Spectrum™ Technology Platform

Version 12.0

Spectrum Spatial Getting Started Guide

Table of Contents

1 - What Is Location Intelligence?

Location intelligence is:

- An awareness of relationships between location information and business analysis and operations
- The ability to use the understanding of geographic relationships to predict how it impacts a business or organization
- The capability to react to how location influences an organization by changing business processes in order to minimize risk and maximize opportunities

Location Intelligence enables a business to measure, compare and analyze its data from business operations, in conjunction with external data such as transportation networks, regulatory jurisdictions, market characteristics or its own customers.

In this section

2 - What is Spectrum Spatial?

Spectrum Spatial is an enterprise location intelligence platform designed to provide organizations with a suite of broadly applicable location capabilities, including spatial analysis, mapping, routing, geocoding, and geoprocessing. These capabilities can be combined with a wide range of data management capabilities in Spectrum™ Technology Platform and an extensive data catalog to solve a diverse variety of business problems.

In this section

Spectrum Spatial Concepts

This section presents concepts of which you will need an understanding as you develop your applications with Spectrum™ Technology Platform.

Named Resources

A resource can be a table, layer, map, tile, or style. In the Spectrum™ Technology Platform, a named resource is data to which a name has been given. This allows the resource to referred by its name and not its properties. Named resources must be stored in a central database called a repository.

For more information, see the Resource and Data section of the *Spectrum Spatial Guide*.

Repository

The repository is a central document database that stores Named Resource definitions.

For more information, see the Resource and Data section of the *Spectrum Spatial Guide*.

REST

Representational State Transfer (REST) is an architectural style that revolves around resources and the actions (analogous to verbs) that one can take with them. The "verbs" are provided by HTTP (for example, GET, PUT, or DELETE). In REST, resources can have one or more representation, typically HTML, JSON and image types. REST services are usually used from within a web application in a browser.

JavaScript API

The JavaScript API includes a set of browser-based user interface components for easily embedding maps and other location-based capabilities in web pages. The API and the components can be used in a wide range of scenarios, from simply embedding maps in your web site to show locations, to creating rich web applications. Built entirely in JavaScript, the controls work without the need for any browser plug-ins and without having to write any server-side code. The JavaScript API components use Web 2.0 techniques to provide functionality such as seamless map panning and the ability to search and display information without requiring a web page refresh.

SOAP

SOAP is short for Simple Object Access Protocol. A SOAP request and response are precisely defined by the server's description known as **WSDL** (Web Service Description Language) and is supported by many tools and programming languages that act as SOAP clients.

Stages

Stages are pieces of technology provided in a module that can be connected with logic, flow controls, readers and writers using the Enterprise Designer and then used as a batch process, web service or a subset of functionality for other flows.

Web Services

The capabilities of Spectrum Spatial are made available as web services using open standards such as SOAP or REST. To access the web services, you construct an HTTP request and submit it directly to the web service. The HTTP request can be in the form of a SOAP request, a REST request, or a POST/GET request, and can be submitted directly from a web browser or from a client application.

WSDL

WSDL is short for Web Service Description Language. It describes the capabilities offered by a web service. Clients connected to a web service can read the WSDL for available operations and send SOAP requests that call those operations.

3 - Modules and Stages

Modules provide specific processing capabilities to solve specific business problems. In Spectrum Spatial the modules perform spatial operations including mapping, geocoding, routing and geoprocessing.

Stages are pieces of technology provided in a module that can be connected with logic, flow controls, readers and writers using the Enterprise Designer and then used as a batch process, web service or a subset of functionality for other flows.

In this section

Location Intelligence Module

The Location Intelligence Module enables an organization to rapidly integrate location information into business applications and processes. This enables organizations to create and embed maps, understand spatial relationships, and carry out spatial calculations. The Location Intelligence Module provides support for standards-based geospatial information sharing through support for certified **OGC** web services.

The Location Intelligence Module provides support for integration at three levels:

- **Web application development** - Rapid application development user interface components and JavaScript API
- **Web service integration** - Support for SOAP, REST and certified **OGC** web services (WMS, WMTS, WFS)
- **Data integration** - Batch and transactional integration of spatial data sources

The Location Intelligence Module is part of the Spectrum Spatial solution encompassing:

- Location Intelligence Module
- Enterprise Geocoding Module
- Enterprise Routing Module
- GeoConfidence Module

The Location Intelligence Module offers the following capabilities:

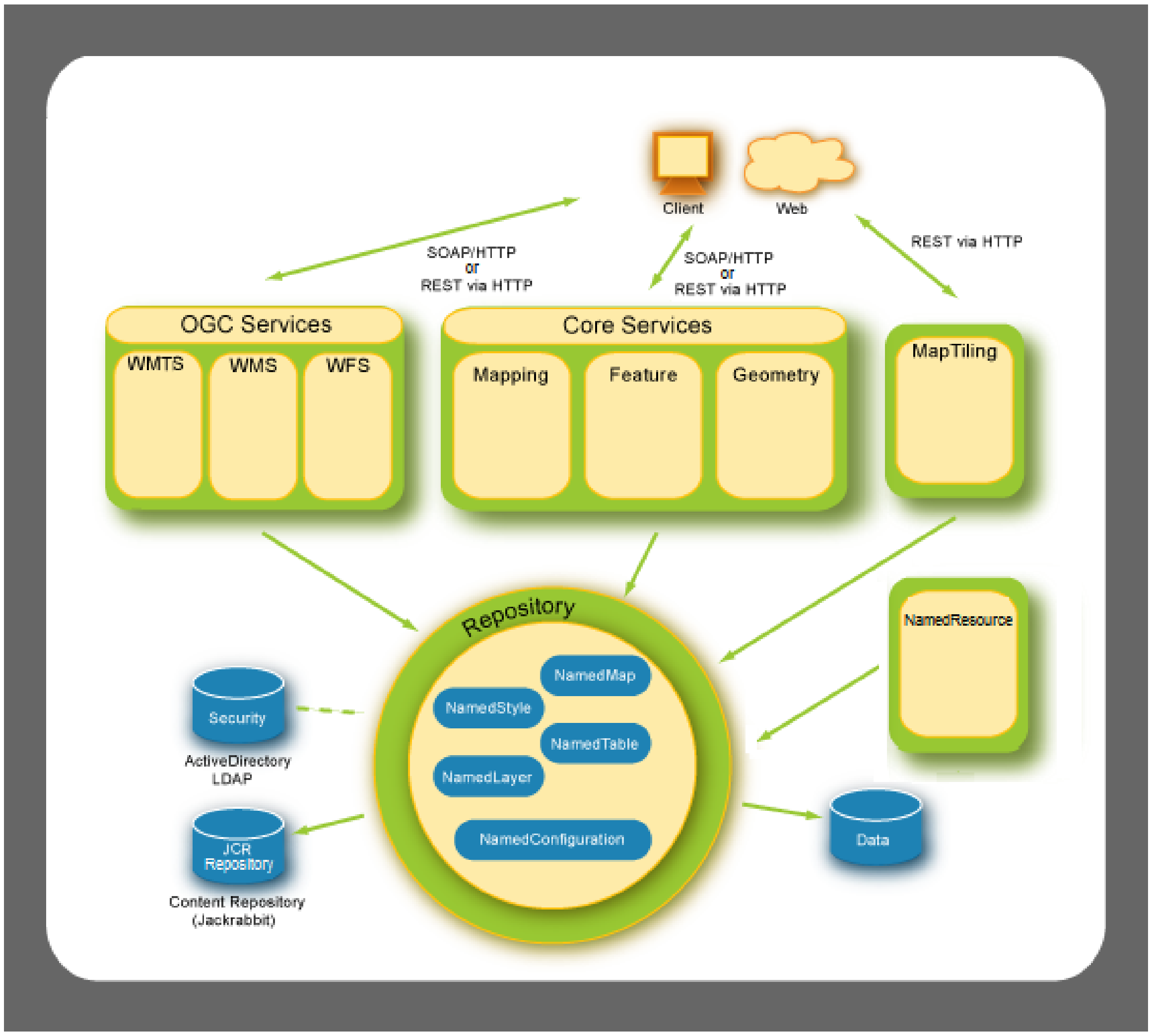
- Map-based data visualization and analysis
- Feature querying and searching
- Geometry operations including measurements (area, perimeter, length), predicates (within, contains, intersects), geometry manipulation (union, buffer, intersection) and coordinate transformation

Some of the things these capabilities enable you to do are:

- Determine which sites are nearest to a given location
- Determine whether or not a location resides in a given area
- Query a spatial database using spatial functions as well as standard SQL functions
- Perform a variety of calculations on spatial data, such as determining the area of a polygon, determining the distance between two points, or measuring the perimeter of an area
- Return a geometry object that represents the union of two input geometry objects

Architecture Overview

The following diagram shows an overview of the fundamental architecture of Spectrum™ Technology Platform Location Intelligence Module, prior to any development, extension, or scaling.



