Envinsa®

Version 4.2

WEB FEATURE SERVICE REFERENCE
Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor or its representatives. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, without the written permission of MapInfo Corporation, One Global View, Troy, New York 12180-8399.

© 2007 MapInfo Corporation. All rights reserved. MapInfo, the MapInfo logo and Envisia are trademarks of MapInfo Corporation and/or its affiliates.

October 3, 2007
# Table of Contents

Chapter 1: Introduction ................................................................. 4  
   Web Feature Functionality ...................................................... 5

Chapter 2: Using WFS Capabilities Through SOAP Requests ............ 7  
   Describing Feature Details ...................................................... 10  
   Getting Available Features ..................................................... 12

Chapter 3: Web Feature Service Fundamentals ................................ 16  
   Specifying Contents in the Request ........................................... 17  
   Understanding Enterprises and Domains .................................... 18  
   Supported Filters ................................................................. 19  
   Functionality Limitations ....................................................... 21  
   Exceptions and Error Handling .............................................. 22

Chapter 4: Using WFS Capabilities Through HTTP Request ............. 23  
   FeatureType and PropertyName Conventions ............................... 29  
   Compliance with GML2 Geometries ......................................... 30  
   Turning off Security Authentication ....................................... 30

Chapter 5: Configuration ............................................................. 31
Welcome to the MapInfo® Envinsa™ Location Platform Web Services. This guide provides descriptions and examples for the developer who is writing applications that will access the Web Feature Service (WFS).

The WFS allows an application to retrieve, query, and filter geospatial Content encoded in Geography Markup Language (GML) from multiple Web Features. Information can be retrieved about a specific feature, or a set of features for analysis. Filters can be applied in the request. For example, select all the water features within a region or custom polygon, such as a new housing development. An application would overlay these retrieved features, such as streets, store locations, and points of interest, onto a map image. Specifically, by adding points and vectors to an image.

In this section:

- What is the Web Feature Service? .........................5
- Web Feature Functionality ..........................5
What is the Web Feature Service?

The Web Feature Service 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 OGC specification. 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.

The WFS has a client SDK on both JAVA and .NET platforms, that currently only supports SOAP requests.

Use WFS 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.

Which type of request are you submitting?

For information on how to use:

- SOAP requests, refer to Chapter 2: Using WFS Capabilities Through SOAP Requests. Also refer to Chapter 3: Web Feature Service Fundamentals for background information.
- XML requests through HTTP, refer to Using WFS Capabilities Through HTTP Request in Chapter 4 on page 23.

Web Feature Functionality

The client workflow involves retrieving geospatial data by first submitting a get capabilities request to determine the list of feature types that the WFS offers. The client then issues a DescribeFeatureType request in order to obtain a description of one or more of the feature types. Finally, the client uses the description of the feature types to formulate a GetFeature operation.

Filters can be applied to the GetFeature operation to limit the results of the response.

Get Capabilities

The Web Feature Service is able to describe its capabilities. It indicates what feature types (Contents) it can service and what operations are supported on each feature type. The service can return what feature types are available based on the feature type alias.

The following section highlights a potential way to apply this capability: Describing Service Capabilities in Chapter 2 on page 8.
Describe Feature Type

The Web Feature Service is able, upon request, to describe the structure of any feature type it can service. The service returns the metadata (description) of a specific feature type (Content) based on the returned GetCapabilities.

The following section highlights a potential way to apply this capability: Describing Feature Details in Chapter 2 on page 10.

Get Feature

The Web Feature Service is able to service a request to retrieve feature instances. In addition, the client is able to specify which feature properties to fetch and is able to constrain the query spatially and non-spatially. The service can return the search results of a feature (Content) based on the description returned in the DescribeFeatureType request.

An application applies filters to retrieve a specific set of features that satisfy the condition specified by the filter. For example, to get the various attributes, geographic or otherwise. Filters also let customers select a set of features for analysis, such as all the water features within a region or a new housing development (a custom polygon).

The following section highlights a potential way to apply this capability: Getting Available Features in Chapter 2 on page 12.
Using WFS Capabilities Through SOAP Requests

This section describes how to create SOAP requests that use WFS capabilities. For a description of how to use XML POST/GET requests through HTTP, refer to Using WFS Capabilities Through HTTP Request in Chapter 4 on page 23.

There are three steps to a Web Feature request process: describe service capabilities, describe feature details, and get features. There are three separate types of feature search capabilities, Attribute, Logical, and Spatial, when getting available features.

In this section:

- Describing Service Capabilities .......................8
- Describing Feature Details .............................10
- Getting Available Features ............................12
Describing Service Capabilities

When sending a Web Feature request, you may have access to multiple instances of the WFS service, and each of these may support different geographic areas and types of capabilities based on the available feature data. Before you send requests to a Web Feature Service instance, it would be useful to explore the capabilities and data available. The Web Feature Service is able to describe these capabilities in a get capabilities request. The service is able to indicate which feature types it can service and what operations are supported on each feature type. For example, how may feature types (FeatureType list) are defined, and what kind of operations and filters can be applied on the available feature types.

