely that new .NET code will have to interoperate with existing COM code (rather than the reverse situation), so we will focus on this scenario. A .NET client accesses a COM server through a runtime-callable wrapper (RCW). The RCW wraps the COM object and mediates between it and the .NET common language runtime (CLR) environment, making the COM object appear to .NET clients as if it were a native .NET object and making the .NET client appear to the COM object just as if it were a standard COM client.

You can create references to these COM object directly from Visual Studio for instance, and simply interact with them just as you would with a .NET class. This allows you to build your .NET application while talking to legacy COM objects and passing the results to MapXtreme .NET objects. Think of the .NET application you are developing as an orchestration piece of your application framework. In this way you can use the latest technologies available in the .NET framework and retain your existing investment in legacy libraries. Because all data from the COM objects has been translated to the .NET environment, passing these to other managed classes is allowed and causes no domain issues.

Sample Applications and Project Templates

MapXtreme integrates into Visual Studio and comes with various sample applications and project templates. The sample applications show different aspects of how to use our object model within the .NET framework and Windows Forms applications. The templates are basic starting points for application development.

The project templates that are part of Visual Studio allow you to create a starter project and demonstrate a generalized application framework. From Visual Studio, create a new project by selecting the MapForm Application template. This creates a basic single document interface (SDI) application. See the sample desktop applications for a basic multiple document interface (MDI) application and how best to integrate MapXtreme into this model.

The .NET Windows Forms development platform is basic in its approach and functionality. We recommend using third-party tools that supplies a more complete UI development platform.

The Desktop sample applications that ship with MapXtreme are designed to show you specific tasks with minimal overhead. Use these as learning tools instead of a starting point for production application as these were not designed as application development examples.

The remainder of this chapter describes the wide variety of controls, dialogs and tools available for building desktop applications.

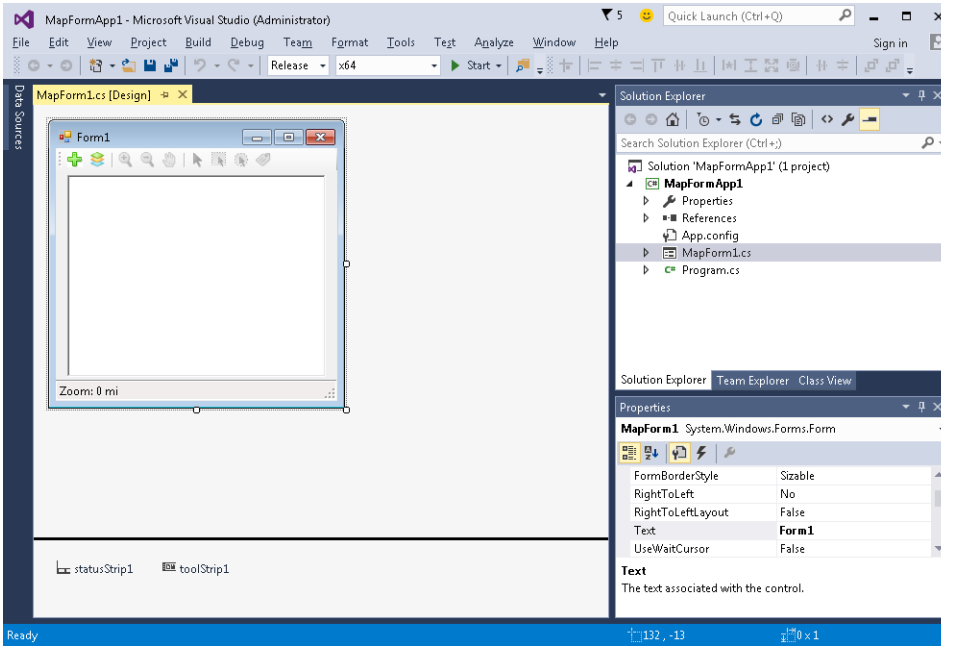
MapInfo.Windows.Controls Namespace

The MapInfo.Windows.Controls namespace contains classes that implement controls in desktop applications. The use of Windows controls is quite simple as you drop the desired controls onto your form and use the various properties to configure them to your specifications. You will find that the controls in this namespace are similar to controls found in the MapInfo.WebControls namespace. However, the controls in each namespace cannot be substituted for each other. The controls in this namespace are specifically to be used in desktop applications. For information about controls for web-based applications, seeChapter 5 Web Applications, Controls, and Tools.

ⓘ Some controls do not display at design time. When dragged and dropped onto a form they merely appear as a rectangle. These controls display properly at run time.

The desktop controls included with MapXtreme can be categorized into standard controls and map-specific controls. Standard controls include buttons, view tools, label tools, all kinds of boxes, etc. These are quite similar to (and in many cases, inherited from) classes that exist within the System.Windows.Forms namespace.

The map-specific controls are specific to Precisely's mapping implementations. These include controls that set or modify aspects of object styles, labels, layers, coordinate systems, themes, and other related functions.



Visual Studio Toolbox and Designer showing the list of available MapXtreme controls.

The most complex of the controls is the Layer Control, which is available as either a modal Dialog Box, or as a control that you can display directly on your form. The Layer Control has an API that allows extensive customization; see the "LayerControlDemo" sample application for examples.

Key Controls to Use in Desktop Applications

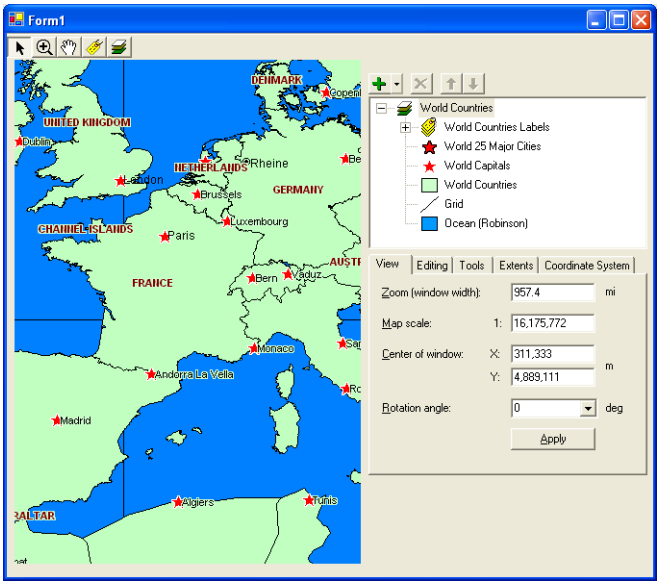
While all the controls included in MapXtreme are useful, several stand out from the rest as being central to most mapping applications. The following sections discuss these. Each of these controls assume the presence of the appropriate references included in your project. (These are automatically included if you have created your project from the MapXtreme MapForm template in Visual Studio).

ⓘ If you are not creating your project from one of our templates, please make sure to add references to the appropriate components (for example, MapInfo.CoreEngine, MapInfo.Windows, MapInfo.Windows.Dialogs, and MapInfo.Windows.Framework). See also MapXtreme Merge Modules.

MapControl

The MapControl is necessary for every application that displays a map. To add a MapControl to your form, drag it onto your form from the toolbox in Visual Studio. Once the MapControl is on your form there are several operations you can perform on it to make your map more useful to the user.

Modifications to the MapControl can be performed via the Properties Window when the MapControl is selected on your form. Right-click on the MapControl in design-mode to display a context menu with the following choices: **Load Map**, **Clear Map**, **Layer Control**, and **Create Thematic**. **Load Map** opens a standard file chooser from which the developer can choose a map to preload into the MapControl. **Clear Map** clears any map already in the control, either when adding the MapControl to the form or somewhere in the middle of the design process. **Layer Control** launches a Layer Control dialog box that allows you to customize the map (see Layer Control). **Create Thematic** launches the CreateThemeWizard (see CreateThemeWizard) allowing you to create a theme on the map.



A Form showing a MapControl, a Layer Control, and a MapToolBar

The MapControl also has several properties relating to the loaded map that can be preset so that the map shown at run time is further customized for the specific application. Properties that can be modified include zoom level, coordinate system, map center, and settings for three mouse buttons during run time. The mouse buttons can be set to Zoom, Pan, Draw geometries, Select, or any of several other map tools. See Overview of the MapInfo.Tools Namespace.

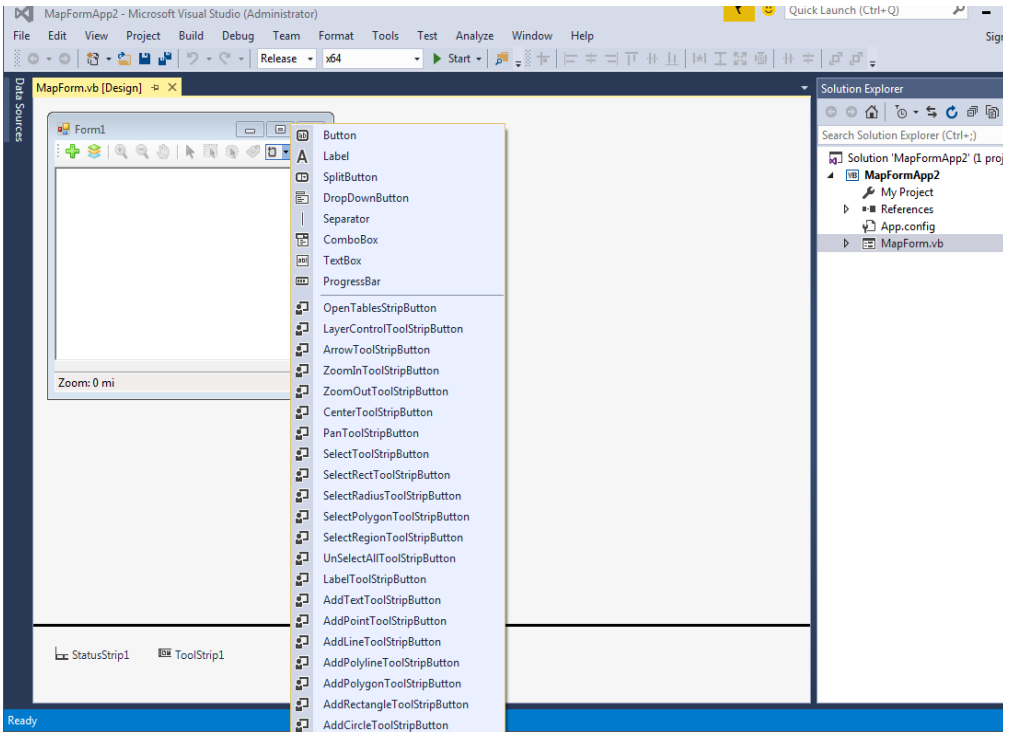
You can also set a design time tool to further manipulate the map while working in Visual Studio. Design time functionality includes Zoom In, Zoom Out, Select, Pan, Center, and the default arrow. Clear Map from the context menu removes all layers from the MapControl's map and closes the corresponding tables (if they are not in use in another MapControl).

MapToolStripButtons

MapXtreme provides ready-to-use map tools in its Windows Form templates and in the Visual Studio designer toolbox. Tools are added to the form by adding a ToolStripButton to a .NET ToolStrip.

The properties for each tool is set on the button. The MapToolBar, which had been responsible for managing the current tool is no longer required. The MapToolBar has been retained for backward compatibility and future use if you choose. It is still provided in the Visual Studio toolbox.

The 22 ToolStripButtons include the same mapping activities that have always been available in the MapToolBar, including navigation, selections, adding features, labeling, opening tables and displaying a layer control.



All but two of the ToolStripButtons can be added to a ToolStrip or StatusStrip. The tools that display the zoom level and scale of the current map view, can only be added to the StatusStrip.

To add a new ToolStripButton, in Visual Studio choose the ToolStrip from the toolbox under Menus and Toolbars. A split button appears on the form displaying a small down arrow. Click the arrow to display the list supported ToolStripButtons and choose what you need. Once added to the MapControl, the tool is immediately associated with it.

The API for the ToolStripButtons is contained in the MapInfo.Windows.Controls namespace. The abstract base class from which all the ToolStripButtons are derived is the MapInfo.Windows.Controls.MapToolStripButtonBase, which in turn was inherited from the .Net System.Windows.Controls.ToolsStripButton class.

For information on how to use the ToolStripButtons programmatically, view the code in the MapForm template and visit the Developer Reference. For information on the behaviors of each tool, see the individual MapTool classes in MapInfo.Tools namespace.

The MapToolBar

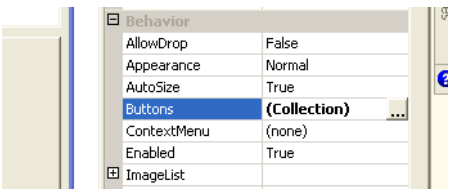
Beginning with version 7.0, the MapToolBar has been replaced in the MapForm template by a ToolStrip. See MapToolStripButtons. The MapToolBar is still available in the Visual Studio toolbox. This section outlines instructions on how to add a MapToolBar to a MapForm.

The MapToolBar combines several of the map tool controls (e.g., zoom in, open table) into a single control. The default collection of tools includes: OpenTable, LayerControl, Select, ZoomIn, ZoomOut, Pan, and Center.

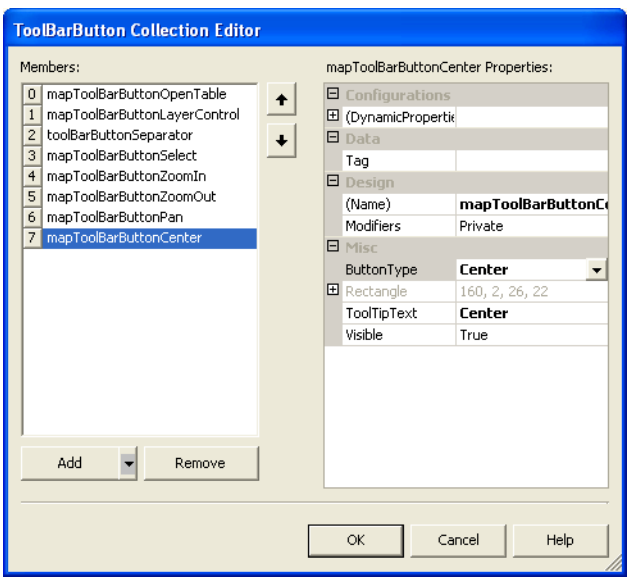
To add a MapToolBar to your MapForm, drag it from the Visual Studio toolbox. (The MapToolBar is no longer provided in the MapForm templates.)

To add tool buttons to the MapToolBar:

1. Open the ToolBarButton Collection Editor by clicking on the ellipsis (...) next to the Buttons property in the Properties window.



The ToolBarButton Collection Editor appears.



2. Click the **Add** button.

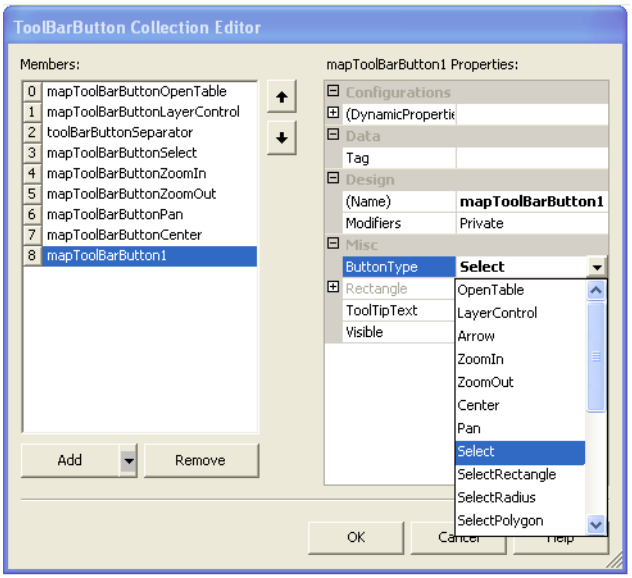
This creates a new MapToolBarButton below the last one in the list.

ⓘ If you would like to add a customized tool you need to click on the arrow next to the Add button and choose ToolBarButton. You need to write a customized handler for the new button.

---

3. Name the new button.
4. Select the ButtonType from the right-hand pull-down menu.

   For example, if you are adding a Select tool button, choose Select from the list.



5. Click **OK**.

   The MapToolBarButton is now created and added to the button bar. The newly added tool works in the default manner.

Adding Custom Buttons to a ToolBar

This procedure assumes you have created a custom tool and added it to the MapControl.Tools collection. Adding custom buttons involves assigning the button's ToolId to the name of the custom tool and adding a custom bitmap to the MapToolBar's ImageList.

For information on how to create a custom desktop tool, see Customizing Tools.

For information on custom bitmap symbols that ship with MapXtreme see Appendix F: Custom Symbols.

To add a custom button to a toolbar:

1. Add a MapToolBar to your form.
2. In the Visual Studio property window, highlight the Buttons property and press the ... button to invoke the ToolBarButton Collection Editor dialog
3. Click the **Add** button to add a new MapToolBarButton.
4. Set the button's ButtonType property to CustomTool (last item the drop-down list). Note that the button will now appear blank, as there is no image associated with the custom button yet.
5. Set the button's ToolId property to the name of the custom tool you added to the MapControl's Tools collection. If the ToolId value does not match a tool in the collection, then a runtime exception will be thrown when the user clicks the button.
6. Click **OK** to close the ToolBarButton Collection Editor dialog.
7. In the form's constructor, after the InitializeComponent call, add code to add a custom bitmap to the MapToolBar's ImageList. How the custom bitmap is associated with the application is up to the programmer. One option is to add an ImageList to the form at design-time, populate it with the custom images, then write code to transfer its images to the toolbar's ImageList at runtime. That code would look like this:

```
// Add custom tool button's bitmap to toolbar's image list
        foreach (Image image in this.imageList1.Images) {
           this.mapToolBar1.ImageList.Images.Add(image);
        }

        // Associate the bitmap with the custom tool's button (last image in the
list)
        this.mapToolBarButtonBlueSelect.ImageIndex =
this.mapToolBar1.ImageList.Images.Count-1;
```

Layer Control

The Layer Control dialog box shows all the layers that make up the current map and the status of the layers' attributes. These attributes are: visible, editable, selectable, and auto label. The icons above each check box column represent the attributes types. ToolTips display over the attribute icons when you move your cursor over them to help familiarize yourself with each icon. It is easy to change a layer's, or multiple layers' attributes using the check boxes.

The Layer Control also has options available to change the Display and Label settings; modify any thematic maps you have created, and reorder, add, or remove layers. You can also export or import the theme and style that you have created, as an xml file.

You can rearrange layers in the LayerControl by dragging and dropping them in the Layer Control Layers list.

ⓘ Dropping a layer onto a Label Layer adds a new set of labels to the Label Layer. This can happen by accident—for example, if you were attempting to move a layer to the spot just above the Label Layer.

Tip: If you want the dropped layer to be placed outside of the Label Layer, hold down the Shift key before you drop the layer. As you press and release the Shift key, the cursor changes to indicate whether the dropped layer will go above or inside the target layer onto which you are dropping.

The Layer Control puts all the functionality of the Layer Control dialog box onto a form. A single line of code is necessary in the Form_Load() method to link the Layer Control to the MapControl's map.

```
layerControl1.Map = mapControl1.Map;
```

To better understand the complexities and capabilities of the Layer Control, run the MapXtreme utility called Workspace Manager. This tool includes a working example of Layer Control. See Chapter 27 Workspace Manager. Workspace Manager is available from the Program Menu under MapInfo\MapXtreme\9.x.x.

Customizing Context Menus

You can create customized context menu items for your Layer Control that appear when the user right-clicks a node in the layer tree. Use the ContextMenuTargetObject property to return the object that the user right-clicked.

A code sample has been provided that shows how to define a LayerControlEnhancer class that allows a user to add custom items to the Layer Control's context menu. This example can be found in the LayerControl sample application located in the ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\LayerControl directory.

Keyboard Shortcut Programmability

You can create keyboard shortcuts to access everything on the LayerControl toolbar. The PerformDown, PerformUp, and PerformRemove methods already provide programmatic access to the Down, Up, and Remove buttons.

The AddMenuMnemonic property provides programmatic access to the key associated with displaying the Add menu. The syntax for the AddMenuMnemonic property is:

```
public System.Windows.Forms.Keys AddMenuMnemonic {get; set;}
```

MapInfo.Windows.Dialogs Namespace

The MapInfo.Windows.Dialogs namespace contains classes that create dialog boxes with specific functions. Contrary to controls, dialog boxes are not visible during design-time and can only be created and configured in code. You can create your own customized dialog boxes using classes from the MapInfo.Windows.Dialogs namespace and then utilize them by calling the System.Windows.Forms.Form.ShowDialog() method.

The stock dialogs are built from the MapXtreme object model's public API. There are no hidden internal fields or anything private. You can use these dialogs or customize them for your needs, or write your own from scratch. You may wish to provide fewer controls on a stock dialog so that your users are restricted from changing some behavior.

To use a particular dialog box in your application, add the MapInfo.Windows.Dialogs namespace to your code as follows:

```
using MapInfo.Windows.Dialogs

private void DoLayerControl()
{
   LayerControlDlg layerControl = new LayerControlDlg();
   layerControl.Map = mapControl1.Map;
   layerControl.ShowDialog(this);
}
```

The code above displays the LayerControl dialog box when the DoLayerControl() method is called.

Stock Dialogs

MapXtreme ships with many stock dialogs and controls to handle tasks of manipulating the MapXtreme objects. All the dialogs and controls used in Workspace Manager are available for use in your application. The dialogs are simply containers for the low level controls. These controls are designed to work specifically with many of the MapXtreme objects.

The controls that ship with MapXtreme are used by the stock dialogs to create the basic UI components. The controls are placed in dialogs to create specific UI components. You can, in the same way, use the base controls to design your own dialogs. Simply create a dialog class and start dragging the controls onto the surface to create your dialog.

The stock dialogs also can be customized through visual inheritance. You can derive a class from our dialogs and customize some of the behaviors by overriding methods and properties. Other dialogs such as the Layer Control are customizable, so you can change the default behavior to hide controls you don't want users to access, change the look of the icons, or remove tabs.

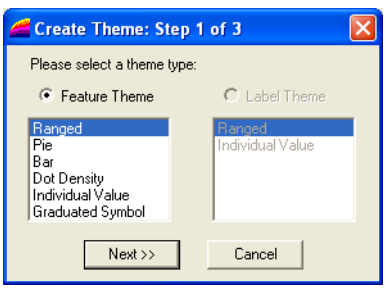
Basically you have control over all the UI components that ship with MapXtreme. This allows you to create your own custom interface. Using any of the methods described allows you to create a customized desktop application that exposes only what you need rather than everything MapXtreme defines.

CreateThemeWizard

The CreateThemeWizard class is a convenience class that you can add to your application. This class is used to guide the user through the process of creating a new theme by launching a wizard (a set of dialog boxes). There are three basic steps to creating a theme: 1) Select the theme type; 2) Select the table and columns to use for the theme; and 3) Modify the theme attributes (style, number of ranges, etc.). The wizard ties all these settings together into a sequence of dialog boxes to make theme creation as simple as possible for the end users of MapXtreme applications.

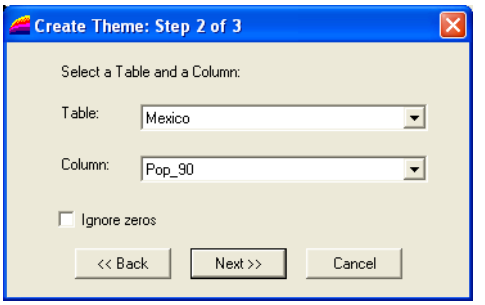
Using the CreateThemeWizard

The CreateThemeWizard displays several different dialog boxes, depending on the choices made by the user. The first dialog box allows you to select a theme type.



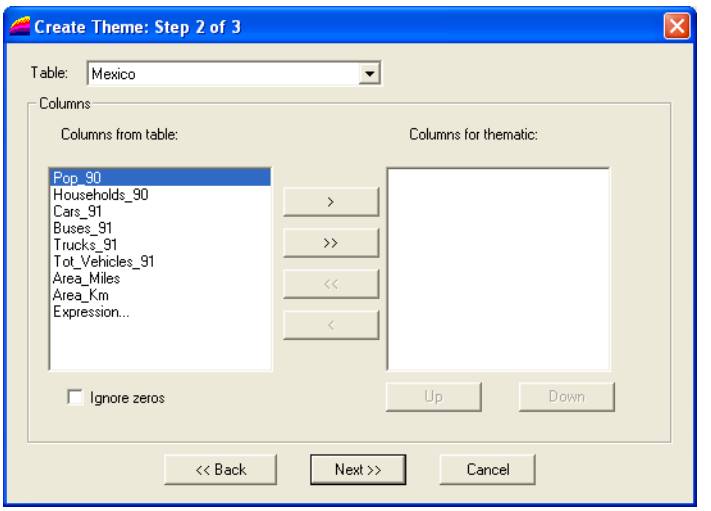
Create Theme: Step 1 of 3 dialog box

If the user chooses a single variable theme type (Ranged, Dot Density, Individual Value, or Graduated Symbol) the second dialog box displayed allows them to choose a table and single column.



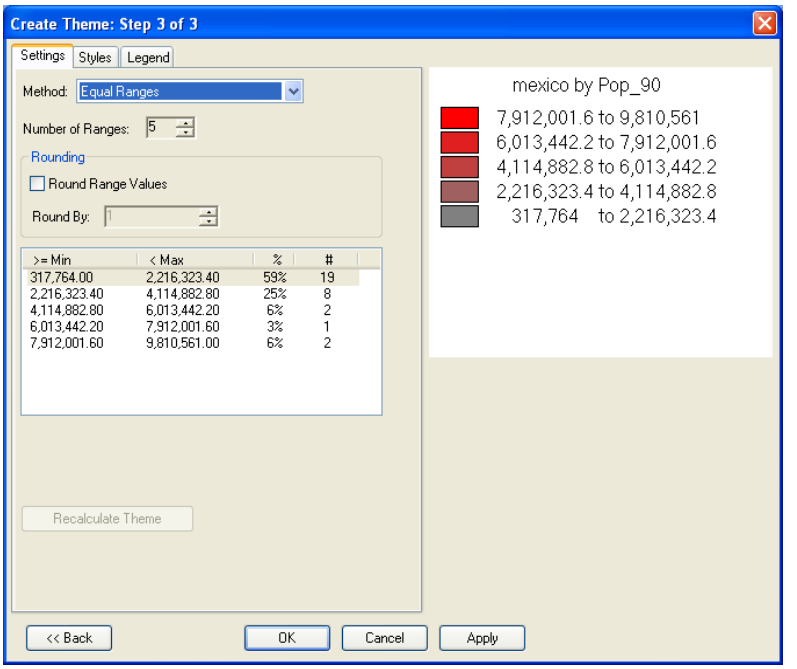
Create Theme: Step 2 of 3 dialog box (single column)

If the user chooses a multiple variable theme type (Pie or Bar), the second dialog box displayed allows them to choose multiple columns of data from which to create the theme.



Create Theme: Step 2 of 3 dialog box (multiple columns)

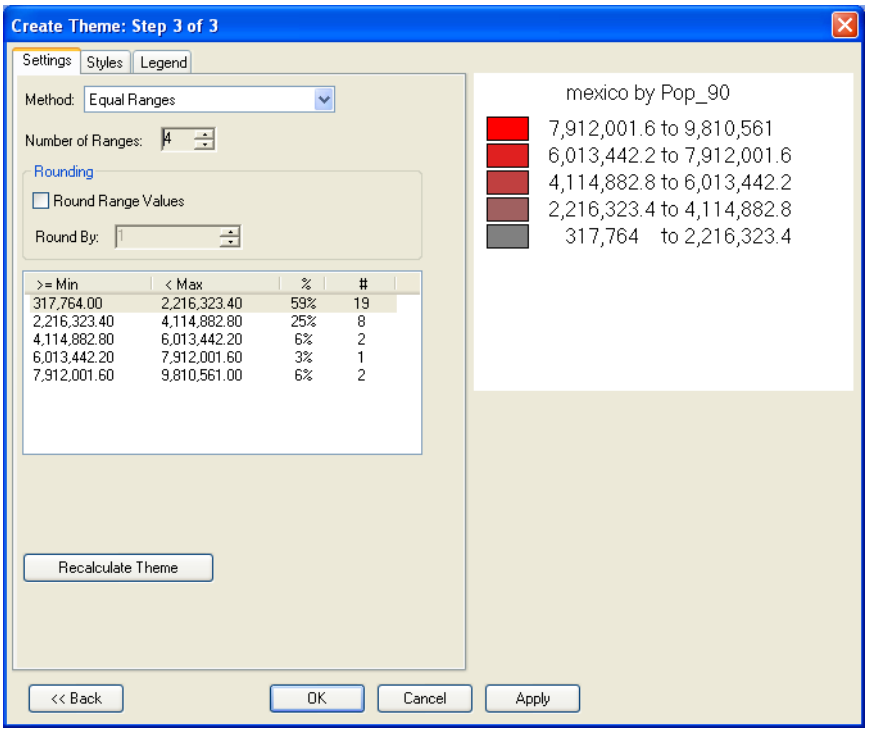
The following dialog box is specific to the type of theme chosen in Step 1. The figure shows a Step 3 dialog box specific to a ranged theme.



Create Theme: Step 3 of 3 dialog box (Ranged Theme)

The dialog box shows the default theme distribution method: EqualRangeSize. The corresponding name of the distribution method in the user interface of the wizard is Equal Ranges.

When a user make a change in the Settings tab, such as changing the number of ranges, the Recalculate Theme button is activated. The user must click the **Recalculate Theme**, **Apply**, **OK** button, or on a new tab to perform the recalculation. The recalculation does not occur automatically. The figure below shows a Step 3 dialog box with a change to the number of ranges. The **Recalculate Theme** button is now available.



Create Theme: Step 3 of 3 dialog box (Ranged Theme) After Settings Change

Programmatic checks look for changes in the values of the Settings tab and if the theme bins need to be recomputed after:

- Tabbing to the Styles or Legend tab
- Clicking the Apply button
- Clicking the OK (accept) button

The CreateThemeWizard is accessible from many different places at design time and at runtime, depending on the controls in your application. At design-time the CreateThemeWizard can be accessed from the menu at the bottom of the MapControl properties window and also from the Layer Control by right-clicking on a layer and choosing **Add Theme**.

Developing with the CreateThemeWizard

To add a CreateThemeWizard programmatically, follow the example code.

```
using MapInfo.Windows.Dialogs;

CreateThemeWizard themeWizard = new CreateThemeWizard(mapcontrol1.Map, this);
```

Methods

Once the wizard is created, call the CreateTheme() method from the CreateThematicWizard class to create the theme. An optional string parameter can be added to set the theme's alias. The alias can be used to access the theme from the Layers collection in the case of an ObjectTheme (Bar, Pie, GraduatedSymbol) or the Modifiers collection in the case of a FeatureStyleModifier (Ranged, RangedLabel, IndividualValue, IndividualValueLabel, Dot Density). You would access the theme to modify or delete it.

```
ITheme theme = themeWizard.CreateTheme("theme1");
```

If you already know which table is to be themed or which label source to use for a label theme, you can use the methods, CreateFeatureTheme() for tables or CreateLabelTheme() for labels.

Properties

There are a few properties that can be useful for obtaining information about the theme creation. WizardResult contains an enumeration WizardStepResult that allows you to check how the user exited the wizard. These choices are WizardStepResult.Done and WizardStepResult.Cancel. These are useful to appropriately update the controls or menus in your application.

The following example shows of the use of this.

```
if (createThemeWizard.WizardResult == WizardStepResult.Done
{
   // Update the menus
   mnuRemoveTheme.Enabled = true;
   mnuModifyTheme.Enabled = true;
}
```

SelectedLabelSource, SelectedLayer, and SelectedThemeType are properties on the CreateThemeWizard class that can be used to find out which layer or label source the theme has been applied to. This is useful when you want to provide the ability to modify a theme and need to access that object.

To modify a theme, determine the type of theme and then launch the appropriate type of modification dialog. Dialog box classes for modifying themes are ModifyBarThemeDlg, ModifyDotDensityThemeDlg, ModifyGradSymbolThemeDlg, ModifyIndValueThemeDlg, ModifyPieThemeDlg, and ModifyRangedThemeDlg.

MapXtreme ships with a ThemeDialogs sample application in which you can analyze the implementation and customize it for your needs. See ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\ThemeDialogs under your installation of MapXtreme.

Customizing Controls and Dialog Boxes

MapXtreme ships with a wide variety of dialog box classes in the MapInfo.Windows.Dialogs namespace. Each of the dialog boxes represented there have possible customizations specific to each one. To modify a specific dialog box, assign values to the different properties that are specific to each dialog box. Since there are so many, it is not practical to list all the possible customizations here. Please refer to the online reference for details about each dialog box.

An example of customizing a dialog box is as follows: The LineStyleDlg class creates a Line Style Dialog Box. This dialog box can be created in either a sparse style mode (where nothing is selected when opened), or in a mode where some choices are visible but not enabled (grayed out).

The customization provided by setting property values is obviously limited to the specific dialog object being modified. To create a new dialog box based upon a design of an established one is done by adding some controls to a new form at design time. For example, to create a modified LineStyle Dialog Box, you can drop a LineStyle Control on a new form and then add other controls to that form, making, in effect, a modified LineStyle Dialog Box.

Overview of the MapInfo.Tools Namespace

MapTool Object Model

The MapTools are the objects you use to interact with the map. These tools implement the basic behaviors you expect of a map. There are view-based tools, zooming and panning, tools for creating geometries on a specific layers, and tools for generating selections based on an area of the map. Each tool is customizable using events to capture pre- and post-process events.

Properties of the methods associated with the tools are on the tool and not the layer. For example, the selectability of a layer is specified on the tool and not the layer. In this way, you can use one tool to select an object on one layer and use another tool to select from a second layer. This gives much more freedom in the tools but also enables you to emulate the properties on layers. The Layer Control operates on the entire tools collection of the map, when selectability or other properties are changed via its UI.

This design also allows you to create tools that insert into specific layers. For example, a city planner may want the application to insert manholes on the water layer only, but to insert trees on the vegetation layer. You could create a custom tool that when selected, inserts only the specified symbol or geometry on the appropriate layer. This also works for

the stock tools, so that drawing a polygon will always be on the specified layer. Again the stock Layer Control manipulates the tools collection to make it seem as if the insertion is on a single layer.

The MapInfo.Tools namespace contains all the classes that allow you to create basic and customizable tools for your MapXtreme application.

You can customize new instances of these stock tools with event code, or write your own tool classes that derive from stock tools but override specific methods. See Customizing Tools on page 155.

Tools can be assigned to a particular mouse button (left, right, or middle). Use the following string tool names for the appropriate mouse button property: "Arrow", "ZoomIn", "ZoomOut", "Center", "Pan", "SelectPoint", "SelectRect", "SelectRadius", "SelectPolygon", "SelectRegion", "AddPoint", "AddLine", "AddPolyline", "AddPolygon", "AddRectangle", "AddCircle", "AddEllipse", "AddText", "Label".

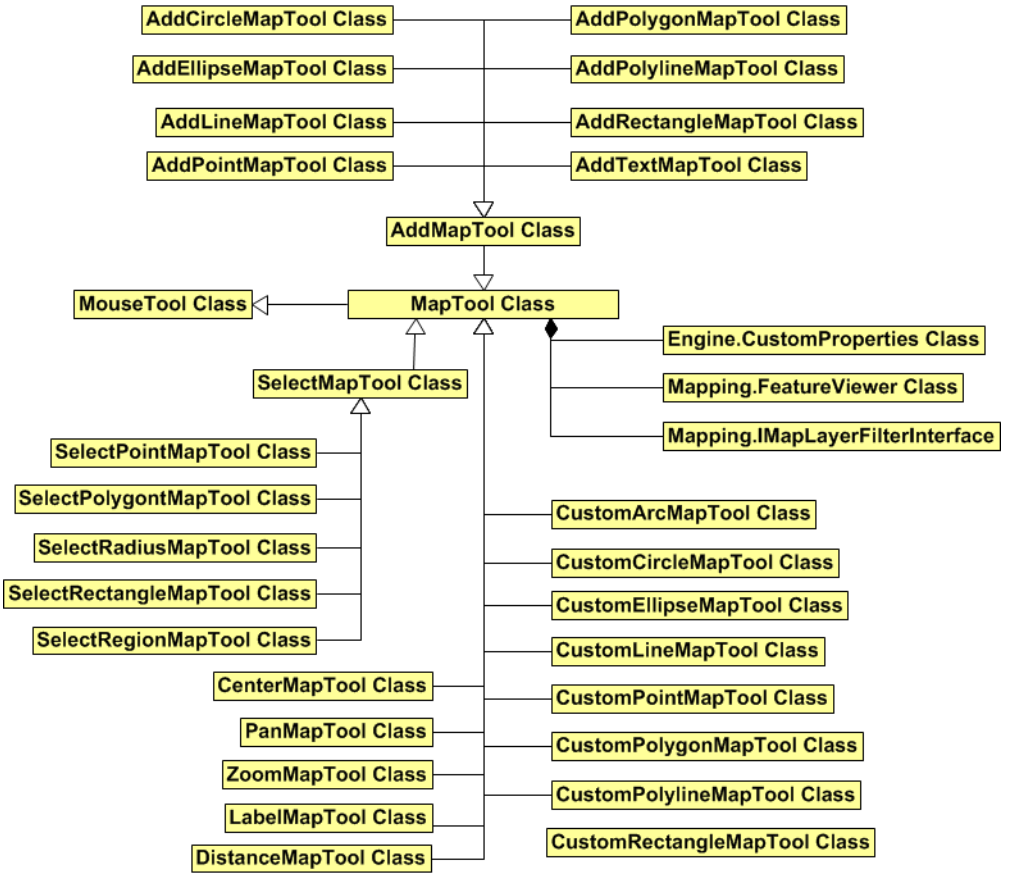
For example, the code below sets the left mouse button to the ZoomIn tool:

```
mapControl1.Tools.LeftButtonTool = "ZoomIn"
```

For an example of using the desktop tools programmatically, see the sample application under ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\DesktopTools under your installation of MapXtreme.

MapXtreme Desktop Tools API

The diagram below shows a UML representation of the MapTool class.



Tools are ultimately derived from the MapTool class. Select and Add tools are derived from the SelectMapTool and AddMapTool classes, respectively.

The Tools included in this namespace can be divided into four categories: View tools, Select tools, Add tools and Custom tools.

Within the Add, Custom and Select tool categories are tools for drawing or selecting these geometric objects: Ellipse, Arc, Circle, Rectangle, Polygon, Point, Line, and Polyline.

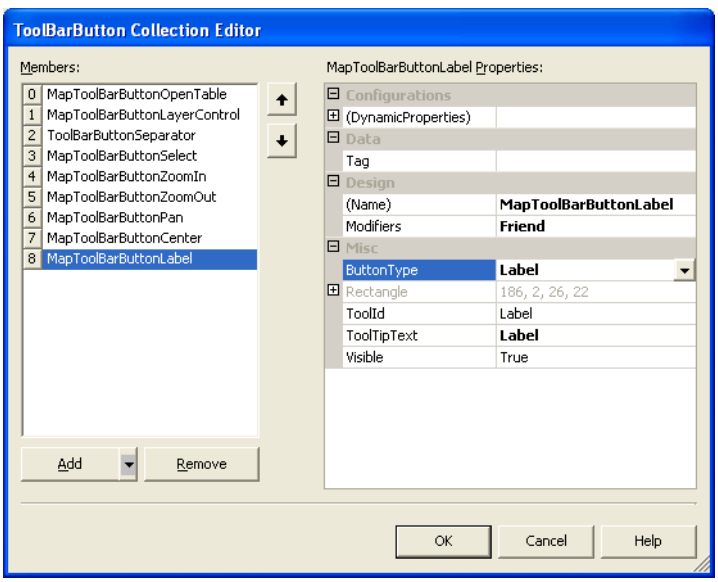
All of the tools have properties associated with them which implement three interfaces: IMapToolProperties, ISelectMapToolProperties, and IAddMapToolProperties.

View Tools

The View tools (ZoomMapTool, CenterMapTool, and PanMapTool) change the view of the map accordingly to the tool being used. These tools are part of the default tool collection. If you begin building your desktop applications with a MapXtreme template, these tools will appear on the default toolbar.

The LabelMapTool allows you to label features on the map. This tool is not included in the default tool collection. on the MapForm template. To add it to the MapControl, first add a ToolStrip from the Visual Studio Toolbox, right-click on the context menu item LabelToolStripButton.

To add it. click on the Collections ... button under the Buttons property of the MapToolBar.



Select Tools

The Select family of tools (SelectMapTool, SelectPointMapTool, SelectPolygonMapTool, SelectRadiusMapTool, SelectRectMapTool, and SelectRegionMapTool) select points and regions that fall within the geometric object shaped by the use of these tools. For example, the SelectRectMapTool selects objects that fall within the rectangle. SelectMapTool is the abstract base class that the other select tools derive from.

ⓘ MapXtreme does not support the intersection of an object with a polyline, hence there is no select polyline tool.

The SelectPointMapTool can be used to rotate objects.

A SelectNode mode allows individual points to be moved, added, or removed from objects. See Editing a FeatureGeometry with the Select Tool on page 157 for an example.

Dynamic Selection shows what objects can be in the selection as the mouse is being moved when using the SelectRectMapTool, SelectRadiusMapTool, and the SelectPolygonMapTool.

The selection map tools have properties to set various options, including which layers are selectable. These default tool properties can be set on the MapTools collection which can be found in MapInfo.Windows.Controls.MapControl.Tools. Optionally, each tool can set its own overriding properties and specify to use the default value on the MapTools collection or its overriding value.

To set the default list of selectable layers, create an IMapLayerFilter which specifies which layers are selectable. Then set the default property on the MapTools for the SelectableLayerFilter.

For example, to set the default list of selectable layers to work on all vector layers:

```
// select from all vector layers
IMapLayerFilter selectableLayerFilter =
MapLayerFilterFactory.FilterAnd(MapLayerFilterFactory.FilterVisibleLayers(true)
, MapLayerFilterFactory.FilterByLayerType(LayerType.Normal));

mapControl1.Tools.SelectMapToolProperties.SelectableLayerFilter =
selectableLayerFilter;
```

For more information on Select tool properties see MapInfo.Tools.MapTool.ISelectMapToolProperties in the MapXtreme Developer Reference.

Add Tools

The Add family of tools (AddArcMapTool, AddCircleMapTool, AddEllipseMapTool, AddLineMapTool, AddPointMapTool, AddPolygonMapTool, AddPolylineMapTool, and AddRectangleMapTool) are all based on the AddMapTool class. The Add tools allows you to draw a particular geometric object on your map. In order to add an object to a map, you must have an active insertion layer in your map where the object resides after it is created. The size and shape of the particular object depends on the tool used and can be constrained through the use of the modifier keys (Shift and Ctrl) while the tool is being used.

For more information on Add tool properties see MapInfo.Tools.MapTool.IAddMapToolProperties in the MapXtreme Developer Reference Help.

Custom Tools

The Custom tools (CustomArcMapTool, CustomCircleMapTool, CustomEllipseMapTool, CustomLineMapTool, CustomPointMapTool, CustomPolygonMapTool, CustomPolylineMapTool, and CustomRectangleMapTool) are very basic tools that only fire events. Use these classes to design custom behaviors for specific tools. For example, you can use the CustomEllipseMapTool to draw green Ellipses with a red outline every time the tools is used. Use the various tool events (see Tool Events on page 156 to specify particular behaviors to respond to specific events generated by the tools.

Custom tools are not included in the default tools collection. If you wish to assign a custom tool to a mouse button, you may provide any string you wish to identify the tool.

For more on customizing tools, see Customizing Tools on page 155.

Shape Tools

MapXtreme provides a group of tools for drawing geometric features on your map. Most of these tools are used by “click-and-drag.” As you click and then drag to another location on your map, a rubber-band image is displayed showing the current size and shape of the object being drawn. Each of these tools uses the Esc key to cancel the current operation (if appropriate).

The following are types of shape tools MapXtreme supports.

Line Tools

The tools included in this group are AddLineMapTool and CustomLineMapTool. These tools draw a Line. The tool is activated by clicking and dragging from the beginning point to the end point. Releasing the mouse button creates the Line. If the Shift key is pressed while dragging, the angle of the Line is constrained to multiples of 45°. If the Ctrl key is pressed while dragging, the line doubles in length and height. Pressing Esc before releasing the mouse cancels the operation.

Polyline Tools

The tools included in this group are AddPolylineMapTool and CustomPolylineMapTool. These tools draw a Line with multiple segments. The tool is activated by clicking on a point and then clicking on a subsequent point. When you are done clicking on points, click

again on the last point and the Polyline is drawn. If the Shift key is pressed while dragging, the angle of the Line is constrained to multiples of 45°. The Ctrl key has no effect on this tool. Pressing Esc before completing the line, cancels the operation.

Ellipse and Arc Tools

The tools included in this group are: AddEllipseMapTool, CustomEllipseMapTool and CustomArcMapTool. The Ellipse tools draw an Ellipse. The CustomArcMapTool draws an arc only; it does not insert the arc into a layer.

The Ellipse tool is activated by clicking and dragging from a point on one side of the perimeter to a point on the other. Releasing the mouse button creates the Ellipse. If the Shift key is pressed while dragging, the Ellipse is constrained to a circle where the radius is constant. If the Ctrl key is pressed while dragging, the Ellipse is drawn with the starting point in the center. Pressing the Esc key before releasing the mouse cancels the operation. Note that the axis of the Ellipse is always aligned to the coordinate system.

MapXtreme provides a lightweight CustomArcMapTool that draws a reference arc on the map, but which does not provide or insert a LegacyArc FeatureGeometry into the layer. To create an arc tool with such behavior, you must create your own implementation. For example, if you want a CustomArcMapTool to have the same operations as an AddMapTool, create a new arc tool class that derives from CustomArcMapTool and implements the IAddMapToolProperties interface.

Circle Tools

The tools included in this group are AddCircleMapTool and CustomCircleMapTool. The Circle tools draw a Circle. This is equivalent to using the Ellipse tool while holding down the Shift key. The tool is activated by clicking and dragging from the center point of the Circle center to a point on the perimeter. As you drag, you will see the size of the Circle expand around that center point. Releasing the mouse button creates the Circle. If the Ctrl key is pressed while dragging, the Circle is drawn from perimeter point to perimeter point. Pressing the Esc key before releasing the mouse cancels the operation.

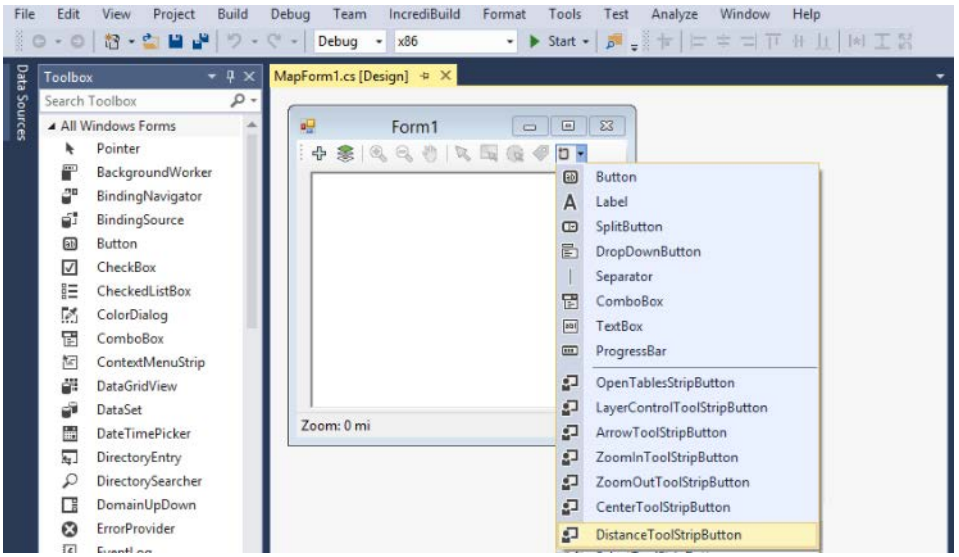
Rectangle Tools

The tools included in this group are AddRectangleMapTool and CustomRectangleMapTool. These tools draw a Rectangle. The tool is activated by clicking and dragging from one corner to the opposite corner. Releasing the mouse button creates the Rectangle. Note that the drawn object is always axis-aligned. If the Shift key is pressed while dragging, the Rectangle is constrained to a square. If the Ctrl key is pressed while dragging, the Rectangle is drawn with the starting point in the center. Pressing the Esc key before releasing the mouse cancels the operation.

Distance Map Tool

The DistanceMapTool allows you to get the distance between two or more points on the map. The tool is activated by selecting the tool in tool bar. To get the distance between points user need to click multiple points on the map. Double-clicking the mouse button completes the operation and fires DistanceComputed event, which gives distance between the start point, intermediate point(s) and the end point.

This tool is not included in the default tool collection on the MapForm template. To add this tool to the MapControl, you need to first add a ToolStrip from the Visual Studio Toolbox, then right-click on the context menu item and choose DistanceMapTool.



Using InfoTips

An InfoTip is a small text box that can appear when the mouse hovers over a map feature. A map is associated with an instance of MapInfo.Tools.MapTools, which contains all the tools (instances of MapInfo.Tools.MapTool) that can be used on the map.

The MapInfo.Tools.MapTools class has three properties that control the appearance of InfoTips. They are:

- InfoTipsEnabled–Gets/Sets whether InfoTips will appear when the mouse is idle.

- InfoTipTimerDelay–Gets/Sets the amount of time (in milliseconds) that must elapse before the InfoTip appears. The default is 500.
- InfoTipDisplayDelay –Gets/Sets the amount of time (in milliseconds) that the InfoTip will be displayed. The default is 0, which means that the Infotip will be displayed until the mouse is moved.

Note that as long as the mouse is moving, InfoTips will not appear. An InfoTip will only appear when the mouse is over a feature with a label and has been idle for at least the interval specified by InfoTipTimerDelay. The InfoTip will disappear when the mouse is moved or when the mouse has been idle for longer than the time specified by the InfoTipDisplayDelay provided that time is greater than zero.

You can prevent InfoTips from being displayed at all with:

```
mapControl1.Tools.InfoTipsEnabled = false;
```

You can set the delay before an InfoTip is displayed to one second with:

```
mapControl1.Tools.InfoTipTimerDelay = 1000;
```

You can set the length of time an InfoTip is displayed even though the mouse remains idle to 2.5 seconds with:

```
mapControl1.Tools.InfoTipDisplayDelay = 2500;
```

See the MapInfo.Tools.MapTools class in the MapXtreme Developer Reference for more information on using InfoTips in the API.

Customizing Tools

Tools for desktop applications can be customized in two ways: by creating a subclass of an existing tool, or by using one of the tools in the Custom Tools group. Any of the Tool classes provided in MapXtreme can be customized by creating a subclass derived from the desired particular Tool class. An example of this is deriving a new tool class from the AddLineTool class. The tool could be changed to a particular behavior (e.g., always beep when line is drawn), a particular appearance (e.g., always draw the line in red), or to a particular functionality (e.g., always constrain line drawing to 90 degree increments).

Another example would be to create two different tools based upon the same tool. The AddPointTool could be used to subclass two other tools that each use different symbols for indicating two kinds of points on a map. Assign one tool to the left mouse button and assign the other to the right mouse button.

The following code example illustrates a customization of the AddPolygonTool.

VB example:

```
Dim insertionlayerfilter As IMapLayerFilter
Dim style As MapInfo.Styles.CompositeStyle
Dim addmaptoolproperties As MapInfo.Tools.AddMapToolProperties
Dim maptool As MapInfo.Tools.MapTool

insertionlayerfilter = _
   MapLayerFilterFactory.FilterByLayerType(LayerType.Normal)

style = New MapInfo.Styles.CompositeStyle

addmaptoolproperties = New _
   MapInfo.Tools.AddMapToolProperties(MapLayerFilterFactory.Filter_
      ForTools(MapControl1.Map, insertionlayerfilter, _
      MapLayerFilterFactory.FilterVisibleLayers(True), _
      "CustomPolygonAddMapToolProperties", Nothing), style)

maptool = New MapInfo.Tools.AddMapTool(MapControl1.Viewer, _
   MapControl1.Handle.ToInt32(), MapControl1.Tools, New _
      MapInfo.Tools.MouseToolProperties(Cursors.Default, _
      Cursors.Default Cursors.Default),_
      MapControl1.Tools.MapToolProperties, addmaptoolproperties)
```

Additionally, MapXtreme ships with a desktop tools sample application that you can use to analyze and customize it for your needs. See ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\DesktopTools\cs in the MapXtreme installation directory on your computer.

Tool Events

The MapInfo.Tools namespace supports tool events with information and the ability to cancel the tool's operation. A Select Tool's event lists the items being selected or deselected. Events can be fired at every stage of use of a particular tool. In your code you can trap certain moments in the lifecycle of the usage of a tool. Adding code to particular events gives you the maximum flexibility on customizing the use of each tool in your mapping application.

The events in the namespace are as follows:

FeatureAddingEventArgs	This event is fired when an add tool is about to draw an object.
FeatureAddedEventArgs	This event is fired when an add tool has added an object to a table and map.
FeatureSelectingEventArgs	This event is fired when a selection tool is about to change the selection.

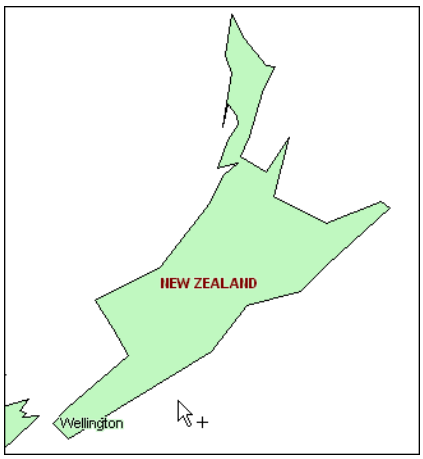
FeatureSelectedEventArgs	This event is fired when a selection tool changes the selection.
FeatureChangingEventArgs	This event is fired when a selection tool is about to change a feature. Use this event to check to see if the change that is going to happen is valid.
FeatureChangedEventArgs	This event is fired after a selection tool changed a feature.
NodeChangedEventArgs	This event is fired when a selection tool alters the nodes in a selection.
NodeChangingEventArgs	Fired when a selection tool is about to change a node of a selected feature. Use this event to check to see if the change that is going to happen is valid.
ToolActivatedEventArgs	This event is fired when a mouse tool is activated.
ToolActivatingEventArgs	This event is fired when mouse tool is about to be activated.
ToolEndingEventArgs	This event is fired when a mouse tool is about to end. This is a good place to have a new action begin.
ToolUsedEventArgs	This event is fired when a mouse tool is in use. Use this event to set flags for the beginning, middle, and end of a the sequence of mouse clicks.
DistanceToolEventArgs	This event is fired before the finish of DistanceMap tool and contains distance, distance type and distance unit.

Editing a FeatureGeometry with the Select Tool

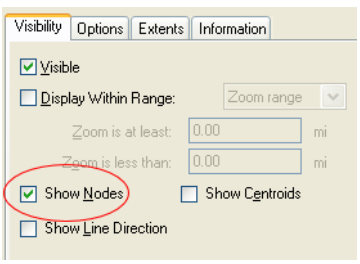
The following section shows you how to edit features by reshaping and adding nodes to a FeatureGeometry. It illustrates the use of a MapControl, LayerControl and the Select tool. See also MapControl on page 134 and Layer Control on page 139.

Reshaping a Feature

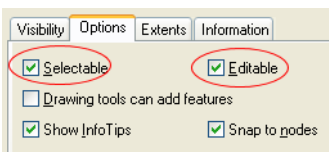
1. Load a map into the MapControl and zoom in on the Feature you wish to modify.



2. Select the object's layer in the LayerControl.
3. On the Visibility tab of the LayerControl, select the Show Nodes check box.

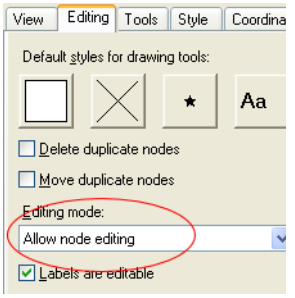


4. On the Options tab, select the Selectable and Editable check boxes.

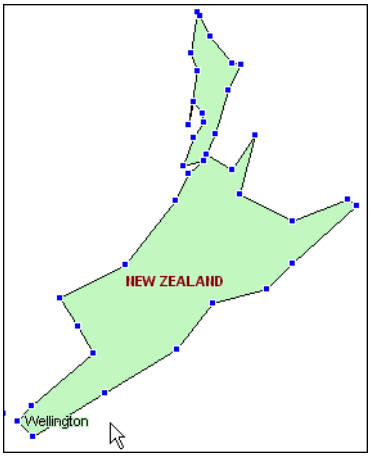


5. Select the root node of the map in the layer control tree view.

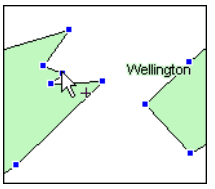
6. On the Editing tab, select Allow Node Editing from the Editing mode combo box.

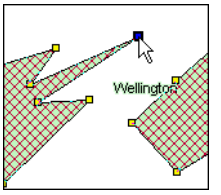


7. Click **OK** to accept your changes. The map display will change and display the object's nodes.



8. Using the Select Item tool, select the polygon you want to modify, then click and drag on a node to change its position. Use the Shift key to select a range of nodes and the Ctrl key to toggle the selected state of a node.

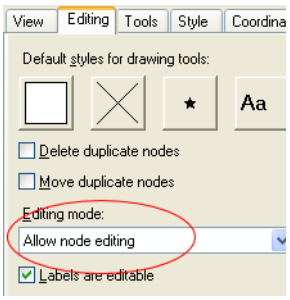




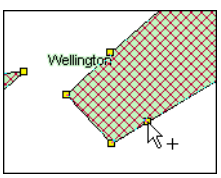
Adding Nodes

To add nodes to a Feature Geometry:

1. Open the LayerControl again and select the root node of the map.
2. On the Editing tab, select Allow node adding from the Editing Mode combo box.



3. Click **OK** to accept your change.
4. On the map, select the polygon you want to modify.
5. Using the Select Item tool, hold down the Ctrl key and click on the edge of the polygon where you want to add the node. The new node will appear.



Reshaping and Adding Nodes Programmatically

This section describes how to reshape and add nodes programmatically. The sample code is provided in C# and VB.

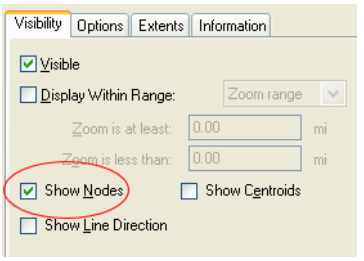
1. Add a button named "btnEditNodeTool" to the main form in the Visual Studio designer.
2. Double-click the button to open the button's handler in the code page. Add the appropriate code sample:

   VB example:

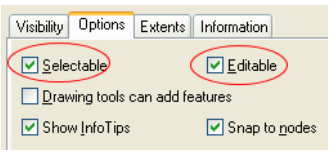
```
Private Sub btnEditNodeTool_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
MapControl1.Tools.LeftButtonTool = "Select"
MapControl1.Tools.SelectMapToolProperties.EditMode =
MapInfo.Tools.EditMode.Nodes
```

```
End Sub
```

3. Run the application.
4. Select the object's layer in the LayerControl.
5. On the Visibility tab of the LayerControl, select the Show Nodes check box.



6. On the Options tab, select the Selectable and Editable check boxes.



7. Click **OK** to accept the change.
8. Now click the new button you added to the form. The cursor will change to the Select arrow.
9. Select an object to modify, then click and drag on a node (as in step 1 of the previous example).

8

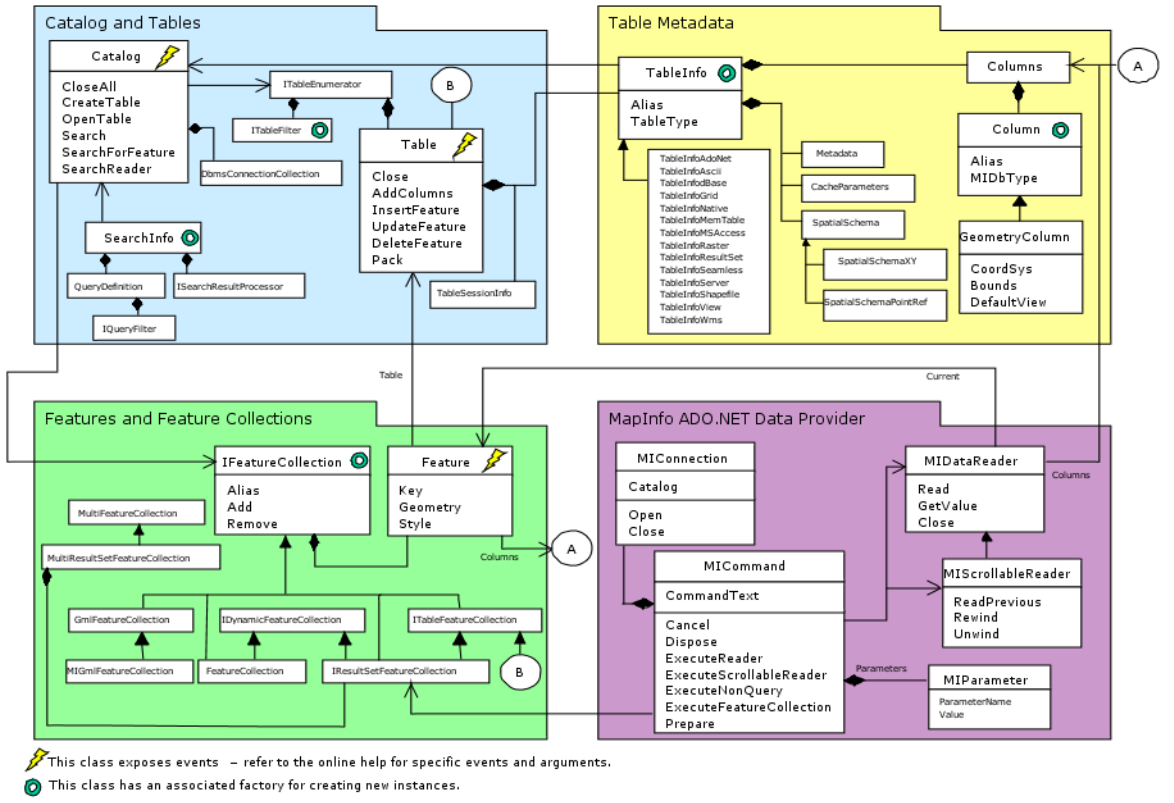
8 – Working with Data

The MapInfo.Data namespace contains the classes and interfaces that provide multiple ways of accessing data from a MapXtreme application.

In this chapter:

Overview of MapInfo.Data Namespace

The MapInfo.Data namespace contains the classes and interfaces that provide multiple ways of accessing data from a MapXtreme application. Within this namespace is the MapInfo ADO.NET data provider with a MapInfo SQL language for standard querying of databases and tables. The Feature object model is another way to access data that uses objects instead of SQL. The Catalog is the starting point for data access, containing methods for managing tables (open, close, create) and searching for data in a variety of ways.



This chapter is organized to follow the MapXtreme Data Access Model diagram above, and includes these topics:

- Catalog and Tables
- Supported Table Types
- Table Metadata (TableInfo)

- MapInfo ADO.NET Data Provider
- Features and Feature Collections

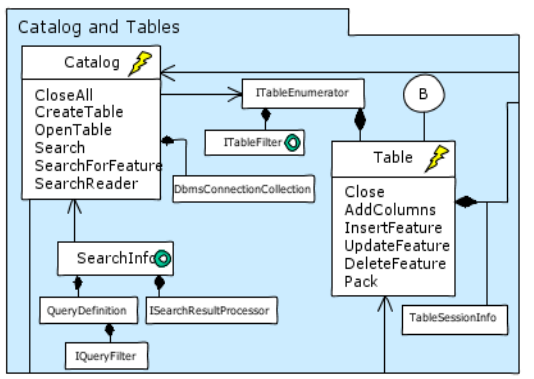
Data access is central to any MapXtreme application, and covers a wide variety of topics. Within the topics listed above are other important topics of information that should not be overlooked.

Following this chapter are two additional chapters related to data access: Chapter 10 Creating Expressions, and Chapter 11 Accessing Data from a DBMS.

Catalog and Tables

Catalog is the manager of the MapXtreme data access model. Tables are a fundamental unit of MapXtreme. Tables hold the data that you want to display and analyze in your application. The Catalog, as manager, holds a list of tables that are currently open in the session. Tables are also opened, created and closed from the Catalog.

Nearly all of MapXtreme's data access operations involve the Catalog and tables.



Tables

The Table class is the basic unit of all data access. Table, Column, and all TAB file metadata information is accessible from a MapInfo Table. Tables may be mappable (contain a column of type FeatureGeometry) or non-mappable. Tables also may be open and accessed without displaying a map.

Table Aliases

When tables are opened, they can be assigned a name (or alias) which is used while the table is open for referencing the table. For example, the table may be referred to by its alias in SQL statements. A table that is opened from a TAB file is assigned a default alias if no alias is specified. The default alias is based upon the name of the TAB file. This property is optional and may be set to null. However, it is good practice to assign an alias.

Columns

A Column object identifies the properties of a column in a table, feature, or feature collection and specifies the column's name (alias), data type, width (for string and decimal columns), and other properties of the column.

Supported data types include:

Data Type	Description
Int	Provides a 32-bit signed integer. This maps to the .NET Framework datatype Int32.
SmallInt	Provides a 16-bit signed integer. This maps to the .NET Framework datatype Int16.
Double	A floating point number within the range of -1.79E +308 through 1.79E +308. This maps to Double.
dBaseDecimal	Provides a floating point number which is treated internally the same as a Double. The dBaseDecimal has a fixed precision and scale when persisted in a table. This is a legacy data type derived, as its name suggests, from the dBase file formats. This maps to Double.
Boolean	Provides a boolean value. This maps to Boolean.
String	Provides a variable-length, null terminated UNICODE string value. This maps to String.
Date * †	Provides a date value. The Date type is implemented as a structure in the MapInfo.Data namespace.
DateTime * ‡	Provides a combined date and time value. The DateTime type is mapped to System.DateTime.

Data Type	Description
Time	Provides a time value. Supports the Time type in MapInfo Professional tables (TAB files) version 9.2 and later. The Time type is implemented as a structure in the MapInfo.Data namespace.
FeatureGeometry	Provides a FeatureGeometry.
Binary	Provides an array of binary data. This maps to an Array of Byte values.
Key	Provides a key from a table. This is the data type of the Key pseudo column on a Table.
CoordSys	Provides a coordinate system. This type exists only for the purposes of binding a coordinate system object to an MICommand for functions which require the specification of a coordinate system.
Style	Provides an instance of a Style class. See MapInfo.Styles.Style. This is the data type of the Style object stored in the style column on a Table.
Raster	Provides a RasterInfo from table's raster column. This is the data type of the RasterInfo object stored in raster column on a Table.
Grid	A GridInfo from table's grid column. This is the data type of the GridInfo object stored in grid column on a Table.
Wms	Provides a WmsClient from the table's Wms column. This is the data type of the WmsClient object stored in the Wms column on a Table.
TileServer	Provides a TileServerInfo from the table’s raster column. This is the data type of the TileServerInfo object stored in the raster column on a Table.

* To ensure backward compatibility with earlier versions of MapXtreme, the MapInfo.Data.MIDataReader.GetDateTime method works with both the DateTime and Date types. In both cases, a System.DateTime value is returned. However, the MapInfo.Data.Column.DataType will reflect the actual data type, either Date or DateTime.

† The Time and DateTime types are not supported for MapInfo SQL functions. However, in MapInfo SQL functions that call/use a DateTime type, the function will return the date portion of the DateTime value. Please see the MapInfo SQL Reference for more information.

‡ The Time and DateTime types are not supported for MapInfo SQL functions. However, in MapInfo SQL functions that call/use a DateTime type, the function will return the date portion of the DateTime value. Please see the MapInfo SQL Reference for more information.

Time and DateTime Data Source Support

MapXtreme can read Date, DateTime, and Time data (and save it back if applied) on the supported data sources and data providers. The different data sources may have different type definitions on date/time, which may or may not match MapXtreme types exactly.

The new data types are supported for the following data sources:

- Mem tables
- Native tables (TAB files)
- ADO.NET
- Oracle via OCI
- MS SQL Server via ODBC

The ASCII and dBase, and Microsoft Access data sources are not supported.

Date and DateTime Support in Remote Databases

Remote databases may not support all the data types that MapXtreme supports. The table below shows the date and time-based types supported in native TAB files and in each supported database.

MapXtreme	Native(X) TAB Files	ADO.NET	Oracle (OCI)	MS SQL Server	GeoPackage
Date	Date		Date		
Time	Time				
DateTime	DateTime	DateTime	DateTime	DateTime	DateTime

Return Column Type and Value Changes in Remote Databases

The addition of the new data types has prompted some changes to the return column types and values for remote databases. The table below shows the return column type and value for MapXtreme 6.7.x and MapXtreme 6.8 for the supported types in each remote database.

Server / Data Type	MapXtreme 6.7.x Return Column Type / Value	MapXtreme 6.8.0 * Return Column Type / Value
SQL Server/DateTime†	Date/System.DateTime	DateTime/System.DateTime
Oracle/TimeStamp	Date/System.DateTime	DateTime/System.DateTime
Oracle/Date	Date/System.DateTime	Date/MapInfo.Data.Date

* applies to v 6.8.0 and higher.
† SQL Server 2005 and earlier.

The following sections provide you with MI_Key, MI_Geometry, and MI_Style column information.

MI_Key

All tables have a pseudo column named MI_Key which returns instances of Key. The MI_Key pseudo column is similar in concept to the rowid pseudo column in MapInfo Professional and MapBasic. Unlike rowid, this column is not a numeric column. A Key instance may be converted to or from a string literal.

MI_Geometry

A Geometry column object in a table, feature, or feature collection contains FeatureGeometry objects and specifies properties such as the coordinate system of the column and the entire bounds of all the geometry objects it contains.

Geometry columns for most table types are given the name “Obj”. To be compatible with previous versions of MapX and MapInfo Professional, the alias “Obj” is resolved to the first GeometryColumn in the table. Additionally, the alias “MI_Geometry” may also be used for any table to refer to the same column that “Obj” refers to.

MI_Style

Tables with a Geometry column also have a column with the name “MI_Style”, or if not found, from the first column with type MIDbType.Style. This column is used to hold the style information for Geometry objects such as line width for polygons and symbol size for points. This column cannot be updated independently. The Style and Geometry columns must be updated at the same time.

The MI_Style column is created automatically when you are opening a table in MapInfo native format (.TAB). For all other table types, you must specifically create the column. If you use MapInfo.Data.ColumnFactory.CreateStyleColumn it will create a column with the name (alias) of "MI_Style" and a data type of MIDbType.Style.

When using MISQL to insert rows into a table, be sure to include the MI_Style column in the insert statement. See the code example below:

```
Table tab = MapInfo.Engine.Session.Current.Catalog.GetTable("MapViewer");
        TableInfo ti = TableInfoFactory.CreateTemp("Test",
((MapInfo.Data.GeometryColumn)tab.TableInfo.Columns["Obj"]).CoordSys);
        Table tabTemp = MapInfo.Engine.Session.Current.Catalog.CreateTable(ti);

MIConnection conn = new MIConnection();
conn.Open();
MICommand comm = conn.CreateCommand();
comm.CommandText = "Insert Into " + tabTemp.Alias +
   " (Obj, MI_Style) SELECT MI_Point(MI_X(Obj), MI_Y(Obj), '" +
   ((MapInfo.Data.GeometryColumn)tab.TableInfo.Columns["Obj"]).Coord
   Sys.SrsString + "'), MI_Style" + " FROM " + tab.Alias + " WHERE msaname
   = 'Minneapolis-St. Paul, MN-WI' AND Not Obj = Null";
MessageBox.Show(comm.CommandText);
int numChanged = comm.ExecuteNonQuery();

mapControl1.Map.Layers.Add(new FeatureLayer(tabTemp));
mapControl1.Map.SetView(mapControl1.Map.Layers["Test"] as FeatureLayer);
```

Catalog

The Catalog is essentially the manager of the MapXtreme data access model. The Catalog holds a list of tables that are currently open in the MapXtreme Session. Tables are also opened, created and closed from the Catalog. The Catalog can be thought of as a single database holding all the tables opened in it, regardless of their actual data source.

Each MapXtreme Session manages a single Catalog.

Catalog initially contains no tables. When a table is opened, an alias (or name) is assigned to the table or provided by the caller. The alias is used to identify the table in queries and other operations.

Tables can be mappable (contain a spatial component) or be non-mappable and contain only data columns. The MapXtreme Catalog can open both types and use either in queries and joins.

Catalog provides facilities for creating new table definitions and enumerating through tables which are currently opened. Catalog also contains search methods that can be used to access data in open tables.

The Catalog has an SQL engine that allows you to select, insert, update, and delete tables and data within tables. The SQL engine allows you to join any tables defined in the catalog (for example, Native to SQLServer, or SQLServer to Oracle). The Catalog handles the integration from various sources so you don't have to. This is a powerful tool when organizing data from various sources.

The MapXtreme Catalog is exposed through the MapInfo ADO.NET Data Provider. Access to tables and result sets is controlled through this interface. See MapInfo ADO.NET Data Provider.

Code Sample

The following example illustrates how to access the Catalog through the MapXtreme Session object, open some tables and enumerate through all the tables in the Catalog followed by only the editable tables in the Catalog.

VB example:

```
Public Shared Sub MapInfo_Data_Catalog()
    ' Catalog is accessible off the Session object
    Dim catalog As Catalog = Session.Current.Catalog

    ' Open a bunch of tables
    Dim table As Table = catalog.OpenTable("States.tab")
    table.SessionInfo.ReadOnly = True ' Make states ReadOnly
    table = catalog.OpenTable("world.tab")
    table = catalog.OpenTable("worldcap.tab", "World Capitals")

    ' Enumerate the catalog directly - includes All tables
    Dim t As Table
    For Each t In catalog
        Console.Out.WriteLine("Table : {0}", t.Alias)
    Next
    Console.Out.WriteLine()

    ' Now enumerate through only tables that are editable (not ReadOnly)
    Dim tEnum As ITableEnumerator = _
    catalog.EnumerateTables(TableFilterFactory.FilterEditableTables())
    While tEnum.MoveNext()
        Console.Out.WriteLine("Table: {0}", tEnum.Current.Alias)
    End While

    Session.Current.Catalog.CloseAll()
End Sub
```

Supported Table Types

One of the strengths of MapXtreme is its ability to access data "where it lives." This means we strive to handle a wide variety of data formats. Here are the supported table types in MapXtreme:

MapInfo .TAB format	MapInfo native table format. This file-based table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .DAT file. TAB is available as a storage format to be used when caching. See Creating a New Table.
dBase	Data stored in a dBase file. The table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .DBF file. An associated .IND file holds one or more B-Tree indices for non-spatial attribute values (strings, numbers, and dates)
ASCII	Data stored in a delimited .CSV or text file. The maximum string length is 255 characters (including up to two quotation marks). ASCII tables are Insert only. The table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .CSV or .TXT file.
MS Access	Microsoft Access database table. This file-based table located inside of a Microsoft Access .MDB database may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in an Access file.

Shapefile	An ESRI Shapefile table. These tables are read-only and support three-dimensional geometries (X, Y, Z, M). Non-spatial attribute data is stored in .DBF file format. FeatureGeometry values stored in ESRI .shp file format. MapXtreme does not have access to the spatial index. Caching is supported as a .MAP file that can be temporary or persistent. A persistent cache can be shared with MapInfo Professional. It is controlled by the PersistentCache property on the TableInfoShapefile class.
MemTable	An in-memory storage of non-spatial attribute data. FeatureGeometry data and indices are stored on disk. These are temporary tables—all data is lost when the table is closed. MemTables are serializable. The data can be persisted in workspaces (data only; keys may be changed when reloading). This table type is available as a storage format to be used when caching. There is no .TAB file equivalent to a MemTable. See Create a Temporary MemTable.
RDBMS Server	A spatial table stored in a remote database management system (such as SQL Server or Oracle). The table is defined by a native SQL SELECT statement. MapXtreme performs query parsing and modification. Caching is enabled by default. Supported protocols (toolkits) include: OCI (Oracle Spatial) and ODBC (, SQL Server, SpatialWare and XY). See Chapter 11 Accessing Data from a DBMS. See Installation Requirements for list of supported RDBMS.

ADO.NET	A table of non-spatial data that is based upon an ADO.NET DataTable or IDbCommand. This table type supports many different data providers with provider-specific implementations. ADO.NET is the choice when there is no MapXtreme supported data provider. ADO.NET is designed to support both Connected (IDBCommand) and Disconnected (DataTable) ADO.NET models. IDBTables are read-only. Cache may be applied forcibly (implicit keys). DataTables are editable and run-time serialization is supported. See Using an ADO.NET Data Provider.
Raster	A table containing a raster image. This typically provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (RasterInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). See Chapter 17 Working with Rasters and Grids.
Grid	A table containing a grid image. This table type provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (GridInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). GridRead class provides access to grid cell values. MapInfo.Raster.GridCreatorFromFeatures class creates a grid using an interpolator. See Chapter 17 Working with Rasters and Grids.
WMS	A table containing an image from a Web Map Service (WMS). This table type provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (GridInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). WMS tables are accessed like dynamic raster through a MapInfo.Wms.WmsClient. See Chapter 24 Web Map Service.

Seamless	A table that combines two or more base tables with contiguous geography. It displays as a single map layer. Seamless tables are specifically tuned for spatial queries, such as drawing a map, which uses seamless tables for optimally querying appropriate component tables. Component tables that make up a seamless table may be vector or raster. They must all have the same schema. They are read-only. The underlying component tables cannot be modified directly. Sorting and aggregating operations examine every record of every component tables (could be have a significant performance impact when working with vector tables.)
View	A view based on a MapInfo SQL Select statement (not a native SQL supported by Server tables). See View Tables.
ResultSet	A table containing the results of a search. ResultSet is used exclusively for IResultSetFeatureCollections. See Result Sets.
TileServer	A table containing a TileServer image. This typically provides a base map for other spatial table types. Tables contain a single record and a fixed column schema (TileServerInfo, MI_Geometry, MI_Style).
Geopackage	A table containing information in a Geopackage format. Indicates a table that has both FeatureGeometry objects and attribute data stored in an OGC Geopackage database file format.
NativeX	MapInfo Extended (NativeX) Tab file formats. This file-based table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .DAT file. TAB is available as a storage format to be used when caching. The NativeX format supports table caches larger than 2GB in size and character sets UTF-8 and UTF-16.

Working with Catalog and Tables

This section covers some basic table operations, including:

- Locating Open Tables
- Closing a Table
- Packing a Table
- Listening to Table and Catalog Events

See also the MapInfo.Data.Table class in the MapXtreme Developer Reference.

Locating Open Tables

To locate open tables, you must enumerate the catalog. This is done by using the methods in the following sections.

Catalog.GetTable

The MapInfo.Data.Catalog.GetTable method returns the Table object referenced by the TableAlias parameter. This must be a table which has already been opened. If no such table is found (or the table has subsequently been closed), then the method returns null.

Catalog.Item (Indexer)

MapInfo.Data.Catalog.Item property can be used as an indexer for locating a Table by its Alias. This is functionally equivalent to using the Catalog.GetTable method but generates code that is easier to read. The Alias must specify a table which has already been opened.

VB example:

```
Public Shared Sub MapInfo_Data_Catalog2()

    Dim tbl As Table
    For Each tbl In Session.Current.Catalog
        System.Console.WriteLine("Table: " + tbl.Alias)
    Next

End Sub
```

TableEnumerators

Table enumerators may be obtained through the various overloaded EnumerateTables methods. A table enumerator may be created with a filter. The filter determines which tables are actually included in the enumeration while the enumerator simply provides the

mechanics of enumeration. You can create your own table filters to use in the TableEnumerator. You can also create your own table enumerator by implementing the ITableEnumerator interface.

VB example:

```
Public Shared Sub MapInfo_Data_Catalog3(ByVal catalog As Catalog)
    Dim te As ITableEnumerator = _
catalog.EnumerateTables(TableFilterFactory.FilterEditableTables())

    While te.MoveNext()
        Dim tbl As Table = te.Current

    End While
End Sub
```

Closing a Table

Three methods are available to close tables. MapInfo.Data.Catalog.CloseAll closes all open tables while Catalog.CloseTable closes a single, open table. The Table class also has a Close method.

Packing a Table

The MapInfo.Data.Table.Pack method removes records from the table that were previously marked for deletion. When the table is packed, the table's TablePacked event is raised. The arguments for the event indicate whether or not the table's keys changed as a result of the pack (which would be caused by removing deleted records). Keys only change if the PackType includes RemoveDeletedRecords and if there actually were deleted records in the middle of the table. If the only deleted records in the table are at the end of the table, then no keys are changed. The event does not indicate that keys were changed.

ⓘ Since ResultSet tables hold collections of keys, these are vulnerable to pack operations on the table from which they were derived. The ResultSet is no longer valid if the keys have changed.

PackType Enumeration provides you with the following options.

- PackGeometry – Indicates that the geometry objects are packed. Packing the objects attempts to remove as much unused space as possible. A fully packed RTree (the spatial index used to spatially access the geometry objects) may reduce performance by causing many more unnecessary reads. To balance disk space and processing

speed, packing the geometry objects may continue to leave some unused space in the RTree. Also note: a packed RTree results in a slight performance penalty for insert and update operations as there is a higher likelihood that the RTree needs to be expanded.

- RebuildGeometry – Rebuilding the geometry objects removes unused space that has resulted from a series of insert, update, and/or delete operations. Unlike packing the geometry objects, this option intentionally leaves unused space in the RTree index to improve the performance of future insert and update operations.
- PackIndex – Non-spatial indices are maintained as B*trees. These structures do not always have filled internal or leaf nodes. This is intentional by default to allow room for the index to accommodate insert and update operations without requiring a significant restructuring of the index. The unused space may be exacerbated by the occurrence of insert, update, or delete operations. Packing an index fully packs every internal and leaf node (except possibly the “last” node). This option reduces the disk space used by the index as much as possible and also improves the read-performance of the index. There is a performance penalty for insert and update operations on a fully packed index.
- RebuildIndex – Rebuilding an index does not fully pack the internal and leaf nodes like the PackIndex option. Instead, rebuilding an index recreates the index with the amount of unused space that is intentionally put into the index to balance disk space, read performance, and modify performance. After several modification operations, an index may contain a considerable amount of unused space. This option regains that unused space.
- RemoveDeletedRecords – Some data sources, including MapInfo Native and dBase data sources, do not physically remove records when they are deleted. To physically remove the deleted records, the table must be packed with this option specified. The record number is typically used as the record key for these data source types. Removing deleted records from a table may cause keys to become invalid since they may change as a result of the pack.
- CompactDb – If the table's data source is Microsoft Access (TableType of Access), then the MDB file containing the table's data may also be compressed using the Pack method and specifying this option.
- All – This is a convenience option that is equivalent to PackGeometry | PackIndex | RemoveDeletedRecords.

Listening to Table and Catalog Events

Table exposes several events which applications may subscribe to. They are:

- RowInsertedEvent – Occurs when a new row is added to the table.
- RowUpdatedEvent – Occurs when an existing row in the table is updated.
- RowDeletedEvent – Occurs when a row in the table is deleted.

- TablePackedEvent – Occurs when the table is packed.
- TableCloseRequestEvent – Occurs when the table has been asked to close.
- TableIsClosingEvent – Occurs when the table is closing.
- TableClosedEvent – Occurs when the table is closed.

Catalog also exposes the following events.

- TableOpenedEvent – Occurs when a table is opened.
- TableCreatedEvent – Occurs when a new table is created.
- TableIsClosingEvent – Occurs when the table is closing.

Table Metadata (TableInfo)

The TableInfo class in the MapInfo.Data namespace is an abstract base class that contains information, or metadata, about an existing table, including:

- Columns – number, names, data types, etc.
- Table alias, and description and pathname of the data source.
- Client metadata (the information between the begin_metadata/end_metadata tags in the TAB file).

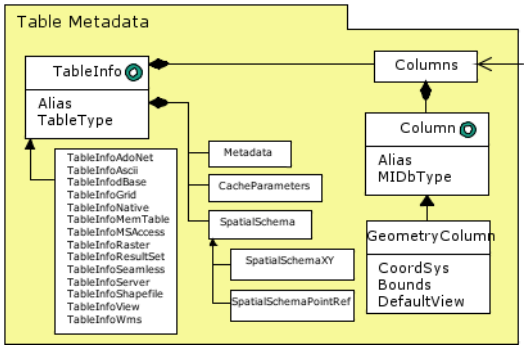
TableInfo is used to open tables and create new tables. It is also used for retrieving information about the open table.

Classes that derive from TableInfo include provider-specific metadata. There is a TableInfo implementation for every MapXtreme supported table type. See Data Sources.

TableInfo instances may be constructed manually, or from a .TAB file definition (without opening the table), as shown below.

```
TableInfo.CreateFromFile(…)
```

TableInfo contains properties for enabling Table Services, including caching and making a table mappable via a spatial schema. See Working with the Cache and Making Tables Mappable.



MapXtreme provides table column metadata support for M and Z values. This feature is useful when you want to know whether geometries of a particular data provider can support 3D and Measured values without evaluating its individual geometries.

Metadata for a table can be accessed from the table's TableInfo property. From the table metadata you can access the GeometryColumn to interrogate if the table supports M or Z values and what the range of values for that table is if the range is known. For more information on support for M and Z values, see Support for M and Z Values.

Examining TAB File Metadata

TAB file metadata is accessible and editable. The TableInfo class can be obtained from the Table to get information about the table structure.

The following code demonstrates how to get the metadata for an open table. The code also demonstrates how the geometry column can be used to determine the coordinate system and bounds of the table. For a code example that returns M and Z values, see MapInfo.Data.TableInfo in the Developer Reference.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfo2()
   ' Get the metadata for an open table
   Dim ti As TableInfo = Session.Current.Catalog("states").TableInfo

   ' Print out some information to the console
   Console.Out.WriteLine("Table Alias={0}, Datasource={1}, _
      Description={2}, Type={3}", _
   ti.Alias, ti.DataSourceName, ti.Description, ti.TableType)

   ' Print out some information about each column
   Dim col As Column
   For Each col In ti.Columns
   Console.Out.WriteLine("Column {0} Type={1} Width={2}", _
      col.Alias, col.DataType, col.Width)
```

```
   ' If the column is a geometry column, print csys and bounds.
   If col.DataType = MIDbType.FeatureGeometry Then
   Dim geocol As GeometryColumn = col
   Dim csys As MapInfo.GeomeTry.CoordSys = geocol.CoordSys
   Console.Out.WriteLine("CSys : {0}", csys.MapBasicString)
   Dim dr As MapInfo.GeomeTry.DRect = geocol.Bounds
   Console.Out.WriteLine("Bounds=({0},{1}),({2},{3})", dr.x1, _
      dr.y1, dr.x2, dr.y2)
   End If
   Next
```

End Sub

Creating a New Table

The following sections illustrate how to create a permanent native table, a temporary native table, and a temporary MemTable.

Create a New Permanent Native Table

The MapInfo.Data.Table.TableInfo property for a MapInfo native table returns an instance of TableInfoNative. A native table is a MapInfo .TAB file. This class may be used to access properties that are specific to native table types. New instances of this class may be created and used to construct new tables. See also Data Sources.

Note the use of the ColumnFactory class. This is provided to help you know which arguments are necessary for different data types. For example, a geometry column requires a coordinate system.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoNative()
   Dim ti As TableInfoNative = New TableInfoNative("NewTable")
   ti.TablePath = "c:\data\Capitals.TAB"
   ti.Columns.Add(ColumnFactory.CreateIndexedStringColumn("Capital",_
       25))
   ti.Columns.Add(ColumnFactory.CreateStringColumn("Country", 30))
   ti.Columns.Add(ColumnFactory.CreateDoubleColumn("Pop_Grw_Rt"))

' Make the table mappable
   ti.Columns.Add(ColumnFactory.CreateStyleColumn())
   Dim Robinson As CoordSys = _
   Session.Current.CoordSysFactory.CreateFromPrjString("12, _
      62, 7, 0")

ti.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn(Robinson))
' Note we do not need to (nor should we) add a column of type Key.
' Every table automatically contains a column named "MI_Key".
   Dim table As Table = Session.Current.Catalog.CreateTable(ti)
```

End Sub

Create a Temporary Native Table

VB example:

```
Public Shared Sub MapInfo_Data_TableInfo3(ByVal conn As MIConnection)
    Dim ti As TableInfoNative = New TableInfoNative("NewTable")
    ti.Temporary = True
    Dim col As Column

    col = New Column
    col.Alias = "FString30"
    col.DataType = MIDbType.String
    col.Indexed = True
    col.Width = 30
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FInt32"
    col.DataType = MIDbType.Int
    col.Indexed = True
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FInt16"
    col.DataType = MIDbType.SmallInt
    col.Indexed = True
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FDouble"
    col.DataType = MIDbType.Double
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FDateTime"
    col.DataType = MIDbType.Date
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FBoolean"
    col.DataType = MIDbType.Boolean
    ti.Columns.Add(col)
    ' Note we do not need to (nor should we) add a column of type Key.
    ' Every table automatically contains a column named "MI_Key".
    Dim miTable As Table = conn.Catalog.CreateTable(ti)
End Sub
```

Create a Temporary MemTable

The MapInfo.Data.Table.TableInfo property for a memory table returns an instance of TableInfoMemTable. This class may be used to access properties that are specific to memory table types. New instances of this class may be created and used to construct new tables.

Data in formats such as XML or GML from a Web service can be brought into the Catalog and used in this fashion. It can be converted to a MultiPolygon, LineString, Point, or other Geometry via the MapXtreme API. MapXtreme then turns the Geometry into a FeatureCollection, and, in turn, saves it to a memTable or native TAB format.

This approach is also appropriate if you wish to make data available for use in MapXtreme, but not necessarily for map display.

MapXtreme supports reading and writing Z and M values to MemTables. M values on MultiCurves allow you to carry out linear referencing operations and dynamic segmentation. See Chapter 22 Linear Referencing.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoMemTable()
   Dim ti As TableInfoMemTable = New TableInfoMemTable("NewTable")
' Note: The TablePath property does not apply - it can be set but it _
' is meaningless.

   ti.Columns.Add(ColumnFactory.CreateIndexedStringColumn("Capital",_
      25))
   ti.Columns.Add(ColumnFactory.CreateStringColumn("Country", 30))
   ti.Columns.Add(ColumnFactory.CreateDoubleColumn("Pop_Grw_Rt"))

' Make the table mappable
   ti.Columns.Add(ColumnFactory.CreateStyleColumn())
   Dim Robinson As CoordSys = _
      Session.Current.CoordSysFactory.CreateFromPrjString("12, 62, _
      7, 0")
   ti.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn(Robinson))
' Note we do not need to (nor should we) add a column of type Key.
' Every table automatically contains a column named "MI_Key".
   Dim table As Table = Session.Current.Catalog.CreateTable(ti)
End Sub
```

Adding Expression Columns to a Table

Use the MapInfo.Data.Table.AddColumns method to add expression columns to a table. The form of AddColumns that takes a Columns object creates temporary columns based on expressions comprised of functions, operators, literal values, and other columns on the table. All instances of Column in the columns argument must have an expression string specified.

ⓘ TableAddColumns is not supported for the following table types: Server, View, Seamless, AdoNet, ResultSet, or Drilldown. MapXtreme checks for the table and throws an exception if it encounters one of these table types.

VB example:

```
Public Shared Sub MapInfo_Data_TableAddColumns(ByVal miTable As Table)
   Dim NewCols As Columns = New Columns
   NewCols.Add(New Column("PopDensity1990", "Pop_1990 / _
      MI_Area(Obj, 'sq mi', 'Spherical')"))
   NewCols.Add(New Column("PopDensity2000", "Pop_2000 / _
      MI_Area(Obj, 'sq mi', 'Spherical')"))
   miTable.AddColumns(NewCols)
End Sub
```

The expression string “Pop_1990 / MI_Area(Obj, 'sq mi', 'Spherical')” represents derived information that will be placed in the temporary column. It says ‘For each record divide population by area in square miles to yield the population density.’ The SQL function MI_Area () in the expression will derive the area from the geometry of the record.

Using the AddColumns method may offer performance improvements in desktop applications where the join can be performed once, rather than on each subsequent access (as in the case of a view).

For more information and code examples, see the MapInfo.Data.Table.AddColumns class in the Developer Reference Help system.

For more information on creating expressions, see Chapter 10 Creating Expressions.

Data Sources

The following table lists the data sources supported by MapXtreme. Each type of data source is accessed by a specific data provider called TableInfo class, that is derived from MapInfo.Data.TableInfo. For a short summary of each data type see Supported Table Types.

Data Source	Class
Native (MapInfo.TAB)	TableInfoNative
dBase	TableInfodBase
MS Access	TableInfoMSAccess
ASCII	TableInfoAscii
RSBMS Server	TableInfoServer
ESRI Shapefile	TableInfoShapefile
Seamless	TableInfoSeamless
Raster	TableInfoRaster
Grid	TableInfoGrid
WMS	TableInfoWMS
ADONET	TableInfoAdoNet
MemTable	TableInfoMemTable
View	TableInfoView
ResultSet	TableInfoResultSet
TileServer	TableInfoTileServer
GeoPackage	TableInfoGeoPackage
NativeX	TableInfoNativeX

Choosing the Correct Data Source

Choosing the correct data source can make a difference in your application's performance. In some cases you will not have a choice, such as native MapInfo files (.TAB), but in other cases there may be multiple choices. In most cases, you will be using a supported data provider for the data source. In cases where the data is not accessible through one of these, you may be able to use the MapInfo ADO.NET data provider. This is the same data provider interface that the Catalog uses to retrieve data.

Each data source has certain performance characteristics. Native tables offer the best access and map drawing times. Data is stored locally on the system and optimized for your current operation. Other file-based table types perform well, depending on current hardware and file size.

Methods for Accessing Data

MapXtreme provides several ways to bring data into the Catalog:

- Direct access to data sources
- Access via an ADO.NET data provider (TableInfoAdoNet)
- XML/GML from third-party web services

The best method to access data is to open it directly using one of the TableInfo classes that are specific to where your data resides.

Use the second method (TableInfoAdoNet) to access data that is not internally supported but has an ADO.NET provider.

A third method allows developers to integrate data to the Catalog who may interact with HTTP services that return XML or GML.

Direct Access to Data Sources

MapXtreme provides native support for accessing data stored in file-based table formats and RDBMS servers, such as SQL Server and Oracle. In the case of file-based access, provide the path and filename in the appropriate TableInfo instance (TableInfoNative, TableInfodBase, TableInfoMSAssess, TableInfoAscii, TableInfoGeopackage, and TableInfoNativeX).

For direct access to data stored in RDBMS serves, use the TableInfoServer class to define the connection string and an SQL statement to execute on the remote table. Internally, MapXtreme uses ODBC or OCI to access the remote database.

TableInfoServer will open a connection to the server, query the table's metadata, and create the appropriate table definition with any spatial characteristics that are defined on the remote server. This tends to be the best performing method with remote data.

Internally, MapXtreme can access only the data necessary to perform the current operation. During a map draw, MapXtreme will construct a query that returns only the geometry column, and not the data columns. This minimizes the network traffic. If caching is on, then this is only an issue for the first access, since all subsequent requests will come from the cache. See Chapter 11 Accessing Data from a DBMS.

Access via an ADO.NET DataProvider

The second data access method is to use an ADO.NET data provider. This requires the definition of ADO.NET classes for data retrieval. Only non-mappable tables may be supplied as an AdoNet table. Non-mappable tables are those that do not contain geometry information about the data. Tables retrieved from an ADO.NET provider, however, can be made mappable by applying a SpatialSchema to the table definition. In this method, the MapXtreme DataAccess engine calls the ADO.NET data provider whenever data is requested by a user. This tends to be a slower method of accessing data. However, when used in conjunction with caching, it performs well. See Using an ADO.NET Data Provider.

Data from Third-Party Web Services

MapXtreme can integrate Web service XML or GML output into the Catalog for use in a MapXtreme desktop or web application. Data can be brought into the Catalog and converted to a MultiPolygon, LineString, Point, or other Geometry via the MapXtreme API. MapXtreme then turns the Geometry into a FeatureCollection, and, in turn, saves it to a memTable or native TAB format.

This approach is appropriate also if you wish to make data available for use in MapXtreme, but not necessarily for map display.

Data Readers, MemTables and Result Sets

The methods to access data return a data reader or result set. A data reader allows access in a sequential manner and does not store copies of data. It retrieves the data from the data source, except in the case where the data source is cached. Result sets are collections of keys. These keys allow you access back to the original tables and do not create copies of the data.

A MemTable also allows you to store data from various sources into one table. This table type stores data in a combination of memory arrays and temporary disk storage. When data is added, the MemTable makes a copy of the data and does not have a key or pointer back to the original table. These are useful for temporary layers for maps and containers for return values of processes such as a geocoding or routing result. MemTable access and map rendering performance is equivalent to native tables.

Result sets are a great tool when you need access to a defined set of rows and when you need to get data from the source. If the source data may change during your session then this method allows you to see the results if the data source supports concurrent access. Since MemTables are copies of data they are a static set of data rows and will not reflect changes from the original data sources.

Using an ADO.NET Data Provider

Data that cannot be directly accessed with a specific TableInfo data source can use TableInfoAdoNet. The ADO.NET table can be in one of two forms: DataTable (a collection of rows from a single table kept in-memory and allows read-write access); or IDbCommand (an SQL statement executed at the data source that yields read only, dynamic data).

Accessing Data in a DataTable

When using a DataTable, the Catalog is essentially holding on to a reference to the DataTable you supply to the call to Catalog.OpenTable (using the TableInfoAdoNet class). DataTables are editable using the MapInfo ADO.NET Data Provider by issuing Insert, Update, and/or Delete commands. Your application may continue to access the DataTable directly as well. Note, however, that the structure of the table should not be changed while the Catalog has a reference to it. Also note that changes to the data outside of the MapInfo Data Provider (e.g., without using the MICommand to issue Insert, Update, or Delete commands) will not result in the raising of the insert, update, or delete table events.

The DataTable contains almost enough information for the Catalog to define the table. For string columns, however, the Catalog needs to assign a length to this field. The length would be used when constructing temporary indices, temporary tables for aggregation, etc. For these types of operations, it is important to get the string length correct. The DataColumn has a MaxLength property that should be set to indicate the maximum length string the column could hold. If not set, this value defaults to -1 in which case the value of 254 is used. Before checking the MaxLength property, the Catalog looks to see if the DataColumn has a property defined in its ExtendedProperties collection with the name “StringWidth”. If found, the value for this property is used as the column's width.

This example illustrates how to create a MapInfo Table whose data is stored in a DataTable.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoAdoNet(ByVal connection As _
     MIConnection)
   ' Create a new DataTable.
```

```
   Dim dt As DataTable = New DataTable("CityData")
   Dim dc As DataColumn
   dc = dt.Columns.Add("City", Type.GetType("string"))
   dc.MaxLength = 30
   dc = dt.Columns.Add("Country", Type.GetType("string"))
   dc.MaxLength = 30
   dc = dt.Columns.Add("Continent", Type.GetType("string"))
   dc.MaxLength = 30
   dc = dt.Columns.Add("Population", Type.GetType("string"))

    ' Populate the DataTable...
dt.Rows.Add(New Object() {"Madrid", "Spain", "Europe", 1500000})
dt.Rows.Add(New Object() {"Stockholm", "Sweden", "Europe". 985000})

    ' Now open a MapInfo Table which accesses this DataTable
    Dim ti As TableInfoAdoNet = New TableInfoAdoNet("Cities")
    ti.ReadOnly = False
    ti.DataTable = dt
    Dim table As Table = connection.Catalog.OpenTable(ti)
End Sub
```

Saving and Restoring ADO.NET objects

Saving and Restoring ADO.NET tables can be accomplished in MapXtreme by using the steps outlined below. Explicit serialization/deserialization of ADO.NET tables is not supported due to limitations on restoring the underlying System.Data.DataTable. An ADO.NET table is a table in the Catalog which was created using a MapInfo.Data.TableInfoAdoNet object.

The proper method of serializing/deserializing the ADO.NET based MapInfo table is in the SaveState method. Serialize all tables that reference the ADO.NET table (i.e., ViewTables or joins) and then close the ADO.NET table. In the RestoreState method, re-create the ADO.NET MapInfo table with the same name and then deserialize any dependent MapInfo tables. Order is important because you have to create the ADO.NET table prior to restoring any other tables.

In the context of a MapXtreme Web Application which implements manual state management, follow these steps to save and restore an ADO.NET table between client requests.

ⓘ The steps outlined below refer specifically to ADO.NET tables created from a DataTable.

---

1. In the MapInfo.WebControls.StateManager.SaveState method:
   a. Place the ADO.NET DataTable or DataSet into the HTTPSession instance.

b. Save any MapInfo.Data.TableInfoView or MapInfo.Data.TableInfoResultSet tables that are dependent on the ADO.NET table using the ManualSerializer.SaveMapXtremeObjectIntoHttpSession.

c. Close the ADO.NET table in the Catalog.

2. In the MapInfo.WebControls.StateManager.RestoreState method:

a. Create a new TableInfoAdoNet object based on the DataTable retrieved from the HTTPSession instance.

b. Open a new ADO.NET table in the catalog based on the TableInfoAdoNet object with the same name as the original.

c. Restore any MapInfo.Data.TableInfoView or MapInfo.Data.TableInfoResultSet tables based on the ADO.NET table using ManualSerializer.RestoreMapXtremeObjectFromHttpSession.

It is important to remember that the order you save and restore is critical to the proper creation of all the tables and their dependencies. For more information on serialization, see Chapter 6 Understanding State Management.

Accessing Data Using an IDbCommand

The second form of ADO.NET table is based on the connected object types in ADO.NET: Connection, Command, and DataReader. MapInfo Tables constructed in this fashion are read-only. These types of tables are created by passing to the Catalog an IDbCommand object that is already configured to return all of the data that is to comprise the table. When the table is initially created (by calling Catalog.OpenTable), ExecuteReader is called on the IDbCommand. The resulting data reader is used to determine the columns and their data types. All subsequent cursor requests (other than cursors which retrieve a specific record - called a key fetch) also call ExecuteReader to fetch the data to satisfy the cursor. Notice that this may be very inefficient. If at all possible, use one of the other table types to access your data.

Since the Command-based form of the ADO.NET table is designed to use the generic interfaces without requiring any specific knowledge of any particular implementation of these interfaces, the table also does not assume that the IDbCommand.CommandText is any form of standard SQL. In fact, it may not be SQL at all. This table type does not access, parse, or modify the CommandText. This means that this table type has no mechanism for knowing which column(s) in the results formulate a unique, non-null key value. For this type of table, it is required to tell the table which column(s) constitute the key. This is accomplished by specifying the KeyType as Explicit and setting the KeyColumns property.

There are many operations inside the MapInfo Data Provider which require the retrieval of a specific record by key (also referred to as a key fetch). Select statements with a where clause of the form MI_Key = '5' is a simple example in which we need to retrieve the record whose MI_Key column can be represented by the string literal '5'. Key retrievals are very common in mapping selections, labeling, and scrolling in a MIScrollableReader (in which case the reader may be scrolling through a list of key values). MapInfo tables are dependent upon the ability to efficiently fetch records by key value. Just as the Command-based form of the ADO.NET table does not read, parse, or modify the CommandText of the IDbCommand object that defines the table (the “Sequential” IDbCommand), it has no ability to modify the IDbCommand object to fetch a specific record. Thus, a second IDbCommand object must be supplied for this purpose. The “FetchByKey” IDbCommand object must meet the following requirements:

- When ExecuteReader is called on this command object, it must produce a data reader that has the same columns as the sequential command object and in the same order.
- The FetchByKeyCommand must contain a Parameters collection and must contain one parameter for each member of the key. For example, if the TableInfo.KeyColumns specifies a key as consisting of the “city” and “state” columns, then the FetchByKeyCommand must contain two parameter objects. The first parameter object is assigned a value representing the first column specified in the TableInfo.KeyColumns collection (e.g., a value for “city”), the second parameter object is assigned a value representing the second column specified in the TableInfo.KeyColumns collection (e.g., a value for “state”), and so on. When ExecuteReader is called on the FetchByKeyCommand, the reader must return the record which represent s the specified key.

This example illustrates how to create a MapInfo Table that accesses data through the ADO.NET connected command objects.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoAdoNet2(ByVal connection _
      As MIConnection)
   Dim ti As TableInfoAdoNet = New TableInfoAdoNet("EuropeanCities")
   Dim _conn As OleDbConnection = New _
      OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data _
      Source=C:\Data\EuropeCities.mdb")
   Dim selectQuery As String = "SELECT City, Country, Continent, _
      Pop_1994 FROM EuropeCities"
   Dim _OleDbCommand As OleDbCommand = New OleDbCommand(selectQuery)
   _OleDbCommand.Connection = _conn

   selectQuery = selectQuery + " where City = @City AND _
      Country = @Country"
   Dim _OleDbKeyCommand As OleDbCommand = New _
      OleDbCommand(selectQuery)
   _OleDbKeyCommand.Parameters.Add("@City", OleDbType.Char)
```

```
   _OleDbKeyCommand.Parameters.Add("@Country", OleDbType.Char)
   _OleDbKeyCommand.Connection = _conn

' The MapInfo Table will Open/Close the connection as necessary.
' If this is expensive the application could open the connection
' before opening the table and closing the connection after the
' table is closed.
   ti.SequentialCommand = _OleDbCommand
   ti.FetchByKeyCommand = _OleDbKeyCommand

' Tell the table which column(s) constitute a key - for this table
' it is acompound key consisting of values from the City and County
' columns.
   Dim sc As StringCollection = New StringCollection
   sc.Add("City")
   sc.Add("Country")
   ti.KeyColumns = sc
   ti.KeyType = KeyType.Explicit

' Ask the Catalog to open the table.
   Dim tbl As Table = connection.Catalog.OpenTable(ti)

' Now the MICommand object may be used to select data from the table
' (by the name EuropeanCities since that is the alias we assigned to
' it). The data in thistable may be joined with any other table and
' it may be used as source data in a call to AddColumns to populate
' temporary columns with data from this table.
End Sub
```

Data Binding

Data binding is the process of bringing data from a data source into MapXtreme. Data binding of external data (ADO.NET and other legacy sources) to MapInfo.Data.Table is accomplished by opening an ADO.NET DataTable as a Table using TableInfoAdoNet. The table can then be joined with another table, joined to itself or use Table.AddColumns to bind columns to a second table.

To join a table to itself, following this example:

```
Select ... From T as A, T as B Where A.X = B.Y
```

If an application has data stored in a DataTable or data that is accessible through an ADO.NET data provider, that data can be presented to the Catalog and treated as a MapInfo table. This would be primarily useful if the data were not accessible through one of the other table types.

For example, if the data is stored in a dBase file, Microsoft Access table, or is accessible through ODBC or Oracle's OCI interface, it is recommended that those TableInfo types be used to access the data. Data which cannot be accessed through one of these types of

tables, but that can be loaded into a DataTable or is accessible through some ADO.NET Data Provider that implements the Command, Parameter and DataReader object types can still be accessed by the Catalog.

An application may need to make data available as a MapInfo native table so that queries can be executed to join the data with other MapInfo table data. It may also need to be made available to the Catalog so it can be used as the source data in a call to the Table.AddColumns.

Code Example: How to Join Data from an Oracle Table

```
public Shared Sub MapInfo_Data_TableAddColumns5(ByVal map As Map)
   Dim Connection As MapInfo.Data.MIConnection = New _
      MapInfo.Data.MIConnection
   Connection.Open()

 'Add the USA table to the map
   map.Load(New _
      MapInfo.Mapping.MapTableLoader("C:\\maps\\usa.TAB"))

   Dim lyr As MapInfo.Mapping.FeatureLayer = map.Layers("usa")

   ' Open the table from Oracle

   Dim ti As TableInfoServer = New TableInfoServer("StateCapXY",_
      "SRVR=tempest;UID=tn;PWD=tn", "Select * from usa_caps", _
      MapInfo.Data.ServerToolkit.Oci)
   Dim StateCapXY As Table = Connection.Catalog.OpenTable(ti)

   ' Add the Oracle columns to the USA table
   Dim states As Table = Connection.Catalog.GetTable("usa")
   states.AddColumns(Nothing, MapInfo.Data.BindType.Static, _
      StateCapXY, "state", MapInfo.Data.Operator.Equal, "state")

 'Create a ranged theme on the USA layer using a field
 'from the Oracle table
   Dim thm As MapInfo.Mapping.Thematics.RangedTheme = New _
      MapInfo.Mapping.Thematics.RangedTheme(lyr, _
      "pop_1990", "popusa", 4, _
      MapInfo.Mapping.Thematics.DistributionMethod.EqualCount_
      PerRange)
   lyr.Modifiers.Insert(0, thm)

'Create a legend to appear on the map
   Dim legend As MapInfo.Mapping.Legends.Legend = _
      map.Legends.CreateLegend(New Size(5, 5))
   legend.Border = True
   Dim frame As MapInfo.Mapping.Legends.ThemeLegendFrame = _
      MapInfo.Mapping.Legends.LegendFrameFactory.CreateTheme_
      LegendFrame(_"Pop", "pop", thm)
```

```
    legend.Frames.Append(frame)
    frame.Title = "pop"
    map.Adornments.Append(legend)
End Sub
```

View Tables

A view is a way to relate information from one or more tables based on a named select statement. The Catalog allows you to create views based on any table definition. View tables have the following characteristics:

- The data is not copied.
- Access to views always accesses its base tables.
- View is an MapInfo SQL Select Statement with a name (Alias).
- Queries may be joins (forms composite keys).
- Membership in the View is live.
- Exception: Views that aggregate cache the data. Data changed events trigger recomputation.
- Can be serialized and persisted in workspaces.

For more information and code examples, see the MapInfo.Data.TableInfoView class in the Developer Reference Help system.

Result Sets

ResultSets are similar to view tables in that both are defined using a MapInfo SQL select statement and have an associated name (Alias). ResultSets, however, have a fixed membership of records based on the evaluation of the where clause (if any) at the time the result set is created. Any access to the data in a ResultSet always reflects the data in the source table. However changes to the source data will not cause the ResultSet to add/remove a record based on the original where clause. ResultSets manage a set of keys internally.

In general ResultSets are lightweight and temporary. Some of the characteristics of result sets are:

- The data is not copied.
- Access to result sets always accesses its base tables.
- A ResultSet is a sorted list of keys, a collection of column definitions, and a name.
- Membership in the ResultSet is fixed.
- Exception: ResultSets that aggregate, cache the data. Data changed events trigger recomputation.

- Can be serialized, but not persisted in workspaces.
- ResultSets are vulnerable to Delete and Pack operations.

For more information, see the MapInfo.Data.TableInfoResultSet class in the Developer Reference Help system.

Source Rows

Source rows represent a match between the table records involved in Table.AddColumns. When adding temporary columns to a table, multiple records from the data source may be aggregated together to compute a value for each record in the destination table (also referred to as the bind table). The MapInfo.Data.SourceRows class is a collection of SourceRows that identify the records from the data source that were aggregated together,

SourceRows only exist if the BindType property is DynamicCopy, which indicates that changes to the source data are propagated to the temporary column automatically.

ⓘ Table.AddColumns is not supported for the following table types: Server, View, Seamless, AdoNet, ResultSet, or Drilldown.

See also Adding Expression Columns to a Table.

The GeoDictionary

The GeoDictionary maintains information about which map entities can be matched to which information. The GeoDictionaries class is a collection of GeoDictionary objects. The MapInfo.Data.GeoDictionary namespace provides support for data autobinding by being a programmatic representation of the GeoDictionary file. The GeoDictionary file contains information about tables (TAB files only). The GeoDictionary is used to automatically determine the table to which application data should be bound. The GeoDictionary is persisted in a file (typically GeoDict.DCT) and is maintained using the GeoDictionaryManager utility application (see Chapter 28 Using the GeoDictionary Manager).

AutoMatching Using the GeoDictionary

The MatchResolver.AutoMatch method in the Data.GeoDictionary namespace initiates the AutoMatching process. It does not call AddColumns, i.e., does not do the binding. A subsequent call to BindColumn is required to perform the autobinding, or a direct call to AutoMatchAndBind.

Automatching can encounter ambiguous situations. These situations include:

- multiple source columns are detected in the user data
- multiple tables/layers are detected that match the source column
- multiple geosets/workspaces are available for the matched table/layer.

It is the MatchResolver object with which the GeoDictionary communicates during the match process to solve the ambiguity. It provides the matching algorithm. The basic class selects the first or the one with the highest matching percentage. This class is not sealed and client applications may derive their own class from this and override its behavior.

Making Tables Mappable

Tables can either be mappable (contain a GeometryColumn) or non-mappable (no spatial attribute data). Mappable tables are added to a MapXtreme application as a layer in a map. Non-mappable tables, such as customer data, can be made mappable when a GeometryColumn is created for it. MapXtreme provides spatial schemas to accomplish this.

Spatial schemas are services that can be applied to a table to enhance its spatial capabilities. There are two type of spatial schemas: XY and PointRef. Non-mappable tables that have attribute columns that represent X and Y values (such as longitude and latitude) use SpatialSchemaXY and tables that have an attribute column which can be used to reference a record in a mappable table uses SpatialSchemaPointRef.

SpatialSchemaXY

SpatialSchemaXY uses the X and Y values of each record in the table to construct point objects and store them in a temporary column known as MI_Geometry. This spatial schema may be applied to tables of any data source except Seamless, View, and ResultSet.

By having a GeometryColumn, the table can now be displayed as a layer in a Map and used for spatial analysis.

SpatialSchemaXY has the following characteristics:

- The Geometry column is editable.
- Editing the Geometry automatically changes the X and Y values.
- You can define styles for each point in the table.
- You can store the spatial information as a TAB file and open like any other table.

This spatial schema can be used for traditional server XY data without a MapCatalog. (Using a MapCatalog may offer better performance on RDBMS's, since more work is done on the server. See The MapInfo_MapCatalog.)

MI_Geometry is a temporary column unless you write out the TAB file explicitly using the TableInfo.WriteToTab method. The schema is automatically regenerated when the table is opened.

VB example:

```
Public Shared Sub MapInfo_Data_SpatialSchemaXY()
Dim ti As TableInfo = _
   TableInfo.CreateFromFile("c:\data\customers.TAB")
      ' a non-mappable table
   Dim xy As SpatialSchemaXY = New SpatialSchemaXY
   xy.XColumn = "Xcoord"
   xy.YColumn = "Ycoord"
   xy.NullPoint = "0.0, 0.0"
      ' Any customer at 0,0 means we don't know their location.
   xy.StyleType = StyleType.None
   xy.CoordSys = _
      Session.Current.CoordSysFactory.CreateLongLat(DatumID.WGS84)
   ti.SpatialSchema = xy
      ' Now set the spatial schema information before
      ' opening the table.
   Dim table As Table = Session.Current.Catalog.OpenTable(ti)
End Sub

Public Shared Sub MapInfo_Data_TableInfoNative2(ByVal ti As _
TableInfoNative)
    ti.WriteTabFile()
End Sub
```

SpatialSchemaPointRef

This spatial schema uses a value in the table's data to create a Point geometry object by matching the value against an equivalent value in a mappable table.

For example, if your table of customers contains addresses with postal codes, the customer records can be tied to the spatial points in a postal code reference table.

SpatialSchemaPointRef is actually a join between two tables, one containing data and the other containing a join column and an object column. The join column contains the same values as the data column in the non-mappable table, such as postal codes. The result of applying SpatialSchemaPointRef is a table that contains a spatial geometry column for records that were previously non-spatial. This geometry column has the following characteristics:

- The data table may match more than one record in the geometry table. When this happens the similar rows are aggregated into a MultiPoint geometry.
- The geometry is the centroid of the geometry from the other table.

SpatialSchemaPointRef has these characteristics:

- The temporary Geometry column is read-only.
- Any edits to a value in the reference table changes the Geometry value in the data table.
- SpatialSchemaPointRef can be applied to any data source except Seamless, View, and ResultSet.
- You can define styles for each point in the table.
- You can store table information as a TAB file and open like any other table.

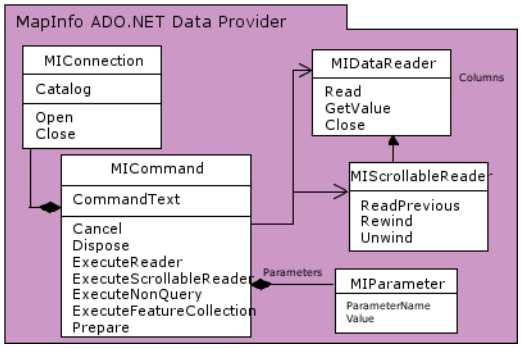
For more information and code examples, see the MapInfo.Data.SpatialSchemaPointRef class in the Developer Reference Help system.

VB example:

```
Public Shared Sub MapInfo_Data_SpatialSchemaPointRef(ByVal _
   map As _Map)

 ' a non-mappable table
   Dim ti As TableInfo = _
      TableInfo.CreateFromFile("c:\data\customers.TAB")
   Dim pr As SpatialSchemaPointRef = New SpatialSchemaPointRef
       pr.CoordSys = map.GetDisplayCoordSys()
       pr.StyleType = StyleType.None
       pr.RefTable = "us_zips"

' the column in RefTable which will match the MatchColumn in my data
      pr.RefColumn = "zipcode"

 ' a column in the Customer table
       pr.MatchColumn = "zip"
      pr.RefTableLocation = "c:\data\us_zips.tab"

' Now set the spatial schema information before opening the table.
   ti.SpatialSchema = pr
   Dim table As Table = Session.Current.Catalog.OpenTable(ti)
End Sub
```

MapInfo ADO.NET Data Provider

MapXtreme provides mechanisms for issuing SQL commands which return record sets from tables using ADO.NET. The MapInfo ADO.NET Data Provider is one mechanism for accessing data in .NET applications in this fashion. For an alternative that uses the Feature class and SearchInfo methods on the Catalog, see Features and Feature Collections.



The following sections present the key interfaces and classes for accessing data via the MapInfo ADO.NET Data Provider.

- MIConnection
- MICommand
- MIDataReader
- MapInfo SQL

MIConnection

An MIConnection represents a connection to the Catalog. The connection provides a starting point for issuing SQL commands and obtaining results. Whereas most data provider connections allow the user to immediately begin issuing queries or other commands against existing tables (or schema objects), the MapInfo ADO.NET Data Provider initially has no tables available. Tables need to be opened or created before they can be accessed. When opened, a name (alias) can be associated with the table which is used when resolving identifiers in the query engine.

Connections are not pooled in the MapInfo Data Provider and there is no connection string required to create a new connection.

The MapInfo.Engine.Session class creates and initializes the Catalog which may be accessed through the Session.Current.Catalog property. The MIConnection.Open method obtains a reference to the Catalog using the Session.Current.Catalog property and the MIConnection.Close method sets the internal reference to the Catalog to null.

VB example:

```
Public Shared Sub MapInfo_Data_MIConnection()
    Dim connection As MIConnection = New MIConnection
    Dim command As MICommand = connection.CreateCommand()
    command.CommandText = "Select * From States Where Pop > 1000000"

    connection.Open()
    Dim reader As MIDataReader = command.ExecuteReader()
    Dim i As Integer, n As Integer = reader.FieldCount
    For i = 0 To n - 1 Step i + 1
        Console.Out.Write("{0}\t", reader.GetName(i))
    Next
    Console.Out.WriteLine()
    While reader.Read()
        For i = 0 To n - 1 Step i + 1
            Dim o As Object = reader.GetValue(i)
            If o Is DBNull.Value Then
                Console.Write("null\t")
            Else
                Console.Write("{0}\t", o.ToString())
            End If
        Next
        Console.Out.WriteLine()
    End While
    reader.Close()
    command.Dispose()
    connection.Close()
End Sub
```

MICommand

MICommand provides the necessary interface for executing SQL commands against the MapInfo Data Provider. MICommand creates MIDataReader and MIScrollableReader instances for obtaining data via the ExecuteReader and ExecuteScrollableReader methods, respectively.

Supported Commands

The commands that are understood by the MICommand are:

Select

```
SELECT < select_list >
   FROM { < table_source > } [ ,...n ]
```

```
   [ WHERE < search_condition > ]
   [ GROUP BY expression [ ,...n ] ]
   [ ORDER BY {expression | column_position [ ASC | DESC ] } [ ,...n ]]

   < select_list > ::=
      {
         *
         | { table_name | table_alias }.*
         | { expression } [ [ AS ] column_alias ]
      } [ ,...n ]

   < table_source > ::=
      table_name [ [ AS ] table_alias ]
```

Insert

```
INSERT [INTO] { table_name } [ ( column_list ) ]
   { VALUES ({expression | NULL}[, ...n]) | query_specification
```

Update

```
UPDATE { table_name }
   SET {{ column_name } = { expression | NULL }} [, ...n]
   [WHERE < search_condition > ]
```

Delete

```
DELETE [FROM] { table_name } [ WHERE < search_condition > ]

< search_condition > ::=
   { [ NOT ] < predicate > | ( < search_condition > ) }
      [ { AND | OR } [ NOT ] { < predicate > |
      ( < search_condition > ) }  [ ,...n ] ]

< predicate > ::=
   {
      expression [ { = | < > | != | > | >= | < | <= } expression ]
      | string_expression [ NOT ] LIKE string_expression [ ESCAPE
              'escape_character' ]
      | expression [ NOT ] BETWEEN expression AND expression
      | expression IS [ NOT ] NULL
   }
```

expression

Is a column name, pseudo column, column alias, constant, function, or any combination of column names, column aliases, constants, and functions connected by an operator(s). Column names and pseudo columns may be prefixed with a table name or a table alias followed by the dot (“.”) character.

group_by_expression

Is a reference to a column in the select list - either an exact copy of the select list expression, the alias, a 1-based number indicating the position of the column, or col*n* where *n* is a number representing a column.

order_by_expression
Is a reference to a column in the select list - either an exact copy of the select list expression, the alias, a 1-based number indicating the position of the column, or col*n* where *n* is a number representing a column.

For more information on expressions, where they are used and how to create them, see Features and Feature Collections.

ExecuteFeatureCollection

The ExecuteFeatureCollection method in the MICommand class is the bridge between the MapInfo ADO.NET Data Provider and the Feature object model. This method executes command text (SQL statements) against the data source connection, and builds an IResultSetFeatureCollection. The Feature model is discussed in Features and Feature Collections.

MIDataReader

The MIDataReader provides forward-only, read-only access to the data returned from executing a SQL Select statement. To create a MIDataReader, you must call the ExecuteReader method of the MICommand object, rather than directly using a constructor.

The MapInfo Data Provider allows multiple MIDataReader instances to be in use on the same connection. However, if the Table being accessed resides on a Microsoft SQL Server database, only one MIDataReader may be open at a time.

IsClosed and RecordsAffected are the only properties that you can call after the MIDataReader is closed.Although the RecordsAffected property may be accessed while the MIDataReader exists, always call Close before returning the value of RecordsAffected to ensure an accurate return value.

You must explicitly call the Close method when you are through using the MIDataReader.

When accessing the DataReader through the IEnumerator or IFeatureEnumerator interface, Close() is automatically called when MoveNext() returns false. Only one enumerator can be used on a DataReader.

ⓘ For optimal performance, MIDataReader tries to avoid creating unnecessary objects or making unnecessary copies of data. As a result, multiple calls to methods such as GetValue may return a reference to the same object. Use caution if you are modifying the underlying value of the objects returned by methods such as GetValue.

The MIDataReader provides a means of reading a forward-only stream of rows from the MapInfo data provider. This cursor type is the best performing for accessing a selection of rows since there is little setup or overhead.

Scrollable Data Readers

MIScrollableReader derives from MIDataReader and offers forward and reverse reading. Some of the options available with MIScrollableReader include:

- ReadPrevious
- Rewind
- Unwind
- ReadTop
- ReadBottom
- AtTop / AtBottom

ⓘ An MIScrollableReader is more expensive to create than MIDataReader. This is the most expensive cursor since there is setup and extra resources necessary to keep track of record order to allow scrolling. Use this cursor only if you need to scroll through the record set.

MapInfo SQL

The MapInfo SQL Language allows you to add powerful analytical processing to your MapXtreme application. MapXtreme exposes SQL processing to users via the MapInfo ADO.NET Data Provider for accessing data (specifically the MICommand object). Expressions are also used for labeling, thematics, legends, AddColumns, Feature searching, and Selection processing.

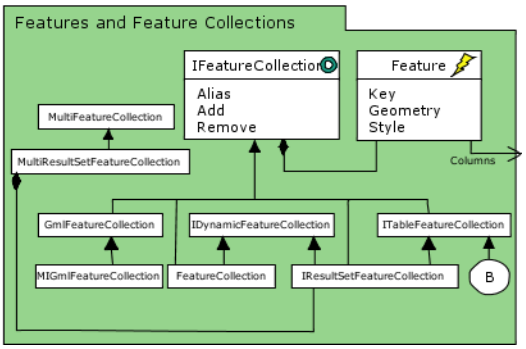
MapInfo SQL is standardized based on SQL-3 Specification. For example, you will find that:

- String constants are enclosed in single quotation marks
- Identifiers may be enclosed in double quotation marks
- Select has no relationship to the Selection

A complete reference including code examples for the MapInfo SQL language is provided in the MapInfo SQL Reference, which you can view directly from Visual Studio's Help system.

Features and Feature Collections

The Feature class object model in MapXtreme offers a non-SQL-based approach to access and manipulate data. This section covers the Feature class and IFeatureCollection interface. A key task in working with features is the ability to search for them using a query definition object.



Feature

Features are described by their geometry, style, data source, key and attributes. Typically a feature is a row in a table. A feature's geometry is a FeatureGeometry object. FeatureGeometries can cover a given area (MultiPolygon), a location (Points, MultiPoints); and distance (MultiCurves, LegacyArcs). Additional Geometry classes that derive from FeatureGeometry and are used for map features are FeatureGeometryCollection and LegacyText. (Rectangle, rounded rectangle and ellipse objects also derive from FeatureGeometry, but are used primarily for cosmetic display purposes.)

One of the main uses of computerized maps is to gather information about the features. In MapXtreme features are returned in FeatureCollections by any of several methods, either created from scratch using a schema, selected using selection tools or methods or by searching the Catalog for those that meet a specific set of criteria.

You can force a Load using the Load method. Changes made to the Feature are not reflected in the underlying table (if there is one) until the Feature is saved back to the table. This is done using the Update method, or UpdateFeature or InsertFeature. You can throw away any edits done to the Feature object before it is saved using the DiscardEdits method.

A Feature has a schema that describes the attributes of the Feature. The Columns property describes the schema.

Retrieving Features from a Table

A Table is a type of Feature collection. As such, the Features within the table may be enumerated directly. For example:

VB example:

```
Dim ftr As Feature
For Each ftr In table
...
```

The default feature enumerator for a table uses an MIDataReader internally with the following command:

```
command.CommandText = "Select MI_Key, * From \"" + table.Alias + "\"";
```

To retrieve a subset of the features in a table, use one of the Catalog.Search methods or use one of the MICommand.ExecuteFeatureCollection methods.

Modifying Features in a Table

To modify features in a table, use one of the following methods.

- Feature.Update
- Table.UpdateFeature
- Table.InsertFeature

Feature Collections

Feature collections are a group of Feature objects. All Features in a collection share the same Schema (columns). The Feature collection has a schema which is the schema of all of its member feature instances. Some Feature collections own their Features while other Feature collections maintain references to Features.

Searching for Features

One of the most common tasks in Precisely's mapping applications is to search for features that meet certain criteria. Once you have the features you are interested in, you can carry out further analysis, such as thematic mapping. In MapXtreme, searching for features can be done in a number of ways: using tools, using Catalog search methods, or using SQL and the MapInfo ADO.NET Data Provider.

The following code sample shows two ways to search for the same thing, in this case, cities in New York.