The architecture consists of the following components:

Repository

The foundation of the Spectrum™ Technology Platform Location Intelligence Module is the repository. It contains named resources such as named styles, named tables, named maps, named layers, and named configurations. See the Resources and Data section in the Spectrum™ Technology Platform *Spectrum Spatial Guide* on **support.pb.com**.

Core Services

The core services that Location Intelligence Module provides are Mapping, Feature, and Geometry services. See the Services section in *Spectrum Spatial Guide* on **support.pb.com** for more information.

OGC Services

The **OGC** (Open Geospatial Consortium) services include:

- Web Map Service (WMS)

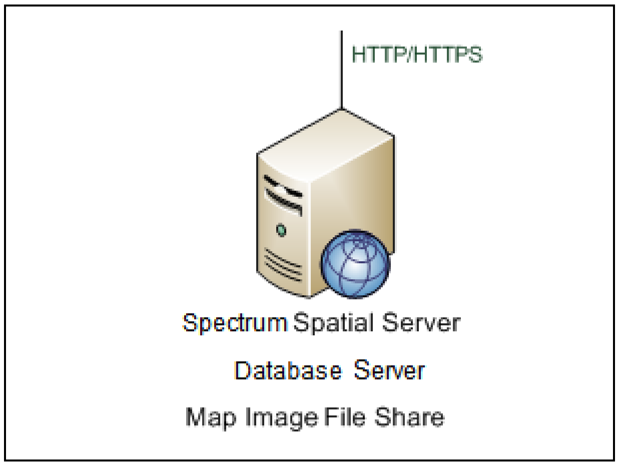
- Web Feature Serivce (WFS)
- Web Map Tiling Service (WMTS)

Map Tiling
The Map Tiling Service returns map tiles on the fly or from a tile cache at the user's request. See the Services section in the *Spectrum Spatial Guide* for more information.

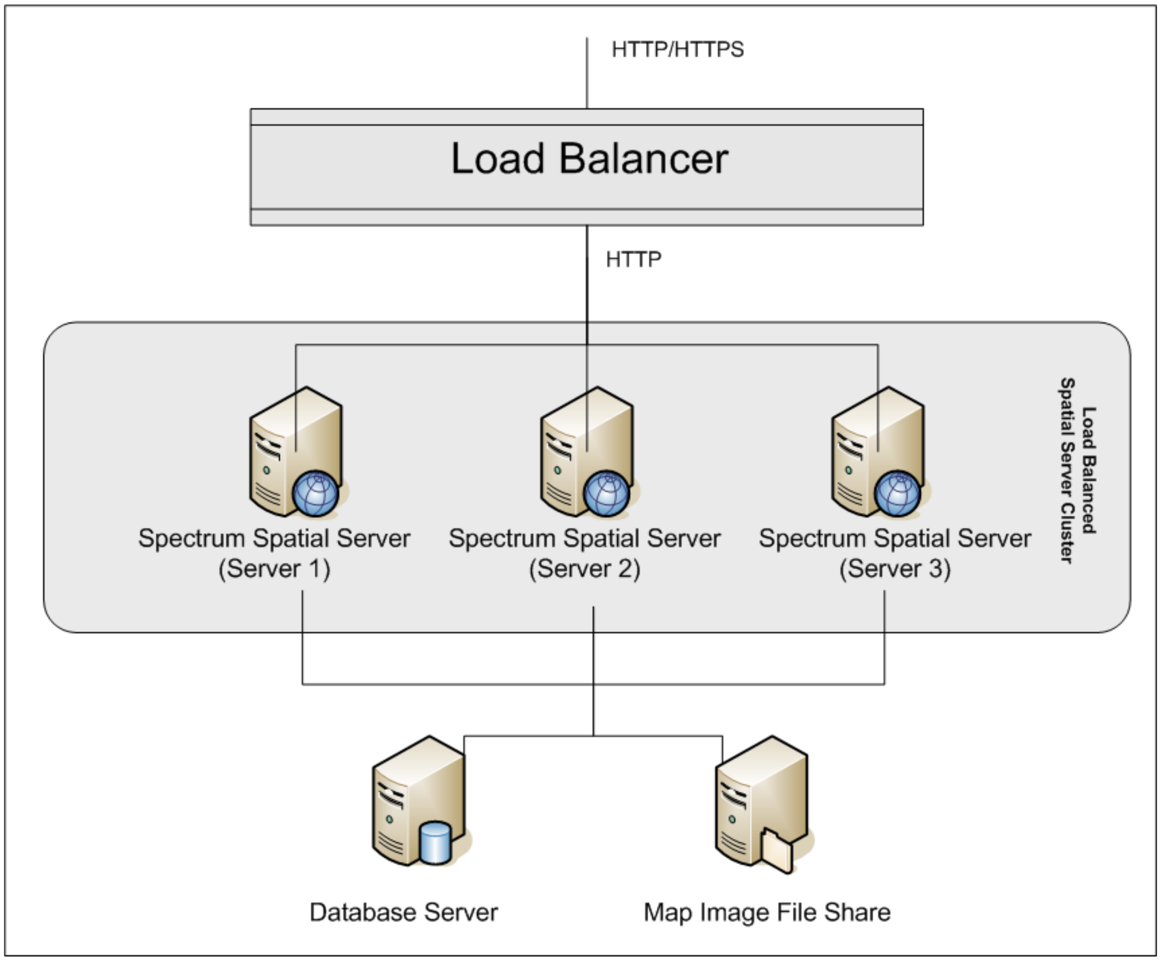
Named Resource
You can use the Named Resource Service to manage almost any type of named resource in the repository. See the Services section in the *Spectrum™ Technology Platform Spectrum Spatial Guide* on **support.pb.com**.

Architecture: Basic and Scaled Diagrams

The diagram below shows an example of the default, initial architecture of Spectrum™ Technology Platform Location Intelligence Module, where there is a single server on which Spectrum Spatial is installed. Everything, including the repository, image files, and tile cache on a single box in the same location as where the product is installed.



To scale Spectrum Spatial from one to three servers, each server leverages a single shared database server and image server to maximize efficiency and to avoid situations where an image or database resource is unavailable to one of the Spectrum Spatial servers. The file image server stores all images between multiple versions of Spectrum Spatial. All instances of Spectrum Spatial store images in a single location


HTTP/HTTPS
Load Balancer
HTTP
Spectrum Spatial Server (Server 1)
Spectrum Spatial Server (Server 2)
Spectrum Spatial Server (Server 3)
Load Balanced Spatial Server Cluster
Database Server
Map Image File Share

For load balancing information, see the Adminstration section in the Spectrum™ Technology Platform *Spectrum Spatial Guide* on **support.pb.com**.

Mapping Concepts

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

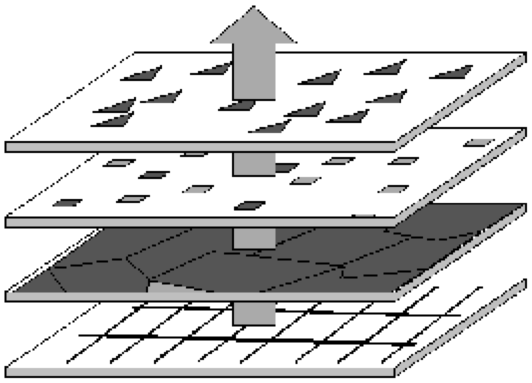
Maps

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.

Layers

Layers contain the rules of how spatial data is rendered on a map, whether that data be vector features, grid or raster images, labels or dynamic generated data such as pie or bar charts. **Thematic rules** are typically used to determine the map styles in a layer. This dynamically connects the map visualization with the runtime data associated with the map elements.