Inputs and Behaviors

To return a list of available feature types, the request document contains the following inputs:

<table>
<thead>
<tr>
<th>Input</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Name</td>
<td>yes</td>
<td>Fixed to WFS.</td>
</tr>
<tr>
<td>Version</td>
<td>yes</td>
<td>Fixed to 1.0.0. This is the version of the Web Feature Service standard implemented in this release. Regardless of the value specified, the service defaults to 1.0.0.</td>
</tr>
</tbody>
</table>

Output

The Web Feature Service response to a get capabilities request is composed of the following main sections:

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>The service section provides information about the service as defined in the configuration. This information includes the service alias name (WFS), title, abstract, keywords, resources, fees, and access constraints.</td>
</tr>
<tr>
<td>Capabilities</td>
<td>The capabilities section specifies the list of requests that the Web Feature Service can handle (GetCapabilities, DescribeFeatureType, and GetFeature).</td>
</tr>
<tr>
<td>Feature Type List</td>
<td>This section defines the list of feature types and operations on each feature type (Contents) that are available from the WFS. Additional information, such as the Spatial Reference System (SRS) for each feature type is also provided.</td>
</tr>
</tbody>
</table>
Java Code Sample

The following example returns the capabilities of the WFS Web Service for the instance with the service alias WFS.

```java
GetCapabilitiesRequest req = new GetCapabilitiesRequest("p", "4.0", "ID", "WFS");
ServiceMessage serviceMessage = new ServiceMessage();
serviceMessage.addRequest(req);
...
try{
    WebFeatureServiceLocator locator = new WebFeatureServiceLocator();
    WebFeature service = locator.getWebFeature(new URL(endpoint));
    ServiceMessage serviceResponse = new
    ServiceMessage(service.perform(serviceMessage.toXLS()));
    Response[] responseList = serviceResponse.getResponses();
    ...
}
```

XML Sample

The following example returns the capabilities of the WFS Web Service for the instance with the service alias WFS.

```xml
<RequestParameters xsi:type="ns3:GetCapabilitiesRequestType"
xmlns:ns3="http://www.mapinfo.com/wfs">
    <GetCapabilities service="WFS" version="4.0" xmlns:ns4="http://
    www.opengis.net/wfs"/>
</RequestParameters>
```
Describing Feature Details

After finding the available features from the Get Capabilities request, use the Describe Feature Type request to obtain detailed information for one or more of the features. The Describe Feature Type request defines how feature instances are expected to be encoded (described) on input and how feature instances will be generated on output.

MapInfo has a generic schema to describe all the feature types, therefore, rather than returning a static schema file, the WFS Web Service generates metadata information of the requested Content, such as attribute (column) names, alias, type, and the boundary of the Content.

Inputs and Behaviors

To obtain a description of a Content returned in the get capabilities response, the request document contains the service name, the version number, and the feature’s TypeName (Content).

<table>
<thead>
<tr>
<th>Input</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>yes</td>
<td>Fixed to WFS.</td>
</tr>
<tr>
<td>Version number</td>
<td>yes</td>
<td>Fixed to 1.0.0. This is the version of the Web Feature Service standard implemented in this release. Regardless of the value specified, the service defaults to 1.0.0.</td>
</tr>
<tr>
<td>Output Format</td>
<td>no</td>
<td>By default the output format is set to XMLSCHEMA. Currently, there are no other options defined.</td>
</tr>
<tr>
<td>Feature Type Name</td>
<td>yes</td>
<td>The name of the Content to be searched. For example, Washington DC.Financial.BANKS.DC Area Banks.</td>
</tr>
</tbody>
</table>

Output

The response contains one or more featureCollectionMetaData elements, which represent the metadata information of the requested Content. It contains the name, alias, and data type of each visible attribute, including geometry attribute in the Content. It also has the geometric boundary of the Content. Use the response as a starting point for performing a geosearch on the Content.

Java Code Samples

The following example will return a description for the Washington DC.DC Travel Agencies Content.

```java
DescribeFeatureTypeRequest req = new DescribeFeatureTypeRequest("p", "4.0", "ID", "Washington DC.DC Travel Agencies");
ServiceMessage serviceMessage = new ServiceMessage();
serviceMessage.addRequest(req);
...
try{
    WebFeatureServiceLocator locator = new WebFeatureServiceLocator();
    WebFeature service = locator.getWebFeature(new URL(endpoint));
```
Chapter 2: Using WFS Capabilities Through SOAP Requests

ServiceMessage serviceResponse = new 
ServiceMessage(service.perform(serviceMessage.toXLS()));

Response[] responseList = serviceResponse.getResponses();
 ...