```
// Using SQL
command.CommandText = "Select Obj From States Where state = 'NY';
```

```
FeatureGeometry nyGeom = command.ExecuteScalar() as FeatureGeometry;
command.CommandText =
             "SELECT * FROM Cities WHERE Obj within @newyork";
command.Parameters.Add("@newyork", nyGeom);
MIDataReader reader = command.ExecuteReader();
// or… to get a FeatureCollection
IFeatureCollection fc = command.ExecuteFeatureCollection();

// Using Features
Feature fNY = catalog.SearchForFeature("States", _
     SearchInfoFactory.SearchWhere("state='NY'"));
SearchInfo si = SearchInfoFactory.SearchWithinFeature(fNY, _
     ContainsFilter.ContainsType.Centroid);
IDynamicFeatureCollection dfc = _
        catalog.Search("Cities", si) as IDynamicFeatureCollection;
Console.Out.WriteLine( _
        "There are {0} cities whose centroid is within NewYork." _
        dfc.Count);
```

SQL searches are more fully discussed in MapInfo ADO.NET Data Provider. The following sections focus on searches using the Catalog and SearchInfo.

Catalog Search Methods

The Catalog has a number of search methods as members. The overloaded Search method can be used to search on one or more tables. They include different arguments to make each search unique. For example, the basic Search (Table, SearchInfo) searches the given table and returns a FeatureCollection. The Search (ITableEnumerator, SearchInfo) method searches on multiple tables and returns a MultiResultSetFeatureCollection.

The SearchForFeature method returns the first Feature from the results. The SearchReader method returns an MIDataReader cursor with the results.

Code Sample: SearchForFeature

The following example shows how to use Catalog.SearchForFeature and Catalog.SearchWithinGeometry. It finds all the cities in the uscty_1k table that are within Florida. It assumes that tables “usa” and “uscty_1k” are open and that there is one map.

VB example:

```
Public Shared  Sub MapInfo_Data_SearchInfo(ByVal catalog As Catalog)
   Dim fFlorida As Feature = _
      catalog.SearchForFeature("usa",MapInfo.Data._
      SearchInfoFactory.SearchWhere_("State='FL'"))
   Dim si As SearchInfo =
      MapInfo.Data.SearchInfoFactory.SearchWithinGeomeTry(fFlorida._
      Geometry,ContainsType.Centroid)
```

```
   Dim fc As IResultSetFeatureCollection =  _
      MapInfo.Engine.Session.Current.Catalog.Search("uscty_1k",si)

' Set the map view to show search results
   MapInfo.Engine.Session.Current.MapFactory(0).SetView(fc.Envelope)
   ' Set the view of the first map.

   ' Add results to selection.
MapInfo.Engine.Session.Current.Selections.DefaultSelection.Add(fc)
End Sub
```

SearchInfo and SearchInfoFactory

The MapInfo.Data.SearchInfo class defines the query used in a search and handles any necessary post processing of the search results.

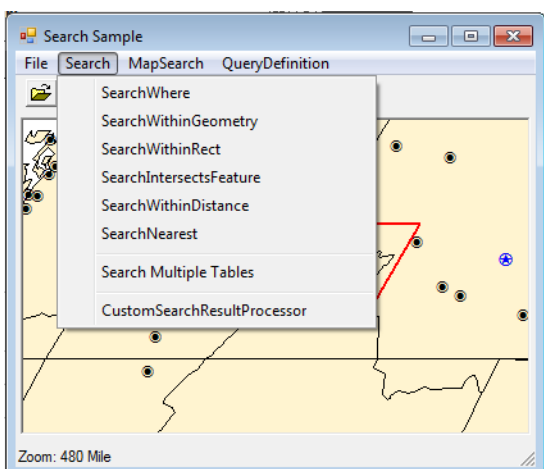
The SearchInfoFactory creates SearchInfo objects. SearchInfoFactory contains a number of search methods that allow you to search using spatial references to your search location or by using geometries that are drawn on the screen.

The following table describes the SearchInfoFactory search methods.

SearchInfoFactory Methods	Behavior
SearchAll	Returns all the rows.
SearchNearest	Returns the rows with table geometries that are closest to the given search point.
SearchWhere	Returns the rows specified by the given where Clause.
SearchWithinDistance	Returns the rows where the table geometry is contained within a distance of the search point, rectangle or geometry. This method uses the Geometry.Distance method to determine if an object is in or out the search area. Previously SearchWithinDistance had buffered the distance and searched within the buffer, leading to less accurate results.
SearchWithinFeature	Returns the rows where the table geometry is contained within the search features's geometry.

SearchInfoFactory Methods	Behavior
SearchWithinGeometry	Returns the rows where the table geometry is contained within the search geometry.
SearchWithinRect	Returns the rows where the table geometry intersects the given rectangle.
SearchIntersectsFeature	Returns the rows where the table geometry intersects with the search features's geometry.
SearchIntersectsGeometry	Returns the rows where the table geometry intersects with the search geometry.
SearchWithinScreenRadius	Creates a SearchInfo that returns the rows where the table geometry intersects a screen circle.
SearchWithinScreenRect	Returns the rows where the table geometry intersects the given screen rectangle.

MapXtreme ships with a Search sample application that you can run and learn more about each search type. The illustration below shows the Search menu with the SearchInfoFactory methods that use a spatial reference. The Map Search menu has methods for searching based on a drawn screen geometry object (circle or rectangle). The Query Definition menu highlights the use of various filters that act on an SQL statement. Find the sample in the ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\Search folder.



Code Samples

This section includes several code samples that pertain to SearchInfoFactory methods.

SearchNearest

Using the SearchNearest method, the code simulates the Select Point tool behavior to select the topmost items under a mouse click and add them to the selection.

VB example:

```
Public Shared Sub MapInfo_Mapping_SearchInfoFactory(ByVal _
   mapControl1 As MapControl)
' Get a point from mouse click. Hard coded value use in sample.
   Dim pt As System.Drawing.Point = New System.Drawing.Point(100, 100)
' Assumes there is a MapControl with a map in it.
   Dim map As Map = mapControl1.Map
   Dim session As ISession = MapInfo.Engine.Session.Current
   Dim si As SearchInfo = _
      MapInfo.Mapping.SearchInfoFactory.SearchNearest(map, _
      pt, 3) ' 3 pixel tolerance radius
   si.QueryDefinition.Columns = Nothing
' fetch all columns instead of just default
' Customize to stop at topmost layer where something is found
   CType(si.SearchResultProcessor, ClosestSearchResultProcessor)._
      Options = ClosestSearchOptions.StopAtFirstMatch
' Puts results of search directly into default selection
' Searches all tables in map in order from top to bottom.
   MapInfo.Engine.Session.Current.Catalog.Search(map._
      Layers.GetTableEnumerator(), si, _
      session.Selections.DefaultSelection, _
      ResultSetCombineMode.Replace)
End Sub
```

SearchIntersectsFeature

VB example:

```
Public Shared  Sub _
     MapInfo_Data_SearchInfoFactorySearchIntersectsGeomeTry(ByVal _
     map As Map)
   Dim ti As Table =  _
      MapInfo.Engine.Session.Current.Catalog.GetTable("usa")
   Dim lParks As MapInfo.Mapping.FeatureLayer = _
      CType(map.Layers("USA"), MapInfo.Mapping.FeatureLayer)
   Dim g As MapInfo.Geometry.FeatureGeometry =  New _
      MapInfo.Geometry.Point(map.GetDisplayCoordSys(),-98,34)
   Dim si As SearchInfo =  _
      MapInfo.Data.SearchInfoFactory.SearchIntersects_
      Geometry(g,MapInfo.Data.IntersectType.Geometry)
   Dim fc As IResultSetFeatureCollection =  _
      Session.Current.Catalog.Search("usa",si)
```

```
   map.SetView (fc.Envelope)
End Sub
```

SearchWithinScreenRadius

This sample illustrates how to search for features using a screen drawn circle.

C# example:

```
// find nearest city to center of map
private void menuItemSearchNearest_Click(object sender, System.EventArgs e)
{
   try
   {
      Cursor.Current = Cursors.WaitCursor;
      // to compare to SearchWithinScreenRadius, we are
      // calculating the search distance the same way it does
      System.Drawing.Rectangle rect=mapControl1.Bounds;
      System.Drawing.Point pt = new System.Drawing.Point(rect.Left, rect.Top);
      pt.X += rect.Width/2;
      pt.Y += rect.Height/2;

      DPoint dpt1 = new DPoint();
      // convert center point to map coords (could use map.Center)
      _map.DisplayTransform.FromDisplay(pt, out dpt1);
      Distance d = MapInfo.Mapping.SearchInfoFactory.ScreenToMapDistance (_map,
3);

      SearchInfo si =MapInfo.Data.SearchInfoFactory.SearchNearest(dpt1,
_map.GetDisplayCoordSys(), d);
      IResultSetFeatureCollection fc = _catalog.Search("uscty_1k", si);

      MapInfo.Geometry.Point p = new
MapInfo.Geometry.Point(_map.GetDisplayCoordSys(), dpt1);
      FeatureGeometry buffer = p.Buffer(d.Value, d.Unit, 20,
DistanceType.Spherical);
      ShowSearchGeometry(buffer);

      SelectFeatureCollection(fc);
   }
   finally
   {
      Cursor.Current = Cursors.Default;
   }
}
```

Code Example: How to Find a Feature Containing a Point

You may also wish to perform the opposite type of search, in which you are looking for a feature that encompasses a point. For example, for delivery planning, you may want to know in which region a customer is located.

VB example:

```
Dim g As MapInfo.Geometry.FeatureGeometry = New _
   MapInfo.Data.SearchInfoFactory.SearchIntersectsGeometry(g, _
   MapInfo.Data.IntersectType.Geometry)
Dim irfc As MapInfo.Data.IResultSetFeatureCollection = _
   MapInfo.Engine.Session.Current.Catalog.Search("british_
   columbia", si)
Me.MapControl1.Map.SetView(irfc.Envelope)
```

Saving Opened Table as GeoJson File

MapXtreme provides an API named ExportToGeoJsonFile(), to save any opened table content as GeoJson file.

Following code provide a summary of this API:

```
// Open Table
Table table = Session.Current.Catalog.OpenTable(@"c:\temp\usa.tab");
// Save table content as GeoJson file.
TableExportFactory.ExportToGeoJsonFile(table, @"c:\temp\usa.geojson");
// Close Opened table.
table.Close();
```

Analyzing Data

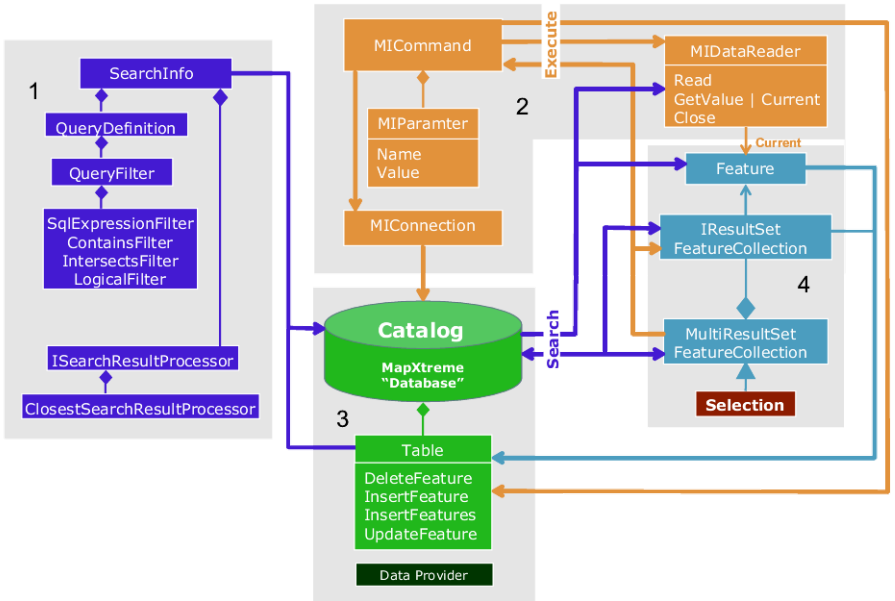
Once your data is available in the Catalog, you will want to analyze it to meet your business objectives. The Catalog has an SQL processor that allows you parse and aggregate your data. Here you have two options:

- OGC object-based query interface
- ADO.NET SQL-based interface

The diagram below shows the relationship between the two.

Group 1 shows the OGC query interface. Use these objects to construct a query. The interface allows you to create queries to filter columns and rows, as well as add spatial and non-spatial conditions. The queries interact through the Search methods off those query objects to return data readers and result sets. Use these objects if you are more comfortable with object-oriented programming and less so with SQL syntax. See SearchInfo and SearchInfoFactory.

The ADO.NET interfaces, shown in group 2, use the defined ADO.NET model to allow access via the MapInfo SQL language. The ADO.NET interfaces use SQL syntax to interact with the Catalog. In this instance you need to generate the SQL statement and assign it to the MICommand object. These objects use the Execute command to return a data reader or result set. See MapInfo ADO.NET Data Provider.



MapXtreme Data Model

Both the OGC query-based and ADO.NET command-based approaches use the Catalog (group 3) to organize the data sources as a response to the object or SQL query. The object-based query API will generate SQL and pass this to the Catalog for processing. In some instances you may be able to generate more efficient SQL by hand, but the objects are well defined and the interfaces restrict how you interact so the SQL tends to be optimal. If you are comfortable with the SQL language using the ADO.NET method may be more comfortable. But if you are inexperienced with SQL then the OGC object based query will work just as well.

The MapInfo SQL syntax is defined in the SQL Reference which ships with MapXtreme. The language is based on SQL3 and has special MapInfo operators defined for spatial analysis. These operators begin with the MI_ prefix.

Data Readers, MemTables and Result Sets

The methods to access data return a data reader or result set. A data reader allows access in a sequential manner and does not store copies of data. It retrieves the data from the data source, except in the case where the data source is cached. Result sets are collections of keys. These keys allow you access back to the original tables and do not create copies of the data.

A MemTable also allows you to store data from various sources into one table. This table type stores data in a combination of memory arrays and temporary disk storage. When data is added, the MemTable makes a copy of the data and does not have a key or pointer back to the original table. These are useful for temporary layers for maps and containers for return values of processes such as a geocoding or routing result. MemTable access and map rendering performance is equivalent to native tables.

Result sets are a great tool when you need access to a defined set of rows and when you need to get data from the source. If the source data may change during your session then this method allows you to see the results if the data source supports concurrent access. Since MemTables are copies of data they are a static set of data rows and will not reflect changes from the original data sources.

Improving Data Access Performance

Performance is always an important aspect to any application that accesses data. Consider the following list in your design and development plans for your application.

- Only request the data you need (especially from an RDBMS). This limits the amount of data sent over the connection.
- For web applications, put attribute column information into the workspace file, so that all needed data will be pre-loaded at the MapInfo Session creation time. For an example, see Overview of the Thematics Sample.
- Only sort tables if you need an ordered list. This process takes time to read through the entire table to build an order. Also it will be slower if there is no index on the column.
- Only scroll if you need random access to a table. This also builds indexes to speed up access and remember order. Data readers access the data directly with no need to read extra data.
- Use consistent coordinate systems for Join and Search operations. This eliminates the need to convert geometries for every access.
- Use indexed columns for Join / Filter / Sort / Aggregate operations.
- Use CentroidWithin, ContainCentroid, and EnvelopesIntersect prior to actually checking for geometry intersects. These tests are very quick and in most cases eliminate a lot of geometries from your list with little effort.
- Use BeginAccess/EndAccess (especially for file-based tables) when performing multiple queries and/or edits.
- Try to avoid calls such as Area and Buffer in the Where clause because the operation will have to be done each time a new cursor is created.
- Try to avoid calls such as Area and Buffer in the Select list when defining a view or result set for similar reasons.
- Use result sets for intermediate results or operations where you manage keys. These are very light weight and afford quick direct access back to the original data.

9 – Working with Core MapXtreme Classes

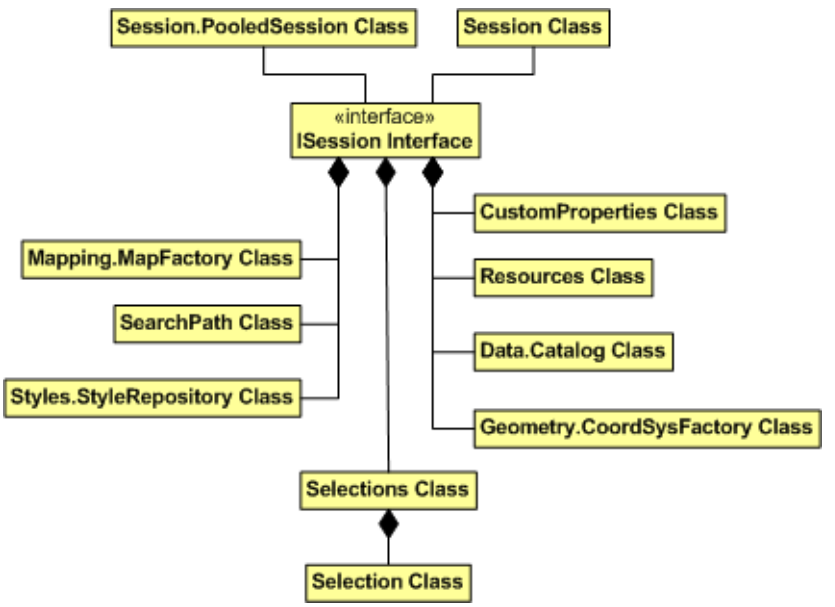
The MapInfo.Engine namespace contains the interfaces and classes that relate directly to the core functionality that drives all applications based on MapXtreme. This includes the core ISession interface which is the starting point for all MapXtreme applications. Classes in this namespace include Session and Selections, and SearchPath. Other types in the namespace are supporting classes, delegates, structures, and enumerations for Collections, Resources, and CustomProperties.

In this chapter:

Session Interface

The ISession interface is the starting point for all MapXtreme-based applications. It manages the initialization of resources needed for a MapXtreme application.

An instance of ISession holds components of the MapXtreme object model such as the DataAccess engine, MapFactory, CoordSysFactory so that the desktop or web application can do work. The following diagram illustrates the classes that implement ISession interface.



For an ASP.NET application each client request has its own ISession instance. This instance resides in the calling context and is available throughout the lifetime of the client's request.

For a single-threaded desktop application there is only one instance. On a multi-threaded desktop application there is one instance per thread.

The MapInfo.Engine.Session class provides access to the ISession object. To get the current ISession instance, access the MapInfo.Engine.Session.Current property.

Session Management

Session management is a key topic to understand when you are designing your application. While a desktop application's session management is straightforward (each user has his own ISession instance), a web application needs to factor in, perhaps, an

unknown number of users who will user your application. It will have to know how to handle each user's state so that the correct information and visual display is returned to the correct user.

MapXtreme provides templates for building web applications that help you manage state properly. This topic is discussed in greater detail in Chapter 6 Understanding State Management. Key decisions about state management, pooling, performance and data access are presented to help you make informed decisions during the design phase of your project, before you start coding.

Using Session.Dispose Method

The MapInfo.Engine.Session class has two overloaded Dispose methods. Your choice will depend on the type of application you are building.

Session.Dispose()

Session.Dispose() disposes the ISession instance that is accessible through the Session.Current property. This method is used only for multi-threaded desktop applications.

Do not use this for web applications or single-threaded desktop applications. For web applications, ISession is managed by WebSessionActivator.

For single-threaded desktop application, Dispose is called automatically when the application is shutdown or when the AppDomain using MapXtreme is unloaded.

Session.Dispose(HttpSessionState)

Use Session.Dispose(HttpSessionState) for web applications that use the default session state settings in which the ISession is stored in memory. Do not call this method for any other configuration since the ISession instance is not stored in memory in any other configuration.

The state settings are represented in the Web.config file of your application project by the following keys:

```
<add key="MapInfo.Engine.Session.State" value="HttpSessionState" />
<sessionState mode="InProc" />
```

The first setting is an application-specific setting that controls the mechanism for saving and restoring the state of the MapInfo.Engine.ISession instance. This instance is accessible through the MapInfo.Engine.Session.Current property. The HttpSessionState

setting indicates that the session is saved and restored through the ASP.NET session state. This state is exposed through the current HttpContext and is of type HttpSessionState.

The second setting is an ASP.NET setting that controls how the HttpSessionState is saved and restored. InProc indicates that the state of the ASP.NET session is to be placed in memory and is unique to each ASP.NET ISession instance. This is the default setting.

When you use these settings, there is an ISession instance for each ASP.NET session and it is stored in the HttpSessionState throughout the lifetime of the ASP.NET session. In order for the ISession instance to be properly disposed of when the session times-out or ends, you must add the following statement to the Session_End method in your Global.asax source code file.

VB example:

```
MapInfo.Engine.Session.Dispose(Me.Session)

Protected Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
    MapInfo.Engine.Session.Dispose(Me.Session)
End Sub
```

Making this call will ensure that the ISession instance is properly disposed of and that memory is reclaimed.

ISessionEventHandlers

MapXtreme provides a MapInfo.Engine.ISessionEventHandler interface for loading custom DLLs that extend MapXtreme's functionality. Use this interface to autoload any extension DLLs that you need for your application, such as extensible data providers, persistence providers, and styles that a default workspace may use. When MapXtreme starts up, it will initialize these assemblies and carry out the required instructions.

For example, MapXtreme provides a SpatiaLite sample implementation that, when compiled, yields an assembly that contains code to support the SpatiaLite data provider as well as to load the data provider when a MapXtreme session initializes.

In order for SessionEventHandler assemblies to be initialized on startup, they must be located in the \Common Files\MapInfo\MapXtreme\9.x.x\SessionEventHandlers folder. In fact, any assembly in this folder with the file suffix of .SessionEventHandler.DLL will be loaded.

The ISessionEventHandler interface provides two methods that can be called to load any extension code you need for your application and provide additional initialization functionality.

```
void BeforeWorkspaceLoad(ISession session)
```

is called before any default workspaces are loaded, and could be used to set up any extensible data providers, edp callbacks, edp persistance providers, named connections, load or create styles, open tables, database connections etc. that might be used by a default workspace.

```
void AfterWorkspaceLoad(ISession session)
```

is called after any default workspaces (specified in app.config or web.config) are loaded. This is a good place to do any final session initialization that your application could need.

When MapXtreme finds classes in the assembly that implement the ISessionEventHandler interface, it constructs an instance of the class and adds it to an internal collection. Whenever a new MapXtreme Session instance is created (once for a desktop application, can be multiple sessions for web applications), all of the Session handlers are called with the instance of the session being initialized.

Since there can be multiple SessionEvent handlers loaded and the order in which they are called is indeterminate, if you are writing a handler, do not make any assumptions about what is already in the session (ie: maps, tables, etc) .

Use this capability with caution! Any MapXtreme session created on that computer will load these session event handlers. Adding handlers that display dialog windows or other user interface components may cause web applications to hang as these dialogs will be displayed on the server.

Serialization and Persistence

A Map is bound to a ISession object. You cannot take a Map object from one ISession object and use it in another. You have two options if you want to clone the entire ISession object; either by using Serialization or by using Persistence.

Serialization

Serialization is the process of converting an object into a stream of data in order to preserve it on the server. This process is an essential part of maintaining objects in MapXtreme web applications. If the objects are not maintained the server would need to recreate the object (such as a map) for each web request. When an object is requested, it is deserialized (or recreated from the stream of data) and then modified. This algorithm does not make a copy of the object (as other serialization algorithms do) such that the object being deserialized is created only once.

The serialization is performed by formatters that are embedded in the Microsoft.NET Framework. Two different formatters are included in the Framework, one for binary objects, BinaryFormatter, and one for SOAP objects, SOAPFormatter (SOAP is a lightweight protocol intended for exchanging structured information in a distributed environment, such as the web.). SOAPFormatter is relatively faster than BinaryFormatter. SOAPFormatter is used for certain basic types of data (Int, Byte, Decimal, etc.) and BinaryFormatter is called for complex of objects. See the Microsoft MSDN documentation for more information about the SOAPFormatter and the BinaryFormatter.

To pass an object to one of these formatters, use the GetObjectData() method. To deserialize the object (restore it from the stream) use the SetObjectData() method.

Any object that supports the ISerializable interface will automatically be restored, or deserialized. The ASP.NET framework automatically deserializes the context.Session[] array after HttpApplication.BeginRequest. MapInfo.Engine.Session is set up in HttpApplication.BeginRequest handler so objects are deserialized into the MapInfo.Engine.Session.

Serialize/Deserialize the Session Object

The following is an example of how to serialize/deserialize the Session object.

```
// Create a MemoryStream to serialize into
MemoryStream stream = new MemoryStream();
// Serialize the MapXtreme Session object
BinaryFormatter formatter = new BinaryFormatter();
formatter.Serialize(stream, Session.Current);
stream.Position = 0;
// Make changes to the Session object to make sure the
// deserialization works correctly
...
// Recreate the MapXtreme Session object from the stream
// Note: this will replace the current MapXtreme Session object with the
// contents of the stream
formatter = new BinaryFormatter();
formatter.Deserialize(stream);
```

The stream parameter passed to the formatter.Serialize method can be anything derived from System.IO.Stream.

This serialization functionality was designed to be used with MapXtreme's state management feature. If you persist this information to disk, then try to reload it in the future with a different version of the product, it's not guaranteed to work.

For more on serialization and state management see Implementing a StateManager.

Persistence

Persistence in MapXtreme is the is the process of storing application objects to MapInfo Workspace (.MWS) files, which exist in XML format. Persisting and de-persisting ISession objects is done using the WorkSpacePersistence and WorkSpaceLoader classes. These two classes will write out and read in .MWS files. The application can write out the workspace (a copy of the Session), and apply the contents of the workspace file to a new Session object, creating a clone of the Session.

VB example:

```
Public Shared Sub MapInfo_Persistence_WorkSpacePersistenceSave()
' Create a named connection point to "D:\data\version2"
   Dim info As NamedConnectionInfo = New NamedConnectionInfo("file",_
      ConnectionMethod.FilePath, "D:\data\version2")
   Session.Current.Catalog.NamedConnections.Add("MyDataFolder",_
      info)

 ' Create a map
   Dim map As Map =
   Session.Current.MapFactory.CreateEmptyMap("MyMap",_
      "MyMapAlias", New Size(400, 400))
   Dim table As Table =
   Session.Current.Catalog.OpenTable("MyDataFolder", _
      "myTableAlias", "Seamless\Lines - NYALBA\SeamCapDist.TAB")
   map.Layers.Add(New FeatureLayer(table))

'   Save the Session to a workspace file
   Dim w As WorkSpacePersistence = New WorkSpacePersistence
   w.Save("c:\workspace\mySeamless.mws")
 End Sub
```

Note that adding a workspace (using the WorkSpaceLoader class) is an cumulative process. To ensure the Mapxtreme Session only contains the contents of the new workspace file, call Reload() method on the ISession object first. This method clears the Session state and reloads pre-defined workspaces in the application's Web.config file.

For more information on persistence and the XML schema, see Appendix C: Understanding the MapInfo Workspace.

Opening and Saving a Workspace Containing Named Resources

MapXtreme supports named resources for map definitions, map layers, data source definitions and styles. Previously MapXtreme supported named connections. Named resources are references to a resource in another location and identified by a name.

MapInfo Workspace format files (.MWS) containing named resources can be opened and saved programmatically through the **INamedResourceResolver** interface in the MapInfo.Engine namespace. This interface contains the functions to resolve the named resources while parsing the MWS file and also to generate the named resources xml node when saving the MWS file.

Opening an MWS: ResolveResource()

On opening and parsing an MWS containing named resources, the ResolveResource() function returns the xmlNode corresponding to the name and type of the named resource.

For example, when parsing a named style for a WMS, you would pass NamedResourceType.Style and XmlNode in the form of a string. ResourceResolver parses the parameters and returns a named resource node in the form of an XmlNode.

Saving an MWS: GetResourceName()

When the MWS file is being saved, file parsed in memory will contain all the expanded nodes. To write back the named Resources into the MWS file, you need to implement all flavors of GetResourceName which takes objects of Table, Map, Style, and IMapLayer. These functions will return the NamedResource node for the corresponding object. The saved MWS file contain the entries of the named resources.

Registering Your Implementation with MapXtreme

To register your implementation of INamedResourceResolver with MapXtreme, you must set the Session.Current.NamedResourceResolver property. Now MapXtreme will refer to your implementation object when opening and saving MWS files.

Setting Preferences

If you only want to save a specific type of name resource and leave the others as is, you must set a preference using PreferenceForNamedResource property available on the WorkspacePersistence object. By default it is set to NameResourcePreference.All. You can set other values as well such as NameResourcePreference.PreferNamedTables to save only Tables as named resources. To store more than one resource as named resources use the OR operator. For example:

```
NameResourcePreference.PreferNamedTables |
NameResourcePreference.PreferNamedLayers.
```

For more information, see INamedResourceResolver in the Developer Reference.

Selection Class

A Selection is a collection of IResultSetFeatureCollection objects that holds lists of features. These features are a subset of rows in a table. They could be property boundaries, street networks, cell tower locations, or natural features such as rivers. They are typically drawn with special highlighting when they display in a Map. There can only be one IResultSetFeatureCollection for a given table in a Selection.

There can be more than one Selection in a ISession. The Selections collection contains all of the selections in the application. There is always at least one selection, known as the DefaultSelection.

Each Selection must have a name and unique alias. By default, map selection tools modify the Selection when used. Each tool can be set to use any particular Selection.

A selection in MapXtreme is not a copy; it is a reference to an IResultSetFeatureCollection for a given table in a Selection. If you attempt to modify a Selection after you have closed the table that you are working with, the reference to the IResultSetFeatureCollection will be invalid, causing an exception.

Features are selected using tools or through search methods. For a discussion of the different selection tools that you can use when building a Windows Forms application, see Chapter 7 Desktop Applications, Controls, Dialogs, and Tools. For selection Web controls and tools, see Chapter 5 Web Applications, Controls, and Tools.

Features can also be selected through search methods from the MapInfo.Data.Catalog class which returns IResultSetFeatureCollection collections. A Selection object can be passed into a search, which can be used to populate or change a Selection.

Features can also be selected via the ExecuteFeatureCollection method from the Data.MICommand class. In this case, you would execute SQL commands against the MapInfo Data Provider.

For more information on features, tables, the Catalog, and the MICommand, see Chapter 8 Working with Data.

Using Selection Properties

The properties of the Selection class are used to set the (required) name and alias, set a selection to be Visible or Editable, or to get the selection's Style. To determine whether a Selection displays its highlighting to indicate it is selected or is available for editing, use the Visible or Editable property, respectively.

If the Editable property is set to true, the table that you are working with must also be editable.

The Style property indicates the Selection's style and returns a reference to the Selection's composite style. When you change a Style property, the Selection object is notified of the change. It will take affect the next time the Selection object is drawn. Styles are discussed in Chapter 15 Stylizing Your Maps.

Selection Highlighting and Exporting

Selections are typically drawn on the map with special highlighting, to distinguish these features from surrounding non-selected features. The highlighting is controlled by the Mapping.FeatureViewer.DrawSelections property. When this property is set to true, they are drawn with selection highlighting, provided they are drawn on a visible layer.

Similarly, the MapExport.ExportSelection property, can be used to control whether the selections are drawn into the exported image.

SelectionChangedEvent

A delegate method is attached to the SelectionChangedEvent in order to receive notification that this selection has changed. For example, if a record is added, the SelectionChangedEvent is fired.

ISerializable Interface on Selection and Selections Classes

The ISerializable interface is implemented on the Selection and Selections classes. The following code example demonstrates how to serialize or deserialize a Selections object:

```
// Create a MemoryStream to serialize into
MemoryStream stream = new MemoryStream();

// Serialize the Selections object
BinaryFormatter formatter = new BinaryFormatter();
formatter.Serialize(stream, Session.Current.Selections);
stream.Position = 0;

// Make changes to the Session's Selections object to make sure the
// deserialization works correctly.
...

// Recreate the Selections object from the stream.
// Note: this replaces the current Session's Selections object with
// the contents of the stream
formatter = new BinaryFormatter();
formatter.Deserialize(stream);
```

Selection Code Examples

The following are code examples of common selection operations. Additional code examples are included in many topics of the MapXtreme *Developer Reference*.

Selecting Features Within Another Feature

A common search technique using MapXtreme is to find features within another feature. You may do this to find all the customers within a postal code boundary or all the highways that are under construction in a sector. Follow the example below. The parameter f is a MapInfo.Data.Feature.

VB example:

```
Dim si As MapInfo.Data.SearchInfo = _
MapInfo.Data.SearchInfoFactory.SearchWithinFeature(f, _
   MapInfo.Data.ContainsType.Centroid)
Dim irfc As MapInfo.Data.IResultSetFeatureCollection = _
   MapInfo.Engine.Session.Current.Catalog.Search("USCty_8k", si)

MapInfo.Engine.Session.Current.Selections.DefaultSelection.Clear()
MapInfo.Engine.Session.Current.Selections.DefaultSelection.Add(irfc)

irfc.Close()
```

Checking a Table for Selections

Follow the code example below to learn how to get a count of selections in a table.

VB example:

```
Public Shared Sub MapInfo_Engine_Selection2()
   Dim session As ISession = MapInfo.Engine.Session.Current
   Dim tableUsa As Table = session.Catalog("usa")

   ' Get fc for selection on usa.
   Dim fc As IResultSetFeatureCollection = _
      session.Selections.DefaultSelection(tableUsa)
   Dim nCount As Integer = 0
   If Not fc Is Nothing Then
      nCount = fc.Count
   End If
End Sub
```

You can also perform selection operations using MapInfo SQL queries and with the ADO.NET data provider. See Chapter 8 Working with Data.

Returning All Columns From a Table

The following sample shows how to return all columns from a selection:

VB example:

```
Dim Connection As MIConnection = New MIConnection
        Connection.Open()
        Dim lyr As FeatureLayer = MapControl1.Map.Layers("usa")
        Dim ti As MapInfo.Data.Table = _
           MapInfo.Engine.Session.Current.Catalog.GetTable("usa")
        Dim si As MapInfo.Data.SearchInfo = _
           MapInfo.Data.SearchInfoFactory.SearchAll()
        si.QueryDefinition.SetColumns("*")
        Dim irfc As MapInfo.Data.IResultSetFeatureCollection = _
           MapInfo.Engine.Session.Current.Catalog.Search(ti.Alias, si)
        Dim l As MapInfo.Data.Feature
        For Each l In irfc
            Dim column As MapInfo.Data.Column
            For Each column In l.Columns
                MessageBox.Show(column.ToString())
            Next
        Next
```

Changing the Map View Following a Selection

The following example shows how to change the zoom to display all the features in a selection.

VB example:

```
Me.MapControl1.Map.Bounds = _
MapInfo.Engine.Session.Current.Selections.DefaultSelection.Envelope.Bounds
```

Event Arguments

The MapInfo.Engine namespace contains various event argument classes that provide data for events. Refer to the online help for more information. Some of the event argument classes include:

- CollectionCancelableEventArgs – Provides data for a collection event that can be cancelled.
- CollectionEventArgs – Provides data for a collection event.
- NodeSelectionChangedEventArgs – Fires these event arguments when the node selection changes.

- SelectionChangedEventArgs – Other objects can attach delegates to this event to get notified when a selection changes.

Exceptions

The Engine namespace contains various exception classes. Refer to the online help for more information. Some of the exception classes include:

- ResourceNotFoundException – Throws this type of exception when the requested object is not found in the Resource table.
- ResourceTypeMismatchException – The exception that is thrown when the object read from Resources was not of the expected type.
- TimeoutException – The exception is thrown on Current timeout while waiting for a pooled ISession to become available.

10

10 – Creating Expressions

Expressions are used throughout MapXtreme to describe the exact pieces of information you need to display and analyze in your mapping application. This chapter covers creating expressions for a wide range of product areas, including data access, creating themes, labeling maps and more.

In this chapter:

Expressions Overview

Expressions are statements that are used to describe and format data. For example, in English, an expression might read like “a median income of more than $50,000, or “female percent of population.”

Expressions are formed using column names, constants (i.e., specific data values), along with functions and operators that act upon the columns and constants. The operators and functions are defined in the MapInfo SQL Language, developed to support MapXtreme and other MapInfo .NET supported products going forward. For details, see the MapInfo SQL Reference via the Help Viewer in Visual Studio.

Use expressions to make the most of your data. By using expressions you can:

- Show only the columns and rows of data that interest you.
- Derive new columns by calculating new values based on the contents of your existing columns.
- Aggregate data to work with subtotals instead of the entire table.
- Combine data from two or more tables into one results table.

Many of the data sets you will use include more objects and information than necessary for your projects. In many cases it is easier to work with a subset of the complete data product. For example, if you were tracking crime statistics for a certain county by census tract, you would not need the census tracts for the entire state. You would use an expression to extract just the census tracts for the county.

Expressions are used throughout MapXtreme, in the following areas:

- SQL statements (select, insert, update, delete, group by, order by)
- SQL functions that take expressions as an arguments (e.g., the geometry argument in MI_Area() is an expression that returns a geometry object.)
- Adding columns (MapInfo.Data.Table.AddColumn creates a temporary column based on an expression.)
- Feature searches (SearchInfo and SearchInfoFactory)
- Themes (FeatureStyleModifier)
- Labels (LabelModifier)
- InfoTips (FeatureLayer, MapTools)
- Expression dialog

Creating Expressions

The simplest possible expression consists of a constant, such as “2” (numeric example) or “Friday” (text example).

Other simple expressions consist of a column name, for example:

```
POP_2000
STATE
```

When you request specific multiple columns in a select statement, for example, these columns together are known as an expression list.

```
Select colA, colB, colC from Table1, Table2
Select colA/2, ColB/ColC from Table1
```

You can also write expressions that perform mathematical operations on your data.

For example, RENT + UTILITIES is an expression that adds two columns together. This expression could be used in a SQL statement to find all apartments that have a total cost of less than $800 per month.

Where Clause – Boolean Expressions

A Boolean expression is a search condition that results in a value of either True or False. For example, the expression

```
2 < 5
```

is a Boolean expression because the result is True.

All expressions that contain relational operators, such as the less than sign (<), are Boolean. The operators AND, OR, and NOT, are Boolean operators. Boolean expressions are also called comparison expressions, conditional expressions, and relational expressions.

```
POP_2000 > 500000
POP_2000 <= POP_1990
PROVINCE <> ‘Ontario’
County = ‘Columbia’ AND VALUE >= 250000
```

Supported operators in MapInfo SQL are defined in the MapInfo SQL Reference online via the integrated Help in Visual Studio (look for MapInfo SQL Reference in the Dynamic Help contents pane).

Boolean expressions are used in the “where clause” of an SQL statement. The where clause is the expression that controls the rows that are returned (the rows that result in True).

For example, the boolean expression in this statement follows WHERE. Only objects in the Europe table that fall within the boundary of France will be returned as True.

```
"SELECT * FROM Europe WHERE MI_Geometry within @France";
```

Functions In Expressions

Functions in MapXtreme are used to create even more complex expressions to retrieve data that meets specific criteria. For example, MapInfo SQL supports many of the usual database functions that work with strings, dates/time, and numbers,

The most powerful functions in MapInfo SQL are those that take advantage of the spatial nature of mapping data. These geographic functions are used to create new geometries, measure area and length, return spatial information, validate spatial relationships among geometries, and others. Supported functions are defined in the MapInfo SQL reference.

An example of using a function in an expression might be when you wish to look at the area of a table of boundaries, such as school districts. Use the function MI_Area() to return the area of each record in the table.

Additional examples of functions in expressions are found in the Expressions Examples section below.

DateTIme and Time Expressions

When using DateTime and TIme expressions with MapXtreme, please be aware of the following:

- If a DateTime column or Time column is used alone in an expression, it is formatted using the current locale.
- If a DateTime or Time column is in an expression, its string value is TimeToNumber or DateToNumber + space + TimetoNumber.
- Operator math on Time or DateTime is not supported. You can add a number to a Date, but not to a Time or DateTime.

Expression Examples

The following highlights some uses of expressions in various areas of MapXtreme.

SQL Statement Examples

This example will select all records from the Eurcity_1K table that are within Germany and have a population of over 1 million.

```
Select * from Eurcity_1K WHERE (MI_Geometry MI_Within @Germany) AND Tot_Pop > 1000000
```

The following examples make selections based on Time and Date columns in the table. This example will select all crime records from a "CrimeActivity" table where the crime occurred between 12:00:00 AM and 6:00:00 AM:

```
SELECT * FROM CrimeActivity WHERE CrimeTime BETWEEN '12:00:00 AM' AND '6:00:00 AM'
```

Where CrimeTime is a Time column that stores the time at which the crime occurred.

This example will select employee Names from an "Employee" table who were born before December 31, 1970.

```
SELECT Names FROM Employee WHERE BirthDay < '12/31/1970 12:00:00 AM'
```

Where BirthDay is a Date column that stores the birthdays of employees.

MapInfo SQL Function Example

The following expression uses a MapInfo SQL function to find features within a buffer.

```
Obj CentroidWithin MI_Buffer(Obj, 5, ‘km’, ‘Spherical’, 24)
```

This expression uses a MapInfo SQL special keyword reserved for geographic objects called ‘Obj’. This keyword describes the geometry of the object such as its coordinate system and bounds. This keyword is compatible with previous versions of MapX and MapInfo Professional. It is equivalent to the column name MI_Geometry.

Note that km and Spherical are enclosed in single quotes. In MapInfo SQL, string literals must be enclosed in single quotation marks while identifiers such as column names, table names, aliases, etc.) should be enclosed in double quotation marks, but only needed if the parsing logic is unable to correctly parse the identifier. This would include identifiers that have spaces in their names or other special characters.

To find features that fall outside the buffer, the expression would look like:

```
NOT Obj CentroidWithin MI_Buffer(Obj, 5, ‘km’, ‘Spherical’, 24)
```

Add Columns Example

When adding temporary (computed) columns to a table using the AddColumns method, the columns supplied contain an expression that defines how the value for the column is computed. The expression may contain an aggregation function if multiple source records are expected to match up to a single record in the table to which the columns are being added.

The example below uses expressions to represent population density "Pop_1990 / MI_Area(Obj, 'sq mi', 'Spherical')". The expressions are preceded by their new column names. PopDensity1990 and PopDensity2000.

VB example:

```
Public Shared Sub MapInfo_Data_TableAddColumns(ByVal miTable _
      As Table)
   Dim NewCols As Columns = New Columns
   NewCols.Add(New Column("PopDensity1990", "Pop_1990 / _
      MI_Area(Obj, 'sq mi', 'Spherical')"))
   NewCols.Add(New Column("PopDensity2000", "Pop_2000 / _
      MI_Area(Obj, 'sq mi', 'Spherical')"))
   miTable.AddColumns(NewCols)
End Sub
```

For more information on adding columns, see Adding Expression Columns to a Table.

Feature Search Example

The following example uses a boolean expression SearchWhere("State='FL'") that, when executed, will return a value of 1 for each row that contains FL.

VB example:

```
Public Shared Sub MapInfo_Data_SearchInfo(ByVal catalog As Catalog)
   Dim fFlorida As Feature = catalog.SearchForFeature("usa", _
      MapInfo.Data.SearchInfoFactory.SearchWhere("State='FL'"))
   Dim si As SearchInfo = _
      MapInfo.Data.SearchInfoFactory.SearchWithinGeometry(fFlorida._
      Geometry, ContainsType.Centroid)
   Dim fc As IResultSetFeatureCollection = _
      MapInfo.Engine.Session.Current.Catalog.Search("uscty_1k", si)

' Set the map view to show search results

   MapInfo.Engine.Session.Current.MapFactory(0).SetView(fc.Envelope)
' Set the view of the first map.

 ' Add results to selection.
MapInfo.Engine.Session.Current.Selections.DefaultSelection.Add(fc)
End Sub
```

For more information on the Feature class and search methods, see Features and Feature Collections.

Thematic Expression Example

A pie theme is an object theme that draws a pie chart based on the numeric value of the theme's expression. The expression for the theme is made up of three columns: Pop_Native", "Pop_Asian","Pop_Other".

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_PieTheme(ByVal _
     map As Map)
' Load a map based on one table
   map.Load(New MapTableLoader("world.tab"))
   Dim lyr As FeatureLayer = CType(map.Layers("world"), FeatureLayer)

' Create a new pie theme
   Dim pieTheme As MapInfo.Mapping.Thematics.PieTheme = New _
      MapInfo.Mapping.Thematics.PieTheme(map, lyr.Table, "Pop_Native", _
      "Pop_Asian", "Pop_Other")

' Create an object theme layer based on that pie theme
   Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer("World _
      Pop Growth Rate", Nothing, pieTheme)

'Add object theme to the map's layer collection.
   map.Layers.Add(thmLayer)
End Sub
```

For more information on themes, see Chapter 14 Using Themes and Legends.

Label Expression Example using Date and Time

Expressions are properties of a label modifier which you use to modify the default properties used to make a label. The expression in the code example says to display the label in this form: Date/Time: <date and time>.

VB example:

```
Public Shared Sub MapInfo_Mapping_OverrideLabelModifier(ByVal _
     modifier As OverrideLabelModifier)
   modifier.Name = "'Date/Time: ' + DateTimeToString( dateTimeColumn, 'm/d/yyyy
hh:mm tt')"
End Sub
```

For more information on labeling, see Labels.

InfoTips Expression Example

InfoTips are text items that display when a tool hovers over a Feature. Expressions can be used to generate text for InfoTips in a similar fashion to the way they are used for labels. Use the MapTool.SetInfoTipExpression static helper function to set the InfoTip. It takes care of creating the InfoTip hashtable and adding the layer entry into the hashtable if it does not already exist. Each tool can be set to have InfoTips enabled or disabled.

The following example will generate a two-line text label to display the Table alias and the X or Y coordinate for the centroid of the object that the cursor is hovering over.

VB example:

```
Public Shared Sub MapInfo_Mapping_HowDoICreateExprForInfoTip(ByVal
mapControl1 As MapControl)
 MapTool.SetInfoTipExpression(mapControl1.Tools.MapToolProperties,_
   CType(mapControl1.Map.Layers(0), FeatureLayer), "@TableAlias + _
   char(13) + _ 'Centroid X:' + MI_CentroidX(obj) + '  Y:' + _
   MI_CentroidY(obj)")
End Sub
```

For more information on InfoTips, see Layer Control.

11

11 – Accessing Data from a DBMS

MapXtreme provides spatial server access. This is a powerful feature that allows developers to connect to live data stored in spatial servers, such as MapInfo's SpatialWare running on Microsoft's SQL Server 2008/2012/ 2014 or the Oracle Spatial databases. Spatial servers allow companies to host their map data in their enterprise database for central management and security. Spatial servers like SpatialWare offer advanced query processing and increased performance on the server for an organization's spatial data.

In this chapter:

Accessing Remote Spatial Data

You can access data using MapXtreme with different DBMS servers. The servers include:

- Microsoft Access 2007 and Excel 2007
- Microsoft Access 2003
- Oracle 11G (11.1.0.6.0 and 11.1.0.7.0)
- Oracle 10G, 10GR2
- Microsoft SQL Server 2012 (with SQL Native Client 11)
- Microsoft SQL Server 2008 (with SQL Native Client 10)
- Microsoft SQL Server 2014

You can add a table from data in a DBMS using the TableInfoServer class in the MapInfo.Data namespace.

The details for adding spatial data are included in the following sections.

Accessing Remote Tables Through a .TAB File

A MapXtreme application can access DBMS data “live”, or can open a MapInfo Professional linked table. However, the linked table will be read-only and cannot be refreshed by your application. The data is actually from the remote database and does not reflect the data in the local linked version.

You can create a .TAB file to provide access to remote data. To generate a .TAB file using MapInfo Professional, choose File > Open a DBMS table.

The .TAB file is a text file; you can create a .tab file using any text editor. Once you have created the .tab file, you can access it the way you access any other MapInfo .TAB file programmatically through the Catalog object or through the Workspace Manager.

Accessing Remote Tables Without a .TAB File

An application does not need a .TAB file to access remote data. The following code sample illustrates this process.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoServer(ByVal connection As _
     MIConnection)
   ' Note: Do not specify any columns. These are determined
```

```
' dynamically from the query

Dim ti As TableInfoServer = New TableInfoServer("Provinces")
ti.ConnectString = "SRVR=ontario;UID=mapx;PWD=mapx"
ti.Query = "Select * From Provinces"
ti.Toolkit = ServerToolkit.Oci

ti.CacheSettings.CacheType = CacheOption.Off ' On is the default
Dim tbl As Table = connection.Catalog.OpenTable(ti)
```

End Sub

Mapping DBMS Data with X/Y Columns

You can access data from a DBMS table that has X/Y coordinates. You need to create a MapInfo_MapCatalog and register the tables as SpatialType 4.0 and specify two column names as the coordinates. The columns should be indexed on the table. Connect to the DBMS via ODBC and specify the new columns as “Obj” or “MI_Geometry” in your query.

Accessing Data from Oracle

To connect to an Oracle database from a MapXtreme application, the Oracle OCI connectivity client must be installed and appropriate permissions granted. See your Oracle documentation for detailed information.

Geometry Conversion

The table below shows the translation from MapXtreme objects to Oracle Spatial (SDO_GEOMETRY).

From MapInfo	To Oracle
NULL geometry	NULL
Point	1 POINT
MultiCurve (with IsLegacyLine = true)	2 LINESTRING Geometry contains one line string
Polygon	3 POLYGON Geometry contains one polygon.

From MapInfo	To Oracle
FeatureGeometryCollection	4 Collection Geometry is a heterogeneous collection of elements.
MultiPoint	5 MULTIPOINT
MultiCurve	6 MULTILINESTRING Geometry has multiple line strings.
MultiPolygon	7 MULTIPOLYGON Geometry has multiple polygons.
Ellipse	NULL
LegacyArc	NULL
Rectangle	NULL
LegacyText	NULL
RoundedRectangle	NULL
PieTheme, BarTheme	NULL

The table below describes the translation from Oracle (GTYPES) to MapInfo Spatial objects.

From Oracle GTYPES	To MapInfo
0 *UNKNOWN_GEOMETRY (Spatial ignores this geometry.)	
1 POINT Geometry contains one point.	Point
2 LINESTRING Geometry contains one line string.	MultiCurve
3 POLYGON Geometry contains one polygon.	MultiPolygon
4 *Collection Geometry is a heterogeneous collection of elements.	FeatureGeometryCollection
5 MULTIPOINT Geometry has multiple points.	MultiPoint

From Oracle GTYPES	To MapInfo
6 MULTILINESTRING Geometry has multiple line strings.	MultiCurve
7 MULTIPOLYGON Geometry has multiple polygons (more than one exterior boundary).	MultiPolygon

*Some data loss may occur when translating to or from MapInfo object format. They are interpreted (when possible) as single point SDO_POINTTYPE values if not already NULL. They “grab” the first point in the ordered array which would be interpreted as a NULL geometry.

Oracle Support for Z and M Values

MapXtreme supports reading and writing Oracle GTYPEs with Z and M values. The presence and order of Z and M is determined by inspecting the DIM_INFO array in the USER_SDO_GEOM_METADATA for the table. MapXtreme checks for the following dimension names (case insensitive):

- For Z dimension: "Z", "Elevation", " Depth" and "Z Dimension"
- For M dimension: “M", "Measure", and "M Dimension"

Tables that contain M and/or Z values now return FeatureGeometry objects that contain the data for the dimensions present. FeatureGeometry instances inserted or updated into an Oracle table will preserve each of the four dimensions of the new geometry (XYZM) that the Oracle table is defined to support. For geometries containing dimensions unsupported by the Oracle table, the values for those dimensions are ignored during insert/update operations. For geometries not containing dimensions supported by the Oracle table, NULL values are supplied for the missing dimensions during insert/update operations. For example, when inserting a geometry with no Z or M values into a table that is defined with dimensions x, y, and m, the M values stored in the table will be NULL.

The success of any of these insert/update operations may also be dependent upon additional server-side validation including explicit column constraints and validation of values against the dimensional extents specified in the SDO_GEOM_METADATA system table.

SDO_GEOMETRY Arc and Circle Translation

Circles and circular arcs can be resolved to MultiCurves with a resolution of 25 segments per 360 degree circle.

Visualization of Non-translatable Oracle Objects

An Oracle Spatial Object that your MapXtreme application is unable to translate produces a Point object with a default style (a black star) at the location of the SDO_Spatial point, or the first SDO_Spatial ordinate in the ordinate array. This is to enable a visual representation of the non-translatable object in the proper geographic area to which it belongs. Examples of non-translatable objects are user-defined objects GTypes 0,4,5, or invalid SDO_geometries containing unrecognized GTypes, ETypes, or interpretations. The second class should also fail using SDO_VALIDATE_GEOMETRY().

Centroid Support

A MapXtreme application uses the SDO_POINT as the centroid value for polygons. This centroid feature is used to position labels, and also affects the tool selection of the object. The Oracle SDO_GEOMETRY.SDO_POINT_TYPE field (if not NULL) is interpreted as the feature centroid if the point exists inside the region. If the point exists outside of the region, its centroid is calculated as always.

ⓘ There is currently no method or tool in MapXtreme to set the centroid of a region feature, but one may read and use a stored centroid.

Oracle Spatial Reference Support (SRID)

An Oracle SDO_GEOMETRY column may be defined with a spatial referencing system. This is done by providing the Oracle SRID in the USER_SDO_GEOM_METADATA and also by assigning that SRID in the stored SDO_GEOMETRY values. If a table contains an Oracle Spatial column with an assigned SRID, your MapXtreme application is able to query and properly interpret the data. The MapInfo_MapCatalog must contain the same MapInfo Professional CoordSys string as indicated in the SRID of the data, since it is the Coordsys in the MapInfo_MapCatalog that is currently used to interpret and update the data.

If the Spatial column does not contain an SRID value, (the value is NULL), your MapXtreme application is also able to interpret the data via the MapInfo Professional Coordsys defined in the MapCatalog.

When loading tables that use the Latitude/Longitude coordinate system (Geodetic Data) to Oracle Spatial, it is important to verify that all geometry coordinates are between (-180,180) longitude and (-90, 90) latitude. Geodetic data coordinates beyond that range

are not supported in Oracle Spatial and may cause problems. You can check your data using MapInfo Professional before loading, or use the Oracle Spatial SDO_GEOM.VALIDATE_LAYER() function on the table after loading it to Oracle Spatial.

OCI Connection Dialog

The MapInfo.Data.DBMSConnectionCollection class supports a ConnectionFailed event by subscribing to the ConnectionFailedEvent event handler. When fired, this event displays an OCILoginDlg to give the user an opportunity to change the login information and retry the connection to the Oracle database one additional time. This handler is specific to the MapXtreme OCI toolkit. The event is also available through the LoadMapWizard class in the MapInfo.Windows.Dialogs namespace.

Accessing Data from MS SQL Server

MapXtreme supports data stored in Microsoft's SQL Server 2008, SQL Server 2012 and SQL Server 2014. The following information pertains to SQL Server 2008.

SQL Server 2008 Support

MapXtreme supports reading and writing data from and to Microsoft's SQL Server 2008, including the spatial data types GEOMETRY and GEOGRAPHY, along with M and Z value support for both spatial formats.

To access data from SQL Server 2008, MapXtreme requires SQL Server Native Client 10. Data is then handled like data from other remote database management systems that MapXtreme supports[1]. Use the MapInfo.Data.TableInfoServer class to define the connection string and an SQL statement to execute on the remote table. Internally, MapXtreme uses ODBC to access the remote database.

1. For a complete list see Installation Requirements on page 22.

The following table shows how objects are handled in MapXtreme given a specific object type from SQL Server 2008.

SQL Server 2008 Spatial GEOGRAPHY or GEOMETRY	MapXtreme FeatureGeometry
Sql Server 2008 Spatial GEOGRAPHY/GEOMETRY	FeatureGeometry
Point	Point
LineString	MultiCurve
Polygon	Multipolygon
MultiPoint	MultiPoint
MultiLineString	MultiCurve
MultiPolygon	MultiPolygon
GeometryCollection	FeatureGeometryCollection
GeometryCollection containing only Points and/or MultiPoints	MultiPoint
Geometrycollection containing only LineStrings and/or MultiLineString	MultiCurve
Geometrycollection containing only Polygons and/or MultiPolygons	MultiPolygon
An EMPTY GEOMETRY/GEOGRAPHY, e.g., Point empty	NULL

This table shows how a MapXtreme FeatureGeometry is written back to SQL Server 2008

MapXtreme FeatureGeometry	SQL Server 2008 Spatial GEOGRAPHY or GEOMETRY
Point	Point
MultiPoint	MultiPoint
MultiPoint containing only one Point	Point

MapXtreme FeatureGeometry	SQL Server 2008 Spatial GEOGRAPHY or GEOMETRY
MultiCurve	MultiLineString
MultiCurve containing only one Curve comprised of two points	LineString
Multipolygon	MultiPolygon
FeatureGeometryCollection	GeometryCollection *
Rectangle	NULL
RoundedRectangle	NULL
Ellipse	NULL
LegacyArc	NULL
LegacyText	NULL

* This GeometryCollection may contain any or all of the following types: MultiPoint, MultiLineString, and MultiPolygon.

SQL Server 2008 provides new types for date and time information. The following table shows how date and time types are mapped to MapXtreme date and time types.

SQL Server	MapXtreme
Date	Date
Time	Time
DateTime	DateTime
SmallDateTime	DateTime
DateTime2	DateTime
DateTimeOffset	No support

Spatial tables from SQL Server 2008 must be registered in the MapInfo_MapCatalog so that MapXtreme understands what it reads.

The MapCatalog provides four new spatialcolumn values to represent SQL Server 2008 tables:

17.x for GEOMETRY without M and Z values

18.x for GEOGRAPHY without M and Z values

20.x for GEOMETRY with M and Z values

21.x for GEOGRAPHY with M and Z values.

Data can be uploaded using MapInfo Professional or EasyLoader or you can use MapInfo Professional to make existing data mappable, which will create the entry in the MapCatalog. See The MapInfo_MapCatalog for more information on the MapCatalog.

MapXtreme supports SQL Server 2008 tables created in MapInfo Professional (table versions 900, 950 and 1000) and EasyLoader.

DBMS Connection String Format

ODBC Connection String Format

The format of the ODBC connection string is defined by several clauses separated by semicolons (;). Each clause has the form Key=Value. Important keys are listed below.

Keyword	Description
DLG=	A number that controls the display of the connection dialog box: 0 – Suppresses the connection dialog. 1 – Displays the connection dialog. 2 – Displays the connection dialog, but only if needed (i.e., if not all required information was provided) [default]. ⓘ ASP.NET Applications which use Pooling must have a DLG=0 clause in the connection string for ODBC.
DSN=	Specifies the ODBC data source name. **Caution:** If you use the DSN= syntax key, the name that you specify must match the data source name in use on the user's system. Note that different users might use different names to refer to the same data source. If you cannot know in advance what data source name to use, use the DRIVER= syntax key instead of the DSN= syntax key.

Keyword	Description
DRIVER=	Specifies the exact driver name of the installed driver. Used instead of the DSN= syntax key. **Example:** `DRIVER={SQL Server}` ⓘ This does not apply to Oracle Spatial.
UID=	Specifies the desired UserId for the data source, if required.
PWD=	Specifies the user's password for the data source, if required. Passwords do not need to be in the connection string for the two strings to match. If two tables are in the same database, the connection string is the same.

ⓘ Connection attributes/parameters do not have to be in order and one may use a dialog to get a connection from an existing connection pool to avoid redundant connections. In previous version of our API, if you used a dialog each time to connect to the same database, or if you did not order the connection keywords in the connection string in the documented order, the connection would not be shared and you would get multiple connections.

ODBC Layers and Pooling in Web Applications

When adding remote layers via ODBC to an ASP.NET application that uses pooling, be sure to have the DLG=0 clause in the connection string. This will avoid the display of unnecessary user and password dialogs that can time out. This applies to TAB files and workspaces. The following connection string example shows the highlighted DLG=0 clause.

```
<ConnectString>DRIVER={SQL Server};DATABASE=Devel;Server=Paladin;
UID=devel;PWD=devel;QuotedID=Yes;Trusted_Connection=No;
Network=DBMSSOCN;Address=PALADIN,1433;DLG=0</ ConnectString>
```

When using TableInfoServer and pooling is on, to access SQL Server specify "DLG=SQL_DRIVER_NOPROMPT" in the connection string to avoid throwing a MapInfo.Data.TableException: Unable to open table.

Oracle Spatial Connection String Format

These are the Oracle Spatial keywords. The string is defined by several clauses separated by semicolons (;). Each clause has the form Key=Value. Important keys are listed in the table below.

Keyword	Description
SRVR=	Reflects the service name for the server set in the Oracle Net8 EasyConfig utility. This is required for Oracle connectivity, but does not apply to ODBC connections.
UID=	Specifies the desired UserId for the data source, if required.
PWD=	Specifies the user's password for the data source, if required.

Sample Connection Strings

Here are sample connection strings for Oracle Spatial and Microsoft SQL Server 2008 ODBC drivers.

Oracle Spatial connection string:

```
UID=george;PWD=password;SRVR=OracleSpatial9i
```

Microsoft SQL Server 2008 connection string:

```
DRIVER={<driver>};
SERVER=<server>;UID=<uid>;PWD=<pwd>;Database=<database>
```

where <driver> for SQL Server Spatial should be the most current available, SQL Server Native Client 10.0 or higher version.

Defining Mappable Tables in Server Table Queries

The query you specify for a server table defines the result set of data from your DBMS that represents the data in the table being added. You can formulate a fairly complex query to do powerful server-side analysis that defines a mappable table in MapXtreme. Your MapXtreme application uses this query internally to access the data.

MapXtreme generates several internal queries based on your query to access the data in a map as well as selection/key based queries. The table from which the geometry column is selected must be registered in the MapInfo MapCatalog on the server. MapXtreme requires this to obtain certain metadata about the geometry column such as the coordinate system, spatial type, and default styles.

In order for a query to define a mappable table, the query must contain both a geometry column and a key column. Sometimes for more complex queries on small sets of data (where the spatial indexing or spatial predicate cause the query to fail), you can specify TableInfoServer.MbrSearch=false to enable the results to be mapped.

The Geometry Column

If you do not specify a geometry column that your MapXtreme application can recognize, the table is opened, but cannot be added to a map (the table is unmappable). MapXtreme determines the geometry column of the table by looking it up in the MapCatalog and by examining the result set datatype of the column. You can reference the geometry column generically via the pseudo column name “Obj”, or you may refer to the geometry column using its specific column name. This form is required to reference the geometry column for an X/Y mappable layer. You can specify a geometry column via any server-supported geometry function/expression.

Example

```
Select Obj from rdbsdata
Select sw_geometry from rdbsdata
select sw_member, ST_Buffer(geometry, 66.0, 0.1) from rdbsdata
     // a geometry function
Select st_geometry(st_point(72.5, 42.5) from rdbsdata
     // a geometry constructor
```

Oracle sdo_buffer example:

```
Select mi_prinx, mdsys.sdo_geom.sdo_buffer(geoloc, (select diminfo from
sdo_geom_metadata where table_name = 'ALINE'), 20) from aline where prinx = 1
```

Oracle constructor example:

```
Select 1 "mi_prinx",
mdsys.sdo_geometry(3,null,null,mdsys.sdo_elem_info_array(1,3,3),
mdsys.sdo_ordinate_array(-79.919909,40.553465,-71.060457,45.363657)) from dual
```

SQL Server 2008 Spatial function example:

```
select location_id, geography::Point(lat, long, 4326 /*WGS84*/) as geog from
dbo.store_locations
```

The Key Column(s)

A key column(s) must be returned in the query to enable it to be opened as a table. This is what enables your MapXtreme application to identify each row in the result set to perform shading, selection, and label operations on the layer.

The key column does not need to be specified in the query in most cases.

Your MapXtreme application can look up and determine the best key column(s) to use in order to uniquely reference a row in the result set, and then add them to the query if they are not present. In most cases, this is the primary key/unique index.

For Oracle Spatial tables, the MI_PRINX may be used.

For some queries, it is not possible for your MapXtreme application to identify the key. This is the case in a query on a view or a synonym. The view or synonym must appear in the MapInfo MapCatalog. They also must be registered as required with the underlying Spatial index system in most cases. Since MapXtreme cannot determine the key on these, a mechanism is provided to allow the application developer/query writer to identify the key column in the result set. The key must be a single column and must be a distinct value in the result set. To identify the column that is to be used as the key column, you can specify column alias of prinx or mi_prinx, (e.g., select custid mi_prinx, custname, Obj from mycust).

Example

```
Select customer_id mi_prinx, obj from customer_view
```

The column alias “mi_prinx” is used to identify and use the customer_id column as the key column for the table. You can alternately alias the desired key column in the create view statement to identify the key column automatically for any query on that view.

Example

```
Create view customer_view as select customer_id mi_prinx, geoloc from customer
```

In general, if a column name or column alias of prinx, or mi_prinx is found in the result set, that column is used as the key column for the table. This enables the application/query writer to specify the key column they desire.

Accessing Attribute Data

To use all available data columns, specify a query such as Select * From tablename. You are not required to specify * (asterisk); instead, you can designate specifically which columns you want to use. For the best performance, limit your query so that it retrieves only the needed columns.

When you add a DBMS table, for performance sake, you should only specify the columns in the query that you intend to use in your application. These are the spatial column, the key column(s), which are added automatically if you do not specify them, and columns you want to label with, or create a theme from. You may use the pseudo columns “OBJ” for any mappable table to refer to the column(s) containing the spatial data. This is required for a table using the MapMarker MDIGEOADDRESS column on a table with an X/Y column.

You can use any server side expression/function to specify a column. Also, avoid select * from tab in a real application.

The following code example defines a server table using a TableInfoServer and adds a layer to a map using this definition. You can now label or place themes based upon the columns in this table.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfoServer(ByVal connection As _
MIConnection)
   ' Note: Do not specify any columns. These are determined
   ' dynamically from the query

   Dim ti As TableInfoServer = New TableInfoServer("Provinces")
   ti.ConnectString = "SRVR=ontario;UID=mapx;PWD=mapx"
   ti.Query = "Select * From Provinces"
   ti.Toolkit = ServerToolkit.Oci

   ti.CacheSettings.CacheType = CacheOption.Off ' On is the default
   Dim tbl As Table = connection.Catalog.OpenTable(ti)
End Sub
```

Performance Issues

Establishing a connection with the database server may take several seconds. This is a one-time cost, incurred when the table is first opened.

The map-display speed depends on how much data is retrieved from the server. In some cases, displaying a map from a server is noticeably slower than displaying a map from a local file. Speed also depends on whether your MapXtreme application has already cached the map features that are being displayed.

Working with the Cache

Knowing how to work with cache in MapXtreme enables you to improve your application's performance. The sections below describe what the cache is, how it works in the MapXtreme object model, and the CacheSettings property of the TableInfoServer object.

What Is the Cache?

In place of local files, applications can access MapXtreme features from a remote database. In place of reading these records from the database each time the map needs to be acted upon, your MapXtreme application can temporarily store these records in the cache. This limits the number of calls between the application and the remote database. Records in a server table can be cached to improve the performance of your application (e.g., drawing, thematics, labeling, etc.). Server table data is cached internally as it is read and drawn to the Map window. All subsequent redraws read from this cache instead of going to the server database for the same data. The cache is able to offer significant redraw performance improvement.

There are several settings that developers can use to customize cache usage. The cache can be enabled (or disabled) when the server table is added by specifying the values for the CacheSettings property of the TableInfoServer object and is On by default.

How the Cache Works

For each record that is cached, each attribute data value is stored in memory and each feature object is stored on disk in a temporary Rtree file. For tables with a lot of records and/or a large record size (for example, number of bytes per record for the attribute data), caching may use a significant amount of memory. If an application tries to cache too much data, too much virtual memory usage may be required, which can degrade performance. Applications should be selective about how the cache is utilized. MapXtreme offers a variety of mechanisms for controlling the cache.

The TableInfoServer Object and the CacheSettings Property

When a table is added to the map, the cache is enabled by default but can be further configured using the CacheSettings property of the TableInfo object. This property has four possible values: ON, OFF, ALL, USER, with ON being the default for TableInfoServer, OFF is default for other TableInfo objects.

Parameter	Description
OFF	A value of 'Off' means that the table will not use the cache at all. All data operations will go directly to the database server.
ON	Caching is enabled and the table automatically performs caching based on the map view (center/zoom). The user may additionally control the cache through the cache constraint objects. The cached table maintains the record cache in a fashion that best improves standard map operations. The cache is maintained to contain, at a minimum, all the records displayed in the current window of each Map the table is in (and visible). Once an initial map window has been cached, pan and zoom operations that fall entirely within the initial extents of the cache access the cached records and do not need to query the database. If a pan/zoom operation falls outside the cached region, the table adds the new map window MBR (view) to the cache and obtains the missing records from the database server and adds them to the cache. The old map view is not initially discarded; rather, an internal history of previous map views is maintained. To avoid having a cache that grows excessively large, there are controls that can be placed on the table's cache to determine when to discard old cached views (map window MBR regions). These controls are properties of the CacheParameters object, which can be set at the time the table is initially opened. This allows the developer to set limits on the maximum amount of memory or disk space used by the cache, the maximum number of previous map window views to maintain in the history, the maximum number of records to maintain in the cache, and/or the maximum amount of time old map window views are allowed to remain in the cache history. These limits can be used individually or in combination to provide the cache management that best suits the application's needs. ON is the default setting for the CacheParameter setting for a TableInfoServer. For other TableInfo data sources, the default is OFF. For example, TAB files are not cached by default.

Parameter	Description
USER	A value of USER for the LayerInfo CACHE parameter means that your application creates a cache, but the only records that are placed in the cache are those specified by the application developer. The mechanisms available for specifying which records are placed in the cache are BoundConstraint, FeaturesConstraint, and AllFeaturesConstraint objects. The word constraint implies that these objects are constraining the cache to include the specified records. The BoundsConstraint object can be used to place all records into the cache for which the MBR of the feature intersects the MBR of the constraint. A FeaturesConstraint object can be used to add specific records to the cache. For example, if an analysis is going to be performed that involves multiple steps and/or reads of the Feature or RowValues of the feature, possibly on a set of features returned from a Layer.Search, Layer.SearchWithinDistance, etc., it may be advantageous to place these records into the local cache for the duration of the analysis and remove them when finished. The FeaturesConstraint provides this capability. If an application is going to perform an analytically intensive operation that may hit every record, it may be desirable to temporarily cache the entire set of data for the layer. This is accomplished by using the AllFeaturesConstraint. These cache constraint objects can also be used when the cache is set to ON. In this case, they may add records to the cache but have no effect on the cache's history of previous map window views. The constraint objects can also be used when the cache is set to OFF or ALL in which case they have no effect. ⓘ The constraint objects have no effect on non-server tables.
ALL	The entire table is cached. With this option, the table's data is retrieved from the server once and accessed locally from that point forward. To refresh the data in the cache, use the Refresh method on the table.

If you try to cache too much data or too many tables, virtual memory usage may be forced, and performance gain could be lost.

Cache Storage Type:

Cache Storage type indicates the table type used to store the cached records of RDB server tables. Following Storage Types are used in MapXtreme:

1. **Native**: The cached records are stored on disk in a temporary MapInfo Native table. If multiple tables are opened with this cache storage type, then a different set of MAP, DAT, etc. files will be created for each table. This storage type has a maximum file-size limits of 2GB.
2. **MemTable**: The cached records are stored in memory in a temporary MemTable table. Geometry objects are stored on disk in a temporary MAP file. If multiple tables are opened with this cache storage type, then a separate set of in-memory cache and on-disk MAP files will be created for each table. This storage type also has an upper size limit of 2GB.
3. **NativeX**: The cached records are stored on disk in a temporary MapInfo NativeX table. If multiple tables are opened with this cache storage type, then a different set of MAP, DAT, etc files will be created for each table. This NativeX storage type supports UTF-8 and UTF-16 charsets as well as it can store the data files that are greater than 2GB in size.
4. **MemNativeX**: The cached records are stored in memory in a temporary MemNativeX table. Geometry objects are stored on disk in a temporary MAP file. If multiple tables are opened with this cache storage type, then a separate set of in-memory cache and on-disk MAP files will be created for each table. MemNativeX cache supports UTF-8 and UTF-16 charsets as well as it can store the data files that are greater than 2GB in size.
5. **Geopackage**: The cached records are stored on disk in a temporary Geopackage database(*.GPKG) file. All RDB server tables opened in a session with this cache storage type are cached in a single Geopackage database and for each open RDB table, a separate Geopackage cache table is created in that single Geopackage database.
6. **MemGeopackage**: The cached records are stored in an in-memory Geopackage database.

The MapInfo_MapCatalog

In order to display data on a map, your MapXtreme application needs to access a special table, known as the MapInfo_MapCatalog. One catalog must be created per database before any tables in that database can be viewed as a map layer in a MapXtreme

application. The MapCatalog must contain information about the spatial columns in each of the mappable tables you want to access from the database. The MapInfo EasyLoader utility automatically inserts the appropriate row into the MapInfo_MapCatalog when the table is uploaded into the database.

Your application can use a MapInfo_MapCatalog that already exists on the server. (This same catalog is shared by various MapInfo client applications). If there is no MapInfo_MapCatalog on the server, you need to create one. MapXtreme supports the storage of style information for individual features in remote databases.

Loading Spatial Data to DBMS

If you have spatial data in the form of a MapInfo table, you can import it into your DBMS database.

To load data into Microsoft SQL Server and Oracle Spatial, use the MapInfo EasyLoader, that is distributed with MapInfo Professional and available for download from http://support.precisely.com. The EasyLoader utility automatically creates a MapInfo_MapCatalog when you upload a table, if there is no MapInfo_MapCatalog already present.

Manually Creating a MapInfo MapCatalog

If you are not a MapInfo Professional or EasyLoader user, you or your database administrator will need to create the MapCatalog manually, as described below. You only have to create the MapCatalog once per server/database.

1. Create the user MAPINFO in the specific database where the mappable tables are located.
2. Create the table MAPINFO_MAPCATALOG in the database.

   The Create Table statement needs to be equivalent to the following SQL Create Table statement:

```
Create Table MAPINFO_MAPCATALOG (
SPATIALTYPE Float,
TABLENAME Char(32),
OWNERNAME Char(32),
SPATIALCOLUMN Char(32),
DB_X_LL Float,
DB_Y_LL Float,
DB_X_UR Float,
DB_Y_UR Float,
VIEW_X_LL Float,
VIEW_Y_LL Float,
```

```
VIEW_X_UR Float,
VIEW_Y_UR Float,
COORDINATESYSTEM Char(254),
SYMBOL Char(254),
XCOLUMNNAME Char(32),
YCOLUMNNAME Char(32),
RENDITIONTYPE INTEGER,
RENDITIONCOLUMN CHAR(32),
RENDITIONTABLE CHAR(32)
NUMBER_ROWS INTEGER
)
```

> ⓘ It is important that the structure of the table looks exactly like this statement. The only substitution that can be made is for databases that support varchar or text data types; these data types can be substituted for the Char data type.

3. Create a unique index on the TABLENAME and the OWNERNAME, so only one table for each owner can be made mappable.

```
create unique index mapcat_i1
on mapinfo.mapinfo_mapcatalog (OwnerName,TableName)
```

4. Grant Select, Update, Insert, and Delete privileges on the MAPINFO_MAPCATALOG. This allows users to make tables mappable.

```
grant select, insert, update, delete on mapinfo.mapinfo_mapcatalog to public
```

Adding Rows to the MapInfo_MapCatalog

For each spatial table that you want to access from your application, you need to add a row to the MAPINFO_MAPCATALOG table. If you do not use MapInfo Professional to manage the MapInfo_MapCatalog, you will have to add rows to the MAPINFO_MAPCATALOG table manually.

The following table describes the syntax and meaning of each column:

Column Name	Values to Assign	Examples
SPATIALTYPE	**MapInfo Spatial Object Format** 1: Point layer in X/Y columns indexed with micode (a serialized quadtree key) 4: Point layer in X/Y columns 5.x: SpatialWare for Oracle 6.x: Ingres SQL - Not Supported 7.x: Sybase SQS - Not Supported 8.x: Oracle SDO version 2 - Not Supported 9.x: MapInfo Geocoding DataBlade SpatialWare Point Module 10.x: MapInfo Geocoding DataBlade XY Module 11.x: SpatialWare IDS/UDO datablade 13.x: Oracle Spatial 14.x: SpatialWare for Microsoft SQL Server 17.x:SQL Server 2008 GEOMETRY with-out M and Z values 18.X: SQL Server 2008 GEOGRAPHY with-out M and Z values	14.0 = SQL Server 14.1 14.2 14.3

SPATIALTYPE (continued)	20.x: SQL Server 2008 GEOMETRY with M and Z values 21.x: SQL Server 2008 GEOGRAPHY with M and Z values Spatial Object Type x.0: Points only x.1: Lines only x.2: Regions only x.3: All types supported ⓘ This column describes the Spatial Object Format of how the data is stored and indexed and the Spatial Object type(s) supported and not supported in the column. The digits to the left of the decimal point are the Spatial Object Format. The digits to the right represent the type of Spatial Object Type that can be stored in the column. Maps to MapInfo.GeometryColumn.PredominantGeometry Type, and Has<Line/Point/Region/Text>Geometries	
TABLENAME	The name of the table.	STATES

SPATIALCOL-UMN	The name of the column, if any, containing spatial features: SW_GEOMETRY (mappable using Spatial-Ware Type/IDS/UDO) NO_COLUMN (mappable using X–Y) MI_SQL_MICODE (mappable using MI Code) Or the name of the IDS/UDO, or Ora-cle column that is ST_SPATIAL datatype. Name of the Oracle SDO_GEOMETRY col-umn.	SW_GEOMETRY
DB_X_LL	The X coordinate of the lower left corner of the layer's bounding rectangle, in units that are indicated by the COORDINATESYS-TEM (see below). Maps to Map-Info.Data.GeometryColumn.Bounds	-360
DB_Y_LL	The lower left bounding Y value.	-90
DB_X_UR	The upper right bounding X value.	360
DB_Y_UR	The upper right bounding Y value.	90
VIEW_X_LL	The X coordinate of the lower left corner of the default view. The default view only applies if this is the first table to be opened. Maps to MapInfo.Data.GeometryColumn.Default-View	-180
VIEW_Y_LL	The lower left bounding Y value of the default view.	-45

VIEW_X_UR	The upper right bounding X value of the default view.	180
VIEW_Y_UR	The upper right bounding Y value of the default view.	45
COORDINATE-SYSTEM	A string representing a MapInfo CoordSys clause (but without the keyword CoordSys at the very start), which specifies a map projection, coordinate units, etc. For simple Lon/Lat maps, specify Earth Projection 1, 0. Maps to MapInfo.Data.GeometryColumn.CoordSys	Earth Projection 1, 0
SYMBOL	A MapInfo Symbol clause (if the layer contains only points); or a Symbol clause followed by a Pen clause (indicating styles for linear features) followed by another Pen clause (indicating styles for the borders of regions) followed by a Brush clause. Maps to MapInfo.Data.GeometryColumn.DefaultStyle	Symbol(35,0,12) Pen(1,2,0) Pen(1,2,0) Brush(2,255,255)
XCOLUMN-NAME	For the X/Y mappable tables, specify the name of the column containing X-coordinates. If there is no such column (i.e., if this table uses a single spatial column instead of a pair of X-Y columns) then specify *NO_COLUMN* or leave empty. Maps to MapInfo.Data.SpatialSchemaXY	NO_COLUMN
YCOLUMN-NAME	For the X/Y mappable tables, specify the name of the column containing Y–coordinates, or specify *NO_COLUMN* Maps to MapInfo.Data.SpatialSchemaXY	NO_COLUMN

RENDITION-TYPE	This indicates how the object style information is applied. 0 – Indicates that all the objects in the table will have the style specified in the symbol field of the MapCatalog applied to them. No per-record styles are in effect. Objects will be read/updated using the default style for the table. 1 – Indicates that the table uses per-record styles. The table has a separate column that contains a MapBasic string representation of the style information for each object in the table (the same format that is currently used in the MapCatalog's SYMBOL column). The style column in the table is recorded in RENDITIONCOLUMN.	0 or 1
RENDITION-COLUMN	If RENDITIONTYPE is 1, this field stores the name of the column in the spatial table that contains style information. This column is automatically added to any query against the table and is maintained (updated) as the object is updated. Users should NOT specify this column in their queries as problems can occur with intersect or update statements. Queries which include this column in the select clause (excluding the wildcard character "*") may access the values through the Dataset object. Rows with a NULL value in their style column will have the style from the SYMBOL field of the MapCatalog applied to the object. Creates a MapInfo.Column.DataType with MIDBType.Style	MI_SYMBOLOGY

	Currently not used, but reserved for future use.	Null
NUM-BER_ROWS	Currently used by MapInfo Professional.	Null

Per-Record Styles

Per-record style support brings a feature to spatial database implementations that has long been available in MapInfo TAB files. Specifically, it allows each geometry in a single table to have its own style. For example, a single 'public institution' table in Oracle Spatial can have schools, town halls, libraries, and police departments and each point type would be represented with its own symbol (i.e., a school symbol for all the schools). Similarly, a single road table in SpatialWare SQL Server may have different road types such that streets are shown as a single pixel black line, secondary roads as a double pixel red line and interstates as parallel red lines.

To use per-record styles, your table must be represented with an entry in the MapCatalog with appropriate settings for RENDITIONTYPE, RENDITIONCOLUMN, and RENDITIONTABLE.

ⓘ If these columns are not present, the table's default style will be applied to all objects.

Symbol, Pen, Brush Clause Syntax

If you are manually creating a MAPINFO_MAPCATALOG table to provide support for a remote spatial database, you will need to specify a symbol style, and possibly line and fill styles as well.

Specifying Point Styles

Use a Symbol clause to specify point styles. There are three types of Symbol clauses: one for specifying MapInfo 3.0-style symbols; one for specifying TrueType font symbols; and one for specifying bitmap symbols.

Symbol Syntax	Example

Symbol(shape, color, size)
or
Symbol(shape,color,size,font,fontstyle,rotation)
or
Symbol(bitmapname,color,size,customstyle)
	`Symbol(35,0,12)`
	
	`Symbol(64,255,12,"MapInfo Weather",17,0)`
	`Symbol("sign.bmp", 255, 18, 0)`

Specifying Line Styles

Use a Pen clause to specify line styles. In a MapInfo_MapCatalog, you may need to specify two pen clauses: one to specify the appearance of linear features, and another to specify the appearance of region borders.

Pen Syntax	Example
Pen(thickness, pattern, color)	`Pen(1, 2, 0)`

Specifying Fill Styles

Use a Brush clause to specify the style for closed features (regions).

Brush Syntax	Example
Brush(pattern,color,backgroundcolor)	`Brush(2, 255, 65535)`

The MapXtreme Styles API is discussed in Chapter 15 Stylizing Your Maps. Style patterns are presented in Appendix F: Style Lookups.

Text Objects Limitation

LegacyText objects have their own way of displaying style that is separate from the use of the MI_Style column. Therefore any form of text object needs to be treated differently than other objects. The style for any text object is embedded and a NULL value is inserted into the style column.

Troubleshooting

If you encounter problems with your SpatialWare or Oracle applications, use the following table to help analyze and solve the problem.

Problem Description	Possible Cause	Solution
The table is not matchable.	Data binding was attempted against a SpatialWare layer.	AddColumns is not currently supported for SpatialWare layers.
No object was found using the index that you specified.	A query was made against a table that does not exist.	Check that the table name is correct and in the proper case. Also, the table may need to be mappable.
	No spatial object is contained in the result of the spatial query.	Use the EasyLoader Upload utility to make the table a mappable table.
	A query was made against a non-spatial table.	Check the query for possible syntax errors. Also make sure that the result of the query includes the field specified in the spatial column in the MapInfo_MapCatalog.
Map appears to have incorrect zoom level. For example, the map may be zoomed out too far to identify any geography.	The MBR for a DBMS layer is determined by the MapInfo_MapCatalog table. The table extents in the MapCatalog result in a different zoom level than the one you desire for your output.	Edit the extents (DB_X_LL, DB_X_UR, DB_Y_LL, DB_Y_UR) in the MapInfo_MapCatalog using the MapInfo Professional MBX tool, MISETMBR.MBX.

12

12 – Adding Mapping Capability to Your Applications

We use the MapInfo.Mapping namespaces to add mapping functionality to your application. This chapter explains how to use the Mapping namespaces to enhance your mapping application.

In this chapter:

Introduction to the MapInfo.Mapping Namespace

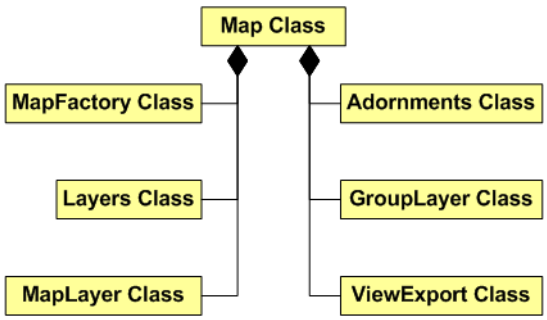
The MapInfo.Mapping namespace contains classes, interfaces, and enumerations for creating, displaying, and exporting maps, layers, modifiers, thematics, legends, and labels.

The Map class is the top level object for a map in a desktop application. Each Map object can contain exactly one map. Each Map has one Adornment collection and one Layer collection. When developing a web application a Map gets attached to a MapExport object in order to export that map image to a bitmap or stream.

As described in Layers, a map is basically a set of layers piled on top of each other. Using the Mapping namespace classes, an application can be designed to manipulate those layers as needed.

Base Mapping Classes

This section discusses the base map classes in use in the MapInfo.Mapping namespace. The diagram below shows a UML representation of the Map hierarchy.



MapExport

The MapExport class is used to export a Map to an image. The properties of this class specify each aspect of the image, such as BorderPen, ExportSize, Format, etc.

MapExport supports several different image formats including: BMP (default), WBMP, WMF, EMF, GIF, J2K, JPG, PNG, PSD, TIFF, TIFFCymk, and TIFFLzw. When performance is an issue, the BMP, GIF, JPG, PNG, and TIFF formats can also be exported using the .NET framework API instead of the LEADTOOLS API. While the export speed may differ, the quality of the images exported is the same for both methods.

To select a format, set the ExportFormat property to one of the above formats (e.g., ExportFormat.WindowsPNG).

ⓘ The LegendExport class also exports files in these formats using the .NET framework and LEADTOOLS APIs.

Example Use of MapExport

If you are using a MapControl, you must clone the map prior to exporting it, as shown in the example below:

VB example:

```
Public Shared Sub MapInfo_Mapping_MapExport(ByVal mapControl1 As MapControl)
    'must clone since map is coming from mapcontrol and is linked to it via the
HWND
    Dim NewMap As Map = CType(mapControl1.Map.Clone(), Map)
    Dim exportObject As MapExport = New MapExport(NewMap)
    exportObject.ExportSize = New MapInfo.Mapping.ExportSize(2931, 4104)
    exportObject.Format = ExportFormat.Gif
    exportObject.Export("C:\Temp\ExportImage.gif")
End Sub
```

Map

The Map class contains everything that you can put in a map. A Map object is placed in a MapControl object for a desktop application or attached to a MapExport object for a web application.

A Map has the following properties:

- Bounds
- Center
- Zoom
- Scale
- Size
- Rotation
- DisplayTransform
- DisplayCoordSys

Each Map has a Layers collection, which holds all the layers that comprise the map (see Layers), and a single Adornment collection, which contains all of the map's adornments. Adornments include Legends, Titles, and Scalebars (see Legends Overview).

MapFactory

The MapFactory class contains the objects used to create maps from Geosets, workspaces, and lists of tables. MapFactory also functions as a container of, and tracks, the collection of all maps created in a particular session.

Example Use of MapFactory

This example creates an empty map 300 by 300 pixels in size using the CreateEmptyMap method of MapFactory.

```
' Create a new map
Dim map As Map = Session.Current.MapFactory.CreateEmptyMap(New Size(300, 300))
```

MapLoader

The MapLoader class provides a mechanism to load the layers of a map from Geoset files, XML Workspace files, or TAB files. For each type of map to load, there is a subclass of MapLoader that is used to load the map. These subclasses include MapGeosetLoader, MapWorkspaceLoader, and MapTableLoader.

VB example:

```
Public Shared Sub MapInfo_Mapping_MapLoad()
    ' Create an empty map
    Dim map As Map = _
MapInfo.Engine.Session.Current.MapFactory.CreateEmptyMap(New Size(400,_
300))
    ' Create a maploader. Make sure that Session.Current.TableSearchPath points
to the folder with the tables in it
    Dim tl As MapTableLoader = New MapTableLoader("ocean.tab", _
"usa.tab", "mexico.tab", "us_hiway.tab")
    ' Load tables into a map
    map.Load(tl)
End Sub
```

MapViewList, MapView

These classes contain the objects that help to maintain a list of the previous and next views for a Map. Use the MapViewList class as a convenience to step through your stack of map views, displaying them with the MapView class which provides the specifics (Name, Center, and Zoom) of each map view.

MapXtreme provides a VB sample for MapViewList in <*MapXtreme Install Directory*>\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features.

MapControl

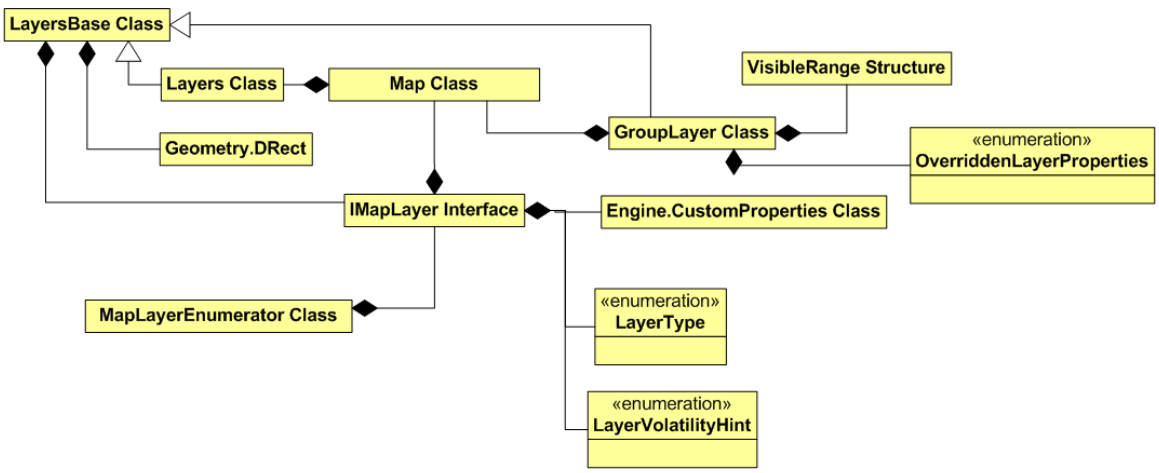
The MapControl class contains the objects that allow the user to visualize a map on a screen. Add a MapControl to a Windows Form to view a map. A MapControl also holds the MapTools collection. A version of MapControl exists for web applications as well.

The MapControl class for use in desktop applications is in the MapInfo.Windows.Controls namespace. For a working example of a desktop MapControl, run Workspace Manager from the Start menu. See also Chapter 27 Workspace Manager.

The MapControl class for use in web applications is in the MapInfo.WebControls namespace. See Web Control Architecture. For a tutorial on how to create an overview map for a web application, see the Tutorials section of Learning Resources, available from the Start menu.

Layers

The following section discusses the Layers objects and classes. The diagram below is a UML representation of the Layers hierarchy.



FeatureLayer

A FeatureLayer is a layer that displays Features from a Table. For example, a layer of region objects representing world countries is a FeatureLayer. FeatureLayers can be native .TAB data, remote RDB, seamless, or raster data.

A new Opacity property has been introduced only for vector layers in version 8.1 and later. It enables you to set the translucency with which the feature will be drawn. However, if any feature style modifiers (explained later in this chapter) are applied on the layer, then their opacity will be used for rendering instead.

Layers

Each map has a collection of FeatureLayers represented by the Layers class. The order in the collection is the order in which the layers are drawn. Methods in the collection class include Add, Insert, Move, and Delete. The Layers collection is derived from the LayersBase class. The best way to enumerate through the layers is to use a Layer filter.

In order to change a layer it first needs to be made editable. This can be done changing its setting in the LayerControl or programmatically by changing its EditMode property. Once a layer is editable, features in that layer can be moved, resized, or deleted.

ⓘ Any edits made to a particular layer take effect immediately so exercise caution when selecting features in an editable layer.

To implement a filter use the MapLayerFilterFactory class. This class allows you to create a filter from a set of stock filters such as layer types or all visible layers. To create your own filter write a class that implements the interface, IMapLayerFilter.

MapLayer

The MapLayer class is the base class of any layer. This class implements the IMapLayer interface. Properties include Enabled, VisibleZoomRange, Name, and Alias. Use this class to access generic layer properties.

UserDrawLayer

The UserDrawLayer is an abstract class that allows you to override the draw method to draw your own layer. It provides an efficient way to add custom objects on top of a MapControl using Windows drawing methods, instead of creating new features using map coordinates and actually adding them to a map.

ObjectThemeLayer

The ObjectThemeLayer class contains one of three different types of themes (pie chart, bar chart, and graduated symbol). This layer behaves just like other layers and can be grouped, have visibility set, etc.

GroupLayer

This is class represents a group of layers that are moved and turned on/off in unison. A GroupLayer is a LayersBase collection combined with an IMapLayer. This object also supports the ability to do AnimationLayers.

If the layer is in a group with a VolatilityHint equal to Animate then only those layers within that group need to be redrawn. If the Layer has a VolatilityHint equal to CacheIfPossible or Normal then all layers need to be redrawn.

LabelLayer

The LabelLayer class is responsible for generating labels and drawing them at a particular layer position in a map. See LabelLayer and Label Layer Settings.

GraticuleLayer

This class is used to display a grid of longitude and latitude lines in the map window. See Graticule Layers.

Layer Filters

The IMapLayersFilter and IMapLayersFilterFactory interfaces provide support for layer enumeration.

IVisibilityConstraint

The IVisibilityConstraint is an interface which determines whether a particular object is visible or not. This interface is implemented by a wide number of types, including all Layer types, LabelModifiers, FeatureStyleModifiers, and Themes.

Code Example: Animation Layer

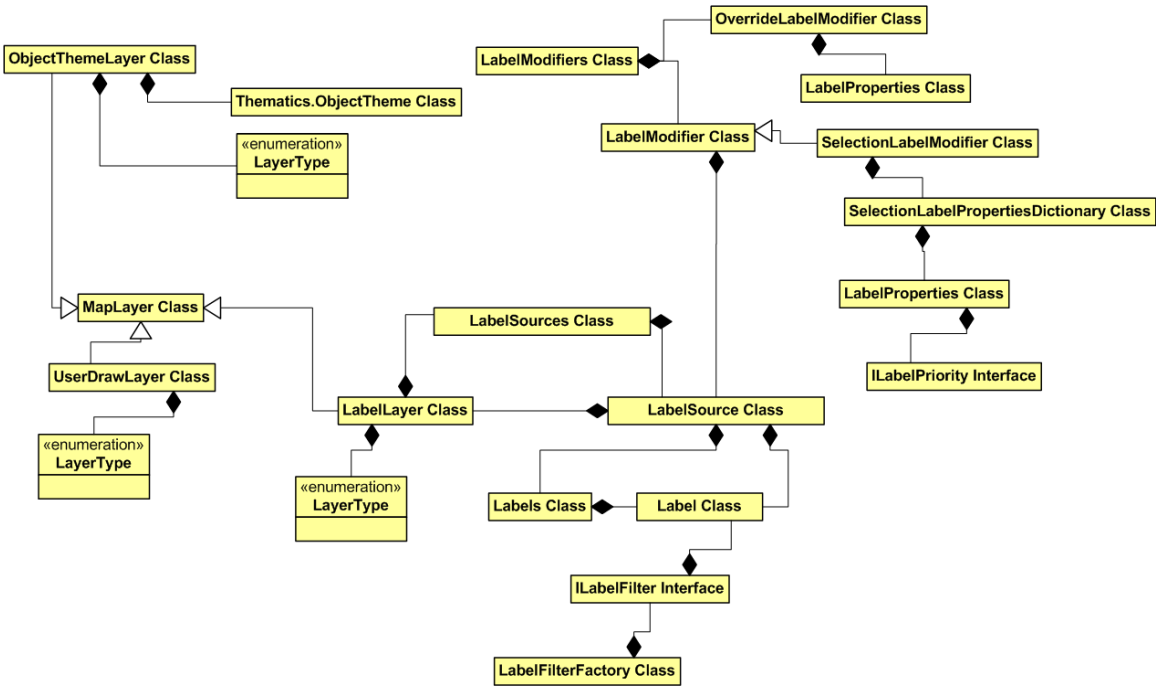
The following VB sample shows how to set the animation of a layer.

```
Private Sub btnInitializeObjects_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnInitializeObjects.Click
Dim Cat As Catalog = MapInfo.Engine.Session.Current.Catalog

'Create Temp layer
Dim tblInfoTemp As New TableInfoMemTable("Animation")
Dim tblTemp As Table = Cat.GetTable("Animation")
If IsNothing(tblTemp) = False Then 'Table exists close it
Cat.CloseTable("Animation")
End If
tblInfoTemp.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn(Map_
Control1.Map.GetDisplayCoordSys()))
tblInfoTemp.Columns.Add(ColumnFactory.CreateStyleColumn())
tblInfoTemp.Columns.Add(ColumnFactory.CreateStringColumn("Name", 40))
tblInfoTemp.Columns.Add(ColumnFactory.CreateStringColumn("Dept", 15))
tblInfoTemp.Columns.Add(ColumnFactory.CreateIntColumn("Level"))

tblTemp = Cat.CreateTable(tblInfoTemp)

Dim lyr As New FeatureLayer(tblTemp)
MapControl1.Map.Layers.Add(lyr)

'Create Points
Dim pt As FeatureGeometry = New Point(lyr.CoordSys, New DPoint(-76, 42))
Dim cs As New CompositeStyle(New SimpleVectorPointStyle(37, _
   System.Drawing.Color.Red, 10))
Dim ftr As New Feature(tblTemp.TableInfo.Columns)
ftr.Geometry = pt
ftr.Style = cs
ftr.Item("Name") = "Kelly"
ftr.Item("Dept") = "Sales"
ftr.Item("Level") = 3
tblTemp.InsertFeature(ftr)

Dim pt2 As FeatureGeometry = New Point(lyr.CoordSys, New DPoint(-119, 34))
Dim cs2 As New CompositeStyle(New SimpleVectorPointStyle(44, _
   System.Drawing.Color.Purple, 10))
Dim ftr2 As New Feature(tblTemp.TableInfo.Columns)
ftr2.Geometry = pt2
ftr2.Style = cs2
ftr2.Item("Name") = "Greg"
ftr2.Item("Dept") = "Marketing"
ftr2.Item("Level") = 2
tblTemp.InsertFeature(ftr2)
End Sub

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
Dim cat As Catalog = MapInfo.Engine.Session.Current.Catalog
Dim tbl As Table = cat.GetTable("Animation")
If IsNothing(tbl) = False Then
'Update the position of the points
```

```
Dim si As SearchInfo = MapInfo.Data.SearchInfoFactory.SearchWhere("Name= _
   'Kelly'")
Dim ftr As Feature = cat.SearchForFeature(tbl, si)
Dim si2 As SearchInfo =_
   MapInfo.Data.SearchInfoFactory.SearchWhere("Name = 'Greg'")
Dim ftr2 As Feature = cat.SearchForFeature(tbl, si2)

If TimeOfDay.Now.Second Mod 4 = 0 Then
ftr.Geometry.GetGeometryEditor().OffsetByXY(-5, -25, DistanceUnit.Mile, _
DistanceType.Spherical)
ftr2.Geometry.GetGeometryEditor().OffsetByXY(0, 25, DistanceUnit.Mile, _
DistanceType.Spherical)
Else
ftr.Geometry.GetGeometryEditor().OffsetByXY(-10, 0, DistanceUnit.Mile, _
DistanceType.Spherical)
ftr2.Geometry.GetGeometryEditor().OffsetByXY(10, 5, DistanceUnit.Mile, _
DistanceType.Spherical)
End If
ftr.Geometry.EditingComplete()
ftr2.Geometry.EditingComplete()
ftr.Update()
ftr2.Update()

End If

End Sub
```

Labels

The following section discusses the Labels objects and classes. The diagram below is a UML representation of the Labels hierarchy.



LabelLayer

The LabelLayer class allows the separate ordering of labels and layers. A LabelLayer is a kind of MapLayer and behaves as such. This similarity to MapLayer supports the placement of a LabelLayer anywhere you can put a MapLayer allowing for relative positioning of the Labels compared to the other layers in the map. Each LabelLayer is made up of LayerSources and acts as a collection of those objects.

LabelSource

The LabelSource class graphically displays geographies as labels based on a data source and rules that define how the source is labeled. The LabelSource is added to a LabelLayer on the Map so that it is drawn. The LabelLayer provides positioning within the Map as well as the rules that govern the interaction with other LabelSources. To use LabelSource class, specify a table from where to get the data (MITable), an expression which defines the label text, and other default label properties accordingly.

LabelModifier

The LabelModifier class is used to modify the default properties used to make a label. When the LabelLayer generates labels for each LabelSource in its Sources collection, it first uses the DefaultLabelProperties to generate each label. It then uses each visible LabelModifier in the Modifiers collection.

ILabelSourceFilter

This interface allows you to enumerate through the collection of LabelSource objects in a LabelLayer filtering on specific rules. You can also implement this interface to define your own filtering rules.

LabelProperties

This class has label property information, such as style, positioning, etc. It supports the representation of sparse label properties. This is very useful when using a LabelModifier to modify only a portion of the label properties. This class also allows you to set prioritization and improved placement of labels.

Generating Labels

The LabelLayer class generates labels when the map draws or when you call the LabelLayer.Refresh method. Each visible label source is considered. You can have more than one layer of labels in a map.

To generate a label for each feature in the source's table that is within current map view, the LabelLayer class does the following:

1. Uses the DefaultLabelProperties property as the starting set of properties used to build the label.
2. Calls the Modify method of each visible label modifier in the Modifiers collection, if any. This allows each modifier the opportunity for changing the label properties used to build the label.
3. Performs visibility constraint checks to decide whether to keep the label.
   a. Checks for label visibility by comparing the visibility constraints against the current map zoom/scale.
   b. If the label is visible and overlaps and duplicates are not allowed on this label, checks for overlap and duplicates against other existing labels. If any are found, uses Label Priorities to decide which label to keep.
4. Adds the label to the generated labels collection if it succeeds visibility constraints.

Note that Label generation rules only apply within each label layer and not the entire map. For example, if you set the AllowOverlap property to False in all label sources contained within all the label layers in a map, labels from one label layer will still overlap with labels from another label layer because the labels are generated independently between label layers.

Use the LabelLayer.Refresh method if the map has not yet drawn to generate the labels based on the current map view.

Accessing Generated Labels

You can access generated labels through the LabelSource.Labels property. This collection represents the labels that are within the current map view. They do not represent all the labels in the map. The items in the collection change as the view of the map changes.

Label Priorities

Label priorities determine which labels within the same LabelLayer are generated when either AllowOverlap or AllowDuplicates is set to False.

ⓘ Note: as stated in Generating Labels, each label layer is independent of each other, so different overlap or duplicate settings in other layers do not factor in here.

When a label overlaps with or is a duplicate of another label in the same layer, the priorities of both labels are compared to determine which label to keep. The process first compares the priority of each label (known as the inter-label source). The label with higher priority is kept.

MapXtreme provides two levels of priorities for controlling the display of labels: Major and Minor. This allows you to group and subgroup prioritization. For example, you might want cities with large populations to have labels with a higher priority than small towns. However, you can also add a modifier to bump up the major priority of one small town to give it a chance over the major cities.

When the Major priority is set to null (nothing in Visual Basic), the value used is the inverse of the label's LabelSource position within a LabelLayer. The higher the index position the lower the priority. For example, if a LabelSource is at index 3 and there are 10 LabelSources (indices from 0 to 9), the Major priority will be 6 (the inverse of the index based on total).

When the Minor priority is set to null (Nothing in Visual Basic), the value used is based on the Key of the Feature being labeled. The inverse of a numeric representation of the key against the number of rows in the Table is used. For example, if there are 10 rows in the Table, the minor priority of the label for feature in row ID 7 defaults to 3 (10 - 7 = 3). If the key is not numeric the minor priority defaults to 0.

When Major priorities are equal, the label with the higher Minor priority is kept.

To create a priority, use an expression that results in a numeric value. For example, the expression in C# that results in a numeric value could be a field of numeric type or an expression like the ASC value of the first letter in the field's value:

```
"(1/Asc(Country))*1e6"
```

Label Layer Selectability

To control selectability of label layers, use the SelectMapToolProperties.LabelsAreEditable property programmatically or via Workspace Manager's Labels are Editable checkbox.

The LayerHelper.SetSelectable method does not affect the selectability of certain layer types, including Labels, WMS/WFS, Raster and Group layers.

Code Example: Creating a LabelLayer

The following example demonstrates the use of the classes associated with Labels.

VB example:

```
' Open usa table using the data catalog
Dim table As Table = Session.Current.Catalog.OpenTable("usa.tab")

' Create a new map
Dim map As Map = Session.Current.MapFactory.CreateEmptyMap(New _
   Size(300, 300))

' Create a new feature layer that references the table and add it to the map
Dim featureLayer As New FeatureLayer(table)
map.Layers.Add(featureLayer)

' Create a new label layer and add it to the map.
' Note that if you call MapInfo.Mapping.LayersBase.Add" method instead of
' MapInfo.Mapping.LayersBase.Insert method it will automatically position the
' label layer before the feature layer

Dim labelLayer As New MapInfo.Mapping.LabelLayer()
map.Layers.Insert(0, labelLayer)
```

```
' Create a new label source that references the table
Dim source As New MapInfo.Mapping.LabelSource(table)

' Change its caption expression to be a specific column from the table
' called "State_Name"
source.DefaultLabelProperties.Caption = "State_Name"

' Append the label source to the label layer so that it shows on the map
labelLayer.Sources.Append(source)
```

Curved Labels

Curved labels are labels that follow the curve of a line. They enhance the appearance of map features that are made up of lines, such as streets and rivers.

Curved labels are generated in the Workspace Manager by selecting the label layer, then selecting **Curve labels along segments** in the Position tab. See Curved Labels.

To render curved labels via the API, use the ILayout interface and the UseRelativeOrientation property. For labels to curve along the geometry, specify MapInfo.Text.RelativeOrientation. FollowPath. See a code example in the Developer Reference under ILayout interface.

Adornments

The Adornments class is an unordered collection of a map's adornments. An adornment is either a Legend, a Title, a Scalebar, or some other user-defined object. Each map contains a single Adornments Collection. Each adornment belongs to a single AdornmentControl.

To create your own adornment, derive a class from the IAdornment interface and the AdornmentControl abstract class.

Legends

Legends are for use in conjunction with themes. See Chapter 14 Using Themes and Legends for more information about themes. Legends are created using the LegendFactory class. A Legend consists of one or more legend frames. Each frame is either a ThemeLegendFrame or a CartographicLegendFrame. Both kind of LegendFrames are created by using the LegendFrameFactory class from the created Legend object. For manipulation and customization of legends and their frames, use

classes in the MapInfo.Mapping.Legends namespace. Classes in this namespace include: CartographicLegendFrame, ThemeLegendFrame, LegendFormat, LegendControl, and others.

You can set the size of the Legend with the Legend.Size property, but you cannot set the size of the LegendFrames. The size of the LegendFrame is controlled by the amount of data it contains.

VB example:

```
Private Sub Page_Load(ByVal sender As Object, ByVal e As _
System.EventArgs)
LegendControl1.Map = MapControl1.Map
If Not IsPostBack Then
   Dim normalLyr() As MapInfo.Mapping.LayerType = New _
MapInfo.Mapping.LayerType(1) {}
   normalLyr(0) = MapInfo.Mapping.LayerType.Normal
   Dim filter As MapInfo.Mapping.IMapLayerFilter = _
MapInfo.Mapping.MapLayerFilterFactory.FilterByLayerType(normalLyr)
   Dim frame As MapInfo.Mapping.Legends.LegendFrame
   Dim NewLegend As MapInfo.Mapping.Legends.Legend = _
MapControl1.Map.Legends.CreateLegend(New System.Drawing.Size(5, 5))
NewLegend.Format.FrameAlignment = _
MapInfo.Mapping.Legends.LegendFrameAlignment.Horizontal
   Dim ftrLayer As MapInfo.Mapping.FeatureLayer
For Each ftrLayer In _
MapControl1.Map.Layers.GetMapLayerEnumerator(filter)
frame = _
MapInfo.Mapping.Legends.LegendFrameFactory.CreateCartographic_
LegendFrame(ftrLayer)
   NewLegend.Frames.Append(frame)
   Next
   LegendControl1.Legend = NewLegend
   Else
   LegendControl1.Legend = MapControl1.Map.Legends(0)
   End If
End Sub
```

ScaleBar Adornment

A scale bar is linear object used to measure map distances in the units of the map (such as kilometers or feet). Use the ScaleBarAdornment class to create a ScaleBarAdornmentControl and add the control to the map. If you attempt to add a ScaleBarAdornment to the map itself, it does display, but without the ScaleBarAdornmentControl, the ScaleBar is not linked to the map itself.

VB example:

```
Public Shared Sub MapInfo_Mapping_ScaleBarAdornment(ByVal mapControl1 As _
```

```
MapControl)
   ' Create a scalebar
   Dim sba As ScaleBarAdornment = New ScaleBarAdornment(mapControl1.Map)

   ' Position the scalebar at the lower right corner of map
   Dim x As Integer = mapControl1.Map.Size.Width - sba.Size.Width
   Dim y As Integer = mapControl1.Map.Size.Height - sba.Size.Height
   sba.Location = New System.Drawing.Point(x, y)

   ' Add the control to the map
   Dim sbac As ScaleBarAdornmentControl = New ScaleBarAdornmentControl(sba, _
mapControl1.Map)
   mapControl1.AddAdornment(sba, sbac)
End Sub
```

Title Adornment

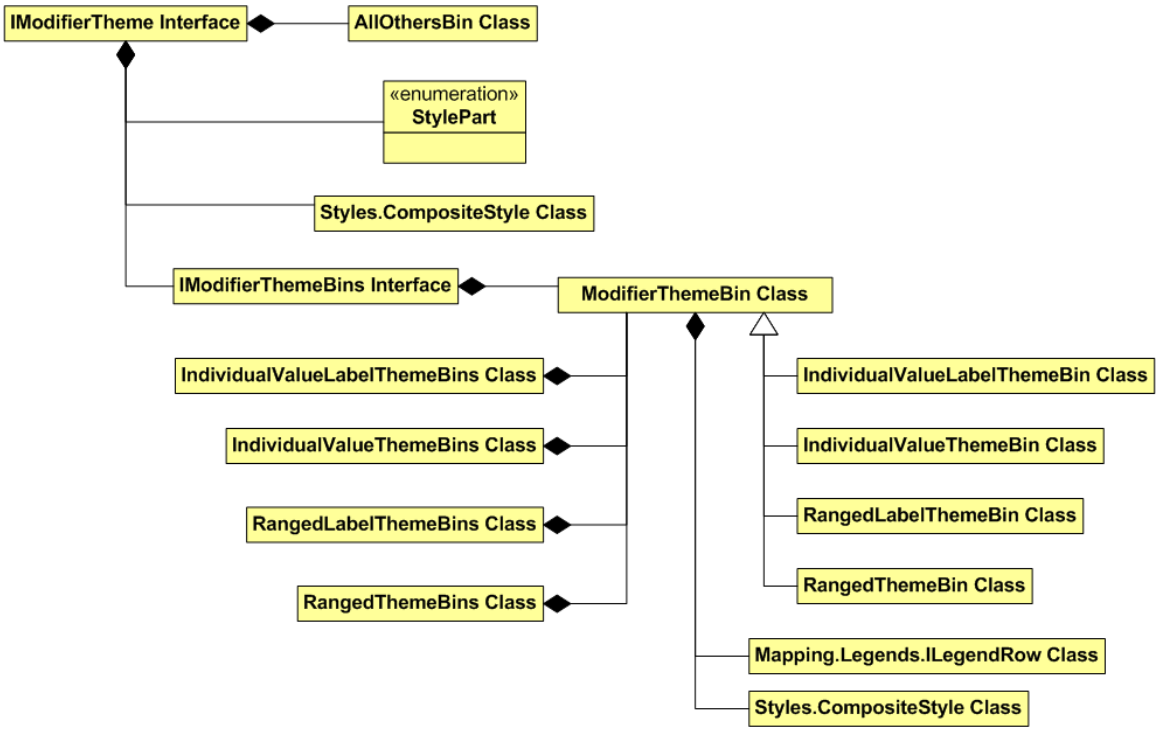
A Title adornment is a text object drawn on the map to represent a map title or to provide a text to clarify other information on the map. In MapXtreme you use the TitleAdornment class to create a title and add it to the map.

VB example:

```
Public Shared Sub MapInfo_Mapping_TitleAdornment(ByVal mapControl1 As
MapControl)
    ' Create a Titlebar
    Dim ta As New MapInfo.Mapping.TitleAdornment(New System.Drawing.Size(100,
50), mapControl1.Map)
    ta.Title = "This is a watermark"
    mapControl1.Map.Adornments.Append(ta)
End Sub
```

Feature Style Modifiers

Feature style modifiers allow a way of changing/modifying the style of a Feature before it is drawn. These classes use the sparse attribute aspect of CompositeStyle object to only modify the part of the style in which you are interested. Range, Individual Value, and Dot Density themes are style modifiers. Thematics are covered in detail in Chapter 14 Using Themes and Legends. The diagram below shows a UML representation of the Modifier and thematics hierarchy.



FeatureStyleModifier

This is an abstract base class from which all modifiers must derive. The IndividualValue, Ranged, and DotDensity themes are all FeatureStyleModifier objects. You can create your own class that derives from FeatureStyleModifier and override the Modify() method.

FeatureStyleModifiers

The FeatureStyleModifiers class is an ordered collection of FeatureStyleModifier objects contained within each FeatureLayer. Each modifier in the collection is called in order before a feature's geometry is drawn.

FeatureOverrideStyleModifier

The FeatureOverrideStyleModifier is a simple kind of FeatureStyleModifier. This class provides style override functionality at the Layer level. The FeatureOverrideStyleModifier has a composite style and implements the IVisibilityConstraint. This is similar to functionality found in MapX and MapInfo Professional.

Printing Your Map

Once your application is created you may want to give your user the opportunity to print out their generated map. The MapInfo.Printing namespace is provided to facilitate including printing functionality into any application. This classes in this namespace are built upon the .NET Framework classes for printing so the necessary constructions are similar to any other Windows application. In addition to regular printing we have provided many additional capabilities to optimize the printed version of a created map.

Refer to Appendix E: Printing From MapXtreme Applications for more detailed information about printing from a MapXtreme application.

13

13 – Finding Locations

The MapInfo.Data.Find namespace contains the classes to use when searching for map features by address, street intersection or name.

In this chapter:

Functional Overview of Find

Find is used when you wish to locate map features by address, street intersection or name. To find features at locations using selection tools or queries, use the Search classes in the Data namespace (see Searching for Features).

A successful Find operation can result in an exact match, one or more close matches, or no match (failed match). The operation is controllable by a variety of properties and fallback options and are discussed in the Overview of the Data.Find Namespace. The section below describes how MapXtreme makes a match. The more you understand the Find process, the more you can control the success of the operation.

The Find Process

MapXtreme locates map features by matching an address, street intersection or place name with information in the feature table. For example, you can find 1600 Pennsylvania Avenue in Washington, D.C. provided a table of Washington D.C. streets is open.

Finding a map feature is similar. For example, you can find The White House if you have a mappable table of landmarks that contains the name “The White House” and its geo-referenced (mappable) location. You don’t need to supply the address to find by place name.

To find a street intersection you must provide both street names that make up the intersection.

MapXtreme attempts to find an exact match in which the result is a character by character match to the input address, place name or intersection. If it cannot make an exact match, MapXtreme attempts to find close matches based on its matching rules and your settings. If it cannot find a close match, it returns a failed match. Note that matching is not case-sensitive. Upper and lower case characters are successfully matched with one another.

A street address is normally made up of a street number, name and abbreviations that represent the street prefix, such as North, and suffix, such as Road. Addresses can take a variety of forms and may include additional information such as apartment number or route. In addition, some of the key components like street type may be missing from the input address. MapXtreme looks at each component of the address and applies specific rules to it to find a match.

The following sections describe how MapXtreme handles specific information and conditions including street name, street abbreviations, address numbers, refining tables and results.

Matching to Street Name

A match to a street name is a straightforward character by character evaluation of the address against the information in the search table. For example, MapXtreme returns an exact match for the street name LaSalle if LaSalle is in the table, but may only return a close match if the address is spelled La Salle or LaSal.

Matching to Street Abbreviations

Street abbreviations vary widely in address records. Sometimes the component is missing altogether. In many cases, however, MapXtreme can make an exact match even if there are slight differences between the address and the search table. MapXtreme refers to a standardized address abbreviation and substitution list, called Mapinfow.abb, to find a suitable match. This list contains standard spellings of street prefix and suffix abbreviations, such as ST for Street and Av for Avenue. You must set a property to tell MapXtreme to use this abbreviations file, but it is a good way to increase the chance of exact matches or find more close matches than without the file.

The table below illustrates variations in address and whether or not it will result in an exact match based on the use of the abbreviations file. The first column contains a street name you want to find. The second column contains the corresponding street name from a source table. The third column says why they do not match. The fourth column indicates whether the particular problem is one that can be corrected by using the abbreviation equivalence file. This table assumes that the addresses are the contents of a single column in a table. While the street number would often be in the same column, we don't indicate street numbers here because they are handled differently than street names.

Address to Find	Search Table Address	Comment	Correctable with Abbreviation File for an Exact Match
LaSalle Street	LaSalle St	"Street" and "St" do not match.	Yes
LaSalle Ave	LaSalle Av	"Ave" and "Av" do not match.	Yes
LaSalle Ave	LaSalle St	"Ave" and "St" do not match.	No

Address to Find	Search Table Address	Comment	Correctable with Abbreviation File for an Exact Match
LaSalle	LaSalle St	“St” is missing from target.	No. If the street abbreviation is missing, MapXtreme does make a guess on what it could be.
LaSalle St North	LaSalle St	“North” is not in the search table.	No
LaSalle St North	LaSalle St N	“North” and N do not match.	Yes
LaSalle St Apt 3	LaSalle St	Apartment number does not match anything in the source.	Yes. The apartment number is ignored.
Tenth St	10th St	“Tenth” and “10th” do not match.	Yes
10th Av	Tenth Av	“10th” and “Tenth” do not match.	Yes
Saint John’s Lane	St John’s Lane	“Saint” and “St” do not match.	Yes

If you find that you have repeated situations that do not match due to abbreviations, you can:

- Edit your addresses to conform closer to the abbreviations file, or
- Edit the abbreviations file with a text editor to add your specific abbreviations. The mapinfow.abb is located in C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x. For more information about editing the abbreviations file, see Fine Tuning the Find Process.

Matching Address Numbers

MapXtreme can match when the address number precedes the street name (as in North American addresses) or when it follows the street name (common in European addresses). By default MapXtreme assumes the address number precedes the street name. You must set a property when the address number follows the street name.

MapXtreme compares the input address number to a range of address numbers .Typically the table contains address number ranges that correspond to the portion of the street the range covers. Address ranges can be matched to the exact side of the street, since often address ranges are odd-numbered on one side of the street and even-numbered on the other.

In cases where MapXtreme cannot match the address range exactly, you may be willing to settle for a close match in which the closest range would be considered a match. You would use the stricter exact match requirement only for very precise finds. Often that level of precision isn't necessary. A close match is usually acceptable.

Matching with a Refining Boundary Table

MapXtreme can also find an address in one table in which there may be more than one possible match. To avoid finding the wrong address, you specify a refining table and column to focus the match to a smaller area.