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. For example, a layer of boundary regions should be placed beneath a layer of points.



To learn about the types of layers that Spectrum™ Technology Platform supports, see the Resources and Data section in the Spectrum™ Technology Platform *Spectrum Spatial Guide* on **support.pb.com**.

Tables

Tables are collections of features from a data source. Tables hold rows and columns of information that describe the features, including their geometry, style, and attributes.

Features

A feature is a row in a table. Features are defined by their data source, ID, style, attributes, and in the case of spatial features, geometry. Features are displayed on a map as a Feature Layer.

Searching features for additional information is one of the main uses of digital maps. In Spectrum™ Technology Platform features are returned in FeatureCollections as the result of a query.

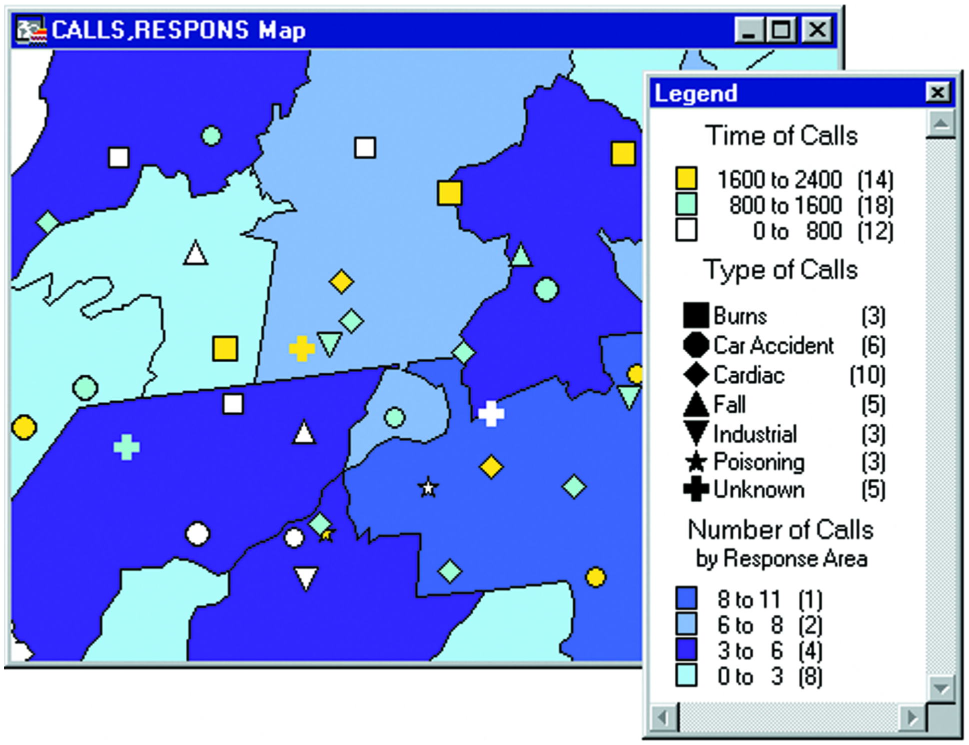
Labels and Legends

Maps need text such as labels and legends to deliver the message of the map.

Labels describe the data in the feature table. They can be customized in many ways, including their visibility, position, style, and content. Themes that change the style and content can be applied to label layers. Use MI SQL expressions to generate custom label strings and rules.

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.

Legends in Spectrum™ Technology Platform are generated on the fly when a map is requested, and based on the defined themes and styles of the map's layers.



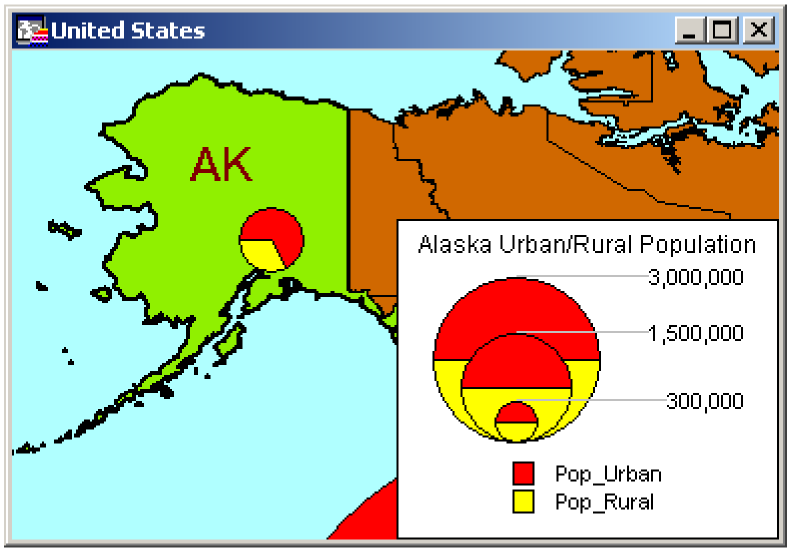
Themes

Themes are the mechanism by which you determine the map display. This can be as simple as a single style for all features (override) or more complex rules to visualize the information from the underlying data.

The main purpose of a theme is to graphically communicate a particular type of data that is associated with the layer, such as population levels or temperature ranges. The particular type of data forms the theme of the map, and the theme can represent the data using shades of color, fill patterns, symbols, bar charts, and pie charts.

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.



Styles

Styles enable you to control the visual appearance of map features, labels, themes and legends. Each feature has a default style. To change the style, apply an override theme.

In Spectrum™ Technology Platform the support for sparse styles allows you to change only the properties you want and merge styles together to form a single style.

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.

Map Tiles

Map tiles are portions of a map that are seamlessly joined on the fly while you are panning and zooming in the map window. Each pan and zoom is a call to the server to request the appropriate map tile(s). Only those that fit the bounds of the map view and match the zoom level are returned. Google Maps and Bing Maps are examples of applications that use map tiles.

Map tiles are created by dividing a map into levels that represents specific zoom levels. Level 0 is the entire map, Level 1 is one level zoomed out from the entire map and divided evenly into four tiles. Level 2 contains 16 tiles, which are the four tiles from Level 1 divided evenly. Level 3 are the 16 Level 2 tiles subdivided into 64 tiles. Levels 0-3 of a world map typically show global levels detail, while higher levels, are zoomed in to show street-level detail. Actual zoom levels are dependent on the bounds of the named map.

At very high levels (16 and above), the number of tiles is huge. For faster retrieval, you can pre-render the tiles if you know the map layers will not change. Or for more dynamic map tiles, you can cache them and provide an expiration date so that new tiles are generated when necessary.

Rasters and Grids

Raster and grid images are special kinds of layers. These are image files plus a registration point TAB file that allows the image to be rendered as a map layer. You can create a registration point file in MapInfo Professional.

A raster is typically a background layer used for reference: aerial photo, topo map, custom graphics. User-controlled properties of raster images include brightness, contrast, opacity, grayscale and transparency.

A grid is an overlay layer that shows a continuous gradation of color to represent interpolated values from the underlying data. User-controlled properties of a grid include brightness, contrast, opacity, grayscale, transparency, and an inflection collection (how the colors are defined and spread).

Stages

The Location Intelligence Module consists of several stages or components:

- **Find Nearest** – Locates the points of interest (POI) that are nearest to a given location.
- **Query Spatial Data** – Allows you to query a spatial database using MI SQL functions that can, for example, search for a point within a given polygon.
- **Read Spatial Data** – Allows you to access spatial data in a variety of commonly-used spatial data formats using MI SQL functions.
- **Spatial Calculator** – Performs string and geometry conversions as well a variety of calculations on spatial data, such as determining the area of a polygon or distance between two points.
- **Spatial Union** – Returns a geometry object which represents the union of two input geometry objects.
- **Write Spatial Data** – Allows you to insert data into a named table or update data in a named table as output of a job. You can also delete data.

- **Closest Site** – Determines which sites are closest to a given location.
- **Point In Polygon** – Determines whether or not a location resides in a given area.

Enterprise Routing Module

The Enterprise Routing Module provides the ability to obtain driving or walking directions, calculate drive time and drive distance, and identify locations within a certain time or distance from a starting point.

Components

The Enterprise Routing Module consists of the following stages:

- **Get Travel Boundary**—Allows you to obtain polygons corresponding to an isochrone or isodistance calculation. 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.
- **Get Travel Cost Matrix**—Calculates the travel time and distances between an array of start and end locations.
- **Get Travel Directions**—Allows you to route from one point to another point or from one point to multiple other points.

Enterprise Geocoding Module

The Enterprise Geocoding Module performs address standardization, address geocoding, and postal code centroid geocoding. You can enter an address and get outputs such as geographic coordinates, which can be used for detailed spatial analysis and demographics assignment. You can also enter a geocode, a point represented by a latitude and longitude coordinate, and receive address information about the provided geocode.

Components

Enterprise Geocoding Module consists of the following stages. The specific stages you have depend on your license.

- **Geocode Address AUS**—Takes an address in Australia and returns latitude/longitude coordinates and other information.

  Note: Geocode Address AUS has been deprecated. GNAF PID Location Search is the only stage used from Geocode Address AUS. For all other Australia geocoding functions, use the Geocode Address Global component.
- **Geocode Address GBR**—Takes an address in Great Britain and returns latitude/longitude coordinates and other information.

Note: Geocode Address GBR supports the GBR AddressBase Plus data source. Use Geocode Address Global for the GBR Streets (TomTom) data source.

- **Geocode Address Global**—Takes an address in any supported country and returns latitude/ longitude coordinates and other information. Geocode Address Global geocodes addresses only from countries you have licensed. It does not support Australia and Great Britain.
- **Geocode Address World**—Takes an address located in any of the supported countries and returns the city centroid or, for some countries, postal centroid. Geocode Address World cannot geocode to the street address level.
- **Geocode Address Africa**—Provides street-level geocoding for many African countries. It can also determine city or locality centroids, as well as postal code centroids for selected countries.
- **Geocode Address Middle East** Provides street-level geocoding for many Middle East countries. It can also determine city or locality centroids. Middle East supports both English and Arabic character sets.
- **Geocode Address Latin America** Provides street-level geocoding for many Latin American countries. It can also determine city or locality centroids. There is postal code coverage for selected countries.
- **Geocode US Address**—Takes an input address and returns latitude/longitude coordinates and other address information.
- **GNAF PID Location Search**—Identifies the address and latitude/longitude coordinates for a Geocoded National Address File Persistent Identifier (G-NAF PID).
- **Reverse APN Lookup**—Takes an Assessor's Parcel Number (APN), Federal Information Processing Standards (FIPS) county code, and FIPS state code and returns the address of the parcel.
- **Reverse Geocode US Location**—Takes as input a geocode (latitude and longitude coordinate) and returns the address of the location.

GeoConfidence Module

The GeoConfidence Module is used to determine the probability that an address or street intersection is within a given area. The module takes an address or intersection's location (determined by Geocode US Address), converts that location to a point, line, or polygon (depending on the precision of the match), then compares that shape with a database of known shapes to see if the two overlap, and the percentage overlap. For example, you could use the GeoConfidence Module to make decisions on a flood zone rating based on how much overlap there is between an address's location and the flood zone data. Anything greater than a 95% overlap with a 100-year flood zone may indicate that the address is in the flood zone. Conversely, anything less than 95% could cause your business process to send the address to exception processing that might include a manual review.

An address or intersection can be geocoded to a point, an address along a street segment (an array of street segment points), ZIP + 4 centroid, ZIP + 2 centroid, or ZIP Code centroid (polygons). You can use these shapes (points, lines, or polygons) to compare with other shapes to determine overlap, which can be used to determine a risk or probability.

Different geoconfidence polygons are generated depending on the GeoConfidence result returned by the Enterprise Geocoding Module. For more information about the GeoConfidence information returned by the Enterprise Geocoding Module, see the Enterprise Geocoding Module documentation.

The GeoConfidence Module supports U.S. locations only.

> **Note:** GeoConfidence uses services provided by the Enterprise Geocoding and Location Intelligence modules.

Components

GeoConfidence deploys three dataflows that you can modify in Enterprise Designer. Each dataflow consists of various components that were installed with the Enterprise Geocoding and Location Intelligence modules.

For information about each component in the installed dataflows, see the relevant component chapter in the *Spectrum™ Technology Platform User's Guide*.

The names of the dataflows are:

- **GeoConfidenceSurface** This is the dataflow that creates the geoconfidence surface that can be used for further analysis. The input is the GeoConfidence information that is returned from the Enterprise Geocoding Module. Currently, only the Geocode US Address stage can return this information.
- **CreatePointsConvexHull** This is a subflow that is used by the GeoConfidenceSurface template. You should not need to make any changes to this subflow.
- **FloodRiskAnalysis** This is an example dataflow.

4 - Named Resources

A named resource is a form of mapping data, such as a map data file or a database table connection, to which a name has been attached. Attaching names to resources offers the following advantages:

- Allows a resource to be known by its name and not by its properties.
- Allows a resource to be located in one spot but be referenced from many locations, which makes administration of resources easier.
- To change the look or behavior of applications or data, only the resource needs to be changed, not each application or data file.

Defining mapping data as named resources enables you to add them to a JCR repository that is accessible to the Repository Service. After a named resource has been added to the repository, you can refer to it by name in XML requests.

In this section

Named Maps

A named map defines the properties of a map, that is, the data, themes, and display conditions that define the content and appearance of a map. A map is always composed of at least one map layer.

For more information about named maps, see the Resources and Data section of this guide.

Named Tables

A named table defines the properties of a mapping data source, such as a connection to a database that contains tables, or a connection to a particular database table.

The spatial data contained in a data source is what is used to construct a map layer.

Currently the supported connections are:

- TAB files
- Shapefiles
- GeoPackages
- Oracle databases
- SQL Server databases
- PostgreSQL/PostGIS databases
- Generic JDBC databases (includes X/Y and SAP HANA)
- View tables (query)

Named tables are created using Spatial Manager.

For more information about named tables, see the Resources and Data section of this guide.

Named Layers

A named layer defines the properties of a map layer. Map layers are used to make up a map, and a map is always composed of at least one map layer. Each map layer is populated with spatial data from a data source such as a TAB file.

For more information about named layers, see the Resources and Data section of this guide.

Named Tiles

A named tile defines the properties of a map tile. A map tile is a rectangular piece of a map and, like pieces of a jigsaw puzzle, map tiles fit together to make up an entire map. A named tile defines the bounds of the map tile and the larger map that the tile is part of. Dividing a map into separate map tiles can greatly reduce the time required to display a map, particularly if you want to display only a small part of it.

Named tiles are created using Spatial Manager.

For more information about named tiles, see the Resources and Data section of this guide.

Named Styles

A named style defines the properties of a map style. Styles enable you to control the visual appearance of your maps by specifying the visual characteristics of various map elements, such as lines, filled areas, and symbols.

For more information about named styles, see the Resources and Data section of this guide.

Named Connections

A named connection is a named resource that contains a name and a connection to a data source and is stored in the repository. The name can be referenced in named tables without having to include the connection information. If the connection changes, only the named connection needs to be updated. The named tables that use the named connection will automatically reference the updated connection.

Named connections are created using Spatial Manager.

For more information about named connections, see the Resources and Data section of this guilde.

Named WMTS Layers

A named WMTS layer is a type of named resource that defines the named tile that is exposed through OGC Web Map Tile Service (WMTS) services. This contains the reference of named tile that is stored in the repository.

Note: You can also view, create, edit, and delete WMTS layers from the Services view (**Services > WMTS > Layers**).

For more information about named WMTS layers, see the Resources and Data section of this guide.

Named Label Sources

A named label source is a type of named resource that can be referenced by a named or inline label layer. It contains label properties such as labeling rules and display properties.

For more information about named label sources, see the Resources and Data section of this guide.

5 - Tools

The Spectrum™ Technology Platform ships with several tools for managing named resources. Some are browser-based, while others are windows applications and command line utilities.