This is useful, for example, when searching a table of streets that cover an entire county and you are looking for Main St. It's likely that more than one town in the county will have a Main St. By providing a refining table of town boundaries, you can specify that you are looking for the Main St. only in Town A.

You can use any type of refining boundary table you wish, for example, postal code boundaries or census regions. Additionally, you can specific a second refining boundary in which to conduct a Find.

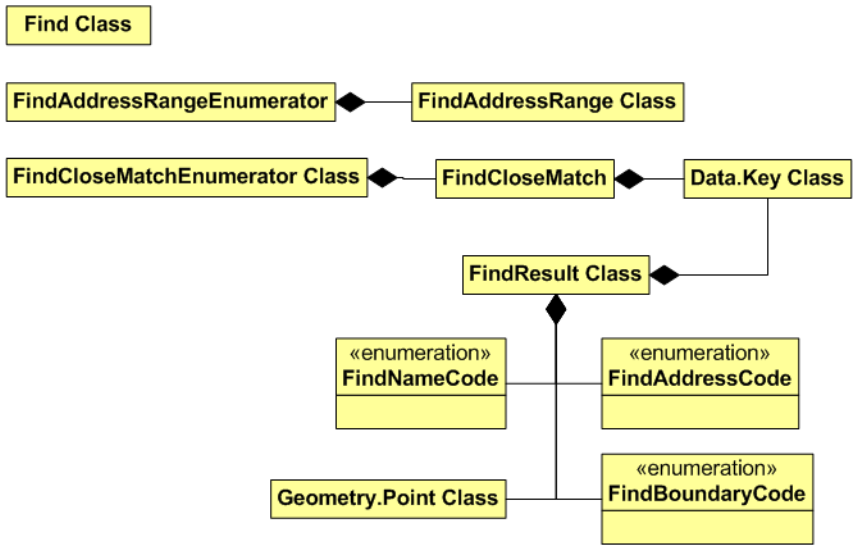
Find Results

MapXtreme returns either an exact match, one or more close matches, or a failed match. The results depend on a number of factors including the quality of the input data and the conditions set for the operation. Information is also returned that explains how well each portion of the address matched (or failed to match).

Overview of the Data.Find Namespace

The MapInfo.Data.Find namespace contains classes that enable you to locate a map feature, street address, or street intersection within a mappable table. The properties and methods of the Find class are used to set up a Find. The results of a Find are returned using the FindAddressRange, FindCloseMatch, and FindResult classes.

The following UML diagram illustrates the Find namespace.



Find

The Find object is used to locate a map object, street address, or street intersection within a given mappable table. Find searches the table for a match and returns the result(s) in a FindResult object.

To use Find in MapXtreme, you must have:

- A mappable table (a table that contains geometry objects)
- An *indexed* column on which to perform the search
- An item to search for, such as a place name, street address or street intersection
- Optionally, a refining table in which to narrow the search to a more specific location.

The Find class provides a number of properties to control the search operation. For example, you can limit the number of close matches to be returned (CloseMatchesMax) or indicate that you wish to use the abbreviation file for increasing the likelihood of a match (UseAbbreviations).

Property	Description
AddressNumberAfterStreet	Specifies whether the address number is located after the street name (for example, “Smith Street 107”).
ChooseAlternateBoundary	Specifies whether to match a record found in a refining region other than the refining region specified.
ChooseClosestAddressRange	Specifies whether to use the closest available address number in cases where the address number does not match.
ChooseClosestObject	Specifies whether or not to find the closest object match when an exact match is not found.
CloseMatchesMax	Specifies how many close matches to return if an exact match is not found.
InsetDistance	A positive value representing how far from the ends of the line to adjust the placement of an address location.
InsetPercentage	Represents the percentage of the length of the line where the address is to be placed.
InsetUnit	Represents the distance unit to use for Inset.
OffsetDistance	Representing the distance to offset the placement of an address location back from the street.
OffsetUnit	Represents the distance unit to use for Offset.
UseAbbreviations	Specifies whether substituting abbreviations from the abbreviations file are used to find a match (for example, “Smith Street” is substituted with “Smith St”).
UseCloseMatches	Specifies whether to return “N” number of close matches if an exact match is not found.
UseInsetAsPercentage	Specifies whether Inset is being used as a percentage or as a distance.

The Find class provides four search methods: two for searching addresses or features with or without a refining boundary, and two for searching street intersections with or without a refining boundary.

Method	Description
Search	Searches a mappable table for a named location and returns a FindResult object.
SearchIntersection	Searches a mappable street table for a given intersection, returning a FindResult object.
Dispose	Releases unmanaged resources held by the Find object. It is necessary to call this method when finished with the Find object.

.FindAddressRange

The FindAddressRange object represents an address range item returned from the Find.Search method. The FindAddressRange object is returned as part of the FindResult object when a street address is not found, the address number is not within the minimum/maximum address ranges for a given street, or an address number was not specified.

Code Sample

```
public void GetAddressRangesOnStreetTable()
     {
        Table_table;
        _table = Session.Current.Catalog.OpenTable("North_Greenbush.tab");

        Find _find = new Find(_table,_table.TableInfo.Columns[1]);
        FindResult _findResult= _find.Search("Meadow Dr");
        If ((!_findResult.ExactMatch) && (_findResult.NameResultCode ==
           FindNameCode.ExactMatch)&& (findResult.AddressResultCode ==
           FindAddressCode.AddressNumNotSpecified))
{
        FindAddressRangeEnumerator _enum =
             _findResult.GetAddressRangeEnumerator();
        FindAddressRange _findAddressRange;
        int _iIndex = 0;

        while (_enum.MoveNext())
        {
           _findAddressRange = _enum.Current;
           Console.WriteLine("_findAddressRange.BeginRange");
```

```
            Console.WriteLine("_findAddressRange.EndRange");


            _iIndex++;
         }

         if(_table != null)
         {
            _table.Close();
            _table = null;
         }
      }
   }
find.Dispose();
}
```

FindCloseMatch

The FindCloseMatch object represents a close match item returned from the Find Search method. The object is returned as part of the FindResult object. A close match item is a returned match that closely matches the name of the requested search.

To use this feature, you must first set the UseCloseMatches and CloseMatchesMax properties of the Find object before you execute your search. For example if you tried to search for “Washington Street,” and UseCloseMatches was set to true, a close match would be “Washington Ave.”

Code Sample

```
public void CloseMatchesOnStreetTable()
{
   Table _table;
   _table = Session.Current.Catalog.OpenTable("Rensselaer.tab");

   Find _find = new Find(_table,_table.TableInfo.Columns[1]);
   _find.UseCloseMatches = true;
   _find.CloseMatchesMax = 5;

   FindResult _findResult= _find.Search("70 Washington");

   if ((!_findResult.ExactMatch) && (_findResult.NameResultCode ==
            FindNameCode.ExactMatchNotFound))
   {
      FindCloseMatchEnumerator _enum =
            _findResult.GetCloseMatchEnumerator();
      FindCloseMatch _findCloseMatch;
      int _iIndex = 0;

      while (_enum.MoveNext())
      {
```

```
            _findCloseMatch = _enum.Current;

            Console.WriteLine(_findCloseMatch.Name);
            _iIndex++;
         }
      }

      if(_table != null)
      {
         _table.Close();
         _table = null;
      }
find.Dispose();
}
```

FindResult

The FindResult class returns information from the Find.Search method in the form of properties that explain what kind of match was made, as shown in the table below. If successful, the FoundKey property contains the key of the object to be located. If successful, the FoundPoint property contains the point of the object located.

Property	Description
AddressOutOfRange	Specifies whether the address passed in was out of range.
AddressResultCode	Identifies the result code for the address part of the search and returns a FindAddressCode enumeration.
BoundaryResultCode	Identifies the result code for the refining boundary part of the search and returns a FindBoundaryCode enumeration.
ExactMatch	Specifies whether an exact match was found.
FoundKey	Specifies the Key of the object that has been located.
FoundPoint	Specifies the Point of the object that has been located.
IntersectionNotFound	Specifies whether the intersection was not found.
MultipleMatches	Specifies whether multiple matches were found.

Property	Description
NameResultCode	Identifies the result code for the name being searched on and returns a FindNameCode enumeration.
ResultCode	If the Find results is an exact match, the value is one. If the Find results in an approximate match, the value is greater than one. If the Find fails to match the address, the result is a negative value.
UseSubstitution	Specifies whether a substitution was used from the abbreviation file.

FindAddressCode Enumeration

Identifies the result code for the address part of the search and is returned by the FindResult.AddressResultCode property.

ⓘ This result code should only be used when searching for a street or intersection.

Member Name	Description
ExactMatch	An exact match was found.
SideOfStreetUndetermined	The side of the street was undetermined.
WithinMinMax	The address number was within minimum/maximum range.
NotWithinMinMax	The address number was not within minimum/maximum range.
AddressNumNotSpecified	An address number was not specified.
StreetsDoNotIntersect	The streets do not intersect.
NoMapObjectForRowMatched	The row matched does not have a map object.

FindBoundaryCode Enumeration

Identifies the result code for the refining boundary part of the search and is returned by the FindResult.BoundaryResultCode property. A refining boundary is used to distinguish between multiple features with the same name.

ⓘ This result code should only be used when a region is being used to refine the search.

Member Name	Description
ExactMatch	An exact match was found.
FoundInOneOtherRegion	The name was found in only one region other than specified region.
FoundInMoreThanOneRegion	The name was found in more than one region other than the specified region.
NoRegionSpecifiedOneMatch	No refining region was specified, and one match was found.
NoRegionSpecifiedMultipleMatches	No region was specified, and multiple matches were found.
MultipleMatchesFound	The name was found more than once in the specified region.

FindNameCode Enumeration

Identifies the result code for the name being searched on and is returned by the FindResult.NameResultCode property.

Member Name	Description
ExactMatch	An exact match was found.
SubstitutionUsed	A substitution from the abbreviations file used.
ExactMatchNotFound	An exact match was not found.

Member Name	Description
NoObjectNameSpecified	No object name specified; match not found.
CloseMatch	A close match was found.

Fine Tuning the Find Process

As we stated in the beginning of this chapter, the more you understand about how Find works, the better you can use its properties and your input information to improve the chances for a successful match. This section provides some help with developing strategies for improved results during a Find.

Editing the Mapinfow.abb File

The Mapinfow.abb file is an abbreviations and substitution file that ships with MapXtreme. This file can be used to improve the chances for a Find if the abbreviation in the input address can be found in the abbreviations file. This is covered in the section Matching to Street Abbreviations. This section covers additional types of information you can include in the file.

You can edit Mapinfow.abb in any text editor or word processor. Open the file and make your additions, adding keywords as necessary. Here is a standard version of the file:

```
!version 3.0
FIRST      1ST
SECOND     2ND
THIRD      3RD
FOURTH     4TH
FIFTH      5TH
SIXTH      6TH
SEVENTH    7TH
EIGHTH     8TH
NINTH      9TH
TENTH      10TH
NORTH      N
SOUTH      S
EAST       E
WEST       W
ALLEY      AL
AVENUE     AV
AVE        AV
BOULEVARD  BLVD
BRIDGE     BR
CIRCLE     CIR
```

```
COURT      CT
DRIVE      DR
EXTENSION  EXT
HIGHWAY    HWY
INTERSTATE I
LANE       LN
MOUNT      MT
PARK       PK
PARKWAY    PKWY
PLACE      PL
PLAZA      PLZ
POINT      PT
RAILROAD   RR
ROAD       RD
ROUTE      RT
SAINT      ST
SQUARE     SQ
STREET     ST
STR        ST
TERRACE    TER
!EOLNOSPACE
,
;
#
!EOLSPACE
FLOOR
SUITE
"P.O. BOX"
!NOSPACE
.
\"
\!
\\
!SPACE
"STATE HIGHWAY"STHWY"
"N ST"NORTH ST"
"S ST"SOUTH ST"
"E ST"EAST ST"
"W ST"WEST ST"
"N AV"NORTH AV"
"S AV"SOUTH AV"
"E AV"EAST AV"
"W AV"WEST AV"
```

You can make additions to this file to take care of various problems. Most importantly, you can make several different kinds of additions. MapXtreme recognizes four classes of substitution items and it interprets these classes differently. Each class is preceded by the keyword used to identify it in the abbreviation file.

Space-delimited simple substitution	!SPACE
Simple truncation	!EOLNOSPACE
Space-delimited truncation	!EOLSPACE
Simple substitution	!NOSPACE

In order for MapXtreme to know how to interpret a line, or set of lines, in the abbreviation file, you have to precede the line with the keyword which indicates the appropriate interpretation strategy.

When all of the entries in the abbreviation file use the default interpretation, there is no need to precede any of them with a keyword. When there is no keyword at the beginning of the abbreviation file, MapXtreme will treat the initial entries as requiring the default interpretation. Once you add other types of substitution pairs, however, you have to start adding keywords.

Space-delimited Substitution

Space-delimited simple substitution is the default. What that means is this: MapXtreme compares spaced-delimited tokens in target addresses with the rows in the address file. A space-delimited token is a string of characters with a space before and a space after. For example, MapXtreme will match “Ave” with “Av” in “Park Ave” but it will not match “Avery Blvd” to “Avry Blvd.” Both street names contain the string “Ave”. But that string is bordered by spaces only in “Park Ave”, not in “Avery Blvd.” In “Avery Blvd”, “Ave” is followed by “r”, not by a space.

All of the entries in the abbreviation file will receive this default interpretation. You can add other items to receive the same treatment. For example, you might want to add the pair “WK WALK” so that MapXtreme knows to interpret “WK” in a target address as though it were “WALK”. Similarly, you might want to add a pair such as: “WAY WY”.

Use the keyword “!SPACE” to indicate space-delimited simple substitution. Entries following “!SPACE” are given the default interpretation (this allows you to arrange the Abb.file contents in some other order). When MapXtreme encounters another keyword, it switches to the indicated interpretation strategy.

Simple Truncation

In simple truncation, MapXtreme finds an item in the address and simply ignores it and everything after it. These items do not have to be space-delimited. This strategy is useful for dealing with addresses such as:

123 Appian Way, Mail Stop 829

7305 Van Zandt # 23

In the first case, you want MapXtreme to ignore the comma and everything after it. In the second case you want MapXtreme to ignore the number sign and everything after it. To deal with such cases add the following to your abbreviation file:

```
!EOLNOSPACE
,
#
```

“!EOLNOSPACE” is the keyword indicating that the following items are to be treated as cases of simple truncation. After that we have one line with a comma and one with a number sign. Whenever MapXtreme encounters a comma or a number sign in an address it will ignore it and everything after. The examples become:

123 Appian Way

7305 Van Zandt

Space-Delimited Truncation

In space-delimited truncation MapXtreme looks for items which are space delimited and eliminates those items and everything following. For example:

73 Appian Way Suite 829

3033 Van Zandt Room 202

To deal with such cases add the following to your abbreviation file:

```
!EOLSPACE
SUITE
ROOM"
```

!EOLSPACE” is the keyword indicating that the following items are to be treated as cases of simple truncation. After that we have one line with “Suite” and one with a “ROOM”. Whenever MapXtreme encounters those tokens it will truncate the address. The examples become:

73 Appian Way

3033 Van Zandt

Simple Substitution

MapXtreme uses simple substitution to remove items from an address and otherwise does nothing. Use it to deal with:

433 Van-Rensselaer

91 St Albans’

The goal is to strip out the hyphen and the apostrophe. Make the following entries to the abbreviation file:

```
!NOSPACE
-
'
```

“NOSPACE” is the keyword calling for simple substitution, and the hyphen and apostrophe on the following lines are the tokens to be removed. The examples become:

369 VanRensselaer

91 St Albans

Legitimate Spaces

There are cases where you want to indicate a substitution in which the searched for string contains spaces. You can use double quotes in such cases. Place a double quote:

- at the beginning of the line; and
- between the searched for string and the substitution; and
- at the end of the line.

For example, you might want to substitute “STHWY” for “State Highway”. To do that, use the following line:

```
"State Highway"STHWY"
```

This provides a solution to a subtle problem, that of street names which match items in the abbreviation file. For example, “North St” and “Park Av” both have initial strings which match terms in the abbreviation file. Consequently, MapXtreme will substitute “N” for “North” to yield “N St” and “Pk” for “Park” to yield “Pk Av.” You could add the following lines to the Abbreviation file to rectify these substitutions:

```
"N ST"North ST"
"PK AV"PARK AV"
```

Note that these lines have to come after the entries which substitute “N” for “North” and “PK” for “Park”. If they came before, they would have no effect. Thus:

```
...
```

```
...
NORTH    N
...
...
PARK     PK
..
....
"N ST"NORTH ST"
"PK AV"PARK AV"
...
...
```

When MapXtreme encounters NORTH N it will turn NORTH ST into N ST. When it encounters “N ST”NORTH ST” it will then turn N ST into NORTH ST. PARK AV is treated similarly.

Special Characters

MapXtreme uses the exclamation point (!), the double quote (") and the backslash (\) as special characters. These characters tell MapXtreme how to treat strings which follow them, but are not themselves ordinarily treated as characters in substitution strings. The exclamation point tells MapXtreme that the string should not be interpreted as an abbreviation. The double quote tells MapXtreme that spaces in the string are legitimate. And the backslash tells MapXtreme to treat a special character as an ordinary character.

When you want to use any of these in a line where they are to be treated as simple characters, precede them by a backslash. Thus:

```
\!
\"
\\
```

Adding Lines to the Abbreviation File

You can add a new item to the file by adding a new row. The order in which you add rows is not significant, except in those cases where you expect one substitution pair to compensate for the effects of another. The number of spaces between the first and second items in a row is not significant either.

Incorrect Address Ranges

When an address contains a number range that is not in the source table, MapXtreme will not be able to match it. Such an address might fall into a gap in range numbers or it might be beyond the ends of the ranges. To handle this problem:

1. Set the Find.ChooseClosestAddressRange property to true and then perform your search.

2. Resolve the failed matches by reviewing the FindAddressCode enumeration which is returned by the FindResult.AddressOutOfRange property.

It is possible that the address is for a street segment that was added after your source table was made. In that case, edit the source table so that it reflects the full range of addresses for that street.

Inaccurate Town Names

MapXtreme's last step in a Find operation is to determine in which region to place a matched street address. MapXtreme only takes this step if you have so specified when you set up the find process. It is common to use town or city name as the refining boundary. This causes problems because people often do not use the town name which the Census Bureau assigns to their address. Since almost all electronic maps of the United States are based on Census Bureau maps, this will cause problems.

For example, the address "50 Wolf Rd., Albany, NY" is actually in the town of Colonie. Thus the address town name in the target address will not match the appropriate town in the source file.

One way to handle this is use the ChooseAlternateBoundary property. When this option is enabled, MapXtreme attempts to match an address to whatever boundary that address is in, providing that address is in only one boundary. When the address is in more than one boundary, the Find will fail.

Another way to deal with this problem is to use the ZIP Code as the refining boundary, rather than the town or city name.

14

14 – Using Themes and Legends

MapXtreme provides you many options for adding thematics and legends to your map. The following sections illustrate the different types of themes and legends available to you and explains how to use them.

In this chapter:

Thematics Overview

Thematic mapping allows you to present trends in data that would be difficult to see from tabular data. The theme is usually some piece or pieces of your data. You thematically shade a map using data from a data source, such as a native MapInfo table. For example, you can thematically shade a map of the United States based on the average temperature of each state. When you see red, you know it is hot (high number of degrees); where you see blue, it is cold (low number of degrees).

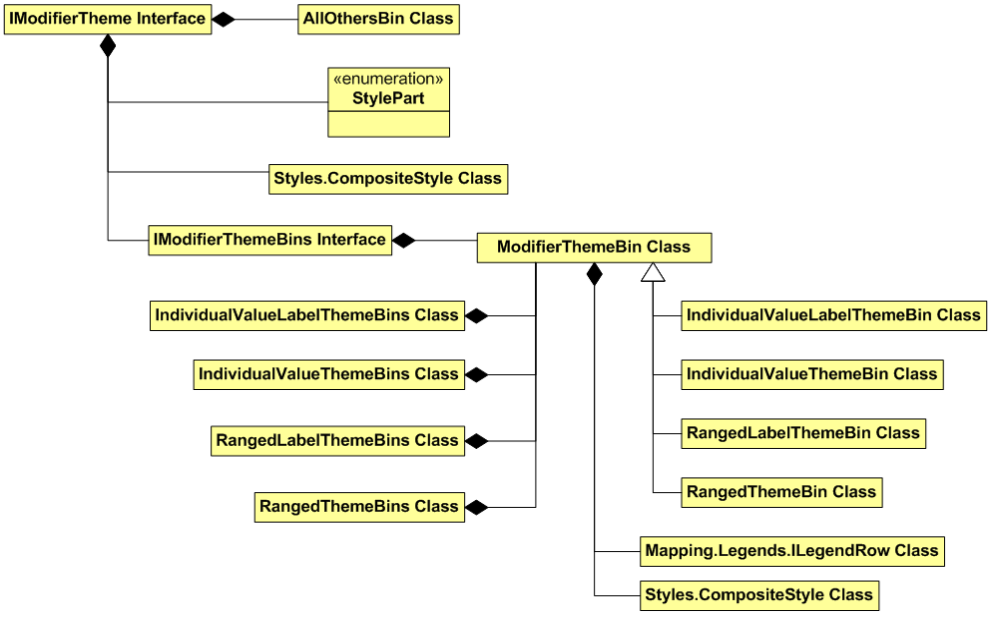
Themes represent your data with shades of color, fill patterns, or symbols. There are many uses for thematic maps to display your data. You create different thematic maps by assigning these colors, patterns, or symbols to map objects according to specific values in your data.

Mapping.Thematics Namespace

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as style overrides on Feature layers and as Object themes. Modifier themes change the style, while object themes add a new layer. All themes implement the ITheme interface.

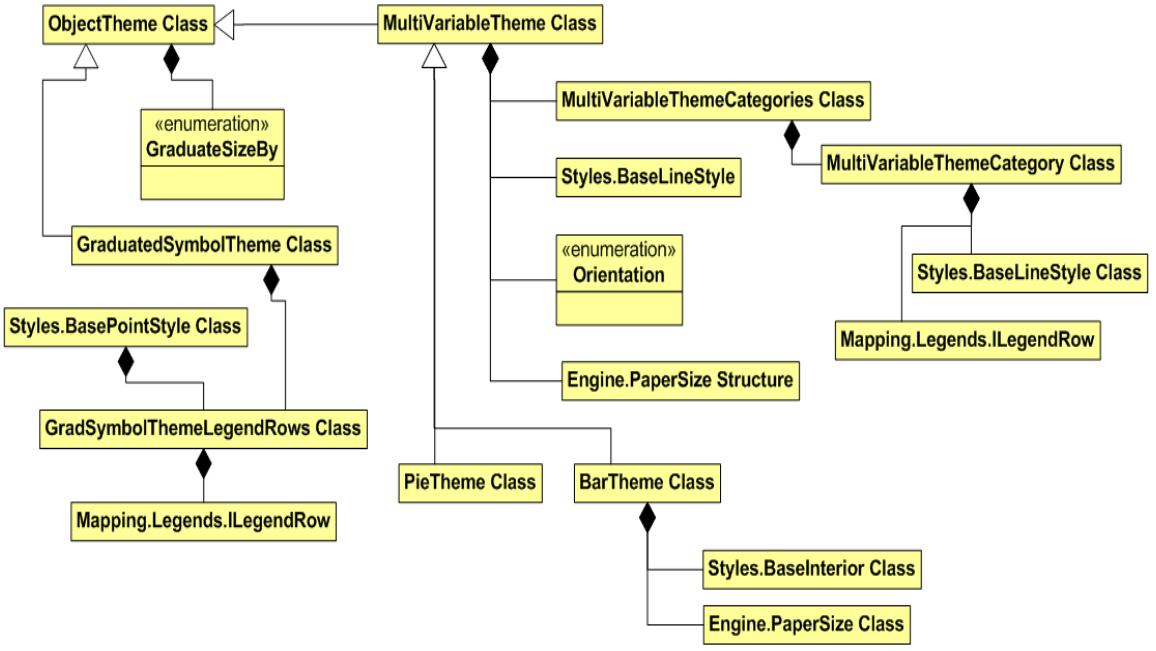
Modifier Themes

An example of a feature modifier theme are ranged, individual value and dot density thematic maps. They modify existing features in a layer. The following UML diagram gives an overview of the modifier theme hierarchy.



Object Themes

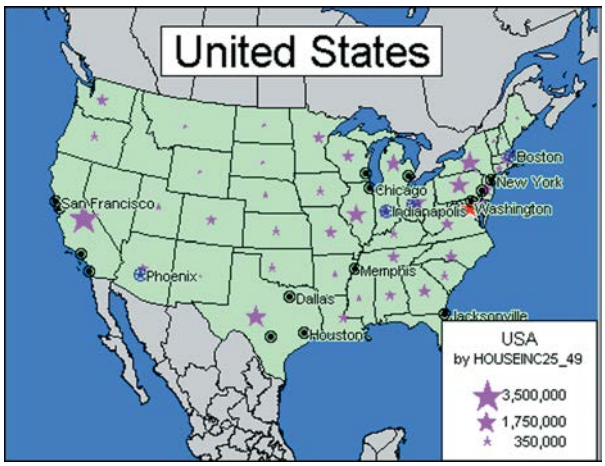
Object themes include graduated symbol, and pie and bar charts. These themes create objects that represent the data values. The following UML diagram gives an overview of the object theme hierarchy.



GraduatedSymbolTheme

A graduated symbol theme is an object theme that contains point features whose symbol sizes are based on the numeric values of the theme's expression.

For instance, use graduated symbols to show housing income for a particular segment of the population across an area.



Example of a Graduated Symbol thematic map.

When To Use a Graduated Symbol Theme

Graduated symbols maps work with numeric data only. It makes no sense to create graduated symbols based on the type of cuisine each restaurant serves. However, graduated symbols are appropriate when you want to show the distribution of housing income by city.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_GraduatedSymbolTheme(ByVal map As
Map)

    ‘ Load a map based on one table
    map.Load(New MapTableLoader(“world.tab”))
    Dim lyr As FeatureLayer = CType(map.Layers(“world”), FeatureLayer)

    ‘ Create a new graduated symbol theme
    Dim gradTheme As GraduatedSymbolTheme = New _
        GraduatedSymbolTheme(lyr.Table, “Pop_Native”)

    ‘ Create an object theme layer based on that graduated symbol theme
    Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer(“World Pop _
        Growth Rate”, Nothing, gradTheme)
```

```
    'Add object theme to the map's layer collection.
    map.Layers.Add(thmLayer)

    ' Adjust how we graduate the size.
    gradTheme.GraduateSizeBy = GraduateSizeBy.Constant
    thmLayer.RebuildTheme()
End Sub
```

PieTheme

A pie theme is an object theme containing pie charts with wedges that represent each data value. In pie charts you compare the wedges in a single pie, or examine a particular pie wedge across all of the pies. Pie charts also enable you to compare parts of a whole.

When To Use a Pie Theme

Pie charts are particularly useful for analyzing demographic data. For example, you have a dataset of demographic information for the United States. Your dataset shows the populations of several major demographic groups. Using pie charts, you can show the population of each demographic group, and see what fraction of the pie it makes up in each chart. This enables you to see the distribution of demographic groups on a per state basis, or across the entire United States. You can also look at one demographic group and see how the relative population of the group varies in different states.

VB example:

```
Public Shared Sub MapInfoMappingThematicsPieTheme(ByVal map As Map)
' Load a map based on one table
map.Load(New MapTableLoader("mexico.tab"))
Dim lyr As FeatureLayer = CType(map.Layers("mexico"), FeatureLayer)

' Create a new pie theme
Dim pieTheme As MapInfo.Mapping.Thematics.PieTheme = New _
     MapInfo.Mapping.Thematics.PieTheme(map, lyr.Table, "Cars_91", _
     "Buses_91", "Trucks_91")

' Create an object theme layer based on that pie theme
Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer("Count by _
     Vehicle Type", Nothing, pieTheme)

'Add object theme to the map's layer collection.
map.Layers.Add(thmLayer)

' DataValueAtSize is calculated automatically if not specified in the
' pie's constructor. But, you can adjust it.  If you do so here, you
' have to rebuild the theme. You can adjust it before creating the
```

```
' object theme layer, and that way the pies won't need to be built
' twice.

pieTheme.DataValueAtSize /= 2
pieTheme.GraduateSizeBy = GraduateSizeBy.Constant
thmLayer.RebuildTheme()
End Sub
```

Printing a Map Containing Pie/Bar Themes

When cloning a map for printing that contains pie or bar themes, be sure to take the size of the paper into consideration to get expected results. The paper size is used to calculate the size of the pie/bars in the printed output. Use the paper size at 100 percent so that the graphs/themes are the same relative size in the print preview as they are on the MapControl.

BarTheme

A bar theme is an object theme that contains bar charts with bars that represent each data value. A bar chart is built for every map object (feature) at the centroid of the object, enabling you to analyze several thematic variables in a particular chart by comparing the height of the bars.

When To Use a Bar Theme

Bar themes are useful for examining the same variable across all the features in your map. For example, you have a table of U.S. state boundaries containing female and male population. Using bar charts, you can create a thematic map that displays a two–bar chart for each state: one bar representing female, and the other representing male population. You can compare the population differences of each state, or you can examine several states and compare population differences to the others.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_BarTheme(ByVal map As Map)
    ‘ Load a map based on one table.
    map.Load(New MapTableLoader(“world.tab”))
    Dim lyr As FeatureLayer = CType(map.Layers(“world”), FeatureLayer)

    ‘ Create a new bar theme.
    Dim barTheme As MapInfo.Mapping.Thematics.BarTheme = New _
     MapInfo.Mapping.Thematics.BarTheme(map, lyr.Table, “Pop_Native”,_
     “Pop_Asian”, “Pop_Other”)
```

```
    ‘ Create an object theme layer based on that bar theme.
    Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer(“World _
      Pop”, Nothing, barTheme)

    ‘ Add object theme to the map’s layer collection.
    map.Layers.Add(thmLayer)

    ‘ Stack the bars and graduate by a constant amount.
    barTheme.Stacked = True
    barTheme.GraduateSizeBy = GraduateSizeBy.Constant
    thmLayer.RebuildTheme()
```

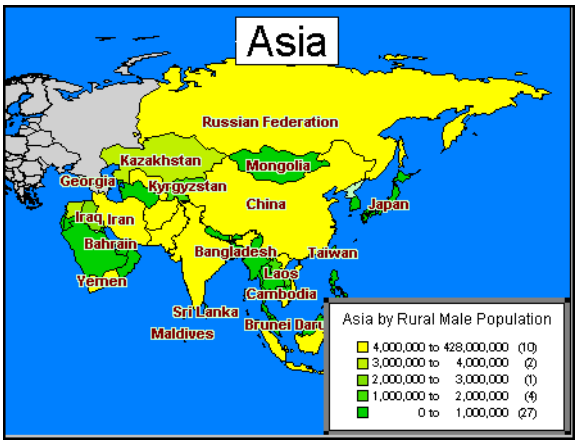
End Sub

Controlling Display Size for Pie and Bar Themes

MapXtreme provides the abstract base class MultiVariableTheme that supports the creation of pie and bar object themes. This class, derived from the ObjectTheme class, provides two properties, DataValueAtSize and Size, that control how large object theme geometries display at particular values. The default value for DataValueAtSize is set to the largest data value of the mapped features. The Size value controls the width of pie charts and the height of bar charts in paper units.

RangedTheme

A ranged theme shows data grouped into ranges (bins) according to specific criteria. In MapXtreme a ranged theme modifies an existing layer to reflect that criteria. It does not create a new layer, as range themes did in previous versions of MapX and MapXtreme. When you create a ranged thematic map, MapXtreme groups all dataset rows into ranges and assigns each row’s object the color, symbol, or line for its corresponding range.



Example of a ranged thematic map.

When To Use a Ranged Theme

A ranged theme is useful, for example, when you have demographic data for an area. For example, rural male population in Asia can be grouped into bins and shaded by color to indicate the population ranges that exist across the area.

All records in the dataset are assigned to a range and then drawn with a style based on that range. When using a yellow to green color range, for example, the countries with the highest population could be shaded yellow, the lowest shaded green and the intermediate ranges some color in between yellow and green. When the map is displayed, the colors make it readily apparent where the highest and lowest populations exist. (See How to Apply Translucent Effects to Themes for instructions on applying translucency effects to a ranged thematic map.)

Ranges are also useful when the size of the region is not directly related to the magnitude of the data values.

Types of Ranged Values

MapXtreme can create ranges automatically using five distribution methods:

- Equal count
- Equal ranges
- Standard Deviation
- Natural break
- Quantile
- Custom

Equal Count

Equal Count has the same number of records in each range. If you want to group 100 records into 4 ranges using Equal Count, MapXtreme computes the ranges so that approximately 25 records fall into each range, depending on the rounding factor you set.

When using Equal Count (or any other range method), it's important to watch out for any extreme data values that might affect your thematic map (in statistics, these values are referred to as outliers).

Equal Ranges

Equal Ranges divides records across ranges of equal size. For example, you have a field in your table with data values ranging from 1 to 100. You want to create a thematic map with four equal size ranges. MapX produces ranges 1–25, 26–50, 51–75, and 76–100. Keep in mind that MapXtreme may create ranges with no data records, depending on the distribution of your data.

Standard Deviation

When you create ranges using Standard Deviation the middle range breaks at the mean of your values, and the ranges above and below the middle range are one standard deviation above or below the mean.

Natural Break

Natural Break is a way to show data that is not evenly distributed. It creates ranges according to an algorithm that uses the average of each range to distribute the data more evenly across the ranges. It distributes the values so that the average of each range is as close as possible to each of the range values in that range. This ensures that the ranges are well-represented by their averages, and that data values within each of the ranges are fairly close together.

Quantile

Quantile is a second way to show data that is not evenly distributed. Quantiling uses two variables expressions. For example, use the Quantile distribution method to show the literacy rate as tied to population.

Custom Ranges

If none of the distribution methods meet your needs, you can create custom ranges using the method DistributionMethod.CustomRanges,. See the code example in the MapXtreme Developer Reference Help under the MapInfo.Thematics.RangedTheme.Recompute method.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_RangedTheme(ByVal map As Map)
    ‘ Create a ranged theme.
    Dim lyr As FeatureLayer = CType(map.Layers(0), FeatureLayer)
    Dim theme As MapInfo.Mapping.Thematics.RangedTheme = New _
        MapInfo.Mapping.Thematics.RangedTheme(lyr, “Pop_1990/Area(obj,_
        'sq mi')”, “PopDensity”, 5,DistributionMethod.EqualCountPerRange)

    ‘ Add the ranged theme to the layer.
    lyr.Modifiers.Append(theme)
End Sub
```

RangedLabelTheme

This class creates a range theme in which labels are drawn with a range style. For a more detailed discussion of ranged themes, see the section RangedTheme.

When To Use a RangedLabelTheme Class

Ranged label themes are useful when you want use the labels to convey information about what you are labeling. For example, you could use a ranged label theme when labeling city or town populations. The label of a city with a large population would have a larger font than the label of a town with a small population.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_RangedLabelTheme(ByVal labelSource
As MapInfo.Mapping.LabelSource, ByVal columnExpr As _
   String, ByVal themealias As String)
    ‘ Create new ranged label theme based on the label source of a
    ‘ LabelLayer already in the map. It will use 5 bins of equal range.
    Dim rangedLabelTheme As RangedLabelTheme = New _
        RangedLabelTheme(labelSource.Table, columnExpr, themealias, 5, _
        DistributionMethod.EqualCountPerRange)

    ‘ Add the label modifier to the label layer.
    Dim labelModifier As MapInfo.Mapping.LabelModifier = _
        CType(rangedLabelTheme, MapInfo.Mapping.LabelModifier)
    labelSource.Modifiers.Insert(0, labelModifier)
```

End Sub

Ranged Themes and Serialization

Beginning with v7.0.0, serialization and deserialization for RangedThemes and RangeLabelThemes has been changed in an effort to boost performance.

The changes are during theme serialization, when the theme's bin or category record counts are serialized. Upon deserialization, the record counts are no longer recomputed; rather the serialized record counts are applied to the theme. This change was put in place to improve upon serialization/deserialization performance.

It is important to note that for applications that are expecting to have their theme record counts updated upon deserialization will have to add logic to update the theme bin or category record counts after deserialization. For RangeTheme and RangeLabelTheme, this is a call to the MapInfo.Mapping.Thematic.IRangedTheme.**Recompute** method.

IndividualValueTheme

An Individual Value theme is a modifier theme that show points, lines, or boundaries that are shaded by individual values contained in a particular field of a dataset. You can use both numerical and nominal values in individual values maps. MapXtreme gives each unique value its own distinct style.

For example, use an IndividualValue theme to show zoning classifications for parcels of land. Each zone (commercial, residential, industrial) would display in a different color. Parcels that match the zone classification would be shaded in that color.

When To Use an IndividualValueTheme Class

If you are shading your points, lines, or boundaries using nominal data, you can shade only by individual values. Nominal data is either non-numeric data (e.g., name, type of cuisine served, or brand of automobile sold) or numeric data where the numbers do not represent measurements. For example, nominal data may be a column containing ID numbers.

Dates are considered numeric data and can be used in both ranged and individual values maps.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_IndividualValueTheme(ByVal_
   map As Map)

    ‘ Load a map based on one table
    map.Load(New MapTableLoader(“World.tab”))
    Dim fLyr As FeatureLayer = CType(map.Layers(“world”), FeatureLayer)

    ‘ Create an individual value theme
    Dim thm As IndividualValueTheme = New _
IndividualValueTheme(fLyr, “Country”, “World Pop”)

    ‘ Add the theme to the FeatureStyleModifiers list
    fLyr.Modifiers.Append(thm)
End Sub
```

Creating an IndividualValueTheme with Custom Bitmap Symbols

The following C# example shows how to create an IndividualValueTheme that use custom bitmap symbols. A table of available bitmap symbols is included in Custom Symbols.

```
// Open a connection to the Catalog
MapInfo.Data.MIConnection conn = new MapInfo.Data.MIConnection();
conn.Open();

// Retrieve a table from the Catalog
MapInfo.Data.Table ti=conn.Catalog.GetTable("usa_caps");
// Add it as a layer to MapControl
MapInfo.Mapping.FeatureLayer fl=mapControl1.Map.Layers["usa_caps"] _
     as FeatureLayer ;

// Create a new IndividualValueTheme
MapInfo.Mapping.Thematics.IndividualValueTheme iv=new _
     MapInfo.Mapping.Thematics.IndividualValueTheme(fl,"state","state");

// Add a custom bitmap symbol
MapInfo.Styles.BitmapPointStyle bitmappointstyle = new _
     MapInfo.Styles.BitmapPointStyle("AMBU1-32.BMP", _
     MapInfo.Styles.BitmapStyles.All ,System.Drawing.Color.Red , 30);
// Set the style
MapInfo.Styles.CompositeStyle cs = new _
     MapInfo.Styles.CompositeStyle(null, null, null, bitmappointstyle);

// Apply the style to the first bin
MapInfo.Mapping.Thematics.ModifierThemeBin mtb= iv.Bins[0];
mtb.Style.ApplyStyle(cs);

// Add another bitmap symbol
```

```
bitmappointstyle = new _MapInfo.Styles.BitmapPointStyle("BADG1-32.BMP", _
     MapInfo.Styles.BitmapStyles.All ,System.Drawing.Color.Red , 30);
// Set the style
cs=new MapInfo.Styles.CompositeStyle(null, null, null, _
     bitmappointstyle);

// Apply the symbol to the second bin
mtb= iv.Bins[1];
mtb.Style.ApplyStyle(cs);

// Append the style modifiers  to the feature layer
fl.Modifiers.Append (iv);

//Close the connection
conn.Close();
```

IndividualValueLabelTheme

This class creates an individual value thematic which operates on a layer's labels. For a more detailed discussion of individual value themes, see the section IndividualValueTheme.

When To Use an IndividualValueLabelTheme Class

As with ranged label themes, individual value label themes are also useful when you want use the labels to convey information about what you are labeling. When working with street data, for example, you could use an individual value label theme to label different types of roads with different fonts. In this case, a highway would be represented by a label with a different style than the label for a county road.

VB example:

```
Public Shared Sub MapInfo_Mapping_Thematics_IndividualValueLabelTheme(ByVal
labelSource As MapInfo.Mapping.LabelSource, ByVal columnExpr As String, ByVal _
     themeAlias As String)
    ‘ Create new individual value label theme
    Dim theme As IndividualValueLabelTheme = New _
     IndividualValueLabelTheme(labelSource.Table, columnExpr, themeAlias)

    ‘ Add the label modifier to the label layer.
    Dim labelModifier As MapInfo.Mapping.LabelModifier = CType(theme, _
     MapInfo.Mapping.LabelModifier)
    labelSource.Modifiers.Insert(0, labelModifier)
End Sub
```

IndividualValue Themes and Serialization

Beginning with v7.0.0, serialization and deserialization for IndividualValueThemes and IndividualValueLabelThemes has been changed in an effort to boost performance.

The changes are during theme serialization, when the theme's bin or category record counts are serialized. Upon deserialization, the record counts are no longer recomputed; rather the serialized record counts are applied to the theme. This change was put in place to improve upon serialization/deserialization performance.

It is important to note that for applications that are expecting to have their theme record counts updated upon deserialization will have to add logic to update the theme bin or category record counts after deserialization. For IndividualValueTheme and IndividualValueLabelTheme, this is a call to the MapInfo.Mapping.Thematics.IModifierTheme.**RecomputeBins** method.

DotDensityTheme

A dot density theme is a style modifier that draws the fill pattern of a region, using dots based on the numeric value of the theme's expression.

Dot density maps use dots to represent the data value associated with a boundary or region. The total number of dots in a region represents that region’s data value. If you have 10,000 senior citizens in a county, and each dot represents 100 senior citizens, there would be 100 dots in the county boundary.

When To Use a DotDensityTheme Class

A dot density theme is useful for showing raw data where one dot represents a large number of something: population, number of fast food restaurants, number of distributors who carry a brand of soda, etc.

For example, if you have a table of population broken down into county boundaries, you could use a dot density theme to show the concentration of people in each county boundary. There are two properties you control for dot density maps. You can specify the value of one dot. For example, to represent 20,000 high school students in Rensselaer County, New York using a dot density theme, you can specify that one dot represents 200 students. When you shade the county, the map would be drawn with 100 dots for that county.

```
VB example:

Public Shared Sub MapInfo_Mapping_Thematics_DotDensityTheme(ByVal map As Map)
```

```
    ' Load a map based on one table
    map.Load(New MapTableLoader("mexico.tab"))

    ' Create a dot density theme.
    ' Add it as a modifier.
    Dim lyr As FeatureLayer = map.Layers("mexico")
    Dim thm As MapInfo.Mapping.Thematics.DotDensityTheme = New
MapInfo.Mapping.Thematics.DotDensityTheme(lyr, "Pop_90", "mexico Pop",
System.Drawing.Color.Red, DotDensitySize.Large)

    ' thm.DotColor is System.Drawing.Color.Red
    ' thm.DotSize is DotDensitySize.Large

    ' Set each dot to represent 20,000 people
    thm.ValuePerDot = 20000

End Sub
```

Bivariate Thematic Maps

Bivariate thematic mapping uses point or line objects to represent two thematic variables. For example, a star can represent one variable, such as the number of teenagers, while a blue fill for the star represents their annual purchase amounts.

To create a bivariate map in MapX, you create two thematic maps, and layer one over the other so that the objects display two variables.

Types of Maps and Variables

The only types of thematic maps suitable for bivariate mapping are ranged and individual values maps. You can choose between two combinations for a bivariate map, depending on your data:

- two ranged maps
- one ranged map and one individual values map

If you have a non-numeric variable, one of your maps must be an individual values map. You cannot create a bivariate map with two non-numeric variables.

Displaying Attributes

To display two variables within one symbol, it is important to choose a different symbol attribute for each variable. For example, you cannot choose color for both variables because one color will overwrite the other. Choose from the following combinations:

- color and symbol type

- color and size
- size and symbol type

Symbol type should only be used for nominal or non-numeric data, as there is no inherent association between a symbol type and a quantity.

VB example:

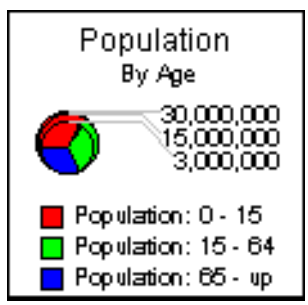
```
Public Shared Sub MapInfo_Mapping_Thematics_RangedThemeConstructor(ByVal lyr As
FeatureLayer)
    Dim thm As MapInfo.Mapping.Thematics.RangedTheme = New _
     MapInfo.Mapping.Thematics.RangedTheme(lyr, “Literacy”, “Pop_1994”, _
     “Literacy Quantile by Pop”, 4)
    lyr.Modifiers.Append(thm)
End Sub
```

Legends Overview

The MapInfo.Mapping.Legends namespace contains classes, interfaces and enumerations for creating and displaying thematic and cartographic legends. Legends are a collection of Thematic or Cartographic LegendFrames. Each frame contains a collection of LegendRows, Each LegendRow has text and a style property.

Theme Legends

Theme legends provide a key of colors, symbols, and styles used for themes. This key explains what the colors, symbols, and styles represent.



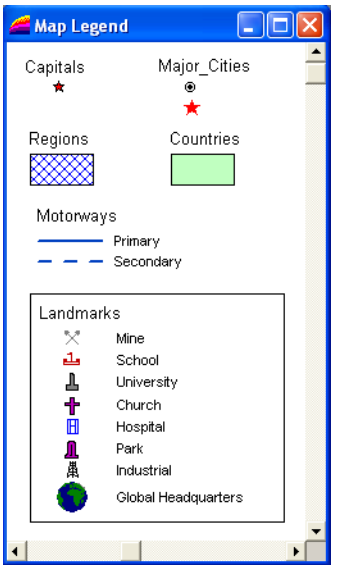
When to Use a Theme Legend

A theme legend is useful whenever you have a map that contains themes. With a weather map that displays precipitation, rainfall may be represented by varying shades of green. A theme legend would be important in visually explaining that the darkest shade of green represents the highest amount of rainfall while lighter shades represent lower amounts of rainfall.

In the \Samples\Desktop\Features\ThemeLegend folder, we provide a sample application that demonstrates how to create a ranged theme on a table, create a legend and legend frame for the them and add it to the map as an adornment.

Cartographic Legends

The cartographic legend class allows you to read and write cartographic legend metadata. The legend identifies each cartographic feature on the map using text and a style from the metadata.



When to Use a Cartographic Legend

A cartographic legend is very useful whenever you have a map that contains objects that represent items on the map. For example, a map with landmarks requires a cartographic legend. Hospitals, schools, churches, and airports would each be represented by a different symbol. The cartographic legend provides a visual explanation of the different types of landmarks that are represented on the map.

See the MapInfo.Mapping.Legends.Legend class in the Developer Reference for a code example that demonstrates how to create a cartographic legend.

Formatting a Legend

The MapInfo.Mapping.Legends.LegendFormat class contains properties that control how frames within a legend are drawn. You can control display properties such as alignment, the number of legend frames per row or column, spacing between frames, and whether the legend size and frame positions should be automatically adjusted.

The LegendFormat.FrameAlignment property is used in conjunction with the FramesPerRow for horizontal alignment and FramesPerColumn for vertical alignment.

For example, if your legend contains 10 frames, and you set the FrameAlignment to Horizontal and the FramesPerRow to 5, the legend will display two rows of frames with five frames in each row. If you had 10 frames per row, the 10 frames would display in a single row 10 frames wide.

Similar behavior happens with vertical alignment. If you have 10 frames and you set the FrameAlignment to Vertical and the FramesPerColumn to 5, you will end up with 5 rows of 2 frames (5 rows in 2 columns). The frames were aligned vertically up to 5 frames per column. When FramesPerColumn is set to 10, the legend would contain 10 rows of 1 frame each (10 frames per column).

The default setting for FramesPerRow and FramesPerColumn is 0. The value used is the current number of frames in the row or column as indicated by LegendFrameRows.Count or LegendFrameColumns.Count properties.

Export/Import Theme and Style

MapXtreme facilitates to export existing themes and styles applied on any layer, in XML format. Later on, this exported theme and style XML can be imported on any similar layer.

To use this feature create a new instance of ExportImportThemeStyle class and call the Export() or Import() method as required.

The sample code below illustrates this:

```
MapInfo.Mapping.MapLayer layer; //initialize your layer here.
string strExportedThemeXml="theme.xml"
Persistence.ExportImportThemeStyle exportImportThemeStyle
                                    = new ExportImportThemeStyle();
exportImportThemeStyle.Export(strExportedThemeXml, layer);
```

Similarly, with the help of Import() method, you can import the exported theme XML file on any layer. Also, an option to export/import themes and styles has been provided via user interface i.e. Layer control. Please refer to Export/Import Theme and Style.

15 – Stylizing Your Maps

Styles in MapXtreme affect many components of a mapping application, not just how a map feature looks. Styles are used for labels, text, themes, legends, selection and presentation with many controllable attributes so you can design practically any style you like.

This chapter discusses styles in terms of the MapXtreme framework, specifically the MapInfo.Styles namespace.

In this chapter:

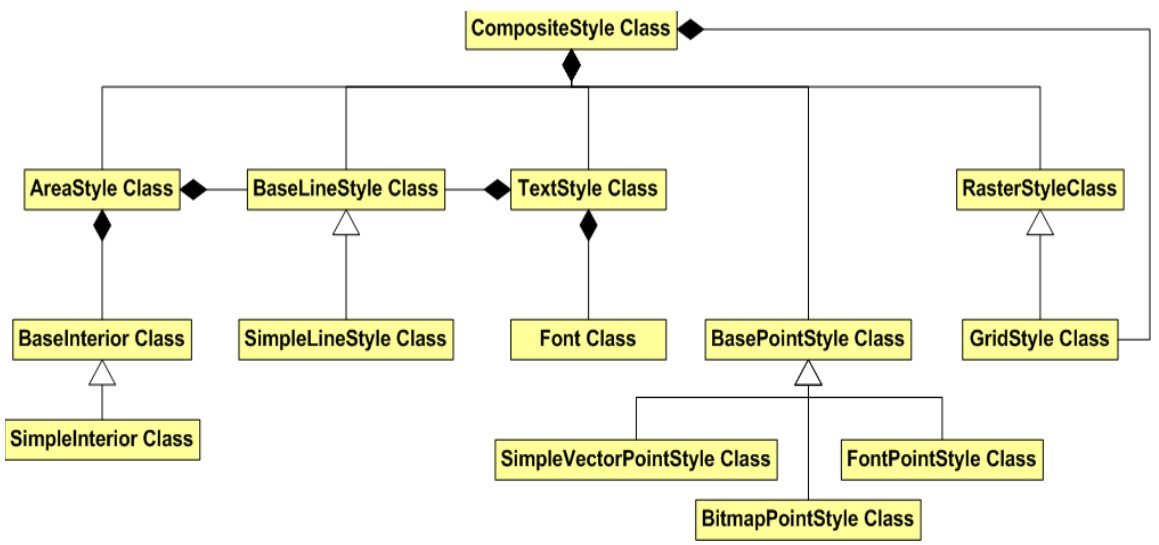
Overview of the MapInfo.Styles Namespace

The MapInfo.Styles namespace highlights the new Styles object model for MapXtreme. The Style class is the base class of all styles. Classes that derive from Style include AreaStyle, BaseLineStyle, BaseInterior, BasePointStyle, CompositeStyle, Font, RasterStyle, and TextStyle. Derived from BaseLineStyle is the SimpleLineStyle class. Derived from BaseInterior is the SimpleInterior class. Derived from BasePointStyle are the BitmapPointStyle, FontPointStyle, and SimpleVectorPointStyle classes. Additionally, GridStyle is derived from RasterStyle to include grid specific style settings. A StockStyles class is provided to create common style types.

You cannot instantiate the abstract style classes Style, BaseLineStyle, BaseInterior or BasePointStyle. You must create a particular type like SimpleLineStyle, or create them as a CompositeStyle.

Style is its own object; it is no longer stored in an object. Each table containing a geometry column also contains a Style column (alias MI_Style) containing a datatype Style. For style overrides, the Feature class provides FeatureStyleModifier and FeatureOverrideStyleModifiers. The Style object model also makes available several collection classes (style repositories) to hold styles for the style dialog controls.

All style classes support the ability to be applied in a sparse manner. See the section on FeatureOverrideStyleModifiers.



Styles are used in many areas of MapXtreme including the lines, interior fills and point styles that represent geographic features. Styles are also an integral part of labels, text, layouts, themes, overrides, legends and selections. Style properties range from standard color fills, line width and point size, to background effects, bitmaps as point styles, and

rotation angle. Practically any style property you can imagine is available for you to incorporate into your application. Styles can be changed globally or per feature, overridden for the current display or made as a permanent change.

MapXtreme comes with a number of sample styles to get you started. There are more than 170 interior fill patterns, approximately 120 line style patterns, and approximately 70 bitmap point style images. You can create bitmap images in any application that can create bitmaps, like MS Paint or Paint Shop Pro. There are virtually no size limitations on the image; however, the ability of MapXtreme to display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To make sure the image is displayed at the height and width you want, select the 'Display at actual size' option for BitmapPointStyles. Once the image is created, place it in the CustSymb directory. Custom symbols are located in programdata\Mapinfo\custsymb folder.

Additionally, MapXtreme ships with style controls and dialogs that allow you to rapidly add them to a form or web application. See the ChangeStyles and FeatureStyles sample applications under the Samples folder which bring together all the style classes discussed in this chapter. For more information on form style controls and dialogs see Chapter 7 Desktop Applications, Controls, Dialogs, and Tools. For more on web controls see Chapter 5 Web Applications, Controls, and Tools.

For visual representations of supported style elements, see Appendix F: Style Lookups.

StyleFactory

A StyleFactory class is available from MapInfo.Styles that contains methods to generate MapInfo Style objects from various types of style parameters. For example, MapInfo.Styles. StyleFactory.FromMBstring takes a MapBasic style clause as string input and returns a CompositeStyle. See the online Developer Reference for more information and a code example.

Style Descriptions

AreaStyle

The AreaStyle class contains style properties used for drawing regions. Regions are drawn using a BaseLineStyle and BaseInterior.

BitmapPointStyle

The MapInfo BitmapPointStyle class contains style properties for drawing points using custom bitmaps. Use this class for marking point locations. BitmapPointStyle is one of three types of supported point styles, the others being FontPointStyle and SimpleVectorPointStyle.

BitmapPointStyle has a ShowWhiteBackground property; if set to false, any white pixel in the bitmap is transparent. By default, ShowWhiteBackground is set to false. For example, you may wish to use your corporate logo to represent office locations worldwide, but do not want to cover up other map features in the immediate area. There are other settings that control how BitmapPointStyles are displayed. See the BitmapStyles enumeration in the online help for more details.

MapXtreme ships with a number of bitmap point styles to get you started. They are located in the CustSymb directory. You can also add your own bitmap images to this directory. The maximum number of images allowed is 32,767.

CompositeStyle

The CompositeStyle class encompasses the collection of all the style types used for default styles, modifier styles, and layer override styles. The CompositeStyle can also be used to describe styles for the Collection object type. The style types contained are AreaStyle, a BaseLineStyle derived class, TextStyle, a BasePointStyle derived class, RasterStyle and GridStyle. The CompositeStyle can be constructed with any or all of these types, but must contain at least one of the above.

For example, you can create a style override (a FeatureOverrideStyleModifier) to alter the appearance of all features in a layer. Since a single layer can contain points and lines and regions, you might need to specify point, line and area styles when you build your style override. You can specify all necessary style types in one CompositeStyle object, which you then pass to the FeatureOverrideStyleModifier constructor.

SimpleInterior

The MapInfo SimpleInterior class contains style properties used for filling the interior of regions. SimpleInterior attributes include pattern, foreground and background color, and background transparency. The default for a SimpleInterior is a solid white interior.

Font

The MapInfo Font class contains style properties used for drawing text. Attributes for fonts, include bold, italic, underline, strikeout, shadow, halo, all caps, double space, size, foreground and background color. You can also change the font type (e.g., Arial, Times New Roman) and the font size. Note that the outline font property has been removed. To create an outline, use halo with a black background.

FontPointStyle

The FontPointStyle class contains style properties used for drawing points using mainly the MapInfo.Styles.Font class. You can customize the point size, font color, angle of rotation and other font properties. The maximum point size allowed is 240 points.

GridStyle

This is a helper class containing display style information about a grid such as color inflections, null cell color and transparency. A grid is a map of continuous color gradation that represents interpolated data values. For more information on grids see Chapter 17 Working with Rasters and Grids.

RasterStyle

This is a helper class containing display style information for a raster image including brightness, contrast, grayscale (color on/off), transparency, and translucency. For more information on raster images see Chapter 17 Working with Rasters and Grids.

Hillshade

This is a helper class used to store the parameters for hill shading on a grid. Hill shading, also called relief shading, can be added to grid maps to show the effect of a light source on the map. This gives a grid map greater definition, particularly useful for elevation maps. Hill shade properties include the horizontal and vertical angles of the light source, and a vertical scale factor. For more information on grids see Chapter 17 Working with Rasters and Grids.

Inflection

This is used to hold a single inflection point which associates a color with a value. A grid has an array of inflections to represent its colors. A grid map is a map that shows a gradual color change across an area. The blending of one color into the next is due to the inflection. For more information on grids see Chapter 17 Working with Rasters and Grids.

SimpleLineStyle

The SimpleLineStyle class contains style properties used for drawing polylines based on the MapBasic Pen clause. It is used for map features such as streets and cable lines, as well as borders around regions. Attributes to describe SimpleLineStyles include pattern, width (in pixels or points) and color. The default for a SimpleLineStyle is a solid black 1-pixel wide line. Units for SimpleLineStyle are pixels (default) or points.

The LineWidth class in the MapInfo.Styles namespace is a helper class used to define the width and units of a line style.

BasePointStyle

This is an abstract base class for all MapInfo point styles. It cannot be instantiated. SimpleVectorPointStyle, BitmapPointStyle and FontPointStyle derive from this class.

BaseLineStyle

This is an abstract base class for all MapInfo line styles. It cannot be instantiated. SimpleLineStyle derives from this class.

BaseInterior

This is an abstract base class for all MapInfo interior styles. It cannot be instantiated. SimpleInterior derives from this class.

StockStyles

This class contains static methods to create various default style objects, including black, blue, red and white interiors, black, blue, and red lines, hollow interiors and lines, and default fonts and point styles.

```
SimpleLineStyle redLine = StockStyles.RedLineStyle( );
```

TextStyle

This class contains style properties used for drawing text. It contains a MapInfo.Styles.Font class and a BaseLineStyle derived class for callout lines. The BaseLineStyle is optional (the TextStyle may or may not contain one).

SimpleVectorPointStyle

This class contains style properties for drawing points using MapInfo's 3.0 Compatible proprietary font (MapInfow.fnt ships with MapXtreme). SimpleVectorPointStyle properties include color, point size and the shape code of the actual symbol you wish drawn for the point. The standard set includes symbols 31 through 67.

ⓘ Another symbol font set, called MapInfo Symbol, is a TrueType font set that displays using the FontPointStyle class.

Pre-defined Styles and the StyleRepository Class

MapXtreme ships with a variety of bitmap images covering many themes that can be used as bitmap point styles. More than 170 fill patterns and line styles are also available for use. These are all installed by the application and accessible either through the style dialogs (e.g., LineStyleDlg) or through the various StyleRepository classes.

For visual representations of supported style elements, see Appendix F: Style Lookups.

StyleRepository Class

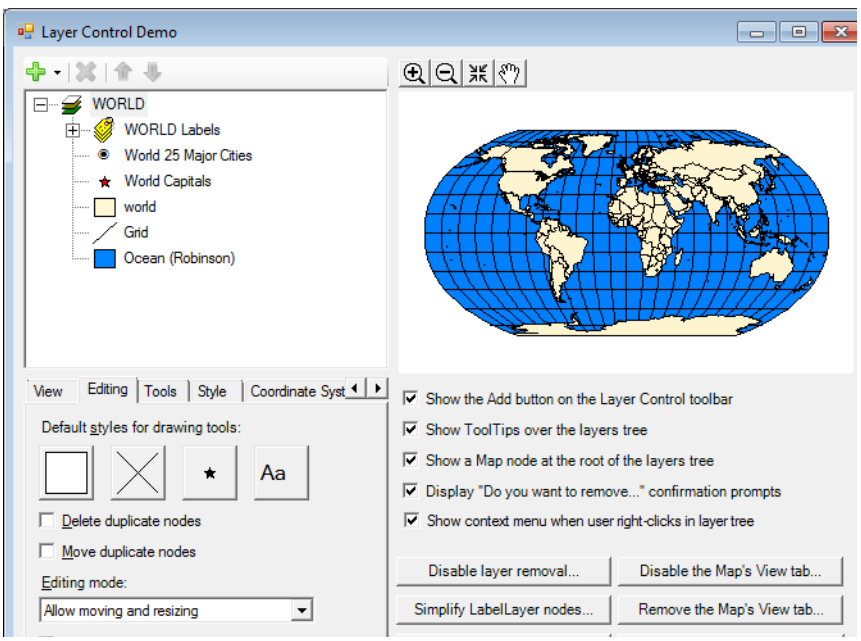
The StyleRepository class contains style collection classes (VectorSymbolRepository, BitmapSymbolRepository, LineStyleRepository, InteriorStyleRepository), which allows you to iterate through all current styles, as well as reload the collections with new styles from a specific file(s) or directory. The StyleRepository class also contains a repository (TrueTypeFontInfoRepository) that represents information about the TrueType fonts installed on the system.

The VectorSymbolRepository represents the set of symbols from the MapInfo 3.0 Compatible symbol set. The BitmapSymbolRepository represents the set of images currently found in the CustSymb directory. The LineStyleRepository represents the current set of patterns available for line styles. And the InteriorStyleRepository represents the set of interior patterns available.

Using Styles

Styles and Layer Control

Use LayerControl at design or runtime to modify and override styles. See the sample application LayerControl for an introduction to the MapInfo.Windows.Controls.LayerControl object model. This code sample uses a LayerControl object directly on a form; it does not demonstrate using the LayerControlDlg dialog box, which is a related, but separate class. Every operation demonstrated in this code sample could be applied to a LayerControlDlg object, as well, since the LayerControlDlg class exposes a LayerControl property.



LayerControl is discussed in Chapter 7 Desktop Applications, Controls, Dialogs, and Tools and in Chapter 27 Workspace Manager.

Creating a Custom Bitmap Style

You can create bitmap images in any application that can create bitmaps, like MS Paint or Paint Shop Pro. There are virtually no size limitations on the image; however, the ability of MapXtreme to display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To make sure the image is displayed at the height and width you want, select the 'Display at actual size option' for BitmapPointStyles. Once the image is created, place it in the CustSymb directory.

Overriding Styles

Styles can be permanently changed for features by saving the new style to the table. Styles for features can also be changed for the current display (not permanent) by overriding the current style For example, a ranged theme overrides the style of a region object to shade it. See Chapter 14 Using Themes and Legends for more information.

Label styles can also be overridden. This section introduces you to the main style override class for features. See more about features and labels in the chapter on the Mapping namespace Layers and Labels.

FeatureOverrideStyleModifiers

This class implements FeatureStyleModifier to override a feature's style. Its Style property is a composite style object that is used to specify what parts of a feature's style to override.

The contents of the style object passed to the Modify() method change dynamically for each feature that is drawn. This increases the drawing speed of layers that contain style modifiers. It is therefore important to make a copy of the style object if you need to use it elsewhere in your application. You should also note that for the CompositeStyles in the Style Stack passed to the FeatureStyleModifier.Modify() method, the Changed event does not fire.

Code Sample: FeatureOverrideStyleModifier

The following sample demonstrates how to use FeatureOverrideStyleModifier and layer FeatureStyleModifiers to change styles of various features within a map.

In this snippet from the ChangeStyles sample application, we want to override the world capitals layer with a single red symbol, but keep the point size.

VB example:

```
'Get the layer we want
   Dim _lyr As FeatureLayer = Me.mapControl1.Map.Layers("worldcap")

'Create a sparse point style
   Dim vs As MapInfo.Styles.SimpleVectorPointStyle = New _
SimpleVectorPointStyle

'Just change the color and code and attributes flag to indicate that
   vs.Code = 55
   vs.PointSize = 25
   vs.Color = System.Drawing.Color.Red
```

```
' And apply to the layer
   Dim fsm As FeatureOverrideStyleModifier = New _
      FeatureOverrideStyleModifier(Nothing, New _
      MapInfo.Styles.CompositeStyle(vs))
   _lyr.Modifiers.Append(fsm)
   Me.mapControl1.Map.Zoom = New MapInfo.Geometry.Distance(6250, _
      MapInfo.Geometry.DistanceUnit.Mile)
End Sub
```

Now MapXtreme facilitates to export/import styles along with thematics. For more information see, Export/Import Theme and Style.

16

16 – Spatial Objects and Coordinate Systems

This chapter covers the MapInfo.Geometry namespace and provides descriptions and examples for writing applications for creating and manipulating geometry objects.

In this chapter:

Introduction to MapInfo.Geometry Namespace

The MapInfo.Geometry namespace is used for creating and manipulating geometry objects, and the coordinate systems in which they are used. Geometry objects are used in maps to represent single points, such as cities (represented as point objects), boundary lines, such as county borders (represented by MultiCurve objects), and regions, such as countries or zip code areas (represented by MultiPolygon objects).

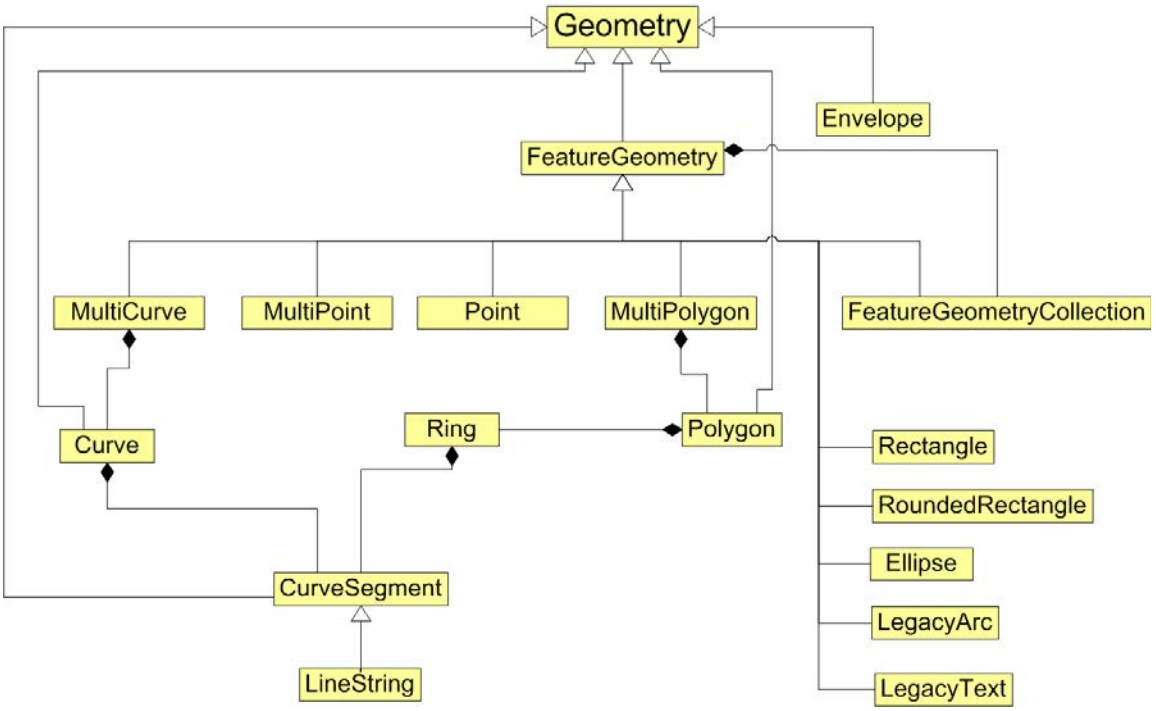
The classes, interfaces, and enumerations in the MapInfo.Geometry namespace define the types representing the geometries and coordinate systems used in displaying geographic features on a map. The Geometry model provides support for Z and M values on FeatureGeometry objects. Interfaces allow for creation and editing of the geometry objects. Methods such as Buffer, Combine, Difference, and Intersection provide object processing on single objects or pairs of objects.

Geometries

The Geometry class allows for the creation, editing, and other manipulation of geometry objects. Classes which inherit from the Geometry class and represent types of Geometry objects include Point, MultiPoint, Polygon, MultiPolygon, Curve, CurveSegment, LineString, and Ring. The following legacy classes are also inherited from the Geometry class: Rectangle, RoundedRectangle, Ellipse, LegacyArc, and LegacyText.

The Geometry class represents the topmost level of the MapInfo Geometry object model. This class is abstract, and cannot be instantiated. All classes that derive from this class contain knowledge concerning their coordinate system. All classes are able to make copies of themselves, and compare themselves to other Geometry objects for equality.

The diagram below shows a representation of the Geometry model.



Geometry Objects

All geometry objects in MapXtreme are created with a specific coordinate system that cannot be changed. If you need to alter the coordinate system of an object you can make a copy of that object in the new coordinate system.

Editing Geometry Objects

All Geometry objects contain a method for retrieving an interface to an editor that places the object into Edit Mode. Once editing is finished the EditingComplete() method needs to be called to signify that the editing of the object is complete. When the EditingComplete() method is called, the order of the objects contained by the Geometry is reshuffled and all references to them are dropped and need to be re-established in order to access them again.

For example, the user creates a MultiPolygon and then edits the MultiPolygon. If the user inadvertently moves a node of the interior ring to be outside of its containing Polygon the Polygon is no longer valid. When EditComplete is called, all the contained objects within the MultiPolygon are reshuffled, fixing the problem.

The geometry objects in the MapXtreme Object Model are described in the following sections.

FeatureGeometry Objects

The FeatureGeometry class is specifically designed to contain classes that can be placed in tables and that can be parts of Features and FeatureCollections. In order for something to be displayed in a map, it needs to be in a table. FeatureGeometry objects are by definition included in tables. Any object that is a subclass of Geometry and not a subclass of FeatureGeometry cannot be saved to a table or included as part of a Feature or FeatureCollection. An exception is thrown, or the program will not compile, if such an operation is attempted. The FeatureGeometry class, like the Geometry class is abstract and cannot be instantiated.

Support for M and Z Values

Feature geometries support reading and writing M and Z values at each node of the object.

Support for reading and writing M and Z values for linear objects was accomplished by extending the MapXtreme Geometry model. FeatureGeometry objects (Point, MultiPoint, MultiPolygon, MultiCurve and FeatureGeometryCollection) can now hold values for X, Y, Z and M for each node.

IsMeasured and Is3D properties allow you to determine whether the object has M or Z values. Additional properties and methods are provided to read and modify M or Z values at each node. The minimum and maximum ranges of M and Z values can be retrieved as well.

MapXtreme provides creation and editing capabilities for FeatureGeometries. For more information, see MapInfo.Geometry.FeatureGeometry class in the Developer Reference.

M values for MultiCurves provide valuable information in linear network applications for tracking and managing assets, events and conditions. See Chapter 22 Linear Referencing.

Point

Points are derived from the FeatureGeometry class and represent a single point on a map. Points can be contained within a MultiPoint collection and then operated upon collectively.

Use the following example code to model the creation of a Point:

```
using MapInfo.Geometry;
using Mapinfo.Design.Windows;

CoordSys longLatNad83;
CoordSysFactory coordSysFactory = new CoordSysFactory();
```

```
longLatNad83 = coordSysFactory.CreateLongLat
     (MapInfo.Geometry.DatumID.NAD83);
DPoint point = new DPoint(0.0, 0.0);
Point pointGeometry = new Point(LongLatNad83, point);
```

MultiPoint

A MultiPoint contains an unordered, disconnected set of Points and is useful for performing multiple operations on multiple points.

Use the following example code to model the creation of a MultiPoint object:

```
using MapInfo.Geometry;

CoordSys longLatNad83;
CoordSysFactory coordSysFactory = new CoordSysFactory();
longLatNad83=coordSysFactory.CreateLongLat
     (MapInfo.Geometry.DatumID.NAD83);
MultiPoint multiPointGeometry = new MultiPoint
     (longLatNad83, pointArray);
```

where `pointArray` is an array of DPoints.

MultiCurve

The MultiCurve class is derived from the FeatureGeometry class, and contains a possibly disconnected set of Curves. These Curves may interact in many ways; they can be connected or disconnected, and can intersect or overlap each other.

Although the Geometry object model supports multiple CurveSegments for each Curve, the current version of the MapInfo engine is limited to having one CurveSegment per Curve that is part of a FeatureGeometry (i.e., MultiCurve). This limitation derives from the current TAB file format, which remains largely unchanged for this version of MapInfo. Hence, the limitation concerns FeatureGeometry objects only.

Upon construction of a MultiCurve where the constructor takes a Curve or Curves which may contain multiple CurveSegments per Curve, the actual Curves contained in the constructed MultiCurve are altered to always contain only one CurveSegment per Curve. Currently, the only types of CurveSegments that exist are LineStrings. Curves containing multiple LineString CurveSegments have the LineStrings combined to form one large LineString.

Upon completion of editing (signified by calling EditingComplete(), any Curve which was added to the MultiCurve and contained multiple CurveSegments is altered in a similar manner as noted above to produce Curves containing single CurveSegments.This limitation, of Curves contained in MultiCurves always containing only a single CurveSegment, should be removed in future versions of MapInfo as new types of

CurveSegments are introduced (e.g., EllipticalArcs, CircularArcs, and Splines), and the TAB file format is altered. Also, during construction and on completion of editing, any empty Curves are automatically removed from the MultiCurve.

Line objects made up of two points that exist in MapInfo TAB files become MultiCurve FeatureGeometry objects. They can be detected as two-point Lines by using the IsLegacyLine property of the MultiCurve:

See the Developer Reference for a code example of creating and editing a MultiCurve object.

Measure Values on MultiCurves

The Geometry object model supports M and Z values on FeatureGeometry objects. M, or measure values, hold data at the nodes of MultiCurve objects that describes anything you wish to map and analyze, including physical assets, conditions or events. The M values play an important role in linear referencing and dynamic segmentation. For more information, see Chapter 22 Linear Referencing.

Curve Sort Order

The order of the Curves in a MultiCurve may be altered during construction, as compared to the array of Curves passed to the constructor, and upon completion of editing. Due to this, plus the removal of empty Curves, and limitations in the current implementation, any references to Curves contained in a MultiCurve prior to and during editing may no longer be valid after editing is completed (i.e., after EditingComplete() is called). If these objects are referenced, they throw an ObjectDisposedException. After editing, the parts of a FeatureGeometry should be reacquired to obtain a valid reference.

Curve sort order becomes an important factor when you are calling some of the linear referencing operations on a MultiCurve. MapXtreme includes the MapInfo.LinearReferencing.ICurveSorter interface to handle the sort order of individual curves. If no sort order is specified, MapXtreme returns the longest curve first, while the remaining curves are returned in an unknown order. When using CalculateMissingMeasures on an unordered MultiCurve, for example, MapXtreme could calculate the wrong M values for a node based on its position in the MultiCurve. Providing the correct sort order would eliminate that problem.

For more information, see Curve Order.

LineStrings

A LineString is a directed collection of sequential points that are connected in a linear manner. Any two consecutive points in the LineString are connected by a straight line. LineStrings can be part of Curves or Rings, or they can exist as a stand-alone Geometry.

LineStrings that are part of Curves or Rings inherit the coordinate system of their container. Stand-alone LineStrings can be empty. A LineString that is contained in a Curve or Ring that is not in Edit Mode cannot be empty, and must contain at least two points.

See the Developer Reference for a code example.

Rectangle

A Rectangle Geometry contains two points representing the lower left hand and upper right hand corners of the Rectangle. The other two points are implied. Rectangles are always axis aligned, and always appear to be rectangular in shape, regardless of the coordinate system, and are not projected. They do not contain any warping that may be represented by the coordinate system.

See the Developer Reference for a code example.

RoundedRectangle

A Rounded Rectangle behaves exactly like a Rectangle but is displayed with the corners appearing rounded as a display-time only feature. The corners display as quarter circles and the radius of the circle is controlled by the CornerRadius parameter.

Because RoundedRectangle objects, like rectangle objects, are defined by two points and always display axis-aligned and unprojected, they are designed to be used primarily for cosmetic display purposes. While many operations are available using Rectangle objects (e.g., Combine), internally, a MultiPolygon copy of the Rectangle is used for these operations. The resulting MultiPolygon contains 5 points (with the first and last points being identical), and are effected by the coordinate system. In some instances, the converted Rectangle may no longer appear rectangular. Use the CreateMultiPolygon method to convert a RoundedRectangle to a FeatureGeometry object.

See the Developer Reference for a code example.

Ellipse

The Ellipse is inscribed in an axis-aligned rectangle defined by a DRect. The DRect is defined by two points, the opposite corners of a rectangle, with the other two corners of the rectangle implied. The Ellipse displays as if it were unprojected, regardless of the coordinate system, and any skew that may be represented by the coordinate system.

Because Ellipse objects are defined by two points and always display axis-aligned and unprojected, they are designed to be used primarily for cosmetic display purposes. While many operations are available using Ellipse objects, internally, a MultiPolygon copy of the Ellipse is used for these operations. The resulting MultiPolygon is effected by the coordinate system and in some cases may no longer appear as a perfect ellipse.

See the Developer Reference for a code example.

LegacyArc

The LegacyArc object is a portion of an Ellipse and is defined by a DRect, a start angle, and an end angle. The Ellipse is constructed to be inscribed in the rectangle defined by the DRect. The rectangle, in which the Ellipse is inscribed, is axis-aligned and is always rectangular regardless of the coordinate system used. The angles are measured in degrees with zero being along the positive X-axis and positive angles being in the counterclockwise direction. The angles are only stored to a tenth of a degree resolution with values between 0.0 and 360.0.

Because LegacyArc objects are defined by two points (for the DRect) and angles, and are always displayed axis aligned, they are designed to be used primarily for cosmetic display purposes. While many operations are available using LegacyArc objects, internally, a MultiCurve copy of the LegacyArc is used for these operations. This can sometimes lead to unexpected results.

See the Developer Reference for a code example.

LegacyText

The LegacyText object is the MapInfo Professional equivalent of a text object. If a given database does not support Text the LegacyText object can be lost when using that format. LegacyText objects are placed within a geographically-sized rectangle with a lower-left anchor point specified. The point-size of the text is based upon what fits best within the rectangle.

LegacyText objects do not fit nicely into the Geometry model. Several methods available on the Geometry FeatureGeometry classes, such as Combine, make no sense for LegacyText and will throw a NotSupportedException. Text objects do exist in MapInfo native TAB files in the Geometry column. The LegacyText class provides a way to access these objects. Refer to online reference for specific behaviors of LegacyText objects.

Geometry Objects

Geometry objects that are not also FeatureGeometry objects need to be converted to a suitable FeatureGeometry object to be displayed on a map. Most FeatureGeometry classes contain constructors that take appropriate Geometry objects and create new FeatureGeometry objects:

```
using MapInfo.Geometry;

Curve curve = new Curve(csys, lineString);
MultiCurve multiCurve = new MultiCurve(curve.CoordSys, curve);
```

The code above creates the Curve using parameters defined elsewhere in the code of a CoordSys (`csys`) and a LineString (`lineString`). A new MultiCurve is then created using the CoordSys property of the Curve and the Curve itself.

In the example above, as in all FeatureGeometries created from objects, a copy of the original object is created because the reference cannot be shared.

Curve

The Curve class inherits from the CurveSegmentList class, and represents a contiguous linear Geometry. Curves contain a collection of CurveSegments that must remain contiguous. This class is included in the model to allow for future expansion and is part of the OGC standards.

Use the following example code to model the creation of a Curve:

```
using MapInfo.Geometry;

DPoint[] points = new DPoint[4];

points[0]= new DPoint(-88.135215,43.998892);
points[1]= new DPoint(-104.875119,43.998892);
points[2]= new DPoint(-120.242895,47.048364);
points[3]= new DPoint(-89.135215 46.998892);

LineString lineString = new LineString(csys, points);
Curve curve = new Curve(csys, lineString);
```

CurveSegments

At present a CurveSegment can only be a LineString. The class is designed to expand in future iterations of the product to include Spline, CircularArc, and EllipticalArc CurveSegments. Curves and Rings are comprised of CurveSegments.

Rings

A Ring is a collection of CurveSegments which must remain contiguous and closed.

Use the following example code to model the creation of a Ring:

```
using MapInfo.Geometry;

dPoints = new DPoint[102];
dPoints[0] = new DPoint(-109.171279,49.214879);
dPoints[1] = new DPoint(-109.169283,49.241794);
...
dPoints[101] = new DPoint(-109.171279,49.214879);
Ring newRing = new Ring(longLatNad83, CurveSegmentType.Linear, dPoints);
```

Polygon

A Polygon is an object made up of Rings. A polygon must have at least a single Ring which defines the exterior boundary of the Polygon. Other Rings can be included inside which then define holes in the Polygon. Once a Ring is placed inside of another Ring the object becomes a MultiPolygon.

Use the following example code to model the creation of a Polygon.

```
using MapInfo.Geometry;

DPoint[][] points = new DPoint[1][];
points[0] = polyPointArrays[0];
Polygon polygon = new Polygon
      (longLatNad83, CurveSegmentType.Linear, polyPointArrays[0]);
```

Including Your FeatureGeometry in a Map

Once a geometry is created you then need to add it to a map, allowing you to display it, select it, label it, or perform any other map-related operations on it.

```
Public Shared Sub MapInfo_Mapping_HowDoICreateFeatureAddToMap(ByVal mapControl1
As MapControl, ByVal connection As MIConnection, ByVal x As _
   Double, ByVal y As Double)
    Dim map As Map = mapControl1.Map

    'uses wldcty25 as a template
    Dim table As Table = _
   MapInfo.Engine.Session.Current.Catalog.GetTable("wldcty25")

    ' create a temp table and add a featurelayer for it
    Dim coordSys As CoordSys = map.GetDisplayCoordSys()
    Dim tableInfo As TableInfoMemTable = New TableInfoMemTable("temp")
    tableInfo.Temporary = True
```

```
    ' add Geometry column
    Dim column As Column

    ' specify coordsys for object column
    column = New GeometryColumn(coordSys)
    column.Alias = "MI_Geometry"
    column.DataType = MIDbType.FeatureGeometry
    tableInfo.Columns.Add(column)

    ' add style column
    column = New Column
    column.Alias = "MI_Style"
    column.DataType = MIDbType.Style
    tableInfo.Columns.Add(column)

    Dim pointTable As Table = _
  Session.Current.Catalog.CreateTable(tableInfo)

    ' Set the location and display style of the point
    Dim Geometry As FeatureGeometry = _
  New MapInfo.Geometry.Point(coordSys, x, y)
    Dim vStyle As SimpleVectorPointStyle = _
  New SimpleVectorPointStyle(37, Color.Red, 14)
    Dim cStyle As CompositeStyle = _
  New MapInfo.Styles.CompositeStyle(vStyle)

    'Update the table with the location and style of the new feature
    Dim cmd As MICommand = connection.CreateCommand()
    cmd.Parameters.Add("Geometry", MIDbType.FeatureGeometry)
    cmd.Parameters.Add("style", MIDbType.Style)
    cmd.CommandText = "Insert Into temp (MI_Geometry,MI_Style) values _
  (Geometry,style)"
    cmd.Prepare()
    cmd.Parameters(0).Value = Geometry
    cmd.Parameters(1).Value = cStyle
    Dim nchanged As Integer = cmd.ExecuteNonQuery()
    cmd.Dispose()

    'add the table to the map
    map.Layers.Add(New MapInfo.Mapping.FeatureLayer(pointTable))
End Sub
```

Checking for Points in Polygons

The following code example shows how to determine whether a point is inside the boundary of a FeatureGeometry (Multipolygon), on the boundary line or falls outside of it.

```
Public Shared Sub MapInfoGeometryContainsPoint()
Dim coordSysFactory As CoordSysFactory = Session.Current.CoordSysFactory
```

```
Dim coordSys As CoordSys = _
   coordSysFactory.CreateLongLat(MapInfo.Geometry.DatumID.NAD83)

Dim points(6) As DPoint
   points(0) = New DPoint(-0.705036, -0.122302)
   points(1) = New DPoint(-0.446043, 0.486811)
   points(2) = New DPoint(0.235012, 0.36211)
   points(3) = New DPoint(0.422062, -0.304556)
   points(4) = New DPoint(-0.244604, -0.71223)
   points(5) = New DPoint(-0.705036, -0.122302)
Dim multiCurve As MultiCurve = New _
   MultiCurve(coordSys,CurveSegmentType.Linear,points)
Dim multiPolygon As MultiPolygon = New _
   MultiPolygon(coordSys,CurveSegmentType.Linear,points)

Dim insidePoint As DPoint = New DPoint(-0.115108,0.160671)
Dim boundaryPoint As DPoint = New DPoint(-0.446043,0.486811)
Dim outsidePoint As DPoint = New DPoint(-1.103118,0.021583)

   If multiPolygon.ContainsPoint(insidePoint) Then _
      Console.WriteLine("Points inside area inclosed by closed _
      (GeometryDimension 2) objects are contained")
   End If
   If Not multiCurve.ContainsPoint(insidePoint) Then _
      Console.WriteLine("But this is not true for linear _
      (GeometryDimension 1) objects")
   End If
   If multiPolygon.ContainsPoint(boundaryPoint) Then _
      Console.WriteLine("Points on the boundary of closed objects _
      are contained")
   End If
   If multiCurve.ContainsPoint(boundaryPoint) Then _
      Console.WriteLine("Points lying on linear objects are contained")
   End If
   If Not multiPolygon.ContainsPoint(outsidePoint) Then _
      Console.WriteLine("Point completely outside closed objects _
      are not contained")
   End If
   If Not multiCurve.ContainsPoint(outsidePoint) Then _
      Console.WriteLine("Point completely outside linear objects _
      are not contained")
   End If
End Sub
```

Coordinate Systems

Coordinate systems describe the domain in which a particular object or set of objects reside. The coordinate system allows for the delineation, in specific terms, of the object or objects being described. The CoordSys classes contain methods, properties and interfaces that allow for the creation, manipulation, and editing of coordinate systems.

When Geometries are created, they are created in a particular coordinate system specified in the creation of the object. Objects cannot change the coordinate system in which they were created. They can only be copied into another coordinate system.

The CoordSys class facilitates the creation and manipulation of coordinate systems. The Coordsys class uses an XML version of the projection file (C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x\MapInfoCoordinateSystemSet.xml).

The CoordSysFactory object contains registered coordinate systems. CoordSys definitions can be registered by loading one or more XML projection files or by using the RegisterCoordSys, or RegisterCoordSysInfo methods. Create CoordSys objects from the factory, or code-codespace (EPSG, SRID), PRJ string, MapBasic string, and other Factory creation methods. There are also Military Grid Reference System conversion methods in the CoordSys class.

Creating a CoordSys Object

The following sample code shows the creation of CoordSys objects several different ways: using a MapInfo codespace; through EPSG; as longitude/latitude from a PRJ string; from a MapBasic string; and through SRID.

VB example:

```
Public Shared Sub MapInfoGeometryCreateCoordSys()
    Dim factory As CoordSysFactory =  Session.Current.CoordSysFactory

    ' create CoordSys objects from srsName
    Dim csysWGS84 As CoordSys = factory.CreateCoordSys("EPSG:4326")
    Dim csysNAD83 As CoordSys = factory.CreateCoordSys_
      ("mapinfo:coordsys 1,74")
    Dim csysNAD27 As CoordSys = factory.CreateCoordSys("SRID:8260")

    ' create CoordSys objects from code/codeSpace
    csysWGS84 = factory.CreateCoordSys("4326", CodeSpace.Epsg)
    csysNAD83 = factory.CreateCoordSys("coordsys 1,74", CodeSpace.MapInfo)
    csysNAD27 = factory.CreateCoordSys("8260", CodeSpace.Srid)

    ' create CoordSys objects from user-defined parameters
    Dim dat As Datum = factory.CreateDatum(DatumID.WGS84)
```

```
    csysWGS84 = factory.CreateCoordSys(CoordSysType.LongLat, _
        dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
    dat = factory.CreateDatum(DatumID.NAD83)
    csysNAD83 = factory.CreateCoordSys(CoordSysType.LongLat, _
        dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
    dat = factory.CreateDatum(DatumID.NAD27ContinentalUS)
    csysNAD27 = factory.CreateCoordSys(CoordSysType.LongLat, _
        dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
    ' create Long/Lat coordinate system
    csysWGS84 = factory.CreateLongLat(DatumID.WGS84)
    csysNAD83 = factory.CreateLongLat(DatumID.NAD83)
    csysNAD27 = factory.CreateLongLat(DatumID.NAD27ContinentalUS)

    ' create from MapBasic string
    Dim csysRGF93 As CoordSys = _
        factory.CreateFromMapBasicString("CoordSys Earth Projection 3, _
        33, ""m"", 3, 46.5, 44, 49, 700000, 6600000")
    ' create from PRJ string
    csysNAD83 = factory.CreateFromPrjString("1, 74")
Sub
```

Changing the Coordinate System of a Geometry Object

The next example illustrates how to convert a Geometry object from one coordinate system to another.

VB example:

```
Public Shared  Sub MapInfoGeomeTryCoordSys(ByRef coordSys As _
     CoordSys, ByRef points() As DPoint, ByRef alternateCoordSys _
     as CoordSys
   ' All Geometry constructors require a CoordSys parameter
   ' Note that the points array is assumed to be in coordSys
        Dim lineString As LineString =  New _
          LineString(coordSys,points)

   ' The Geometry has a reference to the CoordSys used during
   ' construction. Unlike the coordinate data represented by the
   ' points array, the CoordSys' is not copied
      If ReferenceEquals(coordSys, lineString.CoordSys) Then
        Console.WriteLine("Geometry objects hold a reference to _
          the CoordSys used during construction")
        End If

   ' if you want to convert the object to another coordinate
   ' system, you need to make a new copy using one of the copy methods
      If Not coordSys.Equals(alternateCoordSys) Then
        Dim newGeometry as MapInfo.Geometry.Geometry = _
          lineString.Copy(alternateCoordSys)
        End If
      End Sub
```

Determining the Coordinate System of a Map in MapControl

The following code examples shows how to determine the coordinate system for a Map object.

VB example:

```
Public Shared Sub _
   MapInfo_Mapping_FeatureViewerGetDisplayCoordSys(ByVal map As Map)
     'Load the Default Projection File so that we can get the name.

     MapInfo.Engine.Session.Current.CoordSysFactory.LoadDefault_
       ProjectionFile()

     'Get the Coordinate System object for current map in the MapControl
     Dim mapCoordSys As MapInfo.Geometry.CoordSys = _
       map.GetDisplayCoordSys()

   'Assign the name of the Coordinate System to a string variable.
   'note: the CoordSysName function will return a blank string if the
   'Coordinate System is not found in the current CoordinateSystemSet
   '(loaded by the LoadDefaultProjectionFile above).
     Dim mapCoordSysName As String = _
       MapInfo.Engine.Session.Current.CoordSysFactory.CoordSys_
       Name(mapCoordSys)
End Sub
```

Adding Coordinate Systems to MapXtreme

If the MapInfoCoordinateSystemSet.xml file does not contain a coordinate system to match your needs, you may add it to MapXtreme. This feature supports adding EPSG codes and SRID codes to extend MapXtreme's capabilities.

EPSG codes represent a collection of coordinate systems (known as codespaces) maintained in the EPSG Geodetic Parameter Dataset under the auspices of the International Association of Oil & Gas Producers (OPG). The OPG's Survey and Positioning Committee took over this responsibility from the European Petroleum Survey Group in 2005.

SRID codes are unique spatial reference numbers that refer to codespaces for Oracle Spatial tables.

(MapXtreme supports a third codespace called MapInfo.)

MapXtreme provides you with many of the common EPSG and SRID mappings. If you need to register a different EPSG or SRID code to a particular coordinate system, this feature provides you with two methods to do so.

To extend MapXtreme's ability to use any EPSG or SRID codespace, you may add the information programmatically, in which case, the coordinate system information will only last as long as the MapXtreme Session. Or you may add it to the Web.config file for your web application or app.config file for a desktop application as a more permanent solution. Each is discussed below.

Register EPSG and SRID Codes Programmatically

The MapInfo.Geometry.CoordSysFactory class contains methods that allow you to register EPSG and SRID codes to a specified coordinate system.

RegisterEPSGCode() and RegisterSRIDCode() each take two parameters: one being the EPSG or SRID code that represents the codespace, the second is the coordinate system information that first parameter will map to.

The following example demonstrates registering a fictional code with the Long/Lat NAD83 coordinate system.

VB example:

```
Public Shared Sub MapInfo_Geometry_RegisterEPSGCode()
  Dim factory As CoordSysFactory = Session.Current.CoordSysFactory
  ' create CoordSys objects from srsName
  Dim csysNAD83 As CoordSys = _
     factory.CreateCoordSys("mapinfo:coordsys 1,74")
  ' 9998 is a fictional code for demonstration purposes
  Try
    factory.RegisterEPSGCode(9998, csysNAD83)
  Catch ae As ApplicationException
    'code already exists.  Codes cannot be duplicated
  End Try
End Sub
```

ⓘ If the EPSG or SRID code already exists, an exception will be thrown indicating this fact.

To determine if a coordinate system for the MapInfo, EPSG or SRID codespace is already supported, call this method:

- `MapInfo.Geometry.CoordSys.Code(codespace).`

This method returns the first (or only) occurrence of the codespace that matches or null, if it does not exist.

Similarly, to return the first SRSName in the list that matches the input codespace, call this method:

- `MapInfo.Geometry.CoordSys.SRSName(codespace)`.

An SRSName (Spatial Reference System) represents the name of a coordinate reference system written in GML (Geography Markup Language). This is typically, a friendly name for the coordinate system, not a list of parameter values.

To get a list of all the codes and coordinate systems that are mapped to a particular coordinate system, MapXtreme provides the following methods:

- `MapInfo.Geometry.CoordSys.Codes(codeSpace)`
- `MapInfo.Geometry.CoordSys.SrsNames(codeSpace)`

Keep in mind that the coordinate system information you added programmatically, will only be maintained during the lifetime of the MapXtreme Session.

Register EPSG and SRID Codes to a Web or Desktop Configuration File

The second, and more permanent, way to add EPSG or SRID codes to MapXtreme, is by adding the information to your web application's Web.config file or your desktop application app.config[1] file. The code below in bold shows the information to copy and paste into your config file. An explanation of the code follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <section name="MapInfo.CoreEngine"
type="MapInfo.Engine.ConfigSectionHandler, MapInfo.CoreEngine,
Version=6.8.0.536, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1" />
   </configSections>
       <appSettings>
          <add key="MapInfo.Engine.Session.UseCallContext" value="false" />
       </appSettings>
   <MapInfo.CoreEngine>
      <EPSG_Code_Mappings>
         <EPSG_Code_Mapping>
            <srsName>My Custom CRS</srsName>
            <srsID>
               <code>coordsys 8,74,8,-
110.0833333333,47.5,0.9999375,2624666.667,328083.3333</code>
               <codeSpace>mapinfo</codeSpace>
               <remarks>My Custom CRS</remarks>
            </srsID>
            <EPSG_Codes>
               <EPSG_Code>9987</EPSG_Code>
               <EPSG_Code>9988</EPSG_Code>
               <EPSG_Code>9989</EPSG_Code>
```

1. If your desktop application does not have an app.config file, you can create one by adding it to your project from the Visual Studio Application Configuration File template.

```
            </EPSG_Codes>
         </EPSG_Code_Mapping>
      </EPSG_Code_Mappings>
      <SRID_Code_Mappings>
         <SRID_Code_Mapping>
            <srsName>My Custom CRS</srsName>
            <srsID>
               <code>coordsys 8,74,8,-
114.0833333333,47.5,0.9999375,2624666.667,328083.3333</code>
               <codeSpace>mapinfo</codeSpace>
               <remarks>My Custom CRS</remarks>
            </srsID>
            <SRID_Codes>
               <SRID_Code>9990</SRID_Code>
               <SRID_Code>9991</SRID_Code>
               <SRID_Code>9992</SRID_Code>
            </SRID_Codes>
         </SRID_Code_Mapping>
      </SRID_Code_Mappings>
  </MapInfo.CoreEngine>
</configuration>
```

The code above shows that there are two sections of information to add. One is to identify the correct CoreEngine dll and version number[1], the second is to add elements for EPSG and SRID code mappings.

An EPSG code mapping consists of an SRSName, SRS ID, and EPSG code(s). The SRSID is further defined by the parameters and codespace for the coordinate system.

An SRID code mapping is similar to the EPSG code mapping, except it refers to the Oracle Spatial identification number.

For more information on coordinate systems, see Appendix H: Elements of a Coordinate System. For information on the MapInfo codespace, see Appendix G: Defining the MapInfo Codespace.

1. To determine the correct version number for the MapInfo.CoreEngine assembly, from the Start menu, choose Run and type Assembly in the Run dialog. Every registered MapXtreme assembly is listed in the global assembly cache list.

17

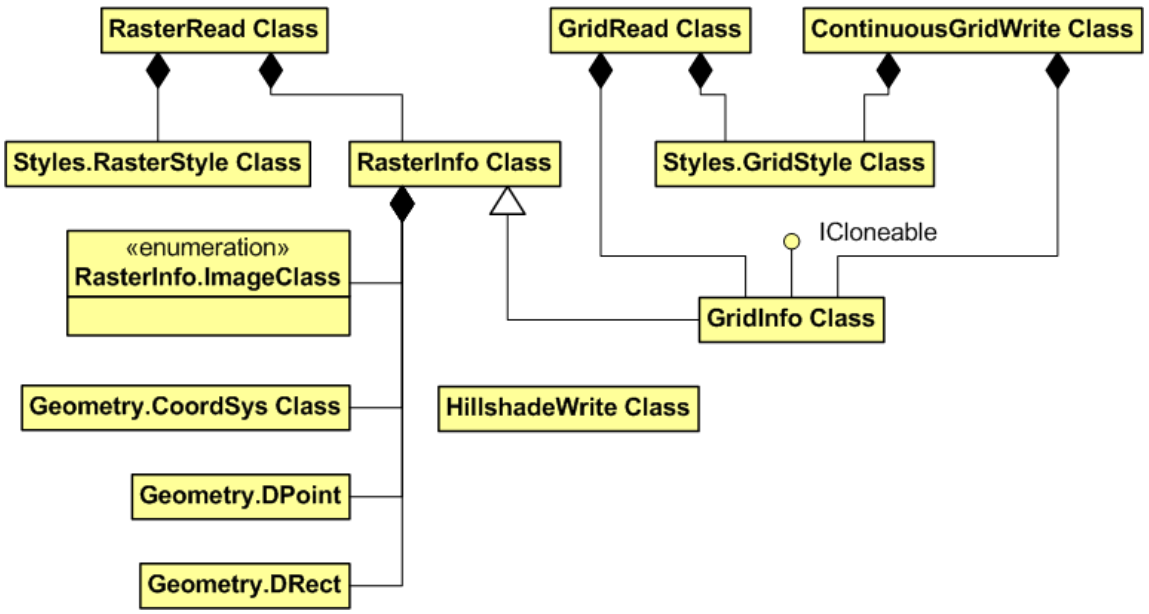
17 – Working with Rasters and Grids

The MapInfo.Raster namespace contains all of the classes that control the use and display of raster and grid images in MapXtreme. Rasters are computer graphics that are composed of pixels that render a whole image. Many satellite images are rendered as raster images. Grid images are thematic maps that show a continuous gradation of color to represent interpolated information.

In this chapter:

Overview of the MapInfo.Raster Namespace

The MapInfo.Raster namespace exposes the full functionality of Precisely's C/C++ Raster and Grid Engine APIs. Raster images are bitmaps that provide useful background and reference layers to maps. Grid images are a type of theme that shows a continuous gradation of color across the image. The gradation represents an interpretation of the underlying data. Grid images are rasters that have data associated with them. A common example is an elevation map. Rasters do not have any underlying data.



Raster Images

Raster images make excellent backgrounds for maps. For example, aerial photographs that show real-world detail such as buildings, refineries, and vegetation are well-suited as base layers for a map. Scanned paper maps are another example of a raster image. Use a raster image as a base layer and overlay vector data such as street networks, point locations representing customers, and postal boundaries, to create useful and visually appealing maps.

Raster images used with vector data must be registered so that known geographic points on the image coincide with the same features on the vector data. Additionally, company logos and other art you wish to display in MapXtreme must be registered to some location on earth, even though they are not true geo-referenced data. Many raster images

available today come with a registration file. Examples include GeoTIFF, ADRG, ASRP, CADRG, and CIB. To register a raster image, you can bring it into MapInfo Professional and register it there. The registration information is stored in a .TAB file.

Below are the raster image formats supported in MapXtreme:

- TIFF and GeoTIFF (*.tif)
- MrSID (*.sid)
- ECW (*.ecw)
- Spot (*.bil)
- JPEG (*.jpg)
- JPEG2000 (*.jp2, *.j2K)
- PCX (*.pcx)
- GIF (*.gif)
- Windows Bitmap (*.bmp)
- PNG (*.png)
- Photoshop (*.psd)
- Targa (*.tga)
- Windows Metafile (*.wmf)
- Windows Enhance Metafile (*.emf)
- Wireless BMP (.WBMP)
- Vertical Mapper Continuous Grid (*.grd)
- Vertical Mapper Classified Grid (*.grc)
- ADRG - ARC Digitized Raster Graphics (*.gen)
- ASRP - ARC Standard Raster Product (various file extensions)
- CADRG - Compressed ARC Digitized Raster Graphics (*.gen)
- CIB - Controlled Image Base (various file extensions)
- NITF - National Imagery Transmission Format (*.ntf)
- MRR - Multi Resolution Raster Format (*.mrr)

Additional raster formats may be supported on a system if a custom raster handler is installed.

Raster Classes

The main classes for raster images are MapInfo.Raster.RasterInfo and RasterRead. Style information is handled by MapInfo.Style.RasterStyle.

RasterInfo provides information about the height and width of the image in pixels, raster format, color depth, and registration information. See the RasterInfo sample application provided in .\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\RasterInfo

RasterRead is the class that reads the raster image and style information in order for the image to be rendered.

RasterStyle is concerned with how the raster looks. You can control brightness and contrast, display a color image as grayscale, set the transparency and translucency. MapXtreme supports transparency for one color per image. That means that everywhere that color exists in the image it will be invisible, which allows the layer below the image to show through. Translucency is the degree of transparency for the entire image. If you need a layer below a raster image to show through, set a high translucency (100 percent is transparent).

Raster columns are read only so you cannot permanently change their style. However you can programmatically set and get the image's attributes such as brightness, grayscale, and translucency. See the RasterStyle class in the online Developer Reference.

Raster Images and Coordinate Systems

When you display a raster, grid or WMS image as a map layer, MapXtreme automatically sets the rotation and projection of all the vector map layers, so that they match the rotation and projection of the raster image.

If a map includes more than one raster, grid or WMS image layer, MapXtreme automatically displays the map in the projection specified by the most visible raster image. The coordinate system could then change as the map view changes (due to zooming or panning) if a different image with a different projection becomes the most visible. In this case, you cannot change the map's display coordinate system.

Raster Reprojection

Raster reprojection enables you to change the cartographic projection of a raster layer in a map. Many types of raster images can be reprojected. Some examples include satellite and aerial photo images, scanned maps, as well as grids and seamless raster tables.

You can control the reprojection of both raster and vector layers. When you add either a raster or vector layer to a map, the new layer is reprojected into the current map window projection.

When you change the projection of a map window that contains a combination of vector and raster layers, all the layers, both raster and vector, can be reprojected to the new map window projection.

You can access raster reprojection settings in the MapXtreme API, as well as in the user interface of the Workspace Manager's Layer Control. For information using raster reprojection in the API, see the MapInfo.Mapping.RasterReprojectionMethod in the MapXtreme Developer Reference. For information on the raster reprojection user interface, see Raster Reprojection.

Reprojected Images in Workspaces

The raster reprojection properties can be read into and from a workspace.

MapXtreme can also write raster reprojection information from MapInfo Professional to an .mws workspace. This capability enables you to create your workspace in MapInfo Professional, and then load it into MapXtreme.

Once your workspace is loaded into MapXtreme, you can modify the raster reprojection via the API or the through the user interface in the Workspace Manager's Layer Control.

Raster Image Limitations

- You cannot select any features a raster layer.
- You cannot search for features a raster layer.

Code Sample: Adding a Raster Image to a Map

Adding a raster image to a map is the same as adding any other layer.

C# example:

```
Table MyTable = Session.Current.Catalog.OpenTable("MyRaster.tab");
FeatureLayer MyLayer = new FeatureLayer(MyTable);
MyMap.Layers.Add(MyLayer);
```

VB example:

```
Dim MyTable As Table = _ Session.Current.Catalog.OpenTable("MyRaster.tab")
Dim MyLayer As FeatureLayer =  New FeatureLayer(MyTable)
MyMap.Layers.Add(MyLayer)
```

Raster Handlers

MapXtreme can use one of many different libraries to load a raster image. When a raster image is loaded by MapXtreme, it searches through these DLLs and checks if the given file can be read by that DLL. Once a DLL/Raster format match is made, MapXtreme knows which DLL does the format handling for the file. The format handlers are DLL's named "xxxxxxxx.RHX". The base part of the name is based on the format. The extension always begins with RH, but can end in any letter (A-Z). When searching for a format handler, MapXtreme searches the format alphabetically, starting with RHA and continuing until RHZ. This process allows MapXtreme to prioritize which handlers are used. For example, SPOT files are checked for before any other formats since they are just raw data that can be confused with other formats. The SPOT handler's extension is RHD. The Halo format handlers are named RHV. The LEADTOOLS format handlers are named RHX.

Raster handlers are located by default in the \Program Files\ Common\ MapInfo\ MapXtreme\ 9.x.x\ RasterGridHandlers folder.

LEADTOOLS Win32 Pro provided by LEAD Technologies, Inc. and HALO Imaging libraries provided by Media Cybernetics are included with MapXtreme. The LEADTOOLS raster handler loads the entire raster image into memory at the time the image is referenced in MapXtreme. This means that the image takes longer to load and may fail if the image is very large due to extreme memory requirements, but panning and zooming are faster.

The HALO raster handler only loads into memory what it needs to display, so it loads the image faster, but panning and zooming is slower. Because of the HALO.RHV comes before LEADTOOL.RHX alphabetically, the HALO handler will attempt to read in the image first. To change this order, see Configuring Custom Raster Handlers.

Supported Formats

The following table shows the provided raster handlers and the formats they support.

Raster Handler	Raster Format
SPOT.RHD	Spot (.bil)
ECW5.RHD	ECW (.ecw, .ers)
MRSID.RHE	MrSID (.sid)
ADRGASRP.RHL	ADRG, ASRP, USRP

Raster Handler	Raster Format
CADRGCIB.RHL	CADRG, CIB, NITF
TIFF.RHL	TIF (.tif) *
VMGRID.RHL	Vertical Mapper (.grd, .grc) †
HALO.RHV	BMP, TIF, GIF, TGA, JPEG, PCX
LEADTOOL.RHX	BMP, GIF, JPEG, JPEG2000, PNG, TIF, PSD, WMF, EMF
MIRASTERHANDLERRT.RHZ	MRR, GRD, GRC, TIF, BIL, BIP, BSQ, ERS, ASC, FLT, ADF, ZIP‡

* Only handler that supports georeferencing for TIF. If another handler that supports TIFF files is tried before this TIFF handler, you can lose support for Georeferenced TIFF files. So, to support georeferenced TIFF files, this TIFF handler should be tried before any other handler that supports TIFF.

† Can be displayed as either grid or raster. The .TAB file determines whether the image should be drawn using the grid handler or the raster handler.

‡ Only in 64-bit environment.

There are a few raster formats that we do not support in multi-threaded applications (e.g., in an ASP.NET application). These formats are as follows:

Read
Vertical Mapper GRD, GRC
JPEG from Halo (The LeadTools JPEG handler is threadsafe)
TIFF w/jpeg compression from Halo (The LeadTools TIFF handler is threadsafe)
Export
JPEG 2000 (Win2K only) from LeadTools
TIFF CMYK from LeadTools

Raster Handler Properties

MapXtreme invokes raster handlers in alphabetic order, based on the last character in the handler's file extension. You can now control the order programmatically with two properties in the MapInfo.Data.TableRaster class.

The **TableInfoRaster.PreferredHandler** property specifies which raster handler to use to open a raster image. The raster engine attempts to use the specified handler to open the raster image. If it fails, the normal process of invoking handlers alphabetically is followed. If successful, the actual handler that is used is reported in the new property **TableInfoRaster.ActualHandlerUsed**.

Note that this value is read from and stored in the ClientMetadata of the table. If a value exists in a .TAB file's metadata it will be read in and recognized. If WriteTabFile is called the value will be persisted into the begin_metadata section of the .TAB file. This property is equivalent to TableInfo.ClientMetadata["\\PreferredRasterHandler"].

MRR - Multi Resolution Raster Format

MRR is a new raster data format that encompasses all types of raster data like image, classified, discrete, and continuous. Though there are already more than a hundred raster data formats, there exists a requirement for a new, unifying, format that handles all types of data and provides new capabilities to improve the user experience when visualizing and processing raster data.

The size and number of raster datasets available to the GIS professional is growing rapidly. The resolution and coverage of remote sensing platforms increases with every new generation of hardware. The physical number of satellites and other remote sensing platforms in operation is increasing year on year. Storing, managing, visualizing and processing this data is becoming increasingly challenging for data providers and consumers.

To meet these challenges, Precisely has developed a new raster storage format called 'MRR'. It provides a flexible solution for the entire spectrum of industry requirements. It is a unifying and enabling technology. It unifies the storage of all kinds of raster data – imagery, spectral imagery, continuous gridded data and thematic data – and removes the barriers to working with different kinds of raster data in the same context. It enables the highest quality visualization and processing of raster data – at any scale and for a raster of any size.

The key enabling capabilities of the MRR format are listed below:

- Storage of image data, classified (thematic) data, continuous and discrete data.

- Removes all restrictions on the size of raster datasets which means it supports raster datasets of virtually unlimited size.
- Extends the concept of a raster from a simple 2D array of cells to an extensible, sparse matrix of tiles.
- Contains a data pyramid that provides access to data at any scale in linear time.
- Achieves efficient storage using lossless and lossy compression techniques via industry standard compression codecs.
- Supports the temporal dimension, allowing data to be accumulated and accessed by time.
- Supports a wide number of data types including complex numbers.
- Supports a wide and extensible number of data types including complex numbers.
- Stores one or more multi-banded fields.
- Provides local registration and cell size for each field, and tile decimation.
- Computes and stores high quality statistics to support rendering.
- For simplicity, an MRR is contained within a single file on disk.

Benefits of MRR Technology

The new MRR file format is an enabling technology which offers the following key benefits:

- **Extremely large raster/grid datasets** - The MRR format has been designed to address the difficulty of handling extremely large datasets. It supports a wide variety of data types, and provides industry standard compression techniques and predictive encoding techniques. The MRR format minimizes storage requirements and provides efficient data access to extremely large raster files. It has been designed to deliver an enhanced visualization experience by maintaining an overview data pyramid. This guarantees efficient data visualization at any scale.
- **Unified data** - The MRR format is capable of storing all types of raster/grid data in common use. It can be used to store numeric gridded data, imagery and classified (thematic) data.
- **Sparse data** - Sometimes raster/grid datasets can be extremely large but only sparsely populated with valid data. An example of such a dataset is a LiDAR survey which follows along a pipeline for several 100 km but is only 100 m wide. In cases such as these, the MRR format does not store values for the regions containing no data, but still remains flexible enough to add additional data to the source file any time. This allows extremely sparse datasets to be stored efficiently on disk, be rapidly accessed and easily updated.

- **Change in raster data** - The MRR format provides unique capability to store changes in raster/grid data over any period of time. Using the MRR format you can store and then view changes in a dataset for a point in time or over a given period of time.
- **Efficient memory management** - The MapInfo Pro Advanced Application Programming Interfaces (APIs) provide intelligent data caching which automatically manage the use of system memory for the user. This automatic caching system simplifies the programmer's tasks when working with extremely large datasets.

Data Storage in MRR

An MRR dataset is stored in a single data file containing all raster data and metadata. The underlying storage format uses 64-bit file structures enabling raster datasets of virtually unlimited size to be created.

The MRR data format is more capable than virtually any other raster dataset currently available and it introduces several new concepts in the raster data structure.

1. Fields
2. Bands
3. Tiles
4. Null Value
5. Pyramids

1. Fields

In an MRR, data is stored in one or more fields, each of which consists of one or more data bands. There are four fundamental field types – Image, Image Palette, Classified and Continuous. A field can contain the following data types:

- **Image** - An Image field contains a single band of color data. MRR supports a wide variety of color data types including single-component and multi-component types such as 24-bit Red Green Blue (RGB).
- **Image Palette** - An Image Palette field uses a restricted color palette. The color palette can store color in any supported color type, just like an Image field. The number of entries in the color palette is determined by the user and is virtually unlimited. The MRR contains an integer index band that stores the color index for every cell.
- **Classified** - A classified field has an integer index band that stores the class index for each cell. An MRR can store many classification tables of varying sizes. The classification table can contain any number of data channels of any supported data type including integer and floating point numeric, date-time, color, strings and Binary Large Objects (BLOB).

- **Numeric (discrete)** - Contains bit (1/2/4) or numeric (integer, float, complex, date time) data bands. The data value for a cell represents the average value of the measured quantity over the cell region.
- **Numeric (continuous)** - A continuous field contains one or more data bands of any supported data type. Typically these bands will contain integer or floating point data, complex numbers, date-time or color data. The data type of each band can be different and each band can have a unique validity mask identifying whether the band value is valid or invalid in each cell.

There is no restriction on how many fields are stored in an MRR - nor on the type of the fields which are present. So it is possible, for example, to store both image data and numeric data within a single MRR file.

Remote sensing platforms return multi-spectral data in multiple resolutions, generally an integer multiple or fraction of the primary resolution. To support this every field in an MRR has its own cell - world transform definition to allow users to set a unique cell size for each field, if desired.

2. Bands

Each field may contain one or more data bands. Data bands may be ‘real’ indicating that the band data is actually stored in the file or ‘virtual’ indicating the data is obtained on the fly either from a ‘real’ band or from a classification table. In general, each band would have the same data type and all bands would be closely related. For example different frequencies of the electromagnetic spectrum from Landsat satellite data would be stored in multiple bands. Similarly, image data fields would store color in bands for each color (RGB) component.

3.Tiles

All data within an MRR is stored in tiles. The data in each tile may be compressed or uncompressed. Image Palette, classified, and continuous data can be stored using lossless compression such as LZ4, Zip (LZ77) or LZMA. Image data can be stored with either lossless compression, or lossy compression such as PNG and JPEG.

4. Null Value

The concept of a ‘null’ value has been advanced in the MRR format. A ‘null’ value can be stored for each field (and applies to all bands of that field) and records whether the value of a cell is valid or invalid. If the cell is invalid, then the value of the data stored in the ‘null’ cell can be used to classify why the cell is invalid.

5. Pyramids

The MRR consists of a data pyramid of overviews to enable efficient and high quality visualization. Each level of overview contains a representation of the data at a resolution twice lower than its previous level. You can read data from any level of the pyramid, but can only insert data into the base level.

Although a pyramid may increase the size of the dataset, it ensures visualization is of the highest quality.

MRR support in MapXtreme

MapXtreme is bundled with the newly added MIRasterHandlerRT.rhz handler which supports the display of MRR raster files created by MapInfo Pro 64-bit. This handler also supports the display of other raster/grid formats including GRD, GRC, TIF, BIL, BIP, BSQ, ERS, ASC, FLT and ADF, from a MapXtreme desktop or web application. The MIRasterHandlerRT.rhz handler is located by default in \Program Files\ Common\ MapInfo\ MapXtreme\ 9.x.x\ RasterGridHandlers.

The “MIRasterHandlerRT.rhz” raster handler is only supported on 64bit platforms, so to load and display MRR files your MapXtreme desktop or web application must be built as a 64 bit application. The raster handler operates in the same way as other raster handlers like SPOT.RHD, ECW5.RHD, TIFF.RHL, LeadTool.RHX (refer to MapXtreme DeveloperGuide.chm Raster Handlers section for additional information on Raster handlers).

All raster/grid files which are loaded by the MIRasterHandlerRT.rhz handler are treated by MapXtreme as “RASTER” image types, so the various “GRID” related MISql functions (please refer MISQL_Reference.chm) like MI_GridMinValue, MI_GridMaxValue and MI_GridValueAtPixel will not work with these files. All the “RASTER” related MI Sql functions like MI_ImageFile, MI_ImagePixelWidth, MI_ImagePixelHeight are fully supported.

On 64 bit platforms, MapXtreme will open Vertical Mapper .GRD and .GRC files using the MIRasterHandlerRT handler, so these formats will be supported as “RASTER” rather than “GRID” type files by MapXtreme. This will limit some of the grid related MISql functions that would otherwise be available on 32 bit platforms.

MIRasterHandlerRT.rhz supports various raster/grid files including MRR, GRD, GRC, TIF, BIL, BIP, BSQ, ERS, ASC, FLT and ADF. Refer to the table Supported Formats on page 356 in this chapter for a list of raster/grid formats supported by this handler.

The MIRasterHandlerRT.rhz handler can be configured in the same way as all the other supported raster handlers (refer Configuring Custom Raster Handlers). In addition to the MIRasterHandlerRT.rhz has some additional configuration settings which are stored in a configuration file called MIRasterPreferences.xml, which is located by default at \Program Files\ Common\ MapInfo\ MapXtreme\ 9.x.x\ RasterGridHandlers. If you wish to limit the MIRasterHandlerRT.rhz handler for a specific raster file format, then you can do so by setting the “RenderEnabled” property for that format to = false. For example, to prevent the MIRasterHandlerRT.rhz handler form loading GeoTif (.TIF) and ESRI ASCII Grid (.ASC) files you could set “RenderEnabled = false” as below;

```
<RasterFormats>
      <Driver Type="Native">
        <Formats>
         <format desc="Multi-Resolution Raster" ext="mrr" RenderEnabled="true"/>
          <format desc="VerticalMapper Grid" ext="grd" RenderEnabled="true"/>
          <format desc="VerticalMapper Classified Grid" ext="grc"
RenderEnabled="true"/>
          <format desc="GeoTiff Image" ext="tif" RenderEnabled="false"/>
          <format desc="ESRI ASCII Grid" ext="asc" RenderEnabled="false"/>
          <format desc="ESRI Float Grid" ext="flt" RenderEnabled="true"/>
        </Formats>
</Driver>
</RasterFormats>
```

The following sample code demonstrates how you might load an MRR raster file using MapXtreme.

C# example:

```
string mrrFile = “AvgFamilyIncome.TAB”;// tab file referring .MRR file
TableInfo tin = TableInfoRaster.CreateFromFile(mrrFile);
TableInfoRaster rasterTIN = tin as TableInfoRaster;
Table table = Session.Current.Catalog.OpenTable(rasterTIN);
// rest of the code
```

Configuring Custom Raster Handlers

You can configure your MapXtreme application to use a different raster handler than the ones included in the distribution of MapXtreme or to support an entirely new raster type. You can also change the precedence in which raster handlers are used.

In the default installation of MapXtreme, all raster handlers are placed in <*program files*>\Common Files\MapInfo\MapXtreme\9.x.x\RasterGridHandlers. This is also the location of the file mirasteru.dll. This is the recommended installation location of any other raster handlers that you use in your application. If you use the default location, no other configuration steps are necessary.

If you want to put your custom raster handler in a non-default location, then you must specify the location of your custom raster handler in an application configuration file for a desktop application, or in the Web.config file for a web application. To do this, define a <Path> or a <SpecialPath> element under <ApplicationDataPaths>, then copy the selected raster handler to that folder.