In this section

Spatial Manager

Spatial Manager

Spatial Manager is a web application for browsing and managing named resources. With Spatial Manager you can:

- Create and modify named connections, tables, WMTS layers, and tiles
- View details for named connections, tables, maps, layers, WMTS layers, and tiles
- Configure WMS, WFS, and WMTS settings as well as create and modify WMS and WMTS layers and WFS feature types
- Configure the Feature Service and Mapping Service
- Search, rename, move, and delete resources that exist in the repository
- Create, rename, and delete folders in the repository

To preserve connections between resources, Spatial Manager is the preferred tool for renaming, moving, or deleting named resources.

For more information about Spatial Manager, see the Utilities section of the *Spectrum Spatial Guide*.

Management Console

Management Console is a browser tool for managing many aspects of Spectrum. Management Console can be found on the Spectrum Welcome Page after installation, under the Platform Client Tools section.

Use Management Console with the Location Intelligence module to configure the pool size, which is the number of requests that the module's remote component can handle concurrently.

Use Management Console with the Enterprise Routing module to configure database resources for use with the routing services.

Enterprise Designer

Enterprise Designer is a visual tool for creating dataflows. Using this client, you can:

- Create and modify jobs, services, subflows, and process flows
- Test dataflows for problems
- Expose and hide services
- Generate reports

Installing the Client Tools

The Spectrum™ Technology Platform client tools are applications that you use to administer your server and design and run flows. The client tools are:

Enterprise Designer	Enterprise Designer is a Windows application that you use to create, modify, and run flows.
Job Executor	Job Executor is a command line tool that you can use to run a job from a command line or script. The job must have been previously created and saved on Spectrum™ Technology Platform using Enterprise Designer.
Process Flow Executor	Process Flow Executor is a command line tool that you can use to run a process flow from a command line or script. The process flow must have been previously created and saved on Spectrum™ Technology Platform using Enterprise Designer.
Administration Utility	The Administration Utility provides command line access to administrative functions. You can use it in a script, allowing you to automate certain administrative tasks. You can also use it interactively.

Note: Before installing, be sure to read the release notes. The release notes contains important compatibility information as well as release-specific installation notes.

To install the client tools:

1. Open a web browser and go to the Spectrum™ Technology Platform Welcome Page at:

   http://<servername>:<port>

   For example, if you installed Spectrum™ Technology Platform on a computer named "myspectrumplatform" and it is using the default HTTP port 8080, you would go to:

   http://myspectrumplatform:8080
2. Click **Platform Client Tools**.
3. Download the client tool you want to install.

Enterprise Designer Examples

Enterprise Designer is a visual tool for creating dataflows. This section provides an example of a Point in Polygon flow.

Point in Polygon Subflow

This procedure describes how to create a subflow using the Query Spatial Data stage to deterimine if a point is located within a given polygon.

This subflow takes as the source an x and y and creates a point geometry using Spatial Calculator's Create Point operation. The resulting geometry is the input for the Query Spatial Data stage which is then output to an output sink.

This procedure is one way to find points in a polygon. An alternative is the Point in Polygon stage that uses a Centrus database. Both operations are discussed in the "Point In Polygon Best Practices" topic in the Development section of the *Spectrum Spatial Guide*.

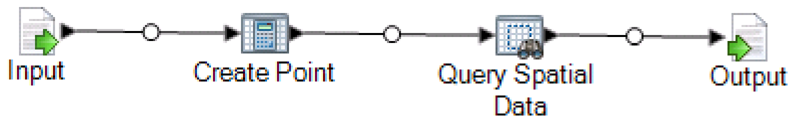
To create a point in polygon subflow:

1. In Enterprise Designer, choose **New** > **Dataflow** > **Subflow** from the File menu or **New** > **Subflow** from the Task pane.

   For more information about creating subflows, see the *Dataflow Designer's Guide* from the Help menu.
2. Drag an Input icon onto the canvas and double-click it to display the Options dialog.
3. Click the **Add** button to display the Add Custom field dialog. Click **Add** and provide the field name 'x' and the type as double. Click **OK**.
4. Repeat Step 3 to add a field 'y' of type double. Click **OK** to leave the Input Options dialog.
5. From the Location Intelligence Stages section in the Palette, drag Spatial Calculator onto the design canvas. Click the solid black triangle on the right side of the Input stage (the output port) and drag it to the left side of the Spatial Calculator stage to create a channel.
6. Double-click on Spatial Calculator to display the Options dialog. Choose **Create Point Geometry** from the list of operations. Change the coordinate system if necessary. Click **OK**. Rename the Spatial Calculator to Create Point so you can remember what it does.
7. Drag the Query Spatial Data stage onto the canvas. Connect the output port of the Create Point operation to the input port of the Query Spatial Data stage.
8. Double-click on the Query Spatial Data stage to display the Options dialog. Create the MISQL statement in the box provided and click **Verify**.

   Here is an example statement. Modify the field and named table to match your data.

   ```
   select STATE from "/Samples/NamedTables/USA" where Obj contains
    ${Geometry}
   ```
9. Drag an Output sink onto the canvas. Connect the output port of the Query Spatial Data stage to the input port of the Output sink.
10. Double-click on the Output sink and check the field you wish to expose. In our example, check the State field. Click **OK**.



11. Save the subflow. To expose the subflow for use in a dataflow, choose **File Expose/ Unexpose and Save** or click the **Expose/Unexpose and Save** button. The subflow displays in the User-Defined Stages folder.

    The subflow is now ready to incorporate into a Point in Polygon **dataflow**.

Point in Polygon Dataflow (Service)

This procedure describes how to create a Point in Polygon dataflow from a Point in Polygon subflow that already contains the necessary processing instructions. By creating this dataflow as a service it can be accessed from a web service or when using the Spectrum™ Technology Platform API. Alternatively, this dataflow could be created as a job for batch processing. A subflow must be incorporated into a dataflow to be used.

To create a point in polygon dataflow:

1. In Enterprise Designer, choose **New** > **Dataflow** > **Service** from the File menu or **New** > **Service** from the Task pane.
2. Drag the **Point in Polygon subflow** from the User-Defined Stages folder.
3. Drag an Input source onto the canvas and connect the output port to the Point In Polygon subflow.
4. Double-click on Input to display the Options dialog. Expose the x and y fields to the dataflow by clicking the Expose checkbox. Click **OK**.
5. Drag an Output sink onto the canvas and connect the output port of the Point in Polygon subflow to the input port of the Output.
6. Double-click on Output to display the Options dialog. Expose the State field by clicking the checkbox alongside the State field.
7. To expose the dataflow, choose **File Expose/Unexpose and Save** or click the **Expose/ Unexpose and Save** button. The dataflow displays in the User-Defined Stages folder.
8. To verify that the service is now exposed as a web service, go to one of the following URLs:

   For REST: http://<server>:<port>/rest

   For SOAP: http://<server>:<port>/soap

Point in Polygon Dataflow (Job)

This procedure describes how to create a Point in Polygon dataflow from a Point in Polygon subflow that already contains the necessary processing instructions. By creating this dataflow as a service it can be accessed from a web service or when using the Spectrum™ Technology Platform API. Alternatively, this dataflow could be created as a service for use with a web service. A subflow must be incorporated into a dataflow to be used.

To create a point in polygon dataflow:

1. In Enterprise Designer, choose **New** > **Dataflow** > **Job** from the File menu or **New** > **Job** from the Task pane.
2. Drag the Point in Polygon subflow from the User-Defined Stages folder.
3. Drag a Read from File icon onto the canvas and connect the output port to the Point In Polygon subflow.
4. Double-click on Read from File to display the Options dialog. In the **File Properties** tab, specify the input file and other information as needed.
5. In the **Fields** tab click **Regenerate** and click **Yes** to populate the list of fields in the input file. Click **Detect Type** to determine the data type for each field. Click **OK**.
6. Drag a Write to File sink onto the canvas and connect the output port of the Point in Polygon subflow to the input port of the Output.
7. Double-click on Write to File to display the Options dialog. In the **File Properties** tab specify the output file and information as needed.
8. In the **Fields** tab click **Quick Add**, then click **Select All** and click **OK**.
9. Select **File Save**. Give your dataflow a name and click **OK**.
10. To test the dataflow, select **Run Run Current Flow**.