For example, to configure a desktop application custom raster handler in the non-default MyAppData directory, you could use the following .config file.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <section name="MapInfo.CoreEngine"
         type="MapInfo.Engine.ConfigSectionHandler, MapInfo.CoreEngine,
         Version=6.8.0.536, Culture=neutral,
         PublicKeyToken=93e298a0f6b95eb1" />
   </configSections>
   <MapInfo.CoreEngine>
      <ApplicationDataPaths>
         <SpecialPath>
            <Personal>MyAppData</Personal>
         </SpecialPath>
         <Path>c:\MyAppData</Path>
      </ApplicationDataPaths>
   </MapInfo.CoreEngine>
</configuration>
```

In this example, the <Personal> tag is a special place defined in the user's My Documents. This would refer to a folder called MyAppData in My Documents. Within the <ApplicationDataPaths> tag, use *either* the <SpecialPath> *or* the <Path> tags. Using both would mean that the raster handler could be put in either "MyAppData" folder. While this would not be an error, it probably is not what you intend to accomplish. Since this is a non-default configuration, you must also place the mirasteru.dll in that same directory.

Use the <SpecialPath> syntax if the application data is stored in a location relative to a .NET Framework special system folder. For example, if your application data is stored in a directory named MyAppData located under the “My Documents” directory, then the entry in the configuration file could be:

```
<Personal>MyAppData</Personal>
```

where “Personal” is the value of the .NET Framework enumeration Environment.SpecialFolder that represents the “My Documents” directory.

This configuration method can also be used to change the preferred raster handler for file types that can be managed by more than one raster handler. For example, a JPEG file can be handled by either Halo or LEADTOOLS (both of which are bundled in MapXtreme). Normally, Halo has priority because its *.rhv file extension alphabetically precedes the LEADTOOLS *.rhx extension. However, because MapXtreme first looks in any <ApplicationDataPaths> defined in the configuration file, it will locate a specified raster handler *before* looking for a handler in the default [CommonFiles] folder. So, for example, you could copy a LEADTOOLS *.RHX handler in the defined folder and configure MapXtreme to find and use that handler first.

Alternatively, you can rename file extensions in the [CommonFiles] folder so that the desired raster handler appears first in the alphabetic list. However, this will affect *all* applications developed with MapXtreme and may produce unintended side effects. Also, if file extensions are changed in this manner, the renamed raster handlers will *not* be deleted if MapXtreme is uninstalled. For these reasons, you may want to use the configuration method to change the location and precedence of raster handlers.

Grid Images

Grid images show an interpolation of data values across an area. A grid image is created from a data file in which data is measured at evenly-spaced points. The entire map area is converted to a grid in which each grid cell represents a value. Grid values don't have to be interpolated to produce a grid, although the data collection points need to be regularly spaced. MapXtreme and MapInfo Professional create grids by interpolating values using a grid handler.

In addition to the color gradations, grid images can also show hill, or relief, shading. Relief shading allows the grid surface to be shaded according to a virtual light source. The brightness of each grid cell corresponds to the light striking the surface and is adjusted based on its orientation to the light source. This is well suited for elevation grid maps where you can take surface slope and direction into account relative to the direction of the

light. Maximum brightness is assigned at points where the sun's rays are perpendicular to the surface. As the slope faces turn away from the light source, the brightness values are lower.

Supported grid formats include:

- MapInfo Grid (*.mig)
- USGS DEM (*.dem)
- GTOPO30 (*.dem)
- DTED (*.dt0, *.dt1, *.dt2)
- Vertical Mapper Continuous Grids[1] [2](*.grd, .*grc)

  Additional grid formats may be supported on a system if a custom grid handler is installed.

Grid Classes

The main classes for grid images are MapInfo.Raster.GridInfo, GridRead, GridCreatorFromFeatures, GridInflectionCalculator and HillShadeWrite. GridInfo and GridRead are like RasterInfo and RasterRead, in this case, provide the ability to get information about a grid file. Grids are created for a layer of data points using GridCreatorFromFeatures. GridInflectionCalculator is used to set inflection values and colors. HillShadeWrite allows you to add or alter relief shading to the grid. Many of the properties of a grid are inherited from RasterInfo, including the image class, coordinate system, raster control points and minimum bounding rectangle (MBR).

Code Sample: Adding a Grid Image to a Map

Adding a grid image to a map is the same as adding any other layer.

C# example:

```
Table MyTable = Session.Current.Catalog.OpenTable("MyGrid.tab");
FeatureLayer MyLayer = new FeatureLayer(MyTable);
MyMap.Layers.Add(MyLayer);
```

VB example:

```
Dim MyTable As Table = _ Session.Current.Catalog.OpenTable("MyGrid.tab")
Dim MyLayer As FeatureLayer =  New FeatureLayer(MyTable)
MyMap.Layers.Add(MyLayer)
```

1. Can be displayed as either grid or raster. The .TAB file determines whether the image should be drawn using the grid handler or the raster handler.
2. Not multi-thread safe.

Code Sample: Retrieving Data from a Grid Map

This example shows how to open and read information from a grid file.

VB example:

```
Public Shared Sub MapInfo_Raster_GridRead(ByVal strGridFilename _
As String)
   Dim strHillshadeFilename As String = _
   MapInfo.Raster.GridRead.DefaultHillshadeFilename(strGridFilename)
   Dim session As ISession

   session = MapInfo.Engine.Session.Current
   Dim gridread As GridRead = New GridRead(strGridFilename,_
       strHillshadeFilename)
   Console.WriteLine(gridread)
End Sub
```

This example shows how to open and read cell values from a grid file.

VB example:

```
Public Shared Sub MapInfo_Raster_GridReadStartRead(ByVal _
gridread As GridRead, ByVal strGridFilename As String)
   Dim strHillshadeFilename As String = _
      gridread.DefaultHillshadeFilename(strGridFilename)

   If gridread.StartRead() Then
      Dim x As Integer = 0   ' TODO  - set to a pixel column value
      Dim y As Integer = 0   ' TODO  - set to a pixel row value
      Dim bIsNull As Boolean
      Dim dValue As Double
      If gridread.GetValue(x, y, bIsNull, dValue) Then
         If bIsNull Then
            ' read a null cell
            Console.Write("{0,20}", "NULL")
         Else
            ' read a non-null cell, with value == dValue
            Console.Write("{0,20}", dValue)
         End If
      End If
      gridread.EndRead()
   End If
End Sub
```

Grid Creation

MapXtreme provides the ability to create continuous grids using a writable grid handler and interpolators. These grids are created programmatically using the MapInfo.Raster.GridCreatorFromFeatures class, the Mig.ghl grid handler and one of two supplied interpolators. Grids can be created from tables of data points or from selections.

Grids produced in MapXtreme are compatible with MapInfo Professional v 10.0.

A continuous grid is a map that is divided into a rectangular grid of cells, where each cell contains a data value representing either a measured data point, or an interpolated value based on the surrounding data points. The continuous grid displays the changing data values using a continuous gradation of color across the map.

Previously MapXtreme was limited to reading a grid and getting back a list of its grid cell values. It also could read information about a grid, such as hillshading and styles. The ContinuousGridWrite class was available, but lacked the ability to use an interpolator which would have created grids that better reflect its data points.

The MapInfo.Raster.GridCreatorFromFeatures is the main class to be called when creating a continuous grid. It uses the MapInfo writable grid handler and an interpolator to produce the continuous grid map.

The interpolators, IDW and TIN, provide algorithms for determining grid cell values according to their particular formulas. IDW is best used for population data, while TIN is used for terrain data. See Inverse Distance Weighted (IDW) Interpolator and Triangulated Irregular Network (TIN) Interpolator.

If one of the provided interpolators does not meet your needs, you can create your own by deriving it from the new IInterpolator interface. See IInterpolator Interface and the MapInfo.Raster.Interpolators namespace in the Developer Reference.

If your data includes coincident (duplicate) data points in the same cell location, the grid API provides an aggregator class to sum or average the points. Other available aggregator methods are count, min and max. You can also create or use any aggregator that implements IGridCellAggregator.

Other enhancements to the Grid API include the ability to clip the grid to match the outline of the map boundary. Grids are created based on the minimum bounding rectangle of the source data. If you wish your grid boundary to follow the outline of your map boundary, you would set the GridCreatorFromFeatures.ClippingGeometry property to the Geometry you wish the grid cells to be clipped against.

Once you have created a grid from your data points, you can modify characteristics such as hill shading, styles and inflection points using the new Grid Style dialog and Workspace Manager. See Grid Style for more information.

Grid Interpolators

MapXtreme provides two grid interpolators for creating a continuous grid: IDW and TIN. The IDW and TIN interpolators are included in a separate namespace called MapInfo.Raster.Interpolators.

In addition, MapXtreme provides an interface for creating your own interpolator.

Inverse Distance Weighted (IDW) Interpolator

The MapInfo.Raster.Interpolators.InverseDistanceWeighted interpolator class is one of two grid interpolators provided with MapXtreme. The IDW interpolator is best suited for data values such as population, or any data that yields arbitrary values over the grid rather than be related to or influenced by, neighboring values. This interpolation method also works well for sparse data.

The IDW interpolator calculates the values contained in the cells that cover the grid. Each data point value that is included in the calculation is weighted by its distance from the center of the cell. Because the interpolation is an inverse distance weighting calculation, the farther the point is from the cell, the less influence its value will have on the resulting cell value.

In IDW, an exponent determines how much influence each point will have on the result. The higher the exponent the greater the influence closer points will have on the cell value. Exponents can range from one to 10.

You can also choose an aggregation method for the values of source data points that are in the same grid cell. Choose from: average, count, sum, min, and max.

For a code example of creating a grid using the IDW interpolator, see MapInfo.Raster.GridCreatorFromFeatures class in the Developer Reference.

Triangulated Irregular Network (TIN) Interpolator

The second provided interpolator is called the Triangulated Irregular Network, or TIN. The TIN works best for terrain data and for data points that have a linear progression or relationship to each other across the grid, such as temperature.

The TIN interpolator produces triangles from a network of points that more closely reproduces the original map terrain than the IDW interpolator. It draws lines between points, dividing them into triangles and connecting all the points that it can. It creates a mesh of connectivity so that the grid points can be interpolated. The interpolation is not influenced by the neighboring original data values, so you do not get the "false bumping" of data that you can get with the IDW interpolator.

The TIN parameters (object properties) can be altered to give more or less detail to the map terrain. These properties include Tolerance (controls whether closely spaced points are discarded), Distance (controls the output), and FeatureAngle (controls the sharpness of the grid's edge).

For more information, see MapInfo.Raster.Interpolators.TriangulatedIrregularNetwork class in the Developer Reference.

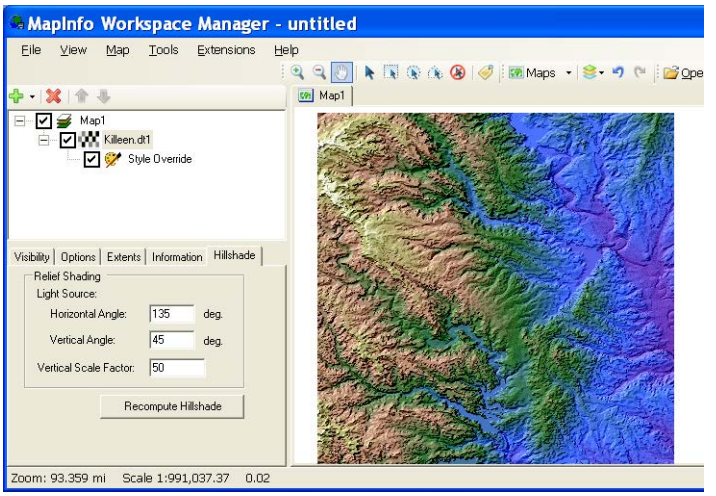
IInterpolator Interface

We also provide support for building your own interpolator through the IInterpolator interface. Derive your own interpolator from this interface to create grid cell values based on your algorithm. See the MapInfo.Raster.IInterpolator interface in the Developer Reference.

Grid Style

MapXtreme provides the ability to modify the inflection values and color of a continuous grid image. Previously MapXtreme could only read the information from the grid.

Grid style support is provided programmatically through the MapInfo.Styles.GridStyle class, via a GridStyleControl, and incorporated into Workspace Manager. A sample application called GridForm also provides this capability. Changes to inflection values and colors can be persisted to a workspace and loaded at a later time.



Grid Images and Inflections

A continuous grid image shows an interpolation of data values across an area. A grid is made up of grid cells where each cell represents a value. These values are represented on the map as a continuous color gradation that is bounded by the range of data values.

The point at which the color changes due to a change in the grid value or percentage is known as an inflection. MapXtreme creates a GridInflectionCalculator object that calculates inflection values based on one of several supported methods. The number of inflection points for a grid coupled with the minimum and maximum range of data values is used to determine the color gradation across the map. The color that represents the inflection is applied to the grid as a FeatureOverrideStyleModifier.

Inflection Methods

MapXtreme supports the following calculation methods:

- *Equal Cell Count*–Calculates inflection values so that an equal number of cell values are within the calculated percentage ranges.
- *Equal Value Ranges*–Calculates inflection values in a manner where each inflection value range is the same size.
- *Custom Cell Count*–Same as Equal Cell Count, except uses custom percentage ranges.
- *Custom Value Ranges*–Same as Equal Value Ranges, except uses custom inflection values.

Inflection values and colors can be modified programmatically through the MapInfo.Raster.GridInflectionCalculator class. A GridStyleControl is available from the MapXtreme toolbox in Visual Studio.

This process is also incorporated into the Grid Style dialog in Workspace Manager. See Grid Style.

ⓘ If you change either the values or percentages while the inflection methods are set to Equal Value Ranges or Equal Cell Count, the methods become Custom Value Range and Custom Cell Count, respectively.

Changes to inflection values and colors can be persisted to a workspace and loaded at a later time.

Calculating Inflection Values and Colors for a Grid Layer

You can calculate inflection values and colors programmatically or through the Grid Style dialog in Workspace Manager. The GridForm sample application also provides this capability.

GridInflectionCalculator Class

The following code example shows how to create an initial set of values and colors and then modify them. This code example is provided in the Developer Reference API documentation under the MapInfo.Raster.GridInflectionCalculator class.

```
public static void MapInfo_Raster_GridInflectionCalculator(TableInfoGrid
tableInfoGrid)
{
    // Create 10 grid inflection values, by using the EqualRangeValues method,
using colors from orange to red.
    GridInflectionCalculator gridInflectionCalculator = new
GridInflectionCalculator(tableInfoGrid, InflectionMethod.EqualRangeValues, 10,
Color.Orange, Color.Red);

    // Modify the colors to go from green to brown.
    gridInflectionCalculator.CalculateStyles(Color.Green, Color.Brown);

    // Calculate 5 grid inflection values, by using the EqualRangeValues method.
    gridInflectionCalculator.CalculateValues(InflectionMethod.EqualRangeValues,
5);

}
```

Relief Shading

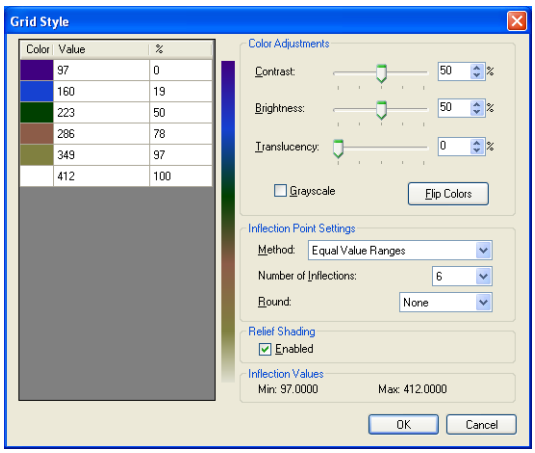
In addition to the color gradations, grid images can also show hill, or relief, shading. Relief shading allows the grid surface to be shaded according to a virtual light source. The brightness of each grid cell corresponds to the light striking the surface and is adjusted based on its orientation to the light source. This is well suited for elevation grid maps where you can take surface slope and direction into account relative to the direction of the light. Maximum brightness is assigned at points where the sun's rays are perpendicular to the surface. As the slope faces turn away from the light source, the brightness values are lower.

Hillshading is enabled via a checkbox in the GridStyle dialog. The horizontal and vertical light source angles and scale factor are set in LayerControl. Programmatically, hillshading can be added to a grid via the MapInfo.Raster.HillshadeWrite class.

Grid Style Dialog

The Grid Style dialog is where you can modify the style settings of a grid, including color, inflection value, contrast, brightness, translucency, and more through a graphical user interface (GUI). The style is applied to the grid layer as a FeatureOverrideStyleModifier.

To reach the Grid Style dialog, open a grid image in Workspace Manager. Select the grid layer and right-click to display the Add Style Override menu item. In the Visibility tab, click the Grid Image Style button to display the dialog. Each component of the Grid Style dialog is explained below.



Color/Value/% (Percentage)

The Color/Value/% (Percent) grid box shows the current settings for inflection colors, values and percentage.

To change the color, double-click on the color swatch and choose a new color from the Color dialog palette.

The value and % settings come from the data in the grid. These values can change when you modify settings in the Inflection Point Settings group of this dialog. That will cause a re-calculation of the inflections and display new values. For example, you can edit the values directly when you choose Custom Value Ranges and the percentage when you choose Custom Cell Count.

ⓘ If you edit either the values or percentages while the inflection methods are set to Equal Value Ranges or Equal Cell Count, the methods become Custom Value Range and Custom Cell Count, respectively.

Color Adjustments

You can make adjustments to affect the color in the grid image. These changes are applied equally to the entire layer.

Contrast
Use the Contrast scroll bar to adjust the contrast in the image. Slide the bar between 0 and 100% to set the grid's contrast level. You can use the cursor keys for fine adjustment.

Brightness
Use the Brightness scroll bar to adjust the brightness in the image. Slide the bar between 0 and 100% to set the grid's brightness level. You can use the cursor keys for fine adjustment.

Translucency
Use the Translucency slide bar to adjust the translucency for the image. Translucency can be set between 0 and 100%. An image with 0% translucency is completely opaque (cannot be seen through). An image with 100% translucency is completely transparent (completely invisible).

Gray scale
Check to show only gray scale colors in the grid.

Flip Colors Button
Click this button to invert the inflection colors. This affects only the beginning and ending colors at this time.

Inflection Point Settings

The inflection point settings allow you to set the method used to calculate inflections to show how your data is distributed across an area. Equal Cell count and Custom Cell Count are based on percentages. Equal Value Ranges and Custom Value Ranges are based on values.

Equal Cell Count
Sets the inflections so that approximately an equal number of grid cells falls within the calculated percentage ranges.

Equal Value Ranges
Spreads the inflections between the minimum and maximum of the source data range, where each inflection value range is the same size.

Custom Cell Count
Choose this method when you want to use your own percentages and not ones that are evenly spread.

Custom Value Ranges
Choose this method when you want to use your own values and not ones that are evenly spread.

Number of Inflections:
Select from a list of 2 to 16 inflections or type a number between 2 and 255.

Round:
Choose from a list of rounding factors to be applied to inflection values. You may not see the effects of this rounding if the spread method is based on cell count until inflection values are actually calculated.

Relief Shading

Relief shading allows you to shade your grid surface map according to the orientation of a virtual light source. This enables you to take surface slope and direction into account relative to the direction of the light. The Grid Style dialog provides a setting to enable or disable relief shading.

To adjust the relief shading settings, you must first highlight the grid layer in Layer Control and choose the Hillshade tab. Relief shading is not applied as a style override, as is the inflection settings.

Light Source

The light source settings for horizontal, vertical angle and vertical scale are set in the Layer Control dialog, under the Hillshade tab. This tab displays only when a grid layer is selected and hillshading is enabled.

Horizontal Angle
Rotates the light source in the horizontal plane. Zero degrees corresponds to the light source shining from due East. Positive angles rotate the light source counterclockwise, so, for example, 90 degrees places the light source due North.

Vertical Angle
Rotates the light source in the vertical plane. Zero degrees places the light source at the horizon, and 90 degrees places it directly overhead. Specify angles between 180 and 360 degrees to have the light source originate from under the surface.

Vertical Scale Factor
Specify a scale factor between 0 and 100. Increasing this scale factor exaggerates the surface vertically, enhancing the shading effect. This can be useful for enhancing detail in relatively flat surfaces.

Recompute Hillshade
Click this button to recompute the hillshade after you make adjustments to the light source settings.

Inflection Values

These values show the minimum and maximum values in the grid table. These values are not editable.

Min:
Shows the minimum data point value in the source table.

Max:
Shows the maximum data point value in the source table.

GridInfoForm Sample Application

MapXtreme provides C# sample applications that incorporate the GridInflectionCalculator. Find these samples under the \Desktop\Features\GridinfoForm under the \Samples\VisualStudio20xx folders.

18

18 – Working with Maps from Tile Servers

MapXtreme supports the display of Tile Server TAB files created by MapInfo Professional. This allows you to open and view map tiles, such as Bing Maps and Spatial Server Tile Maps, from a MapXtreme desktop or web application.

In this chapter:

Tile Server Images

Tiles are portions of maps that are stored on tile servers and on request, delivered as images to the client application for display. MapXtreme opens and displays tile images in the same way that it supports raster or WMS images. Images can be brought into MapXtreme as individual layers or as part of an .MWS workspace. Tiles opened as a layer are added as the bottommost layer to the map control.

Images from tile servers make excellent backgrounds for maps. For example, aerial photographs that show real-world detail such as buildings, refineries, and vegetation are well-suited as base layers for a map. Use a tile server image as a base layer in your MapXtreme web or desktop application. Overlay vector data such as street networks, point locations representing customers, and postal boundaries, to create useful and visually appealing maps.

Tile Server files from MapInfo Professional consist of a TAB file and associated XML file that contains the URL to the tile server. The TAB file provides the coordinate system information MapXtreme needs to display the layer accurately and in line with vector data you are likely to have in your map.

MapXtreme supports two types of tile servers: QuadKey and LevelRowColumn. See Using Tile Server Images on page 379.

MapXtreme has built-in caching support for tiles for faster access of tile images. The cache is maintained during the session that the desktop or web application is open. See Tile Caching on page 378.

You can also apply style overrides to Tile Server layers to control the translucency, brightness, contract and grayscale of the images, in the same way as you do for raster layers. See Raster Classes on page 354.

Tile Caching

MapXtreme caches the tiles of the referred tile server onto disk and performs the rendering. If tiles are needed to be drawn on a map control, typically due to zooming and panning, and are not available in the cache, they are requested from the server and added to the cache. The TileServer cache will persist on disk until you close the MapXtreme-based desktop or web application.

TileSever cache is stored in MapXtreme folder under TEMP or TMP directory by default.

Map Behavior with a Tile Server Layer

The coordinate system for the Tile Server layer is defined in the TAB file, which is not changeable at render time. When you display a tile server image as a map base layer, MapXtreme forces the already visible layers (vector, raster, WMS, etc.) to be drawn in the same projection as the Tile Server layer. The coordinate system of the map is also changed to match the Tile Server layer. If other raster layers are present in the map, raster re-projection is not allowed until the Tile Server layer is visible. To enable raster re-projection for other raster layers, you must remove the visible Tile Server layers.

If a map includes more than one Tile Server layer, MapXtreme automatically displays the map in the projection specified by the most visible tile server image. A newly added Tile Server layer will be added on top of existing Tile Server layers.

For better visibility and smooth text rendering, by default the map's anti-aliasing and translucency properties are set to true whenever any Tile Server layer is being added. These settings persist until you manually disable these properties.

Using Tile Server Images

Tile server images are a very popular way to bring more information into your map. They are more efficient than WMS since only the tiles that cover the map view at the current zoom and center are returned.

MapXtreme supports four types of tile servers: QuadKey, LevelRowColumn, WMTS and Custom Resolution Tile Service.

QuadKey

QuadKey is a mechanism that Bing Maps Tile System uses to uniquely identify its tiles at a particular level. Images from Bing include the attribute QuadKey in the URL. In order to access Bing Maps, you must acquire a Bing Key from Microsoft. See http://msdn.microsoft.com/en-us/library/ff428642.aspx. See Using TableInfoTileServer Class on page 391 for more on how to set the key.

MapXtreme also supports the localized Bing Maps.

MapXtreme provides sample TAB /XML files for three types of Bing Maps: Aerial, Road and Hybrid.

The XML file should be in conformance with the TileServerSchema.xsd available at the installation path “C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x”. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<TileServerInfo Type="QuadKey">
  <Url>http://ecn.t3.tiles.virtualearth.net/tiles/a{QUADKEY}.jpg?g=392</Url>
  <MaxLevel>23</MaxLevel>
  <TileSize Height="256" />
  <AttributionText Font="Font (&amp;quot;Tahoma&amp;quot;,257,8,16777215,0)">Microsoft
Bing © 2011 Microsoft Corporation</AttributionText>
</TileServerInfo>
```

Note that the value of Type attribute of the TileServerInfo element is "QuadKey".

LevelRowColumn

LevelRowColumn is another way to uniquely identify tiles. The level, row and column is expressed as part of the URL.

The XML file should be in conformance with the TileServerSchema.xsd available at the installation path “C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x”. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<TileServerInfo Type="LevelRowColumn">
  <Url>http://tile.openstreetmap.org/{LEVEL}/{ROW}/{COL}.png</Url>
  <MinLevel>0</MinLevel>
  <MaxLevel>18</MaxLevel>
  <TileSize Height="256" Width="256" />
  <AttributionText Font="Font (&amp;quot;Tahoma&amp;quot;,257,8,16777215,0)">© 2011
OpenStreetMap contributors, CC-BY-SA</AttributionText>
</TileServerInfo>.
```

Note that the value of Type attribute of the TileServerInfo is "LevelRowColumn".

WMTS (Web Map Tile Service)

Web Map Tile Service (WMTS) is an OGC standard that provides guidelines for a web service to host pre-rendered map tiles of spatially referenced data using tile images with predefined content, extent, and resolution.

MapXtreme supports consumption of such a service via Tile server TAB/XML files, through which WMTS version, CapabilitiesUrl and GetTilesUrl can be specified.

The XML file should be in conformance with the WmtsServerSchema.xsd available at the installation path “C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x”. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<TileServerInfo Type="WMTS">

<CapabilitiesUrl>http://sampleserver6.arcgisonline.com/arcgis/rest/services/Wor
```

```
ldTimeZones/MapServer/WMTS?request=GetCapabilities&amp;service=WMTS&amp;version
=1.0.0</CapabilitiesUrl>
  <Version>1.0.0</Version>
<GetTilesUrl>http://sampleserver6.arcgisonline.com/arcgis/rest/services/WorldTi
meZones/MapServer/WMTS?&amp;service=WMTS&amp;request=GetTile&amp;version=1.0.0&
amp;layer=WorldTimeZones&amp;style=default&amp;format=image/png&amp;tileMatrixS
et=default028mm&amp;tileMatrix={TileMatrix}&amp;TileRow={TileRow}&amp;TileCol={
TileCol}</GetTilesUrl>
</TileServerInfo>
```

Note that the value of Type attribute of the TileServerInfo is "WMTS".

MapXtreme supports following tile server types via Tile server TAB files created by MapInfo Professional:

- MapInfo Pro
- Spectrum Spatial
- ArcGIS
- OSM
- MapQuest

Custom Resolution Tile Service

MapXtreme.NET supports consumption of tile servers which provide tiles of custom resolution at different zoom levels. Unlike standard tile servers, these tile servers provide a level-wise list of resolutions, which do not necessarily differ by a factor of two with increasing zoom level.

Like other tile servers, this type of tile server can also be consumed in MapXtreme either by a combination of .TAB/.XML files or via APIs.

A new schema “CustomTileServerSchema.xsd” has been introduced to specify a set of rules for the .XML file and is available at common files path.

“C:\Program Files\Common Files\MapInfo\MapXtreme\9.2”.

An example .xml is given below:

```
<?xml version="1.0" encoding="utf-8"?>
<TileServerInfo Type="CustomLevelRowColumn">
   <Url>https://someserver/{LEVEL}/{COL}/{ROW}</Url>
   <MinLevel>0</MinLevel>
   <MaxLevel>9</MaxLevel>
   <Resolutions>
      <Resolution Level="0">41.015625</Resolution>
      <Resolution Level="1">20.5078125</Resolution>
      <Resolution Level="2">10.25390625</Resolution>
      <Resolution Level="3">5.126953125</Resolution>
```

```
      <Resolution Level="4">2.5634765625</Resolution>
      <Resolution Level="5">1.28173828125</Resolution>
      <Resolution Level="6">0.640869140625</Resolution>
      <Resolution Level="7">0.3204345703125</Resolution>
      <Resolution Level="8">0.16021728515625</Resolution>
      <Resolution Level="9">0.080108642578125</Resolution>
   </Resolutions>
   <Origin>NW</Origin>
<TileOrigin X="-48683.0" Y="6708835.0"/>
</TileServerInfo>
```

MapXtreme supports consumptions of custom tile servers with both top-left and bottom-left origins.

ⓘ Note: As of now, implementation of consuming custom tile servers, supports only those servers which supply map tiles of 256X256 pixels.

For more details, please refer “MapXtreme_DeveloperReference.chm”.

Consuming Tile Layers via APIs (without .tab/.xml file)

MapXtreme.Net now supports opening all supported types of tile layers via APIs. Now no need to keep the .TAB/.XML file handy when opening tile layer. User just has to provide the properties of the tile servers in an object and use that object to open layer. Code examples are provided later in this document. For consuming tile layer without .tab/.xml files, user need to create object of “TileServerTableInfo” and fill appropriate properties of this object and then use this table info to open the table.

Details about the properties in “TileServerTableInfo”:

Properties	**Descriptions**
AttributionFont	Specifies the font to be used to display attribution text of the tiles supplied by tile Server. This property is applicable to all tile servers other than TileServerType.WMTS and is optional. Font should be specified as a string. For example, "Font ("Tahoma",257,8,16777215,0)".
Coordsys	Specifies the Coordsys to be used. This property is applicable to all tile servers and is mandatory. Create a coorsys using coordsys factory of your choosing and set it in this property.
CustomResolutions	Specifies the CustomResolutions supported by the CustomLevelRowColumn tile Server. It is list of double values. This property is applicable to all TileServerType.CustomLevelRowColumn type tile servers and is mandatory.
CustomTileOrigins	Specifies the CustomTileOrigins supported by the CustomLevelRowColumn tile Server. It is list of DPoints. This property is applicable to all TileServerType.CustomLevelRowColumn type tile servers and is mandatory.
Extents	Specifies the Extents of the layer of the tile server. This property is applicable to all tile servers and is Optional. Extents can also be set via CoordSys property. If not defined, it will take default extents of the selected Coordsys. To specify extent manually, create a DRect with bounds and set it in this property.

Properties	Descriptions
Extents	Specifies the Extents of the layer of the tile server. This property is applicable to all tile servers and is Optional. Extents can also be set via CoordSys property. If not defined, it will take default extents of the selected Coordsys. To specify extent manually, create a DRect with bounds and set it in this property.
IgnoreExceptions	Specifies whether to ignore exceptions from tileserver or not. This property is applicable to all tile servers and is optional.
MaxLevel	Specifies the max level supported by the tile Server. This property is applicable to all tile servers but is optional for TileServerType.WMTS and mandatory for rest.
MinLevel	Specifies the min level supported by the tile Server. This property is applicable to all tile servers and is optional.
Origin	Specifies the origin of the tiles supplied by tile Server. This property is applicable to all tile servers other than TileServerType.WMTS and is optional. It's possible values are NW/SW only. Default NW is used.
OverrideCoordinateOrder	Specifies whether to flip coordinates of the tilematrix sets or not, if needed.
ReadTimeout	Specifies the read timeout in seconds. This property is applicable to all tile servers and is optional. By default, implementation uses 60 seconds for read timeout.
RequestTimeout	Specifies the request timeout in seconds. This property is applicable to all tile servers and is optional. By default, implementation uses 60 seconds for read timeout.

Properties	Descriptions
StartTileNum	Specifies the Start tile number of the tiles supplied by tile Server. This property is applicable to all tile servers other than TileServerType.WMTS and is optional.
TileHeight	Specifies the Height in pixel of the tiles supplied by tile Server. This property is applicable to all tile servers other than TileServerType.WMTS and is mandatory.
TileWidth	Specifies the Width in pixel of the tiles supplied by tile Server. This property is applicable to all tile servers other than TileServerType.WMTS and is optional.
TileServerUrl	Specifies the URL of the tile Server. This property is applicable to all tile servers other than TileServerType.WMTS is mandatory.
TileServerInfoType	Determines the type of the Tile Server. For different tileserver types, different properties are applicable. This property is mandatory and must be set before calling OpenTable. For supported types, see “TileServerInfoType” enum.
WMTSCapablitiesUrl	Specifies the capabilities URL of the WMTS tile Server. This property is only applicable for TileServerType.WMTS is mandatory.
WMTSGetTilesUrl	Specifies the GetTiles URL of the WMTS tile Server. This property is only applicable for TileServerType.WMTS and is mandatory.
WMTSVersion	Specifies the Version of the WMTS tile Server. This property is only applicable for TileServerType.WMTS and is optional. Default Value for this property is "1.0.0".

TileServerType Enumeration

Enum for different types of TileServers Supported by MapXtreme.

Invalid	Default value of the Enum.
Bing	This type is used to specify that server being used is of Bing type. Bing Maps Platform (previously Microsoft Virtual Earth) is a geospatial mapping platform produced by Microsoft.
QuadKey	This type is used to specify that server being used is of Quadkey type.
LevelRowColumn	This type is used to specify that server being used is of Level Row Column type.
WMTS	This type is used to specify that server being used is of WMTS type. A Web Map Tile Service (WMTS) is a standard protocol for serving pre-rendered georeferenced map tiles over the Internet.
CyberJapan	This type is used to specify that server being used is of Level Row Column Japanese type.
CustomLevelRowColumn	This type is used to specify that server being used is of Level Row Column type and supports custom resolutions at each level.

For more details, please refer “MapXtreme_DeveloperReference.chm”.

Sample Code Snippets

Opening Tile Server via APIs

You can open a normal Tile Server using the example below.

```
//Create object of TableInfoTileServer by passing alias for table.
TableInfoTileServer ti = new TableInfoTileServer("BingAerial");

//Specify the tile server Properties.
//ti.TileServerInfoType, ti.TileServerUrl, ti.MaxLevel, ti.TileHeight,
ti.Coordsys properties are mandatory.
```

```
ti.TileServerInfoType = TileServerType.Bing;
ti.TileServerUrl = "http://someserver/tiles/a{QUADKEY}.jpg?g=392";
ti.MaxLevel = 23;
ti.TileHeight = 256;
ti.AttributionFont = "Font (&quot;Tahoma&quot;,257,8,16777215,0)";

//Create the coordsys
CoordSysFactory factory = Session.Current.CoordSysFactory;
CoordSys coordSys = factory.CreateFromMapBasicString("CoordSys Earth
Projection 10, 157, \"m\", 0");
//Set the coordsys in tableInfo object.
ti.Coordsys = coordSys;
ti.Extents = new DRect(-20037508.3428, -20037508.343, 20037508.3428,
20037508.343);

//Open the table, If any mandatory property is missing for a particular type
of TS,
//a relevant execption will be thrown here.
Table tileTable = Session.Current.Catalog.OpenTable(ti);


//Do other things with opened table. For example, add the table to the current
map and load it in mapcontrol
//MapControl.Map.Load(new MapTableLoader(tileTable));
```

Opening WMTS via APIs

You can open a WMTS layer using the example below.

```
//Create object of TableInfoTileServer by passing alias for table.
TableInfoTileServer ti = new TableInfoTileServer("wmts");

// Specify the Web Map tile service(WMTS) Properties.
//ti.TileServerInfoType, ti.WMTSCapablitiesUrl, ti.WMTSGetTilesUrl,
ti.WMTSVersion,
//t.Coordsys properties are mandatory.
ti.TileServerInfoType = TileServerType.WMTS;
ti.WMTSCapablitiesUrl="http://someserver/WMTService.aspx?request=GetCapabilit
ies&service=WMTS&version=1.0.0";
ti.WMTSGetTilesUrl="http://someserver?&service=WMTS&request=GetTile&version=1
.0.0&layer=orto&style=default&format=image/png&tileMatrixSet=someTileMatrixSe
t&tileMatrix={TileMatrix}&TileRow={TileRow}&TileCol={TileCol}";
ti.WMTSVersion = "1.0.0";

//Create the coordsys
CoordSysFactory factory = Session.Current.CoordSysFactory;
CoordSys coordSys = factory.CreateFromMapBasicString("CoordSys Earth
Projection 1, 115");

//Set the coordsys in tableInfo object.
ti.Coordsys = coordSys;
```

```
ti.Extents = new DRect(51.8749637851, 6.36653951224, 61.3062175074,
11.0821663734);

//Open the table, If any mandatory property is missing for a particular type
of TS,
//a relevant execption will be thrown here.
Table tileTable = Session.Current.Catalog.OpenTable(ti);

//Do other things with opened table. For example,
//Add the table to the current map and load it in mapcontrol
//MapControl.Map.Load(new MapTableLoader(tileTable));
```

Opening Custom Tile Server via APIs

You can open a Custom Tile Server using the example below.

```
//Create object of TableInfoTileServer by passing alias for table.
TableInfoTileServer ti = new TableInfoTileServer("CustomTileServer");

// Specify the CustomLevelRowColumn tile server Properties.
//ti.TileServerInfoType, ti.TileServerUrl, ti.MaxLevel, ti.CustomResolutions,
//ti.CustomTileOrigins, t.Coordsys properties are mandatory.
ti.TileServerInfoType = TileServerType.CustomLevelRowColumn;
ti.TileServerUrl="http://someserver/tile/{LEVEL}/{ROW}/{COL}";
ti.MaxLevel = 10;

//set the resolutions in CustomResolutions
   List<double> resolution = new List<double>();
   resolution.Add(169.33367200067735);
   resolution.Add(84.66683600033868);
   resolution.Add(42.33341800016934);
   resolution.Add(21.16670900008467);
   resolution.Add(10.583354500042335);
   resolution.Add(5.291677250021167);
   resolution.Add(2.6458386250105836);
   resolution.Add(1.3229193125052918);
   resolution.Add(0.6614596562526459);
   resolution.Add(0.33072982812632296);
   resolution.Add(0.19843789687579377);
ti.CustomResolutions = resolution;

//set the TileOrigins\TileOrigin in CustomTileOrigins
List<Tuple<double, double>> tileOrigins = new List<Tuple<double, double>>();
tileOrigins.Add(new Tuple<double, double>(-5220400.0, 4470200.0));
ti.CustomTileOrigins = tileOrigins;

//Create the coordsys
CoordSysFactory factory = Session.Current.CoordSysFactory;
CoordSys coordSys = factory.CreateFromMapBasicString("CoordSys Earth
Projection 8,79,7,-2,49,0.9996012717,400000,-100000");
ti.Coordsys = coordSys;
```

```
//Set the coordsys in tableInfo object.
ti.Coordsys = coordSys;
ti.Extents = new DRect(481634.4445551128, 60383.01832330043,
703156.6809428951, 236630.6370551856);

//Open the table, If any mandatory property is missing for a particular type
of TS,
//a relevant execption will be thrown here.
Table tileTable = Session.Current.Catalog.OpenTable(ti);
//Do other things with opened table. For example,
//Add the table to the current map and load it in mapcontrol
//MapControl.Map.Load(new MapTableLoader(tileTable));
```

Authentication to Tile Server

Depending on the tile server (Quadkey or LevelRowColunm) being referred by the TAB file, MapXteme may need to provide credentials to the tile server. In case of the QuadKey type tile server like Bing Tile Map (Aerial, Roads and Hybrid), a Bing key is required to setup the authentication. The Bing Key can be specified to MapXtreme-based application as described in Tile Server Settings on page 389.

If any tile server referred by the TAB file requires user credentials (user & password), then for a desktop application the authentication dialog will be presented to the user.

Tile Server Settings

Before loading a tile server TAB or workspace into a map control, several settings must be configured, either at the system level or programmatically through the `MapInfo.Engine.TileServerSettings` class. Tile Server settings will be set for the entire MapXtreme application instance.

License Key for Bing Maps

Use the Registry key variable "`SOFTWARE\MapInfo\MapXtreme\9.x.x\TileServerKey`" to set the tile server key required by TileServer TAB file in the workspace.

Use the system environment path variable `MI_TILE_SERVER_KEY` to set tile server key required by TileServer TAB file in the workspace.

A trial Bing Tile Server key can be obtained from Microsoft. For more information, visit http://msdn.microsoft.com/en-us/library/ff428642.aspx.

Via the Web or Desktop Configuration File

For a MapXtreme web application and/or standalone application you can set the tile server settings in Web.Config and/or app.config file as follow:

```
<add key="MapInfo.Engine.Session.Workspace.TileServerSettings"
value="ReadTimeOut:60; RequestTimeOut:60; TileServerKey:XXXXXXXXXXXXXXXXX" />
```

Via MapInfo.Engine.TileServerSettings Class

MapXtreme's API provides the `MapInfo.Engine.TileServerSettings` class to set settings for tile server (See Using TableInfoTileServer Class on page 391). By default MapXtreme has following values for tile server settings:

`MapInfo.Engine.TileServerSettings.ReadTimeOut`	60 seconds
`MapInfo.Engine.TileServerSettings.RequestTimeOut`	60 seconds
`MapInfo.Engine.TileServerSettings.Key`	"" (empty string)

ⓘ Note: These setting override existing settings and will be used until they are changed.

Sample Code for TileServerSettings class

`MapInfo.Engine.TileServerSettings` class can be used to set the values as following code snippet

C# example:

```
public static void MapInfoSetTileServerSettings()
     {
            MapInfo.Engine.TileServerSettings tssetting = new
TileServerSettings();
            tssetting.Key = "";
            tssetting.ReadTimeout = 70;
            tssetting.RequestTimeout = 70;
            Session.Current.TileServerSettings = tssetting;
            //Table and/or workspace opening code, for example, here.
        }
```

VB example:

```
Public Shared Sub MapInfoSetTileServerSettings()
```

```
    Dim tssetting As MapInfo.Engine.TileServerSettings = New TileServerSettings
    tssetting.Key = ""
    tssetting.ReadTimeout = 70
    tssetting.RequestTimeout = 70
    Session.Current.TileServerSettings = tssetting
    'Table and/or workspace opening code, for example, here.
End Sub
```

Using TableInfoTileServer Class

MapInfo.Data.TableInfoTileServer class can be used to open tile server tab file.

C# example:

```
public static void MapInfoDataTableInfoTileServer(Map map)
  {
       TableInfo tableInfo = TableInfo.CreateFromFile("BingAerial.tab");
 TableInfoTileServer tableInfoTileServer = tableInfo as TableInfoTileServer;
      tableInfoTileServer.PreferredHandler = "leadtool.rhx";
     Table table = Session.Current.Catalog.OpenTable(tableInfoTileServer);
     try {
        map.Load(new MapTableLoader(table));
     }catch (Exception ex){
        String str = ex.ToString();
        // user code
     }
 }
```

VB example:

```
Public Shared Sub MapInfoDataTableInfoTileServer(ByVal map As Map)
    Dim tableInfo As TableInfo = tableInfo.CreateFromFile("BingAerial.tab")
    Dim tableInfoTileServer As TableInfoTileServer = CType(tableInfo,
TableInfoTileServer)
    tableInfoTileServer.PreferredHandler = "leadtool.rhx"
    Dim table As Table = Session.Current.Catalog.OpenTable(tableInfoTileServer)
    Try
        map.Load(New MapTableLoader(table))
    Catch e As Exception
        Console.WriteLine("Exception caught : {0}", e)
    End Try
    'user code
End Sub
```

Tile Server Sample Application

A Tile Server sample (C#) application for desktop is available to help you gain more insight on how to use the tiling capability. It is located in your MapXtreme install directory as noted below for 32-bit and 64-bit installations:

C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio2015\Desktop\Features\TileServer

C:\Program Files (x86)\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio2015\Desktop\Features\TileServer

C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio2017\Desktop\Features\TileServer

C:\Program Files (x86)\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio2017\Desktop\Features\TileServer

19

19 – Working with GeoPackage

MapXtreme provides support for opening, displaying, creating and editing GeoPackage files which is an open source format created by OGC.

In this chapter:

Overview

GeoPackage is an open format for Geospatial Information defined by OGC (http://www.geopackage.org). It is a new SQLite-based extension defined by the OGC to promote portability of data across platforms and products.

According to OGC definition for GeoPackage:

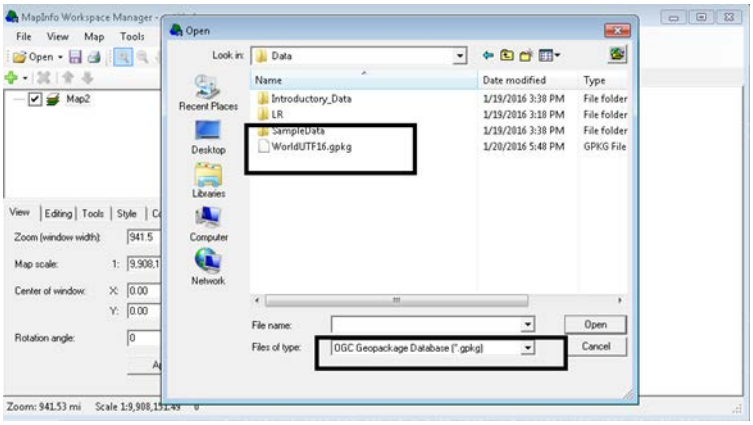
"*A GeoPackage is a platform-independent SQLite database file that may contain:*

- *vector geospatial features*
- *tile matrix sets of imagery and raster maps at various scales*
- *metadata*

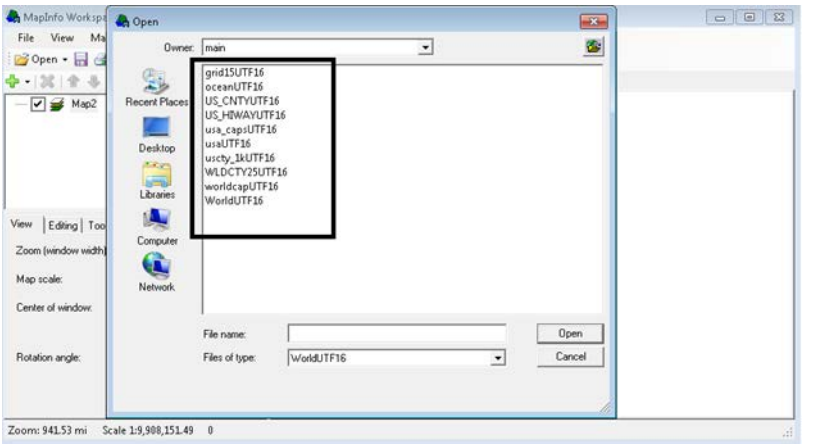
Since a GeoPackage is a database, it supports direct use, meaning that its data can be accessed and updated in a "native" storage format without intermediate format translations. GeoPackages are interoperable across all enterprise and personal computing environments, and are particularly useful on mobile devices like cell phones and tablets in communications environments with limited connectivity and bandwidth. This OGC® Encoding Standard defines the schema for a GeoPackage, including table definitions, integrity assertions, format limitations, and content constraints. The allowable content of a GeoPackage is entirely defined in this specification."

Opening a GeoPackage file

GeoPackage table and GeoPackage Tab file can be opened using Workspace Manager. GeoPackage file can be opened from File Menu by selecting filter as below:



After open it will show the list of tables in the database and particular table can be selected to open.



Opening a GeoPackage file has been added to OpenFileDlg and will be available in all those application which make use of OpenFileDlg of MapXtreme (v8.1 and above).

ⓘ If the GeoPackage table being opened have a Coordinate System which is not supported by MapXtreme then that table will not be opened.

To open a GeoPackage file programmatically use the code below:

```
TableInfoGeopackage infoGeoPackage = new TableInfoGeopackage("table alias");
infoGeoPackage.DatabasePath = tablePath; //Path to the GeoPackage file .gpkg
infoGeoPackage.DatabaseTableName = "Table Name"; //Table to be opened in
GeoPackage
TableInfo tableInfo = infoGeoPackage;//Create the Table Info object.
Table
openTable=MapInfo.Engine.Session.Current.Catalog.OpenTable(tableInfo);//Open
table
```

A GeoPackage database may contain multiple tables so it is required to mention the name of the table to be opened in the database.

ⓘ MapXtreme (v8.1 and above) only supports Feature Tables of GeoPackage.

To return the list of tables stored in the GeoPackage database below piece of code can be used:

```
GeopackageDataSourceDefinition gpkgDSDef = new
GeopackageDataSourceDefinition("c:\data\test.gpkg"); //path to gpkg file
```

```
GeopackageDataSource gpkgDS =
GeopackageDataProvider.Instance.OpenDataSource(gpkgDSDef, null) as
GeopackageDataSource;
List<IDataSourceNamedTable> tablesList = gpkgDS.GetSchemaNamedTables("main",
new string[] { "" }) as List<IDataSourceNamedTable>; //Get the list of tables
in database.
```

Opening a GeoPackage Tab file

A GeoPackage tab file, which references a Feature Table by name within a GeoPackage database, may be created using MapXtreme (v8.1 and above) programmatically and MapInfo Pro (v15.2.2). Multiple tab files may reference to the same database since one database may contain multiple Feature Tables.

To open a GeoPackage Tab in MapXtreme file, use the code below:

```
TableInfo infoGeoPackage = TableInfoGeopackage.CreateFromFile("Path to Tab
file");
Table openedTable = Session.Current.Catalog.OpenTable(infoGeoPackage);
```

Opened table will have table type as GeoPackage

```
openedTable.TableInfo.TableType == TableType.Geopackage
```

GeoPackage tab file created using MapInfo Pro (v15.2.2) can be opened in MapXtreme (v8.1 and above) and vice versa.

Enable GeoPackage as cache for RDB (SQL/Oracle) tables

User has the option to use GeoPackage as Cache for RDB Tables. Following Metadata properties should be configured for to enable this functionality.

```
TableInfoServer tis = new TableInfoServer("GeoPackageCacheAPITest");
tis.ConnectString = "Driver={SQL Server Native Client
11.0};DATABASE=QADB;Server=ServerName;UID=UserName;PWD=Password";//
"DSN=ServerDSN ";
tis.Query = "Select *  From Table";
tis.Toolkit = ServerToolkit.Odbc;
CacheParameters cp = new CacheParameters(CacheOption.All);
cp.StorageType = CacheStorageType.Geopackage;   //Cache Type as Geoackage.
tis.CacheSettings = cp;
```

Multiple RDB tables are cached in the same GeoPackage Cache database. The benefit for using the GeoPackage for the cache is that MXT will use fewer temporary files which is important for environments where the number of available file handles becomes limiting.

ⓘ Note: When using GeoPackage as a cache format, MapXtreme will store information that is not supported by the standard (such as coordinate system information and styles information). It is advisable not to use Cache file directly.

Create and Save GeoPackage file programmatically

GeoPackage table can be created using TableInfoGeopackage API. This will allow user to create and save .gpkg file along with its Tab file at the specified path.

```
TableInfoGeopackage tig = new TableInfoGeopackage("GeoPackageTest");
tig.Temporary = true;
Column col = new Column();
tig.Columns.Add(ColumnFactory.CreateIndexedIntColumn("ROWNUM"));
tig.Columns.Add(ColumnFactory.CreateStringColumn("StrName", 40));
tig.Columns.Add(ColumnFactory.CreateIndexedDoubleColumn("Doubleval"));
CoordSysFactory csf = new CoordSysFactory();
CoordSys wgs84 = csf.CreateLongLat(DatumID.WGS84);
tig.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn("GEO", wgs84));
tig.Columns.Add(ColumnFactory.CreateStyleColumn());
tig.DatabasePath = _tempPath + "GeoPackageTest.gpkg";
tig.DatabaseTableName = "GeoPackageTest";
tig.TablePath = _tempPath + "GeoPackageTest.TAB";
Table _miTable = Session.Current.Catalog.CreateTable(tig);
```

The above code will create a GPKG database along with its TAB file and will save that file at TablePath. If GPKG database already exists then table will be added to the existing Database.

Once the table gets created then Features can be inserted, updated and deleted from the table. MapXtreme can only create GeoPackage tables for which EPSG code and OGC “Well Known Text” description is available for the Coordinate System. List of supported Coordinate Systems in MapXtreme can be found in “MapInfoCoordinateSystemSet.xml” at installation path “C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x”.

For Persistence support if a GeoPackage Table has been opened using Tab file, .GPKG file or has been opened or created programmatically, appropriate information is persisted in the workspace file (.mws) in MapXtreme and will be reloaded when workspace is reopened.

ⓘ Note: As per GeoPackage Specification Requirements - " Every feature table or view in a GeoPackage SHALL have a column with column type INTEGER and PRIMARY KEY AUTOINCREMENT column constraints".

Due to above rule when a new GeoPackage Table is created then ID column is added to the column list by default. So the column order of the newly created table may differ from the order supplied.

GeoPackage standard does not support capture of styling information with the data. As a result, MapXtreme will apply a default styling to the data. Users should prefer to use layer style overrides and/or themes on their GeoPackage feature layers to display the data in the desired way.

MapInfo Pro (v15.2.2) will store default styling in the .TAB file which MapXtreme (v8.1 and above) will recognize and it is applied on the GeoPackage table.

20 – Geocoding

This chapter covers the MapXtreme namespaces for geocoding and provides descriptions and examples for writing applications that will access geocoding servers or services.

In this chapter:

Overview of the MapInfo.Geocoding Namespace

The MapInfo.Geocoding namespace provides interfaces and classes for geocoding address records. Geocoding is the process of determining the geographic location of a street address, intersection, place or postal code in order to pinpoint the location on a map. The geocoding client sends requests via HTTP to Precisely's geocoding server or geocoding web service product. The server and web service are not included in the purchase of MapXtreme.

Perhaps you are building a find the nearest application in which your users provide their geographic location, in the form of an address, and receive a map that shows where they are in relation to the closest ATM machine. To perform this task, you need more than an address to display a location on a map. These geographic locations (addresses, buildings, points of interest) translate to coordinates on the Earth's surface. You need the Latitude/Longitude, or coordinate, values to pinpoint and map the location. The MapXtreme geocoding client gives you these values determined (geocoded) from a place name, street address, postal code, or from the intersection of two streets.

In addition to returning the coordinate values, the geocoding client can return a complete description of the place, which is useful when only partial information is known. The geocoding client may return zero, one, or more responses to a request depending on the request preferences, the method being employed, the specified preferences, and the match criteria.

The MapXtreme geocoding client also has Gazetteer type functionality (world/country/city Geocoder), that geocodes a partial address. For example, an address containing only a city, state, place name, landmark, or airport.

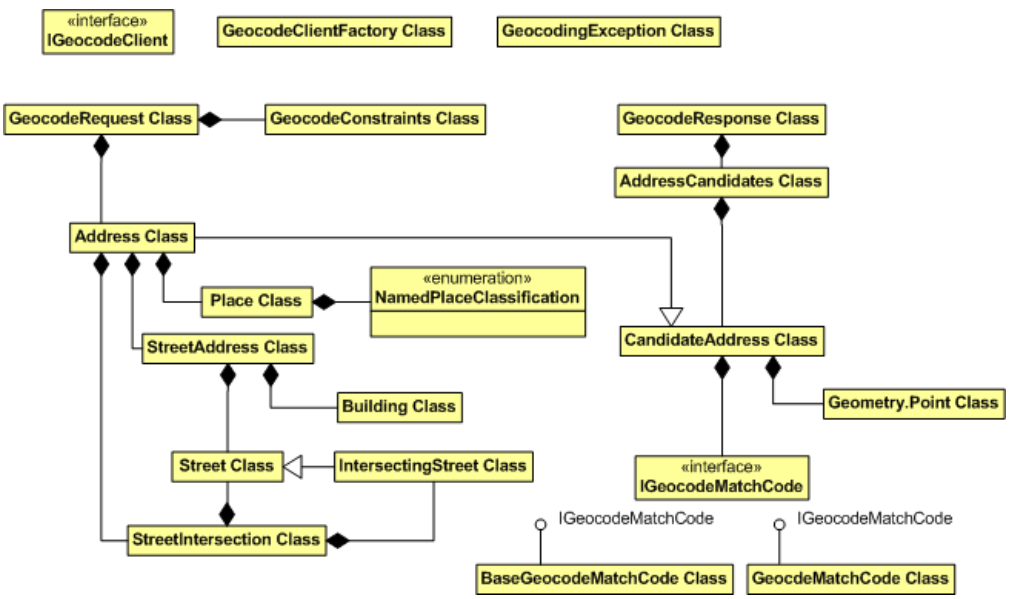
Using the geocoding capabilities, you get those coordinates by matching the input address to one that already has the coordinates. You supply one or more addresses in a single request document. Incomplete addresses can be processed as well. The server returns a complete address that you can use to clean up your input addresses. The server returns zero, one or more responses to a service request, depending on the quality of the data and the request preferences. Responses typically include a list of one or more potential matches, called candidates, and result codes that indicate the positional accuracy of the candidates. Your application will then need to resolve the response by choosing the appropriate address candidate to retrieve the coordinates.

Main Geocoding Classes

Geocoding is supported using either a MapInfo geocoding server or the Envinsa Location Utility Service. One of these resources needs to be deployed and made available via a URL. To create the appropriate geocoding client, use the GeocodeClientFactory class. The interface for both clients is similar as they both use the same classes for geocode requests, preferences, and responses.

The main interface for using the geocoding client is called MapInfo.Geocoding.IGeocodeClient. This interface defines a geocoding client object. It has a single method, Geocode, that takes a request as input and returns a response containing candidate addresses (addresses that are a potential match for the input address). Other classes include: GeocodeRequest, GeocodeResponse, GeocodeClientFactory, GeocodingConstraints, AddressCandidates, GeocodeMatchCode, and CandidateAddress.

The following diagram illustrates the interfaces and classes that make up the Geocoding namespace:



GeocodeRequest

The GeocodeRequest class sends a request to the geocoding server or service. Its properties include AddressList, a list of addresses to be geocoded, and Length, the number of items in the address list. Input addresses can include a variety of information, including street address or intersection, primary and secondary postal code, and country code.

GeocodeResponse

The GeocodeResponse class contains a response for each address in the GeocodeRequest. The response contains address candidates, which are those addresses considered to be possible matches for the input address. Note that a GeocodeRequest that contains multiple addresses will yield a GeocodeResponse object that includes a list of candidate addresses for each input address.

GeocodeClientFactory

This class returns an IGeocodeClient which you use to send a GeocodeRequest and receive a GeocodeResponse. The method GetMmjHttpClient uses the geocoding client that speaks to an instance of the MapInfo MapMarker Java servlet. Use the method GetEnvinsaGeocodeClient to send requests to the Location Utility Service in MapInfo Envinsa 4.0.

GeocodingConstraints

This class contains the preferences that can be set when geocoding. All are set/get properties that return true or false (default).

AddressCandidates

This class is a list of possible address matches that is returned in a GeocodeResponse.

BaseGeocodeMatchCode and GeocodeMatchCode

These are classes that implement the IGeocodeMatchCode interface. The interface exposes a ResultCode property that is a string that describes how well the match was made. BaseGeocodeMatchCode is returned when a request is sent to the Envinsa Location Utility Service. GeocodeMatchCode extends BaseGeocodeMatchCode. It is returned when a geocode request is sent to a MapMarker server. GeocodeMatchCode includes additional convenience properties to determine which parts of the address

matched. Among them, StreetAddressMatch and MunicipalityMatch return a value of true if the candidate matched the street and municipality. For details on result codes see Understanding Accuracy for Close Matches.

CandidateAddress

This class defines an address that was geocoded. Its properties include Address, the address for the geocoded address, GeocodeMatchCode, which explains how well the address was geocoded, and Point, the geometry that represents the candidate address.

Understanding the Geocoding Model

The geocoding client is based on a model of relative matching that is governed by a set of weights that scores each portion of the address against candidate records (possible matches) in the data. The resulting scores are summed and the candidate's total score is used to determine the best match or matches. A close match is made when there is a candidate that scores well above other candidates. In addition, the matching routine uses a set of geocoding preferences to determine whether certain matching conditions are required or relaxed. For example, the default preferences include relaxing a match on postal code, but requiring a match on the house number and street name. This gives the best return of hit rate with the fewest erroneous matches (false positives) and the best performance.

Geocoding Trade-offs

With a relative matching system such as the geocoding client, there are trade-offs that must be considered in light of how you use the geocoded data. Consider questions like the following:

- What level of matching accuracy are you looking for (unique address match, close match)?
- What level of geographic accuracy do you need for your geocoded points (street level, ZIP Code centroid)?
- Is your goal to geocode as many records as possible?

The answers to these questions are driven by how you intend to use the geocoded records. For example, perhaps you are determining the location of a new retail store and need to know the distribution of current and potential customers. In this case, you want to geocode as many of these customers as possible and do not need an exact street address match for each one. Geocoding to postal code centroid is fine for your analysis.

On the other hand, if you, as a utility service coordinator, need to know where your customers are in relation to neighborhood gas lines, the positional accuracy of each customer is of critical importance to you. Geocoding to street level with strict matching preferences is your best strategy.

A Few Words About Addresses

The quality of the address data is of utmost importance in geocoding. A clean input address that follows standard address conventions for the locale, will yield better results than incomplete or poorly formed addresses.

This section is an overview of address quality for both the input addresses as well as the matching reference address. To get the most out of geocoding, it's important to understand the address structure of your data and the data to which you are matching.

Input Addresses

Input addresses are those addresses you wish to geocode. These are addresses that do not contain geographic coordinates and thus cannot be located on a map.

Input addresses are made up of several components that the geocode server examines to determine a match. Address number, street name and street prefixes and suffixes can be contained in an input address. Prefixes are commonly directional in nature, such as North or South and exist only for some addresses. Suffixes are typically the type of street, Street, Road, Avenue Other information you might find in an input address is an apartment or route number.

Input addresses can contain a full or partial address and can contain spelling variations. For example, LaSalle Street as input will match to LaSalle St. However LaSalle without a street suffix may not because the address could really be LaSalle Ave.

The MapXtreme geocoding client can use so-called "dirty" data and incomplete addresses. This client accommodates both U.S. and non-U.S. addresses, provided the server contains the appropriate reference addresses.

Reference Addresses

Reference addresses are those that the server matches against the input addresses. These records contain the geographic coordinates required to locate the address on a map. Reference addresses are maintained on the server and are updated regularly through maintenance programs to accommodate new addresses. They have been standardized based on conventions of the locale. For example, in North America, street numbers precede the street name. In other parts of the world, the street number may follow the street name.

Typically the reference address contains the essential components: address number, street name and prefix, if one exists, and suffix. For non-U.S. locales, additional information such as apartment number, may not be contained in the reference address and thus, ignored in the input address.

What are Custom User Dictionaries?

There may be situations where the MapInfo Address Dictionary, provided with the server or service, does not contain the desired coverage for the areas or regions in which you are interested. Also users may have their own custom data they may prefer to use (for example, the data for all the company stores in the country). For these purposes users can use their own dictionaries that contain this kind of data.

The geocode client supports the use of custom dictionaries. A custom dictionary is a table of streets and address ranges that you match ungeocoded records against. You can use as many custom dictionaries as you need when geocoding. Refer to the server or service documentation for more information on creating custom user dictionaries.

What is World Geocoding?

The geocoding client allows you to geocode in different countries by specifying the country code in the input address. The geocode world component is different from the MapMarker-based component, which provides one country or area per component. The geocode world component provides city and postal code data coverage for 238 countries or areas, and street level data coverage for nine countries. Note that for both options access to data is licensed, and requires having a valid data license file. For information on Geocoding World, please contact your Precisely sales representative.

Geocoding a Location

Geocoding a location helps visualize your data relationships. For example, once address records are geocoded, they can be used by the routing client to display them as driving directions between two addresses (locations). Once data has geographic references, spatial searches can be performed to answer questions such as "Find all customers within 10 miles of this location." All geocode requests can:

- Geocode an incomplete address and return a complete set of address information (a normalized address).
- Indicate the number of exact or close matches in the response for a particular address supplied in the geocoding request.

- Process one or more addresses in a single geocoding request.
- Provide information on the quality of the result by using a match code.

Using the GeocodeRequest class you can send a request to the geocoding server or service. Its properties include AddressList (a list of addresses to be geocoded), Length (the number of items in the address list), and GeocodeContraints (see Using Constraints for Accurate Geocoding for more information). As part of the input address, you can include a variety of information, including street address, intersections, primary and secondary postal codes, place names, and a country code. The level of information included in the input address will determine the level and accuracy of the geocode result candidates.

Using the geocoding client, you can perform various levels of geocoding:

Street Address Geocoding

You can use geocoded street address information to display on a map or to perform spatial searches and queries. For example, this is useful for displaying store locations and the customers who are part of a store loyalty program on a map to determine market regions. For street address geocoding, use the Address class to specify the available information for the input address.

How do I Geocode an Address?

The following example shows how to geocode a street address in C# using the geocoding client. There are three sections to a geocode request: Define the parameters, create the street and address objects, and create the geocode request.

See also a Geocode sample application provided in ..MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\Geocode\cs.

Define your Parameters

To populate the address object you must define your address parameters. In addition to the address parameters, you must also define the server or service URL.

```
//Define the server URL
   String MMJUrl = "";

//Define the address parameters
   String streetName = "One Global View";
   String cityName = "Troy";
   String stateName = "NY";
   String zipCode = "12180";
   String countryCode = "USA";
//The following are optional address parameters
```

```
   String directionalPrefix = "";
   String typePrefix = "";
   String typeSuffix = "";
   String directionalSuffix = "";
```

Create the Address

Populate the address by creating the Street, StreetAddress, and Address objects.

```
//Create a Street object
MapInfo.Geocoding.Street street =
   new MapInfo.Geocoding.Street(
   directionalPrefix,
   typePrefix,
   streetName,
   typeSuffix,
   directionalSuffix);

//Create a StreetAddress object
MapInfo.Geocoding.StreetAddress streetAddress =
   new MapInfo.Geocoding.StreetAddress(street);

//Create an Address object
MapInfo.Geocoding.Address address =
   new MapInfo.Geocoding.Address(streetAddress, countryCode);
address.PrimaryPostalCode = zipCode;
address.PlaceList =
   new MapInfo.Geocoding.Place[]
   {new MapInfo.Geocoding.Place(
      cityName,
      MapInfo.Geocoding.NamedPlaceClassification.Municipality),
   new MapInfo.Geocoding.Place(
      stateName,
   MapInfo.Geocoding.NamedPlaceClassification.CountrySubdivision)};
```

Create the Geocode Request

Once the address object is created, add the address to a list and use the list to create the geocode request. Using the GeocodeClientFactory, define whether the request is being sent to the geocode server or service. Use the GetMmjHttpClient method for a request sent to the geocode server or GetEnvinsaLocationUtilityService method for a request sent to the geocode service.

```
//Add the address to a list of addresses
MapInfo.Geocoding.Address[] addressList = {address};

//Create the geocode request
MapInfo.Geocoding.GeocodeRequest geoRequest =
   new MapInfo.Geocoding.GeocodeRequest(addressList);

//Create the geocoding client
MapInfo.Geocoding.IGeocodeClient geoClient =
```

```
MapInfo.Geocoding.GeocodeClientFactory.GetMmjHttpClient(MMJUrl);

//Send the request and get the response
MapInfo.Geocoding.GeocodeResponse geoResponse = geoClient.Geocode(geoRequest);
```

Street Intersection Geocoding

Street intersections can be made up of different address types or address elements, and the request may have different levels of accuracy or preferences. For example, a mobile subscriber could use street intersection geocoding to enter the nearest street intersection to view a map of their location on the mobile device. For street intersection geocoding, use the StreetIntersection class and specify a Street and an IntersectingStreet property.

Postal Code Geocoding

Postal code centroids represent the centre of a postal code region. Being able to locate these centroids on a map lets you perform demographic analysis to find market regions to target for advertising or direct mail. For postal code level geocoding, use the Address class specifying only the PrimaryPostalCode information.

Gazetteer Type Geocoding

As part of the geocoding client, you can perform a world, country, or city geocode operation that can find the position of a partial address containing only the city, state, or a place name. The following operations are supported:

- Search, based on country, country subdivision, city, city subdivision, landmark, or airport to return the position.
- The pattern search capability supports a wildcard character (*) as a search value in one or more fields to return all available values for that particular field.

Batch Geocoding

Batch geocoding is supported using the IGeocodeClient.BatchRequests property. This property sets the number of addresses that will be sent to the server for a single operation. The defaults number of addresses is 25. The BatchRequest property must be greater than 0 and less than 500. When submitting addresses to the IGeocodeClient.Geocode() method, you may exceed the number specified in the BatchRequest property. If this occurs, the geocode operation containing these addresses will be broken down into multiple operations, where each operation contains no more than the number specified in the BatchRequest property.

ⓘ A small number for BatchRequst (e.g., 2) will slow performance. A high number for BatchRequest (e.g., 500) may result in memory issues. The performance of batch geocoding is dependent on system configuration.

Using Constraints for Accurate Geocoding

The geocode client allows you to set the match restrictions, fallback, and multiple match settings when geocoding input addresses using the GeocodeConstraints class. Here you set the preferences to be as strict or relaxed as you need. The settings that give you the best compromise among match rate, accuracy, and performance, are:

- Require an exact match for house number and street name.
- Do not fall back to postal code centroid.
- Return close matches only.

The strictest matching conditions require an exact match on house number, street name, postal code and no fallback to postal code centroids. The server or service, in essence, looks for an exact street address match within the postal code named in your input address.

What are the Match Constraints?

The match constraints control how to match a given input address, the number or candidates returned for each address, and what auxiliary information will be returned. An input address may not exactly match a real world address possibly due to incomplete or incorrect information. The GeocodeConstraints class contains a series of properties to determine which pieces of the address are considered most important, and what to do when an exact match is not found.

The IGeocodeClient.DefaultGeocodeConstraints property can be used to retrieve the defaults from the server or service if possible. The geocode client will use the defaults from the server or service if no GeocodeConstraints are supplied.

Each of the following preferences impact the level of record matching that the geocode server or service employs:

- **CloseMatchesOnly** – Returns only those geocoded results that are close match candidates. The address candidates are ranked according to how closely the input address matches the MustMatchXXX preferences. Only those candidates flagged as close matches (true) are returned. The default is false.

- **MustMatchAddressNumber** – Only candidates matching the address number are considered close. The default is true.
- **MustMatchMainAddress** – Only candidates matching the street name are considered close. This does not prevent Soundex matches from being considered. (For example, "Muller Strasse" for "Mueller Strasse" may still be considered close). It does prevent expanded street name manipulation (misspelling), such as considering juxtaposed letters (For example, "Muleler Strasse" for Mueller Strasse"), if the input street does not generate a close match. The default is true.
- **MustMatchCountry** – Only candidates matching Country are considered close. The default is true. It is recommended that this preference is not changed.
- **MustMatchCountrySubdivision** – Only candidates matching Country Subdivision are considered close. The default is true. CountrySubdivision may be different in each country. For example, in USA, it is state, in CAN, it is province.
- **MustMatchCountrySecondarySubdivision** – Only candidates matching Country Secondary Subdivision are considered close. The default is false. CountrySecondarySubdivision are different for each country. For example, in the USA it is county, in Canada, it is Census Division.
- **MustMatchMunicipality** – Only candidates matching Municipality are considered close. The default is true. An example of a Municipality is a town or city.
- **MustMatchMunicipalitySubdivision** – Only candidates matching Municipality Subdivision are considered close. The default is false. An example of a MunicipalitySubdivision is an MSA (Metropolitan Statistical Area), or a borough of a city (such as Bronx or Brooklyn in New York City).
- **MustMatchPostalCode** – Only candidates matching postal code are considered close. The default is false.
- **MustMatchInput** – Only candidates matching all input preferences are considered close, regardless of what is set for that particular preference. The default is false.
- **FallbackToGeographic** – In case a street geocode request produces no candidate, this option determines whether geocoding is performed at the centroid of the geographic region. The resulting point is located at the geographic centroid of the area where it is possible to obtain the highest level of accuracy. The default value is set to false.
- **FallbackToPostal** – In case a street geocode request produces no candidate, this option determines whether geocoding is performed on the postal code centroid. The default value is set to false.
- **OffsetFromStreet** – Determines the distance from the street segment at which the address is positioned. This value is used to prevent addresses across the street from each other from being given the same point. For example, a house with number 50 on a segment that spans from 40 to 60 is interpolated as the midpoint of the segment. The OffsetFromStreet positions the point perpendicular from the road, so that houses on

the left side of the street appear on the left, and the houses on the right side of the street appear on the right. The default value for OffsetFromStreet is 25 meters. The Distance class is used to specify the OffsetFromStreet by defining the value and unit of measure.

- **OffsetFromCorner** – Determines the distance from the end (corner) of a street segment at which the address is positioned. This value is used to prevent addresses at street corners from being given the same point of the intersection. OffsetFromCorner positions the point parallel to the street (based on the street address) a distance along the segment. For example, a house with number 40 on a segment that spans from 40 to 60 is positioned so that its point is not located at the street intersection. The default value for OffsetFromCorner is 25 meters. The Distance class is used to specify the OffsetFromCorner by defining the value and unit of measure.
- **MaxCandidates** – Defines the maximum number of candidates returned in a response. The actual number of candidates returned may be less than this maximum value. A value of -1 is used to mean return all candidates that meet the other constraints. The default is 3 when using the MapInfo geocoding server. The default is -1 (all) when using the Envinsa Location Utility Service.
- **MaxRanges** – This preference controls whether a geocode operation returns the exact number for a street address or a range of numbers. If the MaxRanges is set to 0, then no ranges are returned and only the exact address number is returned. If the MaxRanges is set to a value greater than 0, then that number of street address numbers is returned. The default value for MaxRanges is 0. For example, if MaxRanges is set to 5, then the service returns up to 5 matches within the address range. A value of -1 returns all suitable matches.
- **MaxRangeUnits** – Defines the maximum number of candidate range units to be returned per candidate range. The default value for MaxRangeUnits is zero (0).
- **GeocodeType** – Determines the type of geocoding when available from the server. Refer to your server or service documentation for information on geocode type support.
- **LocalGeocodeConstraints** – Allows you to set country specific constraints which are specific to that country and that geocoder. These values are always Key/Value pairs. Refer to your server or service documentation for information on available local constraints.
- **DictionaryUsage** – Specifies if the server address dictionary is to be used by itself, if a configured user dictionary is to be used by itself, or if both of them are to be used. If both are to be used, a preference of one over the other can be requested. By default the AD_AND_UD option is used. This option indicates that the server address dictionary and any user dictionaries that are configured should be used. For information on creating and accessing user dictionaries, refer to your server or service documentation.

- **UseCASSRules** – Specifies if CASS level geocoding should be used by the server (US only). CASS certification is a process by which a table of mailing addresses is standardized to meet U.S. Postal Service® (USPS) requirements for bulk mailing discounts. The geocoding client performs this address standardization under strict matching conditions set by the U.S. Postal Service while it geocodes your records. You must deploy a version of the US geocode data component that supports CASS in order to use CASS geocoding. Contact your MapInfo sales representative to obtain a current version in order to comply with CASS address standardization. The default is false.

Impact of Relaxing Match Constraints

Relaxing the conditions broaden the area in which it searches for a match. For example, by relaxing the postal code, a search will be performed for candidates outside the postal code but within the city of your input address. Consider the following when you are determining the settings for geocoding:

Relaxing Postal Codes

When postal codes are relaxed, a search is performed on a wider area for a match. While this results in slower performance, the match rate is higher because the request does not need to match exactly when it compares match candidates.

Relaxing Subdivisions and Municipalities

When these are relaxed, a search is performed on the street address matched to the particular postal code, and considers other cities or subdivisions that do not match the name, but do match the postal code.

Relaxing Street Name

A search is performed that looks at all candidates with names that sound like the input address or that are spelled improperly. This slows down the performance. On the plus side, since more candidates are examined, the match rate increases. If your table is indexed, the time difference between performance and match rate is reduced.

Relaxing House Number

Performance is not significantly affected when the house number setting is relaxed. It does, however, affect the type of match if the candidate address corresponds to a segment that does not contain any ranges. The type of match can also be affected when

the house number range for a candidate does not contain the input house number. If you are relaxing the house number, it is recommended that you set the maximum ranges to be returned to a value higher than 0.

Understanding Accuracy for Close Matches

The ranking of the results is based on the close match of the request. For the GeocodeMatchCode class, the geocoding server or service returns a result code for every record it attempts to match based on the combination of GeocodeConstraints in the request and server or service configuration. The code represents the success or failure of the geocoding operation and conveys information about the quality of the match. The codes fall into five major categories:

- Single Close Match (S Category)
- Best Match from Multiple Candidates (M Category)
- Postal Code Centroid Matches (Z Category)
- Geographic Centroid Matches (G category)
- Non-Match Codes

For either S or M category result codes, each character of the code tells how precisely the geocode operation matched each address component.

Category	Description	Example
H	House number	115
P	Street prefix direction	North
N	Street name	Main
T	Street type	Place
S	Street suffix direction	SE
C	City name	New York
Z	Postal code	80302
A	Address Dictionary	A

The following example is a close match candidate:

```
S5HPNTSCZA
```

The following example is not a close match candidate:

```
S5-PNTSC-A
```

A dash in the result code indicates a conflict. If the postal code in the request conflicts with one found by the geocode in the data (input was 28031, value found is 28013), the resultant string contains a dash instead of a Z (For example, S5HPNTSC-A).

Single Close Match (S Category)

Matches in the S category indicate that the record was matched to a single address candidate. The first character (S) reflects that the geocoding server found a street address that matches the record. The second position in the code reflects the positional accuracy of the resulting point for the geocoded record.

- S1 – single close match, point located at postal code centroid
- S2 – single close match, point located at ZIP+2 centroid (US centric)
- S3 – single close match, point located at ZIP+4 centroid (US centric)
- S4 – point located at the center of a shape point path (shape points define the shape of the street polyline)
- S5 – point located at a street address position
- S6 – point located at centroid of geometry postal code. (For example, large buildings having their own codes). This represents the highest accuracy available
- SX – point located at street intersection
- SO – no coordinates available (very rare occurrence)

Best Match from Multiple Candidates (M Category)

Matches in the M category indicate that there is more than one close match candidate for the record and the server or service has chosen the best one of those candidates. As in the S category, the second position in the code of M category matches the positional accuracy of the resulting point object.

- M1 – multiple close matches, point located at postal code centroid
- M2 – multiple close matches, point located at ZIP+2 centroid (US centric)
- M3 – multiple close matches, point located at ZIP+4 centroid (US centric)
- M4 – multiple close matches, point located at the center of a shape point path (shape points define the shape of the street polyline)
- M5 – multiple close matches, point located at a street address position (highest accuracy available)

- M6 – multiple close matches, point located at point postal code location
- MX – multiple close matches, point located at street intersection
- M0 – multiple close matches, no coordinates available

Postal Code Centroid Matches (Z Category)

Matches in the Z category indicate that no street match was made, either: 1) because there is no close match and you allowed the server or service to fall back to postal code centroid; 2) the address is a P.O. Box or rural address; or 3) you set the server or service to match to postal code centroids. The resulting point is located at the postal code centroid with four possible accuracy levels.

- Z1 – Postal code centroid match
- Z2 – ZIP+2 centroid match (US centric)
- Z3 – ZIP+4 centroid match (highest accuracy available, US centric)
- Z6 – Postal code centroid match for point ZIP
- Z0 – Postal code match, no coordinates available (very rare)

ⓘ Point ZIPs are ZIP Codes without an area. These include P.O. Box ZIPs and Unique ZIPs (single site, building, or organization). When using a non US address dictionary, the results in this category will still be a Z.

Geographic Centroid Matches (G category)

Matches in the G category indicate that no street match was made. This takes place when no close match is found, and the server or service is configured to fall back to the geographic centroid. The resulting point is located at the geographic centroid with the following possible accuracy levels.

- G0 – areaName0 centroid (country)
- G1 – areaName1 centroid (country subdivision)
- G2 – areaName2 centroid (country secondary subdivision)
- G3 – areaName3 centroid (municipality)
- G4 – areaName4 centroid (municipality subdivision)

Non-Match Codes

The following result codes indicate no match was made:

N – No close match. These records can be re-geocoded interactively using MapInfo geocoding products or during subsequent automatic passes under different matching conditions.

NX – No close match for street intersections.

ND – The server or service could not find the Address Dictionary for the given postal code or municipality/country subdivision. These records can also be re-geocoded once the Address Dictionary is available.

NG – The user marked these records during geocoding as non-geocodable. The server or service does not attempt to match these records again until the code is removed.

21

21 – Routing

This chapter covers the MapXtreme namespace for Routing and provides descriptions and examples for writing applications that will access routing servers or services.

In this chapter:

Overview of MapInfo.Routing Namespace

The MapInfo.Routing namespace provides an interface and classes for incorporating a MapXtreme routing client into your application, in a similar fashion to the geocoding client. The routing client sends requests via HTTP to Precisely's routing server or routing web service product. The server and web service are not included in the purchase of MapXtreme.

Using the routing capabilities, you can determine the shortest or fastest path (route) between two points, with the ability to add various advanced routing options. You can return the route geometry (layout of the route for map display), turn-by-turn directions, and a route summary of total distance and time. Additionally, you can create routes that include intermediate points and preferences that avoid certain road features, such as highways or certain locations.

The routing capabilities include important analytical tools, allowing you to calculate drive-time and drive-distance (isoChrone and isoDistance), and the ability to use transient updates to avoid certain road types or to recalculate a route. The routing client functionality is divided into four logical groups:

- Calculating Routes
- Advanced Route Options
- Iso Routing (Drive-Time and Drive-Distance)
- Updating a Request using Routing Data

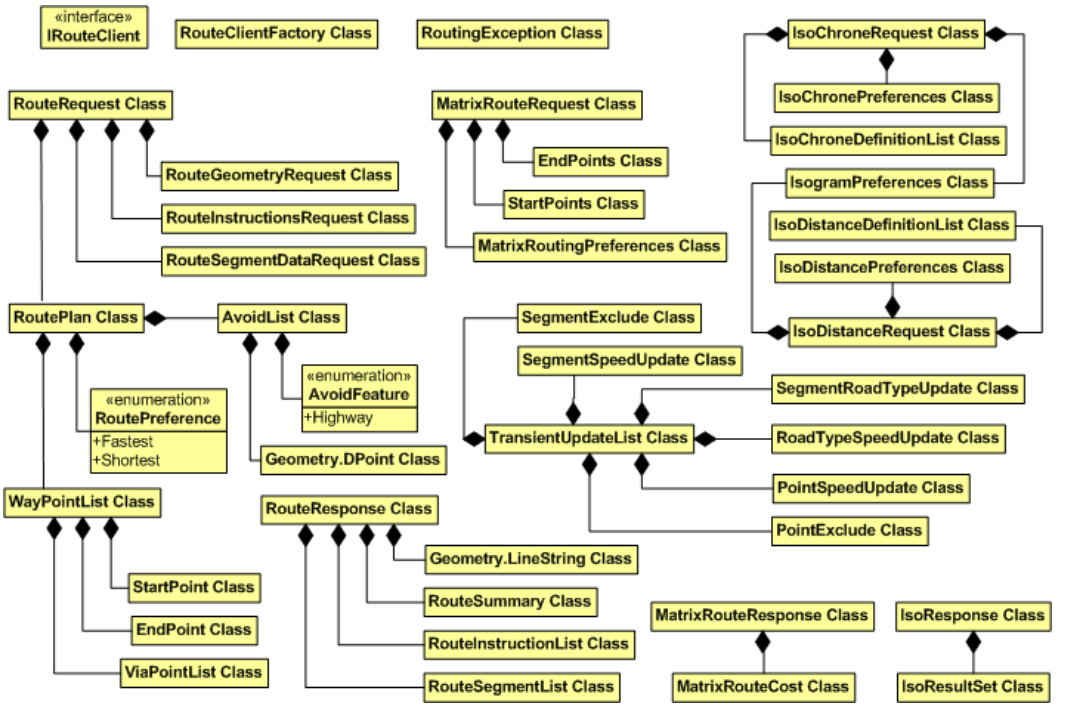
Main Routing Classes

Routing is supported using either a MapInfo routing server or the Envinsa Route service. One of these resources needs to be deployed and made available via a URL. To create the appropriate routing client, use the RouteClientFactory class. The interface for both clients is similar as they both use the same classes for route requests, preferences, and responses.

Using the routing client you can determine the shortest or fastest path (route) between two locations, add intermediate locations (ViaPoint class), avoid locations or road types (AvoidList, PointExclude, SegmentExclude classes), get driving directions (RouteInstructionsRequest class), get the route geometry (RouteGeometryRequest class), and customize your routes using numerous preferences resulting in more meaningful analysis. The routing client includes important analytical tools, allowing you to calculate drive-time and drive-distance (isoChrone and isoDistance), and the ability to use transient updates to avoid certain road types or to recalculate a route. The routing client functionality is divided into three logical groups:

1. Calculating routes using the RouteRequest class for point-to-point and multi-point routes, and the MatrixRouteRequest class for matrix routes.
2. Calculating drive-time and drive-distance using the IsoChroneRequest class for drive-time and the IsoDistanceRequest class for drive-distance.
3. Calculating routes with transient updates to produce more accurate and meaningful routes using the TransientUpdate class.

The following diagram illustrates the interfaces and classes that make up the Routing namespace:



Calculating Routes

There are three types of route calculations (determining shortest or fastest route) that can be performed: simple routes (point-to-point), multi-point routes, and matrix route requests.

Point-to-Point Routing

One of the most common routing requirements is for simple driving directions where a customer inputs two addresses, two points, or any combination, and gets back a route. The route will be calculated from a specified start point to a specified end point.

At the simplest level of point-to-point routing, the response includes a route summary. This summary includes the distance and time of the route. There are numerous additions you can make to a point-to-point route request to return additional information in the response. These additions are:

- Routing Preferences
- Driving Directions
- Route Geometry
- Avoiding Points, Features, and Segments
- Time-Based Routing
- Updating a Request using Routing Data

A point-to-point route is created using the RouteRequest class. Refer to the Developer Reference for API-level syntax and descriptions.

The following code sample is an example of a simple point-to-point route:

```
Public Shared Sub New_RouteRequest()
    ' Create the start point and end point
    Dim coordSys As MapInfo.Geometry.CoordSys = _
        Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt1 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 42)
    Dim dpt2 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 41)
    Dim startPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt1)
    Dim endPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt2)
    ' Create the point list
    Dim pointList As WayPointList = _
        New WayPointList(startPoint, endPoint)
    ' Create the route plan object
    Dim plan As RoutePlan = _
        New RoutePlan(pointList)
    ' Create the request object
    Dim request As RouteRequest = _
        New RouteRequest(plan, DistanceUnit.Mile)
End Sub
```

Multi-Point Routing

Multi-point routing is the ability to route via points. The routing server will find the shortest time or distance through all the points. This can be used for complex directions. These directions are a requirement for customers who wish to plan daily drop-offs, or customer visits such as distribution and repair companies, or general field sales. Typically, a start and end point are defined plus a series of stops are added into the calculation. There is no difference between multi-point route and point-to-point route except for the via points between the start and end locations.

Similar to point-to-point routes, there are numerous additions you can make to a multi-point route request to return additional information in the response. These additions are:

- Routing Preferences
- Driving Directions
- Route Geometry
- Avoiding Points, Features, and Segments
- Time-Based Routing

A multi-point route is created using the RouteRequest class. To add intermediate points to a route, WayPointList must consist of at least one ViaPoint. Refer to the Developer Reference for API-level syntax and descriptions.

The following code sample shows how to create a WayPointList consisting of a startPoint, endPoint, and ViaPoint:

```
Public Shared Sub New_WayPointList()
    ' Create a start point and end point
    Dim coordSys As MapInfo.Geometry.CoordSys =_
        Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt1 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 42)
    Dim dpt2 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-73, 42)
    Dim dpt3 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-73.562, 42)

    Dim startPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt1)
    Dim endPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt2)
    Dim intermediatePoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt3)

    ' Create the intermediate point object
    Dim point As ViaPoint = New ViaPoint(intermediatePoint)
    ' Create the collection
```

```
    Dim pointList As ViaPointList = New ViaPointList
    ' Append the object to the collection.
    pointList.Append(point)
    ' Create the WayPointList object
    Dim list As WayPointList = _
        New WayPointList(startPoint, endPoint, pointList)
End Sub
```

Matrix Routing

Matrix routing allows you to find the shortest or fastest paths between a number of start points and a number of end points, and return the route costs. The costs are the total time and distance of the individual routes. The matrix route feature is extremely useful for situations where you have 3 start points and 50 destination points, where you want to find the shortest paths between all the start points and all the destinations, and return the routes' costs.

The order of the points and number of sequences in the response are determined by the number of start and end points in the request. For example, if two start points and two end points are specified in the request, the response will contain the following order of sequences (where S is start and E is end): S1 to E1, S1 to E2, S2 to E1, S2 to E2.

Typically, this calculation can be used to determine service response time and coverages for specific services such as a fire house or police station. You may require these calculations to ensure they can provide adequate coverage for service level agreements such as having one or more people who can respond to an incident within 20 minutes of first notification. At the simplest level of matrix routing, a list of sequences will be included in the response. Each sequence includes the start and end point, as well as the distance, and time. There are numerous additions you can make to a matrix route request to return additional information in the response. These additions are:

- Routing Preferences
- Avoiding Points, Features, and Segments
- Updating a Request Using Routing Data

A matrix route is created using the MatrixRouteRequest class. The matrix route request must consist of a MatrixRouteCost and may include a MatrixRoutingPreferences. Refer to the Developer Reference for API-level syntax and descriptions.

Advanced Route Options

There are numerous advanced capabilities that can be added to your routing applications. This section highlights these important capabilities used to customize your point-to-point, multi-point, and matrix route requests.

Routing Preferences

You can specify various preferences for each type of route you are calculating. All of these preferences are optional, but can be very important in determining the accuracy of the route and helping you create a more meaningful route. You can return the route with fastest time from the start point to the end point or the shortest distance from the start point to the end point. If specified, these preferences will override the route optimization settings defined in the server or service configuration. The preferences may be overridden by server preferences if the server preferences are more restrictive.

The type of route calculation (point-to-point, multi-point, or matrix) determines which route preferences are available to use. The following list outlines the available route preferences:

- **DistanceUnit** - You can specify the distance unit for all of the route calculations. The most common units are Kilometer, Meter, Mile, Yard, and Foot. If a value is not specified, the default setting will be used (Mile).
- **RouteMethod** - Determines the general type of route preference to perform. The route client can either perform Fastest or Shortest route calculations. If a value is not specified, the default setting will be used (Fastest).
- **OptimizeIntermediatePoints** (only used in multi-point routes) - Determines if the via points are to be traversed in the order specified in the request or have the route client find the optimal route using these via points. By default (false) the route client will traverse the specified points in the order in the request. When set to true, the optimal route using the via points is calculated.
- **StopThreshold** (only used in multi-point routes) - Sets the route calculation algorithm stop threshold. The route algorithm will stop calculating and return the current ‘best’ route when the difference in time or distance between the route candidates remaining in the algorithm reaches the stop threshold. Valid values are any positive numbers. For best performance, a positive number less than one is recommended. Setting the stop threshold is a balancing factor between accuracy and speed. The lower the threshold value on average, the more accurate the result and the longer the route will take to calculate. The stop threshold has a minimal effect on routes with few via points (under 10). The default is 0.01 (1 percent).

- **TimeOut** - Sets the route calculation algorithm time out value. The time out value is in seconds. The default time out is 600 seconds. The route algorithm will stop calculating once the time out value is reached.
- **TimeUnit** - Sets the unit of measure for all routes durations. The default time unit is Minute.
- **TavelPreferences** - Allows you to specify unique routing abilities for modifying road type priority of travel. To add travel preferences you can specify a road type and a road type preference. Routing can be performed on priority basis by setting the RoadTypePreference to High, Medium, Low, or Avoid. The road types with the higher priority will be chosen over those with the lower priorities when the route is calculated. You can also use the Avoid RoadTypePreference to avoid a particular road type. One of the more often used travel preference is to set the FERRY RoadType to Low, which will only use ferry routes if that is the only mean of travel to the destination.
- **TravelTime** (point-to-point and multi-point) - Sets the start or end travel time for the route request. Refer to Time-Based Routing for more information on how to define the travel time and other time-based routing options.
- **Updates** - Allows you to include transient updates in your route request that temporarily modify road or segment speed and road types. Refer to Transient Updates for more information on including updates.

The RoutingPreferences and MatrixRoutingPreferences classes are the starting point when defining route preferences. Refer to the Developer Reference for syntax and descriptions.

Driving Directions

Driving directions can be returned for a point-to-point or multi-point route. These instructions list the step-by-step procedure for the route. Each segment of the route will return the step that needs to be performed (for example, Begin, Turn Right), the name of the street, the direction, the distance, and the time. There are numerous options for driving directions including focusing the directions to a particular section of the route, changing the wording and format of the directions, and changing the language for the directions.

There are two places where this can be configured/requested. The first place is in the server or service settings. The second place is using the RouteInstructionsRequest class setting the ReturnDirections property to true.

Modifying Direction Preferences

The following list describes the available preferences used to modify driving directions:

- **DirectionBreakTurnAngle** - Sets the turn angle value that determines when a street is broken into a new directions string. Sometimes, when following a route a street will make a significant turn while keeping the same name. By using this value, the user can specify the turn angle at which a new direction should be started. Valid values are 0 to 180 degrees. The default is 45 degrees.
- **DirectionsGeneratorName** - Specifies a custom, server-side directions generator to be used to create the route directions. If the server or service is configured with one or more directions generators, this preference may be used to force directions generation to be handled by a specific generator. For instance, setting “myGenerator” will force the server or service to use the generator that has been configured with the name "myGenerator”. Use this preference only if your server or service has been configured with a custom direction generator. Refer to your server or service documentation for instructions on how to create and use direction generators.
- **Focus** - Specifies the focus of the route. A focused route is a subset of the whole route that concentrates on either the beginning or end of the route. A route focused at the start will route the user from their origin to (and including) the first major highway. A route focused at the end will route the user from the last major highway in the route (including that highway) to the destination. If there are no major highways in the route, the focused route will be the same as an unfocused route. NONE signifies directions for the whole route will be returned, START signifies that the start of the route will be returned, or END signifies that the end of the route will be returned. The default is NONE.
- **ShowDistance** - Indicates whether or not to return the distance of a direction in the driving directions. The default is true.
- **ShowPrimaryNameOnly** - Specifies whether or not to show only the primary street names. It is often the case that a street contains multiple names. This is used to indicate whether all names for the street should be shown in the directions or only the primary name. If set to true, only the primary name of the street will be used in the directions. If set to false, the primary name and all alternate names will be used. The default is false.
- **ShowTime** - Indicates whether or not to return the time it takes to follow a direction in the driving directions. The default is true.
- **Style** - Specifies the type of directions to return. The client can return two types of directions, normal and terse. Normal is the standard driving direction instructions. For example “Turn left on Yonge Blvd and travel Southeast (1 s)”. Terse directions are shorter directions that are more suitable for wireless devices. For example “L on Yonge Blvd”. This can be set to NORMAL or TERSE. The default setting is NORMAL.

Setting the Direction Language

You can manipulate the output language of your route instructions by changing the UserLocale property. The route client uses ISO standard locale code <language>_<country> (for example, French is fr_FR, German is de_DE). Currently the route client supports EN, FR, DA, DE, FI, NL, NO, ES, PT, IT, SV. English (en_US) is the default language.

Route Geometry

The ability to create a route geometry is a vital capability in creating maps and analysis of a route. The route geometry is the visual representation of the route. By default the route geometry is not returned. However, you can return the route geometry in the response by setting the ReturnGeometry property to true.

Avoiding Points, Features, and Segments

You can avoid or exclude types of roads, points, or route segments. Before determining a route, you may already know of a road or location that you would like to avoid based on traffic information, accidents, holidays, rush hour, or any other known factor.

There are three types of avoids or excludes: points, features, and segments:

Avoiding Points

Currently there are two methods to avoid points along the route. You can use the Points property of the AvoidList class or you can exclude any number of points from a route request using the PointExclude class.

Using the PointExclude class is the recommended method to avoid points.

Avoiding Features

Currently there are two methods to avoid features (road types) along the route. The TravelPreferences property of the RoutingPreferences class can be used to avoid any road type. See Routing Preferences for more information on using the TravelPreference property.

The route client also provides another method to avoid highways when calculating routes. This feature is useful when you want to limit routes to local roads. You can use the Features property of the AvoidList class and define Highway as the avoid feature.

Avoiding Segments

You can exclude any number of route segments (ID) from a route request using the SegmentExclude class. The segment ID is a unique identifier assigned to each line, road, or section of the route data. You can determine the segment ID for a particular section of the route by returning them in a previous route response. See Returning Segment Information for more information on how segment ID's are returned in a route response.

ⓘ If the route cannot be calculated or there is no way to get a destination using the avoid or exclude, then the route directions will not be returned.

Time-Based Routing

Time-based routing is a key component for delivery systems, mobile work forces, and calculating accurate drive times and routes. You can specify a start or end time for your route or a stop duration for each intermediate point in a multi-point route. For example, you can specify that the route should start at location A and pass through locations B, C, and D where you spend five minutes at each talking to customers or loading your trucks. The route will then end at location E.

Start and End Times

Part of time-based routing is being able to specify the start or end time for a route. Start and end times in routing are important when other time-of-day preferences are used. For example if the server is configured with data that represents rush hour traffic patterns, the start or end time for your route may be of importance for analysis. The start and end time can be set using the TravelTime class. The following properties are used to control the travel time:

- **DateTime** - Specifies the date and time to either start or end a route calculation.
- **TimeZone** - Specifies the time zone used in the travel time. The time zone is defined using an hour offset value from Greenwich Mean Time (GMT).
- **TravelTimePreference** - Specifies if the travel time defined is a start or end time. The two members are defined using StartTravel or EndTravel.

The following code example shows how to set the travel time for the route. This example includes the StartTravel preference, defining the start time (year, month, day, hour (24 hour clock), minute, second) and time zone:

```
Public Shared Sub New_TravelTime()
    Dim startTime As DateTime = New DateTime(2005, 5, 1, 14, 0, 0)
    Dim timeZone As MapInfo.Routing.TimeZone = _
        New MapInfo.Routing.TimeZone(-4)
```

```
    Dim travelTime As MapInfo.Routing.TravelTime = _
        New MapInfo.Routing.TravelTime(startTime, _
        TravelTimePreference.StartTravel, timeZone)
End Sub
```

Stop Times

At any intermediate point during a route you can specify a stop time. This time is added to the overall time along the route. The stop time is particularly useful for defining time that the user is going to spend at a particular location along the route (for example, loading and unloading time of a delivery truck) for a multi-point route. A stop can be added to the ViaPoint class by specifying a Stop and a StopTime property:

- **Stop** - You can specify whether to stop at a viapoint. By default no stops are calculated at viapoints (false). You can specify if you want to stop at the viapoint by defining the Stop equal to true.
- **StopTime** - Adds a stop time to any viapoint along the route. The stop time is defined by the TimeSpan, and will be added to the total time for the route. The default stop time is 0 (zero). A TimeSpan can be represented as a string in the format "[-]d.hh:mm:ss.ff" where "-" is an optional sign for negative TimeSpan values, the "d" component is days, "hh" is hours, "mm" is minutes, "ss" is seconds, and "ff" is fractions of a second. For example, a TimeSpan defined as "11.13:46:40" is equivalent to 11 days, 13 hours, 46 minutes, and 40 seconds.

Iso Routing (Drive-Time and Drive-Distance)

IsoChrones and isoDistances can be extremely valuable information for making decisions. An isoChrone is a polygon or set of points representing an area that can be traversed in a network from a starting point in a given amount of time. An isoDistance is a polygon or set of points representing the area that is a certain distance from the starting point. They can be used to determine a drive-time or drive-distance boundary from a location. Users in retail, banking, and insurance, can use this service to determine the potential market or risk for any given asset. These boundaries may then be used for further analyses such as determining which prospects in a mailing list are within an iso boundary so they may be notified of new services available or a new store opening.

Creating an IsoChrone (Drive-Time)

When creating an isoChrone, you are looking for one or more polygons/set of nodes which represents an area that can be travelled in a given amount of time (drive-time cost) from a starting location.

The cost specifies the time used to calculate the isoChrone. The cost represents the time it takes to travel from the starting point to the calculated points on the road network. You can specify multiple costs to produce concentric bands (multiple isoChrones) that visually represent different distances that can be reached along the road network. Associated with multiple costs are a tags. You can specify a tag, or ID, for each cost in the request, that will identify the appropriate isogram result (geometry) in the response.

There are numerous preferences you can specify in the IsoChronePreferences and IsogramPreferences classes. These preferences allow you to obtain a desired output for your analysis.

IsoChronePreferences

The following preferences are available in the IsoChronePreferences class:

- **DefaultAmbientSpeed** - Determines the off-road network speeds where they are not specified. Roads not identified in the network can be driveways or access roads, among others. For instance, if you are at a point with five minutes left on an isoChrone on the off-road network, boundary points would be put at a distance based on the ambient speed and the time left. So, if the ambient speed in this case was 15 miles per hour, boundary points would be put at a distance of 1.25 miles.
- **AmbientSpeedOverride** - Overrides an ambient speed for a specific road type. The ambient speed can be overridden for all road types. For example, it may be set to 30 Mph for major urban roads on the weekends due to busy traffic:

IsogramPreferences

The following preferences are available in the IsogramPreferences class:

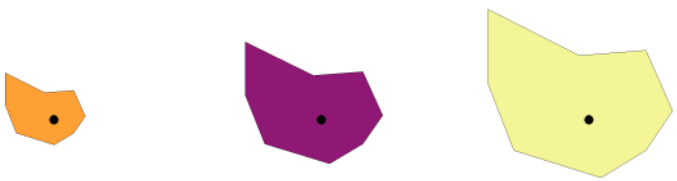
- **BandingStyle** - Specifies the style of banding to be used in the result. Banding styles are the types of multiple time or distance bands that can be displayed based on multiple costs. The styles include Donut (each boundary is determined by subtracting out the next smallest boundary) or Encompassing (each boundary is determined independent of all others). Multiple isoChrone bands may be requested by specifying multiple cost factors, such as asking for the isoChrone 5 minutes away and 15 minutes away from the same starting point. These end up as approximate concentric bands. The user may choose to show both complete sets of data (Encompassing style, which shows everything) or just the band between the two (Donut style), for everything between 5 and 15 minutes away.

  Donut banding can result in routing server problems when two boundaries (times) are almost identical. There are three request settings you can use to avoid this situation:

  a. Do not use the maximum off road distance setting if possible with Donut banding. If you must use this setting, set it as large as possible.

b. Do not use a low ambient speed setting with Donut banding.

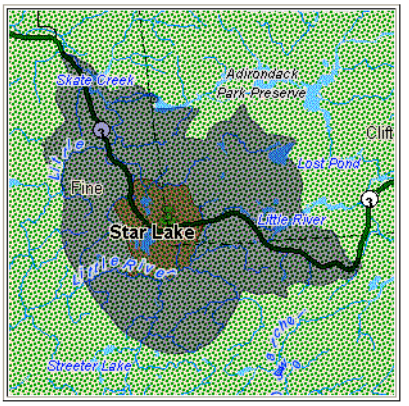
c. Do not make requests with cost increments that are small relative to the cost. For example, requesting 4, 5, and 6 minute costs (1 minute increments starting at 4 minutes) is not likely to be a problem, but 120, 121, and 122 minute costs may be a problem. The larger the cost the larger the cost increments will need to be.

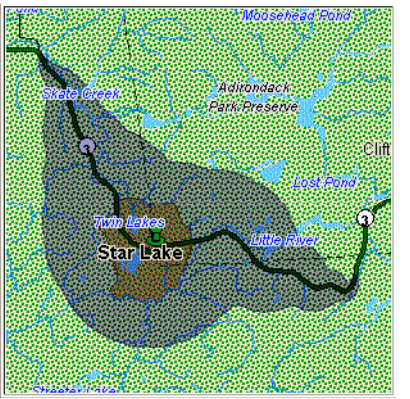


The above example shows the geometries returned for an Encompassing style banding of cost 5, 10, and 15 minutes from the starting point. Notice how each geometry includes the area of the prior isoChrone.

The above example shows the geometries returned for a Donut style banding of cost 5, 10, and 15 minutes from the starting point. Notice how each geometry begins at the end of the prior geometry, excluding the area of the prior isoChrone.

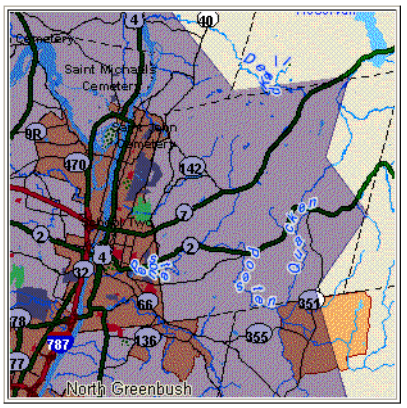
- **MajorRoadsOnly** - Determines what road network is used in the calculation. The network can include major roads only or all roads. A major road is a main road or highway. If you choose to use major roads, performance will improve, but accuracy may decrease. The images below illustrate the behavior of the major roads option. The image on the left shows MajorRoadsOnly set to true while the image on the right shows MajorRoadsOnly set to false. Notice that the service uses side streets and other types of secondary roads when calculating the iso response if MajorRoadsOnly is set to false.

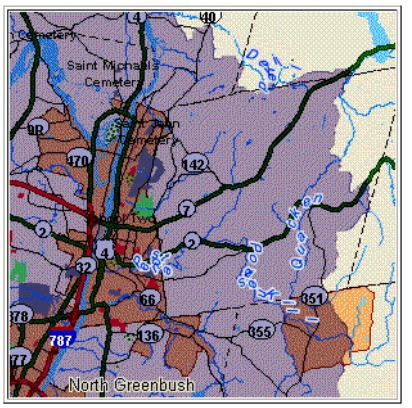




- **MaxOffRoadDistance** - Specifies the maximum amount of distance to come off the road network when using the ambient speed. The default setting for this property is no limit, allowing off-road travel to occur to the maximum of the isoChrone. The server may not be able to generate a response for a maximum off-road distance set to a very small value.
- **ReturnAccesibleNodes** - Specifies the type of isogram or feature to be returned from an isoChrone request be a set of nodes (MultiPoint). By default, ReturnAccesibleNodes is False. When this property is true, all of the points along the road network that can be reached for the isoChrone calculation will be returned.
- **ReturnGeometry** - Specifies the type of isogram or feature to be returned from an isoChrone request be a geometry (MultiPolygon). By default, ReturnGeometry is true. False will return all of the points along the road network that can be reached for the isoChrone calculation.
- **ReturnHoles** - Specifies whether or not to return holes in the response. Holes are areas within the larger boundary that cannot be reached within the desired time or distance, based on the road network. These pockets of territory are often neighborhoods of local roads that are cumbersome to traverse. Holes can be returned as is, or removed entirely. This setting will only apply if ReturnGeometry is set to true (default). If this property is not specified, the default setting, false, will be used.
- **ReturnIslands** - Specifies whether or not to return islands in the response. Islands are small areas outside the main boundary that can be reached within the desired time or distance. These areas are frequently located off exit ramps of major highways. Islands can be returned as is, or removed entirely. This setting will only apply if ReturnGeometry is set to true (default). If this property is not specified, the default setting, false, will be used.
- **SimplificationFactor** - Specifies the reduction factor for polygon complexity. The simplification factor indicates what percentage of the original points should be returned or that the resulting polygon should be based on. The polygon or set of points may contain many points. The simplification factor is a decimal number between 0 and 1.0

(1 being 100 percent and 0.01 being 1 percent). Lower numbers mean lower storage and lower transmission times. This setting will only apply if geometry is the result type. If this property is not specified, the default setting 0.05 (5 percent) will be used. The images below illustrate the behavior of the simplification factor option. The image on the left shows the simplification factor set to 0.01 while the image on the right shows the simplification factor set to 1.





- **TimeOut** - Specifies the amount of time allowed for the server to create the isogram. The default value for this property is 600 seconds (10 minutes).

Creating an IsoDistance (Drive-Distance)

When creating an isoDistance, you are looking for one or more polygons/set of nodes which represent the area that can be travelled along a road network, given a distance (drive-distance cost) from a starting location.

The cost specified for an isoDistance request is similar to an isoChrone except the cost value is a measure of distance used to calculate the isogram. The cost represents the distance required to travel from the starting point to the calculated points on the road network. You can specify multiple costs to produce concentric bands (multiple isoDistances) that visually represent different distances that can be reached along the road network. Associated with multiple costs are tags. You can specify a tag, or ID, for each cost in the request, that will identify the appropriate iso result (geometry) in the response.

The following code sample shows how to define an IsoDistanceDefinition. The response will create an isogram (geometry) using the point as the center and a cost in miles (`pt, DistanceUnit.Mile`) defining the boundary:

```
Public Shared Sub New_IsoDistanceDefinition()
    ' Create a point
```

```
    Dim coordSys As MapInfo.Geometry.CoordSys = _
        Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 42)
    Dim pt As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt)
    ' Create a definition
    Dim def As IsoDistanceDefinition = _
        New IsoDistanceDefinition(pt, DistanceUnit.Mile)
End Sub
```

There are numerous preferences you can specify in the IsoDistancePreferences and IsogramPreferences Classes. These preferences allow you to obtain a desired output for your analysis. The preferences specified in the IsogramPreferences Class are the same for both an isoDistance and isoChrone request. For a description of the IsogramPreferences, refer to IsogramPreferences.

IsoDistancePreferences

The following preferences are available in the IsoDistancePreferences Class:

- **DefaultPropagationFactor** - Determines the off-road network percentage of the remaining cost (distance) for which off network travel is allowed when finding the isoDistance boundary. Roads not identified in the network can be driveways or access roads, among others. The propagation factor is a percentage of the cost used to calculate the distance between the starting point and the isoDistance. For example, if you were at a point with five miles left to go on an isoDistance on the off-road network, boundary points would be put at a distance based on the propagation factor and the distance left. So, if the propagation factor was 0.16, boundary points would be put at a distance of 0.8 miles.
- **PropagationFactorOverrides** - Overrides the propagation factor for a specific road type. The propagation factor can be overridden for all road types. For example, it may be set to 0.24 for major urban roads:

Updating a Request Using Routing Data

Being able to access and your route data, then use this information to create more accurate and meaningful routes, is a very useful capability. After you create a route and get a route response, you may realize that the result contains a location or segment you don't want to pass, and a road type or segment where you may want to reduce or increase the speed. MapXtreme provides capabilities to return segment information on

your route data and use the information in your route data to modify/update route requests (transient updates). This means you can update the routing network without reprocessing the base data.

The transient update process is comprised of two steps:

1. Return the route data segment information in a route response. This information is required to determine the segments used for the road type or speed update. For more information, refer to Returning Segment Information.
2. Based on information returned in your response, make temporary updates to the route data. This is called transient updates. These changes are submitted within a route request, and are only valid when the server or service is handling that particular request. This capability can be used to include some dynamic traffic/accident data or to set the preferences for a particular road type. For more information, refer to Transient Updates.

Returning Segment Information

In order to avoid or update the road type or speed data for a road segment you will need to know the segment ID and other road information. MapXtreme provides the ability to return the road segment data including information such as the segment ID, name, distance, road type, speed, travel time, geometries, and other important data.

By default, segment information is not returned in a route request. Use the class, and set ReturnSegmentData to true.You also have the options of defining the units of measure for both the velocity and angle for the returned information. By default the VelocityUnit is Mph (miles per hour) and the AngleUnit is Degree. The segment data is the data that is necessary to create the driving directions and contains detailed information on each segment in the route. The segment information returned includes:

- the street names (including alternate names)
- the languages for the street names
- compass direction
- segment ID
- one way boolean
- roundabout boolean
- toll road boolean
- road type
- speed limit
- turn angle

- time to traverse the segment
- distance along the segment

Transient Updates

In order to create more accurate and meaningful route calculations, you can make temporary changes to the route data. The changes are submitted within each route request and are only valid when the server or service is handling that particular request. You can use this feature to avoid a particular highway during rush hour or lower the speed of the road segment in the request, representing real life traffic patterns, producing more accurate time based routing. A transient update can be included in any type of route request (point-to-point, multi-point, or matrix route). By making these types of modifications, you have the ability to:

- set the speed of a point, segment, or road type.
- change (increase or decrease) the speed of a point, segment, or road type by some value.
- change (increase or decrease) the speed of a point, segment, or road type by some percentage.
- set the road type for a segment.

When defining speeds in transient updates, there are essentially three types, each speed value defined differently:

- Speed - specifies the new speed of the segment.
- Relative - specifies an increase or decrease for the speed by a relative value. These values represent a change in speed. For example a value of 10, will increase the default speed by 10, while a value of -10 MPH will decrease the default speed by 10 MPH.
- Percentage - specifies an increase or decrease for the speed by a percentage. Values are between -100 and 100. For example a speed of 50 would increase the speed by 50 percent of the default speed (a speed of 30 MPH would increase to 45 MPH), while a speed of -50 would decrease the speed by 50 percent of the default speed (a speed of 30 MPH would decrease to 15 MPH).

Using Transient Update to Modify a Segment

After you get a route response, you may realize that the result contains a particular route segment or group of segments (roads) where you know traffic is slower or faster, or you may want to update the road type in a request. You can use a transient update to set the speed of the road segments or road types, recalculating the route for more accurate results. Use the following classes to perform the various segment transient updates:

- **SegmentRoadTypeUpdate** - Updates the road type for the segment. The following code sample shows how to specify a new road type for the defined segment ID "S1256", RoadType.MAJOR_ROAD_URBAN
- **SegmentSpeedUpdate** - Updates the speed of the segment with a new speed.

The following sample shows how to define a new speed of 50 miles per hour (Velocity(50, VelocityUnit.Mph)) for the specified route segment (S1256):

```
Public Shared Sub New_SegmentSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
    ' Create the update object
    Dim update As SegmentSpeedUpdate = New SegmentSpeedUpdate("S1256", velocity)
End Sub
```

- **SegmentRelativeSpeedUpdate** - Updates the speed of the segment with a change in speed.

The following sample shows how to increase the speed by 5 miles per hour (Velocity(5, VelocityUnit.Mph)) for the specified route segment (T1256):

```
Public Shared Sub New_SegmentRelativeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
    ' Increase the speed by 5 mph.
    Dim update As SegmentRelativeSpeedUpdate = New
SegmentRelativeSpeedUpdate("T1256", velocity)
End Sub
```

- **SegmentPercentageSpeedUpdate** - Updates the speed of the segment with percentage of the default speed.

The following sample shows how to increase the speed for the specified route segment by 20 percent ("S1256", 20):

```
Public Shared Sub New_SegmentPercentageSpeedUpdate()
    ' Increase the speed by 20 percent.
    Dim update As SegmentPercentageSpeedUpdate = New
SegmentPercentageSpeedUpdate("S1256", 20)
End Sub
```

Using Transient Update to Modify a Point

After you get a route response, you may realize that the result contains some locations (points) where you know traffic is slower or faster. You can use a transient update to set the speed of the road segments closest to the point, recalculating the route for more accurate results. Use the following classes to perform the various point transient updates:

- **PointSpeedUpdate** - Updates the speed of the closest segment to the point with a new speed.

The following sample shows how to define a new speed of 50 miles per hour (`Velocity(50, VelocityUnit.Mph)`) for the road closest to the defined point (`pt`):

```
Public Shared Sub New_PointSpeedUpdate()
    ' Create a point
    Dim coordSys As MapInfo.Geometry.CoordSys =
Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
    Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
    ' Create the update object
    Dim update As PointSpeedUpdate = New PointSpeedUpdate(pt, velocity)
End Sub
```

- **PointRelativeSpeedUpdate** - Updates the speed of the closest segment to the point with a change in speed.

The following sample shows how to increase the speed by 5 miles per hour (`Velocity(5, VelocityUnit.Mph)`) for the road closest to the defined point (`pt`):

```
Public Shared Sub New_PointRelativeSpeedUpdate()
    ' Create a point
    Dim coordSys As MapInfo.Geometry.CoordSys =
Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
    Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
    ' Increase the speed by 5 mph.
    Dim update As PointRelativeSpeedUpdate = New PointRelativeSpeedUpdate(pt,
velocity)
End Sub
```

- **PointPersentageSpeedUpdate** - Updates the speed of the closest point to the location by a percentage of the default speed.

The following sample shows how to increase the speed for the road closest to the define point by 20 percent (`pt, 20`):

```
Public Shared Sub New_PointPercentageSpeedUpdate()
    ' Create a point
    Dim coordSys As MapInfo.Geometry.CoordSys =
Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
    Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
    ' Increase the speed by 20 percent.
    Dim update As PointPercentageSpeedUpdate = New
PointPercentageSpeedUpdate(pt, 20)
End Sub
```

Using Transient Update to Modify Road Type Speeds

Before determining a route, you may already know some travel information (for example. construction, accident, or rush hour), and want to reduce or increase the speed of a particular road type. For example, you can lower the speed for all highways. Taking into consideration holiday weekend travel. You also have the option to lower the chance of a particular road type being used in the route calculation. When determining the fastest route, you can decrease the travel speed for a particular road type which will set those roads to a lower priority, and decrease the chance of them being used in the calculation. Use the following classes to perform the various road type transient updates:

- **RoadTypeSpeedUpdate** - Updates the speed of a particular road type with a new speed.

The following sample shows how to define a new speed of 50 miles per hour (`Velocity(50, VelocityUnit.Mph)`) for all urban major roads (`RoadType.MAJOR_ROAD_URBAN`):

```
Public Shared Sub New_RoadTypeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
    ' Create the update object
    Dim update As RoadTypeSpeedUpdate = New
RoadTypeSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, velocity)
End Sub
```

- **RoadTypeRelativeSpeedUpdate** - Updates the speed of the road type with a change in speed.

The following sample shows how to increase the speed by 5 miles per hour (`Velocity(5, VelocityUnit.Mph)`) for all urban major roads (`RoadType.MAJOR_ROAD_URBAN`):

```
Public Shared Sub New_RoadTypeRelativeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
    ' Increase the speed by 5 mph.
    Dim update As RoadTypeRelativeSpeedUpdate = New
RoadTypeRelativeSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, velocity)
End Sub
```

- **RoadTypePercentageSpeedUpdate** - Updates the speed of the road type with percentage of the default speed.

The following sample shows how to increase the speed for all urban major roads by 20 percent (`RoadType.MAJOR_ROAD_URBAN, 20`):

```
Public Shared Sub New_RoadTypePercentageSpeedUpdate()
    ' Increase the speed by 20 percent.
    Dim update As RoadTypePercentageSpeedUpdate = New
RoadTypePercentageSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, 20)
End Sub
```

22 – Linear Referencing

This chapter covers a MapXtreme capability for mapping and analyzing linear networks using M (measure) values associated with MultiCurve feature geometries.

In this chapter:

What is Linear Referencing

Linear referencing is an alternative reference system to the traditional coordinate reference systems that tie locations of linear features to points on the earth. Linear referencing is used in many fields, including water resource management, transportation, and oil and gas exploration. Any physical asset that you can map as part of a linear network can hold data that describes the asset or a condition or event related to that asset. The data is stored as an M, or measure value, on the MultiCurve object along with the X and Y coordinates for the location. The M values can then be further mapped and analyzed for better resource management.

M values are the cornerstone of linear referencing. M values hold the measure, whatever it may be. Points along a linear feature are referenced from an established known point that is relative to something else. A classic example is the mile marker post along a highway. The ID of the mile marker is some M value that refers to some distance from a known location, typically a highway intersection or county boundary.

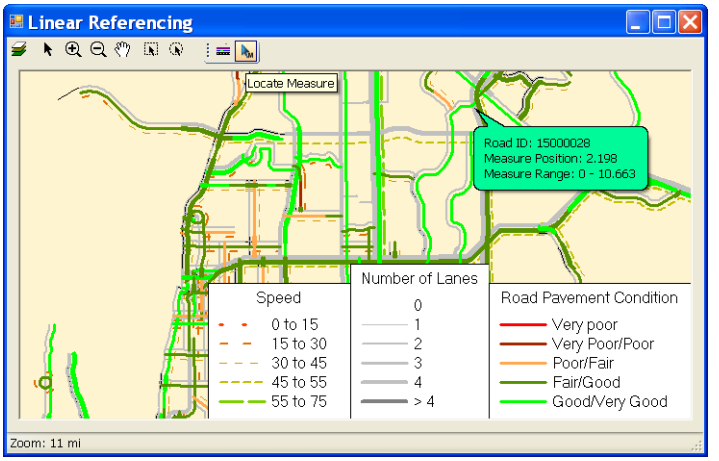
M values are commonly used by infrastructure and transportation data managers to better visualize, query, monitor and analyze assets, conditions and events that exist along a line. For example, an emergency call center operator can determine the location of a stranded motorist from the mile marker the motorist provides to the operator. The mile marker location is already known to the call center. With this information in hand, the call center can dispatch the appropriate personnel to the proper location.

For an overview of the Geometry model that supports MultiCurves and M values, see Chapter 16 Spatial Objects and Coordinate Systems.

Using M values for Linear Referencing

MapXtreme offers a number of operations that use the measure (M) value for MultiCurve geometries to provide linear referencing and dynamic segmentation capabilities. The methods are defined as part of an instantiable MapInfo.Geometry.LinearReferencingOperations class, which provides flexibility for adding properties or extending behaviors as needed.

These operations allow you to show simultaneous multiple attributes of your linear network, known as dynamic segmentation, without requiring the storage of multiple copies of the same geometry. For example, a highway can be represented by separate linear features that represent pavement condition, lane type, and pavement material. They are all representing the same real-world geometry, but this gives you an easier view of multiple types of information.



Linear Referencing Sample Application provided in MapXtreme

The linear referencing methods follow the OGC Simple Feature Specification as it applies to one dimensional MultiCurve geometries. MultiCurves can consist of several disjointed curves. That is, two Curves can be contained within the MultiCurve for which the end of one Curve does not match the beginning of the other Curve. The operations, however, may be given a start and end measure with the expectation that the measure is applied across the entire MultiCurve. In this and similar scenarios, we assume that even though the actual location data differs (X, Y) between the end of one curve and the start of another curve, MapXtreme assumes the measure values are the same. Consider as an example, a highway network in which some highways cross but do not intersect because one highway runs over the other on a bridge. In those cases, the highway underneath may actually have the linear representation show a physical break before and after the other highway.

The methods provided in the LinearReferencingOperations class can be organized into three categories:

- Measure value determination methods
- Linear referencing operations
- Dynamic Segmentation operation (PerpendicularOffset)

The following sections provide a brief introduction to these operations. See also MapInfo.LinearReferencing namespace for compete details.

Measure Value Determination Methods

The operations that form this category are helper methods to assist in setting and managing measure values on a MultiCurve. Use them to fill in missing nodes along a MultiCurve when you have at least two M values, or if no M values are present, using a relative start and end node to set M values. Often the values are known for one or two specific locations and the rest can be proportionally computed based on distance.

- CalculateMissingMeasures(MultiCurve)
- SetMeasures (MultiCurve, double startMeasure, double endMeasure)
- SetMeasuresAsDistance(MultiCurve, double startDistance)
- DropMeasures (MultiCurve)
- ScaleMeasures(MultiCurve, double scale)
- TranslateMeasures(MultiCurve, double offset)
- TranslateMeasures(MultiCurve)
- Reverse(MultiCurve)

Linear Referencing Operations

The second set of operations use a linear referencing system along a MultiCurve geometry to locate points at a specific measure value or extract sub-curves between two measure values. While MapXtreme can store reference measure values at the nodes of the MultiCurve that is input into these methods, not all user data actually has the reference system stored in the geometry (or they are using a storage technology that does not facilitate retaining this information). These methods, therefore, generally have two overloaded forms, one in which the reference system is managed within the input MultiCurve, and another in which the reference system is supplied as a start and end measure corresponding to the first and last node of the input MultiCurve geometry.

- LocateAlong(MultiCurve, double Measure)
- LocateAlong(MultiCurve, double startMeasure, double endMeasure, double Measure)
- LocateMeasure (MultiCurve, Point)
- LocateMeasure (MultiCurve, double startMeasure, double endMeasure, Point)
- LocateBetween(MultiCurve, double subCurveStartMeasure, double subCurveEndMeasure)
- LocateBetween(MultiCurve, double startMeasure, double endMeasure, double subCurveStartMeasure, double subCurveEndMeasure)

FeatureGeometry.Distance Method

The MapInfo.Geometry.FeatureGeometry class includes an overload to the Distance method that MapXtreme uses to locate measure values along a MultiCurve when the input point is not located on the curve. This method returns a MultiCurve that represents the shortest or longest distance between the input point off the curve and the projected point on the curve. To specify shortest or longest, the minimumDistance must be passed in as a boolean. Shortest distance is true. The returned MultiCurve is the input for the LocateMeasure methods which returns a point on the MultiCurve closest (or farthest) from the input point.

If the MultiCurve and the Point that is passed in are in different coordinate systems, then the Distance operation is performed in the coordinate system of MultiCurve instance. The resulting FeatureGeometry is always in the same coordinate system as MultiCurve instance.

Dynamic Segmentation Operation (PerpendicularOffset)

Perpendicular offset is an operation that produces a new MultiCurve from an existing one. This is different from the above methods that extract a sub-curve with the same underlying x,y coordinates. This operation creates a parallel offset MultiCurve that is some distance measured perpendicularly from the original MultiCurve. This new MultiCurve can then be further acted upon, such as querying its attributes and analyzing the results.

PerpendicularOffset is used in the field of dynamic segmentation for a linear network whereby you can subdivide a MultiCurve into segments based on M values. For example, a road construction application would use the M values of a road network that describe the status of sections of a highway under construction or repair. Each segment can be displayed on a map, offset from the actual road, with a distinct color code or marking to indicate at a glance its status. For an example of a dynamic segmentation application using Perpendicular, see Linear Referencing Sample Application.

The following is the syntax of the PerpendicularOffset() method:

- PerpendicularOffset(MultiCurve, double distance)
- PerpendicularOffset(MultiCurve, double distance, DistanceUnit, int resolution)

A distance value that is positive will be interpreted as a perpendicular direction to the right of the line when traversing in the order of the nodes (e.g., node i to node i+1). A negative distance would be to the left. A distance of zero is not allowed.

The resolution is used to interpolate points along the arcs created for rounding corners. Values of 0 or larger are supported, up to an including 36. The default value is 1.

This operation preserves the measure values, if present, of the original MultiCurve. Elevation (Z) values from the original MultiCurve are not preserved.

Curve Order

The MapInfo.LinearReferencing namespace includes the ICurveSorter interface to handle the sort order of individual curves that make up a MultiCurve. Sort order is important when you are calling any of the following linear referencing operations:

- CalculateMissingMeasures
- SetMeasures
- SetMeasuresAsDistance
- LocateAlong
- LocateMeasure
- LocateBetween

Without specifying a sort order, MapXtreme returns the longest curve first, while the remaining curves are returned in an unknown order. When using CalculateMissingMeasures on an unordered MultiCurve, for example, MapXtreme could calculate the wrong M values for a node based on its position in the MultiCurve. Providing the correct sort order would eliminate that problem.

The ICurveSorter interface provides a SortCurves method which returns the curve order as an array of integers. You would then use the array to get to the list of curves. See the Developer Reference for a code example.

ⓘ Ordering of curves does not change the x,y position of the nodes.

The interface also provides a DefaultCurveSorter class that, when implemented, only returns the component curves as advertised by the MultiCurve itself. No ordering is done through its SortCurve method.

Linear Referencing Sample Application

Provided in the Samples folder of your MapXtreme installation is a linear referencing sample application that generates multiple parallel line segments representing different characteristics of the road. In this example, road data that indicates pavement conditions,

maximum speed zones and number of lanes are displayed as offset MultiCurves along the actual roadway to show the condition or characteristic of a particular road section. Each characteristic or condition is shown in a different line style.

In a typical MapXtreme table, attributes are applied to the feature as a whole. In linear networks, a feature has attributes, known as Measure values, that apply to segments of its linear geometry representation. For example, a roadway typically has sections of good quality or poor quality surface conditions, or sections that are under repair. These conditions would rarely coincide with the entire road segment, nor do they coincide exactly with a node on the MultiCurve. However, through a linear referencing system that uses measure values that are relative to the feature, rather than to the location on earth, segment data can be captured, queried, displayed and analyzed in new and useful ways.

Through its linear referencing operations, MapXtreme can dynamically segment MultiCurves based on these M values. See Using M values for Linear Referencing for a description of the supported operations.

23 – Web Feature Service

MapXtreme provides the ability to host and/or access map feature data from internet or private intranet Web Feature Services (WFS).

In this chapter:

Web Feature Service

MapXtreme provides a Web Feature Service (WFS) implementation to send requests over the Internet or through a private intranet to retrieve geospatial data encoded in Geography Markup Language. While a Web Map Service yields a map image, requests to a WFS Server will generate GML, a form of XML that can capture geographic data.

With version 1.0.0, a basic WFS client can send three kinds of requests to a WFS server. A get capabilities request asks a server to list the geographic data it can provide and the operations that can be applied to that data. A describe feature type request asks a server to describe the data it can provide for a geographic feature. Finally the server can be asked to provide the actual data using GetFeature request.

In addition to this, a WFS client can send two additional requests to a WFS 2.0 server i.e., ListStoredQueries and DescribeStoredQueries. A ListStoredQueries operation returns a list of the stored queries currently maintained by the WFS server. A DescribeStoredQuery operation returns detailed metadata about each stored query maintained by the WFS server. These two operations are supported in WFS 2.0.0 version only.

Geographic data from a WFS consists of descriptions about the data. At this point the retrieved data is not viewable as a map layer in MapXtreme. However, using MapXtreme's WfsClient, the GML output can be converted into a form MapXtreme can work with: MultiFeatureCollections. These feature collections can then be treated like any other FeatureCollection in MapXtreme whereby you can apply themes, labels and perform a variety of analytical operations.

The MapXtreme WFS 1.0.0 server implementation complies with the Basic WFS conformance class and WFS 2.0.0 server implementation complies with the Simple WFS conformance class. MapXtreme's WFS implementation does not support the Transaction WFS specification at this time.

Requests for features via MapXtreme's WFS implementation are made with HTTP GET or HTTP POST requests. The response is returned in GML2 by default; however, a request can explicitly ask for a response in GML3.

MapXtreme's support for WFS consists of two parts—a WFS Server and WFS Client. This chapter explains how to configure a WFS server if you want to host your own data for others to access. See Configuring a WFS 2.0.0 Server.

If you are interested in accessing data from other WFS servers on the internet or from a private intranet, see Using the MapXtreme WFS Client Programmatically.

Understanding WFS 1.0.0 Server Operations

There are three WFS Server operations that provide the basis for the MapXtreme WFS server implementation: GetCapabilities, DescribeFeatureType, and GetFeature.

GetCapabilities

A GetCapabilities request is a query of a WFS server to learn more about what the server offers in terms of geographic data and operations that can be performed on that data. The response to a GetCapabilities request is an XML document describing the operations that the WFS supports and a list of all feature types that it can service. You would request the service's capabilities the first time you access a WFS server.

GetCapabilities is supported via HTTP GET and HTTP POST.

The following is the XML output from a GetCapabilities request. Some of the key elements are shown in **bold text**, including supported capabilities, available feature types, and filtering operations for requesting a subset of a feature type's data.

```
<?xml version="1.0" encoding="utf-8" ?>
- <WFS_Capabilities xmlns:ogc="http://www.opengis.net/ogc"
xmlns:miwfs="http://www.mapinfo.com/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd"
version="1.0.0" updateSequence="0" xmlns="http://www.opengis.net/wfs">
- <Service>
  <Name>Sample WFS Server</Name>
  <Title>Sample WFS Server</Title>
  <OnlineResource>http://localhost/Wfs/GetFeature.ashx</OnlineResource>
  </Service>
- <Capability>
- <Request>
- <GetCapabilities>
- <DCPType>
- <HTTP>
  <Get onlineResource="http://localhost/Wfs/GetFeature.ashx" />
  <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
  </HTTP>
  </DCPType>
  </GetCapabilities>
- <DescribeFeatureType>
- <SchemaDescriptionLanguage>
  <XMLSCHEMA />
  </SchemaDescriptionLanguage>
- <DCPType>
- <HTTP>
  <Get onlineResource="http://localhost/Wfs/GetFeature.ashx" />
  <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
  </HTTP>
```

```
  </DCPType>
  </DescribeFeatureType>
- <GetFeature>
- <ResultFormat>
  <GML2 />
  </ResultFormat>
- <DCPType>
- <HTTP>
  <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
  </HTTP>
  </DCPType>
  </GetFeature>
  </Request>
  </Capability>
- <FeatureTypeList>
- <Operations>
  <Query />
  </Operations>
- <FeatureType>
   <Name>miwfs:USA</Name>
  <Title>Title for usa</Title>
  <Abstract>Abstract for USA</Abstract>
  <Keywords>Keywords for USA</Keywords>
  <SRS>epsg:4326</SRS>
  <LatLongBoundingBox minx="-179.62816" miny="18.925255" maxx="-66.951403"
maxy="71.42856" />
  </FeatureType>
  </FeatureTypeList>
- <ogc:Filter_Capabilities>
- <ogc:Spatial_Capabilities>
- <ogc:Spatial_Operators>
  <ogc:BBOX />
  <ogc:Equals />
  <ogc:Disjoint />
  <ogc:Intersect />
  <ogc:Within />
  <ogc:Contains />
  </ogc:Spatial_Operators>
  </ogc:Spatial_Capabilities>
- <ogc:Scalar_Capabilities>
  <ogc:Logical_Operators />
- <ogc:Comparison_Operators>
  <ogc:Simple_Comparisons />
  <ogc:NullCheck />
  </ogc:Comparison_Operators>
- <ogc:Arithmetic_Operators>
  <ogc:Simple_Arithmetic />
  </ogc:Arithmetic_Operators>
  </ogc:Scalar_Capabilities>
  </ogc:Filter_Capabilities>
  </WFS_Capabilities>
```

DescribeFeatureType

After finding the available feature type with GetCapabilities request, the DescribeFeatureType request asks for detailed information about one or more of the available feature types. In MapXtreme a WFS feature type is represented by a table, and a WFS feature is equivalent to a row of information from a table. The response to a DescribeFeatureType request includes the name of the feature type (the name of the table) and the names and types of the properties (names and types of the columns) in the table. The results are returned in an XML schema document in GML format.

DescribeFeatureType is supported via HTTP GET and HTTP POST. The Schema returned will have the form whereby each property in the feature type becomes an element.

MapXtreme does not return the following column types as properties:

- MIDbType.Binary
- MIDbType.CoordSys
- MIDbType.Grid
- MIDbType.Key
- MIDbType.Raster
- MIDbType.Style

The following shows a portion of a DescribeFeatureType response document. In this case, the request is for a feature type called USA. The USA feature type has several properties represented by the geometry column OBJ, and data columns for State and State_Name.

```
<?xml version="1.0" encoding="utf-8" ?>
- <schema targetNamespace="http://www.mapinfo.com/wfs"
xmlns:miwfs="http://www.mapinfo.com/wfs"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" elementFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/2.1.2/feature.xsd" />
  <xs:import namespace="http://www.opengis.net/wfs"
schemaLocation="http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd" />
  <xs:element name="USA" type="miwfs:USA_Type" substitutionGroup="gml:_Feature"
/>
- <xs:complexType name="USA_Type">
- <xs:complexContent>
- <xs:extension base="gml:AbstractFeatureType">
- <xs:sequence>
- <xs:element name="Obj" minOccurs="0" maxOccurs="1">
- <xs:complexType>
- <xs:sequence>
```

```
  <xs:element ref="gml:_Geometry" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  <xs:element name="State" nillable="false" type="xs:string" minOccurs="0"
maxOccurs="1" />
  <xs:element name="State_Name" nillable="false" type="xs:string" minOccurs="0"
maxOccurs="1" />
  </xs:sequence>
  </xs:extension>
  </xs:complexContent>
  </xs:complexType>
  </schema>
```

GetFeature

Once you know the available feature types and their properties, the final step to retrieving WFS feature information is sending a GetFeature request. This request specifies which feature and properties to fetch. In order to retrieve a subset of features, use filtering operations to constrain the query both spatially and/or non-spatially. See Code Example: Requesting Features Using Filters.

GetFeature is supported via HTTP POST.

Among the OGC-supported parameters, Request and TypeName are required. PropertyName, Filter, FeatureID and others are optional. TypeName is optional when FeatureID is specified. For further details, see the OGC Web Feature Service Implementation Specification.

The following is a portion of a sample GetFeatureResponse for a feature type called USA.

```
<?xml version="1.0" encoding="utf-8" ?>
- <wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gml="http://www.opengis.net/gml" xmlns="http://www.mapinfo.com/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs
http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd http://www.mapinfo.com/wfs
http://localhost/Wfs/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVICE=WFS&VE
RSION=1.0.0&TYPENAME=miwfs:USA">
- <gml:boundedBy>
  <gml:null>inapplicable</gml:null>
  </gml:boundedBy>
- <gml:featureMember>
- <USA>
  <State>AK</State>
  </USA>
  </gml:featureMember>
- <gml:featureMember>
- <USA>
  <State>AL</State>
```

```
  </USA>
  </gml:featureMember>
- <gml:featureMember>
- <USA>
  <State>AR</State>
  </USA>
  </gml:featureMember>
- <gml:featureMember>
- <USA>
  <State>AZ</State>
  </USA>
  </gml:featureMember>
- <gml:featureMember>
- <USA>
  <State>CA</State>
  </USA>
  </gml:featureMember>
- <gml:featureMember>
- <USA>
  <State>CO</State>
  </USA>
  </gml:featureMember>
...
</wfs:FeatureCollection>
```

Configuring a WFS 1.0.0 Server

If you have spatial data you wish to make available to others, you first must configure a WFS server to describe what data and capabilities you are offering. There are two configuration files that you need to provide to accomplish this. This discussion assumes you have a working knowledge of schemas and web services.

You do not need to configure a WFS server if you are only interested in accessing someone else's WFS server to retrieve features. See Using the MapXtreme WFS Client Programmatically.

The main configuration steps that are addressed in this section are:

Step1: Create or modify a Web.config file to include the MapXtreme-specific WFS information and the correct handlers for IIS classic or integrated pipeline mode.

Step 2: Create a valid WFS Server configuration file that contains information about the data you are hosting. This file must validate against the WFS schema file (MXP_WFSConfiguration_1_0.xsd) to avoid errors when you run the WFS Server.

Step 3: Configure and test the WFS Server setup. Instructions for IIS 7, IIS 8.5 and IIS 10 configuration are provided.

On the MapXtreme product media, we provide sample Web.config and WFS configuration files that you can use as a guide for creating your own files. The Web.config file defines how the ASP process in handled. The WFSSample.xml defines the data sources and feature definitions you want your WFS server to provide.

The schemas for the MapXtreme workspace and WFS server are also located on the product media.

Step 1: Create a Web.config File

The Web.config is a standard configuration file for a web application. To use it for a MapXtreme WFS server, you must edit it to provide MapXtreme-specific WFS information and to define how the ASP.NET process will be handled.

1. Create a folder to contain the Web.config and the WFSSample.xml. In this example, the location is called **c:\wfs**.
2. Copy the Web.config and WFSSample.xml from the MapXtreme product media to this folder.
3. Open Web.config In a text editor and modify the <appSettings"> line to point to the WFS configuration file.

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

4. **For IIS 7/8.5/10 classic mode**, update the version number and PublicKeyToken (if necessary) for the MapInfo.Wfs.Server and the MapInfo.CoreEngine assemblies installed on your system (**bold type** below).

   Assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 or GAC_64.

```
<system.web>
     <httpHandlers>
        <add verb="GET,POST" path="*.ashx" type="MapInfo.Wfs.WfsHttpHandler,
MapInfo.Wfs.Server, Version=9.2.0.XXX, Culture=neutral,
PublicKeyToken=4ac3224575145b20"/>
     </httpHandlers>
     <httpModules>
        <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.2.0.XXX, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator" />
    </httpModules>
```

5. **For IIS 7/8.5/10 Integrated pipeline mode,** copy the following section into the web.config. You do not need to comment out the <system.web> section to run in

integrated pipeline mode. However, if you need to run in IIS 7 classic mode, you must comment out this <system.webServer> section.

Follow the instructions in step 4 to update the assembly versions for MapInfo.CoreEngine and MapInfo.Wfs.Server.

```
<system.webServer>
      <validation validateIntegratedModeConfiguration="false"/>
      <directoryBrowse enabled="true" />
   <modules>
         <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.2.0.XXX, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator"/>
   </modules>
   <handlers>
         <add name="WFSHandler" verb="GET,POST" path="*.ashx"
type="MapInfo.Wfs.WfsHttpHandler, MapInfo.Wfs.Server, Version=9.2.0.XXX,
Culture=neutral, PublicKeyToken=4ac3224575145b20"/>
   </handlers>
</system.webServer>
```

6. Save this file and copy it to the location you created in step 1.

Step 2: Create a Valid WFS Configuration File for Hosted Features

The WFSSample.xml is a WFS configuration file provided with MapXtreme. This file defines information about your WFS server, including its name, title, abstract, the URL to the WFS Server and the data you want to host.

1. Open WFSSample.xml in a text editor and modify the OnlineResource line to include the URL of your WFS Server. Change `localhost` to what is appropriate for your WFS server.

   ```
   <mxp-wfs:Service>
      <mxp-wfs:Name>Sample WFS Server</mxp-wfs:Name>
      <mxp-wfs:Title>Sample WFS Server</mxp-wfs:Title>
      <!-- The following is the URL of the WFS server -->
      <mxp-wfs:OnlineResource>http://localhost/wfs/GetFeature.ashx</mxp-
wfs:OnlineResource>
   </mxp-wfs:Service>
   ```

   You can modify other elements as you see fit, including the server name, title, abstract, fees, access constraints and more.

2. Register the tables that the WFS serves by creating a Table element for each table. The value for the <mxp-wfs:DataSourceDefinition> id must match the value for <mxp-wfs:Name>. You can include the tables in any order.

3. In the <DataSourceDefinitionSet>, modify the MYPATH variable to reflect the actual path to your data.

   The following is a portion of the WFSSample.xml that identifies two tables of features.

```
<mxp-wfs:Table>
        <mxp-wfs:Name>USA</mxp-wfs:Name>
        <mxp-wfs:Title>Title for usa</mxp-wfs:Title>
        <mxp-wfs:Abstract>Abstract for USA</mxp-wfs:Abstract>
        <mxp-wfs:Keywords>Keywords for USA</mxp-wfs:Keywords>
        <mxp:DataSourceDefinitionSet>
           <mxp:TABFileDataSourceDefinition id="USA">
              <mxp:DataSourceName>USA</mxp:DataSourceName>
              <mxp:FileName>MYPATH\USA.TAB</mxp:FileName>
           </mxp:TABFileDataSourceDefinition >
        </mxp:DataSourceDefinitionSet>
     </mxp-wfs:Table>
     <mxp-wfs:Table>
        <mxp-wfs:Name>US_HIWAY</mxp-wfs:Name>
        <mxp-wfs:Title>Title for US_HIWAY</mxp-wfs:Title>
        <mxp-wfs:Abstract>Abstract for US_HIWAY</mxp-wfs:Abstract>
        <mxp-wfs:Keywords>Keywords for US_HIWAY</mxp-wfs:Keywords>
        <mxp:DataSourceDefinitionSet>
           <mxp:TABFileDataSourceDefinition id="US_HIWAY">
              <mxp:DataSourceName>US_HIWAY</mxp:DataSourceName>
              <mxp:FileName>MYPATH\US_HIWAY.TAB</mxp:FileName>
           </mxp:TABFileDataSourceDefinition >
        </mxp:DataSourceDefinitionSet>
     </mxp-wfs:Table>
```

4. If your WFS will be hosting data that is stored on a RDBMS, specify <ConnectionSet> and <ConnectionMember> elements, following the example below:

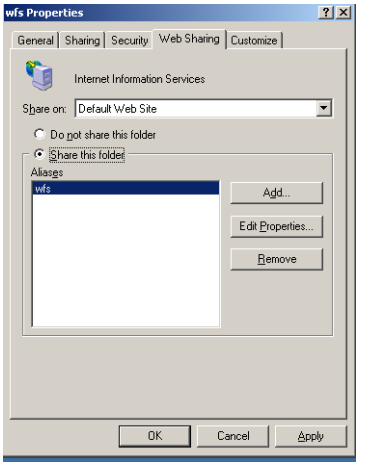
```
   <mxp:ConnectionSet>
      <mxp:DBConnection dbType="sqlserver">
       <mxp:ConnectionName>sqlserver2k</mxp:ConnectionName>
       <mxp:ODBCConnectionString>DRIVER={SQL
Server};DATABASE=YOURDB;Server=YOURSERVER;UID=YOURUSER;PWD=YOURPASSWORD;QuotedI
D=No;Trusted_Connection=No;Network=DBMSSOCN;Address=YOURSERVER,YOURSERVERPORT</
mxp:ODBCConnectionString>
     </mxp:DBConnection
   </mxp:ConnectionSet>
...
     <mxp-wfs:Table>
        <mxp-wfs:Name>MySQLServerTable</mxp-wfs:Name>
        <mxp-wfs:Title>Title for MySQLServerTable</mxp-wfs:Title>
        <mxp-wfs:Abstract>Abstract for MySQLServerTable</mxp-wfs:Abstract>
        <mxp-wfs:Keywords>Keywords for MySQLServerTable</mxp-wfs:Keywords>
        <mxp:DataSourceDefinitionSet>
           <mxp:DBDataSourceDefinition id="MySQLServerTable">
              <mxp:DataSourceName>MySQLServerTable</mxp:DataSourceName>
```

```
        <mxp:ConnectionMember>
<mxp:ConnectionName>my_sqlserver2000_advserver</mxp:ConnectionName>
        </mxp:ConnectionMember>
        <mxp:DBQuery>
           <mxp:Query>select * from MySQLServerTable</mxp:Query>
        </mxp:DBQuery>
        <mxp:DBDataSourceMetadata/>
     </mxp:DBDataSourceDefinition>
  </mxp:DataSourceDefinitionSet>
</mxp-wfs:Table>
```

Step 3: Configuring and Testing the WFS Server

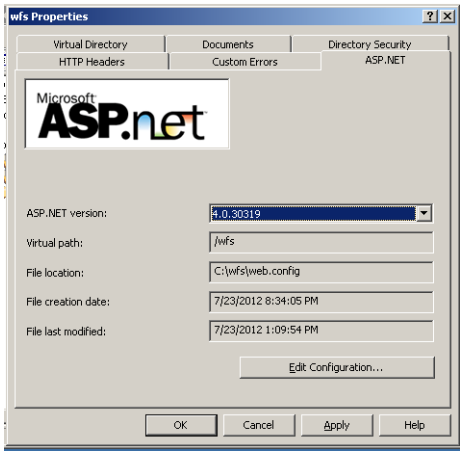
Once you have edited the Web.config and WFSSample.xml files, you must register your WFS server with Internet Information Services and finally, test your setup.

1. Right-click on your WFS folder (for example, c:\wfs) and choose Sharing and Security. From the Web Sharing tab, choose the Share this folder radio button. If you wish to set an alias for your web server, click the Add button.

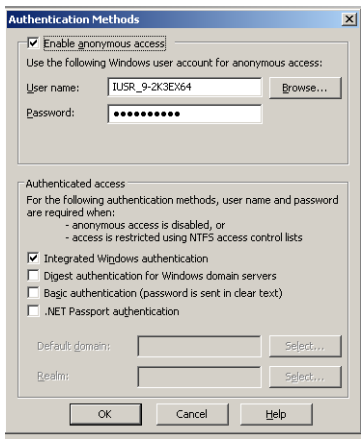


2. Open IIS (From the Start menu > Control Panel > Administrative Tools > Internet Information Services). Expand the Default Web Site and locate your WFS server (by folder name or by alias, if you set one).

3. Right-click on the Web site and choose Properties. Under the ASP.NET tab, choose 4.0.30319 from the drop-down list for the ASP.NET version (The MapXtreme 9.2 assemblies are compiled under the v4.7.2 Framework).



4. In the same Properties dialog, under the Directory Security tab, click the Edit button at the top right. In the Authentication Method dialog box, select the Anonymous Access check box. This allows users of your WFS service to skip the username/password authentication process. Click OK twice and close the IIS window.



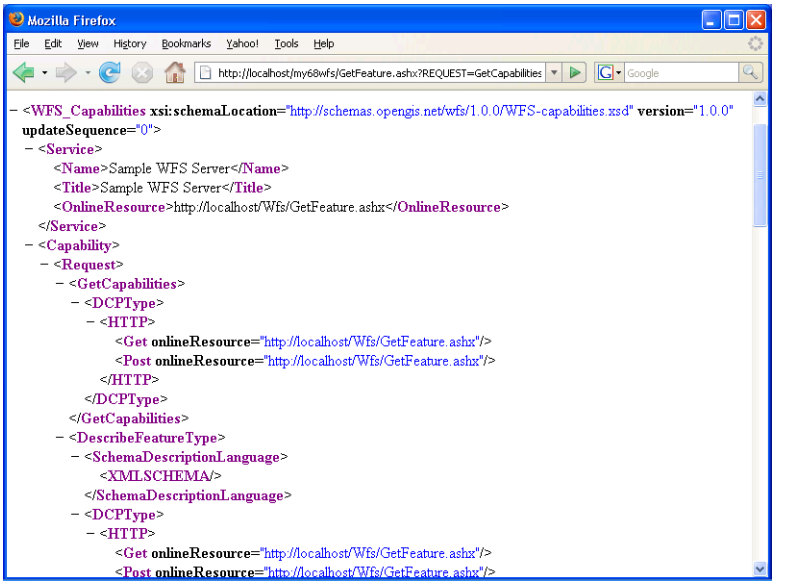
5. Test your setup by sending a GetCapabilities request from a web browser. In the address box type:

   ```
   http://localhost/wfs/GetFeature.ashx?request=GetCapabilities&service=WFS&version=1.0.0
   ```

   substituting your web server for `localhost`. If you have set an Alias to your web server, be sure to include that in your URL. For example:

   ```
   http://localhost/My_WFS/GetFeature.ashx?request=GetCapabilities&service=WFS&version=1.0.0
   ```

A successful test will return a web page similar to the illustration below. If the capabilities are not returned, review your configuration files to ensure everything has been entered correctly. Since you are creating the configuration files by hand, It is very easy to include typos or have missing tags.



6. To learn about the properties for the returned feature types, send a DescribeFeatureType request:

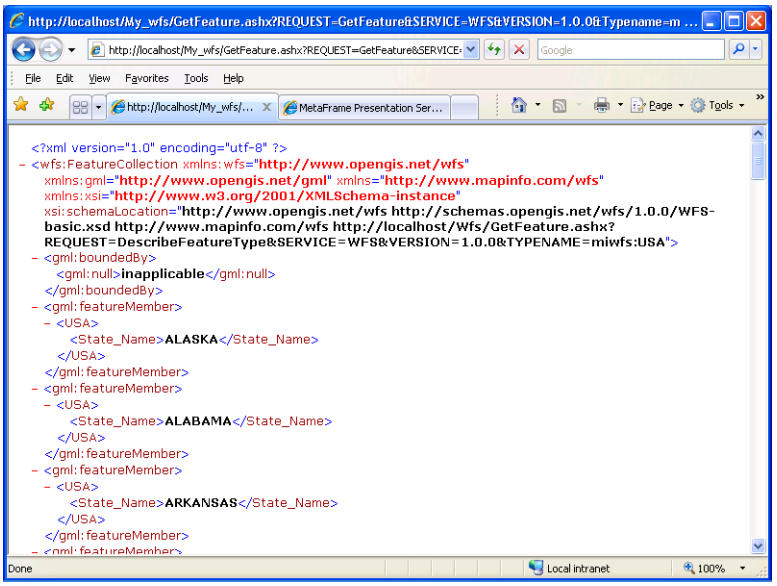
   `http://localhost/My_wfs/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVICE=WFS&VERSION=1.0.0&Typename=miwfs:USA`

   This request returns a description of the properties for the feature type USA, including two column names: State and State_Name.

7. To request features from the USA table, send a GetFeature request.

   `http://localhost/My_wfs/GetFeature.ashx?REQUEST=GetFeature&SERVICE=WFS&VERSION=1.0.0&Typename=miwfs:USA&propertyname=miwfs:State_Name`

   This request returns features from the USA table as a FeatureCollection. Notice in the URL that we requested only one of the two column properties, State_Name, for the USA table. If we had not specified any property name, all properties in the table would be returned.



Understanding WFS 2.0.0 Server Operations

There are five WFS Server operations that provide the basis for the MapXtreme WFS server implementation: GetCapabilities, DescribeFeatureType, GetFeature, ListStoredQueries, and DescribeStoredQueries.

GetCapabilities

A GetCapabilities request is a query of a WFS server to learn more about what the server offers in terms of geographic data and operations that can be performed on that data. The response to a GetCapabilities request is an XML document describing the operations that the WFS supports and a list of all feature types that it can service. You would request the service's capabilities the first time you access a WFS server.

GetCapabilities is supported via HTTP GET and HTTP POST.

The following is the XML output from a GetCapabilities request. Some of the key elements are shown in **bold text**, including supported capabilities, available feature types, and filtering operations for requesting a subset of a feature type's data.

```
<wfs:WFS_Capabilities xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns="http://www.opengis.net/wfs/2.0"xmlns:gml="http://www.opengis.net/gml/3.2
" xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:mxp="http://www.mapinfo.com/mxp/wfs"
```

```
xmlns:xlink="http://www.w3.org/1999/xlink"xmlns:miwfs="http://www.mapinfo.com/w
fs" xmlns:cdf="http://www.opengis.net/cite/data"
xmlns:mxt="http://www.mapinfo.com/mxt"
xmlns:cgf="http://www.opengis.net/cite/geometry"xmlns:xsi="http://www.w3.org/20
01/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs/2.0
http://schemas.opengis.net/wfs/2.0/wfs.xsd" version="2.0.0" updateSequence="0">
<ows:ServiceIdentification>
   <ows:Title>Sample WFS Server</ows:Title>
   <ows:ServiceType>WFS</ows:ServiceType>
   <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
</ows:ServiceIdentification>
<ows:ServiceProvider>
   <ows:ProviderName>Sample WFS Server</ows:ProviderName>
</ows:ServiceProvider>
<ows:OperationsMetadata>
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://localhost/WFS/GetFeature.ashx"/>
            <ows:Post xlink:href="http://localhost/WFS/GetFeature.ashx"/>
         </ows:HTTP>
      </ows:DCP>
   <ows:Parameter name="AcceptVersions">
      <ows:AllowedValues>
         <ows:Value>2.0.0</ows:Value>
      </ows:AllowedValues>
   </ows:Parameter>
</ows:Operation>
<ows:Operation name="DescribeFeatureType">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://localhost/WFS/GetFeature.ashx"/>
            <ows:Post xlink:href="http://localhost/WFS/GetFeature.ashx"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="outputFormat">
         <ows:AllowedValues>
            <ows:Value>application/gml+xml; version=3.2</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
</ows:Operation>
<ows:Operation name="GetFeature">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://localhost/WFS/GetFeature.ashx"/>
            <ows:Post xlink:href="http://localhost/WFS/GetFeature.ashx"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="outputFormat">
         <ows:AllowedValues>
            <ows:Value>application/gml+xml; version=3.2</ows:Value>
         </ows:AllowedValues>
```

```
      </ows:Parameter>
</ows:Operation>
<ows:Operation name="ListStoredQueries">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://localhost/WFS/GetFeature.ashx"/>
            <ows:Post xlink:href="http://localhost/WFS/GetFeature.ashx"/>
         </ows:HTTP>
      </ows:DCP>
</ows:Operation>
<ows:Operation name="DescribeStoredQueries">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://localhost/WFS/GetFeature.ashx"/>
            <ows:Post xlink:href="http://localhost/WFS/GetFeature.ashx"/>
         </ows:HTTP>
      </ows:DCP>
</ows:Operation>
<ows:Constraint name="ImplementsSimpleWFS">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="KVPEncoding">
   <ows:NoValues/>
   <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
<ows:Constraint name="XMLEncoding">
      <ows:NoValues/>
      <ows:DefaultValue>TRUE</ows:DefaultValue>
   </ows:Constraint>
</ows:OperationsMetadata>
<wfs:FeatureTypeList>
<wfs:FeatureType xmlns:miwfs="http://www.mapinfo.com/wfs">
   <Name>miwfs:Deletes</Name>
   <Title>Title for Deletes</Title>
   <Abstract>Abstract for Deletes</Abstract>
   <ows:Keywords>
      <ows:Keyword>Fifteen</ows:Keyword>
   </ows:Keywords>
   <wfs:DefaultCRS>urn:ogc:def:crs:EPSG::32615</wfs:DefaultCRS>
   <ows:WGS84BoundingBox>
      <ows:LowerCorner>-92.999549019421679 4.5240149772284033</ows:LowerCorner>
      <ows:UpperCorner>-92.999549019421679 4.5240149772284033</ows:UpperCorner>
   </ows:WGS84BoundingBox>
</wfs:FeatureType>
</wfs:FeatureTypeList>
<fes:Filter_Capabilities>
   <fes:Conformance>
      <fes:Constraint name="ImplementsQuery">
         <ows:NoValues/>
         <ows:DefaultValue>TRUE</ows:DefaultValue>
      </fes:Constraint>
```

```
      <fes:Constraint name="ImplementsSpatialFilter">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsMinTemporalFilter">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsTemporalFilter">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsMinSpatialFilter">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsFunctions">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
      <fes:Constraint name="ImplementsExtendedOperators">
         <ows:NoValues/>
         <ows:DefaultValue>FALSE</ows:DefaultValue>
      </fes:Constraint>
   </fes:Conformance>
</fes:Filter_Capabilities>
</wfs:WFS_Capabilities>
</WFS_Capabilities>
```

DescribeFeatureType

After finding the available feature type with GetCapabilities request, the DescribeFeatureType request asks for detailed information about one or more of the available feature types. In MapXtreme a WFS feature type is represented by a table, and a WFS feature is equivalent to a row of information from a table. The response to a DescribeFeatureType request includes the name of the feature type (the name of the table) and the names and types of the properties (names and types of the columns) in the table. The results are returned in an XML schema document in GML format.

DescribeFeatureType is supported via HTTP GET and HTTP POST. The Schema returned will have the form whereby each property in the feature type becomes an element.

MapXtreme does not return the following column types as properties:

- MIDbType.Binary
- MIDbType.CoordSys
- MIDbType.Grid

- MIDbType.Key
- MIDbType.Raster
- MIDbType.Style

The following shows a portion of a DescribeFeatureType response document. In this case, the request is for a feature type called USA. The USA feature type has several properties represented by the geometry column OBJ, and data columns for State and State_Name.

```
<schema xmlns:miwfs="http://www.mapinfo.com/wfs"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns="http://www.w3.org/2001/XMLSchema"targetNamespace="http://www.mapinfo.com
/wfs" attributeFormDefault="unqualified" elementFormDefault="qualified">
<xs:import namespace="http://www.opengis.net/cite/data"
schemaLocation="http://localhost/WFS/GetFeature.ashx?REQUEST=DescribeFeatureTyp
e&SERVICE=WFS&VERSION=2.0.0&TYPENAMES=cdf:Fifteen"/>
<xs:import namespace="http://www.mapinfo.com/mxt"
schemaLocation="http://localhost/WFS/GetFeature.ashx?REQUEST=DescribeFeatureTyp
e&SERVICE=WFS&VERSION=2.0.0&TYPENAMES=mxt:Other"/>
<xs:import namespace="http://www.opengis.net/cite/geometry"
schemaLocation="http://localhost/WFS/GetFeature.ashx?REQUEST=DescribeFeatureTyp
e&SERVICE=WFS&VERSION=2.0.0&TYPENAMES=cgf:Points"/>
<xs:import namespace="http://www.opengis.net/gml/3.2"
schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
<xs:import namespace="http://www.opengis.net/wfs/2.0"
schemaLocation="http://schemas.opengis.net/wfs/2.0/wfs.xsd"/>
<xs:element name="Deletes" type="miwfs:Deletes_Type"
substitutionGroup="gml:AbstractFeature"/>
<xs:complexType name="Deletes_Type">
<xs:complexContent>
<xs:extension base="gml:AbstractFeatureType">
<xs:sequence>
<xs:element name="id" nillable="false" type="xs:string" minOccurs="0"
maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</schema>
```

GetFeature

Once you know the available feature types and their properties, the final step to retrieving WFS feature information is sending a GetFeature request. This request specifies which feature and properties to fetch. GetFeature is supported via HTTP POST and HTTP GET.

The following is a portion of a sample GetFeatureResponse for a feature type called USA.

```
<cdf:Fifteen xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.opengis.net/cite/data"xmlns:xsi="http://www.w3.org/2001/XMLSc
hema-instance" xmlns:cdf="http://www.opengis.net/cite/data" gml:id="Fifteen.1"
xsi:schemaLocation="http://www.opengis.net/wfs/2.0
http://schemas.opengis.net/wfs/2.0/wfs.xsd http://www.opengis.net/cite/data
http://182.71.43.203:8080/WFS/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVI
CE=WFS&VERSION=2.0.0&TYPENAME=cdf:Fifteen">
<gml:pointProperty>
   <gml:Point gml:id="Fifteen_b298de14-ef6f-4cbc-a021-f7e73ed70316"
srsName="urn:ogc:def:crs:EPSG::4326">
      <gml:coordinates>-92.999549,4.524015</gml:coordinates>
   </gml:Point>
</gml:pointProperty>
</cdf:Fifteen>
```

ListStoredQueries

A ListStoredQueries operation returns a list of stored queries currently maintained by the WFS server. This operation is valid for WFS version 2.0.0 only.

ListStoredQueries is supported via HTTP GET and HTTP POST. For further details, see the OGC Web Feature Service Implementation Specification.

The following is a portion of a sample ListStoredQueries response:

```
<wfs:ListStoredQueriesResponse xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:fes="http://www.opengis.net/fes/2.0"xmlns:wfs="http://www.opengis.net/wfs
/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:ows="http://www.opengis.net/ows/1.1"xmlns:xlink="http://www.w3.org/1999/x
link" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs/2.0
http://localhost:8080/geoserver/schemas/wfs/2.0/wfs.xsd">
   <wfs:StoredQuery id="urn:ogc:def:query:OGC-WFS::GetFeatureById">
      <wfs:Title xml:lang="en">Get feature by identifier</wfs:Title>
      <wfs:ReturnFeatureType/>
   </wfs:StoredQuery>
</wfs:ListStoredQueriesResponse>
```

MapXtreme supports WFS 2.0.0 Simple Profile so only GetFeatureById stored query is supported. Changing the StoredQueries.xml may have unknown behavior on the server response.

DescribeStoredQueries

A DescribeStoredQuery operation returns detailed metadata about each stored query maintained by the WFS server. This operation is valid for WFS version 2.0.0 only.

A description of an individual query may be requested by providing the ID of the specific query. If no ID is provided, all queries are described.

The following is a portion of a sample DescribeStoredQuery response:

```
<wfs:DescribeStoredQueriesResponse xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:fes="http://www.opengis.net/fes/2.0"xmlns:wfs="http://www.opengis.net/wfs
/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:ows="http://www.opengis.net/ows/1.1"xmlns:xlink="http://www.w3.org/1999/x
link" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs/2.0
http://localhost:8080/geoserver/schemas/wfs/2.0/wfs.xsd">
   <wfs:StoredQueryDescription id="urn:ogc:def:query:OGC-WFS::GetFeatureById">
      <wfs:Title xml:lang="en">Get feature by identifier</wfs:Title>
      <wfs:Parameter name="ID" type="xs:string"/>
      <wfs:QueryExpressionText isPrivate="true"
language="urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression"
returnFeatureTypes=""/>
   </wfs:StoredQueryDescription>
...
</wfs:DescribeStoredQueriesResponse>
```

Configuring a WFS 2.0.0 Server

If you have spatial data that you wish to make available to others, first of all, you must configure a WFS server to describe what data and capabilities you are offering. There are three configuration files that you need to provide to accomplish this. This discussion assumes you have a working knowledge of schemas and web services.

You do not need to configure a WFS server if you are only interested in accessing someone else's WFS server to retrieve features. See Using the MapXtreme WFS Client Programmatically.

The main configuration steps that are addressed in this section are:

Step1: Create or modify a Web.config file to include the MapXtreme-specific WFS information and the correct handlers for IIS classic or integrated pipeline mode.

Step 2: Create a valid WFS Server configuration file that contains information about the data you are hosting. This file must validate against the WFS schema file (MXP_WFSConfiguration_2_0.xsd) to avoid errors when you run the WFS Server.

Step 3: Configure and test the WFS Server setup. Instructions for IIS 7, IIS 8.5 and IIS 10 configuration are provided.

On the MapXtreme product media, we provide sample Web.config, WFS configuration and StoredQueries.xml files that you can use as a guide for creating your own files. The Web.config file defines how the ASP process is handled. The WFSSample.xml defines the data sources and feature definitions you want your WFS server to provide.

The StoredQueries file defines the stored queries supported by the server. WFS 2.0.0 server **Simple Profile** supports only GetFeatureById. There is no need to change this file while setting up WFS 2.0.0 server.

The schemas for the MapXtreme workspace and WFS 2.0.0 server are also located on the product media.

Step 1: Create a Web.config File

The Web.config is a standard configuration file for a web application. To use it for a MapXtreme WFS server, you must edit it to provide MapXtreme-specific WFS information and to define how the ASP.NET process will be handled.

1. Create a folder to contain the Web.config and the WFSSample.xml. In this example, the location is called **c:\wfs**.
2. Copy the Web.config, WFSSample.xml and StoredQueries.xml from the MapXtreme product media folder “WFS_Config_Files\WFS2.0” to this folder.
3. Open Web.config in a text editor and modify the <appSettings"> line to point to the WFS configuration file.

```
<configuration>

<appSettings>
<add key="configFile" value="C:\wfs\WFSSample.xml" />
```

4. **For IIS 7/8.5/10 classic mode**, update the version number and PublicKeyToken (if necessary) for the MapInfo.Wfs.Server and the MapInfo.CoreEngine assemblies installed on your system (**bold type** below).

   Assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 or GAC_64.

```
<system.web>
     <httpHandlers>
        <add verb="GET,POST" path="*.ashx" type="MapInfo.Wfs.WfsHttpHandler,
MapInfo.Wfs.Server, Version=9.2.0.XXX, Culture=neutral,
PublicKeyToken=4ac3224575145b20"/>
     </httpHandlers>
     <httpModules>
        <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.2.0.XXX, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator" />
    </httpModules>
```

5. **For IIS 7/8.5/10 Integrated pipeline mode,** copy the following section into the web.config. You do not need to comment out the <system.web> section to run in integrated pipeline mode. However, if you need to run in IIS 7 classic mode, you must comment out this <system.webServer> section.

   Follow the instructions in step 4 to update the assembly versions for MapInfo.CoreEngine and MapInfo.Wfs.Server.

```
<system.webServer>
      <validation validateIntegratedModeConfiguration="false"/>
      <directoryBrowse enabled="true" />
   <modules>
         <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.2.0.XXX, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator"/>
   </modules>
   <handlers>
         <add name="WFSHandler" verb="GET,POST" path="*.ashx"
type="MapInfo.Wfs.WfsHttpHandler, MapInfo.Wfs.Server, Version=9.2.0.XXX,
Culture=neutral, PublicKeyToken=4ac3224575145b20"/>
   </handlers>
</system.webServer>
```

6. Save this file and copy it to the location you created in step 1.

Step 2: Create a Valid WFS Configuration File for Hosted Features

The WFSSample.xml is a WFS configuration file provided with MapXtreme Media folder “WFS_Config_Files\WFS2.0”. This file defines information about your WFS server, including its name, title, abstract, the URL to the WFS Server and the data you want to host.

1. Open WFSSample.xml in a text editor and modify the OnlineResource line to include the URL of your WFS Server. Change `localhost` to what is appropriate for your WFS server.

   ```
   <mxp-wfs:Service>
      <mxp-wfs:Name>Sample WFS Server</mxp-wfs:Name>
      <mxp-wfs:Title>Sample WFS Server</mxp-wfs:Title>
      <!-- The following is the URL of the WFS server -->
      <mxp-wfs:OnlineResource>http://localhost/wfs/GetFeature.ashx</mxp-
wfs:OnlineResource>
   </mxp-wfs:Service>
   ```

   You can modify other elements as you see fit, including the server name, title, abstract, fees, access constraints and more.

2. Register the tables that the WFS serves by creating a Table element for each table. The value for the <mxp-wfs:DataSourceDefinition> id must match the value for <mxp-wfs:Name>. You can include the tables in any order.

3. In the <DataSourceDefinitionSet>, modify the MYPATH variable to reflect the actual path to your data.

   The following is a portion of the WFSSample.xml that identifies two tables of features.

```
<mxp-wfs:Table>
        <mxp-wfs:Name>USA</mxp-wfs:Name>
        <mxp-wfs:Title>Title for usa</mxp-wfs:Title>
        <mxp-wfs:Abstract>Abstract for USA</mxp-wfs:Abstract>
        <mxp-wfs:Keywords>Keywords for USA</mxp-wfs:Keywords>
        <mxp:DataSourceDefinitionSet>
           <mxp:TABFileDataSourceDefinition id="USA">
              <mxp:DataSourceName>USA</mxp:DataSourceName>
              <mxp:FileName>MYPATH\USA.TAB</mxp:FileName>
           </mxp:TABFileDataSourceDefinition >
        </mxp:DataSourceDefinitionSet>
     </mxp-wfs:Table>
     <mxp-wfs:Table>
        <mxp-wfs:Name>US_HIWAY</mxp-wfs:Name>
        <mxp-wfs:Title>Title for US_HIWAY</mxp-wfs:Title>
        <mxp-wfs:Abstract>Abstract for US_HIWAY</mxp-wfs:Abstract>
        <mxp-wfs:Keywords>Keywords for US_HIWAY</mxp-wfs:Keywords>
        <mxp:DataSourceDefinitionSet>
           <mxp:TABFileDataSourceDefinition id="US_HIWAY">
              <mxp:DataSourceName>US_HIWAY</mxp:DataSourceName>
              <mxp:FileName>MYPATH\US_HIWAY.TAB</mxp:FileName>
           </mxp:TABFileDataSourceDefinition >
        </mxp:DataSourceDefinitionSet>
     </mxp-wfs:Table>
```

4. If your WFS will be hosting data that is stored on a RDBMS, specify <ConnectionSet> and <ConnectionMember> elements, following the example below:

```
   <mxp:ConnectionSet>
      <mxp:DBConnection dbType="sqlserver">
       <mxp:ConnectionName>sqlserver2k</mxp:ConnectionName>
       <mxp:ODBCConnectionString>DRIVER={SQL
Server};DATABASE=YOURDB;Server=YOURSERVER;UID=YOURUSER;PWD=YOURPASSWORD;QuotedI
D=No;Trusted_Connection=No;Network=DBMSSOCN;Address=YOURSERVER,YOURSERVERPORT</
mxp:ODBCConnectionString>
     </mxp:DBConnection
   </mxp:ConnectionSet>
...
      <mxp-wfs:Table>
         <mxp-wfs:Name>MySQLServerTable</mxp-wfs:Name>
         <mxp-wfs:Title>Title for MySQLServerTable</mxp-wfs:Title>
         <mxp-wfs:Abstract>Abstract for MySQLServerTable</mxp-wfs:Abstract>
```

```
   <mxp-wfs:Keywords>Keywords for MySQLServerTable</mxp-wfs:Keywords>
   <mxp:DataSourceDefinitionSet>
      <mxp:DBDataSourceDefinition id="MySQLServerTable">
         <mxp:DataSourceName>MySQLServerTable</mxp:DataSourceName>
         <mxp:ConnectionMember>
<mxp:ConnectionName>my_sqlserver2000_advserver</mxp:ConnectionName>
         </mxp:ConnectionMember>
         <mxp:DBQuery>
            <mxp:Query>select * from MySQLServerTable</mxp:Query>
         </mxp:DBQuery>
         <mxp:DBDataSourceMetadata/>
      </mxp:DBDataSourceDefinition>
   </mxp:DataSourceDefinitionSet>
</mxp-wfs:Table>
```

Step 3: Configuring and Testing the WFS 2.0.0 Server

Once you have edited the Web.config and WFSSample.xml files, you must register your WFS server with Internet Information Services and finally, test your setup.

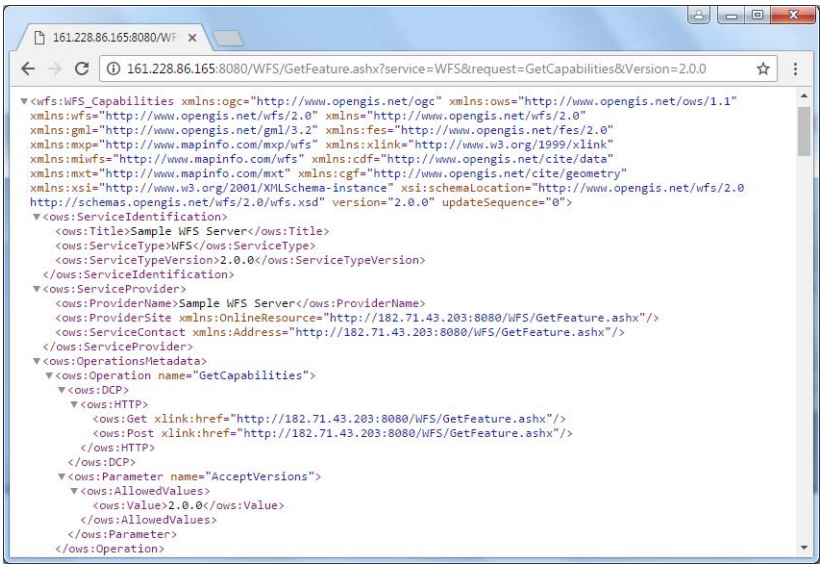
1. Create an Application Pool in IIS 7 or above. Based on the Web Service you want to run i.e., 32-bit or 64-bit, set the application pool to enable or disable 32-bit or 64-bit as needed.
2. Set the Integrated or Classic mode of pool based on the WFS server configuration.
3. Add new application to the just added application pool.
4. Enable Directory Browsing for application settings.
5. Test your setup by sending a GetCapabilities request from a web browser. In the address box type:

   `http://localhost/wfs/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WFS&VERSION=2.0.0`

   substituting your web server for `localhost`. If you have set an Alias to your web server, be sure to include that in your URL. For example:

   `http://localhost/My_WFS/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WFS&VERSION=2.0.0.`

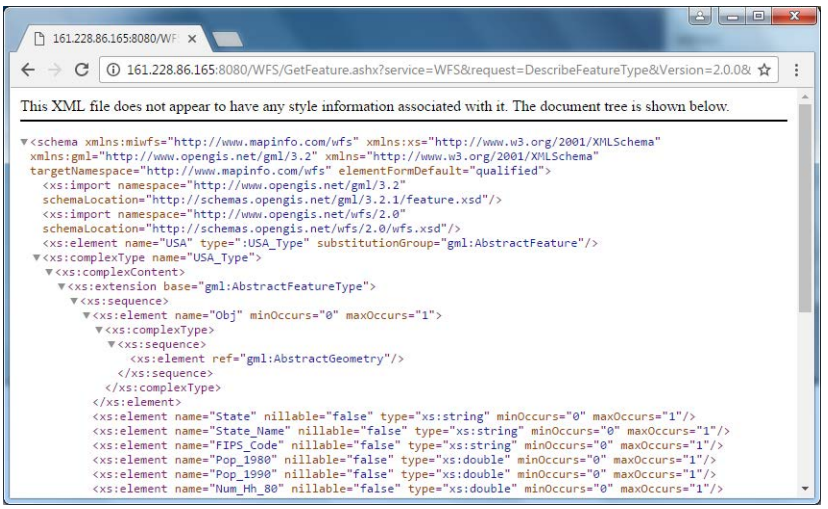
   A successful test will return a web page similar to the illustration below. If the capabilities are not returned, review your configuration files to ensure everything has been entered correctly.



6. To learn about the properties for the returned feature types, send a DescribeFeatureType request:

`http://localhost/My_wfs/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVICE=WFS&VERSION=2.0.0&Typename=miwfs:USA`

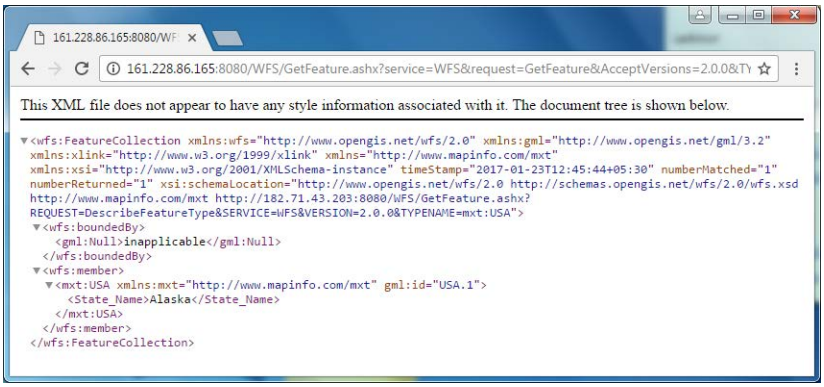
This request returns a description of the properties for the feature type USA, including column names such as, State, State_Name, FIPS_Code etc.



7. To request features from the USA table, send a GetFeature request.

`http://localhost/My_wfs/GetFeature.ashx?REQUEST=GetFeature&SERVICE=WFS&VERSION=2.0.0&Typename=miwfs:USA&propertyname=miwfs:State_Name&count=1`

This request returns one feature from the USA table as a FeatureCollection. Notice in the URL that we requested only one column property State_Name for the USA table.



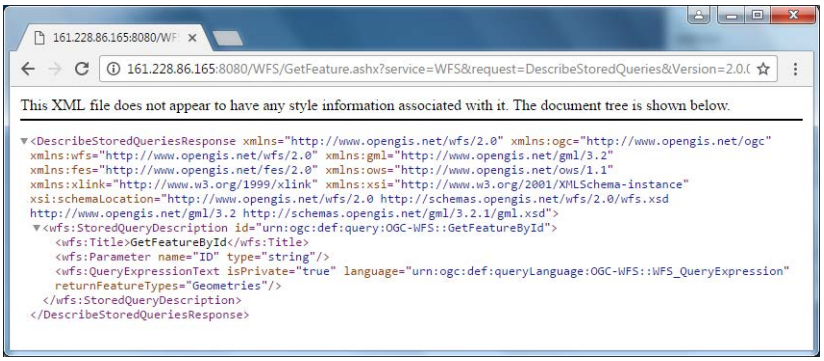
8. To find out all the Stored Queries supported by the server send a request as

```
http://localhost/My_wfs/GetFeature.ashx?REQUEST=ListStoredQueries&SERVICE=WFS
&VERSIO N=2.0.0
```

WFS 2.0.0 Simple Profile only supports GetFeatureById so only this query will be returned in the result.

9. To get the details about the Stored Queries what all parameters are required and their return types send a request as below.

```
http://localhost/My_wfs/GetFeature.ashx?REQUEST=DescribeStoredQueries&SERVICE
=WFS&VERSIO N=2.0.0
```



Using the MapXtreme WFS Client Programmatically

MapXtreme comes with a WFS Client that you can use programmatically to access data from OGC-compliant Web Feature Services. The MapXtreme WfsClient provides the ability to interact with any WFS 1.0.0 and WFS 2.0.0 compliant Server at the API level. The WfsClient and related classes are located in the MapInfo.Wfs.Client namespace and use the WFS Basic portion of the OGC specification.

Using the WfsClient class, you can call the following methods on any WFS 1.0.0 and WFS 2.0.0 compliant server: GetCapabilities, DescribeFeatureType, and GetFeature. With WFS 2.0.0 you can call two additional methods ListStoredQueries and DescribeStoredQueries.

```
public void MapInfo_Wfs_Client(RequestMethod requestMethod, string
wfsServerUrl, string wfsServerVersion)
{
    int httpRequestTimeOut = 1000000;

    // First we need to create  WfsClient object.
    MapInfo.Wfs.Client.WfsClient objWfsClient = new WfsClient(wfsServerUrl,
wfsServerVersion, httpRequestTimeOut);

    // Now get the list of capabilities and list out the tables available
    MapInfo.Wfs.Client.WfsCapabilities capabilities =
objWfsClient.GetCapabilities(requestMethod);

    // Get list of the stored queries currently maintained by the WFS server
    StoredQueries supportedListStoredQueries =
objWfsClient.ListStoredQueries(requestMethod);

    foreach (MapInfo.Wfs.Client.StoredQuery sq in
supportedListStoredQueries.ListStoredQueries){
```

```
            System.Console.WriteLine("\nStoredQuery: " + sq.id);
        }

        // call DescribeFeature() to get the table schema.
        // call GetFeature() to get Features:
        //    MultiFeatureCollection usa = objWfsClient.GetFeature(TypeNames,
    null, null, -1, null);
        // create a new mem table with table schema

        // Get detailed metadata about all stored query maintained by the WFS
    server, if the server version is 2.0.0.
        StoredQueryDescriptions storedQueriesDescs =
    objWfsClient.DescribeStoredQueries(requestMethod);

        foreach (StoredQueryDescription storedQueryDesc in
    storedQueriesDescs.StoredQueryDescriptionList){
            System.Console.WriteLine("\n StoredQueryDescription: " +
    storedQueryDesc.Id);
        }

       // Get detailed metadata about a single stored query maintained by the WFS
    server, if the server version is 2.0.0.
        string storedQueryId = "urn:ogc:def:query:OGC-WFS::GetFeatureById";
    //assign stored query Id here.

        StoredQueryDescription storedQueriesDesc =
    objWfsClient.DescribeStoredQueries(requestMethod, storedQueryId);

        System.Console.WriteLine("\n StoredQueryDescription: " +
    storedQueriesDesc.Id);

    }
```

Using Filters in WFS Queries

Filters can be applied to a GetFeature request using the IFilter interface. A filter may either be spatial or non-spatial (scalar). Spatial queries such as Bbox and Within allow you to get features that exist in a certain area. Scalar filters allow you to query against specific properties of a feature type. Compound filters like AND and OR may also be used.

MapXtreme WFS supports the following filter operations.

- Spatial operators: BBox, Equals, Disjoint, Intersects, Within, Contains
- Non-spatial comparison operators: Logical operators: AND, OR, and NOT; PropertyIsEqualTo, PropertyIsGreaterThan, PropertyIsGreaterThanOrEqualTo, PropertyIsLessThan, PropertyIsLessThanOrEqualTo, PropertyIsNotEqualTo,
- Simple arithmetic operators: Add, Div, Mul, Sub

ⓘ The HTTP protocol mandates a URL length for a GetFeature request of no more than 2048 characters. Keep in mind that a filter could easily create a URL much larger than that, which will cause an exception.

Code Example: Requesting Features Using Filters

The following C# code example illustrates how filters on WFS data can be used to return only the data you need. In this case, we query the WFS server to learn what features are available, then request that it return only features that match the population column of WorldCap with a value of 1,000,000 or greater. The output from the GetCapabilities and GetFeatures methods is displayed after the code example.

```
private void doWFS()
        {
     string wfsUrl = @"http://localhost/MXTWFS/GetFeatures.ashx";
     string wfsVersion = "2.0.0";

/// first we need to get the list of capabilities and list out the tables ///
available
     MapInfo.Wfs.Client.WfsCapabilities capabilities =
        MapInfo.Wfs.Client.WfsClient.GetCapabilities(
     MapInfo.Wfs.Client.RequestMethod.GET, wfsUrl, wfsVersion);
   MapInfo.Wfs.Client.FeatureTypeList featureTypeList =
        capabilities.FeatureTypeList;
     IList featureTypes = featureTypeList.FeatureTypes;
        foreach (MapInfo.Wfs.Client.FeatureType featureType in featureTypes)
            {
        System.Console.WriteLine("FeatureType: " + featureType.Name);
            }

/// Now we can look at a specific table to see its contents
     string featureTypeName = "miwfs:WorldCap"; // Name taken from
                  /// the above output
     MapInfo.Data.MultiFeatureCollection mfc =
          MapInfo.Wfs.Client.WfsClient.GetFeature(wfsUrl,
            new string[] { featureTypeName }, null, "GML3", -1, null,
wfsVersion);
           DisplayFeatureCollection(mfc[0]);

            /// Now we can apply a filter
     MapInfo.Wfs.Client.IFilter filter = new
          MapInfo.Wfs.Client.PropertyIsGreaterThanOrEqualTo(
            new MapInfo.Wfs.Client.PropertyName(wfsUrl, "CAP_POP"),
            new MapInfo.Wfs.Client.Literal("1000000"));

/// Create the Query container
///
```

```
      IList queries = new MapInfo.Wfs.Client.Query[] {
            new MapInfo.Wfs.Client.Query(new
            MapInfo.Wfs.Client.TypeName("http://www.mapinfo.com/wfs",
               "WorldCap"), null, filter)};

/// Run the filter and return the subset.
///
      mfc = MapInfo.Wfs.Client.WfsClient.GetFeature(wfsUrl,
         queries, "GML3", -1, wfsVersion);
      DisplayFeatureCollection(mfc[0]);
         }
```

The output of the above GetCapabilities code is:

```
FeatureType: miwfs:Ocean
FeatureType: miwfs:WldCty25
FeatureType: miwfs:World
FeatureType: miwfs:WorldCap
```

The first GetFeature call returns all the features (rows) in WorldCap, a portion of which is shown below. The first line shows the columns for the WorldCap data, the remaining lines show the individual rows of data.

```
Obj, Capital, Country, Cap_Pop, MI_Style,
Point,Abidjan,IVORY COAST,2700000,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Abu Dhabi,UNITED ARAB
EMIRATES,722000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Accra,GHANA,949000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Addis Ababa,ETHIOPIA,1423111,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Agana,GUAM,132726,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Algiers,ALGERIA,1483000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Alma-ata,KAZAKHSTAN,1108000,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Alofi,NIUE,3300,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Amman,JORDAN,936000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Amsterdam,NETHERLANDS,694656,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
...
```

From the filtered GetFeature call, only those rows of data that satisfy PropertyIsGreaterThanOrEqualTo a population of 1,000,000 are returned. Notice Abu Dhabi is not included in the results since its population is listed as 722,000.

```
Obj, Capital, Country, Cap_Pop, MI_Style,
Point,Abidjan,IVORY COAST,2700000,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Addis Ababa,ETHIOPIA,1423111,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Algiers,ALGERIA,1483000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
```

```
Point,Alma-ata,KAZAKHSTAN,1108000,MapInfo.Styles.SimpleVectorPointStyle: 12
point,
Point,Ankara,TURKEY,2553000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Baghdad,IRAQ,3400000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point,Baku,AZERBAIJAN,1115000,MapInfo.Styles.SimpleVectorPointStyle: 12 point,
...
```

Filtering with a Spatial Operator

The following example shows how to use a spatial query in a GetFeature request. In this case, it is asking for all the rows in the Europe table that satisfy the specified minimum bounding rectangle. This is accomplished using the spatial operator Bbox, a typical spatial operation that most WFS servers support.

```
/// Now lets find rows in an MBR

   // WGS84 - Europe
            MapInfo.Geometry.DRect bbox = new MapInfo.Geometry.DRect(-11.69,
35.36, 48.77, 65.05);
            mfc = MapInfo.Wfs.Client.WfsClient.GetFeature(
                wfsUrl,
                new string[] { featureTypeName },
                bbox,
                null,
                "GML3",
                -1);
            DisplayFeatureCollection(mfc[0]);
```

Creating a Map Layer from a WFS Response

If you wish to bring WFS feature data into MapXtreme as a map layer for further analysis, it is necessary for you to run code that parses the GML and creates a MultiFeatureCollection.

Parsing the WFS Response

Parsing the response is not specified in the WFS specification, nor does the specification describe an exact format for the results of a GetFeature request. The WFS specification only states that the request must be at least GML2. Since there are many versions of GML2, and in order for the MapXtreme WFS client to be able to correctly convert the GML2 response from a WFS Server, you must create a parser to convert the GML2 GetFeature response to a MapXtreme MapInfo.Data.MultiFeatureCollection. This is done by implementing the IWfsReader interface and registering that implementation with the MapInfo.Wfs.Client.WfsReaderFactory class. Registration must occur on a per URL and

wfs version basis, that is, if you want to interact with two WFS Servers that have the same GetFeature response, the specific IWfsReader implementation must be registered twice for each server URL.

The following code example illustrates the complete process from requesting features to creating a map layer and displaying the features in a map. The code for this example is contained in the WfsClient sample application located in the Samples folder under your MapXtreme installation.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Xml.Schema;

using MapInfo.Data;
using MapInfo.Engine;
using MapInfo.Mapping;
using MapInfo.Wfs.Client;

namespace MapInfo.Wfs.Client.Samples {

   /// <summary>
   /// Simple sample to demonstrate how to register a WfsReader to handle
   /// requests from a specific Wfs server, get the capabilities of the server,
   /// get the schema for a feature type located on the server and getting all
   /// of the features from the server.
   /// </summary>
   class SimpleSample {
      private const string URL = "http://www.mapinfo.com/miwfs";

         private const string defaultExportFileName = "WfsClient.gif";

      [STAThread]
      static void Main(string[] args) {

            string exportFileName = defaultExportFileName;
           if (args.Length > 0 && args[0] != null && args[0].Trim().Length != 0)
exportFileName = args[0];

         // register URL with a specific WFS reader
         WfsReaderFactory.RegisterHandler(URL, typeof(WfsReader));

         // Get the WFS capabilities of the WFS server using the HTTP GET method.
         try
         {

            // Get the WFS capabilities of the WFS server using the HTTP GET
method.
            WfsCapabilities Capabilities =
WfsClient.GetCapabilities(RequestMethod.GET, URL);
         }
```

```
        catch
        {
           MessageBox.Show("Please check if " + URL + " is a valid WFS URL");
           return;
        }

        // Do something with the the WfsCapabilities here...

        // Get the schema for the USA feature type
        string[] TypeNames = new string[] { "miwfs:USA" };

        // Do something with the schema here...
        XmlSchema usaSchema = WfsClient.DescribeFeatureType(URL, TypeNames);

        // Get all features from the USA feature type
        MultiFeatureCollection usa = WfsClient.GetFeature(URL, TypeNames, null,
null, -1, null);
        IFeatureCollection fc = usa[0];

        // iterate over the Usa MultiFeatureCollection and add each
        // IFeatureCollection to a MemTable, etc...
        TableInfoMemTable memTableInfo = new TableInfoMemTable("myMemTable");
        foreach (Column c in fc.Columns) {
           memTableInfo.Columns.Add(c);
        }
        Table memTable = Session.Current.Catalog.CreateTable(memTableInfo);
        memTable.InsertFeatures(fc);

        // create a layer from the MemTable
        FeatureLayer featureLayer = new FeatureLayer(memTable);

        // create the map and add the layer
        Map map = Session.Current.MapFactory.CreateEmptyMap(new Size(500, 500));
        map.Layers.Add(featureLayer);

            // export the map to a file
            using (MapExport mx = new MapExport(map))
            {
                mx.Format = ExportFormat.Gif;
                mx.Export(exportFileName);
            }

        // clean up the map
        Session.Current.MapFactory.Remove(map);
     }
   }
}
```

24 – Web Map Service

MapXtreme provides the ability to deploy and access Web Map Services (WMS) so that you can get WMS maps from a variety of sources or provide maps to others.

In this chapter:

Introduction to MapXtreme's Web Map Service

The MapXtreme Web Map Service (WMS) allows clients to request and servers to deliver spatially referenced map images over the Internet or private intranet. MapXtreme gives you the tools to both deploy a WMS Server for others to query, and supports the incorporation of WMS Client capabilities into your application to request images from a WMS Server. The WMS Server and Client implementations are based on the 1.3.0 OpenGIS® Web Map Service Implementation Specification, which can be found at www.opengis.org. The MapXtreme WMS Server meets the compliancy requirements for the 1.3.0 and 1.1.1 Web Map Service.

ⓘ For more information on WMS 1.3.0 support, see the MapXtreme Release Notes.

A basic WMS classifies its geo-referenced information holdings into layers and offers predefined styles in which to display those layers. A WMS that conforms to the OGC specification may support the transparent pixel definition for some image formats as well. Transparent pixels allow you to use retrieved WMS images as raster overlays and not solely as the background layer for a map. The quantity and quality of data available is determined by the individual WMS Server.

There is a growing amount of geo-spatial data available, provided by governments, corporations, and other organizations, that users can retrieve to enhance the accuracy and completeness of their maps. Companies with land use and water use data can add elevation and population information from the U.S. Census Bureau, or from a local data provider. Combining traffic pattern data with store location information can provide insight into establishing additional store locations or can optimize marketing and product placement efforts. As a developer, you can customize routing requests to include particular hotel, attraction, or vacation destinations by extending the WMS code. The possibilities are limited only by your need and your imagination.

Understanding WMS Operations

MapXtreme WMS follows the 1.3.0 and 1.1.1 OGC specifications for basic WMS service. There are three WMS operations:

- GetCapabilities
- GetMap
- GetFeatureInfo

GetCapabilities

Before requesting a WMS map, it is necessary to find out the names of available layers, styles in use, spatial information in use, and other information that the WMS server provides. GetCapabilities is an HTTP request that retrieves service-level metadata over the Internet or intranet, including the server name, the layer names, abstracts about the data and the acceptable request parameters.

In MapXtreme, GetCapabilities is the first step in collecting information from a WMS server. The capabilities are then used to request a map image.

GetMap

Once you know the capabilities of the WMS server, a GetMap request is issued to request a map image of one or more of the WMS server's map layers. Based on the WMS Server capabilities, a GetMap request includes the following.

- Version–Request version
- Layers–one or more layers of map data
- Styles–display styles for rendering the layers. If this is not specified, default styles will be used.
- A bounding box–the area to be included in the map image in
- CRS–for WMS 1.3.0, or SRS (WMS 1.1.1) the coordinate reference system for the map in the form of namespace:identifier.
- Output format–a MIME type such as GIF and PNG, for the output map image
- Output size–height and width in pixels of the map image
- Background color–a hexadecimal red-blue-green color value such as 0xFFFFFF (required when transparency is true)
- Transparency–a true/false setting that indicates if the layer can be used as a transparent overlay to other layers.

GetFeatureInfo

Once a GetMap request has been successfully completed the user may want further information about the features included in the map. The GetFeatureInfo operation returns information regarding the layers in the map and the queryable attributes of each layer. This operation is controlled by the WMS Server, however, the server may not offer this capability.

Using MapXtreme as a WMS Client

You can request a WMS map image programmatically using the MapXtreme WMS client via MapInfo.Data.TableInfoWMS. This section provides code snippets that illustrate how to use MapXtreme as a WMS client. In addition, a WMSPreview sample application is located in the Desktop samples folder after installation.

GetCapabilities

In MapXtreme, you do not call GetCapabilities directly. Your request for a map is contained in a call using an WMSClient that takes a capabilities instance as input.

```
// build the capabilities
ICapabilities capabilities = WmsClientUtilities.GetCapabilities
      (url, "1.3.0");

// create the WMS client
WmsClient wmsClient = new WmsClient(capabilities);
wmsClient.AddLayer("WORLD");
wmsClient.Srs = "EPSG:4326";
wmsClient.BGColor = Color.Blue;
wmsClient.MimeType = "image/gif";
```

You must supply the URL to the server you wish to reach. The version is optional. The MapXtreme Client appends the required information following the ? in an HTTP request to make a complete request of the server's capabilities. For example:

```
http://www.mapsanddata.xyz/gis/services/maps/hydrography/MapServer/WMSServer?re
quest=GetCapabilities&service=WMS&version=1.3.0
```

If the version is not specified and you are accessing a MapXtreme WMS Server that supports both, then the response for 1.3.0 is returned.

There is are overloaded methods that take an array of strings for the version number. The order of the versions is dependent on what you are looking for. The first successful match is returned.

GetCapabilities also supports user defined parameters in a request. Use the method that takes a NameValueCollection. See the Developer Reference for details.

The GetCapabilities response is returned in an XML document that MapXtreme reads and from which it creates the capabilities object.

GetMap

MapXtreme takes care of calling GetMap for you when you place a table object representing the request as a layer in your MapXtreme map. The code example below builds on the example in the GetCapabilities section.

```
// create the table info
TableInfoWms wmsTableInfo= new TableInfoWms
        ("MyWmsTable", wmsClient);

// create the table
Table wmsTable = Session.Current.Catalog.OpenTable(wmsTableInfo);

// creates a FeatureLayer from the table entry
FeatureLayer featLyr = new FeatureLayer(wmsTable);
```

For more information on TableInfoWms see WMS in the Supported Table Types on page 172. See also the TableInfoWms class in the Developer Reference.

GetFeatureInfo

The input for a GetFeatureInfo request are the bounds of a map and the pixel coordinates where the user clicked on the map with an Info tool.

```
// Get the feature info

Byte[] info = WmsClientUtilities.GetFeatureInfo(new DRect(45.0, 45.0, 90, 90),
640, 480, new String[] {"WORLD"}, new Point(300,200), "text/xml"));

MemoryStream memoryStream = new MemoryStream(byteArray);
memoryStream.Seek(0, SeekOrigin.Begin);
XmlDocument doc = new XmlDocument();
doc.Load(memoryStream);
//parse the xml doc as desired.
```

Code Example: Requesting a WMS Layer

To request a WMS layer programmatically with MapXtreme, follow the code example below.

ⓘ MapXtreme's implementation of WMS limits the size of the returned image to 4000 pixels each for width and height.

```
// build the capabilities
ICapabilities capabilities = WmsClientUtilities.GetCapabilities
     (url, "1.1.1");

// create the WMS client
WmsClient wmsClient = new WmsClient(capabilities);
wmsClient.AddLayer("WORLD");
wmsClient.Srs = "EPSG:4326";
wmsClient.BGColor = Color.Blue;
wmsClient.MimeType = "image/gif";
```

```
// create the table info
TableInfoWms wmsTableInfo= new TableInfoWms
        ("MyWmsTable", wmsClient);

// create the table
Table wmsTable = Session.Current.Catalog.OpenTable(wmsTableInfo);

// creates a FeatureLayer from the table entry
FeatureLayer featLyr = new FeatureLayer(wmsTable);
```

WMS and Coordinate Systems

MapXtreme provides support for three CRS authority coordinate systems that can be returned in a GetCapabilities request.

- CRS:84 - Longitude/latitude WGS 84. This is equivalent to EPSG:4236 in WMS 1.1.1.
- CRS:83 - Longitude/latitude NAD 83. This is equivalent to EPSG:4269 in WMS 1.1.1.
- CRS:27 - Longitude/latitude NAD 27. This is equivalent to EPSG:4267 in WMS 1.1.1.

In addition, the MapXtreme API provides a public method **RegisterCRSCode()** for you to register other longitude/latitude projections in the CRS codespace.

Map and Image Bounds

EX_GeographicBoundingBox

The EX_GeographicBoundingBox is a new parameter for WMS 1.3.0 that defines the minimum bounding rectangle of the layer in decimal degrees. Its purpose is to facilitate geographic searches without requiring coordinate transformation by the search engine.

The attributes for EX_GeographicBoundingBox are westBoundLongitude, eastBoundLongitude, southBoundLatitude, northBoundLatitude. EX_GeographicBoundingBox is supplied regardless of what CRS the WMS server supports. It may be approximate if the data are not natively in geographic coordinates.

The equivalent parameter in WMS 1.1.1 is LatLongBoundingBox with attributes minx, miny, maxx and maxy in EPSG:4326 longitude/latitude.

BoundingBox

Each layer delivered by a WMS 1.3 server must have at least one bounding box. The coordinates are presented in the order required by the coordinate system authority. In the example of an EPSG and CRS code, the bounding box coordinates are in reverse order to each other.

```
<BoundingBox CRS="EPSG:4326" minx="-59.100605" miny="-86.389389"
maxx="16.755765" maxy="-32.336389"/>
<BoundingBox CRS="CRS:84" minx="-86.389389" miny="-59.100605" maxx="-32.336389"
maxy="16.755765"/>
```

This means for EPSG:4326, the minx is the southern most latitude and miny is the western most longitude. For the CRS:84, minx is the western most longitude and miny is the southern most latitude.

MapXtreme also supports optional BoundingBox attributes for resx resy to describe the spatial resolution.

Image Stretching

WMS requires that the image returned from a GetMap request correspond to both the BBOX and image size (WIDTH, HEIGHT) parameters. This is so that the returned image not unreasonably stretched to fit the bounding box.

MapXtreme WMS and Authentication

Basic Authentication

MapXtreme's WMS Client implements support for managing the basic authentication protocol when making service requests against a secured WMS server. The authentication credentials can be defined on the WmsRequest object, prior to initiating the service request, allowing for programmatic control of the authentication via the API. If the WmsRequest does not have the proper credentials, and the client receives an Unauthorized error message from the server, a Windows dialog, prompting the user for the credentials, will be displayed. The use of the credentials dialog may be disabled, in which case the authentication exception is returned directly to the calling application for handling.

Upon successful authentication with a WMS Server, the credentials for the server will be cached for the lifetime of the client session. The client can make subsequent WMS requests to the server, or make requests to other servers, with or without authentication, without having to re-authenticate.

To support integration of authentication into existing MapXtreme WMS Client implementations, the semantics of the WmsRequest UserDefinedParameters have been extended to include support for a set of well-known authentication parameters, which maps directly to the set of authentication properties on the WmsRequest object model. Setting the authentication properties via the object model directly, or indirectly via the UserDefinedParameters has the exact same effect.

For more information, see the WmsRequest class in the Developer Reference.

Setting up a MapXtreme WMS Server

MapXtreme provides a WMS server that is compliant with the OGC WMS 1.3.0 and 1.1.1 specifications.

To set up your own WMS Server you must configure a server connection to IIS and create the XML file necessary for providing the data connection required to host a Web Map Service. We assume you have a working knowledge of WMS and the *MapXtreme workspace schema???*.

The WMS Server runs inside of Microsoft Internet Information Services (IIS). The following are the configuration steps for setting up a WMS Server.

Step1: Create or modify a Web.config file to include the MapXtreme-specific WMS information and the correct handlers for IIS classic or integrated pipeline mode.

Step 2: Create a valid WMS configuration file that contains information about the data you are hosting. This file must validate against the WMS schema file (MXP_WMS_Configuration_1_2.xsd) to avoid errors when you run the WMS Server.

Step 3: Configure and test the WMS Server setup. Instructions for IIS7 and IIS6 configuration are provided.

On the MapXtreme product media, we provide sample Web.config and WMS configuration files that you can review and modify for your own needs. The Web.config file defines how the ASP.NET process is handled and the WMS server's relationship to MapXtreme. The WMSSample.xml defines the data sources and layer definitions that you can use as a model for your WMS server implementation.

Step 1: Create a Web.config File

The Web.config is a standard configuration file for a web application. Here the file is modified to provide MapXtreme-specific WMS information and to define how the ASP.NET process will be handled.

MapXtreme supports IIS 7/8.5/10 in both classic and Integrated pipeline modes.

1. Create a folder to contain the Web.config and the configuration file WMSSample.xml. In this example, the location is called **c:\wms**. Copy the Web.config and WMSSample.xml from the MapXtreme product media to this folder.
2. Open Web.config In a text editor and modify the <appSettings"> line to point to the WMS configuration file.

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

3. **For IIS 7/8.5/10 classic mode**, update the version number and PublicKeyToken (if necessary) for the MapInfo.Wms.Server and the MapInfo.CoreEngine assemblies installed on your system (**bold type** below).

   Assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 or GAC_64.

```
<system.web>
     <httpHandlers>
        <add verb="GET,POST" path="*.ashx" type="MapInfo.Wms.WmsHttpHandler,
MapInfo.Wms.Server, Version=9.x.x.x, Culture=neutral,
PublicKeyToken=4ac3224575145b20"/>
     </httpHandlers>
     <httpModules>
        <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.x.x.x, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator" />
    </httpModules>
```

4. **For IIS 7/8.5/10 Integrated pipeline mode,** copy the following section into the web.config. You do not need to comment out the <system.web> section to run in integrated pipeline mode. However, if you need to run in IIS 7/8.5/10 classic mode, you must comment out this <system.webServer> section.

   Follow the instructions step 3 to update the assembly versions for MapInfo.CoreEngine and MapInfo.Wms.Server.

```
<system.webServer>
     <validation validateIntegratedModeConfiguration="false"/>
     <directoryBrowse enabled="true" />
   <modules>
        <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.x.x.x, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator"/>
   </modules>
   <handlers>
        <add name="WMSHandler" verb="GET,POST" path="*.ashx"
type="MapInfo.Wms.WmsHttpHandler, MapInfo.Wms.Server, Version=9.x.x.x,
Culture=neutral, PublicKeyToken=4ac3224575145b20"/>
```

```
   </handlers>
</system.webServer>
```

5. Save the web.config file and copy it to the location you created in step 1.

Step 2: Create a Valid WMS Configuration File for Hosted Data

The WMSSample.xml is a WMS configuration file provided with MapXtreme. This file defines information about your WMS server, including its name, title, abstract, the URL to the WMS Server and the data you want to host.

1. Open WMSSample.xml in a text editor and modify `<mxp-wms:OnlineResource>` line to point to your WMS Server.

   You can modify other elements as you see fit, including the server name, title, abstract, keyword lists and vocabularies, contact information, fees, and access constraints.

```
<mxp-wms:Service>
      <mxp-wms:Name>Sample WMS Server</mxp-wms:Name>
      <mxp-wms:Title>Sample WMS Server</mxp-wms:Title>
      <mxp-wms:Abstract>This is a sample WMS server</mxp-wms:Abstract>
      <mxp-wms:KeywordList>
         <mxp-wms:Keyword vocabulary="ISO 19115:2003">biota</mxp-wms:Keyword>
         <mxp-wms:Keyword vocabulary="ISO 19115:2010">biota</mxp-wms:Keyword>
         <mxp-wms:Keyword>rivers</mxp-wms:Keyword>
      </mxp-wms:KeywordList>
      <!-- The following is the URL of your WMS server, here assume it is
localhost -->
      <mxp-wms:OnlineResource>http://localhost:port/WMS/GetMap.ashx</mxp-
wms:OnlineResource>
      <mxp-wms:Fees>$10</mxp-wms:Fees>
      <mxp-wms:AccessConstraints>none</mxp-wms:AccessConstraints>
   </mxp-wms:Service>
```

2. Register your data layers by modifying the paths in the `<DataSourceDefinitionSet>`. Every layer you wish to serve on your WMS needs an entry. The following is a portion of the WMSSample.xml that identifies TAB files for a sample WMS Server:

```
      <!-- The following data sources reference local TAB files, you need to
replace MYPATH with the real path to those tab files. -->
      <TABFileDataSourceDefinition id="id1" readOnly="false"
xmlns="http://www.mapinfo.com/mxp">
         <DataSourceName>STATES</DataSourceName>
         <FileName>MYPATH\USA.TAB</FileName>
      </TABFileDataSourceDefinition>
      <TABFileDataSourceDefinition id="id2" readOnly="false"
xmlns="http://www.mapinfo.com/mxp">
         <DataSourceName>US_HIWAY</DataSourceName>
         <FileName>MYPATH\US_HIWAY.TAB</FileName>
```

```
      </TABFileDataSourceDefinition>
      <TABFileDataSourceDefinition id="id5" readOnly="false"
xmlns="http://www.mapinfo.com/mxp">
         <DataSourceName>OCEAN</DataSourceName>
         <FileName>MYPATH\OCEAN.TAB</FileName>
      </TABFileDataSourceDefinition>
```

3. Describe the layers you want to host on your WMS server under the `<mxp-wms:WMSLayer>` section. You will need one entry for every layer you plan to offer on your WMS server. Layers can also be nested so that by requesting a parent layer, all the child layers are included in the response.

   The following example is the entry for a single layer called States. The bold text below call out the elements added in support of WMS 1.3.0.

   For more information about how to build these entries, see Configuring Layer Information for a WMS Server.

```
<mxp-wms:WmsLayer>
   <mxp-wms:Name>States</mxp-wms:Name>
   <mxp-wms:Title>States</mxp-wms:Title>
   <mxp:SRSName>EPSG:4326</mxp:SRSName>
   <mxp-wms:Attribution type="FGDC:1998">
      <mxp-wms:Title>Attribution</mxp-wms:Title>
      <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/metadata/roads.txt">
      </mxp-wms:OnlineResource>
      <mxp-wms:LogoURL width="500" height="600">
         <mxp-wms:Format>text/plain</mxp-wms:Format>
         <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/metadata/roads.txt">
         </mxp-wms:OnlineResource>
      </mxp-wms:LogoURL>
   </mxp-wms:Attribution>
   <mxp-wms:MetadataURL type="FGDC:1998">
      <mxp-wms:Format>text/plain</mxp-wms:Format>
      <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/metadata/roads.txt" />
   </mxp-wms:MetadataURL>
   <mxp-wms:MetadataURL type="ISO19115:2003">
      <mxp-wms:Format>text/xml</mxp-wms:Format>
      <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/metadata/roads.xml" />
   </mxp-wms:MetadataURL>
   <mxp-wms:FeatureListURL>
      <mxp-wms:Format>text/xml</mxp-wms:Format>
      <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/featurelist/feature1.xml" />
   </mxp-wms:FeatureListURL>
   <mxp-wms:FeatureListURL>
      <mxp-wms:Format>text/plain</mxp-wms:Format>
```

```
      <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/featurelist/feature2.xml" />
   </mxp-wms:FeatureListURL>
   <mxp-wms:WmsStyleSet>
      <mxp-wms:WmsStyle>
         <mxp-wms:Name>BlueFill</mxp-wms:Name>
         <mxp-wms:Title>Blue Fill</mxp-wms:Title>
         <mxp-wms:Abstract>This is a blue area fill with a red border.</mxp-
wms:Abstract>
         <mxp-wms:LegendURL width="100" height="100">
            <mxp-wms:Format>image/gif</mxp-wms:Format>
            <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/WMS/legends/1.gif" />
         </mxp-wms:LegendURL>
         <mxp-wms:LegendURL width="200" height="200">
            <mxp-wms:Format>image/gif</mxp-wms:Format>
            <mxp-wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/WMS/legends/2.gif" />
         </mxp-wms:LegendURL>
         <AreaStyle xmlns="http://www.mapinfo.com/mxp">
            <!-- The following defines the red border -->
            <LineStyle stroke="red" width="1" width-unit="mapinfo:imagesize
pixel">
               <Pen>mapinfo:pen 2</Pen>
            </LineStyle>
            <!-- The following defines the blue fill -->
            <Interior fill-opacity="0" fill="(#id7)">
<Defs>
   <Pattern id="id7">
      <Bitmap uri="mapinfo:brush 2">
         <ColorAdjustmentSet>
            <ColorAdjustment color-1="nonWhite" color-2="blue" opacity="1"/>
            <ColorAdjustment color-1="white" opacity="1"/>
         </ColorAdjustmentSet>
      </Bitmap>
   </Pattern>
</Defs>
            </Interior>
         </AreaStyle>
      </mxp-wms:WmsStyle>
   </mxp-wms:WmsStyleSet>
   <mxp-wms:MinScaleDenominator>10</mxp-wms:MinScaleDenominator>
   <mxp-wms:MaxScaleDenominator>20</mxp-wms:MaxScaleDenominator>
   <FeatureLayer id="id8" name="STATES" alias="STATES" volatile="unknown"
xmlns="http://www.mapinfo.com/mxp">
      <DataSourceRef ref="id1"/>
   </FeatureLayer>
</mxp-wms:WmsLayer>
```

4. Save WMSSample.xml after you are through adding feature layer information.

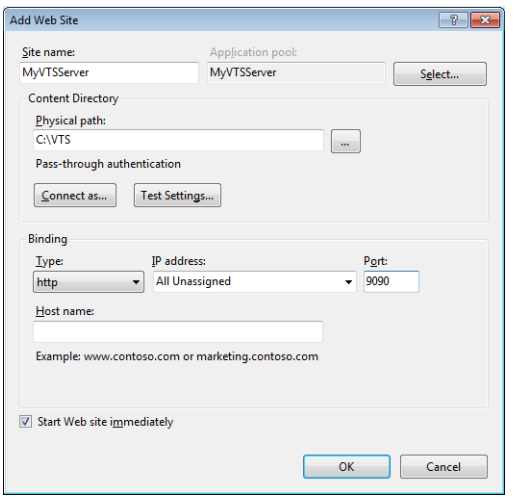
Register your WMS server with Internet Information Services (IIS) and test your setup. Follow the instructions for IIS7

Step 3a: Configure and Test the WMS Server using IIS 7/8.5/10

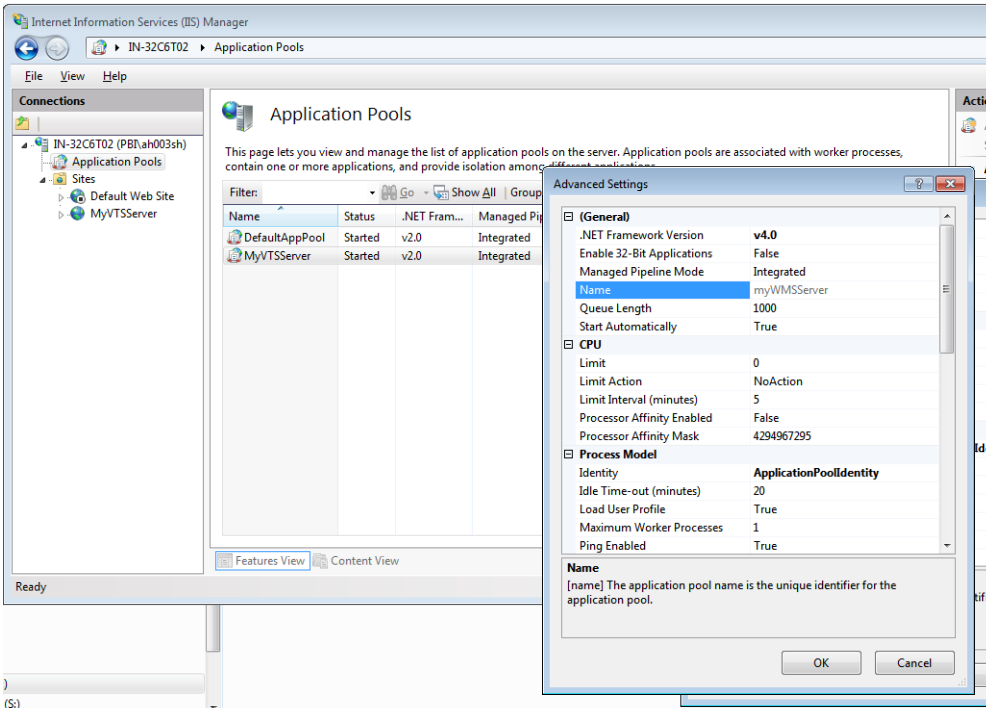
1. Right-click on your WMS folder (for example, c:\wms) and choose Properties. Select the Sharing tab and click the Share button. Add the IUSR account to the user list. Click Share, then Done to return to the Properties dialog. Click the Security tab. Add IUSR account to the security list.

   For information on setting up permissions for the IUSR account, see Understanding Built-In User and Group Accounts in IIS 7/8.5/10.

2. Open the IIS 7 manager. Right-click on Sites and then choose Add Website from the menu. In the dialog that displays, name your site (e.g., "WMS") and navigate to the physical path of the WMS folder. Click the Connect As button and ensure that application use is checked. Assign a free port number to the website, or use the default 80. Click OK.

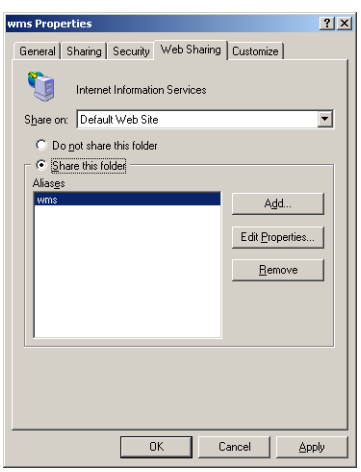


3. Click Application Pools in the left pane. Right-click on the new application pool that IIS7 has created for your WMS website ("WMS") and choose Advanced Settings. Change the .Net Framework Version to 'v4.0' Click OK.



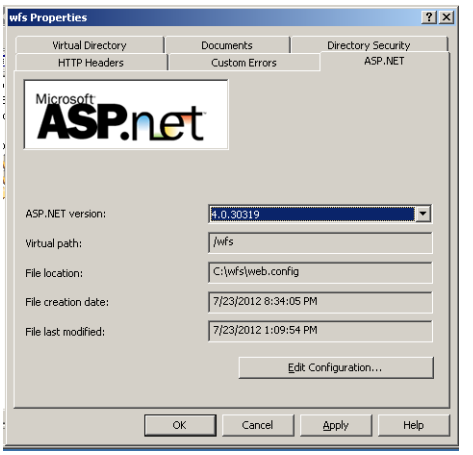
Step 3b: Configure and Test the WMS Server with IIS7/8.5/10

1. Right-click on your WMS folder (for example, c:\wms) and choose Sharing and Security. From the Web Sharing tab, choose the Share this folder radio button. If you wish to set an alias for your web server, click the Add button and supply a name in the dialog.

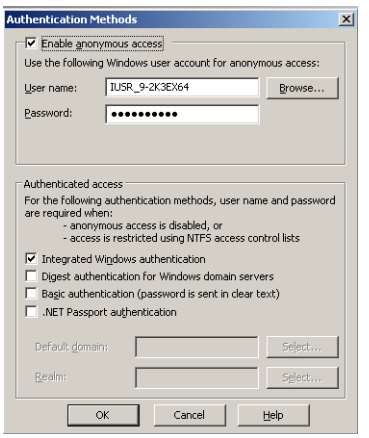


2. Open IIS (From the Start menu > Control Panel > Administrative Tools > Internet Information Services). Expand the Default Web Site and locate your WMS server (by folder name or by alias, if you use one).

3. Right-click on the Web site and choose Properties. Under the ASP.NET tab, choose 4.0.30319 for the ASP.NET version (The MapXtreme assemblies are compiled under the 4.7.2 Framework).



4. In the same Properties dialog, under the Directory Security tab, click the Edit button at the top right. In the Authentication Method dialog box, select the Anonymous Access check box. This allows users of your WMS service to skip the username/password authentication process. Click OK twice and close the IIS window.



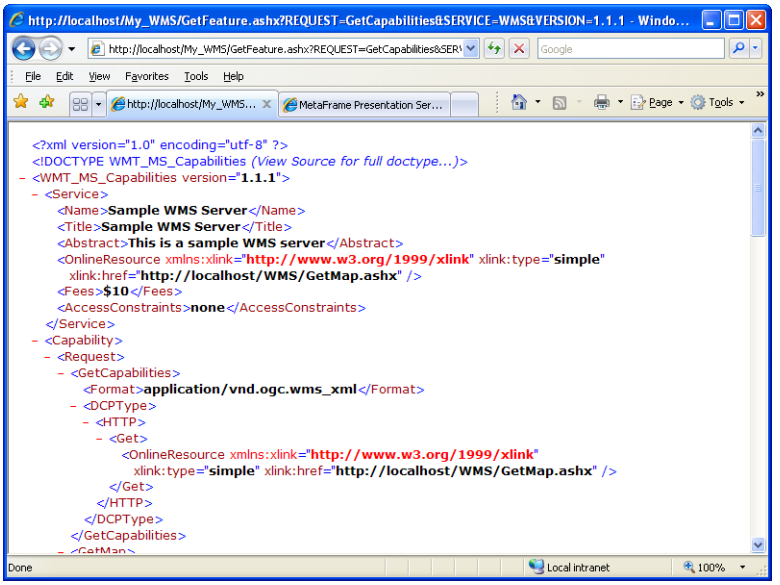
5. Test your setup by sending a GetCapabilities request from a web browser. In the address box type:

```
http://localhost/wms/GetMap.ashx?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.3.0
```

substituting your web server for `localhost`. If you have set an Alias to your web server, be sure to include that in your URL. For example:

```
http://localhost/My_WMS/GetMap.ashx?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.3.0
```

A successful test will return a web page similar to the illustration below. If the capabilities are not returned, review your configuration files to ensure everything has been entered correctly. Since you are creating the configuration files by hand, It is very easy to include typos or have missing tags.



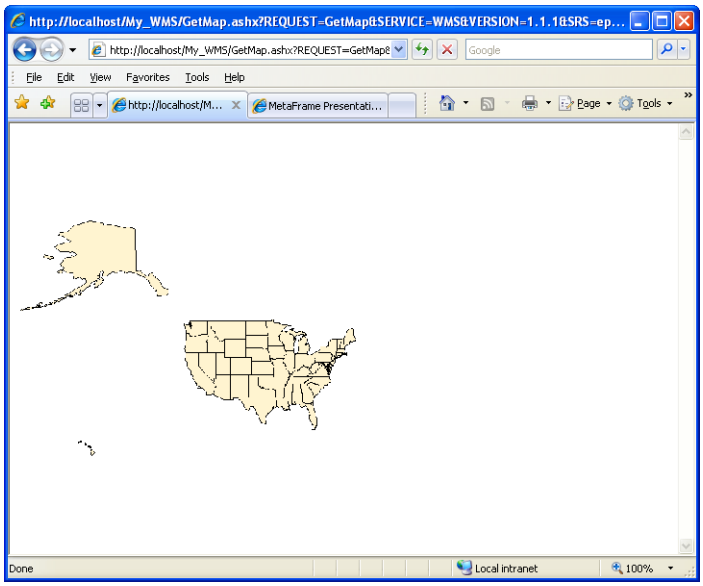
6. If you make any changes to the WMSSample.xml after you access the WMS Server, you will need to reset IIS in order for your changes to take affect. To reset IIS, from a command prompt type:

   ```
   c:\>IISreset
   ```

7. To request a test map image, send a GetMap request, following the example below:

   ```
   http://My_Wms/GetMap.ashx?service=WMS&VERSION=1.3.0&SRS=epsg:4267&REQUEST=GetMap&LAYERS=States&STYLES=&BBOX=-180,0,0,90&WIDTH=800&HEIGHT=600&FORMAT=image/gif
   ```

   The WMS server returns a map image of the States layers.



Configuring Layer Information for a WMS Server

Layers are described in a WMS Server configuration file according to elements laid out in the MXP_WMSConfiguration_1_2.xsd schema. New for WMS 1.3 are in bold. These elements include:

- Name–a name of the layer that is used to reference the layer in requests.
- Title–a name of the layer that is readable by the user.
- Abstract–a longer narrative description of the layer.
- KeywordLIst and **Vocabulary** attribute–List of keywords or keyword phrases to help catalog searching. Vocabulary indicates the authority of the keywords.
- SRSNameSet–the spatial reference system(s) that applies to this layer. If more than one SRS can be used on the layer, each one can be listed. This list is unordered and applies to parent and child layers.
- **Attribution**–identifies the source of the geographic information.
- **MetadataURL**–a link to detailed, standardized metadata about the data corresponding to a particular layer.
- **FeatureListURL**–points to a list of the features represented in a Layer.
- **MinScaleDenominator and MaxScaleDenominator**–define the appropriate range of scales for a WMS map

- WmsStyleSet–An unordered set of pre-defined styles supported by this layer and any child layers. The WmsStyleSet is a collection of WmsStyles, each bearing a name, title, abstract and reference to a style definition. Each style can have a **LegendURL** that points to an image of a legend for that style.

WMS layers can be made up of a single layer of data or a hierarchical collection of layers. A basic implementation of WMS allows the client to specify which layers it needs, as well as the coordinate system and styles for those layers that will be rendered as a map image.

Your job, as developer of a WMS Server, is to decide how to assemble your data into WMS layers. For example, you may have geographic data broken out into 10 separate .TAB files, such as boundary files, point files and polyline files. In your WMS configuration file, you can represent these 10 layers as a single WMS layer or you can keep them as individual layers. If your data is offered to a WMS client as a single layer, the client will get all 10 layers represented in a single WMS image. In this case, the client cannot separate out the information they do not want.

If, however, you provide the 10 layers as individual layers, the WMS client can choose one, some or all of those layers to be returned in a single image, or perhaps in multiple images, depending on their needs. You must decide when configuring your WMS Server how much flexibility you will offer users.

You can also nest your layers so that by requesting the parent layer, the child layers are included in the map image.

Using a MapXtreme Workspace to Build a WMS Configuration File

To build up the layer information in your WMS configuration file, consider extracting information about each layer from a previously saved MapXtreme Workspace (.MWS).

The following is an excerpt from an .MWS created with MapXtreme Workspace Manager that defines a layer called "World Countries". The .MWS layout is an XML file that follows the schema contained in MXP_Workspace_1_5.xsd.

```
<FeatureLayer id="id10" name="World Countries" alias="world" volatile="unknown">
          <Visibility visible="true">
            <VisibleRange enabled="false">
              <ZoomRange uom="mapinfo:length mi" minInclusive="true"
maxInclusive="false">0 0</ZoomRange>
            </VisibleRange>
          </Visibility>
          <DataSourceRef ref="id4" />
        </FeatureLayer>
```

Now, this same layer information is contained in a WMS configuration file. Notice the information is nearly identical, except that each tag contains "mxp:" to indicate these elements belong to the MapXtreme workspace schema (MXP_Workspace_1_5.xsd).

```
<mxp:FeatureLayer id="id10" name="World Countries" alias="world"
volatile="unknown">
      <mxp:Visibility visible="true">
         <mxp:VisibleRange enabled="false">
            <mxp:ZoomRange uom="mapinfo:length mi" minInclusive="true"
maxInclusive="false">0 0</mxp:ZoomRange>
         </mxp:VisibleRange>
      </mxp:Visibility>
      <mxp:DataSourceRef ref="id4" />
   </mxp:FeatureLayer>
```

You can build your layers by designing them in Workspace Manager and copying the information into your WMS configuration file. See Chapter 27 Workspace Manager.

Feature Layers and More

You are not limited to including Feature layer information in your WMS layers. For example, you can also capture label, themes and style overrides in Workspace Manager and paste those sections in your WMS Configuration file.

Here is a portion of an .MWS that defines a graduated symbol theme for a population layer. Notice that every element that defines the theme is captured here. When you paste this into your WMS configuration file, remember to include in each element mxp:

```
<ObjectThemeLayer id="id8" name="Graduated Symbol Theme on World Capitals by
Cap_Pop" alias="GraduatedSymbolThemeLayer1" volatile="unknown">
          <Visibility visible="true">
            <VisibleRange enabled="false">
              <ZoomRange uom="mapinfo:length mi" minInclusive="true"
maxInclusive="false">0 0</ZoomRange>
            </VisibleRange>
          </Visibility>
          <DataSourceRef ref="id4" />
          <FeatureGraduatedSymbolTheme id="id9">
            <Alignment>
              <HorizontalAlignment>center</HorizontalAlignment>
              <VerticalAlignment>center</VerticalAlignment>
            </Alignment>
            <SymbolBaseSize useScale="false">
              <MapScale>1.0</MapScale>
              <DataValueAtSize>20000000</DataValueAtSize>
              <PositiveSymbol>
                <PointStyle>
                  <FontSymbol size="18" size-unit="mapinfo:length pt"
stroke="red" stroke-opacity="1" text="&amp;quot;" family="MapInfo 3.0 Compatible" />
                </PointStyle>
              </PositiveSymbol>
```

```
              <NegativeSymbol visible="false">
                <PointStyle>
                  <FontSymbol size="18" size-unit="mapinfo:length pt"
stroke="blue" stroke-opacity="1" text="&amp;quot;" family="MapInfo 3.0 Compatible"
/>
                </PointStyle>
              </NegativeSymbol>
              <SymbolGraduation>sqrt</SymbolGraduation>
            </SymbolBaseSize>
            <SymbolLayout />
            <NumericValueExpression>
              <AttributeName>Cap_Pop</AttributeName>
            </NumericValueExpression>
            <SymbolLegendRowOverrideSet>
              <SymbolLegendRowOverride visible="false" row="4" />
              <SymbolLegendRowOverride visible="false" row="5" />
              <SymbolLegendRowOverride visible="false" row="6" />
            </SymbolLegendRowOverrideSet>
          </FeatureGraduatedSymbolTheme>
        </ObjectThemeLayer>
```

25

25 – Vector Tile Service

MapXtreme provides the ability to deploy an XYZ Vector Tile Server. It supports the Mapbox Vector Tile (MVT) format for tile generation. Mapbox Vector Tile is an open standard that specifies a space-efficient encoding format for tiled geographic vector data using Google Protocol Buffers. This vector format can be used as an alternative to raster image formats (PNG, GIF, and JPG/JPEG) in the Map Tiling Service and the Web Map Tile Service (WMTS).

In this chapter:

Introduction to Vector Tiles

Vector tiles are square shaped portions of map called tiles. It enables you to deliver map data in small chunks to a browser or other client application for map visualization purposes. The small file size enable faster map rendering and better performance.

Vector tiles contains vector geometries and metadata such as road names, place names or house numbers in structured format. Vector tiles can be rendered on request by a client, like a web browser or a mobile app or a desktop GIS application. They can also be rendered on the fly on a server.

Vector tiles offer the following advantages over raster tiles.

- Tiles can be produced more quickly.
- Compared to a tiled raster map, data transfer is also greatly reduced.
- Map Styles can be changed without downloading more content as styling can be applied at the client end itself.

MapXtreme Vector Tile Service

MapXtreme 9.2 provides the ability to deploy an XYZ Vector Tile Server. It supports the Mapbox vector tile (MVT) format for tile generation. Mapbox Vector Tile is an open standard that specifies a space-efficient encoding format for tiled geographic vector data using Google Protocol Buffers. This vector format can be used as an alternative to raster image formats (PNG, GIF, and JPG/JPEG) in the Map Tiling Service, the Web Map Tile Service (WMTS). For more information about the MVT format, see Mapbox specification.

Setting up a MapXtreme Vector Tile Server

This section contains the guidelines for setting up and consuming an ASP.NET vector tile server using MapXtreme 9.2.

To set up your own Vector Tile Server you must configure a server connection to IIS and create the XML file necessary for providing the data connection required to host a Vector Tile Service.

On the MapXtreme product media, we provide sample Web.config and Vector Tile Server configuration files that you can review and modify for your own needs. The Web.config file defines how the ASP.NET process is handled and the Vector Tile server's relationship to MapXtreme. The VTSample.xml defines the tile sets that you can use as a model for your Vector Tile server implementation.

After installation, these sample files along with a sample Leaflet Client are also placed at the following path:

C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudioxxxx\Web\Features\VectorTiling.

Configure a Vector Tile Server

The Vector Tile Server runs inside of Microsoft Internet Information Services (IIS). The following are the configuration steps for setting up a Vector Tile Server.

Step 1: Create a Web.config File

The Web.config is a standard configuration file for a web application. Here the file is modified to provide MapXtreme-specific Vector Tile Server information and to define how the ASP.NET process will be handled.

MapXtreme supports IIS 7/8.5/10 in both Classic and Integrated pipeline modes.

1. Create a folder to contain the Web.config and the configuration file VTSample.xml. In this example, the location is called **c:\vts**. Copy the Web.config and VTSample.xml from the MapXtreme product media to this folder.
2. Open Web.config In a text editor and modify the <appSettings"> line to point to the Vector Tile Server configuration file.

```
<appSettings>
   <!-- Use this setting to specify the location of the VectorTile Server
configuration file -->
   <add key="configFile" value="C:\Program
Files\MapInfo\MapXtreme\9.2.0\Samples\VisualStudio2015\Web\Features\VectorTil
ing\Server\VtSample.xml" />

  </appSettings>
```

3. **For IIS 7/8.5/10 classic mode**, update the version number and PublicKeyToken (if necessary) for the MapInfo.VectorTile.Server and the MapInfo.CoreEngine assemblies installed on your system (**bold type** below).

   Assemblies are located in C:\Windows\Microsoft.NET\assembly\GAC_32 or GAC_64.

```
<system.web>
   <customErrors mode="Off" />
      <httpHandlers>
         <add verb="GET,POST" path="*.mvt"
type="MapInfo.VectorTile.Server.HttpHandler,MapInfo.VectorTile.Server,
Version=9.2.0.xxx, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1" />
```

```
      </httpHandlers>
      <httpModules>
         <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine,
Version=9.2.0.xxx, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
name="WebSessionActivator" />
     </httpModules>
```

4. **For IIS 7/8.5/10 Integrated pipeline mode,** copy the following section into the web.config. You do not need to comment out the <system.web> section to run in integrated pipeline mode. However, if you need to run in IIS 7/8.5/10 classic mode, you must comment out this <system.webServer> section.

   Follow the instructions step 3 to update the assembly versions for MapInfo.CoreEngine and MapInfo.VectorTile.Server.

```
<system.webServer>
         <directoryBrowse enabled="true" />
      <handlers accessPolicy="Read, Execute, Script">
         <remove name="ISAPI-dll" />
         <add verb="*" path="*.mvt" name="HttpHandler"
type="MapInfo.VectorTile.Server.HttpHandler,MapInfo.VectorTile.Server,
Version=9.2.X.X, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.d
ll" />
      </handlers>
    </system.webServer>
```

5. Save the web.config file and copy it to the location you created in step 1.

Step 2: Create a Valid Vector Tile Service Configuration File for Hosted Data

The VTSample.xml is a Vector Tile Server configuration file provided with MapXtreme. This file defines information about your Vector Tile server, including the data you want to host, the coordsys in which the data will be served, bounds of the tile grid, etc.

1. MapXtreme allows hosting of multiple tile sets over one server instance. Each tile set should have a unique identifier (specified via its "id" attribute) and contains a <DataSourceDefinitionSet>, which in turn can contain one or more map layers. Register your map layers in an existing <TileSet> by adding map layers in its <DataSourceDefinitionSet> or add a new <TileSet>. The following is a portion of the VtSample.xml that defines a `<TileSet>` for a sample Vector Tile Server.

```
<!-- The following data sources reference local TAB files, you need to replace
MYPATH with the real path to those tab files. -->
```

```
<mxp-vt:TileSets>
   <mxp-vt:TileSet id="Set1" CacheEnabled="on">
      <mxp:DataSourceDefinitionSet>
         <mxp:TABFileDataSourceDefinition id="World">
            <mxp:DataSourceName>World</mxp:DataSourceName>
            <mxp:FileName>MYPATH\World.tab</mxp:FileName>
         </mxp:TABFileDataSourceDefinition>
         <mxp:TABFileDataSourceDefinition id="USA">
            <mxp:DataSourceName>USA</mxp:DataSourceName>
            <mxp:FileName>MYPATH\USA.tab</mxp:FileName>
         </mxp:TABFileDataSourceDefinition>
      </mxp:DataSourceDefinitionSet>
   </mxp-vt:TileSet>
</mxp-vt:TileSets>
```

2. By default, the tiles of a `<TileSet>` are served in EPSG:3857 Coordsys with world bounds. Optionally, you can also specify custom Coordsys and TileBounds. For example,

```
<mxp-vt:TileSet id="Set2">
   <mxp:DataSourceDefinitionSet>
         <mxp:TABFileDataSourceDefinition id="World">
            <mxp:DataSourceName>World</mxp:DataSourceName>
            <mxp:FileName>MYPATH\World.tab</mxp:FileName>
         </mxp:TABFileDataSourceDefinition >
         <mxp:TABFileDataSourceDefinition id="Counties">
            <mxp:DataSourceName>Counties</mxp:DataSourceName>
            <mxp:FileName>MYPATH\UK.TAB</mxp:FileName>
         </mxp:TABFileDataSourceDefinition>
   </mxp:DataSourceDefinitionSet>

   <mxp-vt:TargetCoordSys>
      <mxp-vt:SRSName>EPSG:27700</mxp-vt:SRSName>
   </mxp-vt:TargetCoordSys>

   <mxp-vt:TileBounds minx="200000.0" miny="34000.0" maxx="700000.0"
maxy="600000.0"/>
   </mxp-vt:TileSet>
```

3. By default, each tile of a `<TileSet>` is cached inside the directory specified by the <Cache>/<CacheDir> element. This option can be turned off by setting the TileSet's CacheEnabled attribute to "off".

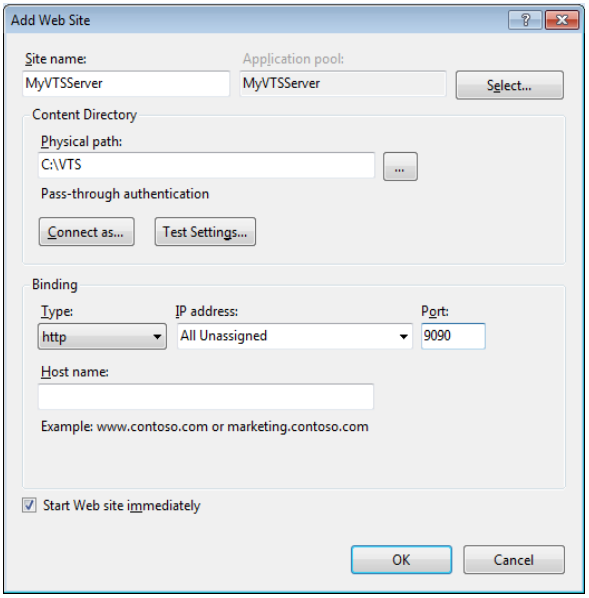
Register your Vector Tile server with Internet Information Services (IIS) and test your setup. Follow the instructions for IIS7.

Step 3: Configure a Vector Tile Server using IIS 7/8.5/10

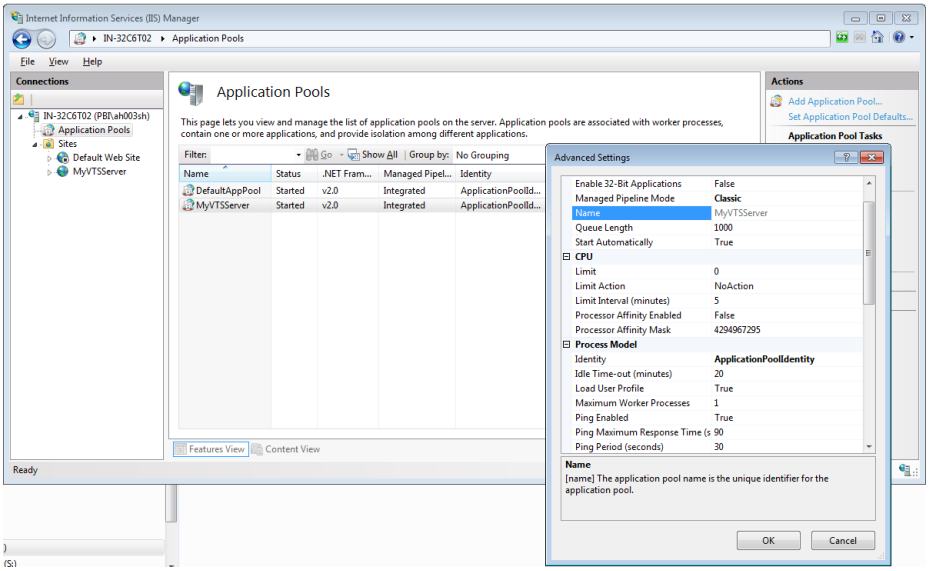
1. Right-click on your VTS folder (for example, c:\vts) and choose Properties. Select the Sharing tab and click the Share button. Add the IUSR account to the user list. Click Share, then Done to return to the Properties dialog. Click the Security tab. Add IUSR account to the security list.

   For information on setting up permissions for the IUSR account, see Understanding Built-In User and Group Accounts in IIS 7/8.5/10.

2. Open the IIS 7 manager. Right-click on Sites and then choose Add Website from the menu. In the dialog that displays, name your site (e.g., "MyVTSServer") and navigate to the physical path of the VTS folder. Click the Connect As button and ensure that application use is checked. Assign a free port number to the website, or use the default 80. Click OK.



3. Click Application Pools in the left pane. Right-click on the new application pool that IIS7 has created for your website and choose Advanced Settings. Change the .Net Framework Version to 'v4.0' Click OK.



Step 4: Testing the Vector Tile Server

Test your setup by requesting a tile from a web browser. In the address box type:

`http://localhost/VTSER/GetTile/Set1/{z}/{y}/{x}.mvt`

Substitute your web server for `localhost`.

A successful test will return a Mapbox vector tile, which is a binary stream and cannot be viewed on a browser. However, the tiles that we get can be consumed by the sample Leaflet Client present in the Samples folder along with the sample server.

The sample client is a small JavaScript application build on top of the Leaflet MapBoxVectorTile Plugin (https://github.com/SpatialServer/Leaflet.MapboxVectorTile).

To access the data hosted over the sample server, you need to provide the server's GetTile url in the L.TileLayer.MVTSource API, as is done in VTClient.js.

```
var mvtSource = new L.TileLayer.MVTSource({
url:"http://localhost/VTSER/GetTile/Set1/{z}/{y}/{x}.mvt", debug: true,
clickableLayers: ["World","USA"], getIDForLayerFeature: function(feature) {
return feature.properties.id;
}
```

Now open the VTClient.html in any web browser to view the data hosted on the server.

If caching is enabled in the server metadata file, then the tiles will be cached in the specified cacheDirectory.

When Error logging option is enabled and there is an exception, a log is generates at the ErrorLogDir path C:\ErrorLog as specified in VtSample.xml.

If you make any changes to the VtSample.xml after you access the Vector Tiles Server, you will need to reset IIS in order for your changes to take affect. To reset IIS, from a command prompt type the following command.

```
c:\>IISreset
```

Configuring Server Metadata Parameters

Layers are described in a Vector Tile Server configuration file according to elements laid out in the **MXP_VT_Configuration.xsd** schema. These elements include:

NamespaceUri - The namespace that the data is located in.

OnlineResource - The URL of this server.

TileSets - Holds the list of tile sets that this server will serve. At least one tile set must be present in the list.

TileSet - Holds the description of a tile set. The server will serve the layers exactly in the order in which the tables are defined here.

TargetCoordSys - The reference projection of the tiles served by this server for this tile set. If not specified, EPSG:3857 (Web Mercator) coordsys will be used by default.

SRSName - A coordinate system identified in the form codespace:code.

TileBounds - Bounds used to form the tile grid for this tile set. If not specified, bounds of the coordsys specified by the TargetCoordSys element will be used. If TargetCoordSys is also not specified, then default bounds of EPSG:3857 coordsys, ie, (- 20026376.39, -20048966.10, 20026376.39, 20048966.10) will be used.

ClippingBufferRatio - This ratio defines the buffer around the tiles at which the features are clipped. Its default value is 0.001, which means that for tiles of extent 4096, the default buffer size is (4096X0.001=) 4.

CacheDir - Cache Directory Path. If CacheEnabled is On for any TileSet, then this element is mandatory. In such a case, the tiles of that tileset will be cached in separate folder inside this directory. The folder name will be same as the identifier (value of the “id” attribute) of this tile-set.

ErrorLog - Option to enable or disable to log the internal server Error for requested tiles. It is Off by default.

ErrorLogDir - ErrorLog Directory Path. By default, ErrorLogDir is temp path of the system if ErrorLogEnabled is ON.

26 – Web Map Tile Service

Web Map Tile Service (WMTS) is an OGC Standard that provides a solution to serve digital maps using pre-rendered map tiles of spatially referenced data using tile images with predefined content, extent, and resolution. For more details, refer OGC document Ref No 07-057r7 (http://portal.opengeospatial.org/files/?artifact_id=35326).

In this chapter:

WMTS support in MapXtreme

MapXtreme provides the ability to access pre-rendered map tiles hosted by an OGC WMTS compliant server that supports HTTP GET requests with KVP (Key-Value Pair) and/or REST(Representational State Transfer) encodings.

This is all possible through MapXtreme's object model, an API of 100 percent managed code that was developed on Microsoft's .NET Framework. The Framework's Common Language Runtime (CLR) provides the foundation that makes simplified development a reality.

There are two ways to consume a WMTS service in MapXtreme:

- For built-in support, use Tile Server TAB/XML files generated by MapInfo Pro similar to Bing tile server tab file. For more information, see section WMTS (Web Map Tile Service) of Chapter 18 of this document.
- Use WMTS client classes for customized support. MapXtreme's MapInfo.Wmts namespace provides WmtsClient class to access the OGC compliant WMTS servers (without TAB file). The MapXtreme WmtsClient can request tiles served by a WMTS server via APIs, and the application can then mosaic the tiles and clip them into a final image.

In this chapter, we will explain the second approach.

WmtsClient Class

The WmtsClient is a concrete class that allows an application to communicate with OGC compliant WMTS server. WmtsClient class have all the APIs support which are mandatory for any OGC WMTS compliant servers i.e. GetCapabilities, GetTile and optional GetFeatureInfo.

The WMTS interface allows a client to receive three types of resources in response to the following three types of requests.

- GetCapabilities
- GetTile
- GetFeatureInfo

GetCapabilities

The client initiates interaction with a WMTS server by requesting its ServiceMetadata document via a GetCapabilities request. In MapXtreme, this happens at the time of instantiation of MapInfo.Wmts.WmtsClient object.

The ServiceMetadata document lists the layers and the tiles available in each layer, in each graphical representation style, in each format, in each coordinate reference system, at each scale, and over each geographic fragment of the total covered area. The ServiceMetadata document also declares the communication protocols and encodings (KVP or REST) through which the client can interact with the server.

GetTile

After fetching the ServiceMetadata document, the client can use the information in that document to discover how to perform valid requests for tiles via GetTile requests.

A GetTile request contains the following parameters.

- Version - WMTS standard version, currently only version 1.0.0 is supported in MapXtreme.
- Layer - Layer identifier
- Style - Style identifier
- Format - Output format of the tile (png, jpeg, etc.)
- TileMatrixSet - TileMatrixSet identifier
- TileMatrix - TileMatrix identifier
- TileRow - Row index of tile matrix
- TileCol - Column index of tile matrix

GetFeatureInfo

WMTS servers may support optional GetFeatureInfo requests for information about the features present at a particular pixel location on a map tile. In such a case, the client can make GetFeatureInfo requests by including the following parameters in addition to the ones that are required in a GetTile request.

- J - Row index of a pixel within the tile.
- I - Column index of a pixel within the tile.
- InfoFormat - Output format of the retrieved information.

Code Example: Requesting a WMTS Layer

The following example demonstrates how to request a WMTS layer programmatically with MapXtreme.

```
string wmtsUrl = @"http://maps.warwickshire.gov.uk/gs/gwc/service/wmts";
        string version = "1.0.0";
```

```
        // Create an instance of WmtsClient. This sends requests to fetch the
ServiceMetadata document.

        MapInfo.Wmts.WmtsClient client = new MapInfo.Wmts.WmtsClient(wmtsUrl,
version);

        //Capabilities property gives access to the ServiceMetadata document.
        MapInfo.Wmts.IWmtsCapabilities capabilities = client.Capabilities;

        //Create a GetTile request from capabilities.