MapInfo Pro

MapInfo Pro is a location intelligence application that offers world class map data creation and spatial analysis functionality. Consider it a co-application with Spectrum Spatial where GIS users build the data and mapping content that is then shared with other users across the organization via Spectrum Spatial's WMS, WMTS, and WFS services. The tool to get the data from MapInfo Professional to Spectrum Spatial is the MapBasic application **Map Uploader**. Once the data is in Spectrum Spatial it can be further analyzed and manipulated through the **Enterprise Designer** and the Feature Service.

Data Compatibility

Spectrum Spatial supports nearly every data format that is familiar to MapInfo Professional users, including native .TAB and native extended .TAB, .TAB DBF, Shapefiles, RDBMS, and rasters and grids. Maps and data created in any version of MapInfo Professional can be brought into Spectrum Spatial. Both products support the same themes, styles, override capabilities and coordinate systems.

Data Connectivity

Both MapInfo Professional and Spectrum Spatial can connect to the same database where no data has to be moved. Spectrum Spatial supports the same remote databases, Oracle, MS SQL Server, PostgreSQL/PostGIS, and SQLite. Both products can access .TAB files on a network file server.

Data Management

Unlike MapInfo Professional, Spectrum Spatial does not use workspaces that store paths to .TAB files. Instead it has map, layer and table definitions that live in a repository and reads those resources to render maps and access data. These resources can be used as a tool to organize your content in one central repository. The resources can point to .TAB files and databases from a hierarchy that doesn't care where the data lives. Only the resources need to be updated if the data is moved.

Data Manipulation and Analysis

Spectrum Spatial supports the most common geometry processing operations, such as Combine (Union), Buffer, Convex Hull, through its dataflow designer or through SOAP/REST API calls. You can search your data via the Spectrum Spatial Feature Service using query syntax that you are already familiar with.

Capability Differences

MapInfo Professional:

- Table creation, editing, schema design and editing, saving and deleting.
- Read MS Access, Excel, ascii, text (CSV) and Lotus files.
- Large set of geometry processing operations including Aggregate operations (combine, erase, split)
- Supports raster reprojection
- Supports queries based on MapBasic SQL
- Desktop user interface for all operations.

Spectrum Spatial:

- Pass down queries to a database for better performance
- Query and create themes on seamless tables
- Supports queries based on MI SQL
- Supports repeatable dataflow data operations that integrate Geocoding, Routing, Spatial as well as all other Spectrum modules.
- Service oriented architecture (SOA) built for scalability and multiple users.

Map Uploader

Map Uploader

Map Uploader is a plug-in for MapInfo Professional (v11.5 or above) where you can add named resources (named maps, named layers, and named tables) to the repository.

Download the Map Uploader installer and view its documentation on the Spectrum Spatial section of the Welcome Page, under the Utilities tab. Documentation for the Map Uploader is in the Utilities section of the *Spectrum Spatial Guide*.

MapInfo Workspace (MWS) Import

The `limrepo mwsimport` command in the Spectrum™ Technology Platform Administration Utility allows you to provision a map from a MapInfo Workspace (MWS) file that has been created either by MapInfo Pro or the MapXtreme Workspace Manager into the Spectrum Spatial repository. The import will create the named map and all its dependent resources (layers, tables and connections). The connection is named by appending 'Connection' to the map name. The named tables and named layers are created in subfolders (NamedTables and NamedLayers, respectively).

See the Spectrum™ Technology Platform *Administration Guide* for more information.

6 - Services

The capabilities of Spectrum Spatial are made available as web services using open standards such as SOAP or REST. To access the web services, you construct an HTTP request and submit it directly to the web service. The HTTP request can be in the form of a SOAP request, a REST request, or a POST/GET request, and can be submitted directly from a web browser or from a client application.

In this section

Mapping Service

The Mapping Service provides a simplified interface to perform Mapping Service rendering, themes, overlays, and conversions. The Mapping Service provides both simple operations that are more frequently used, or more complex operations that require higher levels of customization.

The Mapping Service provides operations that:

- list, get and describe maps, layers, styles, and tables in the repository
- conversion methods between real world coordinates, XY, and screen coordinates
- screen calculations such as length
- navigation methods such as zoom, and pan
- render maps and layers
- overlay, override, and theme maps
- customize your map with legends, watermarks, and other adornments.

In Spectrum Spatial, data access and mapping is independent. The tables define the data access, and the repository takes care of the table management. Therefore, with the Mapping Service, you are able to render maps and perform mapping operations without the worry of where the data is located.

The Mapping Service is configured and managed using the Spatial Manager utility. See the *Spatial Manager Guide* under the Utilities section for more information.

Render a Map

Rendering a map involves making a RenderMap request to the Spectrum Spatial server. For learning purposes, we will use the Mapping Demo Page that include a list of common SOAP requests.

To render a map:

1. Go to the Spectrum Spatial SOAP Mapping Service Demo Page at `http://<server>:<port>/Spatial/MappingService/DemoPage.html`
2. Choose Render Named Map from the drop-down list of mapping requests and click Submit.

   The Spectrum Spatial Mapping Service will process the request and return a SOAP response and an image of the named map. The named map associated with this demo request is using a sample named map that was uploaded to the repository at installation time.
3. To view the response and image for one of your named maps, edit the request and replace the text /Samples/NamedMaps/USA with your named map and click Submit.

   You may need to change the map request to use the center and zoom width appropriate for your data.

Feature Service

The Feature Service provides the ability to search spatial databases. It provides a set of common operations that you can use to query content, regardless of the underlying data provider. The advantage of this is that you can run the exact same operation against many different content stores (such as a native TAB file, an Oracle database table, or a SQL Server database table) by specifying different named tables; the same query will work on each table type.

The Feature Service provides operations that:

- list all tables in the catalog
- describe specific tables in the catalog
- search within a distance, polygon or envelope
- search intersections of geometries and/or bounding boxes
- search within a polygon or bounding box
- search for the nearest feature or edge
- search using a custom MapInfo SQL query
- insert and update features into a database table

The Feature Service is configured and managed using the Spatial Manager utility. See the *Spatial Manager Guide* under the Utilities section for more information.

Search at Point - Java Example

This Java example calls the Feature service to do a Search at Point call. The Search at Point, when used against a table with polygons, is a point in polygon method.

To generate the stub classes to access the Feature Service, see **Generating Stub Code** on page 44

```
import com.mapinfo.midev.service.feature.v1.SearchAtPointRequest;
import com.mapinfo.midev.service.feature.v1.SearchAtPointResponse;
import com.mapinfo.midev.service.feature.ws.v1.FeatureService;
import
 com.mapinfo.midev.service.feature.ws.v1.FeatureServiceInterface;
import com.mapinfo.midev.service.feature.ws.v1.ServiceException;
import com.mapinfo.midev.service.geometries.v1.Point;
import com.mapinfo.midev.service.geometries.v1.Pos;
import com.mapinfo.midev.service.table.v1.NamedTable;
import com.mapinfo.midev.service.table.v1.Table;

import javax.xml.ws.BindingProvider;
import java.util.Map;

public final class FeatureServiceSample {
 // username with the authority to call the Feature service
 private static final String USERNAME = "admin";

 // password of the caller
 private static final String PASSWORD = "admin";

 public static void main(String[] args) throws ServiceException {
  // get the endpoint class
  FeatureServiceInterface featureServiceInterface = new
 FeatureService().getFeatureServiceInterface();

  // if authentication is enabled in Spectrum then the following is
 required
  Map<String, Object> requestContext =
 ((BindingProvider)featureServiceInterface).getRequestContext();
  requestContext.put(BindingProvider.USERNAME_PROPERTY, USERNAME);
  requestContext.put(BindingProvider.PASSWORD_PROPERTY, PASSWORD);

  // start assembling the request
  SearchAtPointRequest searchAtPointRequest = new
 SearchAtPointRequest();

  // create the search point
  {
   Point point = new Point();
   point.setSrsName("epsg:4326");
   Pos pos = new Pos();
   pos.setX(-75.66);
   pos.setY(47.88);
   point.setPos(pos);

   searchAtPointRequest.setPoint(point);
  }

  // set the named table
  {
   NamedTable table = new NamedTable();
   table.setName("/Samples/NamedTables/USA");
   searchAtPointRequest.setTable(table);
  }

  // send the request
  SearchAtPointResponse searchAtPointResponse =
 featureServiceInterface.searchAtPoint(searchAtPointRequest);

  // do what you want with the response ...

 }

}
```

Search at Point - SOAP

The following SOAP Search at Point request against a table of polygons returns a list of points that are contained within the polygon at point -75, 42.

```
<?xml version="1.0"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="http://www.mapinfo.com/midev/service/feature/v1"
 xmlns:ns2="http://www.mapinfo.com/midev/service/geometries/
v1" xmlns:ns3="http://www.mapinfo.com/midev/service/table/v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <S:Header/>
  <S:Body>
      <ns1:SearchAtPointRequest>
        <ns3:Table xsi:type="ns3:NamedTable" name="/Samples/
NamedTables/USA"/>
         <ns2:Point srsName="EPSG:4326">
            <ns2:Pos>
               <ns2:X>-75</ns2:X>
               <ns2:Y>42</ns2:Y>
             </ns2:Pos>
         </ns2:Point>
      </ns1:SearchAtPointRequest>
   </S:Body>
</S:Envelope>
```

To view the SOAP response, go to the Feature Service SOAP demo page at `http://<server>:<port>/Spatial/MappingService/DemoPage.html` and submit the Search at Point request.