        MapInfo.Wmts.GetTileRequest tileRequest = new
MapInfo.Wmts.GetTileRequest();

        //Select the first layer in the layers list.
        IWmtsLayer firstLayer = capabilities.LayerInfos.First();
            tileRequest.Layer = firstLayer.Identifier;

        //Select the first TileMatrixSet of the selected layer.
            tileRequest.TileMatrixSet =
firstLayer.TileMatrixSetLinks.First().TileMatrixSet;

        IWmtsTileMatrixSet tileMatrixSet = null;

        //Get the contents of selected tileMaxtrixSet from the list of
TileMatrixSets.
        foreach (IWmtsTileMatrixSet tmSet in capabilities.TileMatrixSets)
        {
           if (tmSet.Identifier.Equals(tileRequest.TileMatrixSet))
           {
              tileMatrixSet = tmSet;
              break;
           }
        }

        // Select the first TileMatrix of the selected TileMatrixSet.
        tileRequest.TileMatrix = tileMatrixSet.Matrices.First().Identifier;

        // Select the first format in the list of supported image formats.
        tileRequest.Format = firstLayer.Formats.First();

        // Select the tile at position (0,0) in the tile-matrix.
        tileRequest.TileRow = 0;
        tileRequest.TileCol = 0;

        //Execute the GetTile request.
        byte[] tileBytes = client.GetTile(tileRequest);

        System.Drawing.Image img = System.Drawing.Image.FromStream(new
MemoryStream(tileBytes));
        Assertion.AssertEquals(img.RawFormat,
System.Drawing.Imaging.ImageFormat.Png);
```

```
        //Check whether the server supports GetFeatureInfo operation.
        if (capabilities.IsGetFeatureInfoSupported)
        {
          // Check if the layer is queryable, ie, any infoFormats are specified
for this layer.
          if (firstLayer.InfoFormats != null && firstLayer.InfoFormats.Count()
> 0)
          {
            //Create a GetFeatureInfo request to retrieve feature info of pixel
(10,10) of the tile.
            MapInfo.Wmts.GetFeatureInfoRequest fiRequest = new
MapInfo.Wmts.GetFeatureInfoRequest(tileRequest, 10, 10,
firstLayer.InfoFormats.First());
            //Execute the GetFeatureInfo request.
            byte[] featureInfo = client.GetFeatureInfo(fiRequest);
          }
        }
```

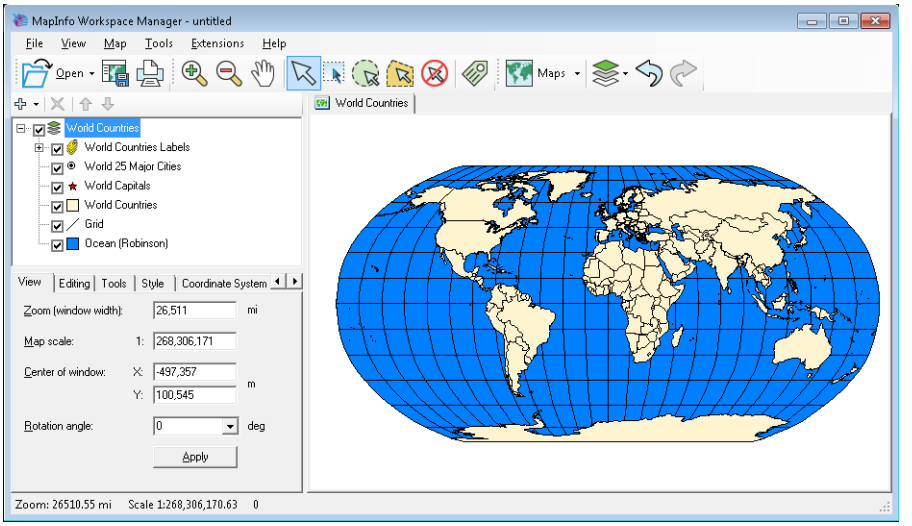
27 – Workspace Manager

The Workspace Manager utility that comes with MapXtreme enables you to create and manage XML-based workspace files (.MWS format). The Workspace Manager's convenient user interface allows you to easily assemble the tables that make up your map, apply many additional settings, and save the map as a workspace. Your completed workspace is then ready for use in your own applications, or for use in print or file output.

In this chapter:

Features of the Workspace Manager

The Workspace Manager allows you to control most of the settings that can be stored in a workspace file. For example, a workspace can contain information about cartographic legends and adornments; however, the Workspace Manager does not provide any options for creating cartographic legends or adornments. To create cartographic legends or adornments, use the API.



Through the Workspace Manager, you can:

- Load XML workspaces, tables, geosets, and MapInfo workspaces (.mws not .wor files).
- Save workspaces as .MWS.
- Control which tables are opened as part of a workspace.
- Create and load named connections using the Named Connection Manager.
- Add, remove, and view one or more maps.
- Toggle among maps using the tabs across the top of the Map window.
- Set properties for map and layer visibility, layer and label styles, and themes with the built in layer control.
- Add, remove, or alter custom labels.
- Create group layers, which allow you to organize your layers into logical groupings, so that you can show or hide the entire group with a single click.
- View multiple next and previous map views.

- Use map tools for navigation and manual label placement, and use selection tools to verify that layer selectability settings are correct.
- Preview and print maps.
- Quickly open recent workspaces from the recent file list
- Create translucent effects for maps, labels, and theme layers
- Create curved labels for polyline features.
- Add a graticule layer to your map.
- Reproject raster layers to current map window projection.
- Reproject a map window made up of raster and vector layers to a different projection.
- Add extensions to add custom functionality to Workspace Manager and Layer Control.

Workspace Format and Contents

The workspace file is an XML document (.MWS) that contains the locations, descriptions, and metadata of all the maps, tables, layers, and settings that make up the workspace. Because it is XML, the workspace is portable, which means that you can share the workspace with other users working on different computers, on different networks, across locales.

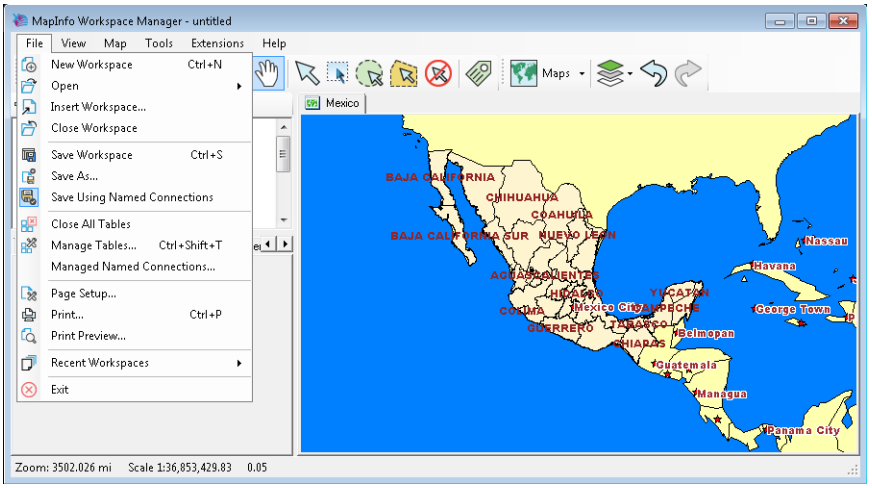
In MapXtreme, the portability of the XML workspace is implemented with named connections. Named connections enable you to define alternative drive, path, and database connection information based on your own environment, so that you can use workspaces created by others. All paths and connection strings are resolved when the workspace is opened. You can set up named connections directly in the Workspace Manager (**File > Manage Named Connections**). For more information on the XML workspace structure, please see Appendix C: Understanding the MapInfo Workspace.

Workspace Manager Menu Commands

This section explains each menu command available in the Workspace Manager.

File Menu Commands

The commands in the File menu provide all the standard File menu capabilities, such as opening, saving and printing files, as well as some features unique to the Workspace manager. Each command is described below.



New Workspace

Creates a new empty map to which you can then add tables using either the Add tool in layer control or the Open Tables command from the File menu. If changes have been made to the current workspace, you will be asked if you want to save the changes before the new workspace is created.

Open

Opens an existing workspace or table. If changes have been made to the current workspace, you will be asked if you want to save the changes before the existing workspace is opened. The Open Tables command enables you to open one or more tables and add them to your map.

Insert Workspace

Adds the maps and tables from one or more workspaces into the current workspace.

Close Workspace

Prompts you to save any changes and closed the workspace.

Save Workspace

Saves your map as a workspace.

Save As

Save a copy of the workspace to a new filename.

Save Using Named Connections

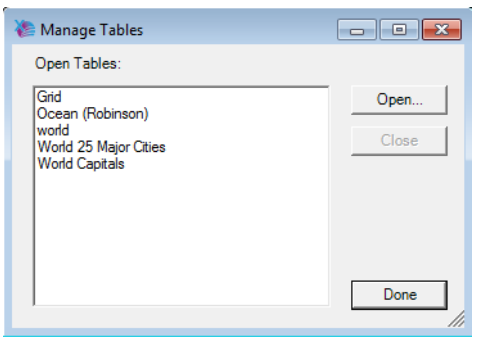
Saves named connection information to the workspace. See also Manage Named Connections.

Close All Tables

The Close All Tables command closes all of the open tables.

Manage Tables

The Manage Tables command displays a dialog box that lists the tables that make up the map and enables you to open additional tables for possible inclusion in the map. Click **Open** to display the Open dialog box and open a table. The table you opened is added to the list of open tables in the Manage Tables dialog box. Then you can add the table to the map using the Add tool, which is located over the Workspace Manager layer control window.

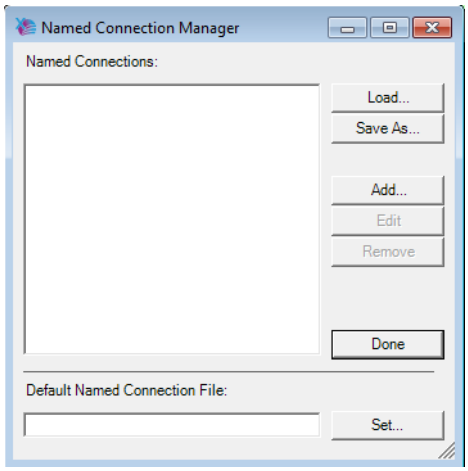


To close a table, click a table from the list to highlight it. The **Close** button is activated. Click **Close** to close the table. Layers referencing the table you closed are removed from the map.

A workspace can open tables that are not already in a map.

Manage Named Connections

A named connection describes a connection to a data source using an alias. You can create the following types of named connections: FilePath, DatabaseSource, ODBC, or Oracle OCI. After you specify the connect string or file path, you can save it as XML for later retrieval. You can set a default connection so that when you run Workspace Manager, the connection to your data source is available. Named connections are also saved to the workspace.



Page Setup

The Page Setup command enables you to specify the paper size, orientation, and margins of the printed map. You can also use this option to access printer-specific settings.

Print

The Print command enables you to print your map to paper or file output. In the Print dialog box, specify the printer to use, the page range you want to print if your job is multiple pages, and the number of copies to print. Printer properties enable you to set layout and other options that are specific to the printer you are using. The **Print to file** check box enables you to print your output to a file.

Print Preview

Use the Print Preview command to see how your output is going to look before you print it.

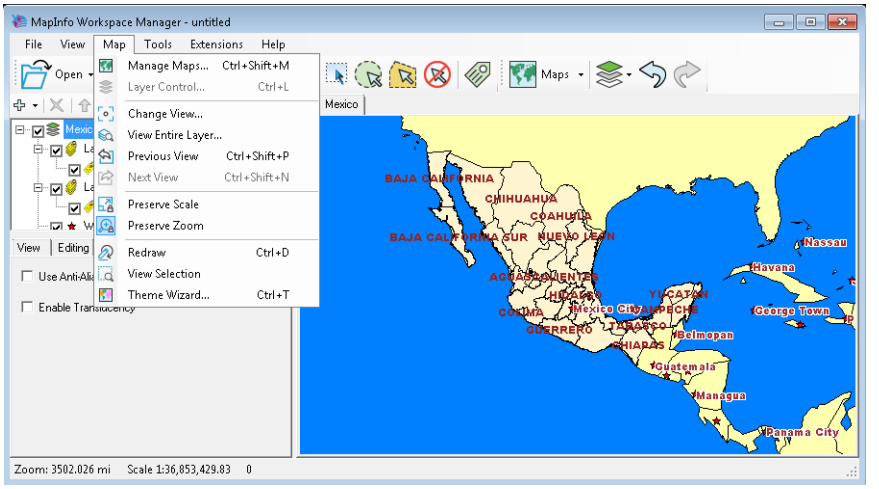
Recent Workspaces

Recent Workspaces shows a list of recently opened workspaces.

View Menu Commands

This menu allows you to show or hide Toolbars, Status Bar and Layer Control.

Map Menu Commands



Use the commands in the Map menu to add and remove maps, manipulate the view of the map, and create thematic maps. The view commands are also available in a popup menu. Right-click in the Workspace Manager map window to display the menu.

Manage Maps

From this menu the Manage Maps dialog appears where you can add or remove maps, set a map as default, rename the map and/or alias and show/hide the legend tab (if map has a legend). The New command enables you to create a new map window using the tables that are currently open. The Manage Workspace Maps also provide the number of layers and legends in the map.

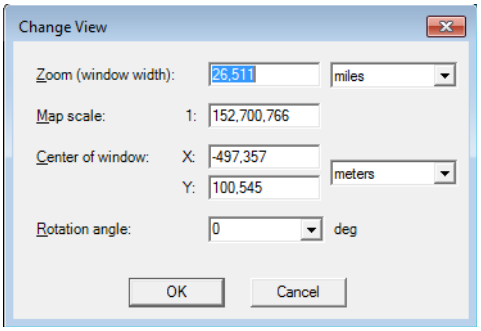
Layer Control

This menu command allows you to bring up the Layer Control dialog. It is active when the View > Layer Control is unchecked and Layer Control is not visible. For a description of Layer Control's features, see Layer Control Tools.

Change View

The Change View command enables you to change the current view of the map—that is, what area of the map is currently displayed in the window. You can change the zoom and scale of the map to your own settings. You can also set the center of the map window, or change the rotation angle.

The Change View dialog box allows you to choose the units (miles, kilometers, etc.) for the zoom width and for the center X/Y coordinates (meters, degrees, etc.). Whatever units you choose in the Change View dialog box are also used in the layer control. For example, if you want all distances in the layer control to be displayed in kilometers instead of miles, display the Change View dialog box and choose kilometers from the units list that appears next to the Zoom field.



View Entire Layer

Use the View Entire Layer command to see an entire layer, or all the layers in the map. The View Entire Layer dialog box shows a list of the layers that make up the map. Select the desired layer from the list, or choose **All Layers** so that all the layers are completely in view, and click **OK**. The map redraws to display the entire layer.

Previous View

Use the Previous View command to return to the previous view of your map.

Next View

The Next View command is available after you have used Previous View. Use it to redisplay the view of the map that was on the screen before you used the Previous View command.

The Previous View and Next View commands can be used together to toggle back and forth between two views of your map. These commands are also available as tools on the toolbar.

Preserve Scale/Zoom

Use the Preserve Scale and Preserve Zoom commands to keep the zoom and/or the scale the same as you change the size and shape of the map.

Redraw

Use the Redraw command to redraw the map.

View Selection

Use the View Selection command to zoom in or out on a selected object or objects.

Theme Wizard

You can create feature themes and label themes via the **Map > Theme Wizard** menu command. Feature themes include ranged, individual value, dot density, graduated symbol, and pie and bar charts. Label themes include ranged and individual value.

If your map includes at least one set of labels, which are displayed in a label layer, you can create a label theme. A label theme assigns different label styles (colors, font size etc.) to each label, based on the data in your table. For example, use a label theme to show the prominence of locations over others. A ranged label theme groups labels based on a similar data value, such as population. Cities that fall within a certain population range are labeled using one style, while cities in other ranges are labeled in another style, typically a less prominent style to indicate city size without having to label using the population value.

The Theme Wizard walks you through 3 dialogs to create your theme.

To create a theme:

1. Choose **Map > Theme Wizard**.

   The Create Theme: Step 1 of 3 dialog box appears.

2. Select either a Feature theme or a label theme, and choose the type of theme you want to create.

3. Click **Next**.

   The Create Theme: Step 2 of 3 dialog box appears.

4. Select the table you want to shade.

5. Choose the data you want to use. Select either a column from the table that contains the data, or select **Expression** to use an expression to derive the data you want from the table.

6. Click **Next**.

The Create Theme: Step 3 of 3 dialog box appears. Here you can customize theme type settings, styles, and the legend. (See How to Apply Translucent Effects to Themes for instructions on applying translucency effects to a thematic map.)

7. Click **Apply** to apply the customized settings.
8. Click **OK** when you are finished.

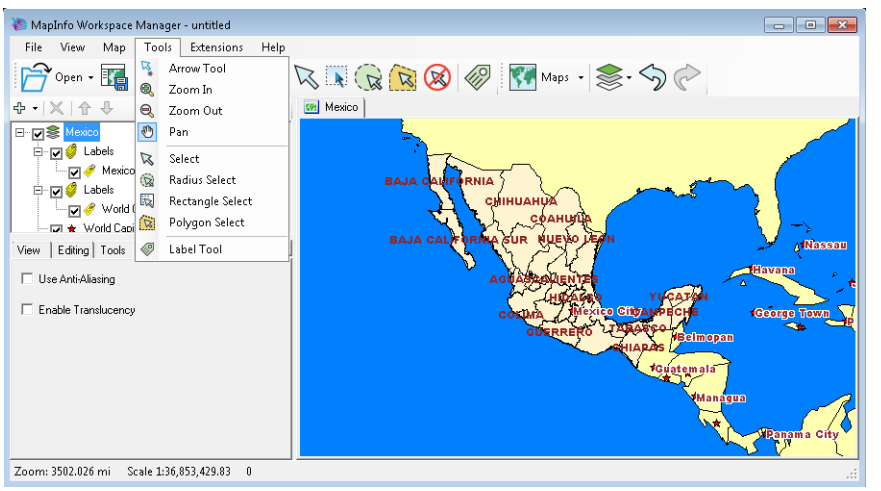
The Ignore Zeros option on the Step 2 of 3 dialog is available when creating Pie and Bar chart themes and IndividualValue themes on Features or Labels. Note that when this option is set, the performance of the theme building operation in Workspace Manager may be affected.

To modify a theme, highlight the theme in the Layer tree and click the Theme tab that displays. Click the Modify Theme button to make any changes to the theme.

To remove the theme from the Layer list, right-click on the name of the theme and choose Remove. You can also rename the theme and show/hide this layer from the Layer tree.

Tools Menu Commands

The Workspace Manager's Tools menu provides access to the map tools via menu commands. These tools enable you to zoom in and out on the map, change the position of the map, and select map objects in various ways. These same tools are also available on the Workspace Manager toolbar. Each tool is explained below.



Arrow

This is a basic pointing tool. It does not select map features.

Zoom In

Use the Zoom In tool to get a closer area view of your map. To zoom in on a map:

1. Choose **Tools>Zoom In** to activate the tool.

   Your cursor changes to a magnifying glass with a plus sign in it.

2. Click on your map.

   The map redraws at a closer area view, centering itself at the point you clicked.

Zoom Out

Use the Zoom Out tool to get a wider area view of your map. To zoom out on a map:

1. Choose **Tools>Zoom Out** to activate the tool.

   Your cursor changes to a magnifying glass with a minus sign in it.

2. Click on your map.

   The map redraws at a wider area view, centering itself at the point you clicked.

One tool Zoom In and Zoom Out

To zoom in and out using the same zoom tool, hold down the Control key. When using the Zoom In tool with Control you will zoom out stepwise with each click. When using the Zoom Out tool while holding down the Control key, you will zoom in stepwise with each click.

Pan

Use the Pan tool to reposition your map without changing the zoom level. For example, you might want to redirect the view of your map so that a certain country or city is in the center. To pan your map:

1. Choose **Tools>Pan** to activate the tool.

   Your cursor changes to a hand icon.

2. Click on the map, and while holding down the mouse button, drag the map to the desired position.

   The map redraws reflecting the new position.

Select

Use the Select tool to select objects one at a time or to select all objects that are generally in the same area.

To select an object using the Select tool:

1. Choose **Tools>Select** to activate the tool.

   The cursor changes to an arrow.

2. Click the object on the map you want to select.

   The selected object is highlighted.

Radius Select

Use the Radius Select tool to select all objects that fall within a given radius. For example, you have a table of blood donors and a table of blood donation sites. Using the Radius Select tool, you could create a temporary list of blood donors that live within a one-half-mile radius of each blood donation site.

Note that the Radius Select tool selects all objects whose centroid falls within the circle. The object does not have to be completely bounded by the circle. To select objects within a radius:

1. Choose **Tools>Radius Select** to activate the tool.

   The cursor changes to a selection arrow with a small circle underneath it when you move the cursor over the map.

2. Click a place on the map that you would like to use as the center point of your radius search. For example, if you want to select all the fire hydrants that fall within two miles of a fire station, click the fire station and use that as the center point.

3. Hold down the mouse button and drag the mouse away from the center point.

   The Workspace Manager draws a circle around the point and reports the radius of the circle in the StatusBar (lower left corner of the screen).

4. When you have the desired radius release the mouse button.

   Workspace Manager highlights all map objects that fall within that circle.

Rectangle Select

Use the Rectangle Select tool to select objects within a rectangle. By clicking and dragging using the Rectangle Select tool, you create a dotted rectangle, or marquee box around objects you want to select.

Note that the Rectangle Select tool selects all objects whose centroid falls within the rectangle. The object does not have to be completely bounded by the rectangle .To select objects within a rectangle:

1. Choose **Tools>Rectangle Select** to activate the tool. The cursor changes to a selection arrow with a small rectangle underneath it when you move the cursor over the map.
2. Click a place on the map outside of the area you want to include in the marquee box.
3. Hold down the mouse button and drag the mouse to form a dotted rectangle around the points you want to select.
4. When you have reached the desired rectangle size release the mouse button.

   Workspace Manager highlights all map objects that fall within that rectangle.

Polygon Select

The Polygon Select tool selects map objects within a polygon that you draw on a map.

To select objects with the Polygon Select tool:

1. Choose **Tools>Polygon Select** to activate the tool.

   The cursor changes to a selection arrow with a small polygon underneath it when you move the cursor over the map.
2. Click the map location at which you want to place the first end point of the polygon. Move the cursor over your map in any direction.

   Workspace Manager draws a line from the point where you clicked to the cursor.
3. Click to create another endpoint. Continue to move the cursor and click until you have the desired number of sides to your polygon.
4. To close the polygon, make your last click as close as possible to the first click, or double-click on the last point.

   Workspace Manager closes the polygon and selects the objects that are within it.

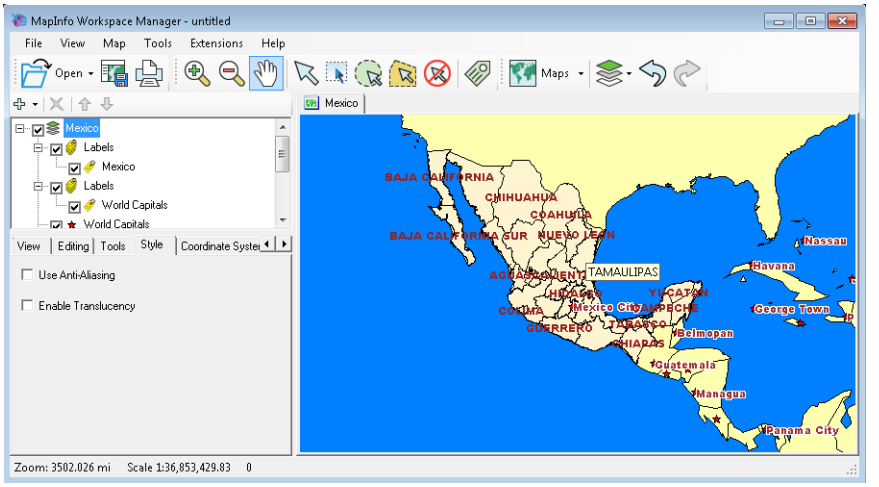
Label

The Label tool is used to manually add labels to a map at the location where the user clicks. Both horizontal and curves labels can be drawn with the label tool. In its simplest use the Label tool draws a label with the default settings. But MapXtreme's extensive labeling capabilities provides a tremendous number of style, position and visibility options that you can use to bring your labels to life. See also Label Layer Settings and Curved Labels.

Extensions Menu Commands

Beginning with MapXtreme v7.0.0, you can load custom .NET assemblies into Workspace Manager to extend its functionality via the Extensions menu. For example, you may wish to have multiple tab windows, each with settings that apply to a different map. Extensions can also add menu items to any of the Workspace Manager menu, including the Extensions menu.

This menu provides two commands: Load Extension and Manage Extensions.



Load Extension

This command displays an Open dialog where you can navigate to the location of your custom extensions. Extensions are created and saved as .DLLs using the file naming convention <*your_extension*>.WorkspaceManager.Extensions.dll.

Manage Extensions

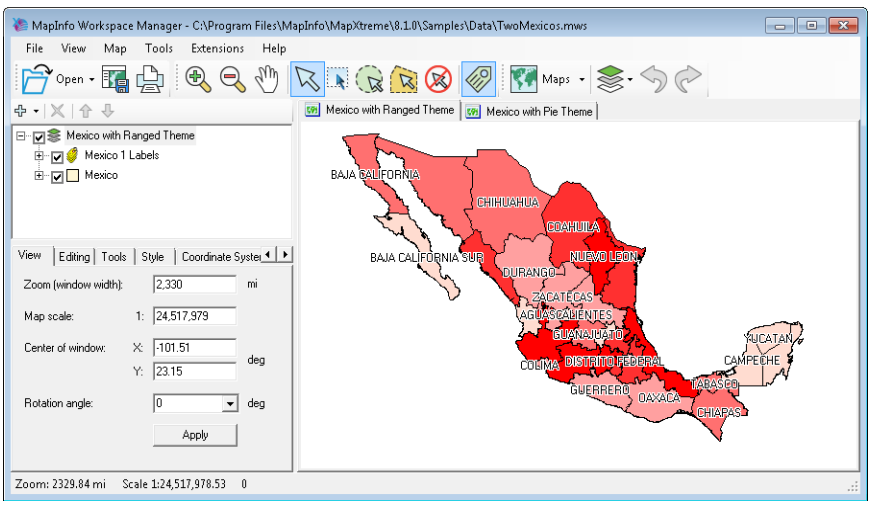
This command displays the Extension Manager dialog where you can load or unload extensions, and modify properties of an extension. The loaded extensions display in the list box showing name, version and assembly location.

To learn how to create custom extensions, see Workspace Manager Extensions in Appendix B: Customizing MapXtreme.

Layer Control

The Workspace Manager application window is divided into two main sections. The layer control window and commands are located on the left, and the map window is on the right. The layer control window consists of the layer tree, which displays the opened map(s) and all of its accompanying layers, and the dialog box tabs that contain map and layer settings.

The layer control features of the Workspace Manager enable you to assemble the layers of your map and apply settings to individual layers or the entire map that govern how the layer(s) or map display.



Layer Control Tools

The tools across the top of the Layer Control allow you to add, move, and remove layers from the layer tree window easily:

- The Add tools allows you to open tables, and insert group layers and label layers into your map.
- The Remove Selected Item tool removes the selected layer from the map.
- The Up and Down toolbar buttons enable you to move layers up and down the layer list, changing the order in which they are displayed.

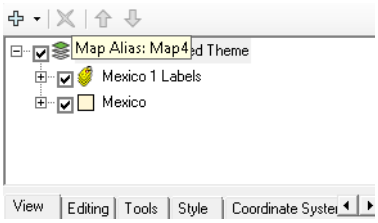
Layer Tree

The layer control displays a tree showing the map and all layers in the map, including theme, label, and graticule layers. The layer tree allows you to perform these operations:

Map Alias

When you hover the mouse over the Map node (the node at the top of the layer tree), the tooltip shows the map alias. The tooltip aids developers who need to know the map alias (for example, you need to specify a map alias when setting properties on web controls).

You can change a map alias by right-clicking on a map name and choosing Set Map Alias.



Displaying Layers

The check box next to each layer on the layer tree allows you to toggle the visibility of a layer with a single click.

Changing the Layer Order

To change the order of the layers, you can select a layer and click the Up or Down toolbar button. Alternately, you can drag a layer up or down to change its position in the list.

There are several special cases that involve drag-and-drop actions:

- To move a layer into a group layer, drag the layer onto the group.
- To add a new label source to a label layer, drag a layer onto the label layer.

If you do not want to move a layer into a group—if, instead, you want to reposition the layer so that it is located above the group layer—hold down the Shift key before completing the drag-and-drop action. Similarly, if you do not want to add a new label source to a label layer, hold down the Shift key.

Displaying Context Menus

Each map or layer in the layer tree has a context menu. To display a context menu, right-click on the map or layer, or press **Shift+F10** to display the menu for the currently-selected layer.

The items on the context menu depend on the type of layer specified. In particular, note the following:

- To add a style override to a layer, display the context menu and choose Add Style Override. Note that each layer can have multiple style overrides, each with a different zoom range; this allows you set up the map so that points appear to grow larger, and roads appear to grow wider, as you zoom in.

  You can also add style overrides to seamless raster layers. When the style override is highlighted, you can access the Raster Image Style dialog from the Visibility tab.

- To rename any item in the layer tree, right-click the item and choose Rename. Alternately, you can press **F2** to rename the selected item. Note that renaming a layer in this manner does not rename the original table; the rename operation simply changes the text that is displayed in the layer tree. The information is stored in the workspace file when the workspace is saved.
- To remove a layer in the layer tree, right-click the item and choose Remove. Alternately, you can press the **Del** key to delete the layer.
- By default, a layer is selectable when it is added to a map. This is controlled by the Selectable check box of the Options tab in Layer Control (see Options). However, you can designate it as the only selectable layer, and all other layers will be set to unselectable. This is very convenient if you have a map with many layers but only want one layer to have selectable features. To specify only one layer as selectable:

  a. Right-click the layer name in the Layer Control or access the context menu.

  b. Select **Make This the Only Selectable Layer**. This single map layer now has selectable features. That is, you can use any of the selection tools to select objects on the map. All other map layers will be unselectable.

Layer Control Tabs

The layer control tabs underneath the layer tree provide additional settings and controls that you can apply to the map as well as to each layer in the map. Different tabs control map and layer settings. The tabs that appear depend on whether the map or a layer is highlighted in the layer tree, as well as what kind of layer is highlighted. For example, when you highlight the map in the layer tree, the map tabs appear; when you highlight a layer, the layer tabs appear. If you highlight a theme, label, or graticule layer, tabs that are specific to those layers will appear. The next sections explain the options in each of the tabs.

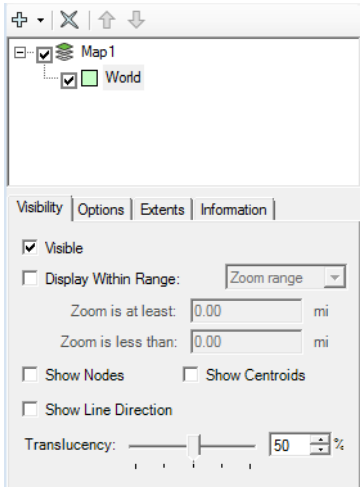
Map Settings

When you select a map in the layer tree, the following tabs are available: View, Editing, Tools, Style, Coordinate System, and Extents.

View

The View tab enables you to control the overall appearance of the map. You can set the zoom level, scale, a center point (in degrees), and a rotation angle. Click the **Apply** button to apply your settings.

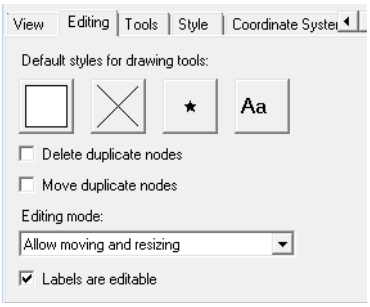
A new Translucency trackbar has been added in this tab in version 8.1 and later that enables you to set the translucency of the selected layer.



ⓘ For raster reprojection, set the rotation angle in this tab and turn the reprojection on/off from the Raster Reprojection tab. See Raster Reprojection.

Editing

The options in the Editing tab enable you to control certain map editing tasks such as the styles used for drawing objects (if your application uses drawing tools), resizing objects, and moving and deleting object nodes.



The style boxes enable you to specify the default styles of any drawing tools that your application uses. Click on a box to open the corresponding style dialog box. The settings you select are saved in the workspace. When a user opens the workspace in an application that uses drawing tools, the application uses these style settings when the user draws objects on the map.

You can also specify whether you want to delete or move duplicate nodes. Check the appropriate checkbox.

You can specify an Edit mode for the map:

- *None* – No editing can be done on the map.
- *Allow moving and resizing* – Map objects can be moved and resized.
- *Allow node editing* – Nodes can be moved or deleted.
- *Allow node adding* – You can add nodes to objects.

The Edit mode you select applies to all editable layers in the map. You control a layer's editability by checking the box Editable on the layer's Options tab.

When in Edit mode, it is a good idea to turn on the nodes for your map features. Highlight the layer node (not the map node) and check the box on the Visibility tab.

To control whether labels are editable, check or clear the checkbox *Labels are editable*.

For an example that utilizes Edit mode, see Editing a FeatureGeometry with the Select Tool.

Tools

The Tools tab enables you to control the display of InfoTips, activate Snap to Nodes and set a snap tolerance, and activate Dynamic Selection tools.

The *Show InfoTips* checkbox controls whether information about the feature displays in a pop-up when you hover over the feature with a select tool. See Options.

If the *Snap To Node* check box is selected, map tools such as the Select tool will automatically search for nodes that are nearby. If a node is nearby, a crosshair will appear to indicate the position of the nearest node. For example, you might want to select the Snap To Node check box if you are using the Radius Search tool, and you want to make sure that the search is centered at the exact location of a point feature on your map. The Snap To Node feature is particularly important in applications that provide drawing tools, because users often need to draw features at the exact location of existing features.

The *Snap Tolerance* setting specifies how far the tools will search for "snappable" nodes. You can choose which layers use the Snap to Node feature. For example, you might want to turn on Snap To Node, but only have the snap crosshair appear when the cursor is

near a feature in a particular layer. To turn the Snap To Node feature on or off for a specific layer, select the layer in the layer tree, then select or clear the **Snap To Node** check box in the Options tab.

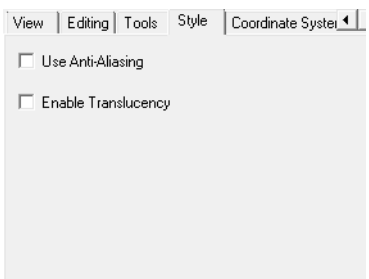
The *Dynamic Selection Tools* check box controls whether features are selected immediately (while you are using the selection tool) or selected when you release the mouse button to finish using the selection tool:

If the Dynamic Selection Tools check box is not selected, selection tools do not actually select any features on the map until you finish using the tool. For example, the Radius Select tool will not select any features until you specify a radius and release the mouse button.

If the Dynamic Selection Tools check box is selected, features become selected or de-selected dynamically as you drag the mouse. For example, if you use the Radius Select tool, you will see more features become selected as you drag the mouse to enlarge the radius.

Style

The options in the Style tab enable you to control translucency and anti-aliasing properties.



- **Use Anti-Aliasing**–Use this option to smooth jagged edges of lines, curves, and region borders when representing a high-definition rendition at a lower resolution. When you select Use Anti-Aliasing, Enable Translucency is also selected automatically. Whenever Enable Translucency is deselected, Use Anti-Aliasing is automatically deselected.
- **Enable Translucency**–Use this option to allow translucent values in style colors and layers when drawing the map onto the screen, printer, or file export. When translucency is enabled, you can use the translucency trackbar in style dialogs.

  This property has no effect on raster translucency; however, Enable Translucency must be selected to print translucent raster images.

ⓘ Rendering higher quality maps by enabling translucency and anti-aliasing, particularly in a map with three or more transparent layers, will often result in a slower rendering speed.

For more information on anti-aliasing and translucency, see Enhanced Rendering with GDI+ Translucency and Anti-Aliasing.

Coordinate System

The Coordinate System tab indicates the coordinate system of the map, and enables you to change the coordinate system.

To do this:

1. Click the **Coordinate System** button to display the Choose Coordinate System dialog box.
2. Select a coordinate system from the list, and click **OK**.

Extents

In the map, the Extents tab shows the extents of the current map view. Click the **View Entire Map** button to see all of the map.

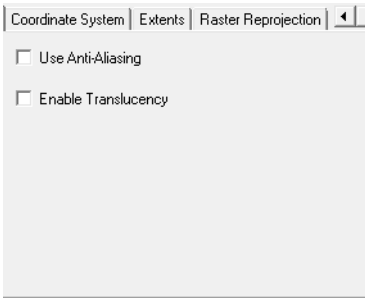
Raster Reprojection

The Raster Reprojection tab enables you to specify reprojection settings for raster and vector layers in your map.

You can control the reprojection of both raster and vector layers. When you add either a raster or vector layer to a map, the new layer can be reprojected into the current map window projection.

When you change the projection of a map window that contains a combination of vector and raster layers, all the layers, both raster and vector, can be reprojected to the new map window projection.

These settings also exposed in the MapXtreme API. See the RasterReprojection and RasterReprojectionResampling properties in the MapInfo.Mapping.FeatureViewer class in the Developer Reference for more information.



Use the Raster Reprojection check box to turn Raster Reprojection on or off. Select the check box to turn it on; clear it to turn it off. Raster reprojection is off by default, which means that reprojection is not performed when a raster layer is added to a map.

Use the Reprojection Method settings to specify how raster reprojection is performed. Choose one of the following:

- **Optimized**–Raster reprojection for an image is determined by the look of the destination rectangle after transformation into the source image space. If the destination rectangle looks like a "rigorous" rectangle, that is, two sides of the rectangle are parallel to the x-axis and two sides are parallel to the y-axis, then the standard Windows functions stretch the source image in both directions. If the image fails the rectangle text, the reprojection is performed using the specified resampling method.
- **Always–Raster** reprojection is always performed. MapXtreme calculates the image's coordinates based on a precise formula and then the pixels are resampled using either the Cubic Convolution or Nearest Neighbor methods.

Use the Reprojection Resampling settings to specify a resampling method for the reprojected raster image.

Resampling is a method of restoring the pixel value (usually proportional to the brightness of a spot on the ground object covered by the instant field of view of the image sensor) of the destination raster image based on the source image being a discrete representation of the initial continuous brightness field.

The pixel values of the destination image can be calculated using one of two resampling methods: Cubic Convolution and Nearest Neighbor. Choose one of the following:

- The **Cubic Convolution method** provides the best "restoration" of pixel values because of their discreteness. In Cubic Convolution, a pixel in the destination image is calculated based on the pixel values in a 4x4 pixel window centered at the basic pixel in the source image. The coordinates of the basic pixel are calculated for every pixel of the destination image based on a special optimized procedure. Pixels are then

weighted based on the basic pixel coordinates. In general, we recommend that you use the Cubic Convolution resampling method for aerial images and satellite raster images to get a better image quality. [1]

- The **Nearest Neighbor** method replaces the pixel value in the reprojected image with the base pixel value from the source image. This resampling method takes less time to render than the Cubic Convolution method, but it may be less precise. In general, we recommend that you use the Nearest Neighbor resampling method for raster maps, grids, and scanned maps to get faster results.

To change the rotation angle of the raster reprojection, go to the View tab, provide an angle and click Apply. See View.

Layer Settings

When you click on a map layer node in the Layer tree, the following tabs are available in Layer Control.

Visibility

Select the **Visible** check box to make the map layer visible. Selecting the check box next to the layer in the layer tree has the same effect.

Select the **Display Within Range** check box to specify either a zoom range or scale range in which the layer appears. If you select a zoom range, specify the minimum and maximum zoom in miles. The layer appears within this range. If you select a scale range, specify the closest and farthest scale. The layer appears within this scale range.

You can also select **Show Nodes**, **Show Centroids**, and **Show Line Direction** to display these items on the map layer. Nodes are the points that define segments of a line or multiline or polygon. A centroid is the center of a map object. Line direction is the direction in which the line was drawn (this is helpful on street layers to indicate the proper addressing sequence). Display these elements when you wish to edit map features. The Editable checkbox is located on the Options tab.

Options

The Options tab check boxes facilitate editing and customizing a feature layer:

- **Selectable**–When the Selectable check box is selected, objects in the layer can be selected using either the Tool menu commands or the Selecting tools in the toolbar. Clear the Selectable check box for any layer you do not want to select from.

1. The Cubic Convolution algorithm used in MapXtreme is based on the work of S.K. Park and R.A. Schowengerdt, *Computervision*, Graphics and Image Processing (1983, Volume 23, pp. 258-272).

- **Editable**–Select the check box to make the layer editable.
- **Drawing Tools can add features** –Select this check box if you are preparing this workspace for use in an application that provides drawing tools, and you want the drawing tools to create new features in this layer.
- **Show InfoTips**–Select the Show InfoTips check box to display InfoTips when you hover over map objects in the selected layer. The InfoTip text consists of the result of the expression in the InfoTip Expression field. For example, if the expression is a column in your table, the InfoTips comprise the values from that column. If the expression is a calculation that uses column information in your table, the InfoTips comprise the results of that calculation.
- **Snap to Nodes** – Select to turn the Snap To Node feature on or off for a specific layer, select the layer in the layer tree, then select or clear the Snap To Node check box.
- **InfoTip Expression** – Control the InfoTip you want displayed in the InfoTip for the layer. You can display more than the first column in your table. See Chapter 10 Creating Expressions.

Settings in the Options tab are unaffected by the layer's visibility. You will be able to see the Options tab settings regardless of whether the layer is currently visible.

Extents

For a layer, the Extents tab shows the extents of the selected layer. Click the **View Entire Layer** button to see all of the layer, or click **View Default Area** to see the default view of the layer.

Information

The Information tab provides information about the selected layer. It gives the name of the table and its path, the type of table, e.g., MapInfo table, and its coordinate system.

Theme Layer Settings

When you click on a theme layer, the following tabs are available in Layer Control.

Visibility

When a theme layer is selected, the Visibility tab options control the display of the selected theme. Select the **Visible** check box to display the theme layer; clear the check box to turn off the theme display.

Select the **Display Within Range** check box to specify either a zoom range or scale range in which the theme appears. If you select a zoom range, specify the minimum and maximum zoom distance. Your theme appears within this range. If you select a scale range, specify the closest and farthest scale. Your theme appears within this scale range.

Theme

The Theme tab indicates the type of thematic map and the expression used to obtain the values. The Theme tab also enables you to modify your thematic map. Click **Modify Theme** to change the styles or legend.

Label Layer Settings

When you click on a label layer in the layer tree, the Visibility tab is available at the bottom of the Layer Control. When you expand the label layer to see the label sources, additional tabs display that control the appearance and content of labels in label sources: AutoLabel, Style, Text, Position, and Rules.

Visibility

When a label layer is selected, the options in the Visibility tab control the display of labels. Select the **Visible** check box to display labels; clear the check box to turn off label display.

Select the **Display Within Range** check box to specify either a zoom range or scale range in which the labels display. If you select a zoom range, specify the minimum and maximum zoom distance. Your labels display within this range. When you use a zoom range, the maximum value is exclusive—the layer is only visible if the map's zoom is less than the maximum value. So, if you set the maximum zoom value to 5000 miles, and then you zoom the map to exactly 5000 miles, the layer disappears.

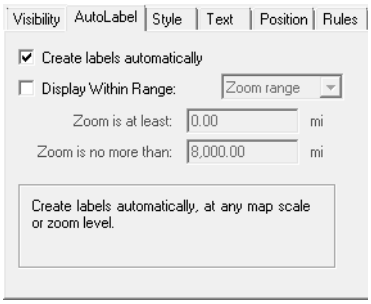
If you select a scale range, specify the closest and farthest scale. Your labels display within this scale range.

Click **Clear Label Modifications** to return the labels to their default state. This button removes individual labels that were manually added with the Label tool and restores labels to their original position.

AutoLabel

The AutoLabel tab enables you to create and manage the display of autolabels. Select the **Create labels automatically** check box to generate autolabels for your map. Select the **Display Within Range** check box to specify either a zoom range or scale range in

which the autolabels display. If you select a zoom range, specify the minimum and maximum zoom distance. If you select a scale range, specify the closest and farthest scale. Your autolabels display within this scale range.



Style

The Styles tab controls the style of label text and label lines. For label text, use the Text style box to access the Text Style dialog box. You can specify the font, color, background, and other text effects for the labels. For label lines, use the Line style box to access the Line Style dialog box, where you set the style of the label lines. In the Label Lines group, choose whether you want no label lines, simple lines, or lines with an arrow.

ⓘ Label lines are not supported for curved labels. MapXtreme can load label lines created in MapInfo Professional or if label lines are created using a customized label tool. See Curved Labels.

Text

The Text tab enables you to specify an expression that produces the label text from a column or derived information in the table.

Position

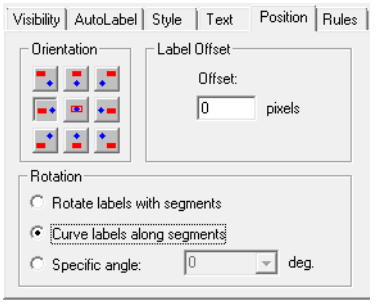
Use the settings in the Position tab to set the orientation, offset, and rotation of the labels.

The label's orientation is the label's position relative to its anchor point. Click one of the buttons to select an orientation.

Label offset is how far away a label is from its anchor point in pixels.

The label rotation is the angle at which the label is drawn. There are three Rotation settings:

- **Rotate labels with segments**–Select this option if you are labeling line features such as highways, and you want each label to be drawn at an angle that will make the label run parallel to the nearest segment of the highway.
- **Curve labels along segments**–Select this option if you are labeling line features such as highways, and you want the label text to follow the shape of the highways (that is, you want curved labels). Turning anti-aliasing on via the Style tab in Layer Control will improve the look of the resulting curved labels. For more information, see Curved Labels.
- **Specific angle**–Specify an angle in degrees, such as zero degrees to make all labels horizontal.



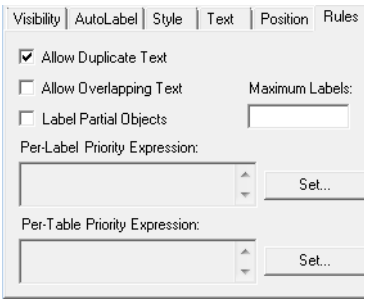
Rules

The Rules tab enables you to set certain conditions for displaying labels on your map:

Allow Duplicate Text Select the Allow Duplicate Text check box to allow duplicate labels for different objects to display, e.g., Portland, OR and Portland, ME. This option is also used with street maps to label street segments individually.

Allow Overlapping Text Select the Allow Overlapping Text check box to allow labels to be drawn on top of each other. Some labels do not display because they overlap labels that have been given higher priority on the map.

Label Partial Objects Select the Label Partial Objects check box to label polylines and objects whose centroids are not visible in the Map window.



Maximum Labels Specify the maximum number of labels you want displayed. For example, sort your data so that the most prominent (largest population, highest revenue, highest growth rate,etc) is at the top of your table. To display only the top 100 of these records, put 100 in the Maximum labels box. Records that are below the 100-record cutoff will not display labels, thus the 100 labels that do display will make a bigger impact on your map.

Per-Label Priority Expression This expression field is optional. If you leave this expression field blank, features within a single label source are labeled in an unpredictable manner. For example, you might find that some small cities are labeled, while some major cities are not labeled because there is not enough room. If you specify an expression (which must be numeric), then the expression will be calculated for each feature on the visible portion of the map, and features that have a larger value will be given a higher labeling priority. To specify an expression, click the **Set** button.

For example, suppose you are configuring the labels for the WorldCapitals layer, which contains point features that represent cities. You probably want the cities with the largest population to have the highest labeling priority. In this case, you would specify a labeling expression such as:

```
cap_pop
```

The cap_pop column represents the population of each capital city. When you specify a Per-Label Priority Expression of cap_pop, you are specifying that the cities with the largest population should have the highest labeling priority. As a result, the most populous cities will be labeled first, while other cities will be labeled only if there is enough room left over.

Per-Table Priority Expression This expression field is optional. A label layer can contain multiple label sources; for example, you might have one label source representing a set of labels for World Countries, and another label source representing a set of labels for World Capitals (cities). By default, the label source at the top of the list has the highest priority. If you want to assign a higher priority to a label source, you can either move that label source up (in the layer control's list of label sources), or you can specify a Per-Table Priority Expression for each label source.

For example, if you give the World Countries label source a per-table priority expression of 10, and give the World Capitals label source a per-table priority expression of 5, country labels will have priority over capital city labels.

Group Layer Settings

Highlight the Group Layer node in Layer Control and a Visibility tab displays. Here you control visibility settings that apply to all layers in the group.

To apply different visibility settings to each layer in the group, highlight the layer that is shown under the Group Layer and make the changes in the Visibility tab that displays.

Style Override Settings

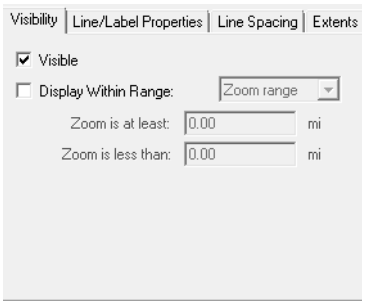
When you highlight a style override layer in Layer Control, the Visibility tab displays. Here you control whether the style override is visible and at what zoom range. The Style buttons are accessible here for you to set the area, line, symbol and text styles for the layer.

Choose from two drawing modes: **Normal** and **Superimpose**. Normal drawing mode draws the layer using the style override. Superimpose draws the style override on top of the layer.

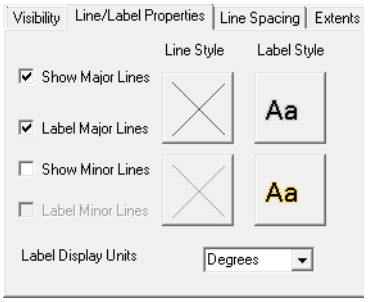
Graticule Layer Settings

When you click on a graticule layer, the following tabs are available at the bottom of the Layer Control.

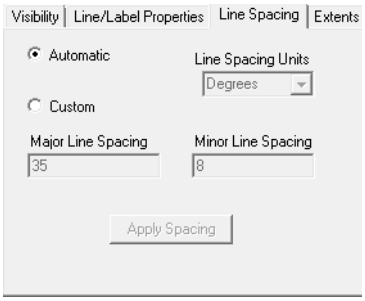
Visibility



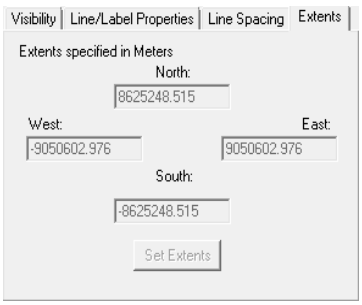
Line/Label Properties



Line Spacing



Extents



After making any changes in this tab, click **Set Extents** to activate the changes.

For more information on Graticules, see Graticule Layers.

Export/Import Theme and Style

This feature allows you to export themes or styles applied on a given layer to an XML file. Later on, this exported theme and style XML can be imported on any similar layer. This is particularly useful when you create and apply complex display setting on a layer and you want to preserve it for reuse.

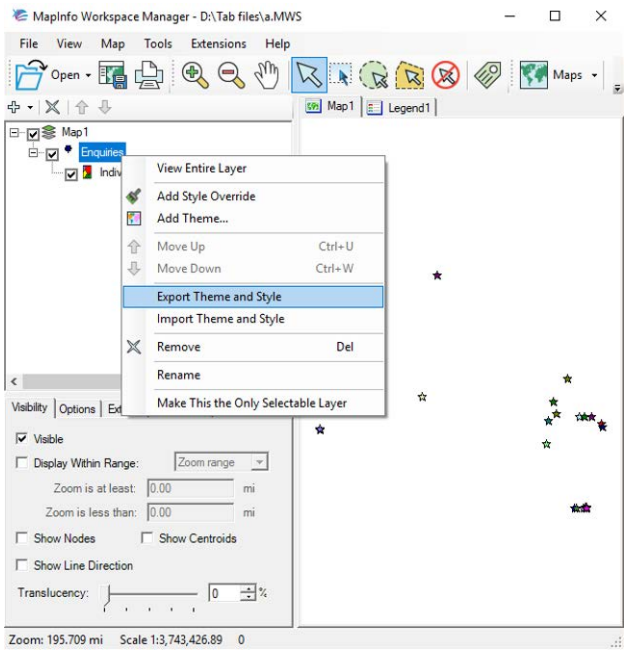
There are two ways to use this feature:

- Using Layer Control
- Using MapXtreme API interface

Using Layer Control

1. Once you are ready to export the theme and style, display the context menu by right-clicking on the map or layer.

2. From the context menu, choose **Export Theme and Style**.



3. Enter a name for your xml file.

This exported theme XML can be imported on any similar layer.

Using MapXtreme API interface

You can also use this feature programmatically. Refer to Export/Import Theme and Style.

ⓘ If there are two or more layers in the map, which belongs to same table (.TAB file) and you apply any of Pie, Bar or Graduated symbol theme on any of these layers (for example, layer1), then theme/style export operation on any of these layer (for example, layer2) will contain Pie, Bar or Graduated symbol theme as well. The reason behind this behavior is that Pie, Bar and Graduated symbol themes are connected to underlying table (.TAB) instead of layers.

Using Workspace Manager Features

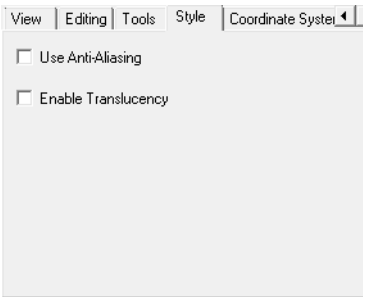
This section shows you how to apply some of Workspace Manager's features to enhance the cartographic quality of your maps. The following topics are covered:

- Enhanced Rendering with GDI+ Translucency and Anti-Aliasing
- Creating Translucent Effects
- Curved Labels
- Graticule Layers

Enhanced Rendering with GDI+ Translucency and Anti-Aliasing

Microsoft Windows GDI+ is the portion of the Windows XP operating system or Windows Server 2003 operating system that provides two-dimensional vector graphics, imaging, and typography. GDI+ improves on Windows Graphics Device Interface (GDI, the graphics device interface included with earlier versions of Windows) by adding new features and optimizing existing features. GDI+ rendering in MapXtreme allows you to create translucent labels, themes, and layers, as well as apply anti-aliasing that will smooth the jagged edges of lines, curves, and region borders when representing a high-definition rendition at a lower resolution.

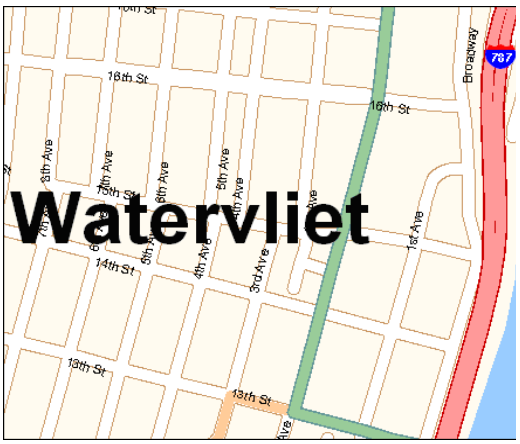
You can enable GDI+ rendering in the API through two new properties—EnableTranslucency and SmoothingMode— in the MapInfo.Mapping.DrawingAttributes and MapInfo.Mapping.LegendDrawing Attributes classes. You can also enable these properties using the Style tab in Workspace Manager's Layer Control.



ⓘ Anti-aliasing can only be used when translucency is enabled. This is enforced both by the user interface via the Workspace Manager and programmatically. When you select Use Anti-Aliasing, Enable Translucency is also selected automatically. Whenever Enable Translucency is deselected, Use Anti-Aliasing is automatically deselected.

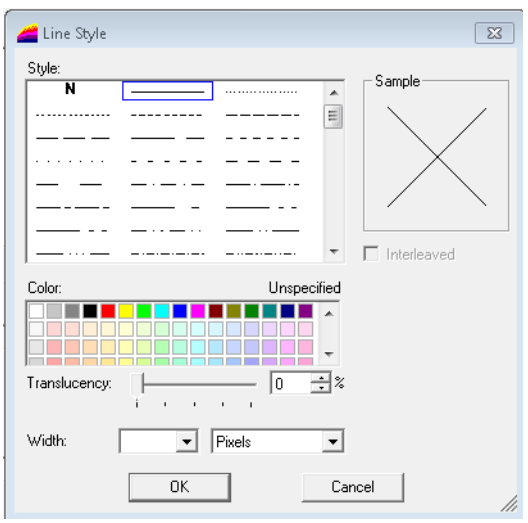
The following maps demonstrate label translucency and anti-aliasing effects on roads and highways. Notice the smoothness of the region borders and polylines when anti-aliasing is enabled.

Before GDI+ Enhanced Rendering:



After GDI+ Enhanced Rendering:

A translucency trackbar (TrackBarValuePicker control) in all style dialogs (Line Style, Area Style, Symbol Style, Text Style, etc.) enables you to pick a percent value between 0-100 using the sliding trackbar or the numeric selection box. However, this trackbar only works when translucency is enabled.



To programmatically change a color's translucency, use the System.Drawing.Color.FromArgb() method. This allows you to specify an alpha value for the desired color (in GDI+, the alpha channel is the portion of pixel color data reserved for transparency information). For more information, see the .NET documentation..

Translucency in all colors is supported and tools will work properly when translucency is enabled or disabled. The Enable Translucency option has no effect on the display or export of translucent raster images, although it must be enabled to print them.

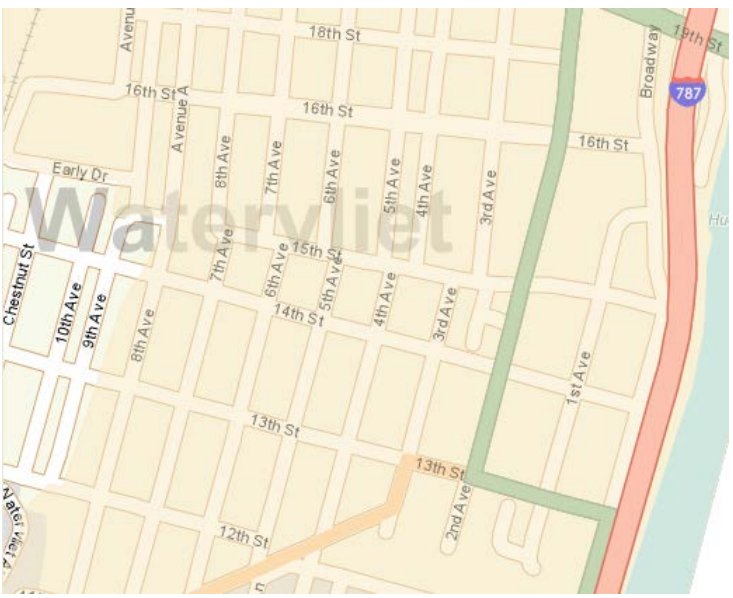
ⓘ Rendering higher quality maps by enabling translucency and anti-aliasing, particularly in a map with three or more transparent layers, will often result in a slower rendering speed.

Creating Translucent Effects

Applying translucent effects to the colors of map features and labels or to the color of spread styles of a theme layer enables you to see other features of the map through the color.

How to Apply Translucent Effects to a Map

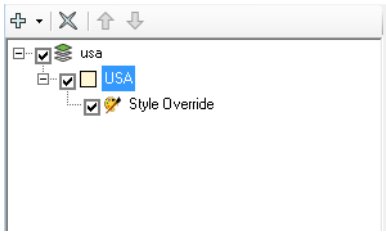
The following map contains a translucent city boundary region. The partial overlap of this region over the “Watervliet” label shows the difference between the uncovered part of the label (the first two letters of “Watervliet”) and the covered part of the label. The label itself is also translucent over the rest of the map.



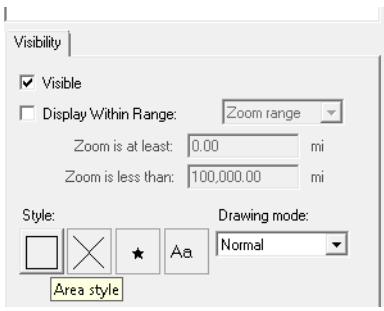
To add translucent effects to a layer, use a style override to change the original style of the layer.

ⓘ When you programmatically superimpose an override style modifier (OverrideType = AddNew), the first style drawn is the feature's style. Since the superimposed style is translucent, the features show through it.

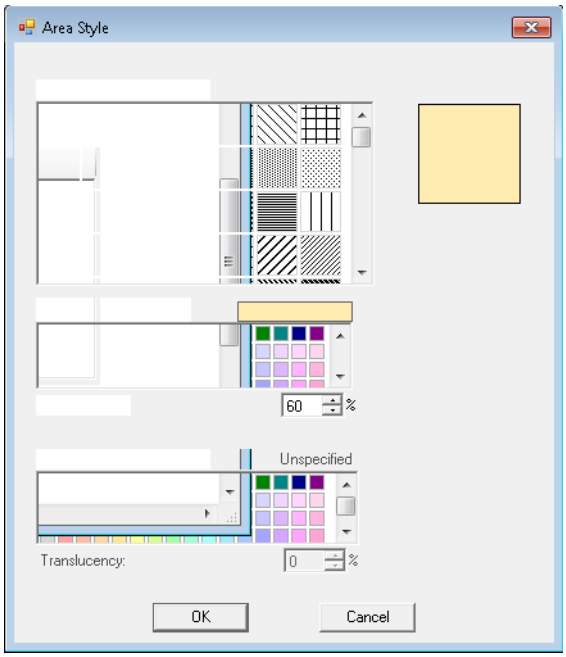
1. In Workspace Manager's Layer Control, select the "root" node in the layer control (this corresponds to the map). In the Style tab select **Enable Translucency**.
2. Select the layer you want to add translucency to, move it to the top of the list, then add a style override.



3. On the Visibility tab for the style override, click the style button that is specific to the objects in that layer. For this example click the **Area style** button.



The Area Style dialog appears.



4. Use the translucency trackbar to select the level of translucency you want to apply. Each color has its own translucency level.

Alternatively, the newly added Opacity property or the translucency trackbar added in the LayerControl's View tab can be used.

How to Apply Translucent Effects to Labels

The previous example also showed a translucent label similar to a watermark. If you turn off visibility on the translucent city boundary region, the label looks as follows:

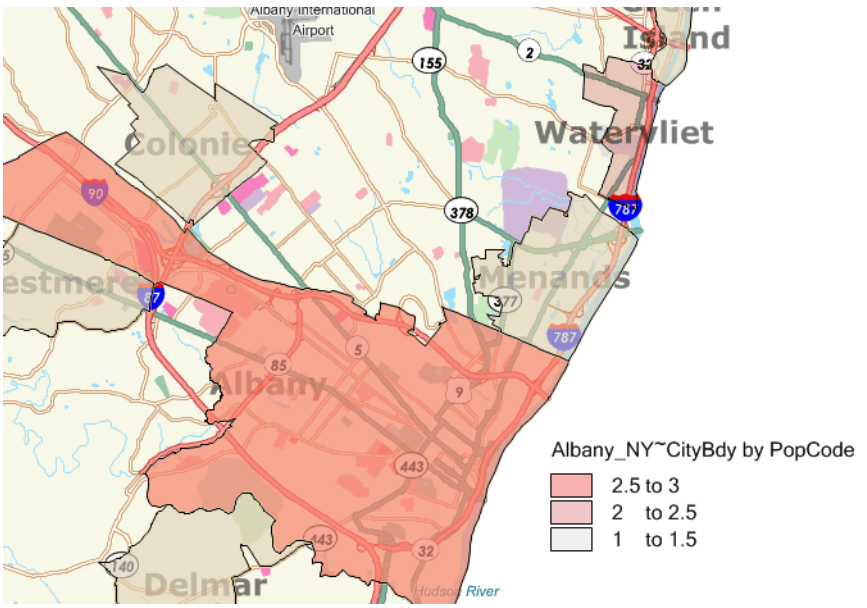
If you want to add a translucent label over your map as shown by this example:

1. In Workspace Manager's Layer Control, select the "root" node in the layer control (this corresponds to the map). In the Style tab select **Enable Translucency**.
2. Select the label source in the layer control and go to the Style tab.

3. Click the **Text style** button to display the Text Style dialog.
4. Use the translucency trackbar to select how translucent you want your label color to be. The above example shows text at 80% foreground translucency.

How to Apply Translucent Effects to Themes

You can apply translucent effects to thematic maps. The following example shows a ranged theme applied to a city boundary layer. The layer is positioned on top of the rest of the layers.



If translucency is enabled, you can select a translucent value for the start and end theme ranges (bins), and automatically spread the color. This will also automatically spread the translucent value between the start and end theme bins. For example, the theme in the above map has a translucency value of 75% for the start bin (gray) and 50% for the end bin (red). Since this theme has 3 bins, the middle bin is automatically given a translucency value of 63% (50 through 75 spread equally).

Curved Labels

Curved labels are labels that follow the curve of a line. They enhance the appearance of map features that are made up of lines, such as streets and rivers.

Curved labels are generated in the Workspace Manager by selecting the label layer, then selecting **Curve labels along segments** in the Position tab. For information on rendering curved labels via the API, see the ILayout Interface section of the MapXtreme Developer Reference.

MapXtreme attempts to create a curved label for every arc and polyline record in a map, just as it does for non-curved labels. For example, in street maps, the street can be made of several polylines or one long polyline. The length and number of the polylines, the rules that govern whether a curved label can be created, and the labeling options you choose, all affect which curved labels are created and where they display.

Some polyline and arc segments in your layer data may not contain label name entries. When this occurs, MapXtreme cannot display labels for that segment.

Several rules determine whether MapXtreme can display a curved label:

- MapXtreme can only draw curved labels using TrueType fonts. If you select a non-TrueType font, a comparable TrueType font is substituted and the label you chose may display differently than expected. Also, if you change a horizontal label using a non-TrueType to a curved label, the new label may display differently due to the font substitution.
- Part of the label string must fit along the arc or polyline that it is labeling. If it cannot fit, MapXtreme determines that the label is too long and discards it.
- MapXtreme cannot draw curved labels for polylines that are very jagged, however, it depends on the curvature of the line.
- Although a street segment appears to be straight, the label may be curved. This happens because the polyline data for the street segment contains a curve that is not visible at the current zoom level. The label is following the curve of the street even though the curve isn't visible. If you zoom in on the map to a close enough distance, you will be able to see the curve in the street.
- Labels that curve onto themselves are discarded and do not display.
- Curved labels follow the same rules for overlap detection, duplicate text, and partial segment labeling as non-curved labels. Each of these rules affect how and when the labels are displayed.
- You can create curved labels with the Label tool at any point along an arc or polyline.
- You cannot drag curved labels as you can other labels; however, you can reposition them with the Label tool.
- The Label Lines controls are disabled for curved labels.
- Curved and non-curved labels persist for layers in the workspace.
- You cannot underline curved labels.

- Curved labels are always drawn smoothly whether or not translucency and anti-aliasing are enabled.

Bi-Directional String Support in Curved Labels

Curved labels are supported in languages that read right to left. Support was added for the Uniscribe library that ships with Windows, which handles contextual glyph shaping and reordering bi-directional scripts such as Hebrew and Arabic. Its methods can analyze and break apart strings using complex scripts into separate runs which can then be reordered for display.

Curved Labels Created in MapInfo Professional

Curved label display settings available in MapInfo Professional can be saved to an .MWS workspace and loaded into MapXtreme. Changes to the position of a curved label made in MapInfo Professional can also be saved to an .MWS workspace and loaded into MapXtreme. The following describes the display settings and position modifications that MapXtreme can load:

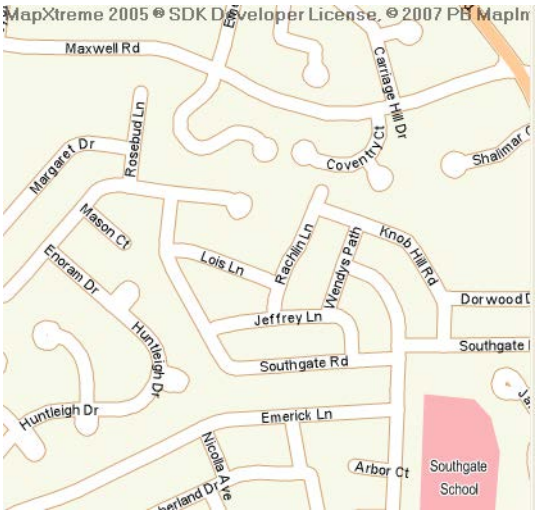
- Although MapXtreme does not support creating label lines for curved labels in Workspace Manager, it can load label lines created in MapInfo Professional or if label lines are created using a customized label tool.
- MapXtreme can load callout lines for labels that contain a custom (user-defined) location for the line endpoint. For example, if you move the endpoint of a label in MapInfo Professional and save it to a workspace (.MWS), the endpoint is positioned as expected when the workspace is opened in MapXtreme. MapXtreme loads and saves custom line ends but does not support creating them.
- Curved labels created in MapInfo Professional and saved to an MWS display in the same location in MapXtreme.
- Labels that have been dragged to a new location in MapInfo Professional can be displayed as curved labels in MapXtreme. When the label changes are saved to an .MWS workspace and the workspace is opened in MapXtreme, the labels will display as curved labels when the curved label option is turned on. Note, however, if the label in MapInfo Professional was dragged and then rotated prior to being saved to the workspace, MapXtreme will not curve the label. This is because MapInfo Professional has modified the angle of the label, which MapXtreme honors over the curved label setting.

Creating Curved Labels

To position labels along a curve:

1. In Workspace Manager, open the map you want to change the labels for.
2. Highlight the layer source that contains the labels you want to change.

3. Verify that the labels are visible.
4. Click the Position tab. In the Rotation section, select **Curve labels along segments** to display the labels along the curve of the line.



5. If necessary, use the Orientation buttons on the Position tab to set the label's position relative to its anchor point. When you select:

   Left, the curved labels are left-justified starting at the beginning of the arc/polyline

   Center, the curved labels are centered on the midpoint of the arc/polyline

   Right, the curved labels are right-justified at the end of the arc/polyline

---

ⓘ The length of the polyline(s) affect how the label is positioned. The longer the polyline(s), the more predictably the labels display.

Repositioning Curved Labels

You can use the Label tool on the Main toolbar to reposition curved labels. Make sure you have already selected the **Curve labels along segments** option on the Position tab for the label source you want to change.

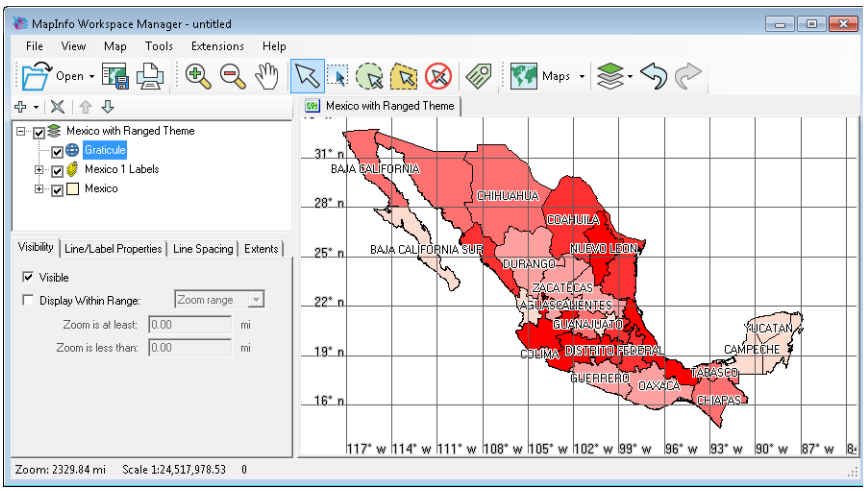
1. Highlight the layer you want to move the labels for in the list.
2. In the Options tab, select the **Selectable** check box.
3. Highlight the label source where you want to move curved labels.
4. Click  (the Label tool) on the Main toolbar.

5. Click the line on which you want to reposition the label.
6. Click the new location for the label until the label is positioned where you want it.

> ⓘ If the segment you select does not have a label name associated with it in the data, no label is displayed.

Graticule Layers

Graticules are grids (lines of latitude and longitude) that overlay the map, spaced at a regular distance (for example, every five degrees, every fifteen degrees). They are used to establish a frame of reference.

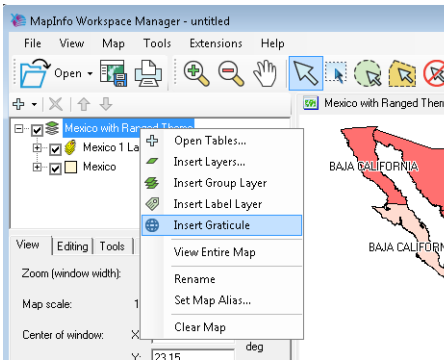


In many ways, graticules behave just like other layers. However, a graticule layer differs from other layer types in the following ways:

- Graticule layers are not editable like other types of layers. That is, you cannot add new features to them.
- Although you can configure graticule label style and position, you cannot create a label layer on a graticule layer.
- You cannot create a thematic layer on a graticule layer.

Adding Graticule Layers

You can add graticules directly from the Layer Control (right-click on the map node to access the context menu):



By default, the graticule spacing and extents are determined based on the zoom and size of the map window. So when you add a graticule layer to a map, graticule lines appear regardless of the zoom level.

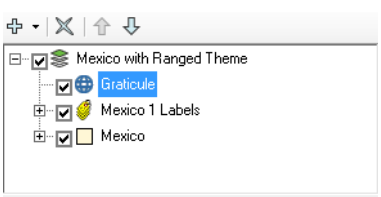
Managing Graticule Layers

After you have added a graticule layer, you can manage and customize the layer from the Layer Control in a number of ways.

- Control graticule layer visibility and zoom/scale settings
- Show and configure spacing of major and minor graticule lines
- Control visibility and style of graticule labels and lines
- Specify the extents (in degrees, feet, or meters) of the graticule (east, west, north, south).
- Move the layers up and down in the Layer Control order
- Remove the graticule layer (the context menu is the only way to remove a graticule layer)
- Create multiple graticule layers and Group Layers

See Graticule Layers for more information.

Graticule layers are indicated by a graticule icon, as shown below:



28

28 –Using the GeoDictionary Manager

The GeoDictionary Manager is a utility that comes with MapXtreme to aid in your Mapping applications. This chapter deals with how to use this tool.

In this chapter:

Using the GeoDictionary Manager

The GeoDictionary Manager application is designed to support the manipulation of a GeoDictionary file. The GeoDictionary is an XML file containing registration information about the MapInfo tables that can be matched by your application during automatic databinding. Only MapInfo tables that can or will be matched against should be registered in the GeoDictionary.

There is no need to register every .tab file that an application uses in the GeoDictionary, and in fact there is some overhead in having unnecessary files registered. You only need to register those tables against which you would like to match.

Changes in the GeoDictionary Manager

The GeoDictionary Manager for MapXtreme is, in appearance, very similar to the utility that was included in MapX and previous versions of MapXtreme. The major difference is that now the underlying file (*.dct) is an XML file and completely editable using a text or XML editor.

The GeoDictionary Manager's User Interface

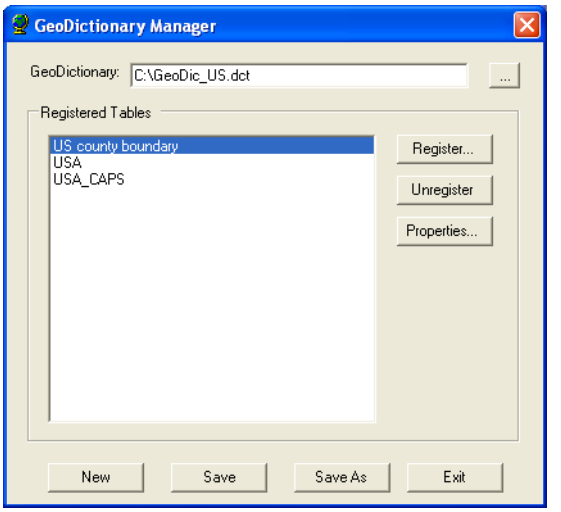
This section describes the user interface for the GeoDictionary Manager.

Run GeoDictionary Manager

To run the GeoDictionary Manager when you want to manually register layers:

- Start -> All Programs -> MapInfo -> MapXtreme -> GeoDictionaryManager.

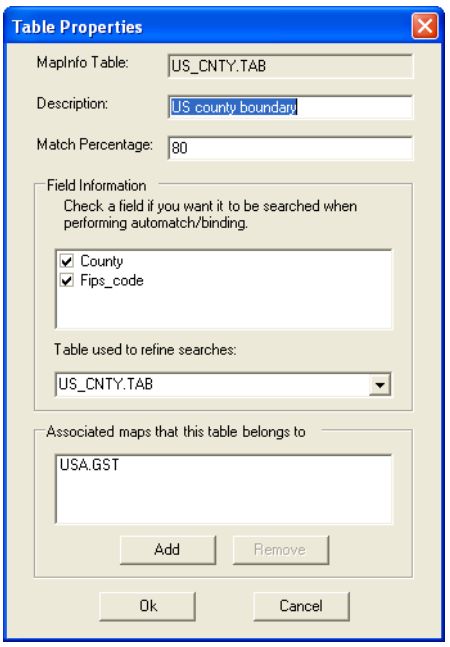
  The GeoDictionary Manager is displayed. The file last opened is reloaded upon start-up.



Part	Description
GeoDictionary	The GeoDictionary edit box contains the full path to the GeoDictionary that is currently being managed. The button (to the right of the GeoDictionary edit box) allows the user to browse for another GeoDictionary to manage. Both binary and XML GeoDictionary files can be opened, but only the XML format is saved.
Registered tables	The Registered Tables list box contains a list of the friendly names for all tables registered in the GeoDictionary. Highlight a particular table to either unregister it or modify its properties. Double-clicking a table brings up the Properties dialog box to edit those properties.
Register	The Register button brings up the common File Open dialog box, with the Files of Type combo box set to “MapInfo Tables (*.tab)”. After the table has been chosen from the file picker, the Table Properties dialog for that table is displayed. If you choose multiple files in the File Open dialog box, a Properties dialog box is opened for each table added.
Unregister	Pressing this button removes the selected table from the GeoDictionary. You can also select multiple tables (by shift-clicking) for unregistering. The Unregister button does not remove the files from the disk.

Part	Description
Properties	The Properties button brings up the Table Properties dialog box for the selected table. See the Table Properties dialog box below.
New	The New button clears the fields on the screen allowing the creation of a new GeoDictionary file. When you choose to close this GeoDictionary file you are prompted to save your changes.
Save	The Save button saves the current file as XML, regardless of the format of the file when it was opened.
Save As	The Save As button allows you to save the current file to a new name or location.
Exit	The Exit button closes the application. If you have modified the GeoDictionary file you are prompted to save your changes.

Set the matching table properties for a given table using the Properties dialog box. To access the Properties Dialog box, click the Properties button, or double-click on any table name in the Registered Tables list.



Part	Description
MapInfo Table	Read-only edit box containing the file name of the MapInfo table if it is located in the same directory of the GeoDictionary, or the relative pathname to the file if it is not. If the file is located on a different drive or volume, the full path is displayed.
Description	This field provides a mechanism for changing the friendly name for the table. This control is defaulted to the Description tag in the .TAB file, or the filename if the Description tag is not found, but can be changed by the user. Note that changes to the description in the GeoDictionary Manager are only be stored in the GeoDictionary and is not reflected in the table itself. This allows the GeoDictionary Manager to easily work with read-only data, e.g., data on CD-ROMs.
Match Percentage	This field is initially populated with the default value for the GeoDictionary and can be altered for specific tables by changing this value. Values must be between 1 and 100.
Field Information	This list box contains a list of all the indexed columns in the table. If the box for a given column is checked, that field is searched during the matching process.
Table used to refine searches	This field allows the user to set a refining table to determine exact match for data that is not unique in a particular index. Some tables, e.g., US Counties, contain indexed columns that are not unique. In that situation, a refining table is necessary to determine an exact match for data. If the table has non-unique indexed columns, use this field to specify a table to use to match against to find unique entries.
Associated maps that this table belongs to	This list box shows the particular Workspaces, GeoSets, or other files of which this particular table is a part.

Part	Description
Add	This displays a common file picker to allow you to choose GeoSets, Workspaces, or other files to associate with the particular Tab file. Selecting one or more files adds them to the list of associated maps. You can select multiple files by holding down the shift or control keys while clicking.
Remove	Deletes the selected geoset, workspace, or other filename from the list of associated maps.

The GeoDictionary File

The GeoDictionary file (*.dct) can be manipulated manually if you choose to skip using the GeoDictionary Manager. If you understand and can write XML easily then this is a viable alternative to using the GeoDictionary Manager. The .dct file is straight XML.

Sample .dct file

The following is a sample GeoDictionary file, GeoDic_US.dct. This file is a very simple GeoDictionary provided as an example of the structure.

```
<?xml version="1.0" encoding="Windows-1252" standalone="yes"?>
<!--GeoDictionary file-->
<GeoDictionary>
   <DefaultMatchThreshold>80</DefaultMatchThreshold>
   <MatchTables>
      <MatchTable>
         <TablePath>US_CNTY.TAB</TablePath>
         <TableDescription>US county boundary</TableDescription>
         <RefineTableName>USA.TAB</RefineTableName>
         <MatchThreshold>90</MatchThreshold>
         <AssociatedMaps>
            <AssociatedMap>USA.GST</AssociatedMap>
         </AssociatedMaps>
         <MatchFields>
            <MatchField>
               <FieldName>County</FieldName>
            </MatchField>
            <MatchField>
               <FieldName>Fips_code</FieldName>
            </MatchField>
         </MatchFields>
      </MatchTable>
```

```
   </MatchTables>
</GeoDictionary>
```

The elements in the structure correspond to individual fields and controls in the GeoDictionary Manager as defined in the following table:

XML Element	Table Properties Dialog Box Field	Description
GeoDictionary	none	The root element of the file
DefaultMatchThreshold	none	The DefaultMatchThreshold is the threshold of matching when a specific MatchThreshold is not defined for a particular table. This value cannot be set using the GeoDictionary Manager.
MatchTables	none	The container elements for the registered MatchTables. There should be a single MatchTable element for each item in the Registered Table list.
MatchTable	MapInfo Table.	The filename for each table to be matched.
TablePath	MapInfo Table.	The relative path for each MatchTable.
TableDescription	Description field (also the name that appears in the Registered Tables list in the main dialog box.	The friendly name for the MatchTable.
RefineTableName	Table used to refine searches	The related table with which to refine matching.
MatchThreshold	Match Percentage	The match threshold for this MatchTable. If none is specified, the DefaultMatchThreshold is used.

XML Element	Table Properties Dialog Box Field	Description
AssociatedMaps, AssociatedMap	Associated maps to which this table belongs.	This list box shows the particular Workspaces and GeoSets of which this particular table is a part.
MatchFields	none	This element is a container for MatchField and FieldName elements.
MatchField, FieldName	Field Information list box.	Each item in the list box that is checked to be searched for automatch and binding.

29

29 – Location Intelligence API Integration in MapXtreme

This chapter contains information about accessing Precisely's Location Intelligence APIs using the application created in MapXtreme.

In this chapter

Overview

The Location Intelligence API enables developers to access Precisely's extensive collection of industry-trusted location data. The Location Intelligence APIs facilitate you in building innovative location-based applications for multiple platforms. The developers can use this locational data to develop apps for commercial and consumer markets. For more information visit, https://developer.precisely.com.

You can now make use of Location Intelligence API SDK (downloadable from Precisely's LI API portal and github) and MapXtreme together in the same application to build innovative location-based applications. We are providing a Sample application as well to demonstrate how these two SDKs can be used together to create location based application. The sample applications are placed at C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio2017.

MapXtreme LIAPI Integration

In order to support LIAPI integration, MapXtreme has provided the following capabilities:

Token Management

The LI API uses OAuth 2.0-based security for authentication and authorization to ensure security for any business-critical private information. MapXtreme supports the OAuth2 token for the client application. For this we have provided a property named "PBAPI.AccessToken" in the session class itself. Whenever the client application asks for token, it returns a new valid token. For example, "`String token = Session.Current.PBAPI.AccessToken`".

You can cache this token for all subsequent LI API calls until the token expires. Alternatively, you can ask for a new token from the session for every LIAPI call.

For generating token, MapXtreme needs user to specify the credentials. We have introduced two variables PBAPI_Key and PBAPI_Secret for users to specify their credentials. These variables are present in the registry key as well as in the environment variables. User can specify them in any or both of these places. MapXtreme first looks for the credentials into the registry key entry, and if it does not find it there then it will look into the environement variables.

Geometry Conversion

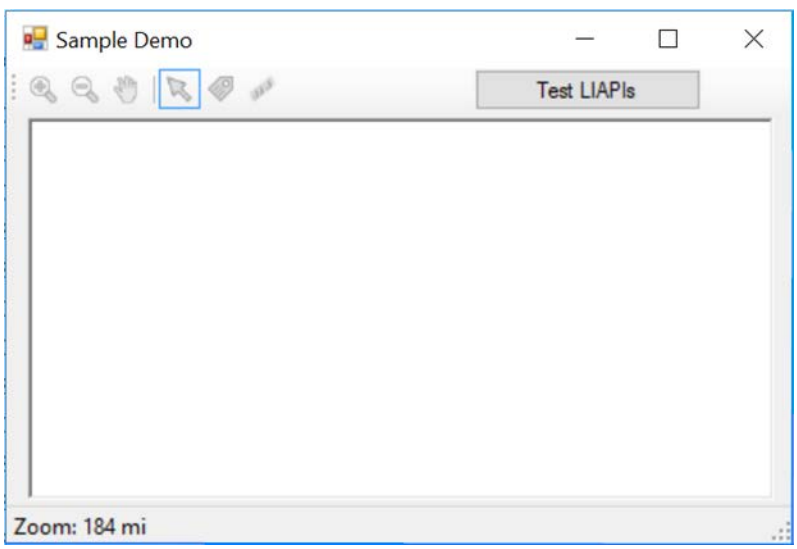
Location Intelligence API gives response in JSON format. Based on the type of LIAPI called, the response may or may not contain spatial data. If the response contains spatial information, MapXtreme converts the spatial data or geometries from the JSON response into MapInfo geometry. The client application can use this MapInfo geometry to show it on map.

The MapXtreme API takes geometry part of the JSON response as input and returns feature geometry as response as displayed in the image below.

```
String accessToken = Session.Current.PBAPI.AccessToken;
CoordSys csys = Session.Current.CoordSysFactory.CreateCoordSys("EPSG:4267");
pb.locationIntelligence.Client.Configuration config = new pb.locationIntelligence.Client.Configuration();
config.AccessToken = accessToken;
LIAPIGeocodeServiceApi instance = new LIAPIGeocodeServiceApi(config);
pb.locationIntelligence.Model.GeocodeServiceResponse response = instance.Geocode("premium", "USA", "walnut",
    settingsForm.geoCodeAddress);//arguments like datapackBundle, country, placeName, mainAddress etc.
FeatureGeometryFactory factory = new FeatureGeometryFactory(csys);
if (response.Candidates.Count > 0)
{
    foreach (Candidate c in response.Candidates)
    {
        FeatureGeometry ft = factory.FeatureGeometryFromGeoJson(c.Geometry.ToJson());
        Feature sf = new Feature(Session.Current.Catalog["temp"].TableInfo.Columns);
        sf.Geometry = ft;
        Session.Current.Catalog["temp"].InsertFeature(sf);
    }
    mapControl1.Map.SetView(Session.Current.Catalog["temp"]);
```

Sample Application

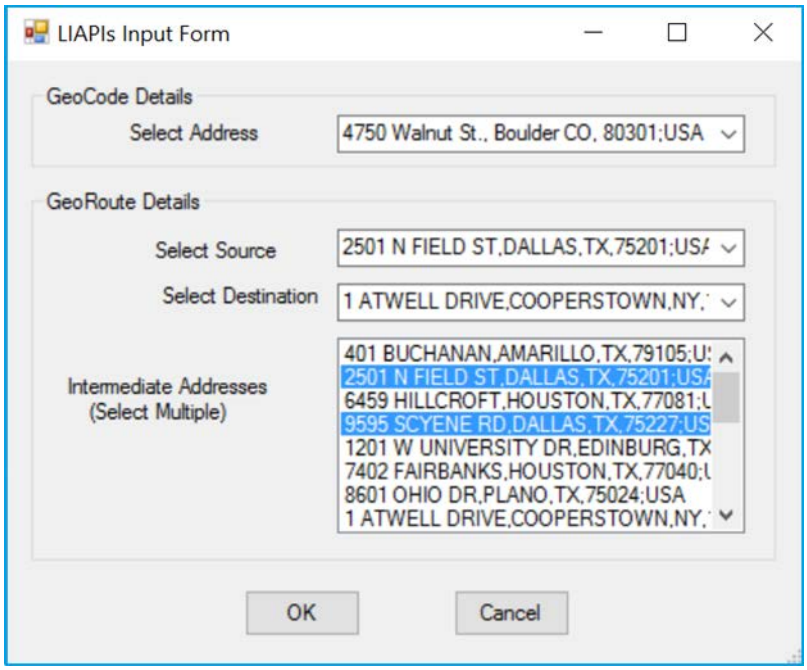
In the sample application, the first dialog looks like the image below. Click on “Test LIAPIs” button.



The **LIAPIs Input Form** dialog box opens. In this dialog you can test GeoCoding API or GeoRoute API or both at the same time.

ⓘ To use this sample you need to provide the credentials, i.e., PBAPI_Key and PBAPI_Secret.

For using GeoCoding API, select an address that you want to GeoCode. For GeoRoute API, select Source and Destination addresses, along with zero, one or multiple intermediate addresses and click OK. Refer to (image) below.



Now, the you can see the geocoded location of address on the map and a route from source to destination on the map.

A – How to Create and Deploy a MapXtreme Application

This appendix is designed as a tutorial to demonstrate the simplicity of creating, packaging, and deploying a MapXtreme application. At the end of this tutorial you will have gone through all the steps to successfully develop, package, and deploy a well-implemented mapping application.

This tutorial assumes that you have already successfully installed Visual Studio and are familiar with the use of that product. If you are not familiar with Visual Studio, there are many resources available on the Microsoft MSDN web site. Look at msdn.microsoft.com/developerscenter/. You should also have installed MapXtreme. The code fragments contained within this document were implemented using Visual Basic. For many of the sample applications, we also provide C# .NET samples.

In this appendix:

Customizing MapXtreme Samples

This appendix contains two tutorials:

- Building a Desktop Application
- Building a Web Application

These tutorials guide you through customizing MapXtreme desktop applications and Web Applications and preparing them for deployment.

Building a Desktop Application

We are going to base our desktop application on one of the thematic sample applications provided with MapXtreme. The desktop sample applications are already working mapping applications. With some simple modifications, you can customize a sample application for your own needs. You can use the modifications outlined in this tutorial as a model for customizing any desktop MapXtreme project. All the desktop sample applications provided with MapXtreme were created from the map application template provided in the product. See Creating Applications in Visual Studio for more information.

Running a Sample Application

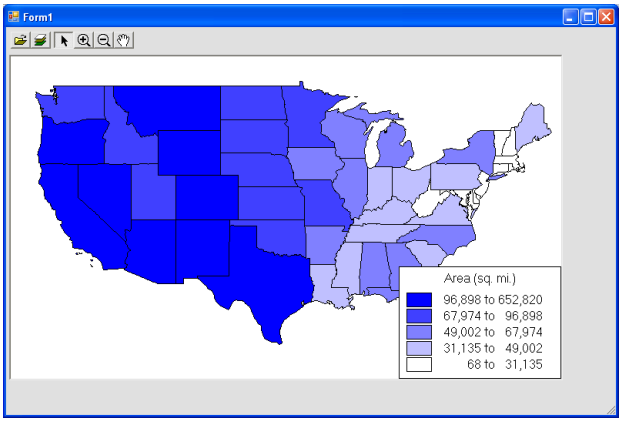
Let's start by running the ThemeLegend application to see what it looks like.

1. In Visual Studio, choose **File>Open > Project/Solution**, and open the project for the sample application called ThemeLegend. The default path to this project is:

   C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\VisualStudio\Desktop\Features\ThemeLegend\vb\ThemeLegendVB.vbproj

   (where *x* is the release number of the version of MapXtreme you are using).

2. Choose **Debug > Start Debugging**, or press F5 to run the application.



   The application displays a standard map of the United States with a theme based on the land area of each state. The larger states are shaded in blue and the smaller states are shaded in white. The legend shows the color assigned to each range of land area values.

   The tools provided with the application enable you to perform some basic functions. You can open tables, zoom in or out on a particular region, select regions on the map, move the map, and access Layer Control to apply specific settings to each layer. You might want to practice using the tools to get a feel for how the application works and what your users will experience.

3. When you are finished, close the active Form1 window to stop the application and return to Design mode.

Modifying Your Application

Next, we'll make some modifications to the ThemeLegend code to show how you can customize a sample application. We will change the following elements of the application:

- data source
- number of ranges (bins)
- range type
- range colors
- legend location
- toolbar

1. In Visual Studio, move your cursor to the Solution Explorer and right click on MapForm1.vb. Choose **View Code** to display the code page.

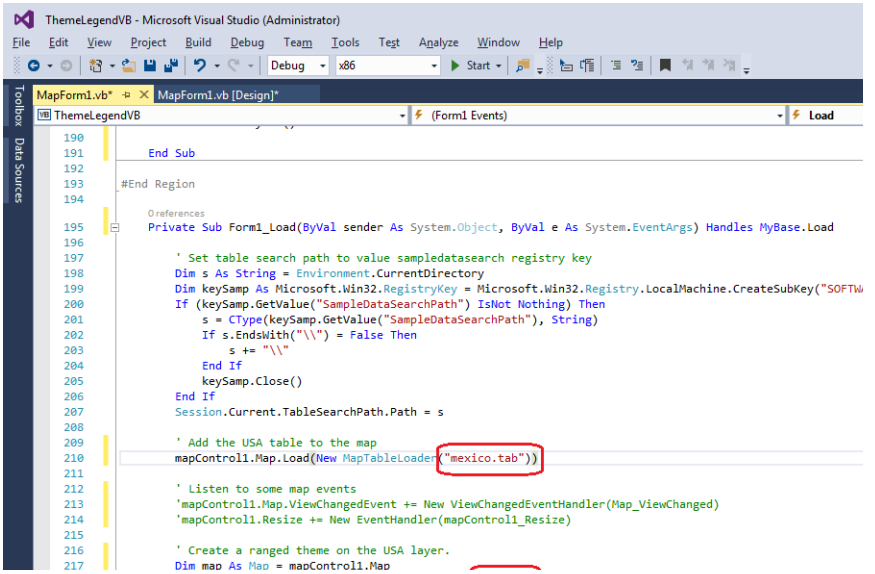
   If you have MapForm1.vb displayed in Design mode, you can right-click anywhere on the form, and choose View Code.

   ⓘ All sample applications discussed in this tutorial were taken from the MapXtreme distribution DVD. The line numbers referenced may not correspond exactly with the code in your installation.

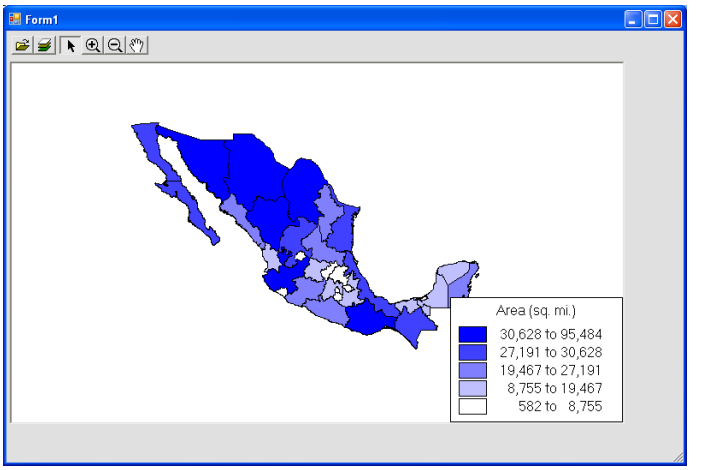
   ---

2. A new tab displays to show the application code. Scroll down the code page to familiarize yourself with the code. Notice that the majority of the action in the application takes place in the `Form1()` class. This is where we will make our changes.
3. Let's change the data source on which the theme is based. To do this, we'll need to change the name of mexicothe table that the application uses and specify a different FeatureLayer. Make the following changes:
   - On line 210 change the table from `usa.tab` to `mexico.tab`.

     This loads the Mexico map instead of the USA map.
   - On line 218, change the FeatureLayer from `usa` to `mexico`.
4. The image below highlights the code changes with red boxes.



5. Choose **Debug > Start Debugging** (or press F5) to run the application and see your changes. Close the Form1 window when you are finished.



The application creates the same type of theme using Mexico data. You can pass in any data source that fits this theme by changing the name of the table and the FeatureLayer.

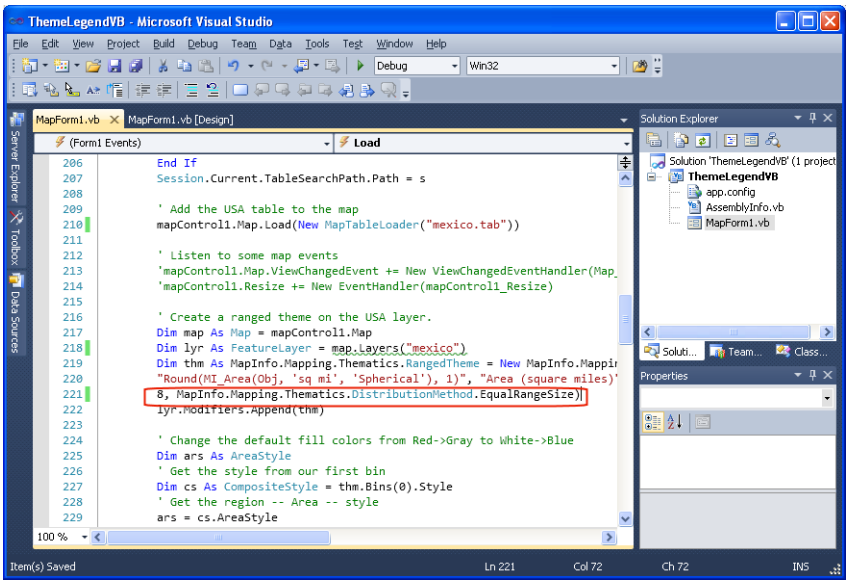
Next, let's make changes to the theme itself. By manipulating the parameters passed to the RangedTheme constructor we can change the way the theme is calculated and displayed.

6. Highlight the word RangedTheme in the code and press F1 (line 212). The help topic for RangedTheme displays on the screen. You can read this topic to see what each parameter does when it is passed to the constructor.

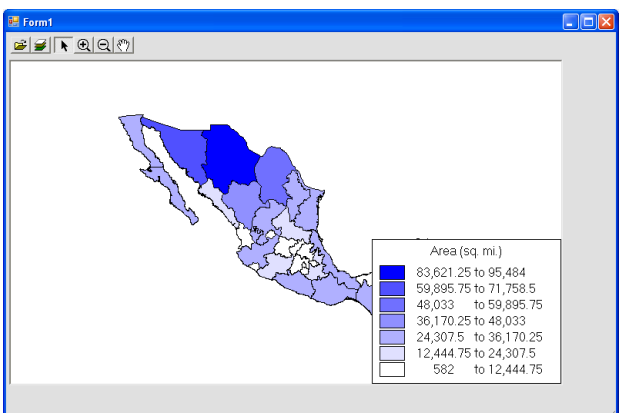
7. We'll increase the number of ranges, or bins, in the theme. On line 221 change the number of bins from `5` to `8`. (A bin is a range consisting of a maximum and minimum value and is used by themes to group together like values for the purpose of shading.)

8. Next, we'll change the distribution method of the ranges. The distribution method indicates how the ranges are calculated. On line 221, change `EqualCountPerRange` to `EqualRangeSize`. In the EqualRangeSize distribution method, each range has an equal number of values.

9. The image below highlights the code changes with a red box.



10. Choose **Debug > Start Debugging** (or press F5) to build and run the modified application. Notice that the number of theme bins has increased and the distribution method has changed. Close the Form1 window when you are finished.



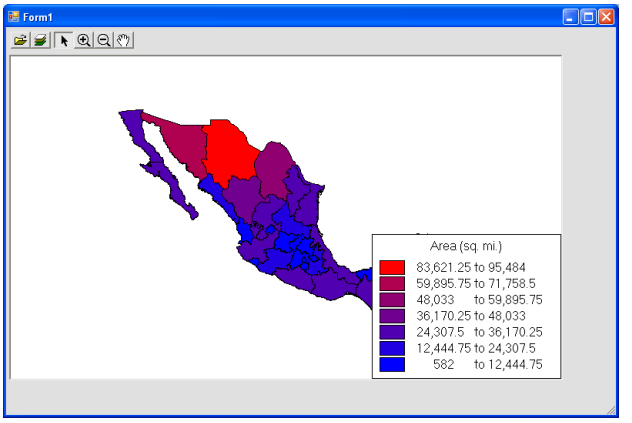
Next we'll change the color of the ranges. You have two options for how you would like to modify the colors. You can set a particular color for each bin, or you can set colors for the first and last bin. The shading of the bins in between will graduate from the first to the second color. Currently, the code specifies that the color ranges from white to blue. Let's change the colors to range from blue to red.

11. To change the color of the ranges, we'll need to edit the fill style colors. Make sure the code view is displayed and make the following changes:

- To change the color of the first bin, on line 231, change `WhiteFillStyle` to `BlueFillStyle`
- To change the color of the second bin, on line 241, change `BlueFillStyle` to `RedFillStyle`

Every bin in between will have a shade between blue and red.

12. Choose **Debug > Start Debugging** (or press F5) to build and run the modified application. The colors of the map have changed to reflect our new settings. Close the Form1 window when you are finished.



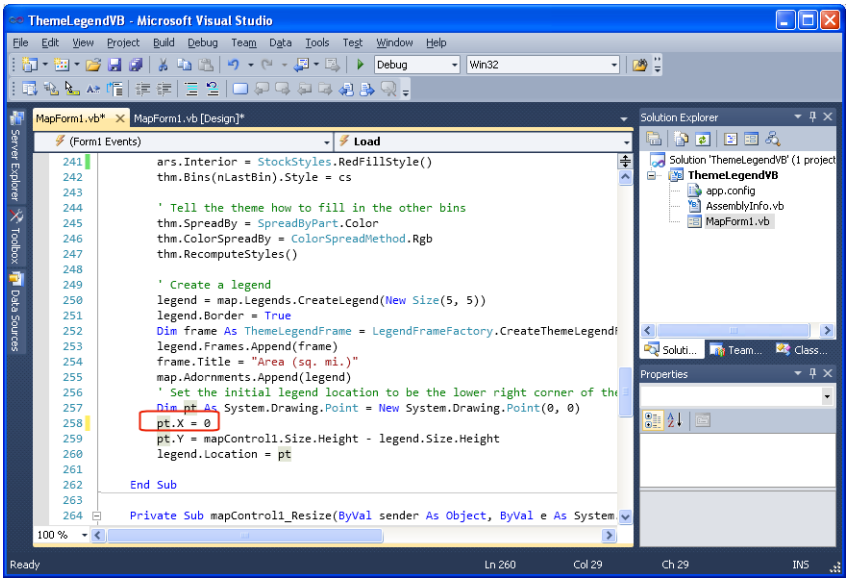
Because we have made a number of changes, the legend is now partially blocking the view of the map. We'll move the legend so that we can see all of the map. We could use the pan tool to move the map, but panning would not be a permanent change to the application. The new map position would be in effect only for as long as the application is running. Instead, we'll change the position of the legend programmatically.

13. Make sure the code page is displayed and make the following change to the legend location:

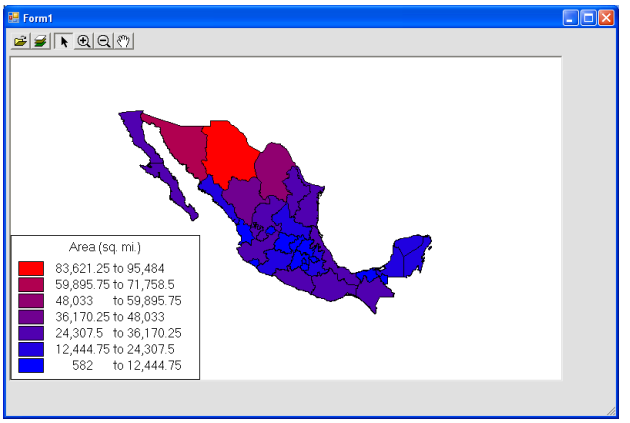
- On line 258, replace "`mapControl1.Size.Width - legend.Size.Width`" with "`0`"

  This changes the X coordinate of the legend location to be at the left side of the frame. We are leaving the Y coordinate as it is.

14. The image below highlights the code changes with a red box.



15. Choose **Debug > Start Debugging** (or press F5) to build and run your application. Now the legend is in its new location. Close the Form1 window when you are finished.

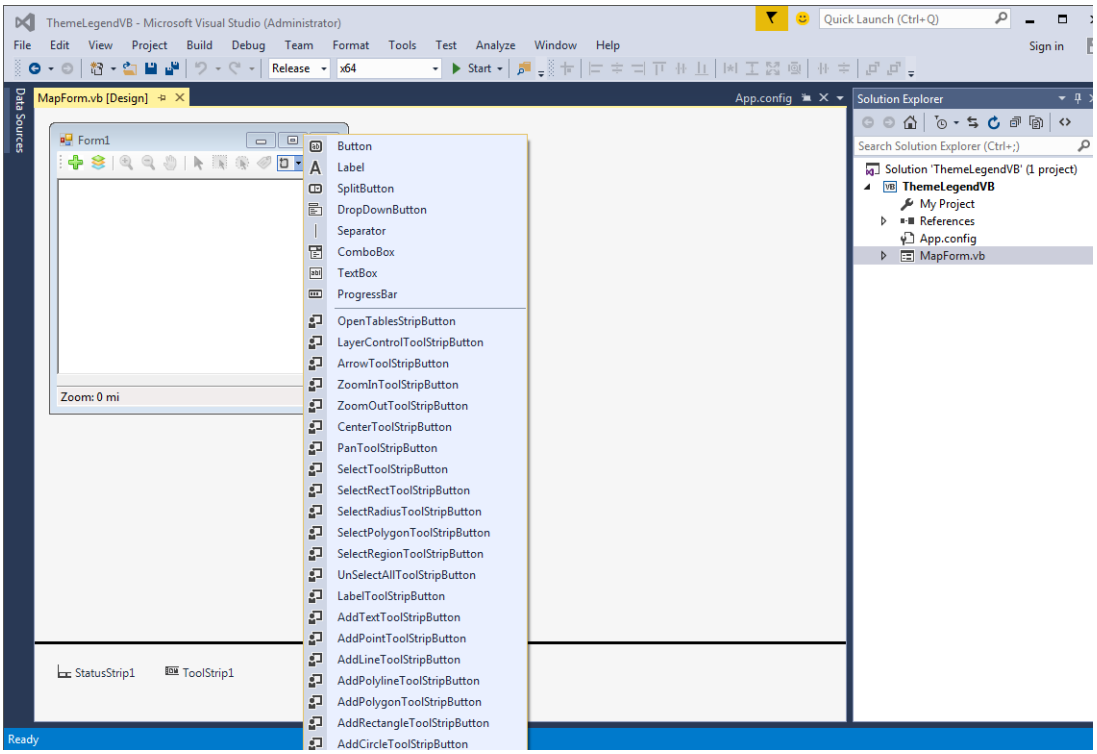


The last modification we'll make is to add a Label tool to the toolbar.

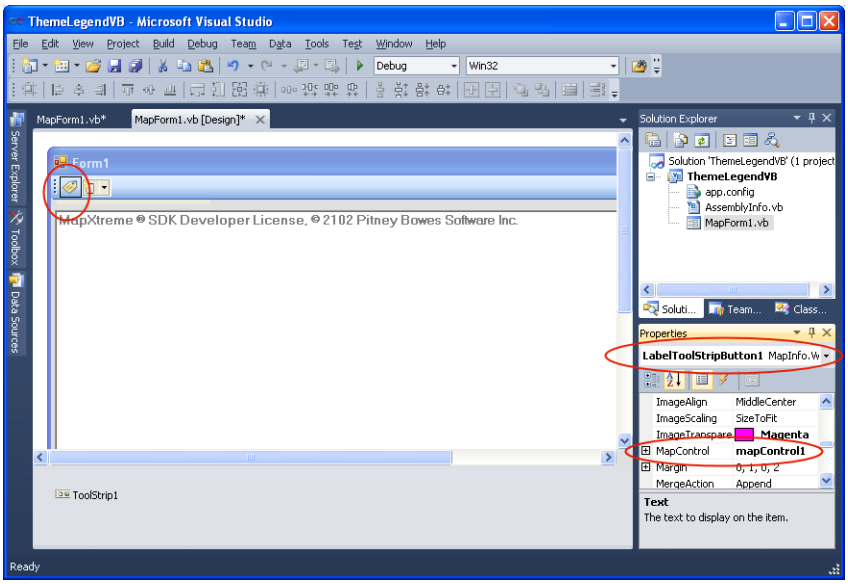
16. In Visual Studio, open MapForm1.vb in Design mode.

17. From the Menu and Toolbars group in the Toolbox, choose ToolStrip and click on the MapControl where you wish to add a Label tool. A split button appears.

18. Click the down arrow on the split button and choose the LabelToolStripButton from the list of buttons.



Notice that a Label tool displays on the form in the upper left corner and is automatically assigned to mapControl1 in the Properties windows.



As you can see, you can use the samples as a base for your own customization. You can substitute your own data, modify themes, and add tools easily. Other modifications we could make to this thematic map include:

- Using a different column from the data source (table) so that the theme is based on a different value, for example population. Don't forget to refresh the legend to match.
- Adding additional themes and legends.
- Adding additional tools.

Building Under Release Mode

When you have completed the modifications, you are ready to do a release build. The release build is the build of the application that you will release to customers. or internal users. To make a release build:

1. Choose **Build > Configuration Manager**.
2. In the **Active Solution Configuration** drop-down list, change the build type to **Release**.
3. Click **Close** in the Configuration Manager dialog box.
4. In Solution Explorer, right-click on the ThemeLegendVB project and choose Build.

When you are finished, see Packaging Your Desktop Application.

Packaging Your Desktop Application

This section of the tutorial shows you how to package your desktop application. A package is a collection of files and directories required for a software product. A product must be built into one or more packages so that it can be transferred to a distribution medium, such as a CD-ROM or DVD-ROM.The package for a desktop application consists of a Setup.exe file, which contains all the files needed to install and run your desktop application.

MapXtreme was designed to make this process as simple as possible. Using features in Visual Studio as well as automation in MapXtreme, the correct merge modules will be included in your package. A merge module (MSM) is a single package that contains all files, resources, registry entries, and setup logic necessary to install a component. A list of MSMs can be found in Deploying Your Application as well as a discussion about MapXtreme's runtime installer and other options available to you.

Creation of setup project can be enabled in Visual Studio 2015/2017 by installing "Visual Studio Installer Projects Extension". For more information see:

https://visualstudiogallery.msdn.microsoft.com/9abe329c-9bba-44a1-be59-0fbf6151054d

or

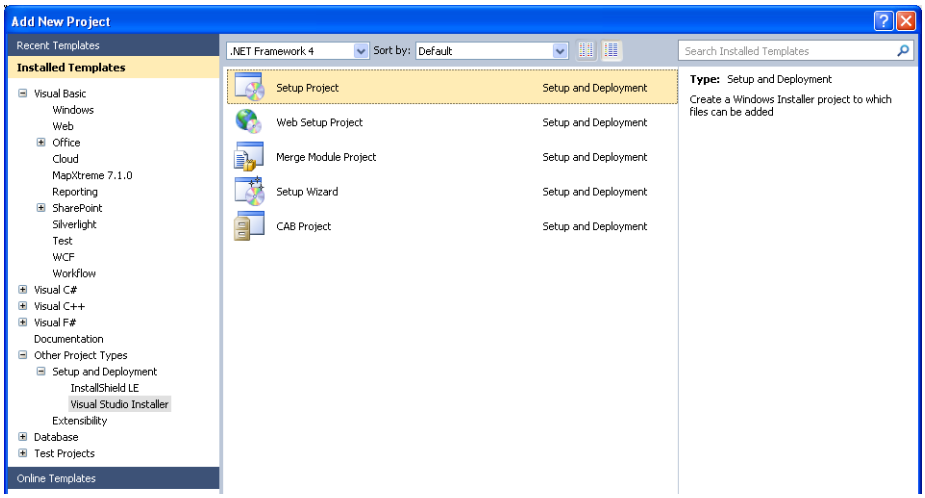
http://blogs.msdn.com/b/visualstudio/archive/2014/04/17/visual-studio-installer-projects-extension.aspx

Create a Setup Project

We will first need to create a setup project for our solution. The Setup project enables you to create the Setup.exe file.

1. In the Solution Explorer, click on the name of the Solution to highlight it and choose **File > Add New Project**. The Add New Project dialog box displays on the screen.
2. In the **Project types** list, expand **Other Project Types**, and click on **Setup and Deployment** and choose Visual Studio Installer.

   If your project is not part of a Solution, you will not see the Add Project shortcut. You can add the Setup project from File > Add Project and choose Setup. A solution will be created along with the Setup project.
3. Highlight **Setup Project**. Enter a name and location for the Setup project, and click **OK**. Visual Studio displays the File System of the new setup project.



4. Next, we need to add the data files. In the Solution Explorer, right-click on your setup project and choose **Add > File** from the menu.
5. In the **Add > File** dialog box, select All Files from the dropdown list in the Files of Type box, then navigate to the location of your data files.
6. Select all of the files you need to add to the setup project.
7. Click **Open**. The files are added to the Solution Explorer, under the ThemeLegendVB project.
8. Repeat steps 5-8 if you have files in different paths to add to your setup project.

ⓘ If you are deploying this application to another machine, the absolute paths for the data on the development computer and the deployment computer must be identical.

To view or change the path that the setup file is written to:

1. Right-click on the setup project and choose **Properties** from the menu.
2. In the Setup Property Pages, the Output File Name can be changed. This is also where you choose the configuration (debug or release) to build into the setup file.

If you wish to place your built application in the Start menu, on the File System Tab, right click on the User's Programs Menu and click Create shortcut to User's Programs Menu and set the name and properties as needed. Alternatively, a script can be written to place the program on the start menu exactly where you would like it placed (for example, in a sub-menu).

Similarly, to place a shortcut on the user's desktop, right click on the User's Desktop and click Create shortcut to User's Programs Menu and set the name and properties as needed. Alternatively, a script can be written to create the shortcut and place it on the user's desktop.

Optional Step: Add Workspace Manager and GeoDictionary Manager

If you wish, include the MapXtreme utilities Workspace Manager and GeoDictionary Manager in your package, so your customers can use them.

Build the Setup Project

When you build the setup project, a Setup.exe file is created that contains all the data, compiled code, and necessary MSMs for the project. This Setup project is part of your MapXtreme solution.

When you are finished building the setup project, then build the solution. When you build the entire solution, Visual Studio will compile and build the mapping application and then compile and build the Setup.exe file for installation. Your package is now complete and ready for deployment.

Deploying Your Desktop Application

Now that you have built your desktop application and packaged all the components for deployment, the final step is to get it onto the production workstation.

To deploy your application, you must deliver the Setup.exe file you created to your end user. You can copy the Setup.exe file to a CD-ROM or DVD-ROM to provide a means of delivering the file. When the user launches the Setup.exe executable file on their computer, the installation starts and walks them through the installation. That's all there is to it!

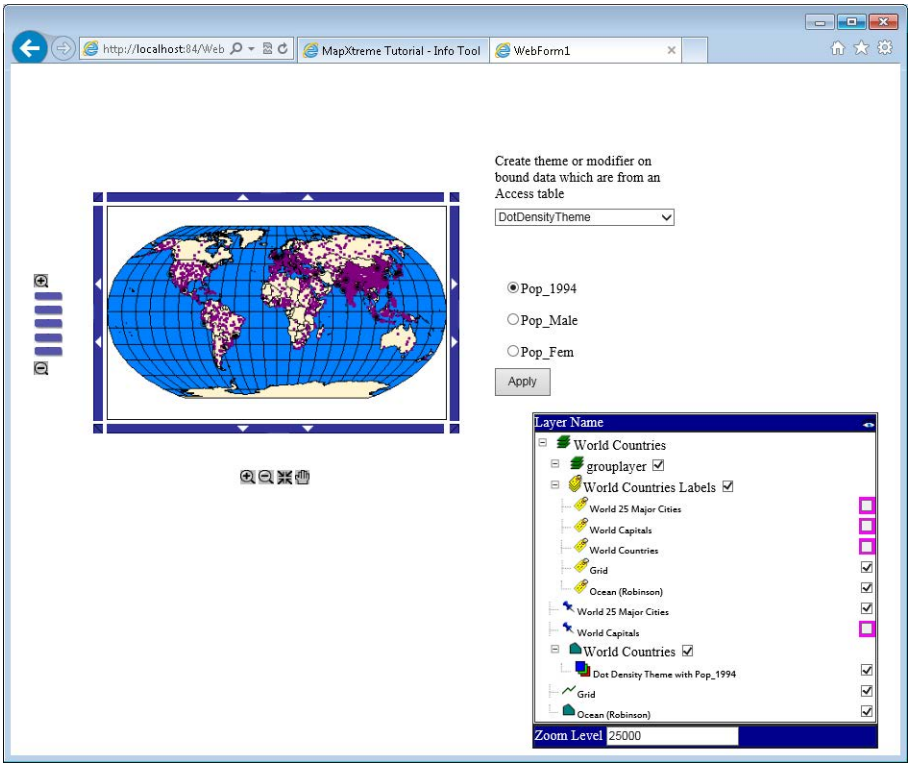
Building a Web Application

In this tutorial, we are going to base our application on a Web Thematics sample that is provided with MapXtreme. The MapXtreme Web samples are Visual Studio Web Application projects. They contain a fully functioning Web application, include pre-loaded data to support the application, and handle state management.

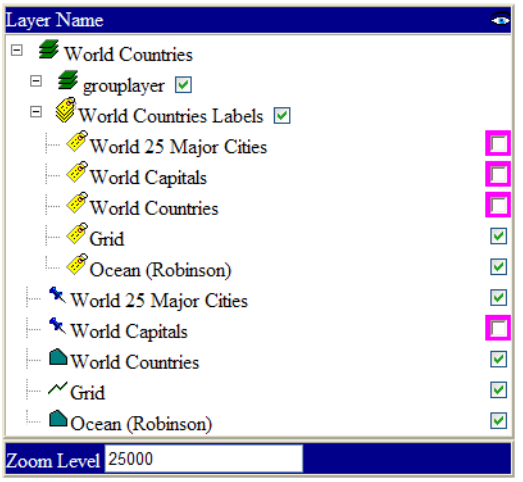
Running a Sample Web Application

Let's start by running the Thematics Web sample to see what it looks like.

1. In Visual Studio, choose **File > Open Project** and open the folder for the sample ThematicsVB. The default path to this folder is:C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\Web\Features\Thematics\ThematicsVB (where *x* is the release number of the version of MapXtreme you are using).
2. Choose **Debug>Start Debugging** (or press F5 to run the sample in Debug mode).



The Thematics sample Web Application project contains a background map of the world, a pull-down menu listing different themes to be applied and attribute columns from a MS Access table as well as an Apply button that applies the themes. The Web page contains the standard tool buttons that are included in MapXtreme web templates: ZoomIn, ZoomOut, Center and Pan. The application also includes a Layer Control so that you can set which layers are visible and view the current map zoom level.



For more information, see Components of a MapXtreme Web Application.

When you are finished, close the Web page, but leave the project open.

Modifying Your Application

To modify this sample application, let's look at the code behind page, WebForm1.aspx.vb. The code behind page is the code that represents the HTML for the Web form.

ⓘ This Thematics sample Web application is designed very differently than its desktop sample application counterpart. The most important difference is that the Web sample is tightly coupled to the data table and data column names. These items cannot simply be changed as was the case in the desktop sample.

In the code behind page, we will make some simple theme property value and range color changes. These modifications will change the themes display when the user runs The method where we will make our changes is in the CreateThemeOrModifier method, on line 244. The first change is to the Dot Density theme and how it will be displayed.

1. In Visual Studio, move your cursor to the Solution Explorer and right click on WebForm1.aspx. Choose View Code to display the code behind page, WebForm1.aspx.vb.

   If you have WebForm1.aspx displayed in Design mode, you can right-click anywhere on the form, and choose View Code.

> ⓘ All sample applications discussed in this tutorial were taken from the MapXtreme distribution DVD. The line numbers referenced may not correspond exactly with the code in your installation.

Scroll down the code page to familiarize yourself with the code.

2. Make the following changes:

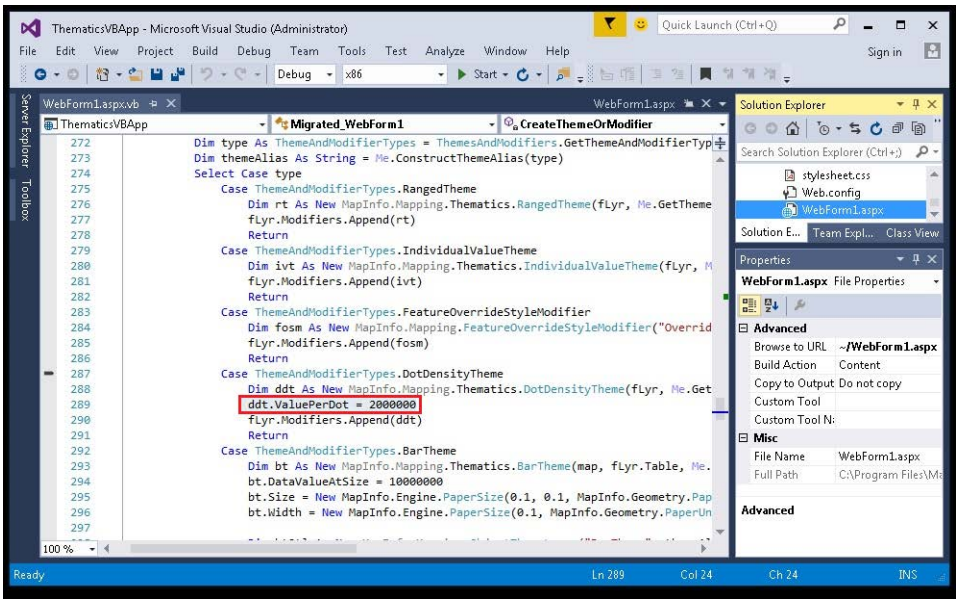
   - ·Under the ThemeAndModifierTypes.DotDensityTheme case statement, go to line 288 and change Color.Purple to Color.DarkGreen

     When the application is run, this will change the display color of the dots when selecting the Dot Density Theme.

   - ·On line 289, change the ValuePerDot property from 2000000 to 5000000.

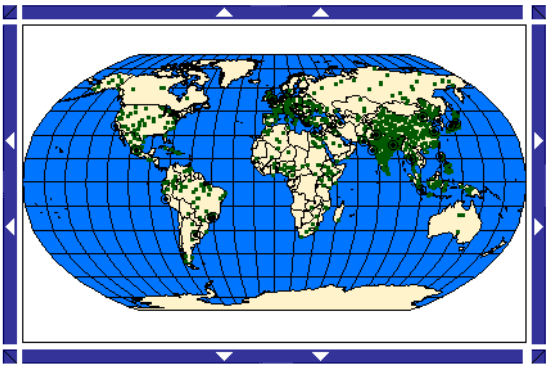
     This will change the number of dots displayed based on population. The result will be a fewer number of dots.

   In the image below, the code changes are highlighted with red boxes.



3. Choose **Debug>Start Debugging** (or press F5) to build and run the Web application. Once the page is displayed in the browser, select DotDensityTheme theme, and then check Pop_1994.

   The color of the dots on the map is now dark green.



Now let's make one more simple change to the Web sample. This change is to the Range Theme.

4. Make the following changes:

   - Under the ThemeAndModifierTypes.RangedTheme case statement, go to line 276 and change the bin count parameter from 5 to 8.

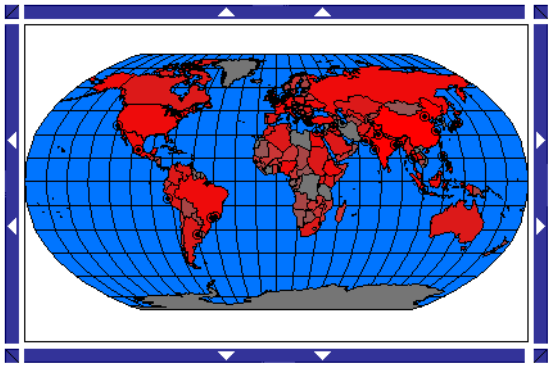
     This will change the number of data bins for the theme.

   - On the same line, line 276, change the bin distribution method from EqualCountPerRange to NaturalBreak.

     This will change the way that the data is divided in data bins.

5. Choose **Debug>Start Debugging** (or press F5) to build and run the Web application with these new changes. Once the page is displayed, select RangedTheme, and then check Pop_1994.

Notice how the theme has changed with more data bins (color codings) and how the countries have changed theme color based on the new data distribution method.



As you can see, making simple display modifications to this sample Web Application is relatively easy. Other possible modifications to this sample include:

- Changing the web page layout, such as moving the controls around.
- Changing the web page styles such as colors and fonts.

State Management Considerations

Customizing a Web application sample not only involves changing elements of the application, but also building it with state management best practices in mind. Because of the intricacies of handling applications and user state in a web environment, you must understand how MapXtreme handles them and how you can apply these concepts in your own development. State Management for Web Applications is discussed in Chapter 6 Understanding State Management.

This sample, in terms of best practices, uses a pre-loaded workspace of maps and settings, manages application and user state manually, and uses pooled session objects so that the application can service multiple requests efficiently.

It is designed to detect if the user is making a first time request, in which case the application is presented in its initial state, or if the user is revisiting the application, in which case the user's personal settings are maintained. Because this sample is sharing session objects with multiple users (known as pooling), it detects if the session is in its initial state ("clean") or contains another user's changes ("dirty").

To maintain all users' settings separately, this sample sets Session.State to manual. In a Web application where the MapXtreme Session state is saved automatically, the Session.State would be set to HttpSessionState. This means that the entire session is saved to the HTTP Session. Although this option automatically saves state, performance degrades because it does not determine what to save. The entire session is saved.

Configuring for Release Mode

When you have completed the modifications, you are ready to do a release build.

An important concept to understand in ASP.NET 2.0 is that Visual Studio knows nothing about compiling a Web Application project. Previously, ASP.NET 1.1 and Visual Studio .NET would build the code-behind source code and web forms and place the compiled code in an output directory. Visual Studio delegates all compilation responsibilities to the ASP.NET platform. The Visual Studio build command only validates the Web Application configuration.

To configure your Web Application project to be packaged without debugging turned on, you can change a configuration setting in the Web Application project's Web.config file. In the Web.config, find the XML element "compilation" (line 27) and change the value of the attribute "debug" to "false". The result should read as follows:

```
<compilation defaultLanguage="vb" debug="false">
```

This Web.config setting lets the ASP.NET compiler know to build all source code in release mode.

To build the application, right-click on the ThematicsVB project in Solution Explorer and choose Build Web Application.

Packaging Your Web Application

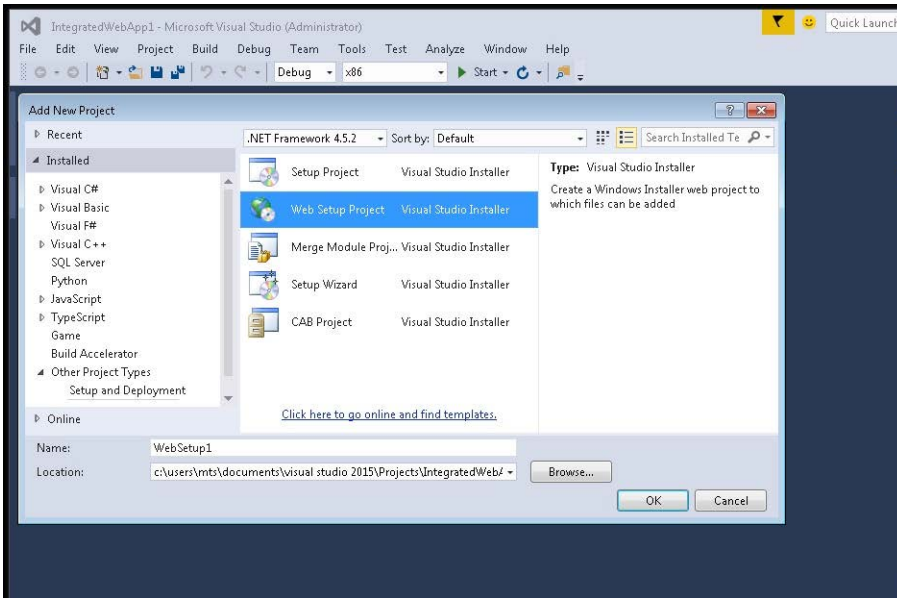
Creating a package for a Web application is similar to creating a package for a desktop application.

Create a Web Setup Project

To start we are going to add a Web setup project to our solution. To create a Web setup project:

1. In the Solution Explorer, right-click the Solution and choose **Add > New Project**. The Add New Project dialog box displays on the screen.
2. In the **Project types** list, expand **Other Project Types**, and click on **Setup and Deployment**.
3. From Setup and Deployment Projects, select **Web Setup Project**. Click **OK.** A Web setup application project will be created.

   If your project is not part of a Solution, you will not see the Add Project shortcut. You can add the Setup project from File > Add Project and choose Setup. A solution will be created along with the Setup project.

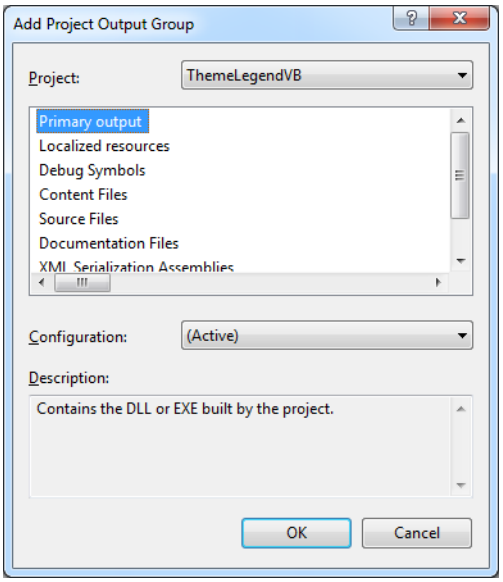


The next step is to indicate which parts of the solution to include.

4. In Solution Explorer, right-click on the newly-created Web Setup project and choose **Add > Project Output**. The Add Project Output Group dialog box displays on the screen.

5. Click on **Contents Files** and click **OK**.

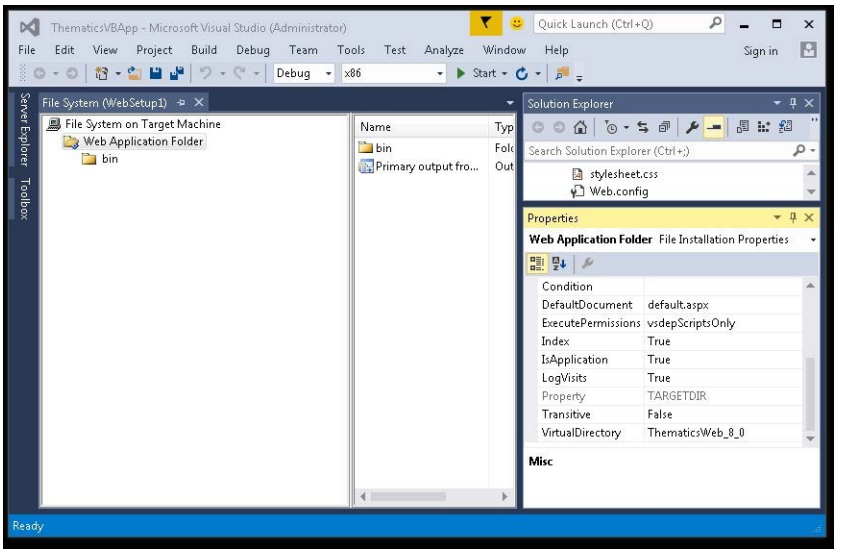
   You must include the content files because the web pages in the application are generated as HTML files rather than executables as in the desktop application.



6. If you create your own deployment for a Web Application, you will need to add the MSMs manually. For more information, see Deploying a Web Application

7. Next, you must include your data on the production server. You have the option of copying the data to the production server manually, or installing it in a separate installer.

8. In the File System window in Visual Studio for the Web Setup project, right-click the Web application folder and open the Properties window. Change the Virtual Directory name to ThematicsWeb_8_0, which defines the virtual directory on the target machine.



Build Web Setup Project

Now you are ready to build your Web setup project.

1. Choose **Build > Batch Build**. The Batch Build dialog box displays on the screen.
2. Find the Release configuration of the Web Application project and select its corresponding **Build** check box.
3. Click **Build**.

When the build is finished, using Windows Explorer go to the Release directory of the Web Setup project. You will see either a Setup.exe file or a setup.msi file. Use this file for deployment.

Deploying Your Web Application

In a Web deployment, the Setup.exe file must be run on a production server by the organization that is hosting the Web Application. The Setup file must be transferred from the development server to the production server and then installed on that server. In small organizations the deployment may be done by the developer. In larger organizations, an Administrator might perform the deployment. Your users then access the site by entering its URL in their Internet browser. For specific information on different installers, accessing data, proxy servers, and permissions, see Deploying Your Application.

For the purposes of this tutorial, let's assume that you, as the developer, will do the deployment. The production server must have the .NET Framework v. 2.0 or later, and MapXtreme installed.

Once you install your application on the production server, modify the Web.config file so that the pre-loaded workspace is referenced on the local machine.

When your Web Application is deployed, the setup application creates the necessary virtual directory in IIS and places all the other necessary components in the right places.

To see the Web Application run, open a browser, and enter the URL: http://<ProductionServerMachine>/ThematicsWeb_8_0/. The URL should be the location of the Web application on the production server, using the virtual directory that you specified when you packaged the application.

After installation you should restart the IIS server on the machine by issuing an *iisreset* in a command prompt. Another option is to recycle the app pool that the application is assigned.

B

B – Customizing MapXtreme

This appendix contains examples of classes, interfaces, and other elements of MapXtreme which can be customized.

In this appendix:

Customizable Classes

The following classes have been designed to facilitate sub-classing to create custom subclasses that more closely fit your development needs.

- MapInfo.Data.Provider Namespace
- ADO.NET
- Engine.CustomProperties
- Search
- FeatureStyleModifier or FeatureOverrideStyleModifier
- UserDrawLayer
- Windows.Controls
- Tools
- Styles
- GmlFeatureCollection
- WorkSpacePersistence and WorkSpaceLoader

MapInfo.Data.Provider Namespace

MapXtreme provides an extensible data provider model for accessing data in forms that MapXtreme cannot natively understand. This model requires an extensive amount of development and should be undertaken only when the other methods of data access provided by MapXtreme is insufficient. The model is explained in Appendix D: Extensible Data Providers.

ADO.NET

Before attempting to implement your own data provider using the Extensible Data Provider model discussed above, consider the MapInfo.Data.TableInfoAdoNet class. This class provides access to non-mappable data for which we have not provided a dedicated data source. See the Developer Reference for more information.

Engine.CustomProperties

Use the CustomProperties class to add custom information to an object. The different kinds of objects that CustomProperties can add to are: FeatureCollection, FeatureStyleModifer, GmlFeatureCollection, GroupLayer, IFeatureCollection, IMapLayer, ISession, LabelModifier, LabelSource, Legend, LegendFrame, Map, MapLayer, MapTool,

MultiFeatureCollection, Session.PooledSession, and Table. Add information using the Add method. Retrieve information using the Item method. CustomProperties can be of any type.

ⓘ Do not add MapXtreme objects directly into a CustomProperties collection. Doing so causes errors during serialization. Instead add an Alias. For example, do not add a Map to a CustomProperties collection. Instead add Map.Alias.

Here is an example of how CustomProperties is used to add and retrieve properties:

```
Public Shared Sub MapInfo_Engine_CustomProperties()
    Dim bag As CustomProperties = New CustomProperties
    bag.Add("One", 1)
    bag.Add("DateNow", DateTime.Now)
    Dim i As Integer = CType(bag("One"), Integer)
    Dim ts As DateTime = CType(bag("DateNow"), DateTime)
End Sub
```

Search

To customize your search functions, there are a few classes that you can work with to accomplish almost any kind of search that you desire. The QueryFilter class allows you to create custom “where” clause to be used in SQL queries; the QueryDefinition class allows you to define a custom SQL query to be executed; and a SearchResultProcessor sets up the post processing of the results of your query.

For an example of how these classes are used, refer to the Search sample application included in the Samples directory of your MapXtreme installation (the default installation location is: C:\Program Files\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features\Search).

QueryFilter

The IQueryFilter interface defines the interface that all query filters must support. A QueryFilter is used to define all or part of the “where” clause of a QueryDefinition.

QueryDefinition

The QueryDefinition class defines a query to be executed by a search. The QueryDefinition is made up of a filter (IQueryFilter), Columns, and OrderBy. If no Columns are specified, then “*” is used.

ISearchResultProcessor or SearchResultProcessor

The SearchResultProcessor implements the ISearchResultProcessor interface which is used to do post processing on the results of a search to narrow down the rows selected.

FeatureStyleModifier or FeatureOverrideStyleModifier

Create your own FeatureStyleModifiers to change the appearance of a layer by deriving from FeatureStyleModifier or FeatureOverrideStyleModifier. These classes are used to specify a specific FeatureStyle modification for the objects in a specific layer.

The following example creates a modifier that sets regions in USA.TAB to red if the population has decreased from 1990 to 2000.

```
using MapInfo.Mapping;
using MapInfo.Styles;
using System.Drawing;

internal class UsaPopulationDecreasedModifier :
MapInfo.Mapping.FeatureStyleModifier
{
   public UsaPopulationDecreasedModifier() : base(null, null)
   {
      // use 2 columns for expressions
      Expressions = new string[] { I18N.Wrap(
              "Pop_1990",I18N.WrapType.Column),
              I18N.Wrap("Pop_2000",I18N.WrapType.Column) };
   }

   // called during draw events
   protected override System.Boolean
         Modify(MapInfo.Styles.FeatureStyleStack styles,
         object[] values)
   {
      // compares the values from each column
      if ( double.Parse(values[0].ToString()) >
         double.Parse(values[1].ToString()) )
      {
         // if population decreased, color the region red
         CompositeStyle style = styles.Current;
         if (style.AreaStyle.Interior is SimpleInterior)
         {
            ((SimpleInterior)style.AreaStyle.Interior).ForeColor =
                 Color.Red;
            // modifies the region
            return true;
         }
      }
         // does not modify the region
         return false;
```

```
   }
}
```

UserDrawLayer

The UserDrawLayer class is used to draw a custom layer in your map. You can populate this layer with anything you like, including a customized logo, a transparent overlay of points, etc. To use this class derive a new class from it and overload the draw method. The sample code below illustrates this:

C# example:

```
using System.Runtime.Serialization;
using System;
using System.Drawing;
using MapInfo.Mapping;

[Serializable]
    class MyUserDrawLayer : UserDrawLayer {
    // Call the base class constructor with name and alias.
    public MyUserDrawLayer(string Name, string Alias) : base(Name, Alias) {}

    // Override the abstract Draw method to draw a rectangle.
    public override void Draw(System.Drawing.Rectangle ClientRect,
System.Drawing.Rectangle
        UpdateRect, Graphics graphics) {
        // Create a pen.
        Pen blackPen = new Pen(Color.Black, 3);
        // Create location and size of rectangle.
        float x = 10.0F;
        float y = 10.0F;
        float width = 200.0F;
        float height = 200.0F;
        // Draw rectangle to screen.
        graphics.DrawRectangle(blackPen, x, y, width, height);
    }
}
```

A VB UserDrawLayer sample application is provided with MapXtreme in the ..\MapInfo\MapXtreme\9.x.x\Samples\Desktop\Features folder.

Windows.Controls

Many of the classes in the Windows.Controls namespace can be sub-classed to implement custom behavior. Below are two examples of customizing the LayerControl using this approach.

MapInfo.Windows.Controls.PropertiesUserControl

If you want to add your own custom tabs to the LayerControl, create a class that subclasses PropertiesUserControl. Then add your class to the collection of other tab classes by calling the LayerControl's GetLayerTypeControls() and SetLayerTypeControls() methods. For an example of how this works, refer to the sample application included in your Samples\Features directory of your MapXtreme installation. Look in the MapBackgroundControl.cs file in the LayerControl sample application for the class MapBackgroundControl which subclasses PropertiesUserControl.

MapInfo.Windows.Controls.LayerNodeHelper and all of its subclasses

LayerNodeHelper classes control the appearance and behavior of nodes in the LayerControl's layer tree. For example, these helper classes dictate which layers the user is allowed to remove. If the properties of the ILayerNodeHelper interface do not give you enough control over the appearance or behavior of layer nodes—for example, if you want to apply per-table logic, where the user is allowed to remove some map layers, but not others—you can subclass any of these helper classes, override the appropriate method, and perform your per-layer logic in your overridden method. Then put your new helper class to use by calling the LayerControl.SetLayerTypeHelper method.

By default, when the user removes a GroupLayer, a confirmation dialog appears, except in cases when the GroupLayer is completely empty—in that case, the empty GroupLayer is simply removed with no confirmation prompt (LabelLayers work similarly). To force the Layer Control to always display a confirmation prompt, even when the GroupLayer is empty, create a class that derives from GroupLayerNodeHelper and override the IsRemovalConfirmed method. The code example below illustrates how this is done.

```
using System;
using MapInfo.Windows.Controls;

namespace MapInfo.Samples.LayerControl
{
   public class CustomLayerNodeHelper : GroupLayerNodeHelper
   {
      public CustomLayerNodeHelper()
      {
      }
      public override bool IsRemovalConfirmed(object obj)
      {
         return true;
      }
   }
}
```

Then, to use this new helper class in your application, call the LayerControl's SetLayerTypeHelper method, using syntax such as this:

```
layerControlDlg.LayerControl.SetLayerTypeHelper(
   typeof(GroupLayer), new CustomLayerNodeHelper() );
```

The subclasses of LayerNodeHelper are:

- MapNodeHelper
- MapLayerNodeHelper
- GroupLayerNodeHelper
- LabelLayerNodeHelper
- LabelSourceNodeHelper
- LabelModifierNodeHelper
- FeatureStyleModifierNodeHelper
- RangedThemeNodeHelper
- DotDensityThemeNodeHelper
- IndividualValueThemeNodeHelper
- PieThemeNodeHelper
- BarThemeNodeHelper
- GradSymbolThemeNodeHelper

Tools

Tools can be customized in two ways: by sub classing an existing tool or by adding code to a Custom base tool. See Chapter 7 Desktop Applications, Controls, Dialogs, and Tools for more information desktop tools. See Chapter 5 Web Applications, Controls, and Tools for information on web tools.

Styles

The following classes can be used to add customization to your collections of styles to apply to objects.

BitmapPointStyleRepository

The BitmapPointStyleRepository class is used to iterate through all current bitmap point styles and allows you to add new bitmap images from a specified directory. Using the Refresh() method places any new bitmap images into the repository that holds all images. The following code sample demonstrates the reloading of the collection of images used for BitmapPointStyles from a directory named “C:\MyData\MyBitmapImages.”

```
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
BitmapPointStyleRepository bmpPointStyleRepository =
             styleRepository.BitmapPointStyleRepository;
bmpPointStyleRepository.Reload("C:\MyData\MyBitmapImages");
```

InteriorStyleRepository

The InteriorStyleRepository class is used to iterate through all current interior styles and allows you to add new bitmap images for new interior styles from a specified directory. To add onto the current set of interior patterns used to fill regions use the AppendBitmapPattern() method. The following code sample demonstrates adding a new BMP image to the InteriorStyleRepository.

```
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
InteriorStyleRepository interiorStyleRepository =
             styleRepository.InteriorStyleRepository;
// AppendBitmapPattern returns the zero-based index of the pattern in
// the repository. The index returned may be used to
// retrieve a SimpleInterior style using the repository indexer.
int index = interiorStyleRepository.AppendBitmapPattern(
     "C:\MyImages\trees.bmp"
     );
if (index >= 0)
{
   SimpleInterior style =
             (SimpleInterior)interiorStyleRepository[index];
}
```

LineStyleRepository

The LineStyleRepository class is used to iterate through all current line styles and allows you to add new line styles. You can use the Reload() method to put the new file into the repository. The following sample code demonstrates reloading the collection of line style patterns used for SimpleLineStyles from a PEN file in the “C:\MyData” directory.

```
using MapInfo.Styles;
```

```
StyleRepository styleRepository = Session.Current.StyleRepository;
LineStyleRepository lineStyleRepository =
          styleRepository.LineStyleRepository;
lineStyleRepository.Reload("C:\MyData\MyLineStyles.PEN");
```

VectorPointStyleRepository

The VectorPointStyleRepository class is used to iterate through all current vector symbols and allows you to add new vector symbols from a specified file. In order to create your own version of the symbol set, you need to use the Symbol Maker MapBasic application. Once your vector symbol set is changed you use the Reload() method to read the symbol set into the repository. The following sample code demonstrates reloading the collection of SimpleVectorPointStyles from an FNT file in the “C:\MyData” directory.

```
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
VectorPointStyleRepository vecPointStyleRepository =
                styleRepository.VectorPointStyleRepository;
vecPointStyleRepository.Reload("C:\MyData\MyVectorSymbols.FNT");
```

GmlFeatureCollection

The GmlFeatureCollection class is used to import features from an XML file. You can add Features to a map by creating a FeatureCollection object and constructing FeatureObjects to add to it. Then you can insert the FeatureCollection into a table.

WorkSpacePersistence and WorkSpaceLoader

The WorkSpaceLoader can be subclassed so that the persistence file being used can hold extra information saved from the application. The developer can have the application write any information that they would like saved into the persistence XML file by adding it under the UserData node. When the WorkSpacePersistence class writes the persistence file and there is some content in this node, the content is automatically written to the file. The WorkSpaceLoader class or some derived version of the class reads in all the data including what is a child of the UserData node. The following code shows an example of this.

```
using MapInfo.Persistence;

class myWorkSpaceLoader : WorkSpaceLoader
{
   public myWorkSpaceLoader(string fileName): base(fileName)
   {
      // empty
   }
```

```
   override public void Load()
   {
      // called parent Load() method.
      base.Load();

      XmlNode userData = this.UserDataXmlNode;
      foreach (XmlNode childNode in userData.ChildNodes)
      {
         string text = childNode.Name;
         text = childNode.InnerText;
         // user can do their own load stuff here
      }
   }
}
```

Workspace Manager Extensions

MapXtreme's workspace-building tool Workspace Manager can be extended to add new menu commands, tools and tab windows that make creating workspaces easier.

For details on the Workspace Manager capabilities and user interface, see Chapter 27 Workspace Manager.

Workspace Manager extensions are .NET assemblies that you write to extend the functionality of Workspace Manager. Most likely you will build these from the MapXtreme API to add capabilities from the object model that are not exposed in Workspace Manager.

For example, you can add more menu items, tools, tab windows or react to change events. Layer Control is also extensible, so you can add new capabilities to its menu in the same way.

Once your extension is ready to use, simply load it via the new Extensions menu on Workspace Manager's updated menu strip. It can also be autoloaded to be available when Workspace Manager starts.

Examples of Workspace Manager extensions you might want to build include a table browser with sorting capabilities, custom theme templates, or new tools for object creation and editing.

Creating a Workspace Extension

MapXtreme provides an assembly called MapInfo.WorkspaceManager.Extension.dll that defines the interfaces to help you add your own functionality to Workspace Manager.

```
public interface IWorkspaceManagerExtension
```

```
{
   string Name
      {
      get;
      }
   string Version
      {
      get;
      }
   bool HasPropertiesDialog
      {
      get;
      }
   void ShowPropertiesDialog(IWin32Window owner);

   IWorkspaceManagerNotifications Load(IWorkspaceManager callback);

   void Unload();
}
```

To create an extension, create an assembly with at least one class that implements the IWorkspaceManagerExtension interface. The class should have a constructor that takes no arguments.

Load Method

The main method on IWorkspaceManagerExtension is:

```
IWorkspaceManagerNotifications Load(IWorkspaceManager callback);
```

This method is called after the extension class is constructed. It is passed an instance of an object that implements IWorkspaceManager which allows the class to call back into Workspace Manager to access elements of the user interface such as menus, toolstrips, tabcontrol and to execute commands such as loading and saving workspaces.

```
public interface IWorkspaceManager
   {
      bool NewWorkspace();
      bool LoadWorkspace(string filename);
      void SaveWorkspace(string filename);
      void MarkWorkspaceChanged();
      bool HasWorkspaceChanged();

      string LoadedWorkspaceFileName{get;}

      int AddMapTab(MapInfo.Mapping.Map map);
      int FindMapTab(MapInfo.Mapping.Map map);
      int AddLegendTab(MapInfo.Mapping.Legends.Legend legend);
      int FindLegendTab(MapInfo.Mapping.Legends.Legend legend);
      void RemoveTab(int tab);
```

```
    Microsoft.Win32.RegistryKey GetRegistryKey();

    MapInfo.Windows.Controls.MapControl MapControl{get;}

    MapInfo.Windows.Controls.MapControl GetMapControlFromTab(int tab);
    MapInfo.Windows.Controls.LegendControl GetLegendControlFromTab(int tab);
    MapInfo.Windows.Controls.LayerControl LayerControl{get;}

    System.Windows.Forms.Form MainForm{get;}

    System.Windows.Forms.MenuStrip Menu{get;}
    System.Windows.Forms.ToolStrip FileToolStrip{get;}
    System.Windows.Forms.ToolStrip MapToolStrip{get;}
    System.Windows.Forms.ToolStrip ToolsToolStrip{get;}
    System.Windows.Forms.StatusStrip StatusStrip{get;}
    System.Windows.Forms.TabControl MainTabControl{get;}

    ContextMenuStrip MapContextMenu{get;}
    ContextMenuStrip LegendContextMenu{get;}
    ContextMenuStrip GenericTabContextMenu{get;}
  }
```

Event Handling

If your extension needs to receive event notifications from Workspace Manager, add a class that implements the IWorkspaceManagerNotifications interface and returns it from the Load() method. Workspace Manager calls this class when a workspace is created, loaded, or saved.

```
public interface IWorkspaceManagerNotifications
  {
    void OnNewWorkspace();
    void OnWorkspaceLoaded(WorkSpaceLoader workspaceLoader, string filename);
    void OnWorkspaceSaving(WorkSpacePersistence workspacePersistence, string
filename);
    void OnWorkspaceSaved(string filename);
    void OnLayerControlDialog(MapInfo.Windows.Dialogs.LayerControlDlg dlg);
  }
```

Loading Your Extension

Your Workspace Manager extension can be loaded from the Extensions > Load Extension menu command. The Open dialog displays where you can navigate to the location of your assembly.

Use the provided Extension Manager dialog to show the loaded and unload extensions. This is reachable from the Extensions > Manage Extensions command. It can invoke a properties dialog if one is available.

Loading an Extension from In-memory Assemblies

Extensions can also be autoloaded. On startup, Workspace Manager looks for extensions to load in two places.

First, it scans all of the assemblies already loaded into the current AppDomain to see if any types implement IWorkspaceManager. This is an advanced scenario and the only likely case for using it is if an SessionEventHandler (see ISessionEventHandlers) happens to also contain a Workspace Manager extension.

Second, Workspace Manager looks in a folder called Extensions in the sample location as WorkspaceManager.exe. The assembly must be named `<my_extension>.WorkspaceManagerExtension.dll`. All extension assemblies in this location are loaded into the same AppDomain as Workspace Manager.

Command Line Arguments for Loading Extensions

You can also control from the command line where Workspace Manager should look for extensions to load:

```
/LoadExtensions=[All|Folder|None]
```

Where:

`All` means to look in loaded assemblies and look in the Extensions folder

`Folder` means only look in the Extensions folder and skip the loaded assemblies

`None` means do not autoload any extensions

Unloading Your Extension

An extension can be unloaded by the user or when Workspace Manager is exiting. The extension should be designed to remove any added menus, toolbars, or tab windows etc. and free up as many resources as it can. The extension assembly will be unloaded from the AppDomain once Workspace Manager is closed.

Sample Extension

Here is an example extension for Workspace Manager that loads the previously-used workspace on startup:

```
using MapInfo.WorkspaceManager.Extension;
using System.Windows.Forms;
using System;

public class LoadLastWorkspaceExtension : IWorkspaceManagerExtension
```

```
{
      private string _name;
      private IWorkspaceManager _callback;

      private LoadLastWorkspaceExtension()
      {
            _name = "Load Last Workspace Extension";
      }

      public string Name
      {
            get { return _name; }
      }

      public string Version
      {
            get { return "0.57"; }
      }

      // this get called when extension is first loaded
      // use it to hook up and UI like menu items, toolbars, etc
      public IWorkspaceManagerNotifications Load(IWorkspaceManager callback)
      {
            _callback = callback;

            Microsoft.Win32.RegistryKey key = _callback.GetRegistryKey();
            string s = (string)key.GetValue("RecentFiles");
            if (s != null && s.Length > 0)
            {
                  try
                 {
                        string[] files = s.Split('|');
                        callback.LoadWorkspace(files[0]);
                 }
                  catch { }
            }
            key.Close();

            return null; // returning because we don't need the notifications
      }

      public void Unload()
      {
             // nothing to clean up
      }
      public bool HasPropertiesDialog
      {
            get { return false; }
      }

      public void ShowPropertiesDialog(IWin32Window owner)
      {
```

```
            throw new NotImplementedException();
        }


}
```

Location of Application Data Files

Any MapXtreme application uses data stored in the following files.

File Type	Filename
Abbreviation file	MAPINFOW.ABB
Pen file	MAPINFOW.PEN
Projection file	MapInfoCoordinateSystemSet.xml
Vector symbol file	MapInfow.fnt
Custom symbol directory	CustSymb
Nadcon files	*.las, *.los
jgd2000 files	jgd2000.*

By default, MapXtreme applications look in the following directories for data files:

- C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x—This is the directory the MapXtreme installer places these files.
- The directory where your application is located. For a Windows application, this is the directory where the .exe file is located. For web applications, this is the directory where the Web.config file is located.
- MapInfo.CoreEngine assembly folder—This directory is the last place any application looks.

The list of directories corresponding to the above locations is obtained via the ISessions.AppDataPaths property which returns the list as an array of strings. While you cannot add to this list programmatically, you can add to it by defining a custom section in the application's configuration file. This is illustrated in the sample code below. Note that the order in which additional paths are defined determines the order in which they are searched, and all custom locations are searched before the default locations listed above.

The following example shows the <SpecialPath> tag:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <section name="MapInfo.CoreEngine"
      type="MapInfo.Engine.ConfigSectionHandler,
      MapInfo.CoreEngine, Version=6.8.0.536, Culture=neutral,
      PublicKeyToken=93e298a0f6b95eb1" />
   </configSections>
   <MapInfo.CoreEngine>
      <ApplicationDataPaths>
         <SpecialPath>
            <LocalApplicationData>YourCorporation\Your
Application<\LocalApplicationData>
         <\SpecialPath>
      </ApplicationDataPaths>
   </MapInfo.CoreEngine>
</configuration>
```

In this example MapXtreme will search for custom symbols in a folder like C:\Documents and Settings\username\Application Data\Your Corporation\Your Application\CustSymb. Note that the <SpecialPath> element should *not* include the CustSymb folder name. If it does, the path would be interpreted as ...\Your Application\CustMapXrtreme 2004\CustSymb\CustSymb. The duplicated "CustSymb" would be incorrect.

Use the <SpecialPath> syntax if the application data is stored in a location relative to a .NET Framework special system folder. For example, if your application data is stored in a directory named MyAppData located under the “My Documents” directory, then the entry in the configuration file could be:

```
<Personal>MyAppData</Personal>
```

where “Personal” is the value of the .NET Framework enumeration Environment.SpecialFolder that represents the “My Documents” directory.

Each element in the configuration element above is defined as follows:

<configSections>
This is where handlers for custom sections are declared. In this case the ConfigSectionHandler class defined in assembly MapInfo.CoreEngine is responsible for parsing the MapInfo.CoreEngine section of the configuration file.

<MapInfo.CoreEngine>
This section contains settings for the MapInfo.CoreEngine assembly. Specifically, application data search paths.

<ApplicationDataPaths>
This section defines additional paths used by MapXtreme when searching for application data files.

<Path>
An element to use if the full path to the application data is known.

<SpecialPath>
Use this syntax if the application data is stored in a location relative to a well-known system folder. For example, if your application data is stored in a directory named MyAppData, located under the “My Documents” directory, then the entry in the configuration file looks like `<Personal>MyAppData</Personal>`, where “Personal” is the value of the Environment.SpecialFolder enumeration that represents the “My Documents” directory. Any of the enumeration values in the Environment.SpecialFolder can be used to define an application data path in the configuration file.

Find Abbreviation File

The Abbreviation file can be customized to match your data to make your find operations more efficient. See Fine Tuning the Find Process for more information.

C

C – Understanding the MapInfo Workspace

MapXtreme provides a workspace format that is portable, interoperable, and uses the MapInfo codespace definition. This appendix covers its definition, capabilities and use. For more about the MapInfo codespace definition, see Appendix G: Defining the MapInfo Codespace.

In this appendix:

What is the MapInfo Workspace?

Using MapXtreme, you can persist the maps you create as XML-based workspaces (with an .MWS extension) that are portable and interoperable. You will then be able to share these maps with anyone else using MapInfo products regardless of their working environment.

If you are familiar with MapXtreme Java, MapX, or MapInfo Professional, you know that currently users persist maps using different file formats:

- MapXtreme Java maps are saved as Map Definition Files (.MDF files)
- MapX maps are saved as geosets (.GST files)
- MapInfo Professional maps are saved as workspaces (.WOR and .MWS files)

These files have been the way for users of individual MapInfo products to share maps. MapInfo Professional workspaces can also contain settings for browser and layout windows, graphs, legends, and sometimes even printer settings.

The MapInfo Workspace format supports the creation of **named resources** for easier access and portability to connections, map definitions, map layers, data source definitions and styles. The main workspace schema is called MXP_Workspace_1_5.xsd. It is located on the product media. Other supporting schemas include MXP_MapDefinition_1_5.xsd for map definitions and MXP_NamedReources_1_5.xsd for named layers, connections, data source definitions and styles.

Our XML documents use textual identifiers and, where possible, we have used identifiers defined by XML standards organizations. However, XML, being eXtensible, allows for identifiers to be added by any document author to clarify the meaning of the data used in the document. To ensure that these identifiers are clearly labeled as defined by Precisely, we specify them in what we call the MapInfo Codespace. To review the MapInfo Codespace identifiers, see Appendix G: Defining the MapInfo Codespace.

For more information about creating workspaces and about using the Workspace Manager, see Chapter 27 Workspace Manager.

ⓘ You cannot use .WOR workspaces created in MapInfo Professional with MapXtreme. MapXtreme can read MapX geosets.

Structure of a Workspace

To give you a sense of the structure of a workspace, let's look at a workspace and the XML code behind it. We will be looking at the World.MWS workspace, which is in the Sample directory of your MapXtreme installation. Here, we describe four of the five types of data in the workspace.

1. The Header contains the basic file information including the version type and the creation date.
2. The Connection Section defines the database, file and WMS connections that are contained in the workspace. This is where the named connection information is stored.
3. The DataSourceDefinition Section lists the definitions of the data and where it is located. If you have named data source definitions, they would be included here.
4. The MapDefinition Section contains definitions of layer, theme, and label features associated with the workspace, such as the label details, the zoom range, the colors used, etc. Named definitions for the map, layers, themes, styles would be included here.

An additional section in the schema is for user-defined data. This allows applications to persist their own data. User data is a "wild card" element meaning you can enter any content you want because it will not be validated against the schema.

For more information on named resource support in MapXtreme, see Opening and Saving a Workspace Containing Named Resources.

Header Section

The file begins with the required XML file header information that describes the XML version the file conforms to, the encoding description, and a field that indicates whether the file is a standalone. The top level element in the file is the Workspace Element, which contains attributes for the file version, date, and XML namespace definitions.

ⓘ The supported encoding for workspace persistence is UTF-8 for MapXtreme.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<WorkSpace name="" version="MXP_WorkSpace_1_5"
xmlns:gml="http://www.opengis.net/gml" xmlns="http://www.mapinfo.com/mxp">
```

Connection Section

The ConnectionSet section defines the database connections and file connections that are necessary to use the workspace. Currently supported database connections include MS Windows ODBC connections, Oracle (OCI) connections, and JDBC driver connections. File connections identify the path to the file. What follows are some examples of these connections.

```
<ConnectionSet>
     <DBConnection dbType="sqlserver">
        <ConnectionName>JCD_SS2005_MIGS_BOUNDARY2</ConnectionName>
        <ODBCConnectionString>driver={SQL Native Client};server=JODEMPSE-
W3\MIGS;database=MIUS06_BOUNDARY2;uid=MIGS;</ODBCConnectionString>
     </DBConnection>
   </ConnectionSet>

<ConnectionSet>
    <FileConnection dbType="file">
      <ConnectionName>MapStorage1_US</ConnectionName>
      <FilePath>c:\data\maps</FilePath>
    </FileConnection>
   </ConnectionSet>
```

DataSource Definition Section

The DataSource Definition section defines the data files (for example, TAB files) and database tables that are retrieved at each connection location. Optional metadata may also be included to describe the data that is retrieved. This example shows a named data source definition, a database datasource definition, and a TAB data source definition.

```
<DataSourceDefinitionSet>
   <NamedDataSourceDefinitionRef resourceID="MyDataSource"/>
   <DBDataSourceDefinition id="id1" readOnly="false">
      <DataSourceName>alias_Line_ontario_oracle_9i_rel
      </DataSourceName>
      <ConnectionMember>
         <ConnectionName>ontario_oracle_9i_release_1</ConnectionName>
      </ConnectionMember>
      <DBQuery>
         <Query>select * from us_hiway_extra</Query>
      </DBQuery>
      <DBDataSourceMetadata>
         <FeatureGeometryAttribute srsName="mapinfo:coordsys 1,62">
            OBJECT
         </FeatureGeometryAttribute>
      <KeyAttributes />
      </DBDataSourceMetadata>
   </DBDataSourceDefinition>
   <TABFileDataSourceDefinition id="id2" readOnly="false">
      <DataSourceName>alias_Line_us_hiway_extra</DataSourceName>
```

```
      <FileName>FeatureLayerBuilder\us_hiway_extra.tab</FileName>
   </TABFileDataSourceDefinition>
</DataSourceDefinitionSet> ...
```

Map Definition Section

The Map Definition section defines one or more maps and their constituent layers. Each map has display conditions that include the size of the image, the zoom and center settings, the coordinate system of the rendered map. This example shows a named map definition reference and a map definition. Under the LayerList, there are two named layers, three defined layers and a named style reference for a label layer.

```
<MapDefinitionSet>
   <NamedMapDefinitionRef resourceID="MyWorldMap"/>
   <MapDefinition id="id7" name="Map1" alias="Map1" uniqueId="4adeb0e9-7c77-
4957-a3fb-a1a0677756ef">
      <DisplayConditions>
         <MapSize uom="mapinfo:imagesize pixel">
            <ImageWidth>600</ImageWidth>
            <ImageHeight>400</ImageHeight>
         </MapSize>
         <ZoomAndCenter>
            <MapZoom uom="mapinfo:length mi">3000</MapZoom>
            <gml:Point srsName="EPSG:4269">
               <gml:coordinates>-79.771366,38.003251000000006
               </gml:coordinates>
            </gml:Point>
         </ZoomAndCenter>
         <DisplayCoordSys>
            <SRSName>EPSG:4269</SRSName>
         </DisplayCoordSys>
      </DisplayConditions>
      <LayerList>
         <NamedLayerRef resourceID="NamedLayer_World 25 Major Cities"/>
         <NamedLayerRef resourceID="NamedLayer_World Capitals"/>
         <FeatureLayer id="id8" name="World25Cities"
               alias="World25Cities"
               volatile="unknown">
               opacity="0.251">
            <DataSourceRef ref="id4" />
         </FeatureLayer>
         <FeatureLayer id="id9" name="WorldCapitals"
               alias="WorldCapitals"
               volatile="unknown">
               opacity="0.502">
            <DataSourceRef ref="id2" />
         </FeatureLayer>
         <FeatureLayer id="id10" name="Regions"
               alias="Regions"
               volatile="unknown">
```

```
            opacity="0.753">
          <DataSourceRef ref="id6" />
        </FeatureLayer>
<LabelSource maxLabels="2147483647" name="World 25 Major Cities">
             <Visibility visible="true">
               <VisibleRange enabled="true">
                 <ZoomRange uom="mapinfo:length mi" minInclusive="true"
maxInclusive="true">0 8000</ZoomRange>
               </VisibleRange>
             </Visibility>
             <DataSourceRef ref="id3" />
             <BaseLabelProperties>
               <LabelProperties>
                 <LabelVisibility visible="true">
                   <VisibleRange enabled="false">
                     <ZoomRange uom="mapinfo:length mi" minInclusive="true"
maxInclusive="true">0 8000</ZoomRange>
                   </VisibleRange>
                 </LabelVisibility>
                 <LabelText justification="right">
                   <BaseLabelStyle>
            <NamedStyleRef resourceID="Layer_id7" id="Layer_id7"/>
                                      </BaseLabelStyle>
                   <StringTokenList>
                     <StringToken>
                       <StringValueExpression>
                         <AttributeName>Place_Name</AttributeName>
                       </StringValueExpression>
                     </StringToken>
                   </StringTokenList>
                   <LabelCharacterLimit
spacing="1">2147483647</LabelCharacterLimit>
                 </LabelText>
                 <LabelCallout visible="true" end="none">
                   <LineStyle stroke="black" width="1" width-
unit="mapinfo:imagesize pixel">
                     <Pen>mapinfo:pen 2</Pen>
                   </LineStyle>
                 </LabelCallout>
                 <LabelReferencePosition recalculate="outOfView">
<LineLabelPositionInterval>0</LineLabelPositionInterval>
                 </LabelReferencePosition>
                 <LabelLayout>
                   <Alignment>
                     <HorizontalAlignment>right</HorizontalAlignment>
                     <VerticalAlignment>center</VerticalAlignment>
                   </Alignment>
                   <Offset uom="mapinfo:imagesize pixel">
                     <XOffset>-2</XOffset>
                     <YOffset>0</YOffset>
                   </Offset>
<LabelRelativeOrientation>parallel</LabelRelativeOrientation>
```

```
                    </LabelLayout>
                    <LabelBaseSize useScale="false">
                      <MapScale>0</MapScale>
                    </LabelBaseSize>
                  </LabelProperties>
                </BaseLabelProperties>
                <LabelThemeList />
              </LabelSource>
      </LayerList>
    </MapDefinition>
</MapDefinitionSet>
```

The layer list displays with the label properties that are set for this map. These properties include the layer the labels are on, the file source for the labels, the visibility rules for the labels including the range at which the labels are visible, if applicable, the font and size properties of the labels, the label character limits, callouts, if applicable, label position settings, alignment, justification, and orientation of the label with regard to the point it is labeling.

Creating an .MWS Workspace Programmatically from a .GST

In MapX, you used the Map.SaveMapAsGeoset method to create a geoset from the existing map. You give users the ability to save their workspaces programmatically as well and view these workspaces in the Workspace Manager later on using the code fragment shown below.

ⓘ Users need to have a map open with one or more layers to save it as a workspace.

```
// Reads a MapX geoset, writes a MapXtreme workspace.
using MapInfo.Persistence;
using MapInfo.Mapping;
.
.
.
MapLoader MapLoader = MapLoader.CreateFromFile("my.gst");
MapExport MapExport = new MapExport();
MapExport.Map.Load(MapLoader);
WorkSpacePersistence wsp = new WorkSpacePersistence();
wsp.Save("c:\\temp\\newwork.mws");
```

Partial Workspace Loading:

Previously, if you opened a multi-layered .mws workspace file where one or more layers were corrupted or damaged, MapXtreme would throw exception on the first corrupted layer and stop loading of subsequent layers into the map.

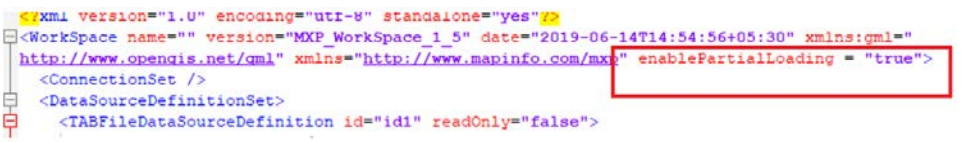
To overcome this limitation, we have introduced a property *PartialWSLoadingEnabled* in the MapXtreme application session.

When the property *PartialWSLoadingEnabled* is set to true, MapXtreme will try to load all layers except the corrupted or damaged layers. If MapXtreme finds a corrupted or damaged layer in a workspace then it skips those layers and proceeds to load the subsequent layers. It then generates an event called `WorkSpaceErrorEvent` with `WorkSpaceErrorEventArgs` as event arguments.

The `WorkSpaceErrorEventArgs` contains a list of corrupted or damaged layers and the reason of failure for each layer.

MapXtreme remembers the partial layer loading settings while saving the workspace. When the workspace is opened again the partial workspace loading settings take effect.

ⓘ When a workspace with corrupted or damaged layers is loaded and saved with the same name, the workspace file will be overwritten and you may loose existing layers from the workspace. It is advisable to save the workspace with a different names using "Save as" command.



The following classes, *WorkSpaceLoader*, *MapWorkSpaceLoader* and *MapLoader* can trigger this event. To handle this event, you must write event handler and extract error details.

Following code snippet gives a brief explanation about this:

```
private static void OnWorkSpaceEventHandler(object sender, WorkSpaceErrorEventArgs e)
{
    //Handle the event for workSpaceLoader exceptions.
    string str = null;
    foreach (var tuple in e.ErroneousLayersDetails)
    {
        str += string.Concat(new string[] { "ErrorneousLayer:", tuple.Item1, " Reason:", tuple.Item2, "," });
    }
    Console.WriteLine("Captured exception message :" + str);
}
```

Enable Partial Loading Programmatically

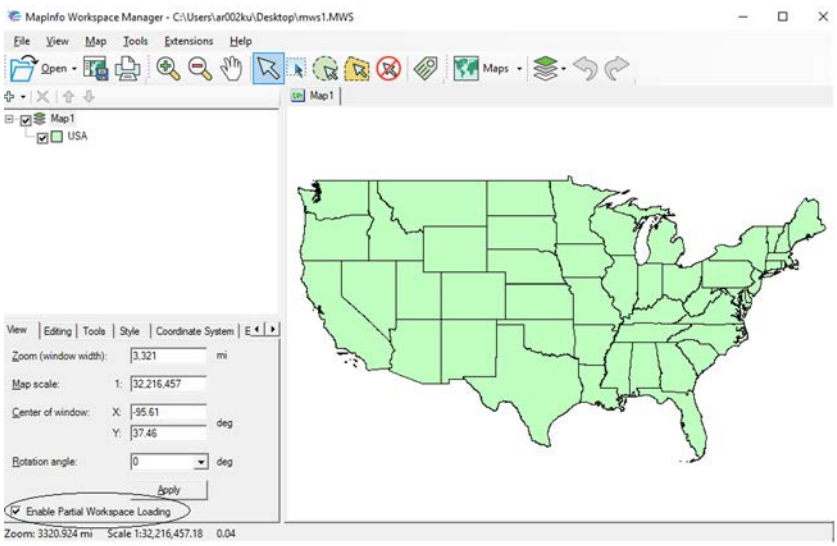
This partial loading feature can be achieved programmatically by setting the property (*PartialWSLoadingEnabled*) in MapXtreme application session as following:

```
Session.Current.PartialWSLoadingEnabled = true;
```

Enable Partial Loading through User Interface

The Partial Layer Loading feature can also be achieved from the MapControl user interface. We have added **Enable Partial Workspace Loading** checkbox in the layer tree dialog.

When the checkbox ('*Enable Partial Workspace Loading*') is selected, MapXtreme internally sets `Session.Current.PartialWSLoadingEnabled` to true.



You can enable or disable partial layer loading using the **Enable Partial Workspace Loading** checkbox in the layer tree dialog.

1. Open a map.
2. Click the map in the layer tree dialog.
3. Select **Enable Partial Workspace Loading** checkbox to enable it. Clear the checkbox to disable the feature.

D – Extensible Data Providers

This appendix presents MapXtreme's Extensible Data Provider architecture and information on how to construct one for spatial data that MapXtreme does not otherwise provide access for.

In this appendix:

Introduction

MapXtreme provides an Extensible Data Provider model that can be implemented to access data formats that are not supported in MapXtreme. This model is consists of a collection of required and optional interfaces, building blocks of abstract base classes and utilities.

Extending MapXtreme's data provider model is a difficult undertaking that requires a major commitment of development and testing resources. Most MapXtreme users find the existing MapXtreme Data Providers for spatial data formats or Microsoft's ADO.NET for non-spatial data completely sufficient for their needs. See Chapter 8 Working with Data for a complete discussion of MapXtreme's data access options.

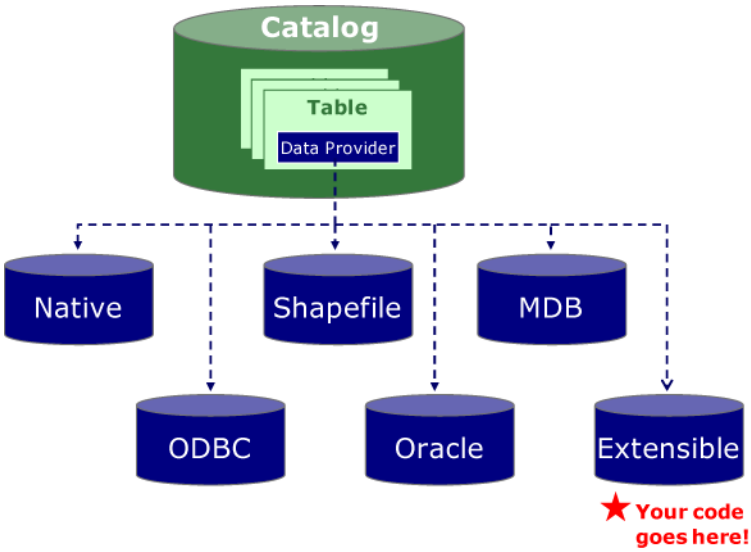
If you have data access requirements that cannot be satisfied through the MapXtreme data model, consider extending it using the interfaces and guidance presented here.

MapXtreme's Extensible Data Provider interface is organized under the *MapInfo.Data.Provider* namespace in MapXtreme's object model. Supporting these interfaces are classes in *MapInfo.Data* and *MapInfo.Data.Common* namespaces.

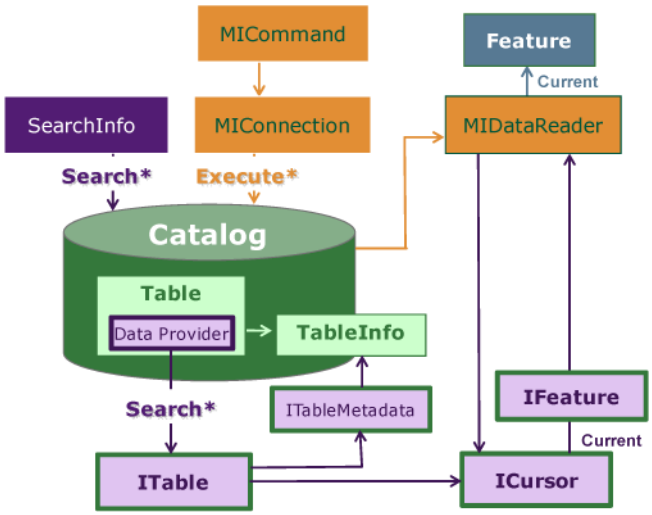
The MapXtreme extensible data provider currently supports opening a table, reading the table contents and associated metadata, searching the table contents using several methods, and modifying table content through insert, update, and delete operations.

Extensible Data Provider Overview

The Catalog in MapXtreme is built with an internal data provider architecture. The extensible data provider model is an adaptation that exposes this internal architecture through a set of .NET interfaces as the following figure illustrates.



There are a few central concepts to understanding how a data provider works and how to go about creating one. The figure below illustrates the relationship between a few of the key interfaces and the existing components of the MapXtreme data access engine. The components of an extensible data provider are shown in lavender with thick borders.



Data Provider

A data provider is a bridge between an application and a data source, which provides mechanisms for accessing data for use in the application.

The MapXtreme Extensible Data Provider is a collection of interfaces that allows you to access data from any data source in any data format. It extends the MapXtreme Data Provider which provides the connection between data and the capabilities of MapXtreme, such as display, query, edit and analyze.

The term Data Provider is used to refer to a specific implementation of the extensibility interfaces. For example, MapXtreme includes a Data Provider implementation for accessing SpatiaLite (based on SQLite) databases and FDO extension. There is an interface in the data provider collection of extensible interfaces called IDataProvider which forms the basis for a data provider implementation.

Data Source

A data source is a database management system, web service, or other engine or software API that exposes data and data access capabilities, such as describing, querying, manipulating and relating data.

An example of a data source is a WFS server that returns map data as a collection of features. The WFS server is the data source from which Feature types can be exposed as tables. The URL for the service, and possibly other properties for authentication, define how the data provider will access the data source. This information is called the data source definition.

Most data provider implementations will contain a data source; however, this concept is not required. Data providers for file based formats may contain only a table model. MapInfo tables and ESRI Shapefiles are examples of data providers that do not require a connection to a data source.

Table

A table is a set of features that have the same schema (or set of columns). Tables do not have to have a geometry property; however, it is likely that if you are building a data provider you probably have a geometry property on some of your tables. Geometry data is treated by MapXtreme as a column on a table just like other columns with simpler data types like strings or numbers.

The table exposes metadata to describe the data it contains and provides access to query and edit that data. Every feature in a table must be uniquely identified by the data provider using a key. Keys are used by MapXtreme for selections, result sets, and for editing operations.

Cursor

A cursor is an object that enumerates through a set of features. MapXtreme will request features from the data provider by calling one of the search methods on the table. The table will return a cursor that will allow MapXtreme to enumerate through the features that satisfy the criteria specified. Features that are returned through cursors are expected to be transient. That means that the data that is obtained from the current feature is assumed to be only valid while the cursor is open and positioned on that current record. Once the position of the cursor is changed (by moving to the next feature or by closing the cursor), then that feature is no longer assumed to be accessible. Robust and scalable data provider implementations should be able to use the transient nature of the cursor model to reuse memory, especially for returning FeatureGeometry objects.

A feature accessor is a special type of cursor that is used to access features by key. Any time MapXtreme needs to access one or more features by key, it will request a feature accessor and then request the features from it. The same transient expectations described for a cursor hold for a feature accessor.

Getting Started

Now that you have decided to go ahead and extend MapXtreme's data provider model, your first question is likely "Where do I begin."

Where do I begin?

We recommend that you start by reviewing this entire appendix and the reference implementation to familiarize yourself with the concepts and how they related to each other. When you are ready to begin, start by building a very simple data provider in order to get a good understanding of the requirements this undertaking involves. Even if your ultimate data provider needs are extensive, building a basic data provider using only the required elements will provide you with a good understanding of the data model.

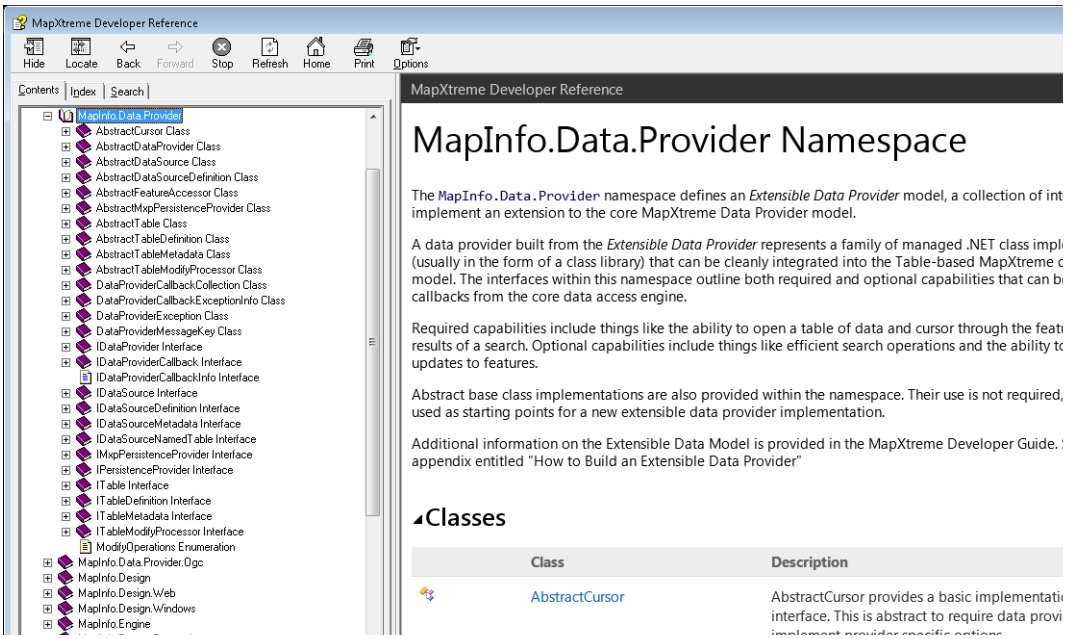
The bare minimum elements are provided in the sections Required Components. Study the Extensible Data Provider Overview to understand how the various interfaces relate to each other and to MapXtreme's data access model. Review the provided Sample: COTW (Center of the World) Data Provider to see how we have implemented a basic data provider and the SpatiaLite Sample Data Provider which is a full featured data provider installed with source code with the MapXtreme samples.

We also recommend that you study the optional building blocks provided in the SDK to learn what has already been implemented for you. See Optional Building Blocks: Base Classes, Helpers and Utilities.

One of the first steps to creating a data provider is to determine if your data provider has a data source. Not all data providers have or require this type of model and may be only table-based. A data source is any service or database that you communicate with to access potentially multiple sets of features (or tables) through the same connection. Database servers are always exposed as data sources while the tables at the database are exposed as tables. A WFS server would be modeled as a data source while the FeatureTypes that the WFS server exposes would be modeled as tables. An ADO.NET DataSet could be exposed as a data source while the DataTables it contains would be exposed as tables. An Excel spreadsheet might be exposed as a data source with selection ranges exposed as various tables. If you have a more simple data model such as an ASCII text file or some other file format that can only contain a single set of features, then you probably don't need a datasource and can simply model your data provider with only a table model. Your data may be very complex and still not require a data source model. For example, some file-based data formats may store features split across multiple files, may broken out by state or province or according to some other rules. This would still be a table-based data provider and there would be no need for a datasource.

Where do I find more detailed help?

The interfaces that comprise the Extensible Data Provider architecture are defined in the MapInfo.Data.Provider and MapInfo.Data namespaces. We recommend that you familiarize yourself with the MapInfo.Data.Provider namespace in the MapXtreme Reference Guide.



Required Components

To build a basic data provider, the following components must be implemented. These classes support basic data access operations including opening a table, reading the table contents and associated metadata, and searching the table contents.

Refer to the SpatiaLite Sample Data Provider as a complete and robust provider implementation for tips and guidance on how to implement these interfaces.

IDataProvider Interface

The IDataProvider interface provides the basis for a custom data provider implementation by exposing the capabilities for opening the table formats serviced by this provider. This also exposes capabilities related to data sources that manage those tables. Implementing the IDataSource interface is an optional task as you can build a data provider without going through a data source. This class is typically defined to be a singleton instance.

ITableDefinition Interface

The ITableDefinition interface provides the main link between your extensible data provider and the MapXtreme public API Table model. It provides the properties necessary to instantiate an ITable for a specific data provider. Classes that implement this interface identify the information required to be supplied by the user of MapXtreme to open a table.

The MapXtreme Catalog provides a few overloads to the OpenTable method that allow you and other users of your data provider to instantiate a table from an ITable. After the call to OpenTable, all other interfaces such as ITableMetadata, ICursor, IFeature, etc are hidden to the user and used internally by MapXtreme.

This interface is also typically the information that you will want to have written out to the workspace file. For more information, see Persistence Providers.

ITable Interface

The ITable interface defines the interface for a Table, the basic container of information that MapXtreme can read. query and display in a map. Methods on ITable include SearchAll, SearchByEnvelope (area of interest) and SearchByKey. A table represents a single collection of features and all features must contain the same structure (or schema).

ITableMetaData Interface

The ITableMetaData interface is used to convey important information about the properties and supported capabilities of an open table, such as the columns and their data types, the types of geometries contained in the table, and whether it supports editing operations.

The table metadata exposes the schema of the table as a collection of column definitions. Columns are exposed through the IColumn and IGeometryColumn interfaces. Any column that returns FeatureGeometry objects must be exposed as an IGeometryColumn. The IGeometryColumn also enables the provider to indicate whether it supports Z and M dimensions in the geometry values it returns and accepts during insert and update operations. Most data provider implementations will have custom implementations for IColumn and IGeometryColumn to assist with the communication of information between the MapXtreme search requests and the underlying provider implementation.

ICursor Interface

The ICursor interface is an IEnumerator that returns IFeature objects. When a search request is issued against an ITable, the result is an ICursor which provides the access to the features. The features returned from the cursor may be transient meaning that they are only expected to be valid while the state of the cursor is unchanged. Advancing the cursor to the next record (feature) may return the same feature object which may have been updated to the new values of the current record. Robust data provider implementations should take advantage of this behavior to reuse memory.

Search requests may request a subset of the attributes (columns) that the table exposes. The features returned from the cursor must match the structure of the columns requested by the search.

IFeatureAccessor Interface

IFeatureAccessor is used to fetch features by key from a table. It has one primary method that it exposes called FetchByKey which returns the feature identified by a given key. Each feature returned by the data provider must contain a key. The definition and contents of a key are determined by the data provider. When MapXtreme gives a key back to the data provider for FetchByKey requests or editing operations, the data provider will use the key value(s) to identify the correct record. The IFeatureAccessor interface provides a way for MapXtreme to bracket a set of calls so that the data provider can, if desired, prepare a command and then bind in just the key value(s) and execute the command for each FetchByKey call.

Optional Building Blocks: Base Classes, Helpers and Utilities

The MapXtreme Extensible Data Provider is a collection of interfaces that you can implement to access data in formats that are currently not supported. MapXtreme provides a collection of utilities, samples and guidance that can help you jump start your implementation. These are common implementations that most people will want to use. They cover the following areas:

- Abstract Base Classes
- Utility Classes
- Key implementations

It is well worth understanding what pieces are provided here before starting to build even the simplest implementation, as some pieces may already be built.

Abstract Base Classes

Abstract base classes for the many interfaces are provided. Their purpose is to handle the default implementations of the interface, allowing you to implement only what you need.

For example, the IDataProvider provides an OpenDataSource method and two flavors of OpenTable method (one that uses a data source, and one does not). If you are using a data source, then you have to implement OpenDataSource and the OpenTable that is based upon the data source. If you don’t have data source, you only need to implement the OpenTable that doesn’t need a data source.

The abstract base class for IDataProvider has a default implementation for each of those methods, indicating they are not supported. These are tagged as "virtual" so that when you pick which one(s) to implement you will provide an override implementation of only what you need.

Utility Classes

SimpleFeature

The basic goal of a data provider is to access features. A feature is typically a row in a table. Spatial features are described by their geometry, style, key, and attributes.

We provide a utility class called SimpleFeature, that implements the IFeature interface. It implements a relationship of defining metadata (SimpleFeatureMetdata), and a list of Attributes (AttributeValues) to a Feature, along with an identifying key (IKey)value. Most data providers will want to use the SimpleFeature class.

SimpleFeature also implements FeatureChangedEventHandler event management.

SimpleFeature is documented under the MapInfo.Data.Common namespace.

OGC Conversion for Geometries and Coordinate Systems

The MapXtreme Extensible Data Provider includes utilities that help in the conversion process from an OGC geometry (OpenGIS® Simple Features Interface Standard) into a MapXtreme geometry and vice versa. This includes OGC-regulated well-known text and well-known binaries that a number of spatial data systems take advantage of, such as PostGIS, mySQL and SQL Server 2008. Well-known text refers to a standard textual representation for spatial reference systems. Well-known binaries are a standard binary representation for geometries.

The utilities are included in the MapInfo.OGC namespace. It implements data readers and writers for well-known text and well-known binaries and supports both big endian and little endian byte ordering.

Key implementations

A requirement of data that is to be accessed by a MapXtreme data provider is that every feature has a key. We provide in this SDK two common key implementations: integer key and string key.

If your data is integer or string-based, you can skip the implementation of a key from the IKey interface, and use one of these.

Sample: COTW (Center of the World) Data Provider

The "Center of the World" Data Provider is a very simple, contrived example used to illustrate how to put a data provider implementation together. This data provider manages a table with a single row containing a spatial point at longitude/latitude (0,0).

A logical place to start a new extensible data provider implementation is with the class implementing IDataProvider. This class is required to be implemented as a singleton instance so that all references to it (for example, from ITableDefinition and IDataSourceDefinition) uniquely resolve to the same instance. Here's the beginning portion from the sample implementation:

```
using System;
using MapInfo.Engine;
using MapInfo.Data.Provider;

namespace COTW
{
      public sealed class COTWDataProvider :
      {
            private static string PROVIDER_NAME = "Center of the World Sample
Extensible Data Provider for MapXtreme";
            private static COTWDataProvider m_singleton = null;

            private COTWDataProvider(string name)
                  : base(name)
            {
            }

            public static COTWDataProvider GetInstance()
            {
                  if (m_singleton == null)
                        m_singleton = new COTWDataProvider(PROVIDER_NAME);
                  return m_singleton;
            }
     }
}
```

Note that this class is extended from AbstractDataProvider - not IDataProvider itself. As discussed in the Optional Building Blocks: Base Classes, Helpers and Utilities, we provide abstract base classes as building blocks to help provide suitable default implementations wherever possible. In the case of IDataProvider, this abstract class manages the name property and default the implementations of its OpenDataSource and

OpenTable methods to throw a not implemented exception. This allows us to decide which are relevant and provide implementations for only those thereby keeping our implementation uncluttered.

Since the table we're defining has a fixed structure, we don't need much for the ITableDefinition implementation. To make it non-trivial, we'll have the table definition accept a string value that can be used as an externally specified label to be used for the point. A complete implementation would look something like this:

```
using System;
using MapInfo.Data.Provider;

namespace COTW
{
      public sealed class COTWTableDefinition : AbstractTableDefinition
      {
            private string m_label;

            public COTWTableDefinition(string label)
                  : base()
            {
                  if (label == null)
                        throw new ArgumentNullException("label");
                  m_label = label;
            }

            public override IDataProvider DataProvider
            {
                  get { return COTWDataProvider.GetInstance(); }
            }

            public string Label
            {
                  get
                  {
                        return m_label;
                  }
            }
      }
}
```

We've once again used the abstract base class instead of the interface directly. In this case, it allows us to ignore anything related to accept the defaults for the DataSourceDefinition and CustomMetadata properties on the interface. Note how the DataProvider property references the singleton COTWDataProvider instance implemented earlier. This example also introduces a new property of our own.

Optional Interfaces

The MapXtreme Extensible Data Provider model includes optional interfaces to provide more capabilities when accessing data, including:

- IDataSource
- IDataSourceDefinition
- ITableModifyProcessor

IDataSource

The IDataSource interface is used in Catalog.OpenTable to associate a new table with a specific data source. This is optional as you can build a data provider without going through a data source.

A data source is instantiated when an ITableDefinition is being opened which contains an associated IDataSourceDefinition or directly through Catalog.DataSources.OpenDataSource method. The extracted IDataSourceDefinition is supplied to IDataProvider.OpenDataSource to connect to the data source and return the associated IDataSource instance.

IDataSourceDefinition

A DataSourceDefinition is only necessary when you are using a data source to access data. It includes the information you expect the user to supply in order to be able to instantiate a data source. If you are not using a data source, you only need to provide a Tabledefinition in order to open a table.

In the process of the table being opened, the IDataSourceDefinition is extracted from the DataSourceDefinition property and passed into OpenDataSource(IDataSourceDefinition, CustomProperties) to try and establish a connection to the data source.

When implementing a DataSourceDefinition, you should provide a meaningful override implementation of the System.Object.Equals(object) method.

ITableModifyProcessor

The table metadata indicates whether Insert, Update, or Delete operations are supported. If any of these are true, then you must supply an implementation for the ITableModifyProcessor. Implementations for the specific insert, update, and delete methods must be provided according to the corresponding individual metadata

properties. Some data providers may only support insert operations for example. The ASCII data provider supplied with MapXtreme is an example of a data provider that only supports insert operations and not update or delete operations.

Note that the ITable interface also includes a property named ReadOnly. This property is an extra level of control through which you can indicate if the table is read only. Even if your provider supplies a fully implemented modify processor, a specific table may be read only for other reasons such as insufficient access permissions or the data files are on read only media. Your data provider can check for these up front and flag the whole table as ReadOnly or through the individual table metadata properties.

Building and Testing Your Data Provider

Building a data provider is a complex task and is difficult to debug because most of the calls into your data provider code is made by MapXtreme. You will want to build up your data provider in stages and test just the pieces that you have implemented as you go. To better control this, you will probably want to start with a sample project that allows you to write code that exercises only what you expect to be implemented and you can expand this sample as you expand your data provider. You shouldn't include this directly into your data provider project; however, Visual Studio allows you to create multiple projects within a single solution.

If you have a data source as part of your data provider model, you will want to start there. Create an implementation of IDataSourceDefinition and IDataSource and try using the Catalog.DataSources.OpenDataSource method to see if MapXtreme calls into your data provider and ends up with your data source object in the DataSources collection.

The first major milestone for the table model is to be able to open a table and display the table's metadata. You can develop and test this without writing any searching or cursor code which is far more complex. You will need to create an implementation of ITableDefinition, ITable, and ITableMetadata. Initially you can implement the search and modify interfaces to throw a NotImplementedException. In your test application you should then be able to create and populate an instance of your table definition class, instantiate a table by supplying this definition to a call to Catalog.OpenTable and then examining the column definitions that are supplied by the table's TableInfo property. The column definitions should accurately reflect the information your table metadata supplies to MapXtreme including the appropriate coordinate system, data bounds, and default view for your geometry column(s).

The next step to build out and test your provider is to add very simple search logic and a provide a cursor implementation. In your test application, you can add a simple block of code like this:

```
Table table = catalog.OpenTable(myTableDefinition, "MyTable");
foreach (Feature feature in table)
{
   // inspect the feature, write out values to the console, etc...
}
```

For this code to work, you will need to implement the ITable.SearchAll method and provide a cursor implementation. You will also need to provide a key implementation and the features returned in the code example above should reflect the appropriate value for your key (although values are serialized to strings).

Once you have this basic level of code working you can start to test your data provider in a map so that you can see visual results to see if the geometries are being returned as you expect, with the proper styles, etc. You will require some implementation of SearchByEnvelope before you do. You can ignore the areaOfInterest argument and return all records initially just to see how the data looks. Your performance will be poor since every request will retrieve all of the features. You should turn off the InfoTip property on your layer which will prevent MapXtreme from trying to display the InfoTip as you move your mouse over the map. The InfoTip calls down into your data provider with a small areaOfInterest around the point where the mouse is positioned and if you are returning all of the table features your performance will be really bad. You will also not be able to test selections or result set capabilities until you have a feature accessor implemented.

Before moving on, you should add support for the SearchByEnvelope method. This will help you get a feel for the performance of your data provider since you now have the ability to send back to MapXtreme only the features that it needs to render the map. You can also turn back on the InfoTip setting and set the InfoTip expression to various values to see if you are getting the right data passed up to MapXtreme. If your performance is unacceptable, this would be a good time to evaluate your design and tune it before adding more complexity.

Implementing the feature accessor interface would be the next major hurdle. The easiest way to test this interface is to create a resultset feature collection and then enumerate through the features. Internally, the resultset feature collection caches the keys for the records that satisfy the search criteria and then uses a feature accessor whenever the resultset is accessed.

Once you have gotten all of these pieces working you have done the largest portion of the hardest work for building your data provider. You will most likely want to then focus on building a persistence provider so that you can persist your definitions into a MapXtreme workspace file. This would be a good time to look into the new WorkspaceManager extensibility capabilities and consider adding a WorkspaceManager extension that

provides some user interface for defining a table and enables the hooks into the persistence provider extensions so you can set up and create a workspace and also read workspaces with your data provider persistence content.

To this point, you should focus on testing your data provider in a simple desktop application environment. Once you have persistence implemented, it would be a good time to start thinking about a web-based deployment. This will require serialization support so that the table or the entire catalog or session can be serialized across user requests.

Data Provider

SpatiaLite Sample Data Provider

Included in the Extensible Data Provider SDK is a sample implementation of a data provider that highlights many of the capabilities of the Extensible Data Provider architecture. MapInfo.SpatiaLite.sln is a Visual Studio solution that uses an SQLite3 Ado.Net data provider to establish a connection to an underlying SQLite3 database and open tables in a MapXtreme application.

This sample implementation also highlights other extensible aspects of MapXtreme, including autoloading custom code at session initialization, and extending the capabilities of Workspace Manager. This sample is located in the \Samples\DataProviders folder under your MapXtreme installation directory.

ⓘ Unlike MapXtreme's other sample applications that compile and provide a runnable application, this sample implementation yields .NET assemblies. The focus on the development for this sample was to showcase extensibility options in MapXtreme rather than a working application.

MapInfo.SpatiaLite.sln contains two projects:

- SpatiaLite Data Provider
- SpatiaLite Workspace Manager Extension

SpatiaLite Data Provider Extension

The code in this project shows how the MapXtreme Extensible Data Provider classes and interfaces have been extended to create a SpatiaLite data provider that allows you to open SpatiaLite tables, insert/update/delete records, search tables, changes the styles,

and import new tables into an SQLite3 database. Abstract base classes have been used to make coding more efficient. See the Readme.rtf that accompanies the sample for details on which classes and interfaces were extended.

At compile time, the SpatiaLite Data Provider project creates an assembly called MapInfo.SpatiaLite.DataProvider.SessionEventHandler.dll that serves as the main data provider implementation and also a session event handler that makes the data provider available for autoloading at MapXtreme startup.

The SpatiaLite data provider assembly also encompasses an implementation of a persistence provider, an optional component for a data provider, but a useful one. This allows SpatiaLite table and data source definitions that MapXtreme does not natively read/write to/from a MapInfo workspace (.MWS). See Persistence Providers.

The implementation of ISessionEventHandler in the SpatiaLite data provider is another useful component, also optional. Containing this in the data provider assembly allows the data provider to be autoloaded when MapXtreme starts up. This means that SpatiaLite tables definitions can be loaded and available in your MapXtreme application as soon as the MapXtreme session has initialized.

The SessionEventHandler is not specific to data provider extensibility. This can be used to load any custom code that you need at session startup. We took advantage of this capability to persist custom SpatiaLite table and data source definitions. In order to use this for other custom code, the assembly must contain the suffix SessionEventHandler.dll and put in the \Common Files\MapInfo\MapXtreme\9.x.x\SessionEventHandlers in order for MapXtreme to be aware of it. For more on this topic see ISessionEventHandlersSpatiaLite Workspace Manager Extension

Another major enhancement to MapXtreme is the ability to extend the functionality of Workspace Manager. An example of this is the SpatiaLite Workspace Manager Extension project contained in the SpatiaLite Data Provider sample. This project extends Workspace Manager to provide SpatiaLite menus, dialogs and code connecting to a database and opening a table.

When this project is compiled it generates an assembly with the suffix WorkspaceManagerExtension DLL. This assembly is dependent on the SpatiaLite data provider which provides the table definitions for the SpatiaLite database.

Workspace Manager extensions can contain any functionality you may need as you create and save workspaces. They are discussed in Appendix B: Customizing MapXtreme.

GeoJSON Data Provider

Another sample data provider implementation demonstrating the capabilities of Extensible Data Provider architecture is the GeoJSON Data Provider. As the name indicates, it extends MapXtreme to consume data which is in GeoJSON format.

Like the sample SpatialLite Data Provider implementation, GeoJSON Data Provider also highlights other extensible aspects of MapXtreme, including autoloading custom code at session initialization, and extending the capabilities of Workspace Manager. This sample is located in the \Samples\DataProviders folder under your MapXtreme installation directory.

ⓘ This sample implementation also produces .NET assemblies. The focus on the development for this sample was to showcase the extensibility options in MapXtreme rather than a working application.

MapInfo.GeoJson.sln contains two projects:

- GeoJson Data Provider
- GeoJson Workspace Manager Extension

The GeoJson Data Provider project demonstrates how the MapXtreme Extensible Data Provider classes and interfaces have been extended to create a GeoJSON data provider that allows you to open GeoJSON data and insert/update/delete features. See the Readme.rtf that accompanies the sample for details about the classes and interfaces that were extended.

When successfully compiled, this project creates an assembly called MapInfo.GeoJson.DataProvider.SessionEventHandler.dll that serves as the main data provider implementation and also a session event handler that makes the data provider available for autoloading at MapXtreme startup. This assembly also encompasses an implementation of a persistence provider, which allows GeoJSON table definitions that MapXtreme does not natively reads from or writes to a MapInfo workspace (.MWS).

The implementation of ISessionEventHandler in the GeoJSON data provider assembly allows it to be autoloaded during start up of a MapXtreme application. This means that GeoJSON table definitions can be loaded and made available in your MapXtreme application as soon as the MapXtreme session is initialized. To make it happen, the assembly must contain the suffix SessionEventHandler.dll and must be present in the \Common Files\MapInfo\MapXtreme\9.x.x\SessionEventHandlers folder in order for MapXtreme to be aware of it.

Similar to SpatiaLite Workspace Manager Extension, the GeoJSON Workspace Manager Extension project present in this sample extends Workspace Manager to provide an open GeoJSON table dialog. When successfully compiled, this project generates an assembly with a suffix WorkspaceManagerExtension DLL, which is dependent on the GeoJSON data provider for table definitions for the GeoJSON data.

Advanced Topics / Important Considerations

This section includes a variety of topics worth considering as you design and build your extensible data provider:

- Creating Geometries
- Coordinate Systems
- Styles
- Exception Handling
- Persistence Providers
- Serialization
- Authentication
- Thread safety

Creating Geometries

Extensible data providers implementing support for 3rd party spatial formats must convert spatial data between their format and the MapXtreme FeatureGeometry format. FeatureGeometry objects are returned as values of the IFeature objects obtained from cursors and feature accessors. The features and the geometry objects they contain are transient. MapXtreme will only assume that these objects are valid while the cursor is open and positioned on the current feature. As a result, the most robust and scalable data provider implementations will try to re-use the same feature and feature geometry objects for the life of the cursor or feature accessor.

A geometry that is created once and continually updated for the current feature is referred to as a “transient” geometry. MapXtreme’s geometry model contains constructor and method signatures that facilitate geometries being used in this way. Following the initial construction of MultiPoint, MultiCurve, and MultiPolygon objects, geometry editor interfaces can be used on these existing instances to make the changes for the new feature. For MultiCurve and MultiPoygon objects, the Clear method can be used to empty the geometry, and then AddCurve and AddPolygon can be used respectively to redefine the geometries. For MultiPoint objects, the ReplaceAll method can be used for

redefinition. Furthermore, the constructor signatures and Add/Replace methods that use input arrays support an optional size designation enabling single array instances to be efficiently re-used as well.

The OGC conversion code works in this manner. The internal byte arrays used for parsing the well known binary and well known text are reused and grown as needed. When data is supplied to the appropriate geometry methods (like AddPolygon) the forms that accept an array size are used since the input arrays may be longer than the data being supplied.

MultiPolygon objects composed of multiple polygons are generally checked by MapXtreme to determine if any of the polygons have interior/exterior relationships to one another; for example, a doughnut shaped MultiPolygon instance might be comprised of an exterior polygon with a second interior polygon interpreted as a cutter such that the two together logically represent a single geometry. The geometry code does not make any assumptions by default, and when editing is complete it will analyze all of the constituent polygons searching for the existence of these relationships. If the 3rd party spatial format being converted already knows these relationships and can add the polygons in the correct sequence (exteriors followed immediately by any of its associated interiors), there's no need for MapXtreme to incur the potentially expensive cost of re-analyzing all of the polygons. This operation can be suppressed using the overloaded EditingComplete method on the MultiPolygon class and passing "true" to the withinSpecified argument.

This processing time can be expensive. It is necessary for interactive editing operations where moving a single node may make an interior polygon suddenly become an exterior polygon. Data providers typically already know that their data has been formatted and stored in a way that does not require this expensive processing every time a geometry is built.

For large numbers of geometries and/or complex geometries with large numbers of nodes, understanding and exploiting these geometry options will be critical for implementing a performant data provider.

Coordinate Systems

Geometry columns in MapXtreme must return FeatureGeometry objects in the same coordinate system. Different column and different tables may use different coordinate systems but all geometry values returned from a single column in a table must be in the same coordinate system and must match the coordinate system that is supplied by the table's metadata through the IGeometryColumn.CoordSys property.

Most data providers will determine the coordinate system through some metadata that is stored and managed with the data. However, if that information is not available, the data provider can request that the suer supply the coordinate system as part of the table's definition.

Coordinate systems can be constructed through the CoordSysFactory instance which is available fro the Session.CoordSysFactory property. There is no need to create new coordinate system objects for every FeatureGeometry object returned by the data provider. The coordinate system object returned by the IGeometryColumn interface can be referenced directly by each FeatureGeometry object. FeatureGeometry objects supplied to the data provider for insert and update operations will be transformed to the column's coordinate system by MapXtreme before handing to the data provider.

Styles

MapXtreme's Extensible Data Provider model supports styles for feature geometries in two ways:

- as a style property that is applied to all features in the table
- as a style attribute column, which contains style information on a per-feature basis

The MapInfo.Data.Table that is opened from an extensible data provider table will have a style column, provided there is at least one geometry column (mappable table). The data for that column comes from either a style column in the extensible data provider table, or if there is not one, from the default style specified on the IGeometryColumn.DefaultStyle property of the geometry column in the data provider table.

ITableMetadata.Columns

Style information is communicated via the column definitions provided in the MapInfo.Data.Provider.ITableMetadata interface. A style attribute column is an IColumn instance from ITableMetadata.Columns whose DataType property is assigned the value MIDbType.Style. A style attribute column is only necessary when two or more features within the table may contain distinct style values.

IGeometryColumn.DefaultStyle

A style attribute column is not required when you want to apply a single style uniformly when rendering the objects within one of the table’s feature geometry columns. For this, a style object instance can be directly managed as the DefaultStyle property of the feature geometry column (IGeometryColumn.DefaultStyle).

Providing a default style is always recommended, even when a style column exists. The default style is used to fill in missing style attribute values if the style column contains null or missing values. In the absence of an explicitly provided default style, MapXtreme does have an internal default style it can apply as needed.

If using only a default style on the feature geometry column, a style column will still be shown in the columns collection associated with the resulting Table instance opened within the Catalog. This column will be nullable and read-only, and all values are defaulted to the specified default style instance value.

While no explicit proscription is made against defining multiple geometry column and/or multiple style column data configurations within your extensible data provider implementation, the metadata model does not support the explicit association of a style column to a geometry column. When rendering a map layer, there's an implicit assumption that the table servicing that layer contains a single geometry column with at most one adjacent style column. Similarly, where the model also permits extensible data providers to define tables containing style columns without an adjacent geometry column, these tables cannot currently be cached, and may not be exportable to other formats.

Exception Handling

Exception handling in the data provider is very important. The data provider will usually be handling system resources such as file handles or database connections. If these resources are not carefully cleaned up in normal and exception code paths, they may be leaked giving applications unusual behaviors or memory leaks that build up over time. These problems are often accentuated in web applications that utilize pooled sessions in which memory leaks may eventually cause the system to shut down or multiple processes to hang waiting for leaked connections to be released.

Exceptions thrown should follow standard .NET practices as prescribed by Microsoft in Best Practices for Handling Exceptions. When throwing a custom exception, use the MapInfo.Data.Provider.DataProviderException class. This class may be subclassed to provide additional behavior if necessary.

Exception handling is also a good time to think about externalizing the resources for your data provider. By properly capturing resources such as strings, bitmaps, etc. in resource files, your data provider can be localized for other cultures. The SpatiaLite Data Provider sample provides a reference implementation for handling resource strings for exception handling (refer to the Resources.cs file in the project).

Persistence Providers

MapXtreme offers support for saving data access information, namely ITableDefinitions and IDataSourceDefinitions, to a MapXtreme XML-based workspace (.MWS). This is an optional component of the Extensible Data Provider that you may wish to take advantage of if you need to share your workspace with others or to simply re-use the information at a later time.

To add persistence support to your Extensible Data Provider, in its simplest form, you would write a persistence provider class that implements the MapInfo.Data.Provider.IMxpPersistenceProvider interface.

For a more sophisticated persistence provider, consider providing a schema to support XML validation of the resulting workspace file. This would only be necessary if you have an explicit requirement, as MapXtreme does not automatically validate workspace XML.

You may also consider writing your persistence provider as its own assembly. Providers may be pre-loaded independent of the remaining extensible data provider components. A provider assembly would defer the loading of the remaining components by controlling just-in-time loading of the data provider when and if it is explicitly needed.

How Are Persistence Providers Used?

Persistence providers are managed in a PersistenceProviderCollection on the Catalog. This collection may be automatically initialized during session startup when MapXtreme searches the assemblies in a default location for persistence providers they might contain. The default location is \Program Files\Common Files\MapInfo\MapXtreme\9.x.x\SessionEventHandlers. Providers can also be programmatically added into the PersistenceProviderCollection using its Add and AddFromFolder methods.

When saving session context to a MapInfo workspace file (.MWS) via the MapInfo.Persistence.WorkspacePersistence class, the PersistenceProviderCollection is consulted for any tables in the catalog that were opened from an Extensible Data Provider. If a persistence provider is identified for that data provider, it is used to persist the extensible table information into the workspace. Similarly, when a workspace is being loaded, the collection of available persistence providers is searched to identify a data provider capable of understanding the custom tags in the workspace file associated with an Extensible Data Provider table.

Implementing Your Persistence Provider

In order for the PersistenceProviderCollection.AddFromFolder method to load your persistence provider class, it must contain a public, zero-argument constructor signature. This is the same method used internally during MapXtreme's session initialization. Additional constructor signatures can be supplied to support programmatic use with the PersistenceProviderCollection.Add method.

MapXtreme provides an abstract base class named AbstractMxpPersistenceProvider you may wish to use as a starting point for implementing your persistence provider class.

One of the things you must provide in your persistence provider are methods that check whether there is a supported data provider for your persistence provider, a supported entity name (for the XML tags) and a supported schema namespace. This is done by implementing the three inquiry methods included in the IMxpPersistenceProvider interface.

SupportsDataProvider() is used to identify providers responsible for ITableDefinition and IDataSourceDefinition constructs for a given IDataProvider. Generally, persistence providers are written to support a single data provider, so the implementation for this method can often be as simple as determining if the provided IDataProvider is an instance of your data provider class.

The SupportsSchemaNamespace and SupportsEntityName work in tandem during workspace depersistence to identify the persistence provider supporting a namespace and tag name identified for extensible data provider content. For example, when reading through the workspace file, the following content is encountered for an extensible data provider:

```
<sample:SampleTableDefinition xmlns:sample="http://sample/sample">
  <sample:TableName>test</sample:TableName>
</sample:SampleTableDefinition>
```

An XMLNode object reference is obtained for the outer tag. The NamespaceURI property ("sample") is provided to SupportsSchemaNamespace, and the LocalName property ("SampleTableDefinition") is provided to SupportsEntityName to identify a persistence provider supporting both. The constructor signature available through the abstract class also supports default implementations of these methods.

The AbstractMxpPersistenceProvider contains a constructor signature that works in conjunction with the default implementation of IMxpPersistenceProvider.Schema to return an XMLSchema reference to an .xsd schema file attached as an embedded resource within your assembly. See the Developer Reference for details.

The remaining methods on IMxpPersistenceProvider are the read/write pairings for data source and table definitions. For data providers that do not use data sources, only the read/write methods for the table definitions require implementation. If data sources are applicable, the data source definitions would need to be persisted. MapXtreme handles that for you. You can focus exclusively on persisting only those table definition properties that are unique to your table.

The read methods contain an XmlElement handle to the XML tag for the extensible content. The components of the data source and table definitions are managed as child nodes within that element. Following the SampleTableDefinition example above, the implementation code might resemble the following:

```
ITableDefinition td = null;
if (node.LocalName.Equals("SampleTableDefinition") &&
node.NamespaceURI("http://sample/sample"))
{
   string tableName = null;
   foreach (XmlNode childNode in node.ChildNodes)
   {
if (childNode.LocalName.Equals("TableName") &&
childNode.NamespaceURI.Equals("http://sample/sample"))
      {
         tableName = childNode.InnerText;
      }
   }
   td = new SampleTableDefinition(tableName);
}
```

The write methods must create an XmlElement that is inserted into the workspace XML. The elements are bound to an XML document instance, so the XmlDocument is provided along with the extensible data provider definition interface to be persisted. Note, for the DataSourceDefinition write method, the XmlDocument is provided directly as a method argument, whereas, for the TableDefinition write method, the XmlDocument is provided via the Document member of the IMxpPersistenceServices argument.

```
XmlElement outerTag = xmlDocument.CreateElement("sample",
"SampleTableDefinition", "http://sample/sample");
XmlElement innerTag = xmlDocument.CreateElement("sample", "TableName",
"http://sample/sample");
innerTag.InnerText = "test";
outerTag.AppendChild(elNode);
```

For additional information and examples, consult the Developer Reference Guide documentation, the SpatiaLite Sample Data Provider, and other data provider examples located on the MapXtreme Code Exchange.

Serialization

Serialization is the process of converting an object into a stream of data in order to preserve it in a permanent form or in memory for the duration of its usefulness. This process is an essential part of maintaining objects in MapXtreme web applications and multi-threaded desktop applications.

Without serialization, objects would need to be recreated, for example, every time there was a web request for that object during a session.

When a serialized object is requested, it is deserialized (or recreated from the stream of data) and then modified. MapXtreme's serialization algorithm does not make a copy of the object (as other serialization algorithms do) such that the object being deserialized is created only once.

For proper state management of web applications and multi-threaded desktop applications, application developers often need to serialize MapXtreme Table instances directly, or via serialization of the MapXtreme Catalog which contains the collection of all tables. When a table type is not supported, it places a burden on the application developer to figure out how to explicitly manage those tables and the overall state of their Catalog to operate around this limitation. For that reason, Extensible Data Provider developers are encouraged to support serialization of their provider to properly integrate into MapXtreme's Table serialization workflow.

What components of my provider implementation do I have to serialize?

MapXtreme distinguishes tables as being either permanent or temporary. Permanent tables only have to serialize their definition/structure in order to be properly re-constructed upon deserialization. Temporary tables have the additional responsibility of serializing their data. Extensible Data Provider tables are considered permanent tables thereby simplifying the overall serialization responsibilities, although several provider classes must still be serialized.

For those providers supporting the concept of data sources, the classes implementing the IDataSourceDefinition and IDataSource interfaces must both be serializable. Whether or not data sources are supported, every provider must also support serialization of the classes implementing the ITableDefinition and ITable interfaces. The ITableDefinition and IDataSourceDefinition interfaces must also provide meaningful overrides to the Equals method in order for deserialization to work properly.

The classes implementing ITableMetadata are not required to be explicitly serializable, although there are practical reasons to do so. If not serializable, the deserialization of their respective table and data source instances may need to reconstruct them, and the re-acquisition of that metadata could be potentially expensive in terms of performance.

To assist you, the Extensible Data Provider API provides serializable abstract base classes provided for each of these interfaces. Serialization support is also provided for the relevant properties being managed by their default implementations.

How do I serialize a class?

In .NET terms, a class is made serializable through some combination of implementing the ISerializable interface from the System.Runtime.Serialization namespace and/or applying the [Serializable] attribute to the class definition. Provider developers are free to follow general guidance regarding .NET serialization in providing your implementation; however, we strongly recommend applying both – particularly if deriving your classes from the abstract base classes provided.

What if I decide not to support serialization?

We recommend that you provide serialization support in your extensible data provider so you can provide better and broader support to application developers who will be employing your provider as a component of their solution. However, there is no strict requirement that a provider implementation must support serialization. In fact, in some cases there are legitimate reasons why providing this support is fundamentally difficult or unreliable.

Application developers may be able to use .NET reflection APIs to discover whether or not your provider appears to support serialization; however, this still may not provide them with everything they need to know regarding how to properly employ your provider in an application with state management requirements. We strongly encourage you to provide documentation that includes specific information regarding if/how your provider implementation can be used as a component for such solutions.

Serialization and Dependent Relationships

MapXtreme's Extensible Data Provider includes many dependent relationships that exist amongst the required and recommended classes. An ITableDefinition contains an IDataSourceDefinition property. An IDataSource also contains an IDataSourceDefinition and an ITable contains IDataSource, ITableMetadata, and ITableDefinition properties.

As a rule, you may not need to serialize any property whose references may be shared. For example, the IDataSourceDefinition property on ITableDefinition may represent a data source definition that gets re-used to open multiple tables. You do not need to include it in the serialization of the class implementing ITableDefinition. The same is true for the IDataSource and ITableDefinition properties within ITable.

We do recommend including the definition properties when serializing – even if they are later overwritten to use a shared reference.

The IDataSource reference within the ITable should not be serialized as MapXtreme will take care of automatically serializing the datasource for the table.

Shared object references are re-established via the serialization logic provided by MapXtreme. The extensible data provider API contains the hooks necessary for the core MapXtreme data access engine to accomplish this work. Specific examples include:

- The ReAssociate method on the IDataProvider interface
- The settable DataSourceDefinition property on the ITableDefinition interface

Implementing Serialization

Classes are serialized by being tagged with the [Serializable] attribute or by implementing the System.Runtime.Serialization.ISerializable interface. We recommend you do both. The ISerializable.GetObjectData method must be implemented and it must be public:

```
public void GetObjectData(SerializationInfo info, StreamingContext context)
```

For classes derived from the abstract base classes, this signature should also contain the override keyword, and the first line of the implementation should delegate to the base class for serialization of the components it is managing. For example:

```
base.GetObjectData(info, context);
```

The remaining serializable members within the instance are serialized into the SerializationInfo argument named info by providing a string key and the member value to its AddValue method. For example:

```
info.AddValue("TableName", _tableName);
```

Implementing Deserialization

To support deserialization, provide a protected deserialization constructor whose arguments match those on GetObjectData above. For example:

```
protected COTWTableDefinition(SerializationInfo info, StreamingContext ctxt)
```

For classes derived from the abstract base classes, this constructor should delegate to the base class for deserialization of the components it is managing. For example:

```
protected COTWTableDefinition(SerializationInfo info, StreamingContext ctxt) :
base(info, ctxt)
```

Within this constructor, the class instance assigns its member variables by retrieving values using the “get” methods available on the SerializationInfo argument named info. For example, to deserialize the TableName value serialized in the GetObjectData example above, the code might resemble the following:

```
_tableName = info.GetString("TableName");
```

There are different recommendations regarding override support, method attributes, etc., for methods pertaining to serialization depending, at least in part, upon whether or not the implementing classes are sealed. Using a code analysis tool like FxCop can provide valuable assistance in providing proper recommendations.

For additional information and examples, consult the MapXtreme Developer Reference (online Help), the SpatiaLite Sample Data Provider presented in this appendix, and other data provider examples located on the MapXtreme Code Exchange.

Authentication

Many Data Providers require authentication when opening a Data Source or Table where the details of the authentication are kept secure and unavailable from a publicly accessible and shared workspace file.

To support run-time authentication of the OpenDataSource and OpenTable processing, the MapXtreme Extensible Data Provider model provides a new interface called the IDataProviderCallback with support for user-defined callback methods that the Extensible Data Provider can use to resolve data source and table definitions that are insufficient in some way that prevents the opening of a data source or table.

For the simplest implementation, the client code for a custom Data Provider would contain a class that implements the IDataProviderCallback interface and provides implementations of the IDataSourceDefinition and/or ITableDefinition callback methods that are then used directly to open the data source or table.

For most users, however, it will be necessary to load a separate assembly containing the IDataProviderCallback initialization and load the assembly as part of the MapXtreme Session initialization. See ISessionEventHandlers. This would be required when you are attempting to resolve data source or table definitions at the time a default workspace is loading. Web-based applications, for example, will require the session initialization support for callbacks.

How Are Data Provider Callbacks Used?

How the callback methods are invoked is handled by your Data Provider implementation; MapXtreme manages support for the collection of callback methods and passes it to the Data Provider via the defined interfaces,. It does not invoke any of the IDataProviderCallbacks.

The IDataProviderCallback instances are managed by a DataProviderCallbacksCollection on the MapXtreme Session Catalog. This collection of callback collections associates callback objects to a System.Type, and supports multiple callbacks per each Type association. The DataProviderCallbacksCollection may be

managed in several ways: during Session initialization (see SessionEventHandler/DataProvider load section); during application initialization, and, for very simple usage, immediately prior to an explicit OpenTable call from the client application.

MapXtreme queries the Session Catalog for the callbacks associated with the OpenDataSource or OpenTable method, and provides a reference to the collection to the DataProvider. The IDataProvider interface supports a reference to an IEnumerator that iterates over a collection of IDataProviderCallback[s] during the OpenDataSource and OpenTable operations. The DataProvider class may use any callback referenced by the IEnumerator to implement the appropriate logic required for resolving an OpenDataSource or OpenTable failure, if the failure is addressable by modifying the input DataSourceDefinition or TableDefinition. Or the DataProvider may implement a well-known strategy for completing a definition object which is insufficient by design.

A typical example of a callback is one which, in a desktop application environment, the OpenDataSource callback resolves a credential authentication error by presenting the user with a dialog, requesting a valid username/password combination for the data provider.

Implementing Your Data Provider Callback

The IDataProviderCallback interface specifies method signatures for modifying data both of the IDataSourceDefinition and ITableDefinition data provider definition types. Classes implementing IDataProviderCallback must provide appropriate implementations of the methods to provide run-time updates of the definition objects in order to fulfill the OpenDataSource/OpenTable requirements.

Additionally, an IDataProviderCallbackInfo interface can be specified on the callback method in order to provide additional runtime state information to the callback in order to determine an appropriate course of action. The IDataProviderCallbackInfo is a marker interface only, and does not specify the nature of the information provided to the callback. Typical implementations could include a run-time Exception or state enumeration attribute.

The MapInfo.Data.Provider namespace contains a canonical implementation that may be used by Data Provider implementors and/or client applications. The DataProviderCallback class implements a delegate-based callback harness, allowing client code to assign a callback method outside of the Extensible Data Provider assembly. The DataProviderCallbackExceptionInfo class implements IDataProviderCallbackInfo as a carrier for a System.Exception reference, while the DataProviderCallbackCollection implements an ICollection of IDataProviderCallback[s].

The following is an example of how a credential resolution callback scenario can be implemented.

```
// Create a method to handle the OpenDataSource InvalidCredentials state
// Implements IDataProviderCallback.Callback signature
public static IDataSourceDefinition InvalidCredentialsCb(
IDataSourceDefinition dsd, IDataProviderCallbackInfo info)
{
  // Implementation details are specific to the Data Provider, specifically, to
the appropriate IDataSourceDefinition implementation
  DataProviderCallbackExceptionInfo pcinfo
   = info as DataProviderCallbackExceptionInfo;
  if (pcinfo != null)
  {
    if (pcinfo.Exception == /* invalid credentials */)
    {
   // Example: present user dialog
        string newpwd = GetPasswordFromUser();

   // Example: create new EDP implementation specific DataSourceDefinition
        //  with updated password credentials
   return new EDPDataSourceDefinition(newpwd);
    }
  }
}

// MXT/EDP client code
{
  // Create a callback harness and assign the callback method
  DataProviderCallback cb = new DataProviderCallback();
  cb.DataSourceDefinitionCallback = InvalidCredentialsCb;

  // Associate the IDataProviderCallback with the EDP definition data type

Session.Current.Catalog.DataProviderCallbacksCollection.AddProviderCallback(
typeof(EDPDataSourceDefinition), cb);

  // Create an initially invalid Data Source Definition
  EDPDataSourceDefinition dsd = new EDPDataSourceDefinition("badpassword");

  // open table
  EDPTableDefinition tableDef = new EDPTableDefinition (dsd);
  Table t = Session.Current.Catalog.OpenTable(tableDef, "TESTTABLE");
}
```

Implementing IDataProviderCallback usage in the Data Provider

The OpenDataSource and OpenTable methods on IDataProvider include an enumerator of IDataProviderCallback instances. This enumerator is supplied by the Catalog automatically.

The DataProvider implementation is responsible for iterating over and invoking the callbacks if necessary to attempt to complete the OpenDataSource|OpenTable operation.

An example DataProvider implementation might proceed along the following lines, to ensure that a valid open data source is constructed:

```
public override IDataSource OpenDataSource(
IDataSourceDefinition definition,
CustomProperties customProperties,
IEnumerator<IDataProviderCallback> callbacks)
{
  EDPDataSourceDefinition dsDef = definition as EDPDataSourceDefinition;
  if (dsDef == null)
  {
     // invalid definition
     // throw exception
  }

  EDPDataSourceDefinition tmpDsDef = dsDef;
  EDPDataSource ds = null;
  bool needDS = true;
  while (needDS)
  {
    needDS = false;
    DataProviderCallbackExceptionInfo cbinfo = null;
    try
    {
       // Data Source ctor will fail if data source definition is invalid, e.g.,
invalid credentials
   ds = new EDPDataSource (tmpDsDef, customProperties);
    }
    catch (Exception e)
    {
   cbinfo = new DataProviderCallbackExceptionInfo(e);
    }
    if (ds == null
   && callbacks != null)
    {
   while (!needDS
      && callbacks.MoveNext())
       {
          // callback will return null if it is unable to modify the
          //   the definition appropriate for a retry attempt
          //   Example:  User selected cancel from a credential input dialog
          tmpDsDef = callbacks.Current.Callback(dsDef, cbinfo) as
EDPDataSourceDefinition;
      if (tmpDsDef != null)
      {
      needDS = true;
      }
   }  // while (!needDS and callbacks.MoveNext())
```

```
    } // if (ds == null && callbacks != null)

    if (!needDS
    && cbinfo != null)
    {
   throw cbinfo.Exception;
    }

  } // while(needDS)

  return ds;
}
```

Thread safety

MapXtreme is thread safe meaning that different threads may concurrently be accessing different sessions, catalogs, tables, maps, etc. without any unintended side-effects. MapXtreme objects are not multi-threaded meaning that the same instance of a map, table, catalog, etc. cannot be accessed from multiple threads at the same time. Data provider implementations must at least follow this model if there is ever an expectation that the data provider will be deployed in a web environment.

Generally your code should be thread safe as long as you do not rely on any global variables, singleton objects, etc. You should also not rely on use of Thread Local Storage since ASP.NET may actually cause the executing thread of your request to change. If your data provider follows these guidelines, you should not require much, if any, synchronization locks throughout your code which may choke the scalability of the solution. The actual data that you are accessing, however, may have implied synchronization that is either unavoidable or preferred. For example, if your data provider exposes data from an Excel spreadsheet, the spreadsheet may be locked from editing operations while it is open for read operations in a different thread (or different process altogether). For these scenarios, it is important to consider how you architect your data provider. Cursors, feature accessors, and modify processors are all intended to imply a certain type of locking to the data where that may be required. However, if there are no active cursors, accessors, or modify processors, the underlying data files (if applicable) should not be held open.

The IDataProvider is the one exception where we actually recommend that you implement this as a singleton. This interface is intended to be a factory interface and should not contain any state at all (meaning no class member variables other than the static INSTANCE property). The definition objects must reference their data provider for MapXtreme to be able to call into it to open a data source or a table. For this mechanism, an instantiable class is required although a singleton is perfectly suitable.

You may be inclined to want to cache connections and possibly even cache your datasource or table objects and reuse them when calls to OpenTable or OpenDataSource. We strongly recommend that you do not do this. MapXtreme provides collections for data sources and tables on the Catalog and will rely on the Equals logic that you supply in your ITableDefinition and IDataSourceDefinition classes to find and reuse the correct instances. This will prevent you from running into cross-thread, multi-threaded situations. Your underlying data access technology, such as an ADO.NET data provider, may cache and reuse data base connections. This is an acceptable architecture since they assure the proper behavior in a multi-threaded environment.

E

E – Printing From MapXtreme Applications

This appendix will guide you in printing the best possible map images from your MapXtreme development project. We begin by giving you an overview of printing functionality and some helpful tips and tricks, then we help you troubleshoot issues you may be experiencing printing with your MapXtreme application.

In this appendix:

Overview

Printing from MapXtreme-developed applications can usually be straight forward. However as the variety of printing devices continue to expand, device-specific problems do occur. We provide a variety of features to you, the developer, to customize your user's printing experience. These options are designed to optimize printing depending on the map being printed and the device being used. These different settings are designed to meet the needs and nuances of a variety of printers and plotters.

MapXtreme supports:

- Printing maps – Use the MapPrintDocument class.
- Printing legends – Use the LegendPrintDocument class.
- Printing either directly to a device or by using an Enhanced Metafile (EMF).
- Printing maps in different sizes
- Printing maps that have translucent style colors and layers

  See GDI+ Translucency and Anti-Aliasing for information.

- Printing translucent raster images

  EnableTranslucency must be enabled to print translucent raster images.

MapXtreme does not support:

- The use of a printing options dialog box.
- Printing of layouts.

  Layouts are important for the printing of legends in a map. See Printing a Legend in Your Map for an example of how to get around this limitation.

- Printing multiple page maps.

  If a map does span across multiple pages, only the first page is printed.

- Printing to a file programmatically.

  You would need to use the Microsoft.NET 2.0 Framework System.Printing.PrinterSettings class to implement this functionality.

The classes that handle printing in MapXtreme applications are derived from the Microsoft.NET 2.0 Framework System.Drawing.PrintDocument class and inherit that functionality. Device output control is managed by the Microsoft.NET 2.0 Framework System.Printing.PrinterSettings and System.Printing.PageSettings classes. The MapInfo.Printing.MapPrintDocument class is specifically for printing maps, while the MapInfo.Printing.LegendPrintDocument is used specifically for printing legends.

Understanding the Print Options in MapXtreme

When printing a map or a legend from a MapXtreme application you can provide the following options for printing to your user. In order to give your users control over these settings, you need to build this functionality into the printing dialog boxes of your application. If you do not specifically allow them to change these options the default settings are used. You can also adjust these settings programmatically.

Printing Sizes

MapXtreme provides the options to print your map in different sizes. To alter the size in which maps are printed set the MapPrintDocument.MapPrintSize property to one of the values in the MapPrintSize enumeration. The values are as follows:

Fit to Page

This option is the default and the resulting map is printed with its aspect ratio maintained, but scaled to fit on the page.

```
MapPrintDocument.MapPrintSize=MapPrintSize.FitPage
```

Fill Page

This option prints the map so that it fills the page. This method does not maintain the aspect ratio and the resultant map may be skewed.

```
MapPrintDocument.MapPrintSize=MapPrintSize.FillPage
```

Current Map Size

This option prints the map in its original size. This may be fine, however if the map is larger than the page on which it is printed, the overflow is lost.

```
MapPrintDocument.MapPrintSize=MapPrintSize.MapSize
```

ⓘ The following properties are common to both the MapInfo.Printing.MapPrintDocument and MapInfo.Printing.LegendPrintDocument classes.

Special Transparent Raster Handling

This sets your application to internally manage the transparent pixel display and printing for raster images. On screen, the transparent image is rendered using a raster operation (ROP) to handle the transparent pixels. This method may or may not work when printing since MapXtreme uses a different method to decide the transparent method for printing. You must determine if your particular print driver handles ROP correctly, and set it to `true` or `false` accordingly. Under most printing conditions, this value should be set to `true`.

When drawing to the screen, this value is normally set to `false`. Use `DrawingAttributes.SpecialTransparentRasterHandling` to set this value. Setting this value to `true` (the default) allows the application to internally manage the printing.

ⓘ This setting has no effect when printing vector layers.

Special Transparent Vector Handling

This sets your application to internally manage the transparent vector fill patterns when you print. Use `DrawingAttributes.SpecialTransparentVectorHandling` to enable or disable this functionality. Set this to `true` to allow MapXtreme to perform special handling when printing transparent fill patterns or transparent bitmap symbols. This setting is normally used for printing. Set this to `false` to let the printing device handle how it prints transparent fill patterns and transparent bitmap symbols. See Need for Speed When Using Fill Patterns? for details about which fill patterns are bitmaps and which are vectors.

Display Raster in True Color When Possible

Some printers do not support 24-bit (true color) images. MapXtreme has an internal functionality that accommodates this. Use `DrawingAttributes.TrueColorRaster` to enable or disable this functionality.

Set this to `true` (default) to use the 24-bit (true color) to print raster and grid images. This is possible when the image is 24-bit and the printer supports more than 256 colors (8 bit). If your printer does not support 24-bit images, set this attribute as `false` to render the raster image using only 256 colors with dither specified in DitherMethod. When printing a 24-bit image with this value set as `false`, the resulting printed image will be quite deteriorated. when this attribute is set as `false`, you will need to set the dither method as well (see below).

GDI+ Translucency and Anti-Aliasing

GDI+ rendering in MapXtreme allows you to create translucent lines, labels, and layers, as well apply anti-aliasing that will smooth the jagged edges of lines, curves, and the edges of filled areas when representing a high-definition rendition at a lower resolution.

When printing a map or a legend using MapPrintDocument or LegendPrintDocument, the DrawingAttributes.EnableTranslucency and DrawingAttributes.SmoothingMode properties are automatically set to the values from Map.DrawingAttributes or

Legend.DrawingAttributes respectively. After initialization, the user can override the properties by setting them with their own value. This allows these two display options to carry over from the map or legend without having to explicitly set them.

EnableTranslucency: Obtains or sets whether to honor translucent values in style colors and/or layers when drawing the map onto the screen, printer or file export. When this property is false, translucency values in styles (e.g., Color.A) and layers (e.g., MapLayer.Alpha) are ignored. A value of 255 is used instead. This property has no effect on raster translucency (e.g., RasterStyle.Alpha; however, EnableTranslucency must be enabled to print translucent raster images. This is not required for screen display or exporting. If you set this property to false, the DrawingAttributes.SmoothingMode property is automatically set to SmoothingMode.None. In other words, if translucency not enabled, anti-aliasing cannot be used.

SmoothingMode: Obtains or sets the rendering quality of the map. This property specifies a member of the MapInfo.Mapping.SmoothingMode enumeration. The default value is SmoothingMode.None. The smoothing mode specifies whether lines, curves, and the edges of filled areas use smoothing (also called anti-aliasing). If you set this property to SmoothingMode.AntiAlias, the MapInfo.Mapping.DrawingAttributes.EnableTranslucency property is automatically set to true. In other words, translucency is always honored when anti-aliasing is used.

If you set EnableTranslucency to false, SmoothingMode will automatically be set to None if it is not already. If you set SmoothingMode to AntiAlias, EnableTranslucency will automatically be set to true if not already. While EnableTranslucency is true, you can switch SmoothingMode between None and AntiAlias without losing translucency.

MapStyleControl Class

The MapStyleControl (MapInfo.Windows.Controls namespace) can be displayed on a tab in the desktop LayerControl to allow the user to set map style and rendering options, such as anti-aliasing and translucency.

Dither Method

Dithering is a technique that blends pixels electronically to maintain the look of an image when decreasing the color depth. Select a dither method when you are converting a 24-bit image to 256 colors.

- **Halftone dithering** calculates a series of half tone differences in color between high-contrast elements in your image to create a smooth transition of color. This option is selected by default for display, print, and export options.

- **Error diffusion dithering** calculates an interim color between contrasting colors and shades the surrounding pixels to blend evenly toward that interim color.

Set this value by using the `DrawingAttributes.RasterDitherMethod` property. You can set this to a value from the DitherMethod enumeration.

Example:

```
mapPrintDocument.DrawingAttributes.RasterDitherMethod =
MapInfo.Mapping.DitherMethod.HalfTone;
```

Special Polygon Hole Handling

MapXtreme applications can draw complicated Polygon objects by using several separate pieces and then merging them together. When a polygon is printed with holes or islands, the pieces may not match well, and can overlap or leave gaps. There is some internal programming that accommodates this errant behavior to make sure that the polygons are displayed correctly and print properly. To take advantage of this functionality set the value of `DrawingAttributes.SpecialPolygonHoleHandling` property to `true`. When drawing to the screen, this value should be set to `false`.

Scale Patterns

This setting either prints non-transparent bitmap fill patterns that look like what you see on your screen or allows the printer driver to have exclusive control over rendering the fill patterns. Set this value with `DrawingAttributes.ScaleBitmapPatterns`. If this value is `true`, the bitmap fill patterns are scaled up to compensate for the difference in printer and screen resolutions. `False` does not scale the pattern and relies on the printer driver for scaling the fill pattern. If the printer driver can scale fill patterns, you should set this value to `false`. If the printer driver cannot scale fill patterns, you should set this value to `true` in order to prevent the printed output from displaying too small and appearing like a solid fill. See Need for Speed When Using Fill Patterns? for details about which fill patterns are bitmaps and which are vectors.

If this property is set to `true`, the print preview and the resulting printed document may not look the same, as the printer driver scales the pattern, but the display does not scale the pattern. It will print correctly, even if the display doesn't show it properly.

ⓘ If a fill pattern has a background color it is considered non-transparent, while one without a background color is considered a transparent fill and is always scaled, regardless of the value of this property.

Print Directly to Device

This option allows you to print your image directly to your printer. Set this value with `PrintMethod.Direct`.

Print Using Enhanced Metafile (EMF)

Use this option to generate an enhanced metafile of your image before sending it to the printer. This option takes advantage of current printer technology to shrink the spool size and print your file quicker without sacrificing quality. Set this value with `PrintMethod.Emf`.

Implementing Printing in Your Application

The printing namespace in MapXtreme contains a set of classes to help you print maps. Use these classes to handle basic printing of maps to any connected printing device.

Use the MapPrinting class to access the dialogs and printer control dialogs. It is important to allocate a single MapPrinting object for your application and use this same instance for print/print-preview/page setup.

```
Setup a MapPrinting and assign a map:
   this.mapPrinting = new MapPrinting();
   this.mapPrinting.Map = this.mapControl1.Map;
```

Following is sample code to print a map:

```
   this.mapPrinting.ShowDialog = true;
   this.mapPrinting.Print();
```

Following is sample code to display a print preview of a map:

```
   this.mapPrinting.PrintPreview();
```

Following is sample code to show the Page Setup dialog:

```
   this.mapPrinting.PageSettingsDialog();
```

You have access to the printer settings through the print document, which is part of the main printing class. Because we add setting and options to our derived class that support our mapping needs, you must get the class off the map printing object and re-cast it to our derived class.