Search at Point - REST

The following REST request performs a point in polygon search.

```
http://<machine>:<port>/rest/Spatial/FeatureService/tables/Samples/
NamedTables/USA
/features.json?q=SearchAtPoint&point=-75,42,epsg:4326
```

The JSON response:

Note: Some of the points have been removed from this example to shorten it.

```
{
 "type": "FeatureCollection",
 "Metadata": [{
  "type": "Geometry",
  "name": "Obj",
  "crs": {
   "type": "name",
   "properties": {
    "name": "epsg:4267"
   }
  },
  "bbox": [-79.762182,
  40.4966,
  -71.85664,
  45.012533],
  "styleColumn": "MI_Style"
 },
 {
  "type": "Style",
  "name": "MI_Style"
 },
 {
  "type": "String",
  "name": "State_Name"
 },
 {
  "type": "String",
  "name": "State"
 },
 {
  "type": "String",
  "name": "Fips_Code"
 },
 {
  "type": "Decimal",
  "name": "Pop_1990",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Num_Hh_1990",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Integer",
  "name": "Num_Hh_2000"
 },
 {
  "type": "Decimal",
  "name": "Med_Inc_1990",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Double",
  "name": "Med_Inc_2000"
 },
 {
  "type": "Integer",
  "name": "Pop_Urban_2000"
 },
 {
  "type": "Integer",
  "name": "Pop_Rural_2000"
 },
 {
  "type": "Decimal",
  "name": "Pop_Male",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Female",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Cauc",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Black",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Native",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Asian",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Other",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Sales_1990",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "AmerIndianAlaskaNat_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "AsianHawaiianAlone_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Black_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Cauc_Alone_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Pop_Asian_Alone_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Area_in_Miles_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "Decimal",
  "name": "Area_in_Km_2000",
  "fractionalDigits": 0,
  "totalDigits": 10
 },
 {
  "type": "String",
  "name": "URL"
 },
 {
  "type": "Double",
  "name": "Distance"
 }],
 "bbox": [-79.762182,
 40.4966,
 -71.85664,
 45.012533],
 "crs": {
  "type": "name",
  "properties": {
   "name": "epsg:4267"
  }
 },
 "features": [{
  "type": "Feature",
  "id": "33",
  "geometry": {
   "type": "MultiPolygon",
   "coordinates": [[[[-79.762182,
   42.2698029],
   [-79.75,
   42.273044],
   [-79.721352,
   42.282801],
   [-79.712698,
   42.287149],
   [-79.708078,
   42.288247],
   [-79.701425,
   42.291359],
   [-79.699361,
   42.291497],
   [-79.6910439,
   42.295637],
   [-79.6898729,
   42.295683],
   ...
   [-73.8109,
   40.6057999]]]]
  },
  "properties": {
   "MI_Style": {
    "type": "MapBasicAreaStyle",
    "MapBasicPen": {
     "width": 1,
     "pattern": 2,
     "color": "8421504"
    },
    "MapBasicBrush": {
     "pattern": 2,
     "foregroundColor": "16767152",
     "backgroundColor": "16777215"
    }
   },
   "State_Name": "New York",
   "State": "NY",
   "Fips_Code": "36",
   "Pop_1990": 17990455,
   "Pop_2000": 18976457,
   "Num_Hh_1990": 6639322,
   "Num_Hh_2000": 7056860,
   "Med_Inc_1990": 32965,
   "Med_Inc_2000": 43393.0,
   "Pop_Urban_2000": 16601126,
   "Pop_Rural_2000": 2375331,
   "Pop_Male": 8625673,
   "Pop_Female": 9364782,
   "Pop_Cauc": 13385255,
   "Pop_Black": 2859055,
   "Pop_Native": 62651,
   "Pop_Asian": 693760,
   "Pop_Other": 989734,
   "Sales_1990": 124478903,
   "AmerIndianAlaskaNat_2000": 82461,
   "AsianHawaiianAlone_2000": 1053794,
   "Pop_Black_2000": 3014385,
   "Pop_Cauc_Alone_2000": 12893689,
   "Pop_Asian_Alone_2000": 1044976,
   "Area_in_Miles_2000": 48785,
   "Area_in_Km_2000": 126352,
   "URL": "http://www.state.ny.us/",
   "Distance": 0.0
  }
 }]
}
```

Geometry Service

The Geometry Service provides a simplified interface to perform Geometry Service measurements, conversions, and operations. The Geometry Service provides the following capabilities:

- Aggregate (buffer, centroid, envelope, union, intersection, convex hull, difference, symmetrical difference)
- Transformation (coordinate transformations)
- Measurement (area, distance, length, perimeter)

• Predicate (contains, intersects, within)

This service provides a set of common operations that you can use to perform operations on content, regardless of the data provider. One advantage is that you can run the exact same operation against many different content repositories (e.g., a native TAB file, an Oracle table, or a SQL Server table) by specifying different named tables; the same operation will work on each table type. Using the Geometry Service, you only need to worry about the request, not the location or type of data in the repository.

The Geometry Service is based on the well known mathematical concepts described in the OGC Simple Features Specification.

Create Buffer - Java Example

This Java example shows how to use the SOAP interface to the Geometry Service to create a buffer around a point.

To generate the stub classes for the Geometry Service, see **Generating Stub Code** on page 44

```
import com.mapinfo.midev.service.geometries.v1.MultiPolygon;
import com.mapinfo.midev.service.geometries.v1.Point;
import com.mapinfo.midev.service.geometries.v1.Pos;
import com.mapinfo.midev.service.geometry.v1.BufferRequest;
import com.mapinfo.midev.service.geometry.v1.BufferResponse;
import com.mapinfo.midev.service.geometry.ws.v1.GeometryService;
import
 com.mapinfo.midev.service.geometry.ws.v1.GeometryServiceInterface;
import com.mapinfo.midev.service.geometry.ws.v1.ServiceException;
import com.mapinfo.midev.service.units.v1.Distance;
import com.mapinfo.midev.service.units.v1.DistanceUnit;

import javax.xml.ws.BindingProvider;
import javax.xml.ws.handler.MessageContext;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public final class GeometryServiceDriver {

 // the number of line segments that will compose the buffer
 private static final Integer RESOLUTION = 64;

 // username with the authority to call the Geometry service
 private static final String USERNAME = "admin";

 // password of the caller
 private static final String PASSWORD = "admin";

 public static void main(String[] args) throws ServiceException {
  // get the endpoint class
  GeometryServiceInterface geometryServiceInterface = new
 GeometryService().getGeometryServiceInterface();

  // if authentication is enabled in Spectrum then the following is
 required
  Map<String, Object> requestContext =
 ((BindingProvider)geometryServiceInterface).getRequestContext();
  requestContext.put(BindingProvider.USERNAME_PROPERTY, USERNAME);
  requestContext.put(BindingProvider.PASSWORD_PROPERTY, PASSWORD);

  BufferRequest bufferRequest = new BufferRequest();

  // point to buffer
  {
   Point point = new Point();
   point.setSrsName("epsg:4326");
   Pos pos = new Pos();
   pos.setX(-75.66);
   pos.setY(47.88);
   point.setPos(pos);

   bufferRequest.setGeometry(point);
  }

  // buffer distance
  {
   Distance bufferDistance = new Distance();
   bufferDistance.setUom(DistanceUnit.MILE);
   bufferDistance.setValue(12.0);
   bufferRequest.setDistance(bufferDistance);
  }

  bufferRequest.setResolution(RESOLUTION);

  BufferResponse bufferResponse =
 geometryServiceInterface.buffer(bufferRequest);

  MultiPolygon bufferGeometry = (MultiPolygon)
 bufferResponse.getGeometry();

  // do what you want with the bufferGeometry ...

 }

}
```

Generating Stub Code

Use the following steps to generate the stub code for a Java web services client application, using the `wsimport` tool that is included with the Java Development Kit.

1. Open an xterm or command prompt window and change to the directory that contains the `wsimport` program, usually `JDK_install_dir/bin/`.
2. At the command prompt, type `wsimport -s source_dir -d classes_dir URL_TO_WSDL` and press Enter.

   For example, to generate stub classes for the Geometry Service, enter the following command:

   ```
   wsimport -s output/source -d output/classes http://<host:port>/soap/
   GeometryService?wsdl
   ```

After the command has been executed, the generated source `.java` files are placed within the directory you specified with the `-s` option, and the compiled `.class` files are placed within the directory you specified with the `-d` option.

Map Tiling Service

The Map Tiling Service dynamically renders subsets of a map on a per-request basis. This subset is called a tile. The tiles rendered from the Map Tiling Service can be used individually, or combined to form larger maps, in applications for seamless map interaction. This service provides fast, simple, light-weight map rendering where more complex map rendering can be performed using the Mapping Service.

This service provides operations that:

- lists the available tiles
- describes the metadata for a specific named tile
- gets tiles based on the input parameters

Web Feature Service (WFS)

The Web Feature Service (WFS) is used for searching, obtaining metadata descriptions, querying, and filtering spatial data (feature types) at the service level. The XML interface and syntax follow the WFS 1.0.0 and 1.1.0 OGC specifications. Both SOAP and HTTP POST/ GET requests are supported. As a result, a standard compliant WFS client, such as MapInfo Professional, can access data by submitting an XML request through HTTP to display it on a map or to get vector geometries for calculations.

Use the Web Feature Service to help perform a search for features within a given distance from a point (or other type of geometry). For example, a real estate application determines a realistic value for a home by comparing the distance of it to numerous features, including railroad tracks, highways, shopping malls, and police stations. This application would call WFS to select a highway feature and a home (a point). The application would then calculate distance values for proprietary rating calculations.

WFS provides operations that:

- describe the service capabilities
- get the schema description for features
- get features and information about them

Web Map Service (WMS)

The Spectrum™ Technology Platform Web Map Service (WMS) allows software clients to reference map images over the Internet or a private intranet. The WMS implementation is based on the WMS 1.1.1 and 1.3.0 OGC specification. Using HTTP GET requests, the WMS provides georeferenced data to a client as an image for that would then displays this data as an image. Georeferenced data is information associated with maps that describe the real world extents of specific features and the projection upon which it is based.

The images can be provided as GIF, JPEG, PNG, and other image formats.

WMS provides operations that:

- describe the service capabilities such as available layers and coordinate system
- get an image of a map
- get information about features

Web Map Tile Service (WMTS)

Spectrum Spatial provides an OGC-compliant Web Map Tile Service (WMTS) built to the OCG specifications for WMTS versions 1.0.0 (located at **http://www.opengeospatial.org/standards/wmts**). WMTS is a standard implemented by OGC to provide a performance oriented and scalable service to its users. A WMTS server achieves this by using image caching strategies to serve pre-rendered georeferenced map tiles. Spectrum Spatial allows clients to request and servers to deliver tiled mapping data over the Internet or private intranet through the HTTP protocol. Any client that is built to the OGC specification can view your service and make requests for map tiles.

The Spectrum Spatial WMTS provides operations that:

- describe the service capabilities

- get a map tile

The georeferenced resources provided in a basic WMTS are organized into "WMTS layers".

Named Resource Service

The Named Resource Service provides the ability to work with resources in the repository. It provides a set of common operations that you can use to add, list, update, delete, and search resources in the repository. The advantage of this is that you can administer the repository's resources at the service level, with a simple SOAP API that allows you to manage your content.

The Named Resource Service provides operations that:

- list all resources (or a subset) in the repository
- add a resource to the repository
- modify a resource that exists in the repository
- delete a resource from the repository
- read a resource from the repository to return the definition
- search for resources in the repository

Enterprise Routing Services (REST)

The stages in the Enterprise Routing Module have been exposed as REST web services.

GetRoute

Description

The GetRoute service returns routing information for a set of two distinct points or multiple points. It takes a starting location and an ending location with optional intermediate points as input, and returns the route that is either the fastest or the shortest.

> **Note:** The response from the REST service will be in JSON format and the geometry returned will be in GeoJSON format. When a request contains invalid query parameters in the GET URL or an invalid payload for POST, a cumulative error response will be returned in a JSON array. The `value` node in the response JSON is deprecated. For error checking, only the `errors` node should be utilized.

GetRouteCostMatrix

Description

The GetRouteCostMatrix service calculates the travel time and distances between an array of start and end locations and returns the route that is either the fastest or the shortest. The result determines the total time and distance of the individual routes (the route costs). For example if you input four start points and four end points, a total of 16 routes will be calculated.

> **Note:** The response from the REST service will be in JSON format and the geometry returned will be in GeoJSON format. When a request contains invalid query parameters in the GET URL or an invalid payload for POST, a cumulative error response will be returned in a JSON array. The `value` node in the response JSON is deprecated. For error checking, only the `errors` node should be utilized.

GetSegmentData

Description

The GetSegmentData service returns segment information for a point or segment ID. When a point is specified, the closest route segments are returned. When a segment ID is specified, the route data for that specified route segment is returned.

> **Note:** The response from the REST service will be in JSON format. When a request contains invalid query parameters in the GET URL or an invalid payload for POST, a cumulative error response will be returned in a JSON array. The `value` node in the response JSON is deprecated. For error checking, only the `errors` node should be utilized.

GetTravelBoundary

Description

GetTravelBoundary determines a drive or walk time or distance boundary from a location. This feature obtains polygons corresponding to an isochrone or isodistance calculation. 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. The GetTravelBoundary operation (also known as an iso definition) takes a starting point, a unit (linear or time), one or more costs as input and returns the resulting travel boundary. Costs refer to the amount of time or distance to use in calculating an iso. Multiple costs can also be given as input. In case of multiple costs, costs can also be provided as a comma delimited string.

Note: The response from the REST service will be in JSON format. When a request contains invalid query parameters in the GET URL or an invalid payload for POST, a cumulative error response will be returned in a JSON array. The `value` node in the response JSON is deprecated. For error checking, only the `errors` node should be utilized.

PersistentUpdate

Description

The PersistentUpdate service allows a user to override aspects of the network. The overrides can be done on a per-road type, at a specific point or at a specific segment. The persistent update is valid only for a specific data source and may not be valid after a data update.

Using persistent updates to make these types of modifications, you have the ability to:

- Exclude a point
- Exclude a segment
- Set the speed of a point, segment, or road type
- Change (increase or decrease) the speed of a point, segment, or road type by a value
- Change (increase or decrease) the speed of a point, segment, or road type by a percentage
- List persistent updates

Note: Since persistent updates are changes made on a system-wide basis for routing data and all updates will persist, they should be used with caution. The response from the REST service will be a success message. When a request contains invalid query parameters in the GET URL or an invalid payload for POST, a cumulative error response will be returned in a JSON array. The `value` node in the response JSON is deprecated. For error checking, only the `errors` node should be utilized.

Enterprise Routing Services (SOAP)

The stage in the Enterprise Routing Module has been exposed as a SOAP web services.

GetRouteData

Get Route Data returns routing segment information for a point or segment ID. When you specify a point, the closest route segments are returned. When you specify a segment ID, the route segment for that segment ID is returned.

Note: Get Route Data is only available as a service (Management Console and SOAP web service). Get Route Data is not available through a stage or REST API. It is also not available through the Java, C++, C, .NET, or COM APIs.

Get Route Data is part of the Enterprise Routing Module.

Notices

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com