```
   mapPrinting = new MapPrinting();
   mapPrinting.Map = mapControl1.Map;
   MapPrintDocument mapPrintDocument = mapPrinting.PrintDocument
```

```
as MapPrintDocument;
   if (mapPrintDocument != null) {
      // here are some examples of how to set print options
      // Set these based on your needs
      mapPrintDocument.DrawingAttributes.SpecialTransparentRasterHandling =
true;
      mapPrintDocument.PrintMethod = PrintMethod.Direct;
      // and set other properties of mapPrintDocument
   }
```

We do not provide any UI for changing our map print options. The dialogs allow the user to change system printer settings only. If you want to provide the user with additional options, you'll must provide your own UI to set the properties on MapPrintDocument (such as DrawingAttributes and PrintMethod and PrintSize).

The print document is also where you gain access to the print process and where you add callbacks when printing events occur. For example, you may want to add a logo or other graphics to each printed page.

C# example

```
mapPrinting = new MapPrinting();
mapPrinting.Map = mapControl1.Map;
mapPrinting.PrintDocument.PrintPage += new
PrintPageEventHandler(mapPrintDocument1_PrintPage);

private void mapPrintDocument1_PrintPage(object sender, PrintPageEventArgs e)
{
   // add customization for each page (ie; title, page #, etc.)
   Graphics g = e.Graphics;
}
```

VB example

```
mapPrinting = New MapPrinting()
mapPrinting.Map = mapControl1.Map
AddHandler mapPrinting.PrintDocument.PrintPage, AddressOf
mapPrintDocument1_PrintPage

Private Sub mapPrintDocument1_PrintPage(ByVal sender As Object, ByVal e As
PrintPageEventArgs)
    ' add customization for each page (ie; title, page #, etc.)
    Dim g As Graphics = e.Graphics
End Sub
```

Using this method, you have access to the graphics object and can use any of the available graphics routines to print extra graphics or text.

General Printing Tips and Tricks

ⓘ MapInfo cannot, and does not, recommend one printer/plotter over another, nor verify that a particular printer works 100 percent of the time. There are too many variables that affect the output to be able to make a recommendation.

The following tips and suggestions come from our knowledgebase of specific common problems and solutions our users have found while printing.

Good first steps in troubleshooting a printing problem are to make sure you have downloaded and installed the patch for your release of MapXtreme, if any, and are using the latest printer driver for your printer/operating system. Exceptions are noted in this document.

How to Overload a Print Page Event

Our .NET printing API provides a mechanism for overloading the print page event. This allows you to add customizations to each printed page. For example, you could add a title, page number, logo, etc.

To do this, you need to implement a PrintPageEventHandler. Here's an example:

```
this.mapPrinting = new MapPrinting();
this.mapPrinting.Map = this.mapControl1.Map;
this.mapPrinting.PrintDocument.PrintPage += new
System.Drawing.Printing.PrintPageEventHandler(this.mapPrintDocument1_PrintPage)
;

private void mapPrintDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{
   // TODO - add your code here to alter the printed page
}
```

Printing a Legend in Your Map

As mentioned at the beginning of this section, MapXtreme does not support the printing of layouts. If you want to print a legend in your map, you need to add the legend as an adornment to your map. Here is some sample code to demonstrate how this works:

```
//create cartographic legend
MapInfo.Mapping.Legends.Legend legend = mapControl1.Map.Legends.CreateLegend(new
System.Drawing.Size(5, 5));
legend.Border = true;
MapInfo.Mapping.LayerType[] normalLyr = new MapInfo.Mapping.LayerType[1];
normalLyr[0] = MapInfo.Mapping.LayerType.Normal;
```

```
MapInfo.Mapping.IMapLayerFilter filter =
MapInfo.Mapping.MapLayerFilterFactory.FilterByLayerType(normalLyr);
MapInfo.Mapping.Legends.LegendFrame frame;

foreach(MapInfo.Mapping.FeatureLayer ftrLayer in
mapControl2.Map.Layers.GetMapLayerEnumerator(filter))
{
   frame =
MapInfo.Mapping.Legends.LegendFrameFactory.CreateCartographicLegendFrame(ftrLay
er);
   legend.Frames.Append(frame);
}

//set legend location on the map
System.Drawing.Point pt = new System.Drawing.Point(220, 200);
pt.X = mapControl2.Size.Width - legend.Size.Width;
pt.Y = mapControl2.Size.Height - legend.Size.Height;
legend.Location = pt;

//append legend as map adornment
mapControl1.Map.Adornments.Append(legend);

mapPrinting.print ();
```

Old Driver Works, New Driver Doesn't

When in doubt, if an older driver worked and the new one does not, go back to the older driver.

HP 755 Driver Suggestion

If you are having difficulties printing using this plotter model, try the plotter driver for the HP 650C (C2859B) instead. In many cases, if you are having difficulty with a printer or plotter model and a similar model exists, you can substitute the similar model's driver for the current one and get good results. For example, you can use the printer driver for the HP 8500 DN Color LaserJet with the HP8550 color LaserJet driver.

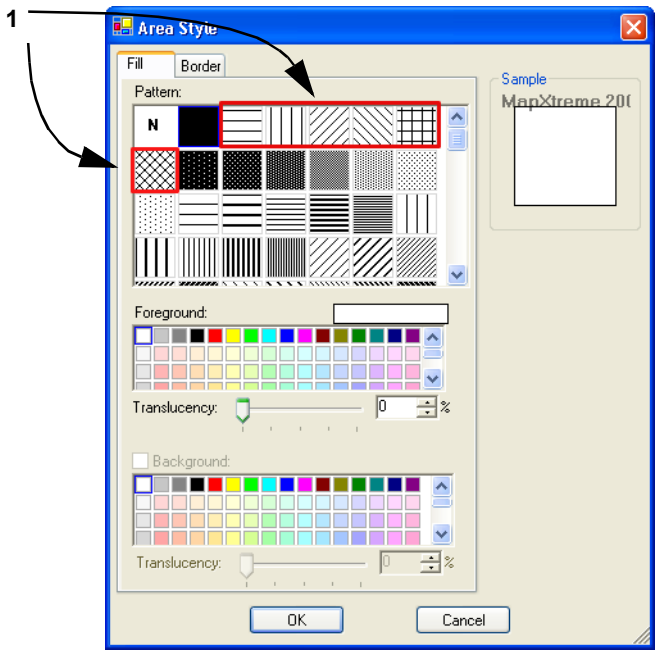
Recommendations for Effective Pattern Scaling

When you are printing, find out what type of printer driver you are using. Many PCL6 and some HPGL drivers handle fill pattern scaling and give you control over this feature. Turning off their scaling may be the difference between what you see in print and what you see on your monitor. We recommend that you try turning off your driver's scaling options and try ours first, because we have enhanced our method to better meet your Map and Legend printing requirements. To turn our pattern scaling options on, enable Scale Patterns in your application (see Scale Patterns. Try disabling our scaling to see which you like better. Tests show that our scaling produces color output that more closely matches a screen's display.

If you are printing to PostScript drivers using LanguageLevel 2 or 3, we find that some of the Microsoft drivers do not support pattern scaling. As a result, our scaling method may not help you. Microsoft recommended that you reset the language level of the PostScript driver to LanguageLevel 1 to remove this restriction. We did find some exceptions to this condition. On Windows 2000 and Windows NT, some HP Laser Jet and Color Laser Jet PostScript drivers, using our scaling option, printed correctly.

Need for Speed When Using Fill Patterns?

Note that the first six fill patterns (after the solid fill) in of the AreaStyle dialog box are Windows standard and tend to print faster. These fill patterns are vector-based. The rest of the patterns are bitmaps that ship with MapXtreme. You might want to consider this when you are selecting fill patterns.



1 Use these fill patterns for fast printing results.

Speed Still an Issue?

If you want to improve printer speed, and your printer has Fast, Normal, Best print quality options, we suggest you select Fast. This will also lower your output resolution.

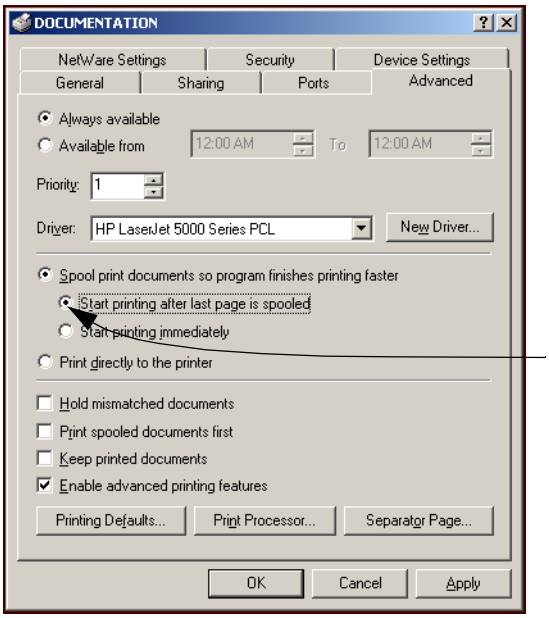
Disk Space an Issue?

Make sure you have plenty of temporary disk space, particularly if you are using the Print using the Enhanced Metafile option. The system is trying to create a layered bitmap locally on disk.

Print Globally? Spool Locally!

Try spooling[1] print jobs locally rather than at the plotter. This allows the computer to rasterize your output rather than the printer, which can be more efficient.

1. Spooling is the process of storing data constituting a document to be printed in memory or in a file until the printer is ready to process it.



1 Click Spool Print Documents and Start printing after last page is spooled button.

1. To set up local print job spooling, choose **Start > Settings > Control Panel > Printers**.
2. Right-click the printer and choose Properties in the menu to display the printer's properties.
3. Click the Advanced tab to display the advanced property options.

ⓘ If you do not have administrative rights to your computer, you may not be able to use the spooling option. Contact your IT department if you want to make this change to get their support.

---

4. Click **OK** to save your changes.

Resolutions to Known Printing Issues

There are many variables that affect printing and plotting with any application, and MapXtreme applications are no exception. MapXtreme does not provide printer drivers; it uses the existing ones installed under the current Windows operating system. This section addresses specific printer/plotter issues that have been uncovered by users and partners. We are providing these in an effort to support you to support your users.

Platform Independent Issues

These issues occur regardless of operating system or hardware/software, except where specifically noted. Look for the issue you are working with below and refer to the resolution.

Hatch Patterns Printing as Black Polygons

Printers/Plotters: HP Designjet Series Plotters

Issue: When you print maps with hatch patterns, some regions display as solid black.

Resolution: According to Hewlett Packard support, the new HP printer driver (4.63) handles non-Windows standard hatch patterns properly. We recommend that you download and install the new HP 4.63 driver to resolve this problem.

Hatch Patterns are Compressed when I Print

Operating System: WindowsNT 4.0/2000

Printers/Plotters: HP LaserJet Series, tested on HP LaserJet 4050 Series PCL 6

Issue: When you use a MapInfo hatch pattern (a pattern that is not one of the first 6 patterns AreaStyle dialog box), you may notice that the pattern may become very compressed. This is because the patterns are a bitmap that get drawn at a higher printer resolution. The standard Windows patterns do not exhibit this problem. The printer resolution affects the amount of compression.

Resolution: Some printer drivers have a new output setting called Scale Patterns (WYSIWYG) that enables the patterns to scale correctly. In the HP printer we tested, this option is available when you select the **Printing Preferences > Print Quality > Details** options.

Users should turn off either the driver's, or our scaling, as doing both scales fill patterns twice. Try both types of scaling to find the one you like better.

Platform-Specific Issues

These issues pertain to particular operating systems and/or hardware except where specifically noted. These issues are grouped by operating system.

Print output repeats itself every 2 to 4 cm using the HP 500, 800, 5000 DesignJet plotter when printing Rasters

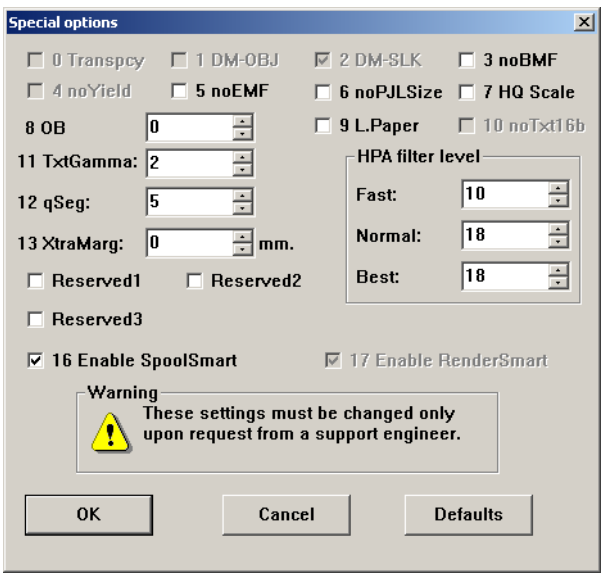
We have two suggestions to working around this problem. First, try spooling the printer locally either from the driver level or the printer level.

ⓘ You must have read/write rights to your printer and printer driver to resolve this issue.

To work around this problem, there is an advanced hidden setting that you can change within the properties of the driver.

To prevent the printer from repeating itself every 2-4 cm. when printing rasters with vector overlays:

1. Locate the printer driver's properties in the Control Panel. To get to this setting, right-click the printer icon and choose Properties.
2. Click the Advanced tab. Select the following options:
   - Select the **Avoid out of memory** option.
   - If you are using the Windows 2000 or XP operating system, select a scaling factor of 100%.
3. From this dialog box, click the **Printing Defaults** button.
4. In this dialog box, select the **About** button. The About <*Driver Name*> dialog box displays.
5. Holding down the **F8** key on your keyboard, click the **OK** button. The Special Options dialog box displays:



6. Clear the Enable SpoolSmart check box and click **OK**:

ⓘ When you modify settings in this driver option box, make sure to note the original settings in case you need to change them back. These were designed for HP Support Engineers and not for general public use.

You can save these settings as a Quick Set to ensure that they run every time you print raster files.

1. Navigate to the HP printer driver and access the driver preferences. Use one of these methods based on your operating system:
   - **Using Windows NT 4.0** — From the Windows desktop, select **Start > Settings > Printers**. Then, right-click the appropriate plotter driver icon and choose Document Defaults.
   - **Using Windows 2000/XP (Classic View**) — From the Windows desktop, select **Start > Settings > Printers**. Then, right-click the appropriate plotter driver icon and choose Printing Preferences.
2. Set the preferences the way you want them in the hidden menu.
3. In the Quick Sets box, type a name for the selected settings (for example, "Raster Print Settings") and click **Save**. All current driver settings (paper type, hidden menu settings, etc.) are saved under the Quick Set name. The printer driver remembers these settings and they can be used for future print jobs.

To print a raster map later with the same settings:

1. Choose the Print command to display the Print dialog box.
2. Choose the **Properties** button. The Properties dialog box displays.
3. Make sure the name you entered (as in “Raster Print Settings” in our example) displays in the Quick Sets drop-down list and click **OK**. The Print dialog box redisplays.
4. Click **OK** to print.

ⓘ If you do not need this special setting in Quick Sets, remember to change this setting back to the default machine setting.

Cannot Print with HP Designjet Printers (Driver 5.31 or 5.32)

When you attempt to use these drivers one of these things happens:

- If you select the **Maximum Performance** option, you get memory error messages and the printer prints only part of the image
- If you select the **Avoid out of memory** option, the computer crashes and displays both an Out of Hard Disk Space message and an Out of Virtual Memory message

The resolution of this issue is similar to the preceding issue. The Special Options dialog box is the same.

1. Programmatically set the following printing options:

```
MapPrintDocument mapPrintDocument = this.mapPrinting.PrintDocument as
MapPrintDocument;
mapPrintDocument.PrintMethod = PrintMethod.Direct;
mapPrintDocument.DrawingAttributes.SpecialTransparentVectorHandling = true;
mapPrintDocument.DrawingAttributes.SpecialTransparentRasterHandling = false;
mapPrintDocument.DrawingAttributes.TrueColorRaster = true;.
```

2. From the Control Panel, select Printers or Printers and Faxes and find the printer you want to print to.
3. Right-click on the printer and select the Properties option. The Properties dialog box displays.
4. Click the **About** button to display the About <*Printer Driver*> dialog box.
5. Holding down the F8 key on your keyboard, click the **OK** button. The Special Options dialog box displays.
6. Clear the Enable SpoolSmart check box.

F – Style Lookups

This appendix contains lookup tables for supported styles, including fill patterns, line styles, vector symbols and custom bitmap symbols. For more information on style descriptions and how to use them, see Chapter 15 Stylizing Your Maps.

In this appendix:

Fill Patterns

The following table summarizes the MapXtreme fill patterns (also referred to as interior styles within the MapXtreme programming API). Each fill pattern has an associated Index Number, which is used for programmatic access into an InteriorStyleRepository object, and a Pattern Number, which is an internal descriptive name of the fill pattern.

More specifically:

Fill Pattern
The graphical fill pattern itself.

Index Number
The 0-based index used to retrieve the interior style that represents the fill pattern from the InteriorStyleRepository.

Pattern Number
The numeric identifier for the fill pattern, which may be used to construct an interior style object. This is an internal number only and can *not* be used to access the InteriorStyleRepository programmatically. However, these pattern numbers are used to indicate fill patterns within MapXtreme workspace files (.mws).

Understanding the Index Numbering Schemes

As seen in the Fill Patterns table, the fill pattern indexes (listed in the Index Number column) range from 1 to 172. This 1-based indexing scheme is used throughout other MapInfo products (for example, MapInfo Professional and MapBasic). However in the MapXtreme API, fill patterns (also called interior style objects) are acquired through an InteriorStyleRepository object that uses 0-based indexing.

Specifically, an InteriorStyleRepository object allows users to iterate through all of the available interior style objects via a 0-based indexing scheme that ranges from 0 to 172. So there are actually 173 interior style objects to choose from, however, the interior style objects at indexes 0 and 1 are identical. The equivalence of the objects at elements 0 and 1 is necessary for the following reasons:

First, we must accommodate the inherent 0-based indexing functionality of the InteriorStyleRepository class itself. That is, the InteriorStyleRepository class permits only 0-based index access for retrieving elements from its IList. (IList is a standard .NET interface that the InteriorStyleRepository class implements.)

Second, the 0-based indexing scheme must remain synchronized with the 1-based indexing scheme shown in the Pattern Number column. So there are only 172 distinct fill patterns whose indexes range from 1 to 172. The programmatic fill pattern at index 0 was added to give a definition for that index's element.

ⓘ You will notice that the first eight fill patterns have the same Index Number and Pattern Number. Subsequently, the Pattern Number is always three greater than the corresponding Index Number (and hence the Pattern Numbering sequence goes up to 175 while the Index Numbering stops at 172). This is not an error. There are still 172 unique fill patterns. Nothing is missing from the table.

Fill Patterns and Associated Index and Patterns Numbers

Fill Pattern	Index Number	Pattern Number
N (None)	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	12
	10	13

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	11	14
	12	15
	13	16
	14	17
	15	19
	16	18
	17	20
	18	21
	19	22
	20	23
	21	24
	22	25
	23	26

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	24	27
	25	28
	26	29
	27	30
	28	31
	29	32
	30	33
	31	34
	32	35
	33	36
	34	37
	35	38
	36	39

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	37	40
	38	41
	39	42
	40	43
	41	44
	42	45
	43	46
	44	47
	45	48
	46	49
	47	50
	48	51
	49	52

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	50	53
	51	54
	52	55
	53	56
	54	57
	55	58
	56	59
	57	60
	58	61
	59	62
	60	63
	61	64
	62	65

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	63	66
	64	67
	65	68
	66	69
	67	70
	68	71
	69	72
	70	73
	71	74
	72	75
	73	76
	74	77
	75	78
	76	79

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	77	80
	78	81
	79	82
	80	83
	81	84
	82	85
	83	86
	84	87
	85	88
	86	89
	87	90
	88	91
	89	92
	90	93

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	91	94
	92	95
	93	96
	94	97
	95	98
	96	99
	97	100
	98	101
	99	102
	100	103
	101	104
	102	105

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	103	106
	104	107
	105	108
	106	109
	107	110
	108	111
	109	112
	110	113
	111	114
	112	115
	113	116
	114	117

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	115	118
	116	119
	117	120
	118	121
	119	122
	120	123
	121	124
	122	125
	123	126
	124	127
	125	128
	126	129

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	127	130
	128	131
	129	132
	130	133
	131	134
	132	135
	133	136
	134	137
	135	138
	136	139
	137	140
	138	141

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	139	142
	140	143
	141	144
	142	145
	143	146
	144	147
	145	148
	146	149
	147	150
	148	151
	149	152
	150	153

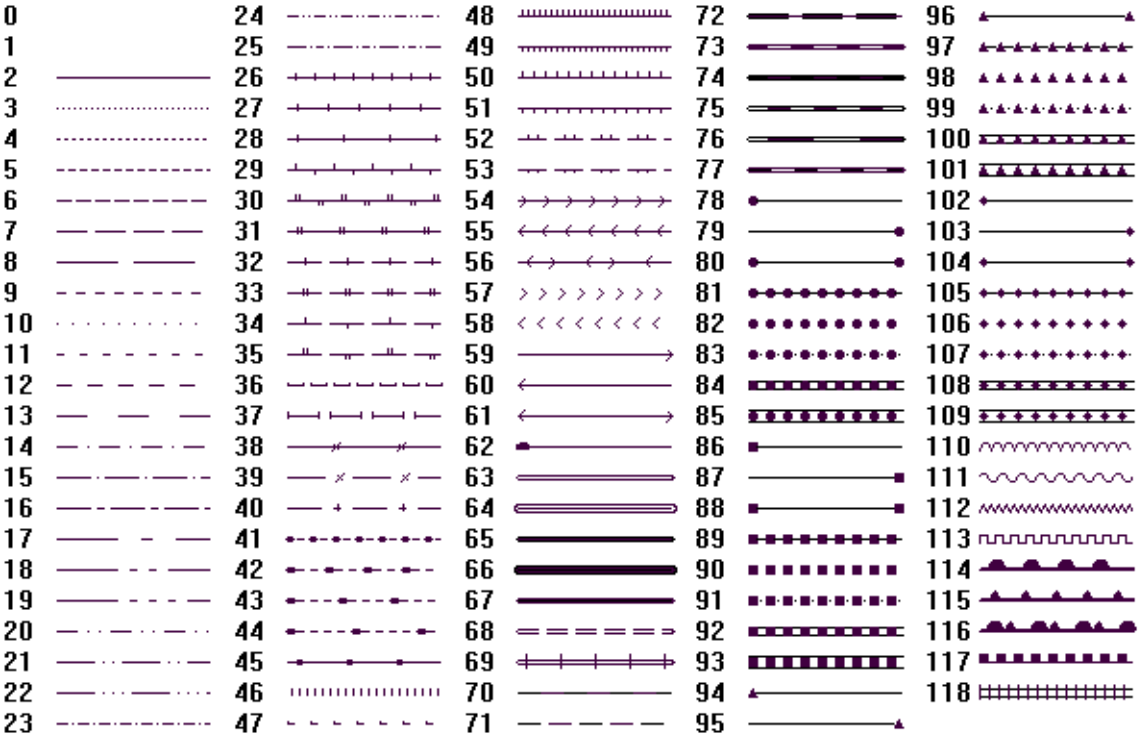
Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	151	154
	152	155
	153	156
	154	157
	155	158
	156	159
	157	160
	158	161
	159	162
	160	163
	161	164
	162	165

Fill Patterns and Associated Index and Patterns Numbers *(continued)*

Fill Pattern	Index Number	Pattern Number
	163	166
	164	167
	165	168
	166	169
	167	170
	168	171
	169	172
	170	173
	171	174
	172	175

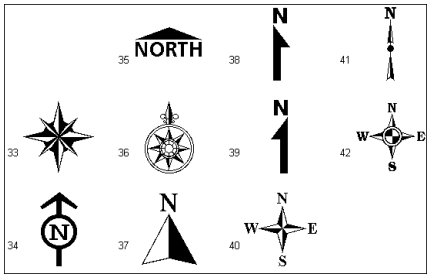
Line Styles



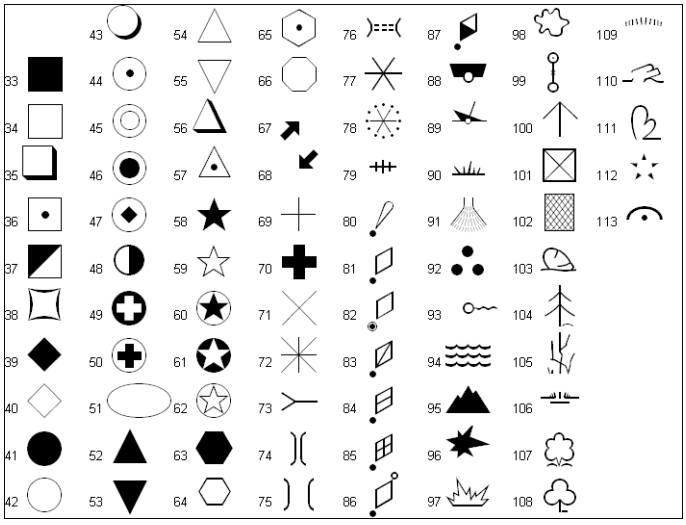
Vector Symbols

MapXtreme automatically installs 10 MapInfo-specific TrueType fonts during its installation process. These fonts offer the user glyph symbol choices that range from Weather, Real Estate, and Transportation, and others. The glyph numbers are Unicode character values, which, since they fall within the first Unicode character code block range, are also assignment-wise compatible with the ASCII character set.

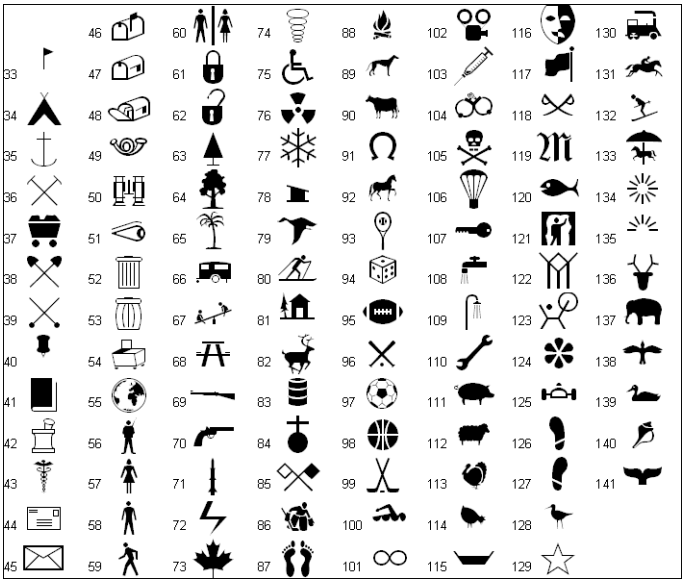
MapInfo Arrows



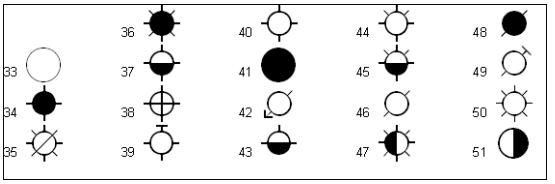
MapInfo Cartographic



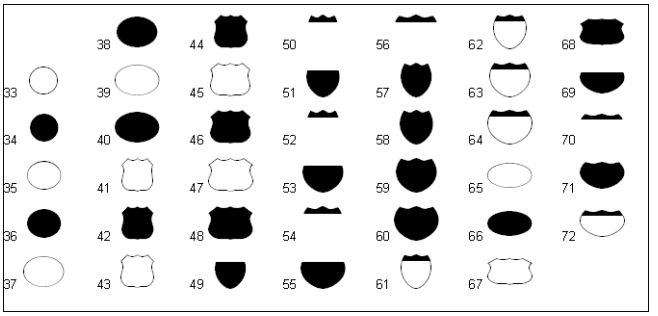
MapInfo Miscellaneous



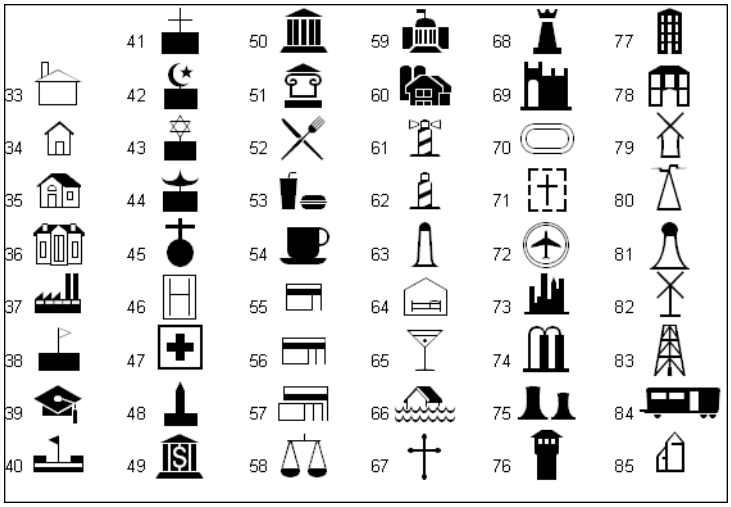
MapInfo Oil &Gas



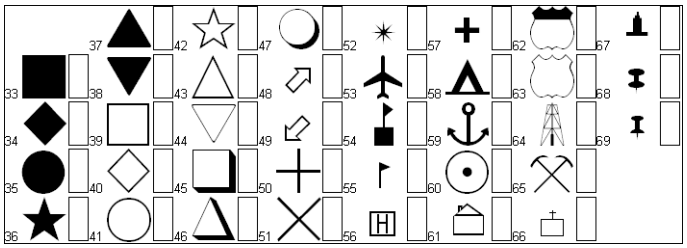
MapInfo Shields



MapInfo Real Estate

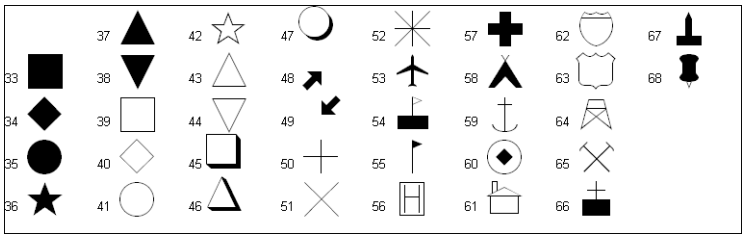


Map Symbols

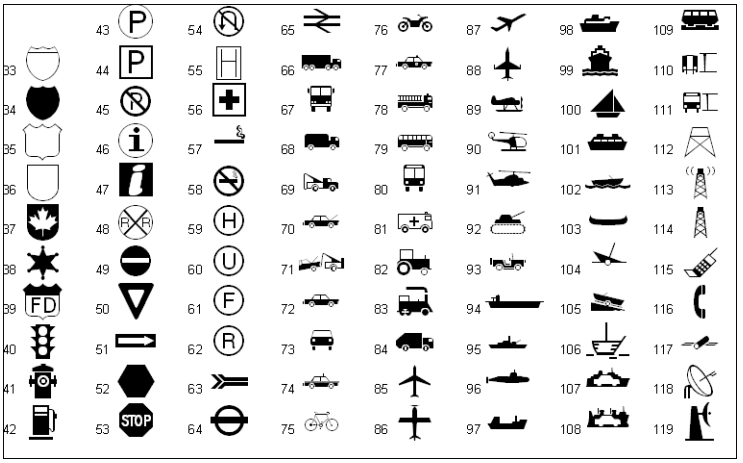


MapInfo Symbols

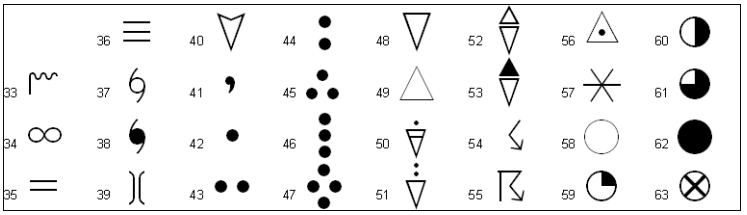
MapInfo 3.0 Compatible symbols



MapInfo Transportation



MapInfo Weather



Custom Symbols

The following symbols are located in programdata\Mapinfo\custsymb. The file extension for each image is .BMP.

These symbols can be accessed programmatically via the BitmapPointStyleRepository collection class in the MapInfo.Styles namespace.

You can create your own bitmap images and add them to the CustSymb directory. There are virtually no size limitations on an image that you create; however, the ability of MapXtremeto display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To ensure that your image is displayed using its natural width and height, you must set the Boolean "NativeSize" property to true within the image's respective BitmapPointStyle object.

AMBU1-32

AMBU-64

BADG1-32

BADG2-32

BANK1-32

BANK2-32

BANK-64

BOOK1-32

BUILDINGS

CAMP1-32

CAR1-32

CAUT1-32

CHUR1-32

COMP1-32

FARM1-32

FAST1-32

FIRE1-32

FIRE-64

FOOD-64

GLOB1-32

GOLF1-32

HOSP1-32

HOUS1-32

HOUS2-32

HOUS3-32	HOUS-64	HYDR1-32	INTE1-32	LITE1-32	LITE2-32
					
MAIL1-32	MBOX1-32	MBOX2-32	MOSQ1-32	ONEW1-32	ONEW2-32
					
PENC1-32	PIN1-32 (CYAN)	PIN2-32 (RED)	PIN3-32 (YELLOW)	PIN4-32 (GREEN)	PIN5-32 (BLUE)
					
PIN6-32 (PURPLE)	PINB-64 (BLUE)	PING-64 (GREEN)	PINGY-64 (GRAY)	PINR-64 (RED)	POLI1-32
					
RAIL1-32	RAIL2-32	RAIL3-32	RAIL-64	RED-CAR	REST1-32
					
STAT1-32	STOP1-32	SYNA1-32	TARG1-32	TAXI1-32	TEMP1-32
					
TOWE1-32	TOWE2-32	TRAF1-32	TRUC1-32	TRUC2-32	TRUC-64
					

YIEL1-32

YIEL2-32

MapXtreme Icons

This table of thumbnails represent a collection of toolbar icons available for your use in your MapXtreme-based application. They are Installed into the \Samples\Icons folder under the MapXtreme installation folder. There are two forms of each .PNG: small (16x16 pixels) and large (24x24 pixels).

 ADD_NODE	 ADORNMENT	 ARC	 ARROW	 ASSIGN_TARGET	 CLIP_MODE
 CLIP_REGION	 CLOSE_ALL	 COPY	 CREATE_DRIVE_REG	 CROSSHAIR	 CUT
 DRAG_HANDLE	 ELLIPSE	 FIND_ADDRESS	 GEOCODE_USE_SRV	 GRABBER	 GRAPH_SELECT
 HELP	 HOT_LINK	 INFO	 INVERTSELECT	 LABEL	 LAYERS
 LEGEND	 LINE	 LINE_STYLE	 MB_12	 MB_6	 MB_7

MB_8

NEW_BROWSER

NEW_DOC

NEW_GRAPHER

NEW_LAYOUT

NEW_MAPPER

NEW_REDISTRICTER

ODBC_DISCONNECT

ODBC_MAPPABLE

ODBC_OPEN

ODBC_REFRESH

ODBC_SYMBOL

ODBC_UNLINK

OPENWFS

OPENWMS

OPEN_FILE

OPEN_WOR

PASTE

POLYGON

POLYGON_STYLE

POLYLINE

PRINT

PRINT_PDF

RECT

RESHAPE

ROUND_RECT

RULER

RUN

SAVE_FILE

SAVE_WIN

SAVE_WOR

SCALEBAR

SEARCH_BDY

SEARCH_POLYGON

SEARCH_RADIUS

SEARCH_RECT

SET_TARGET_MAP

STATISTICS

SYMBOL

SYMBOL_STYLE

TEXT

TEXT_STYLE

UNDO

UNSELECT_ALL

WEB_SERVICE_PREF

WINDOW_FRAME

WRENCH

ZOOM_IN

ZOOM_OUT

ZOOM_QUESTION

G

G – Defining the MapInfo Codespace

The MapInfo Codespace is a list of definitions and standards that are commonly used in creating MapInfo maps and workspaces. You can refer to these definitions to assist you in using MapXtreme. You may want to compare our codespace definitions with another commonly used codespace, the European Petroleum Survey Group (EPSG), which is available on their website: www.epsg.org/.

In this appendix:

Defining the MapInfo Codespace

In the following table, we show the current MapInfo Codespace. These definitions will allow your XML-based map documents to use our codespace and refer to common values, such as srsName. An example of a point geometry definition, for example, could look like this:

```
<gml:Point srsName="mapinfo:coordsys 1,74"> ...</gml:Point> or
<gml:Point srsName="epsg:4269"> ...</gml:Point>
```

MapInfo Codespace Definition

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
coordsys	CoordSys file*				A coordinate reference system. Example: mapinfo:coordsys 1,74
pen	[1-255]				Simple Mapinfo pen pattern; numbering [1-255] based on default MapInfo MapInfow.pen definitions. Follows all established rules. Example: mapinfo:pen 46
brush	[1-8, 12-175]				Simple Mapinfo brush pattern number [1-8, 12-175]. (Note: 9-11 are reserved.) Based on default MapInfo bitmap definitions. Example: mapinfo:brush 17

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
length	in | ft | yd | mi mm | cm | m | km | sft | nmi | li | ch | rd | pt | twip | pica | deg pt= 1/72 in pica = 12 pt twip = 1/20 pt 1 deg = (pi/180) * 6370997 meter = 69.09329 mi (based on great circle)				Unit of length. Supports inch | foot | yard | mile | and millimeter | centimeter | meter | kilometer | and survey foot | nautical mile | and link | chain | rod | point | twip | pica | degree. Examples: mapinfo:length m mapinfo:length pt mapinfo:length deg
imagesize	pixel				A non-linear unit of image dimension. Example: mapinfo:imagesize pixel
type					A data type.
	boolean				Example: mapinfo:type boolean
	byte				Example: mapinfo:type byte
	date				Example: mapinfo:type date
	datetime				Example: mapinfo:type datetime

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
type	time				Example: mapinfo:type time
	decimal	[| (n	n) |]	n.m n is the total digits and m is the number of digits reserved for the right of decimal point, m<=n	optional min optional max optional precision Examples: mapinfo:type decimal (100 -- all d>100 mapinfo:type decimal 100) -- all d<100 mapinfo:type decimal [200 300] -- all 200<=d<=300 mapinfo:type decimal 10.7 -- all decimals with no more than 10 digits with 7 digits reserved for right of decimal point, thus max of 3 digits to left of decimal point. mapinfo:type decimal [200 300] 10.7 -- combination of previous two examples.

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
type (continued)	double	[| (n	n) |]		Example: mapinfo:type double (0 150)
	float	[| (n	n) |]		Example: mapinfo:type float 30.25]
	int	[| (n	n) |]		Example: mapinfo:type int [0 180]
	short	[| (n	n) |]		Example: mapinfo:type short
	string	n			Optional max length n. Example: mapinfo:type string 256
	char				Example: mapinfo:type char

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
operators					A scalar attribute comparison operator or a geometry operator or a boolean operator.
	eq | neq | lt | gt | lteq | gteq				Binary scalar value operator. Example: mapinfo:op eq
	in | not_in				Scalar value in/not_in an enumerated set of scalar values, e.g., value in {2, 50, 88, 95} value not_in {“NY”, “NJ”}. Examples: mapinfo:op in mapinfo:op not_in
	between | not_between				Scalar value is between two other scalar values based on the order relation of that scalar value type, e.g., value between {5, 25} value not_between {1may2000, 30may2000} Examples: mapinfo:op between mapinfo:op not_between

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
operators (continued)	like | not_like				A string value matches the RHS pattern. The format of the RHS pattern may be specific to the data source, e.g., lastname like “Jo%”. (Uses the Oracle wildcard character '%'). Examples: mapinfo:op like mapinfo:op not_like
	intersects				The feature geometry fg intersects the given Polygon. Example: mapinfo:op intersects
	mbr_intersects				The the mbr (minimum bound rectangle) of the feature geometry (fg) intersects the mbr of the given Polygon. Example: mapinfo:op mbr_intersects
	contains				The feature geometry (fg) contains the given Point. The given Polygon contains the feature geometry fg. Example: mapinfo:op contains
	contains_centroid				The feature geometry (fg) contains the centroid of the given Polygon. The given polygon contains the centroid of the feature geometry (fg) Example: mapinfo:op contains_centroid

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
operators (continued)	and				Boolean and. Example: mapinfo:op and
	or				Boolean or. Example: mapinfo:op or

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
area					A unit of area.
	sq	in | ft | yd | mi | mm | cm | m | km | sft | nmi | li | ch | rd | pt | twip | pica | degree			Square linear units. Example: mapinfo:area sq mi
	acre | a | ha | perch | rood a = are (an area of 10m x10m) ha = hectare (an area of 100m x 100m) perch = 272.25 sq ft (one sq rod 16.5 ft) rood = 40 perch (1/4 acre)				A predefined area unit. Example: mapinfo:area acre
time	msec | sec | min | hr | day | week | month | year				A unit of time. Millisecond | Second | Minute, Hour | Day | Week | Month | Year Example: mapinfo:time hr

MapInfo Codespace Definition *(continued)*

Category	Position 1	Position 2	Position 3	Position 4	Explanation and Example
angle	deg | rad				A unit of angular measure. degree | radian Example: mapinfo:angle rad
temp	K | F | C				A unit of temperature. Kelvin | Fahrenheit | Celcius Example: mapinfo:temp C

* Coordinate System information for MapXtreme is located in the MapInfoCoordinateSystemSet.xml, located in C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x.

H – Elements of a Coordinate System

This appendix provides a detailed look at the elements of coordinate systems, including supported datums, ellipses, and transformations.

In this appendix:

Projections and Their Parameters

The following table indicates the parameters applicable to each projection, which are listed in the order they appear in the relevant coordinate system lines in the MapInfoCoordinateSystemSet.xml. The document is located in C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x.

	Datum	Units	Origin, Longitude	Origin, Latitude	Standard Parallel 1	Standard Parallel 2	Azimuth	Scale Factor	False Easting	False Northing	Range	Bearing
Albers Equal-Area Conic	X	X	X	X	X	X			X	X		
Azimuthal Equidistant	X	X	X	X							X	
Cassini-Soldner	X	X	X	X					X	X		
Cylindrical Equal Area	X	X	X		X							
Double Stereographic	X	X	X	X				X	X	X		
Eckert IV	X	X	X									
Eckert VI	X	X	X									
Equidistant Conic	X	X	X	X	X	X			X	X		
Gall	X	X	X									
Hotine Oblique Mercator	X	X	X	X			X	X	X	X		
Lambert Azimuthal Equal-Area	X	X	X	X							X	
Lambert Conformal Conic	X	X	X	X	X	X			X	X		
Longitude-Latitude	X											
Mercator	X	X	X									
Miller	X	X	X									

	Datum	Units	Origin, Longitude	Origin, Latitude	Standard Parallel 1	Standard Parallel 2	Azimuth	Scale Factor	False Easting	False Northing	Range	Bearing
Mollweide	X	X	X									
New Zealand Map Grid	X	X	X	X					X	X		
Polyconic	X	X	X	X					X	X		
Regional Mercator	X	X	X		X							
Robinson	X	X	X									
Sinusoidal	X	X	X									
Stereographic	X	X	X	X				X	X	X		
Swiss Oblique Mercator	X	X	X	X					X	X		
Transverse Mercator	X	X	X	X				X	X	X		
S-JTSK (Krovak)	X	X	X	X				X	X	X		
Cylindrical	X	X	X	X				X	X	X		
Rectified Skewed Orthomorphic (RSO)	X	X	X	X				X	X	X		X

Projection

The projection is the equation or equations used by a coordinate system. The following list names the projections MapInfo uses and gives the number used to identify the projection in the MapInfoCoordinateSystemSet.xml file:

Number	Projection
9	Albers Equal-Area Conic
28	Azimuthal Equidistant (all origin latitudes)
5	Azimuthal Equidistant (polar aspect only)
30	Cassini-Soldner
2	Cylindrical Equal-Area
31	Double Stereographic
14	Eckert IV
15	Eckert VI
6	Equidistant Conic, also known as Simple Conic
17	Gall
7	Hotine Oblique Mercator
4	Lambert Azimuthal Equal-Area (polar aspect only)
29	Lambert Azimuthal Equal-Area
3	Lambert Conformal Conic
19	Lambert Conformal Conic (modified for Belgium 1972)
1	Longitude/Latitude
10	Mercator
11	Miller Cylindrical
13	Mollweide
18	New Zealand Map Grid
27	Polyconic
26	Regional Mercator
12	Robinson

Number	Projection
16	Sinusoidal
20	Stereographic
25	Swiss Oblique Mercator
8	Transverse Mercator, (also known as Gauss-Kruger)
21	Transverse Mercator, (modified for Danish System 34 Jylland-Fyn)
22	Transverse Mercator, (modified for Danish System 34 Sjaelland)
23	Transverse Mercator, (modified for Danish System 34/45 Bornholm)
24	Transverse Mercator, (modified for Finnish KKJ)
32	Krovak
33	Equidistant Cylindrical
34	Extended Transverse Mercator
35	Rectified Skewed Orthomorphic

For example, a Longitude/Latitude projection is listed in the MapInfoCoordinateSystemSet.xml as:

```
<gml:GeographicCRS>
   <gml:srsName>Longitude / Latitude (Porto Santo 1936) [EPSG
4615]</gml:srsName>
   <gml:usesEllipsoidalCS />
   <gml:usesGeodeticDatum />
   <gml:metaDataProperty>
      <gml:category>Longitude / Latitude (v 6.0 and later
projections)</gml:category>
   </gml:metaDataProperty>
   <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 1,143</gml:name>
   </gml:srsID>
   <gml:srsID>
      <gml:name gml:codeSpace="EPSG" axis1="north" axis2="east">4615</gml:name>
   </gml:srsID>
   <gml:srsID>
```

```
     <gml:name
gml:codeSpace="WKT">GEOGCS[""MADEIRA"",DATUM[""MADEIRA"",SPHEROID[""Internation
al -
1924"",6378388,297.0000000000014]],PRIMEM[""Greenwich"",0],UNIT[""degree"",0.01
74532925199433]]"</gml:name>
   </gml:srsID>
</gml:GeographicCRS>
```

Projection numbers in the MapInfoCoordinateSystemSet.xml may be modified by the addition of a constant value to the base number listed in the Projection table, above. Valid values and their meanings are tabulated below:

Constant	Meaning	Parameters
1000	System has affine transformations	Affine units specifier and coefficients appear after the regular parameters for the system.
2000	System has explicit bounds	Bounds appear after the regular parameters for the system.
3000	System with both affine and bounds	Affine parameters follow system's parameters; bounds follow affine parameters.

Example:

Assume you want to work with a simple system based on the Transverse Mercator projection and using the NAD 1983 datum. You might have a line such as the following in your MapInfoCoordinateSystemSet.xml file:

```
<gml:ProjectedCRS>
   <gml:srsName>UTM Zone 1 (NAD 83) [EPSG 26901]</gml:srsName>
   <gml:baseCRS />
   <gml:definedByConversion />
   <gml:usesCartesianCS />
   <gml:metaDataProperty>
      <gml:category>Universal Transverse Mercator (NAD 83)</gml:category>
   </gml:metaDataProperty>
   <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 8,74,7,-
177,0,0.9996,500000,0</gml:name>
   </gml:srsID>
   <gml:srsID>
      <gml:name gml:codeSpace="EPSG" axis1="east" axis2="north">26901</gml:name>
   </gml:srsID>
   <gml:srsID>
```

```
     <gml:name gml:codeSpace="WKT">PROJCS[""NAD83 UTM, Zone 1 North,
Meter"",GEOGCS[""NAD83"",DATUM[""North_American_Datum_1983"",SPHEROID[""Geodeti
c Reference System of
1980"",6378137,298.2572221008916,AUTHORITY[""EPSG"",""7019""]],AUTHORITY[""EPSG
"",""6269""]],PRIMEM[""Greenwich"",0],UNIT[""degree"",0.0174532925199433],AUTHO
RITY[""EPSG"",""4269""]],PROJECTION[""Transverse_Mercator""],PARAMETER[""latitu
de_of_origin"",0],PARAMETER[""central_meridian"",-
177],PARAMETER[""scale_factor"",0.9996],PARAMETER[""false_easting"",500000],PAR
AMETER[""false_northing"",0],UNIT[""METER"",1],AUTHORITY[""EPSG"",""26901""]]"<
/gml:name>
   </gml:srsID>
</gml:ProjectedCRS>
```

Now let's say that you want a system based on this, but with an affine transformation specified by the following parameters:

```
Units=meters; A=0.5; B=-0.866; C=0; D=0.866; E=0.5; and F=0.
```

The required line in the MapInfoCoordinateSystemSet.xml file is:

```
<gml:ProjectedCRS>
    <gml:srsName>UTM Zone 1 (NAD 83) [EPSG 26901] - rotated 60
degrees</gml:srsName>
    <gml:baseCRS />
    <gml:definedByConversion />
    <gml:usesCartesianCS />
    <gml:metaDataProperty>
      <gml:category>Universal Transverse Mercator (NAD 83)</gml:category>
    </gml:metaDataProperty>
    <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 8,74,7,-
177,0,0.9996,500000,0,7,0.5,-0.866,0,0.866,0.5,0</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="EPSG" axis1="east" axis2="north">26901</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="WKT">PROJCS[""NAD83 UTM, Zone 1 North,
Meter"",GEOGCS[""NAD83"",DATUM[""North_American_Datum_1983"",SPHEROID[""Geodeti
c Reference System of
1980"",6378137,298.2572221008916,AUTHORITY[""EPSG"",""7019""]],AUTHORITY[""EPSG
"",""6269""]],PRIMEM[""Greenwich"",0],UNIT[""degree"",0.0174532925199433],AUTHO
RITY[""EPSG"",""4269""]],PROJECTION[""Transverse_Mercator""],PARAMETER[""latitu
de_of_origin"",0],PARAMETER[""central_meridian"",-
177],PARAMETER[""scale_factor"",0.9996],PARAMETER[""false_easting"",500000],PAR
AMETER[""false_northing"",0],UNIT[""METER"",1],AUTHORITY[""EPSG"",""26901""]]"<
/gml:name>
    </gml:srsID>
</gml:ProjectedCRS>
```

Alternatively, if you want to bound the system to (x1, y1, x2, y2)=(-500000, 0, 500000, 1000000), the required line is:

```
<gml:ProjectedCRS>
    <gml:srsName>UTM Zone 1 (NAD 83) [EPSG 26901] - bounded</gml:srsName>
    <gml:baseCRS />
    <gml:definedByConversion />
    <gml:usesCartesianCS />
    <gml:metaDataProperty>
      <gml:category>Universal Transverse Mercator (NAD 83)</gml:category>
    </gml:metaDataProperty>
    <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 8,74,7,-177,0,0.9996,500000,0,-
500000,0,500000,1000000</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="EPSG" axis1="east" axis2="north">26901</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="WKT">PROJCS[""NAD83 UTM, Zone 1 North,
Meter"",GEOGCS[""NAD83"",DATUM[""North_American_Datum_1983"",SPHEROID[""Geodeti
c Reference System of
1980"",6378137,298.2572221008916,AUTHORITY[""EPSG"",""7019""]],AUTHORITY[""EPSG
"",""6269""]],PRIMEM[""Greenwich"",0],UNIT[""degree"",0.0174532925199433],AUTHO
RITY[""EPSG"",""4269""]],PROJECTION[""Transverse_Mercator""],PARAMETER[""latitu
de_of_origin"",0],PARAMETER[""central_meridian"",-
177],PARAMETER[""scale_factor"",0.9996],PARAMETER[""false_easting"",500000],PAR
AMETER[""false_northing"",0],UNIT[""METER"",1],AUTHORITY[""EPSG"",""26901""]]"<
/gml:name>
    </gml:srsID>
</gml:ProjectedCRS>
```

To customize the system using both of these modifications, the xml should be changed to:

```
<gml:ProjectedCRS>
    <gml:srsName>UTM Zone 1 (NAD 83) [EPSG 26901]</gml:srsName>
    <gml:baseCRS />
    <gml:definedByConversion />
    <gml:usesCartesianCS />
    <gml:metaDataProperty>
      <gml:category>Universal Transverse Mercator (NAD 83)</gml:category>
    </gml:metaDataProperty>
    <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 8,74,7,-177,0,0.9996,500000,0,
7,0.5,-0.866,0,0.866,0.5,0,-500000,0,500000,1000000</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="EPSG" axis1="east" axis2="north">26901</gml:name>
    </gml:srsID>
    <gml:srsID>
      <gml:name gml:codeSpace="WKT">PROJCS[""NAD83 UTM, Zone 1 North,
Meter"",GEOGCS[""NAD83"",DATUM[""North_American_Datum_1983"",SPHEROID[""Geodeti
c Reference System of
1980"",6378137,298.2572221008916,AUTHORITY[""EPSG"",""7019""]],AUTHORITY[""EPSG
```

```
"",""6269""]],PRIMEM[""Greenwich"",0],UNIT[""degree"",0.0174532925199433],AUTHO
RITY[""EPSG"",""4269""]],PROJECTION[""Transverse_Mercator""],PARAMETER[""latitu
de_of_origin"",0],PARAMETER[""central_meridian"",-
177],PARAMETER[""scale_factor"",0.9996],PARAMETER[""false_easting"",500000],PAR
AMETER[""false_northing"",0],UNIT[""METER"",1],AUTHORITY[""EPSG"",""26901""]]"<
/gml:name>
    </gml:srsID>
</gml:ProjectedCRS>
```

ⓘ A new codespace WKT has been added in the above examples. It has been introduced in MapXtreme 8.0 and above to identify coordinate reference systems through their OGC well-known text (WKT) representations. For more details regarding OGC WKT representation, visit http://www.opengeospatial.org/standards/wkt-crs.

Projection Datums

The datum is established by tying a reference ellipsoid to a particular point on the earth. The following table lists these details for each datum:

- The number used to identify the datum in the MapInfoCoordinateSystemSet.xml file.
- The datum's name
- The maps for which the datum is typically used
- The datum's reference ellipsoid

Number	Datum	Area Maps	Ellipsoid
1	Adindan	Ethiopia, Mali, Senegal, Sudan	Clarke 1880
2	Afgooye	Somalia	Krassovsky
1007	AGD 66, 7 parameter	Australia, A.C.T.	Australian National
1008	AGD 66, 7 parameter	Australia, Tasmania	Australian National
1009	AGD 66, 7 parameter	Australia, Victoria/NSW	Australian National
1006	AGD 84, 7 parameter	Australia	Australian National
3	Ain el Abd 1970	Bahrain Island	International

Number	Datum	Area Maps	Ellipsoid
118	American Samoa	American Samoa Islands	Clarke 1866
4	Anna 1 Astro 1965	Cocos Islands	Australian National
119	Antigua Island Astro 1943	Antigua, Leeward Islands	Clarke 1880
5	Arc 1950	Botswana, Lesotho, Malawi, Swaziland, Zaire, Zambia, Zimbabwe	Clarke 1880
6	Arc 1960	Kenya, Tanzania	Clarke 1880
7	Ascension Island 1958	Ascension Island	International
9	Astro B4 Sorol Atoll	Tern Island	International
8	Astro Beacon “E”	Iwo Jima Island	International
10	Astro DOS 71/4	St. Helena Island	International
11	Astronomic Station 1952	Marcus Island	International
151	ATS77 (Average Terrestrial System 1977)	Canada	ATS77
12	Australian Geodetic 1966 (AGD 66 - 3 param)	Australia and Tasmania Island	Australian National
13	Australian Geodetic 1984 (AGD 84 - 3 param)	Australia and Tasmania Island	Australian National
120	Ayabelle Lighthouse	Djibouti	Clarke 1880
110	Belgium	Belgium	International

Number	Datum	Area Maps	Ellipsoid
14	Bellevue (IGN)	Efate and Erromango Islands	International
15	Bermuda 1957	Bermuda Islands	Clarke 1866
16	Bogota Observatory	Colombia	International
121	Bukit Rimpah	Bangka and Belitung Islands (Indonesia)	Bessel 1841
17	Campo Inchauspe	Argentina	International
18	Canton Astro 1966	Phoenix Islands	International
19	Cape	South Africa	Clarke 1880
20	Cape Canaveral	Florida and Bahama Islands	Clarke 1866
1005	Cape, 7 parameter	South Africa	WGS 84
21	Carthage	Tunisia	Clarke 1880
22	Chatham 1971	Chatham Island (New Zealand)	International
23	Chua Astro	Paraguay	International
122	Co-Ordinate System 1937 of Estonia	Estonia	Bessel 1841
24	Corrego Alegre	Brazil	International
123	Dabola	Guinea	Clarke 1880
124	Deception Island	Deception Island, Antarctica	Clarke 1880
1000	Deutsches Hauptdreicksnetz (DHDN)	Germany	Bessel
25	Djakarta (Batavia)	Sumatra Island (Indonesia)	Bessel 1841

Number	Datum	Area Maps	Ellipsoid
26	DOS 1968	Gizo Island (New Georgia Islands)	International
27	Easter Island 1967	Easter Island	International
115	EUREF 89	Europe	GRS 80
28	European 1950 (ED 50)	Austria, Belgium, Denmark, Finland, France, Germany, Gibraltar, Greece, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland	International
29	European 1979 (ED 79)	Austria, Finland, Netherlands, Norway, Spain, Sweden, Switzerland	International
108	European 1987 (ED 87)	Europe	International
125	Fort Thomas 1955	Nevis, St. Kitts, Leeward Islands	Clarke 1880
30	Gandajika Base	Republic of Maldives	International
116	GDA 94	Australia	GRS 80
32	Geodetic Reference System 1967 (GRS 67)	Worldwide	GRS 67
33	Geodetic Reference System 1980 (GRS 80)	Worldwide	GRS 80
126	Graciosa Base SW 1948	Faial, Graciosa, Pico, Sao Jorge, and Terceira Islands (Azores)	International 1924
34	Guam 1963	Guam Island	Clarke 1866

Number	Datum	Area Maps	Ellipsoid
35	GUX 1 Astro	Guadalcanal Island	International
150	Hartbeesthoek 94	South Africa	WGS 84
127	Herat North	Afghanistan	International 1924
128	Hermannskogel	Yugoslavia (Prior to 1990), Slovenia, Croatia, Bosnia and Herzegovina, Serbia	Bessel 1841
36	Hito XVIII 1963	South Chile (near 53×S)	International
37	Hjorsey 1955	Iceland	International
38	Hong Kong 1963	Hong Kong	International
1004	Hungarian Datum (HD 72)	Hungary	GRS 67
39	Hu-Tzu-Shan	Taiwan	International
40	Indian	Thailand and Vietnam	Everest (India 1830)
41	Indian	Bangladesh, India, Nepal	Everest (India 1830)
129	Indian	Pakistan	Everest (Pakistan)
130	Indian 1954	Thailand	Everest (India 1830)
131	Indian 1960	Vietnam	Everest (India 1830)
132	Indian 1975	Thailand	Everest (India 1830)
133	Indonesian 1974	Indonesia	Indonesian 1974
42	Ireland 1965	Ireland	Modified Airy
134	ISTS 061 Astro 1968	South Georgia Island	International 1924

Number	Datum	Area Maps	Ellipsoid
43	ISTS 073 Astro 1969	Diego Garcia	International
152	Japanese Geodetic Datum 2000 (JGD2000)	Japan	GRS80
44	Johnston Island 1961	Johnston Island	International
45	Kandawala	Sri Lanka	Everest (India 1830)
46	Kerguelen Island	Kerguelen Island	International
47	Kertau 1948	West Malaysia and Singapore	Everest (W. Malaysia and Singapore 1948)
135	Kusaie Astro 1951	Caroline Islands, Federated States of Micronesia	International 1924
48	L.C. 5 Astro	Cayman Brac Island	Clarke 1866
136	Leigon	Ghana	Clarke 1880
49	Liberia 1964	Liberia	Clarke 1880
113	Lisboa (DLx)	Portugal	International
50	Luzon	Philippines (excluding Mindanao Island)	Clarke 1866
51	Luzon	Mindanao Island	Clarke 1866
52	Mahe 1971	Mahe Island	Clarke 1880
53	Marco Astro	Salvage Islands	International
54	Massawa	Eritrea (Ethiopia)	Bessel 1841
114	Melrica 1973 (D73)	Portugal	International
55	Merchich	Morocco	Clarke 1880
56	Midway Astro 1961	Midway Island	International

Number	Datum	Area Maps	Ellipsoid
57	Minna	Nigeria	Clarke 1880
137	Montserrat Island Astro 1958	Montserrat, Leeward Islands	Clarke 1880
138	M'Poraloko	Gabon	Clarke 1880
58	Nahrwan	Masirah Island (Oman)	Clarke 1880
59	Nahrwan	United Arab Emirates	Clarke 1880
60	Nahrwan	Saudi Arabia	Clarke 1880
61	Naparima, BWI	Trinidad and Tobago	International
109	Netherlands	Netherlands	Bessel
117	New Zealand Geodetic Datum 2000 (NZGD2000)	New Zealand	GRS 80
31	New Zealand Geodetic Datum 1949 (NZGD 49)	New Zealand	International
1010	(New Zealand (NZGD49 7-parameter)	New Zealand	International
62	North American 1927 (NAD 27)	Continental US	Clarke 1866
63	North American 1927 (NAD 27)	Alaska	Clarke 1866
64	North American 1927 (NAD 27)	Bahamas (excluding San Salvador Island)	Clarke 1866
65	North American 1927 (NAD 27)	San Salvador Island	Clarke 1866
66	North American 1927 (NAD 27)	Canada (including Newfoundland Island)	Clarke 1866
67	North American 1927 (NAD 27)	Canal Zone	Clarke 1866

Number	Datum	Area Maps	Ellipsoid
68	North American 1927 (NAD 27)	Caribbean (Turks and Caicos Islands)	Clarke 1866
69	North American 1927 (NAD 27)	Central America (Belize, Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua)	Clarke 1866
70	North American 1927 (NAD 27)	Cuba	Clarke 1866
71	North American 1927 (NAD 27)	Greenland (Hayes Peninsula)	Clarke 1866
72	North American 1927 (NAD 27)	Mexico	Clarke 1866
73	North American 1927 (NAD 27)	Michigan (used only for State Plane Coordinate System 1927)	Modified Clarke 1866
74	North American 1983 (NAD 83)	Alaska, Canada, Central America, Continental US, Mexico	GRS 80
139	North Sahara 1959	Algeria	Clarke 1880
107	Nouvelle Triangulation Francaise (NTF) Greenwich Prime Meridian	France	Modified Clarke 1880
1002	Nouvelle Triangulation Francaise (NTF) Paris Prime Meridian	France	Modified Clarke 1880
111	NWGL 10	Worldwide	WGS 72
75	Observatorio 1966	Corvo and Flores Islands (Azores)	International

Number	Datum	Area Maps	Ellipsoid
140	Observatorio Meteorologico 1939	Corvo and Flores Islands (Azores)	International 1924
76	Old Egyptian	Egypt	Helmert 1906
77	Old Hawaiian	Hawaii	Clarke 1866
97	OldTokyo	Japan, Korea, Okinawa	Bessel 1841
78	Oman	Oman	Clarke 1880
79	Ordnance Survey of Great Britain 1936	England, Isle of Man, Scotland, Shetland Islands, Wales	Airy
80	Pico de las Nieves	Canary Islands	International
81	Pitcairn Astro 1967	Pitcairn Island	International
141	Point 58	Burkina Faso and Niger	Clarke 1880
142	Pointe Noire 1948	Congo	Clarke 1880
157	Popular Visualization	Worldwide	Popular Visualization
143	Porto Santo 1936	Porto Santo and Madeiras Islands	International 1924
1000	Potsdam	Germany	Bessel
82	Provisional South American 1956	Bolivia, Chile, Colombia, Ecuador, Guyana, Peru, Venezuela	International
36	Provisional South Chilean 1963	South Chile (near 53×S)	International
83	Puerto Rico	Puerto Rico and Virgin Islands	Clarke 1866
1001	Pulkovo 1942	Germany	Krassovsky
1012	PZ90	Russia	PZ90

Number	Datum	Area Maps	Ellipsoid
84	Qatar National	Qatar	International
85	Qornoq	South Greenland	International
1000	Rauenberg	Germany	Bessel
86	Reunion	Mascarene Island	International
112	Rikets Triangulering 1990 (RT 90)	Sweden	Bessel
1011	Rikets Triangulering 1990 (RT 90), 7 parameter	Sweden	Bessel
87	Rome 1940	Sardinia Island	International
88	Santo (DOS)	Espirito Santo Island	International
89	São Braz	São Miguel, Santa Maria Islands (Azores)	International
90	Sapper Hill 1943	East Falkland Island	International
91	Schwarzeck	Namibia	Modified Bessel 1841
144	Selvagem Grande 1938	Salvage Islands	International 1924
145	Sierra Leone 1960	Sierra Leone	Clarke 1880
146	S-JTSK	Czech Republic	Bessel 1841
1013	SK42	Russia	PZ90
1014	SK95	Russia	PZ90
92	South American 1969	Argentina, Bolivia, Brazil, Chile, Colombia, Ecuador, Guyana, Paraguay, Peru, Venezuela, Trinidad, and Tobago	South American 1969

Number	Datum	Area Maps	Ellipsoid
93	South Asia	Singapore	Modified Fischer 1960
94	Southeast Base	Porto Santo and Madeira Islands	International
95	Southwest Base	Faial, Graciosa, Pico, Sao Jorge, Terceira Islands (Azores)	International
1003	Switzerland (CH 1903)	Switzerland	Bessel
147	Tananarive Observatory 1925	Madagascar	International 1924
96	Timbalai 1948	Brunei and East Malaysia (Sarawak and Sabah)	Everest (India 1830)
1015	Tokyo	Japan	Bessel 1841
98	Tristan Astro 1968	Tristan da Cunha	International
99	Viti Levu 1916	Viti Levu Island (Fiji Islands)	Clarke 1880
148	Voirol 1874	Tunisia/Algeria	Clarke 1880
149	Voirol 1960	Algeria	Clarke 1880
100	Wake-Eniwetok 1960	Marshall Islands	Hough
101	World Geodetic System 1960 (WGS 60)	Worldwide	WGS 60
102	World Geodetic System 1966 (WGS 66)	Worldwide	WGS 66
103	World Geodetic System 1972 (WGS 72)	Worldwide	WGS 72
104	World Geodetic System 1984 (WGS 84)	Worldwide	WGS 84

Number	Datum	Area Maps	Ellipsoid
105	Yacare	Uruguay	International
106	Zanderij	Surinam	International
158	CH1903+ datum for Switzerland	Switzerland	Bessel 1841
159	Updated Schwarzeck	Namibia	Modified Bessel
160	SIRGAS 2000	Central and South America	GRS 1980
161	NOAA Sphere		GCS Sphere
162	Japanese Geodetic Datum 2011 (JGD 2011)	Japan	GRS 80
163	Timbalai 1948, Everest 1968	Asia - Brunei and East Malaysia	Everest 1830 (1967 Definition)
164	Geodetic Datum of Malaysia (GDM2000)	Malaysia	GRS 1980
165	MAGNA-SIRGAS	Colombia	GRS 1980
166	Aratu	Brazil	International 1924
167	Barbados	Barbados	Clarke 1880
168	Beduaram	Niger	Modified Clarke 1880
169	Conakry 1905	Guinea	Modified Clarke 1880
170	Dealul Piscului 1930	Romania	International 1924
171	Douala	Cameroon	International 1924
172	Final Datum 1958	Iran	Clarke 1880
173	Makassar	Indonesia	Bessel 1841
174	Manoca	Cameroon	Modified Clarke 1880

Number	Datum	Area Maps	Ellipsoid
1016	KKJ	Finland	International 1924
1017	Xian 1980	China	IAG 1975
1018	Lithuanian Pulkovo 1942	Latvia	Lithuania Krassovsky
1019	Belgian 1972, 7 parameter	parameter	Belgium International 1924
1020	S-JSTK (Ferro meridian)	Czech Republic	Bessel 1841
1021	Serbia datum MGI 1901	Republic of Serbia	Bessel 1841
1022	North Sahara 7-parameter	Algeria	Clarke 1880
1023	HD72 (Hungarian Datum of 1972) - UPDATED	Hungary	GRS 67
1024	S - JSTK (Czech)	Czech Republic	Bessel 1841
1025	JTSK03 (Slovak Republic)	Slovakia	Bessel 1841
1026	NGO_1948	Norway onshore	Modified Bessel
1027	Palestine_1923	Asia - Middle East - Israel, Jordan and Palestine onshore	Clarke 1880 (Benoit)

Units

The following table lists the available coordinate units and the number used to identify the unit in the MapInfoCoordinateSystemSet.xml file:

Number	Units
6	Centimeters
31	Chains
3	Feet (also called International Feet)*
2	Inches
1	Kilometers
30	Links
7	Meters
0	Miles
5	Millimeters
9	Nautical Miles†
32	Rods
8	US Survey Feet (used for 1927 State Plane)‡
4	Yards
13	Degree

* One International Foot equals exactly 30.48 cm.

† One Nautical Mile equals exactly 1852 meters.

‡ One US Survey Foot equals exactly 12/39.37 meters, or approximately 30.48006 cm.

Coordinate System Origin

The origin is the point specified in longitude and latitude from which all coordinates are referenced. It is chosen to optimize the accuracy of a particular coordinate system. As we move north from the origin, Y increases. X increases as we move east. These coordinate values are generally called northings and eastings.

For the Transverse Mercator projection the origin's longitude defines the central meridian. In constructing the Transverse Mercator projection a cylinder is positioned tangent to the earth. The central meridian is the line of tangency. The scale of the projected map is true along the central meridian.

In creating a Hotine Oblique Mercator projection it is necessary to specify a great circle that is not the equator nor a meridian. MapInfo does this by specifying one point on the ellipsoid and an azimuth from that point. That point is the origin of the coordinate system.

Standard Parallels (Conic Projections)

In conic projections a cone is passed through the earth intersecting it along two parallels of latitude. These are the standard parallels. One is to the north and one is to the south of the projection zone. To use a single standard parallel specify that latitude twice. Both are expressed in degrees of latitude.

Oblique Azimuth (Hotine Oblique Mercator)

When specifying a great circle (Hotine Oblique Mercator) using a point and an azimuth (arc), the azimuth is called the Oblique Azimuth and is expressed in degrees.

Scale Factor (Transverse Mercator)

A scale factor is applied to cylindrical coordinates to average scale error over the central area of the map while reducing the error along the east and west boundaries. The scale factor has the effect of recessing the cylinder into the earth so that it has two lines of intersection. Scale is true along these lines of intersection.

You may see the scale factor expressed as a ratio, such as 1:25000. In this case it is generally called the scale reduction. The relationship between scale factor and scale reduction is:

scale factor = 1-scale reduction

In this case the scale factor would be 1-(1/25000) or 0.99996.

False Northings and False Eastings

Calculating coordinates is easier if negative numbers aren’t involved. To eliminate this problem in calculating State Plane and Universal Transverse Mercator coordinates, it is common to add measurement offsets to the northings and eastings. These offsets are called False Northings and False Eastings. They are expressed in coordinate units, not degrees. (The coordinate units are specified by the Units parameter.)

Range (Azimuthal Projections)

The range specifies, in degrees, how much of the earth you are seeing. The range can be between 1 and 180. When you specify 90, you see a hemisphere. When you specify 180 you see the whole earth, though much of it is very distorted.

Polyconic Projection

The following description is copied from *Map Projections – A Working Manual*, USGS Professional Paper 1395, by John P. Snyder.

The Polyconic projection, usually called the American Polyconic in Europe, achieved its name because the curvature of the circular arc for each parallel on the map is the same as it would be following the unrolling of a cone which had been wrapped around the globe tangent to the particular parallel of latitude, with the parallel traced onto the cone. Thus, there are many ("poly-") cones involved, rather than the single cone of each regular conic projection.

The Polyconic projection is neither equal-area nor conformal. Along the central meridian, however, it is both distortion free and true to scale. Each parallel is true to scale, but the meridians are lengthened by various amounts to cross each parallel at the correct position along the parallel, so that no parallel is standard in the sense of having conformality (or correct angles), except at the central meridian. Near the central meridian, distortion is extremely small.

This projection is not intended for mapping large areas. The conversion algorithms used break down when mapping wide longitude ranges. For example, the sample table WORLD.TAB may exhibit anomalies if reprojected using Polyconic.

Datum Conversion

When converting coordinates from one datum to another, MapInfo has used the Molodensky (3-parameter) and Bursa-Wolf (7-parameter) methods. These are general-purpose methods that can convert coordinates from any datum to any other datum.

After the NAD 83 datum was introduced, NOAA developed a program called NADCON, which stands for North American Datum CONversion. This is a very specialized program that converts coordinates only from NAD 27 to NAD 83 and vice versa. For this specialized task, it's much more accurate than the Molodensky general-purpose method; NADCON is accurate to about 0.1 meter, and Molodensky is accurate to only 10–30 meters. Most U.S. government agencies, including the Census Bureau, have standardized on NADCON for converting between NAD 27 and NAD 83.

Beginning with MapInfo 4.1.2, the NADCON algorithm is used to convert coordinates between NAD 27 and NAD 83 if those coordinates lie within the areas covered by NADCON (United States, Puerto Rico, and the Virgin Islands). If the coordinates lie outside those areas, or if they use datums other than NAD 27 or NAD 83, MapInfo uses the Molodensky or Bursa-Wolfe conversion methods.

Due to the file access required, the NADCON conversion method can be slightly slower than the Molodensky method. If you want to turn off the NADCON conversion, remove the *.las and *.los files from the MapXtreme program or the MapXtreme Common directory.

Custom Datums

A datum is a mathematical description of the earth's shape and orientation. Because the earth's shape is not uniform, there are many different local datums used in different parts of the world. These local datums provide a close approximation to the earth's surface in a particular area.

Each Earth coordinate system uses a specific datum to approximate the earth's surface. If two coordinate systems use different datums, then Precisely's mapping products must perform a datum transformation when it converts coordinates from one coordinate system to the other. Precisely uses the Bursa-Wolfe datum transformation method, which is generally accurate to within 10 meters. (When the conversion is between two coordinate systems that use the same datum, no datum transformation is performed, and the results are generally accurate to within 0.1 meter.)

Defining Custom Datums

Most coordinate systems use one of Precisely's predefined datums, listed in this appendix. If you need to use a datum that isn't in the list, and you know what the datum's mathematical parameters are, then you can define the coordinate system using a custom datum. MapInfo uses the following information to define a datum:

- An ellipsoid, also called a spheroid. This is an ellipse rotated around its minor axis to form a three-dimensional surface. The ellipsoid is described by two mathematical parameters: the length, in meters, of its semi-major axis (denoted by the letter a) and its degree of flattening (denoted by the letter f). MapInfo supports over 40 predefined ellipsoids, which are listed in the next table.
- Three shift parameters specifying the distance, in meters, to shift the ellipsoid along each of its axes. These parameters are usually denoted by dX, dY, and dZ. You may also see them denoted by DX, DY, and DZ, or by u, v, and w.
- Three rotation parameters specifying the angle, in arc-seconds, to rotate the ellipsoid around each of its axes. These parameters are usually denoted by EX, EY, and EZ. You may also see them denoted by eX, eY, and eZ, or by e, y, and w.
- A scale correction factor specifying the amount, in parts per million, to adjust the size of the ellipsoid. This parameter is denoted by the letter m, or sometimes k.

- The longitude of the prime meridian, in degrees east of Greenwich. The prime meridian specifies which location on earth is assigned longitude 0×. Most datums use Greenwich as the prime meridian, so this parameter is usually zero. However, some datums use a different location as the prime meridian. For example, the NTF datum uses Paris as its prime meridian, which is 2.33722917 degrees east of Greenwich. If you use the NTF datum in a coordinate system, all longitudes in that coordinate system are relative to Paris instead of Greenwich.

You can define a custom datum in any coordinate system definition. Use datum number 9999 followed by the datum parameters, in this order:

```
9999, EllipsoidNumber, dX, dY, dZ, EX, EY, EZ, m, PrimeMeridian
```

Some datums specify only an ellipsoid and shift parameters (dX, dY, dZ), with no rotation parameters, scale correction, or prime meridian. In those cases, you can use datum number 999 instead of 9999, to simplify the definition:

```
999, EllipsoidNumber, dX, dY, dZ
```

The ellipsoid number must be chosen from the following list. Currently, there is no way to define a custom ellipsoid. If you need to use an ellipsoid that does not appear on this list, please notify MapInfo Technical Support so that we can add your ellipsoid to a future MapInfo release.

Number	Ellipsoid	a	1/f
9	Airy 1930	6377563.396	299.3249646
13	Airy 1930 (modified for Ireland 1965)	6377340.189	299.3249646
51	ATS7 77	6378135.0	298.257
2	Australian	6378160.0	298.25
10	Bessel 1841	6377397.155	299.1528128
35	Bessel 1841 (modified for NGO 1948)	6377492.0176	299.15281
14	Bessel 1841 (modified for Schwarzeck)	6377483.865	299.1528128
36	Clarke 1858	6378293.639	294.26068
7	Clarke 1866	6378206.4	294.9786982

Number	Ellipsoid	a	1/f
8	Clarke 1866 (modified for Michigan)	6378450.047484481	294.9786982
6	Clarke 1880	6378249.145	293.465
15	Clarke 1880 (modified for Arc 1950)	6378249.145326	293.4663076
30	Clarke 1880 (modified for IGN)	6378249.2	293.4660213
37	Clarke 1880 (modified for Jamaica)	6378249.136	293.46631
16	Clarke 1880 (modified for Merchich)	6378249.2	293.46598
38	Clarke 1880 (modified for Palestine)	6378300.79	293.46623
39	Everest (Brunei and East Malaysia (Sabah and Sarawak))	6377298.556	300.8017
11	Everest (India 1830)	6377276.345	300.8017
40	Everest (India 1956)	6377301.243	300.80174
50	Everest (Pakistan)	6377309.613	300.8017
17	Everest (W. Malaysia and Singapore 1948)	6377304.063	300.8017
48	Everest (West Malaysia 1969)	6377295.664	300.8017
18	Fischer 1960	6378166.0	298.3
19	Fischer 1960 (modified for South Asia)	6378155.0	298.3
20	Fischer 1968	6378150.0	298.3
21	GRS 67	6378160.0	298.247167427
0	GRS 80	6378137.0	298.257222101

Number	Ellipsoid	a	1/f
5	Hayford	6378388.0	297.0
22	Helmert 1906	6378200.0	298.3
23	Hough	6378270.0	297.0
31	IAG 75	6378140.0	298.257222
41	Indonesian	6378160.0	298.247
4	International 1924	6378388.0	297.0
49	Irish (WOFO)	6377542.178	299.325
3	Krassovsky	6378245.0	298.3
32	MERIT 83	6378137.0	298.257
33	New International 1967	6378157.5	298.25
43	NWL 10D	6378135.0	298.26
42	NWL 9D	6378145.0	298.25
44	OSU86F	6378136.2	298.25722
45	OSU91A	6378136.3	298.25722
46	Plessis 1817	6376523.0	308.64
54	Popular Visualization	6378137.0	0.0
52	PZ90	6378136.0	298.257839303
24	South American	6378160.0	298.25
12	Sphere	6370997.0	0.0
47	Struve 1860	6378297.0	294.73
34	Walbeck	6376896.0	302.78
25	War Office	6378300.583	296.0
26	WGS 60	6378165.0	298.3

Number	Ellipsoid	a	1/f
27	WGS 66	6378145.0	298.25
1	WGS 72	6378135.0	298.26
28	WGS 84	6378137.0	298.257223563

The shift and rotation parameters describe the ellipsoid's orientation in space, as compared to the WGS 84 datum. It's important to make sure that these parameters have the correct signs (positive or negative). Usually, a document describing a local datum will list the parameters required to convert coordinates from the local datum to WGS 84. (This is the same as saying that the parameters were derived by subtracting the local datum from WGS 84.) In that case, you can use the parameters exactly as they appear in the document. However, if you have a document that lists parameters for converting coordinates in the opposite direction — from WGS 84 to the local datum — then you must reverse the signs of the shift, rotation, and scale correction parameters.

It's also very important to list the parameters in the correct order. Some documents list the rotation parameters with EZ first, like this: EZ, EY, EX. In those cases, you must reverse the order of the rotation parameters when defining the custom datum. This is especially easy to overlook when your document uses Greek letters to denote the parameters. If the document lists the parameters in order as w, y, e, then you must reverse their order in the custom datum definition.

Here's an example of a local datum description (we'll call it LD-2) as it might appear in a technical article:

LD-2 ellipsoid: Krassovsky

a	6378245.0 m
f	1 / 298.3

Converting from LD-2 to WGS 84

u	+24 m
v	–123 m
w	–94 m

w	+0.13
y	+0.25
e	-0.02
m	+1.1 Þ 10-6

This datum uses the Krassovsky ellipsoid, which is number 3 in the ellipsoid table above. We do not need to reverse the signs of the parameters, since they describe a conversion from the local datum to WGS 84. However, the rotation parameters are listed with w first, so we must reverse their order in the custom datum definition:

```
9999, 3, 24, -123, -94, -0.02, 0.25, 0.13, 1.1, 0
```

Here's a final example, LD-3, that provides only the ellipsoid and shift parameters:

LD-3 ellipsoid: Clarke 1880

a	6378249.145 m
f	1 / 293.465

WGS 84 minus local datum LD-3

dX	-7 m
dY	36 m
dZ	225 m

This datum uses the Clarke 1880 ellipsoid, which is number 6 in the ellipsoid table above. We do not need to reverse the signs of the parameters or worry about the order of the rotation parameters (since they aren't present). In this case, you can use datum number 999 instead of 9999 in the custom datum definition. These two definitions are equivalent, and you can use either one:

```
999, 6, -7, 36, 225
9999, 6, -7, 36, 225, 0, 0, 0, 0, 0
```

As with the other custom datum definitions, you would insert one of these definitions in place of the datum number in a MapInfoCoordinateSystemSet.xml line, as follows:

```
"Longitude / Latitude (LD-3)", 1, 999, 6, -7, 36, 225

<gml:srsName> UTM Zone 30 (LD-3)</gml:srsName>
```

```
    <gml:metaDataProperty>
      <gml:category> Universal Transverse Mercator (LD-3)</gml:category>
    </gml:metaDataProperty>
    <gml:srsID>
      <gml:name gml:codeSpace="mapinfo">coordsys 8,999,6,-7,36,225,7,-
3,0,0.9996,500000,0</gml:name>
    </gml:srsID>
```

National Transformation v. 2 (NTv2)

The National Transformation v. 2 (NTv2) algorithm and grid shift file format, developed by the Geodetic Survey Division of Geomatics Canada, enables you to convert data from the NAD 27 reference system to the NAD 84 reference system. Grid shift files used with the algorithm contain one or more rectangular grids that indicate the coordinate differences between NAD 27 and NAD 83.

The National Transformation was originally designed to convert Canadian data from NAD 27 to NAD 84. This format has been adopted for datum conversion in several other countries, such as Australia, Brazil, Canada, France, Germany, New Zealand, Portugal, South Africa, Spain, Switzerland, United Kingdom and Venezuela.

Binary grid shift files for the following countries have been added in the MapXtreme Common Files folder, i.e., C:\Program Files\Common Files\MapInfo\MapXtreme\9.x.x.

- Canada
- Australia
- New Zealand
- Germany
- UK
- Spain
- Brazil
- Netherlands

In addition to the algorithm and grid shift files, an XML configuration file, NTv2.xml, has been installed to enable you to extend the NTv2 algorithm to support additional datum transformations. You can add new grid shift files to the configuration file, or you can turn the NTv2 algorithm on or off for particular grid shift files. The NTv2.xml file is located in the MapXtreme Common Files folder.

The NTv2 algorithm and grid shift files for Canada are protected under the following copyright:

© 1995 Her Majesty the Queen in Right of Canada, represented by the Minister of Natural Resources.

The next sections describe the supported transformations for each country and how to use the configuration file.

Canada

The following grid shift files are included for Canada:

NTV2_0.GSB–used for converting NAD 1927 to NAD 1983

MAY76V20.GSB–used for converting NAD 1927 (Definition 1976) to NAD 1983 for Ontario

These files convert between the NAD 1927 or NAD 1927 (Definition 1976) datums and the NAD 1983 datum. The NAD 1927 datum (Definition 1976) coordinate system is a readjustment of NAD 1927 for Ontario.

Detailed information about the algorithm, software, and grid shift files can be downloaded from the Geodetic Survey Division of Geomatics Canada web site:

www.geod.nrcan.gc.ca/index_e.php

Australia

The following grid shift files are included for Australia:

A66_National.gsb–used for converting AGD 1966 to GDA94.

National_84.gsb–used for converting AGD 1984 to GDA94

The Australian grid shift files convert between either the AGD 1966 or AGD 1984 datums and the GDA 1994 datum.

Detailed information about conversion and grid shift files can be downloaded from the Intergovernmental Committee on Surveying and Mapping (ICSM) web site:

www.icsm.gov.au/icsm/gda/gdatm/

New Zealand

The following grid shift file is included for New Zealand:

nzgd2kgrid0005.gsb–used for converting the NZGD49 datum to the NZGD2000 datum.

The New Zealand grid shift file converts from the NZGD49 datum to the NZGD2000 datum.

Detailed information and an online converter can be found at the Land Information New Zealand (LINZ) web site:

www.linz.govt.nz/geodetic/conversion-coordinates/online-conversion-service

Germany

The following grid shift file is included for Germany:

BETA2007.gsb–used for conversion from DHDN datum to ETRS89

For more information and to download the official grid shift file, see the Information and Service System for European Coordinate Reference Systems (CRS EU) web site:

www.crs-geo.eu/

United Kingdom

The following grid shift file is included for United Kingdom:

OSTN02_NTv2.gsb – used for conversion from OSGB36 (Ordnance Survey Great Britain 1936) datum to World Geodetic System 1984 (WSG84) and ETRS89 datums.

Spain

The following grid shift file is included for Spain:

BALEARES.gsb and PENINSULA.gsb – used for converting European 1950 (ED 50) to WGS84 and ETRS89 datums.

Brazil

The following grid shift file is included for Brazil:

CA61_003.gsb – used for converting Corrego Alegra to WGS84 and SIRGAS 2000 datums. SAD69_002.gsb – used for converting South American 1969 to WGS84 and SIRGAS 2000 datums.

Netherlands

The following grid shift file is included for Netherlands:

rdtrans2008.gsb – used for conversion from Netherlands datum to WGS84 and ETRS89 datums.

Configuration File

The NTv2 configuration file (NTv2.xml) is an XML file that you can edit to add new grid shift files for other datum conversions and turn NTv2 on or off for particular grid shift files.

To add new grid shift files, you must specify three parameters:

- grid shift file name
- source datum
- destination datum

You can define a datum in various ways. For example, when both the source and destination datums are defined in MapXtreme, the datums are defined in the configuration file by the numbers assigned to them in the MapXtremeCoordinateSystemSet.xml file. In this example for New Zealand, the XML in the configuration file will look like this (application of bold text is for emphasis only):

```
- <NTv2Conversion>
  <Description>NTv2 Conversion for New Zealand</Description>
  <GridFile>NZGD2KGRID0005.GSB</GridFile>
  <Enabled>true</Enabled>
- <SourceDatum>
  <DatumID>1010</DatumID>
  </SourceDatum>
- <DestinationDatum>
  <DatumID>117</DatumID>
  </DestinationDatum>
  </NTv2Conversion>
- <NTv2Conversion>
```

Sometimes, however, datums are not that easily defined. You may need to supply datum shift values (and sometimes even ranges for them) in order to define a datum. For example, the sample XML below shows the definition of the NAD 27 datum for Canada. Here, the ellipsoid, shift values for x,y,z, and their ranges have been used to define the datum (bold text is for emphasis only):

```
- <NTv2Conversion>
  <Description>NAD 1927 to NAD 1983 conversion for Canada</Description>
  <GridFile>NTV2_0.GSB</GridFile>
  <Enabled>true</Enabled>
- <SourceDatum>
  <EllipsoidID>7</EllipsoidID>
  <ShiftX>-8</ShiftX>
  <ShiftY>150.5</ShiftY>
  <ShiftZ>186</ShiftZ>
  <dShiftX>17</dShiftX>
  <dShiftY>19.5</dShiftY>
  <dShiftZ>8</dShiftZ>
  </SourceDatum>
```

```
- <DestinationDatum>
   <DatumID>74</DatumID>
   </DestinationDatum>
   </NTv2Conversion>
```

Information on Coordinate Systems and Projections

The first three publications listed are relatively short pamphlets. The last two are substantial books. We've also given addresses and phone numbers for the American Congress of Surveying and Mapping (the pamphlets) and the U.S. Geological Survey (the books).

- American Cartographic Association. Choosing a World Map—Attributes, Distortions, Classes, Aspects. Falls Church, VA: American Congress on Surveying and Mapping. Special Publication No. 2. 1988.
- American Cartographic Association. Matching the Map Projection the Need. Falls Church, VA: American Congress on Surveying and Mapping. Special Publication No. 3. 1991.
- American Cartographic Association. Which Map is Best? Projections for World Maps. Falls Church, VA: American Congress on Surveying and Mapping. Special Publication No. 1. 1986.
- John P. Snyder. Map Projections—A Working Manual. Washington: U.S. Geological Survey Professional Paper 1395. 1987
- John P. Snyder and Philip M. Voxland. An Album of Map Projections. Washington: U.S. Geological Survey Professional Paper 1453. 1989.
- Contact Information
- American Congress on Surveying and Mapping, 5410 Grosvenor Lane, Suite 100, Bethesda, MD 20814 2212; (301) 493-0200
- Earth Science Information Center, U.S. Geological Survey, 507 National Center, Reston, VA 22092; (703) 860-6045 or (800) USA-MAPS
- Peter H. Dana of the Department of Geography, University of Texas at Austin has also put up a website for explanations of Map projections, Geodetic Datums, and Coordinate systems. It is a valuable as many of these explanations were also presented using MapInfo Professional. The materials may be used for study, research, and education, but please credit the author:
- Peter H. Dana, The Geographer's Craft Project, Department of Geography, The University of Texas at Austin.

For Geodetic Datum information and explanations, go to:

www.colorado.edu/geography/gcraft/notes/datum/datum.html

For Information on Coordinate systems and other principles, go to:

www.colorado.edu/geography/gcraft/notes/coordsys/coordsys.html

For Information on Map Projections go to:

www.colorado.edu/geography/gcraft/notes/mapproj/mapproj.html

I – User-Defined Metadata

This appendix contains information on user-defined metadata support for TableInfoServer queries.

In this appendix

Metadata and the MapCatalog

When working with spatial database queries, the MapCatalog has long been used as the sole source of metadata when trying to infer accurate definitions for FeatureGeometry and Style columns in the query result set. In some cases, however, the MapCatalog may not be a convenient, appropriate, or reliable source. For example:

- Views–When defining view tables in the remote database server, the onus is upon the database administrator to add a corresponding entry into the MapCatalog. This may be inconvenient, and there is a risk of inaccuracy in the content.
- Stored Procedures–As a potentially dynamic generator of spatial result sets, it may not be feasible to place a single, static entry into the MapCatalog that accurately represents all of the result sets capable of being generated as output to invocations of the stored procedure.

> ⓘ Stored Procedure queries are currently supported for SQL Server only.

- Tables containing multiple spatial columns–Although spatial databases may allow for tables to contain more than one spatial column, the MapCatalog schema currently only permits the metadata to be defined for a single spatial column of a table.
- Complex queries–Even when the MapCatalog contains accurate information for base tables referenced in complex queries, the inferencing logic may be unreliable for several reasons:
  - It may be difficult for the metadata inferencing logic to properly locate the correct metadata (e.g., complex join queries).
  - The MapCatalog metadata may be inaccurate due to functions applied on the geometries (e.g., coordinate system transformations, buffering points, convex hull aggregations, etc.)
  - The inferencing logic may not know how to properly reconcile metadata identified for two or more base tables (e.g., UNION queries).

User-Defined Metadata Support for TableInfoServer Queries

The MapInfo.Data.TableInfoServer class contains properties that enable you to define metadata for FeatureGeometry and Style columns in a table. This user-defined metadata, or column hints, identify these columns in the absence of a MapCatalog.

ColumnHints Property

The ColumnHints property added to TableInfoServer gives users an explicit means of providing user-defined metadata to influence the proper identification of FeatureGeometry and Style columns in a query result set. These "hints" are principally intended for providing complete metadata in cases where MapCatalog metadata could not be located; however, these hints may also be used as potentially sparse definitions to selectively override components of the MapCatalog metadata. In either case, user-defined metadata is considered definitive, so users are cautioned to use column hints only when necessary, and when the details of the query results are well understood.

The ColumnHints property is defined as a Columns collection. After creating an instance of the collection, and inserting Column instances for each hint within it, this collection is assigned to the ColumnHints property; for example, for a TableInfoServer variable named tblInfoSrv…

```
[C#]
Columns hints = new Columns();
// insert Column instances into the collection to serve as hints (see below)
tblInfoSrv.ColumnHints = hints;
```

Column instances for types other than FeatureGeometry and Style may be added into the Columns collection for the hints, but these other column definitions will currently be ignored.

Applying a FeatureGeometry Column Hint

A FeatureGeometry column hint may be used to indicate that a named column within the query result contains spatial values to be interpreted as feature geometries. For example, if a query returns a column named geo that contains spatial objects (e.g., a SpatialWare ST_Spatial binary, or an Oracle Spatial MDSYS.SDO_GEOMETRY structure), we can identify this column as such by constructing a hint and adding it into the Columns collection (from above) as follows:

```
[C#]
GeometryColumn geoCol = new GeometryColumn("geo");
...
hints.Add(geoCol);
```

The ellipsis in the code fragment above indicates that additional properties of the GeometryColumn instance will also need to be set before completing the hint and adding it into the Columns collection. FeatureGeometry hints are principally intended to be used for explicitly providing the same information that the MapCatalog provides, specifically:

- Coordinate System
- FeatureGeometry Type
- Default View

- Default Style

Taking all of these properties into account, a more complete example of a FeatureGeometry hint might look something like this:

```
[C#]
CoordSysFactory cfs = new CoordSysFactory();
CoordSys cs = cfs.CreateFromMapBasicString("Earth Projection 1, 0");
GeometryColumn geo = new GeometryColumn("geo", cs);
geo.PredominantGeometryType = GeometryType.Polygon;
geo.DefaultView = new DRect(-74, 40, -70, 44);
geo.DefaultStyle = new MapInfo.Styles.AreaStyle(
new MapInfo.Styles.SimpleLineStyle(new LineWidth(2.0,
MapInfo.Styles.LineWidthUnit.Pixel)),
new MapInfo.Styles.SimpleInterior(10));
hints.Add(geo);
```

If a MapCatalog entry is known to exist which provides default metadata for the spatial column, a "sparse" hint can be used as a means of overriding specific properties. Usually these would be preference variety properties such as default view and default style, although any set of the properties can be overridden depending upon the unique requirements of the complex query.

ColumnHints are not intended to be used for (x,y) data tables. For these, SpatialSchemaXY should continue to be used. When a SpatialSchemaXY is applied to a table, any associated ColumnHints are disregarded.

Since a MapXtreme table currently supports only a single FeatureGeometry column, the Columns collection is expected to contain no more than a single FeatureGeometry column definition; however, there's nothing in the Columns collection that precludes users from adding in as many as they'd like. In this case, the last non-sparse value encountered for each property is the one that will be honored in compiling the final metadata definition.

Applying a Style Column Hint

A Style column hint may be applied as a way of indicating that a column within the query result contains values (notably, MapBasic style strings) used to create Style instances for each feature, which correspond to the associated FeatureGeometry within the feature. For example, if a query returns a column named mb_style that contains MapBasic style strings, we could construct the hint and add it into the hints collection (from above) as follows:

```
[C#]
Column styleCol = new Column("mb_style", MIDbType.Style);
hints.Add(styleCol);
```

Since a table currently supports only a single Style column, the hints collection is expected to contain no more than a single Style column definition; however, there's nothing in the Columns collection that precludes users from adding in as many as they'd like. In this case, the last one inserted is the one that will be honored.

Style column hints are not required within the hints collection. If the query does not contain a column containing style information, this hint can be excluded.

KeyType and KeyColumns Properties

Each table must have a "key" that provides a unique value for each feature (row) within the table. While not supplied by the MapCatalog, this is another important piece of metadata that is inferred for remote database tables by looking at things like primary key definitions.

Many of the content sources listed above, as well as tables containing only composite primary keys, also present challenges for inferring a unique key to use. In these situations, the KeyType and KeyColumns properties provided through the base TableInfo class can be used to provide explicit instructions. For example, consider a database view of customer information that contains a column named CUSTID that is known to contain unique and non-null values. We can direct this column to be used as the key for the resulting MapXtreme table as follows:

```
[C#]
tblInfoSrv.KeyType = KeyType.Explicit;
StringCollection keyColumns = new StringCollection();
keyColumns.Add("CUSTID");
tblInfoSrv.KeyColumns = keyColumns;
```

In this example, the StringCollection class resides in the System.Collections.Specialized namespace.

Composite keys are allowable and would be defined by adding two or more column names into the string collection.

Explicit key definitions may also be used as overriding definitions in cases where the database table contains two or more candidates, and the developer has a specific preference for which gets used.

Workspace Persistence

User-defined metadata constructs for column hints and key definitions are persisted as part of the data source definition tag within a workspace file (.mws).

J – Migrating to MapXtreme

This appendix is for existing users of MapX to become familiar with the .NET-based object model in MapXtreme and how it differs from the architecture of MapX.

In this appendix:

Comparing MapXtreme's Object Model to MapX

If you have used MapX and MapXtreme for Windows, you will find similarities and differences in the way they work from MapXtreme. This section describes some of the major differences between MapXtreme and previous versions of our APIs.

Specific Object Model Implementation Differences

The Map object from MapX is now represented by 3 or 4 classes. The Map contains the layers, area of interest (view), Adornments, Legends, regardless of how it is being viewed. The MapControl holds the Map object and acts as a control for the Map and tools which interact with the Map. There is a MapControl for both Windows Forms (desktop) applications and ASP.NET (web) applications. The MapExport class can be used to export a Map to a file or stream. The Session holds the collection of Maps, Selections, and Tables. Cartographic scale is now fully supported. Label layers can be used to better control the position of labels. Group Layers allow for treating many layers as one in the layer hierarchy.

The MapX Datasets concept has been replaced by a more flexible set of options in the MapInfo.Data namespace. MapXtreme uses ADO.NET for data access. ADO.NET allows a wide variety of data formats to be accessed with very similar code, facilitating easy transition between data sources. We have added a wider variety of cursors for forward and backward access through a database table.

MapXtreme uses a table-centric data model. In previous versions of our API, maps were made up of layers, but there was no specific table class. In the MapXtreme object model, a table is the central object used for accessing data. The Session.Catalog is used to open and enumerate tables. Layers can reference tables, but all table specific methods were moved from Layer to table. Also, Searching is done on tables in MapXtreme and not layers.

Thematics that modify styles, such as ranged themes and individual value themes, no longer use their own layer. In the previous models such a theme occupied a separate layer and worked independently of other layers of a map. Object themes still retain their own layer and can now be displayed in any way you choose, independent of the original map objects on which the theme was based.

Geometry has been changed to be a hierarchical model with much deeper access to properties of the objects. Styles are also represented in a hierarchy.

A larger quantity of map tools have been added for you to use. Their extensibility and functionality has been enhanced to make it much easier to develop customized tools. There are also tools for use in web applications that are also extensible.

For web applications, the MapXBroker and MapXServer are no longer needed. Instead MapXtreme uses standard COM+ to pool the Session object.

Application state is managed via serialization and the ASP.NET state server. See Chapter 6 Understanding State Management for more details about state management and serialization.

The following table lists the objects in the MapX 5.0 object model and the equivalent in the current object model for MapXtreme. Note that with the re-architecture of such an extensive product as MapXtreme, the equivalent may approximate.

MapX 5.0	MapXtreme
AffineTransform	AffineTransform It is non-mutable (no set). Use CoordSysFactory.CreateAffineTransform to create one.
AllFeaturesConstraint	
Annotation	Obsolete. New Adornments class. The MapControl contains adornments derived from Control
Annotations	Obsolete.
BindLayer	Obsolete. Replacement is in Table.AddColumns() arguments.
BitmapSymbol	BitmapPointStyle
BitmapSymbols	
BoundsConstraint	
CoordSys	CoordSys. It is non-mutable (no set). Use CoordSysFactory to create one.

MapX 5.0	MapXtreme
Dataset	Obsolete. Replacements: • Expression columns: Table.AddColumns() allows you to create expression columns. • Manual Data binding (both dynamic or static): Table.AddColumns() using a Table opened w/ an TableInfoClient which points to a ADO.NET DataTable. • Auto Data Binding: Handled by Geodictionary functionality. • Non ADO.NET External Data • XY Binding: SpatialSchemaXY applied to the TableInfo.SpatialSchema property. • PointRef Binding: SpatialSchemaPointRef applied to the TableInfo.SpatialSchema property. RowValues: MIDataReader, MIScrollableReader, see MICommand. Refresh: Table.RefreshColumns()
Datasets	Obsolete. See Dataset.
Datum	Datum It is non-mutable (no set). Use CoordSysFactory.CreateDatum() to create one.
Feature	Geometry, style and key information is a Feature.
Features	See Feature.
FeaturesConstraint	

MapX 5.0	MapXtreme
FeatureFactory	Table-level object processing • Buffer: For one feature, it is at the Geometry level (FeatureGeometry.Buffer). For multiple features, use FeatureProcessor.Buffer. • Combine: For one feature, it is at the Geometry level (FeatureGeometry.Combine). For multiple features, use FeatureProcessor.Buffer. Intersect: For one feature, it is at the Geometry level (FeatureGeometry.Intersect). For multiple features, use FeatureProcessor.Intersect.
Field	
Fields	
Find	Find
FindFeature	FindResult
FindMatch	FindCloseMatch, FindAddressRange
FindMatches	FindCloseMatchEnumerator, FindAddressRangeEnumerator
FindResult	FindResult
Geoset	Obsolete. Similar functionality: MapLoader
Geosets	Obsolete. MapXtreme opens geosets.
Graphic	Obsolete.
IndividualValueCategory	IndividualValueThemeBin
IndividualValueCategories	ModifierThemeBins
Label	LabelLayer, LabelSource
Labels	LabelLayer, LabelSource

MapX 5.0	MapXtreme
LabelProperties	LabelProperties
Layer	UserDrawLayer, LabelLayer, FeatureLayer, ObjectThemeLayer, GroupLayer... • AddFeature: The current equivalent is to bind a Geometry to the geometry column during an insert operation using MICommand. • AllFeatures • AutoLabel: This no longer exists. The equivalent is to have the LabelSource.DefaultLabelProperties.Enabled property set to true. • Begin/EndAccess: Not addressed • Bounds: FeatureLayer.Bounds • ClearCustomLabels: Not addressed • ClippedBounds: Obsolete. Use Map.SetView() • CoordSys • Datasets: Obsolete. A layer has a Table (FeatureLayer.Table.) • DeleteFeature: The equivalent is to delete row through MICommand. • DrawLabelsAfter: This concept is deprecated. Similar functionality is to use a LabelLayer instance after each Layer. The only drawback is that each LabelLayer works off its own label cache so the labels will not interact between layers. • Editable: FeatureLayer.Editable • FeatureIDFromFeatureName: Equivalent would be an SQL command through MICommand.

MapX 5.0	MapXtreme
Layer (continued)	• Find: Find • GetDrillDownFeaturesByID: Not addressed • GetFeatureByID: Equivalent functionality is a key cursor using MICommand. • Invalidate: IMapLayer.Invalidate • KeyField: Obsolete. Equivalent functionality is MICommand. • LabelAtPoint • LabelProperties: LabelSource.DefaultLabelProperties. • Labels • Name: FeatureLayer.Name or Alias. • NoFeatures • OverrideStyle: Equivalent is the FeatureLayer.Modifiers collection. • Pack: Equivalent is FeatureLayer.Table.Pack • PredominantFeatureType:MISpatialColumnInfo.PredominantObjectType. You can get the column from the Table.TableInfo.ColumnInfos property. • Refresh: Flushes the cache. Equivalent: Table.Refresh() • Search: Equivalent is MICommand. the result, however, can contain more than just feature keys. • SearchAtPoint: No “easy” equivalent. • SearchWithinDistance: No “easy” equivalent. • SearchWithinRectangle: No “easy” equivalent. • SearchWithinFeature: No “easy” equivalent. • Selectable: FeatureLayer.Selectable. • Selection: Selection class in Session.Sessions collection. • ShowCentroids: FeatureLayer.ShowCentroids • ShowLineDirection: FeatureLayer.ShowLineDirection • ShowNodes: FeatureLayer.ShowNodes • Style: Equivalent is now the FeatureLayer.Modifiers collection.

MapX 5.0	MapXtreme
Layers (continued)	• SupportsPack: Table.SupportsPack • Type: IMapLayer.Type • UpdateFeature: Equivalent through MICommand • Visible: IMapLayer.Enabled. IMapLayer.Visible takes into account: • ZoomLayer: IMapLayer.ZoomRangeEnabled • ZoomMax: IMapLayer.ZoomRange.End • ZoomMin: IMapLayer.ZoomRange.Start
Layers	Layers • AnimationLayer equivalent: GroupLayer and BackingStoreState.Off • InsertionLayer equivalent: AddMapTool.InsertionLayer • Selections: Live in the Session object.
LayerInfo	Obsolete. Equivalent is TableInfo
Legend	Legend
LegendText	LegendRow
LegendTexts	LegendRows

MapX 5.0	MapXtreme
Map	Map has now been separated into two entities: Map and MapControl. MapControl contains functionality applicable to the System.Windows.Forms.Control; an object that can be embedded in a WinForm. Map contains the actual mapping functionality, regardless of how it is being viewed (whether it is a WinForm or an export.) The following is a list of properties/methods in the former Map class and how they are being handled in the MapXtreme model: • Annotations: Obsolete. CenterX/Y: Map.Center ConvertCoord: DisplayTransform.ToDisplay(), DisplayTransform.FromDisplay() • CreateCustomTool: Inherit from MapTool or any of the Custom* tools available that provide marquee-line drawing capabilities using the mouse. • CurrentTool: MapTools.LeftButtonTool, MapTools.MiddleButtonTool, MapTools.RightButtonTool. You can access MapTools through the MapControl.MapTools property. • Datasets: Obsolete. Tables are accessible through the MICatalog. • Dataset: Obsolete. • DatasetGeoField: Obsolete. See Map.Dataset. • DatasetTheme: Obsolete. See Map.Dataset. DefaultStyle • DisplayCoordSys: Map.GetDisplayCoordSys(). • Distance: CoordSys.Distance(). • DynamicSelectionSupport: MapTools.DynamicSelectionModeSelectMapTool.DynamicSelectionEnabled

MapX 5.0	MapXtreme
Map (continued)	• EditableLabels • ExportMap: MapExport class that allows exporting a map with a wide-range of options. • ExportSelection: MapExport.ExportSelection • FeatureEditMode enums: • miEditModeFeature -> MapTools.NodeMode = false • miEditModeNode -> MapTools.NodeMode = true • miMoveDuplicateNodes -> MapTools.MoveDuplicateNodes • miDeleteDuplicateNodes -> MapTools.DeleteDuplicateNodes • miEditModeAddNode -> MapTools.AddNodeMode • FeatureFactory: FeatureProcessor • GeoDictionary: Geodictionary • Geoset: Obsolete. • Geosets: Obsolete. • GeosetWidth • hWnd • InfotipPopupDelay • InfotipSupport: Not addressed. • IsPointVisible • Layers: Layers • MapPaperHeight/Width: PaperSize (through MapExport.ExportSize) • MapScreenHeight/Width: MapControl.Size, Map.Size • MapUnit: CoordSys.Units of the display coordsys

MapX 5.0	MapXtreme
Map (continued)	• MapUnit: CoordSys.Units of the display coordsys • MatchNumericFields: GeoDictionary functionality. • MatchThreshold: GeoDictionary functionality. • MaxSearchTime: GeoDictionary functionality. • MilitaryGridReferenceToPoint: CoordSys.MilitaryGridToPoint() • MilitaryGridReferenceFromPoint: CoordSys.PointToMilitaryGrid() • Mouseicon: MapControl.Cursor • MousePointer: MapControl.Cursor • MouseWheelSupport: MapControl.MouseWheelSupport.MouseWheelBehavior • NumericCoordSys: Obsolete. Geometries have their own CoordSys now. Everything that has a coordinate has a coordsys. • Pan • PanAnimationLayer • PreferCompactLegends • PrintMap: Map.Draw() • PropertyPage • RedrawInterval: Map.IncrementalDraw.Interval. • Refresh • ReuseEquivalentOnRestore • Rotation: Map.Rotation. • SaveMapAsGeoset • SearchPath. • SelectionStyle: Selection.Style • SetSizeSnapToNodeSupport

MapX 5.0	MapXtreme
Map (continued)	• SnapToNodeSupport • SnapToNodeSupport • SnapTolerance: MapTools.SnapTolerance • Title: Adornments • TitleText • Version • WaitCursorEnabled • Zoom • ZoomTo
MultivarCategory	MultiVariableThemeCategory
MultivarCategories	MultiVariableThemeCategories
NotesQueryInfo	Obsolete.
NotesViewInfo	Obsolete.
OCIQueryInfo	Obsolete.
ODBCQueryInfo	Obsolete.
Parts	Equivalent
Point	DPoint
Points	Obsolete
RangeCategory	RangedThemeBin
RangeCategories	ModifierThemeBins
Rectangle	DRect
ResolveObject	MatchResolver
ResolveObjects	MatchResolver Collection.
RowValue	MICommand
RowValues	MICommand

MapX 5.0	MapXtreme
Selection	Selection
SourceRow	SourceRow
SourceRows	SourceRows collection
State	
Style	
Theme	ObjectTheme, FeatureStyleModifier
Themes	Obsolete. Themes are now contained by either an ObjectThemeLayer (for ObjectThemes) or in the Layer's Modifiers collection (for FeatureStyleModifier themes.)
ThemeProperties	Obsolete. Properties are within the theme classes themselves.
Title	Adornments
Variable	MIParameter. For binding in expressions, the MICommand.Parameters property allows you to define variables used within the command.
Variable	MIParameterCollection.

K – Localization Kit

The Localization Kit is a Visual Studio solution for translating software text elements (error messages, dialog/control text) into a language other than English, Japanese, and Simplified Chinese.

In this appendix:

Localization Kit

MapXtreme provides a Visual Studio solution for developers who wish to translate error messages and dialog/control text elements for use in their own MapXtreme-based applications.

This “localization kit” contains resource projects for all runtime components of MapXtreme. It is organized as a Visual Studio solution to make it convenient to edit the resource strings and build the assemblies while much of the behind-the-scenes resource management is taken care of. The solution can be used in Visual Studio, Visual C# Express Edition, and by the MSBuild command line build utility.

Included in each project are the English resource strings for translating and a strong named key (.snk) file. that will compile into an assembly that can be incorporated into your MapXtreme application. By following the information in the table below, you can easily identify which project(s) you need to translate for your application. You can also drill down inside each project to find only the resource files or individual strings you need to translate.

The LocalizationKit.sln includes the following projects:

Project	Description
GeoDictionaryManager.resources	Provides the error strings and dialog text elements for the GeoDictionaryManager.exe
MapInfo.CoreEngine.resources*	Provides the strings related to the core components of MapXtreme (mapping, data access, features, styles, themes, spatial processing, etc.)
MapInfo.LinearReferencing.resources	Provides the strings related to Z and M value support and linear referencing operations
MapInfo.Ogc.resources	Provides the strings related to converting FeatureGeometries to/from OGC well-known text and binaries.
MapInfo.Services.resources	Provides the strings for geocoding and routing services
MapInfo.Web.resources	Provides the strings related to web applications

Project	Description
MapInfo.WebControls.resources	Provides the strings related to web controls and tools
MapInfo.Wfs.Server.resources	Provides the strings related to WFS server error messages
MapInfo.Windows.Dialogs.resources	Provides the strings related to Windows dialog controls
MapInfo.Windows.resources	Provides the strings related to Windows controls
MapInfo.Wms.Client.resources	Provides the strings related to WMS error messages
WorkspaceManager.resources	Provides the error strings and dialog text elements for the WorkspaceManager.exe

* See the following table for resource information related to this namespace.

Resource File	Product Area
EllisAllTypeResources.en-US.resx	Error strings for reading from datasources
EllisCommandProcessorResources.en-US.resx	Error strings for reading from datasources
EllisCoordSysExceptions.en-US.resx	Error strings for bad coordinate system values
EllisCoordSysResources.en-US.resx	Error strings for bad coordinate system values
EllisDAEngineResources.en-US.resx	Error strings for datasource query Errors
EllisDBInfoResources.en-US.resx	Error strings for database Errors
EllisDBLayerResources.en-US.resx	Error strings for map Errors
EllisExceptions.en-US.resx	Error strings for general system Errors
EllisExprPacketCreatorResources.en-US.resx	Error strings for SQL queries

Resource File	Product Area
EllisExprPacketResources.en-US.resx	Error strings for SQL queries
EllisFcnInfoServerResources.en-US.resx	Error strings for SQL queries
EllisFindResources.en-US.resx	Strings for Find operations
EllisGeoObjectProcessResources.en-US.resx	Error strings for GeoObject manipulation (Lines, polygons, etc.)
EllisGeoObjectResources.en-US.resx	Error strings for GeoObject manipulation (Lines, polygons, etc.)
EllisGeoResources.en-US.resx	Error strings for GeoDictionary and file loading
EllisGeosetResources.en-US.resx	Error strings for Geoset files
EllisGmlXlatResources.en-US.resx	Error strings for GML files
EllisLegendResources.en-US.resx	Strings for use in Legends
EllisMapBasicInternalFcnResources.en-US.resx	Error strings for MapBasic functions
EllisMapBasicTranslatorResources.en-US.resx	Error strings for MapBasic functions
EllisMapperResources.en-US.resx	Error strings for Map manipulation/search functions
EllisMILexerResources.en-US.resx	Error strings for string parsing (SQL, MapBasic, etc.) functions
EllisMILicensingResources.en-US.resx	Error strings for copy protection/licensing components
EllisMIRDBResources.en-US.resx	Error strings for database connections
EllisMIRDBSpatialResources.en-US.resx	Error strings for database connections
EllisMIWindowResources.en-US.resx	Strings for use in GUIs
EllisProgramResources.en-US.resx	Error strings for file operations

Resource File	Product Area
EllisRasterResources.en-US.resx	Error strings for WMS connections and raster/grid images
EllisTextFileReaderResources.en-US.resx	Error strings for file operations
EllisThematicsResources.en-US.resx	Error strings for theme operations
EllisUtilityResources.en-US.resx	Error strings for file operations, and generic Error strings
EllisXMLUtilResources.en-US.resx	Error strings for XML/GML operations
strings.en-US.resxGeneric	Error strings, units, coordinate system, datum, and ellipsoid names, assorted GUI strings, WFS strings, serialization and raster Errors.
MapInfo.TileServer	Error strings for Tile Server operations, and generic Error strings.

System Requirement

Visual Studio 2015 or Visual Studio 2017.

How to Use the Localization Kit

The Localization Kit is provided as a Visual Studio solution containing resource strings that you can translate into any language. The Visual Studio solution can be used in Visual Studio 2015 with Update 4 or Visual Studio 2017, and by the MSBuild command line build utility. This can be done simply by opening the appropriate .resx file in Visual Studio's Resource Editor and editing the name/value pairs. You can translate as little as one line in a single .resx file or go all the way translating every line in every .resx file for all the assemblies. It just depends on your application requirements.

The files to be translated are named with the language/locale identifier that follows Microsoft's conventions. For example, the English-U.S. exception strings for the LinearReferencing.resources project are in a file called

LinearReferencingExceptionStrings.en-US.resx. For more information on language identifiers, see the System.Globalization.CultureInfo class in the Microsoft .NET Framework library.

Most of the .resx files are contained in Resources folders under each project. The Windows and Windows.Dialog resources are located in Controls or Dialogs folders.

Building the Satellite Assemblies

Satellite assemblies are optional, standalone assemblies that contain just the compiled resources. Once you have your .resx strings translated, you are ready to build your satellite assemblies. Right-click on the project in Visual Studio and choose Build. Each project will generate a bin folder with a language/locale-specific folder containing the assembly. The file naming convention for the assembly is patterned after the English assembly name with the extension of .resources.dll. For example, the WebControls assembly would be called MapInfo.WebControls.resources.dll.

At runtime, MapXtreme will use the appropriate language/locale resource dll when one is provided and the .NET Resource Manager is set to look for that satellite assembly. If no localized assembly is found, the English assembly is used instead.

Signing Assemblies

The satellite assemblies you build with the Localization Kit are signed with one of the following strong named public keys (included in the kit):

- MapInfo.CoreEngine.Public.snk
- MapInfo.Ogc.Public.snk
- MapInfo.WebControls.Public.snk.

MapInfo.CoreEngine.Public.Snk is the public key for all assemblies except for MapInfo.Ogc.resources and MapInfo.WebControls.resources, which have their own public keys. Signing your satellite assemblies with these public keys will allow you to use the assemblies for testing and debugging. In order to run MapXtreme using these assemblies, they must be put in the global assembly cache (GAC). Only signed assemblies can be placed in the GAC.

Registering the Satellite Assemblies

The output satellite assemblies from the Visual Studio solution are not quite ready to be installed into the GAC and tested. After the satellite assemblies are built and signed with the public keys, they must be registered so that the .NET runtime will allow them to be loaded into the GAC. The Strong Name Utility (sn.exe) must be run on each of the

satellite assemblies to register the assemblies for verification skipping. This command is `sn.exe -Vr <assembly_name>`. Once the satellite assembly is registered to skip verification, the satellite assembly is ready to be tested.

Building from the Command Line

As a part of the MapXtreme Localization Kit, two MSBuild project files have been provided to simplify the process of preparing the satellite assemblies for testing. The MSBuild project files will build, sign, register the assemblies to skip verification, and install them in the Global Assembly Cache. For more information on MSBuild and its usage, visit MSDN MSBuild Reference page. (http://msdn.microsoft.com/en-us/library/0k6kkbsd.aspx.)

The two MSBuild project files are *LocalizationKit.proj* and *LocalizationKit.Utilities.proj*.

LocalizationKit.proj

The LocalizationKit.proj targets are:

build - builds the LocalizationKit solution and calls the LocalizationKit.Utilities project's "RegisterSkipVerifiection" target (default target)

rebuild - rebuilds the LocalizationKit solution and calls the LocalizationKit.Utilities project's "RegisterSkipVerification" target

clean - deletes all the output satellite assemblies and any intermediate files as a part of the build process

LocalizationKit.Utilities.proj

The LocalizationKit.Utilities.proj targets are:

RegisterSkipVerification - Registers all the satellite assemblies to skip verification by running the `sn.exe -Vr <assembly_name>` command (default target)

UnregisterSkipVerification - Unregisters all the satellite assemblies to skip verification by running the `sn.exe -Vu <assembly_name>` command

InstallGac - Installs the satellite assemblies into the Global Assembly Cache by running the `gacutil.exe /I <assembly_name> /f` command

UninstallGac -Uninstalls the satellite assemblies from the Global Assembly Cache by running the `gacutil.exe /uf <assembly_name>` command

Acceptable values for the project's configuration property are debug (default) and release. See below for usage.

The LocalizationKit project file depends upon and uses the targets in the LocalizationKit.Utilities project file.

ⓘ For developers using Visual C# Express Edition, you will have to add MSBuild and the directory containing the Strong Name Utility and Global Assembly Cache Utility to your system path.

Given these path locations,

MSBuild (msbuild.exe) - C:\WINDOWS\Microsoft.NET\Framework\v3.5

Strong Name Utility (sn.exe) and Global Assembly Cache Utility (gacutil.exe) - C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin

,the line in the command prompt would be,

```
LocalizationKit> set
path=%path%;C:\WINDOWS\Microsoft.NET\Framework\v3.5;C:\Program Files\Microsoft
SDKs\Windows\v6.0A\bin
```

The directory paths on your machine may be very different, so be sure to use the correct paths in your environment.

To build the Localization Kit, run this command from the kit's root directory:

```
LocalizationKit> msbuild LocalizationKit.proj /target:build
/property:configuration=release
```

This command will build the release versions (no debug information) of the satellite assemblies, signing each with the appropriate public strong name key file, and registering the satellite assemblies for skipping verification, then installing the satellite assemblies into the Global Assembly Cache.

The satellite assemblies are ready to be tested.

Private Key Signing for Satellite Assemblies

When you are ready to include your localized assemblies in your application, they must be signed with a private key by Precisely. Contact your Technical Support representative for information on this process.

L – Log Files in MapXtreme

This appendix contains information about the application Logging feature in MapXtreme. Logging helps you to trace the valuable information during the execution of application flow.

In this appendix:

Logging in MapXtreme

We have provided Logging support in MapXtreme to help you trace the valuable information during the flow of execution. Currently, MapXtreme supports logging for ODBC and OCI levels only.

The application logs provide useful information for tracing and debugging. The Log file path field determines the location of the trace file. MapXtreme writes the log information in a text file, however, users have the flexibility to configure the logging.

Logging Configuration Options

The configuration options allow users to describe how and where logging information should be written.

- You can enable/disable logging by setting the `ApplicationLogEnabled` property as `true/false.` For example, `Session.Current.ApplicationLogEnabled = true`.
- You can change the name and location of the log file by using the `Session.Current.ApplicationLogPath = "C:\Test\MapXtreme\mylog.log` property.

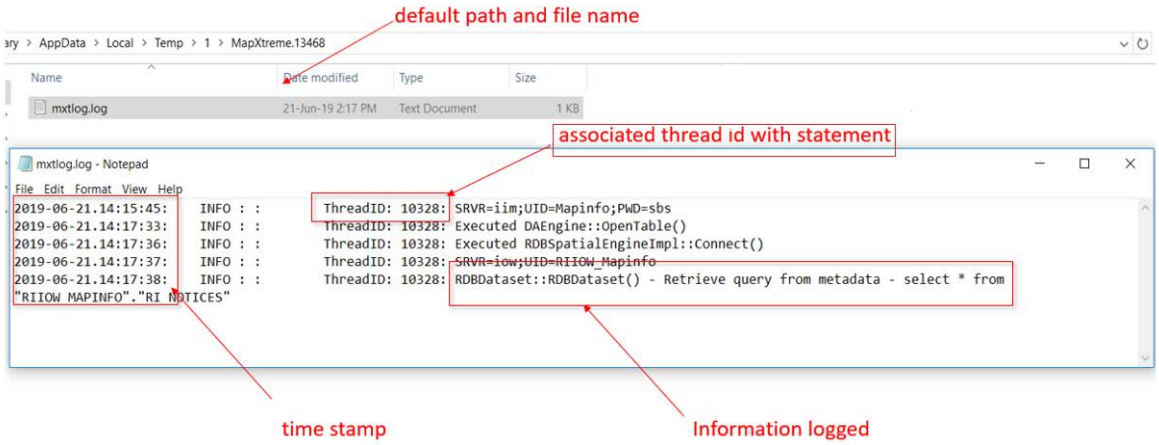
By default, a file path is provided as:

- File name: mxtlog.log
- File path *C:\Windows\Temp or ..\AppData\Local\Temp*

The log file reports the following information.

1. Time Stamp: Time of statement execution
2. Log level e.g., INFO, WORRNING and ERROR etc.
3. Thread ID
4. Logging information

Here is an example of what a MapXtreme log file looks like:



Log File Directory and Structure

Max Log Size

The maximum limit of a log file (maxlog.log) is 5MB. MapXtream supports a maximum of 50 log files in each session on a machine. After reaching the maximum file count, MapXtream automatically overwrites the first file in the sequence.

The parameter maxlog.log specifies the size of each log file in Megabytes.

Glossary

This glossary defines terms used in this guide and in MapInfo products that are necessarily understandable or are used in a way specific to MapInfo products and technology.

Terms

Adornment
A MapXtreme map element that consists of either a legend, title, or scalebar.

Affine Transformation
A linear transformation, such as a rotation, scaling or shearing, of a geometric object along with a shift from that transformation. Used in GIS for transforming maps from one coordinate system to another.

Anti-aliasing
Smooths the jagged edges of lines, curves, and the edges of filled areas when representing a high-definition rendition at a lower resolution.

Cartesian
A coordinate system using an x,y scale not tied to any real-world system. Most CAD drawing uses this method of registering objects (for example, a drawing of a ball-bearing assembly or a floor plan). If a drawing uses Cartesian coordinates, one corner of the drawing probably has coordinates 0, 0.

Cartesian Coordinates
The conventional representation of geometric objects by x and y values on a plane.

Centroid
Usually the center of a map object. For most map objects, the centroid is located at the middle of the object (the location halfway between the northern and southern extents and halfway between the eastern and western extents of the object). In some cases, the centroid is not at the middle point because there is a restriction that the centroid must be located on the object itself. Thus, in the case of a crescent-shaped region object, the middle point of the object may actually lie outside the limits of the region; however, the centroid is always within the limits of the region.

Character Encoding
A method of converting a sequence of bytes into a sequence of characters. See also Universal Character Set (UCS) and Unicode Transformation Format-8 (UTF-8).

Class
In an object-oriented language, a class is an object or a set of objects that contain(s) methods for performing some type of function, similar in meaning to a derived type in procedural languages.

Codespace
See MapInfo Codespace.

Convex Hull Buffer
A type of buffer that creates a region object that represents a polygon based on the nodes from the input object. You can think of the convex hull polygon as an operator that places a rubber band around all of the points. It consists of the minimum number of points so that all points lie on or inside the polygon. With convex hull buffers, no interior angle can be greater than 180 degrees.

COM+ Pooling
A Microsoft component service in which objects are pre-loaded and pooled to save resources.

Coordinate
An x,y location in a Cartesian coordinate system, or a Latitude, Longitude location in an earth coordinate system. Coordinates represent locations on a map relative to other locations. Earth coordinate systems may use the equator and the Greenwich prime meridian as fixed reference points. Plane coordinate systems describe a two-dimensional x,y location in terms of distance from a fixed reference and are usually in the first quadrant so that all coordinates are positive numbers.

Coordinate System
A coordinate system is used to create a numerical representation of geometric objects. Each point in a geometric object is represented by a pair of numbers. Those numbers are the coordinates for that point. In cartography, coordinate systems are closely related to projections. You create a coordinate system by supplying specific values for the parameters of a projection.

Data Provider
A bridge between an application and a data source, which provides mechanisms for accessing data for use in the application.

Data Binding
The association of a data source with a server control. The MapXtreme DataBinding class contains information about a single data-binding expression in an ASP.NET server control, which allows rapid-application development (RAD) designers in Visual Studio to create data-binding expressions at design time.

Decimal Degree
The decimal representation of fractions of degrees. Many paper maps express coordinates in degrees, minutes, seconds (for example, 40_30i10I), where minutes and seconds are fractions of degrees. Thirty minutes equal half a degree, and 30 seconds equal half a minute. MapXtreme expresses coordinates in decimal degrees (e.g., 72.558 degrees). Thus, the longitude: 40 degrees, 30 minutes, would be expressed in MapXtreme as 40.5 degrees.

Degrees Longitude, Degrees Latitude, Decimal Degrees
Degrees longitude and degrees latitude are coordinates used to represent locations on the surface of the earth. Longitude, or X-coordinate, represents a location's east-west position, where any location west of the prime meridian has a negative X value. Latitude, or Y-coordinate, represents a location's north-south position, where any location south of the equator has a negative Y value.

Derived Class
A class that contains all of the features of its base class but contains either additional functionality or enhanced functionality with respect to its base.

Enumerate
A type that contains all of the variables and their possible values.

Event Handler
An attribute for an object on a page that can be written in either JavaScript or VBScript. For example, an event handler describes what to do when a user clicks a button or selects text in a list box. Both VBScript and JavaScript support explicitly defined event handlers, for example, on Click and on Select. In addition, you can define functions that replace the explicit event handlers. Such functions are called implicit event handlers.

Feature
A row in a table that has geometry, style, and attributes. A Feature usually has a Table and Key to identify which row it represents.

FeatureLayer
A MapXtreme layer that displays Features from a Table. For example, a layer of region objects representing world countries is a FeatureLayer. A FeatureLayer must be added to a Map via the Map's Layer collection. FeatureLayers can be native .TAB data, remote RDB, seamless or raster data.

Geocode
The process of assigning X and Y coordinates to records in a table or database so that the records can be displayed as objects on a map.

GeoDictionary
A MapXtreme file that contains information about tables (TAB files only). The GeoDictionary is used to automatically determine the table to which application data should be bound.

GeoDictionaryManager
A MapXtreme tool for maintaining the Geodictionary.

Geographic Information System (GIS)
An organized collection of computer hardware and software designed to efficiently create, manipulate, analyze, and display all types of geographically or spatially referenced data.

Geometric Centroid
A centroid point (see Centroid) that does not need to be contained within the object (usually a FeatureGeometry).

Geography Markup Language (GML)
A markup language specific to mapping. The GML is being developed by the Open GIS Consortium (OGC), an international organization that develops and promotes geographic standards.

Graticule
A grid of horizontal (latitude) and vertical (longitude) lines displayed on an earth map, spaced at a regular distance (for example, every five degrees, every fifteen degrees). Used to establish a frame of reference.

Grid
An interpolation of data values across an area. A grid is created from a data file in which data is measured at evenly-spaced points. The entire map area is converted to a grid in which each grid cell represents a value. See Chapter 17 Grid Images.

Hillshading
Relief shading of a grid map according to a virtual light source. The brightness of each grid cell corresponds to the light striking the surface and is adjusted based on its orientation to the light source.

Hypertext Markup Language (HTML)
A plain text (Ascii) language that enables developers to create Web pages that can be displayed by different Web browsers on different computing platforms. Html uses tags to specify the structure of the various parts of a document. Html supports links (using URLs) that point to other web documents and files.

Hypertext Transfer Protocol (HTTP)
The message-based network interface between a Web client and a Web server. HTTP runs on top of TCP/IP.

Inflection
The point in a grid map at which the color changes due to a change in the grid value or percentage. See Chapter 17 Grid Images and Inflections.

Internet Information Services (IIS)
The software services provided by Microsoft that support the creation, configuration, and management of web sites. Specifically some commen Internet Information Services include: FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol). In MapXtreme, the WMS Server we provide must be configured to work with IIS to run the Server.

ISession Interface
The MapXtreme MapInfo.Engine namespace interface that provides the starting point for all MapXtreme applications. ISession manages the initialization of resources needed for a MapXtreme application, and defines all data and functionality applicable to an instance of an application.

Latitude
The horizontal lines on a map that increase from 0 degrees at the Equator to 90 degrees at both the North (+90.0 degrees) and South (-90.0 degrees) poles. Used to describe the north-south position of a point as measured usually in degrees or decimal degrees above or below the equator.

Layer
A basic component of map display in MapInfo products, typically consisting of several superimposed layers (e.g., a layer of street data superimposed over a layer of county or postal code boundaries). When a table appears in a Map window, it occupies a layer in that Map window. Typically, each map layer corresponds to one open table.

Linear Referencing
An alternative reference system to the traditional coordinate reference systems that tie locations of linear features to points on the earth. Any physical asset that you can map as part of a linear network can hold data that describes the asset or a condition or event related to that asset. In MapXtreme the data is stored as an M, or measure value, on the MultiCurve object along with the X and Y coordinates for the location. The M values can then be further mapped and analyzed for better resource management. See Chapter 22 Linear Referencing.

Longitude
The vertical lines on a map, running from the North to South poles, used to describe the east-west position of a point. The position is reported as the number of degrees east (to -180.0 degrees) or west (to +180.0 degrees) of the prime meridian (0 degrees). Lines of longitude are farthest apart at the Equator and intersect at both poles, and therefore, are not parallel.

Longitude/Latitude
The default coordinate system for representing geographic objects in a map in MapInfo products.

MapControl
A MapXtreme object that enables you to view a map on a form. The MapControl owns the window that the map draws to. It also controls the size of the map and interacts with the map tools. MapXtreme provides desktop and web versions of MapControl.

MapInfo Codespace
A list of definitions and standards that are commonly used in creating MapInfo maps and workspaces. The MapInfo Codespace includes coordinate system settings; pen, brush, and distance settings and abbreviations; image size settings; frequently used types and their abbreviations; a list of available operators; time, date, and temperature unit settings, and abbreviations. For details, see Appendix G: Defining the MapInfo Codespace.

MapInfo MapCatalog

A server table containing column information about spatial tables. See The MapInfo_MapCatalog.

MapInfo SQL Language

A reference of SQL syntax used in MapInfo mapping products. The language is based on SQL3 and has special MapInfo operators defined for spatial analysis.

Meridian

A line or a portion of a line running from the North to the South pole. A longitudinal line.

Namespace

A hierarchal naming system that provides a way to group classes together independently of inheritance. For example, two unrelated classes with the same name can exist in different namespaces: System.Utilities.FileFinder and MyCompany.Utilties.FileFinder could have the same name, but different functionality. Namespaces also help to prevent the compiler from referencing the wrong class (a 'collision').

Non-Earth Map

A map whose objects are not explicitly referenced to locations on the earth's surface. Floor plans are typical examples.

Persistence

Persistence refers to the way MapXtreme (and other MapInfo products) manages data and ensures that maps created using this API can be used by other MapXtreme users. Persistence is concerned with loading and saving XML-based workspaces and for parsing and publishing MapInfo Geometry objects from/to GML, and vice versa.

PointRef Schema

A spatial schema that can be applied to a non-mappable table to make it mappable. The schema references a Geometry object in another table by matching the values in a column (MatchColumn) of the non-mappable table with a column (RefColumn) in a mappable table. When the table is opened, it contains a read-only Geometry column. The table can then be added to a Map as a layer.

Pooling

To shared resources for better performance and scalability. In a MapXtreme web application, MapXtreme Session instances are available for use in a COM+ pool and ready to service requests from clients.

Projection

A mathematical model that transforms the locations of features on the earth's surface to locations on a two-dimensional surface, such as a paper map. Since a map is an attempt to represent a spherical object (the earth) on a flat surface, all projections have some degree of distortion. A map projection can preserve area, distance, shape or direction but only a globe can preserve all of these attributes. Some projections (for example, Mercator) produce maps well suited for navigation. Other projections (for example, equal-area projections, such as Lambert) produce maps well suited for visual analysis.

Region

A region is a MultiPolygon with one exterior Ring and zero or more interior Rings (holes).

Serialization

Serialization is the process of converting an object into a stream of data in order to preserve it on the server. This process is an essential part of maintaining objects in MapXtreme web applications. If the objects are not maintained the server would need to recreate the object (such as a map) for each web request.

Spatial Schema
A service that can be applied to a table to enhance its spatial capabilities. There are two types of spatial schemas in MapXtreme: PointRef and XY. Non-mappable tables that contain either a column that can be referenced to a column in a mappable table, or columns that represent XY values, can use these schemas to create a Geometry column. These tables can then be added to a Map as a layer. See the PointRef Schema and XY Schema glossary definitions for information about each type of schema.

State Management
A general term in web application development that deals with saving and restoring information from a browser session.

Table
A collection of data organized in row and column format. In MapXtreme, tables contains the data you wish to display on the map. Tables hold information that describe the features, including their geometry, style, and attributes. MapXtreme supports tables from a wide variety of sources including, native tables (MapInfo .TAB), relational database management systems (RDBMS), dBase, MS Access, ASCII files, and ESRI ShapeFiles. Speciality tables include raster, grid, seamless, views, WMS, and ADO.NET. The type of table is available through the TableInfo class. Tables are opened and closed via the Catalog in the Data namespace. See Chapter 8 Working with Data.

Tile Handler
An HTTP handler that processes requests for map tiles from web applications. See MapXtreme Tile Handler.

Uniform Resource Locator (URL)
The underlying implementation of a hypertext link or image map that contains the address of a Web page or file somewhere on the World Wide Web. A URL contains information about the network protocol to use (usually HTTP) and the path to the page or file. An URL example is “http://www.mycompany.com/index.html,” which points to the index page for the “my company” web site.

Universal Character Set (UCS)
The international standard ISO 10646 defines the Universal Character Set (UCS). UCS is a superset of all other character set standards. UCS also defines several methods for encoding a string of characters as a sequence of bytes, such as UTF-8 and UTF-16.

Unicode Transformation Format-8 (UTF-8)
An octet (8-bit) lossless encoding of Unicode characters. MapXtreme supports UTF-8 only as indicated in the workspace persistence schema.

Web Controls
An element on a web page that users interact with to send requests to the web server.

Web Map Service (WMS)
WMS is an OGC-compliant Web service that provides map images for use as layers in a mapping application. MapXtreme provides a server implementation of WMS if you wish to host a WMS server and a client for accessing any OGC-compliant (1.0.0, 1.1.0, 1.1.1, or 1.3.0) WMS.

Web Feature Service (WFS)
An OGC-compliant Web service that offers geo-referenced map features for use in mapping applications. MapXtreme provides an implementation of WMS Basic, a read-only service and a client for accessing WMS servers.

Web server

A computer system that runs the Hypertext Transfer Protocol and Web Server software. A Web server accepts URL-based HTTP requests from a Web user's browser and sends HTML pages back to the browser. A Web server can manage one or many Web sites. A commercial server, for example, would typically have many Web sites.

Workspace

An XML-based persistence file format that allows users of MapXtreme to share maps they have created in a wide variety of environments. This is the file format that all future MapInfo products will conform to. For more about creating workspaces, see Chapter 27 Workspace Manager. For more about the structure of workspaces, see Appendix C: Understanding the MapInfo Workspace.

XY Schema

A spatial schema that can be applied to a non-mappable to make it mappable. The table must contain X and Y coordinate values, which the schema accesses to create a Geometry column for the table.

precisely

2 Blue Hill Plaza, #1563
Pearl River, NY 10965
USA

www.precisely.com