.NET Code Sample

The following example will return a description for the Washington DC.DC Travel Agencies Content. You must specify the OutputFormat as XMLSCHEMA to return formatted output in the response.

private static Request newDescribeFeatureTypeRequest()
{
  DescribeFeatureTypeRequest pr = new DescribeFeatureTypeRequest();
  pr.RequestID = "R1";
  //set the output format
  pr.OutputFormat = "XMLSCHEMA";
  //set the type name
  string [] types = {"Washington DC.DC Travel Agencies"};
  pr.TypeName = types;

  return pr;
}

XML Sample

The following example returns a description for the "Washington DC.DC Travel Agencies" Content. Specify the outputFormat as XMLSCHEMA to return formatted output in the response.

<_RequestParameters xsi:type="ns3:DescribeFeatureTypeRequestType"
xmlns:ns3="http://www.mapinfo.com/wfs">
  <ns4:DescribeFeatureType outputFormat="XMLSCHEMA" service="WFS"
version="4.0" xmlns:ns4="http://www.opengis.net/wfs">
    <ns4:TypeName>Washington DC.DC Travel Agencies</ns4:TypeName>
  </ns4:DescribeFeatureType>
</_RequestParameters>

For multiple features (that are comma separated), Repeat <ns4:TypeName> .... </ns4:TypeName>
for each feature. For example:

<ns4:TypeName>Washington DC.DC Travel Agencies</ns4:TypeName>
<ns4:TypeName>Washington DC.DC Child Day Care Services</ns4:TypeName>
Getting Available Features

The Get Feature request returns details of specific features (Content) based on a query. Filters can be included in the request. The query is described by the filter and the list of attributes together.

If no filter is specified in a Get Feature request, then all features for the Content specified are returned. However, this could result in large quantities of unwanted results.

There are three types of filters supported by the Get Feature operation: Attribute, Spatial, and Logical.

Input and Behaviors

To query a specific feature (Content) based on the returned feature description in the DescribeFeatureType response, the request document contains a handle, OutputFormat, Service, and Version parameter. The actual query is the operation to be performed on the FeatureType. It contains the name of the Content the query is to be performed on and the following search parameters:

<table>
<thead>
<tr>
<th>Input</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Format</td>
<td>yes</td>
<td>The format to use to generate the result set. The default value is GML2. Currently, only GML2 is supported and returned in the result set.</td>
</tr>
<tr>
<td>Maximum Feature</td>
<td>no</td>
<td>Limits the number of features that a GetFeature request retrieves. Once the maxFeatures limit is reached, the result set is truncated.</td>
</tr>
<tr>
<td>Type Name</td>
<td>yes</td>
<td>Indicates the name of the feature type or class to be queried.</td>
</tr>
<tr>
<td>Property Name</td>
<td>no</td>
<td>One or more attribute names that represent the desired feature information to be returned. The attribute information can be found through the response document of the DescribeFeatureType operation, which contains the attribute (property) definitions of a given Content.</td>
</tr>
<tr>
<td>Filter</td>
<td>no</td>
<td>The constraints for the query. Both AttributeFilter and/or SpatialFilter constraints can be specified. If no Filter element is contained within the query, then the query is unconstrained, and all feature instances will be retrieved. For details about filters, refer to Supported Filters in Chapter 3 on page 19.</td>
</tr>
</tbody>
</table>

Output

The response document will have one or more FeatureCollection elements returned, which represent each of the Contents being queried. These include the Contents, metadata, description information (name, alias and data type of each visible attribute including geometry attributes) and the geometric boundary of the Content. For each Content being searched a list of features are returned (the attribute information), and that features spatial location based on the filters in the request.
Java Code Sample

The following example returns all of the Washington DC travel agencies that are within (SpatialFilter.TYPE_WITHIN) the specified geometry. The geometry is defined using an envelope. You must specify the attribute (column) names that you wish to be returned, such as ID, ZIP, Address, City, STATE, and the attribute name that the spatial operation is performed on (GeoLoc).

```java
ServiceMessage serviceMessage = new ServiceMessage();

String[] columns = {"ID", "Zip", "Address", "City", "STATE"};
double x1 = -77.040600;
double y1 = 38.899100;
double x2 = -77.044900;
double y2 = 38.900600;
Envelope geom = GeometryUtils.newEnvelope(x1, y1, x2, y2, "EPSG:4326");
SpatialFilter sf = new SpatialFilter("GeoLoc", geom, SpatialFilter.TYPE_WITHIN);

Query[] querys = new Query[1];
querys[0] = new Query("Washington DC.DC Travel Agencies", columns, sf);

GetFeatureRequest req = new GetFeatureRequest("p", "4.0", "ID", querys, -1);

serviceMessage.addRequest(req);

try{
    WebFeatureServiceLocator locator = new WebFeatureServiceLocator();
    WebFeature service = locator.getWebFeature(new URL(endpoint));
    ServiceMessage serviceResponse = new
    ServiceMessage(service.perform(serviceMessage.toXLS()));

    Response[] responseList = serviceResponse.getResponses();

}...
```

.NET Code Sample

The following example returns all of the Washington DC travel agencies that are within the 20433 ZIP Code™. You must specify the attribute (column) names that you wish to be returned, such as ID, ZIP, Address, City, STATE. The attribute filter is described by specifying the attribute name and value and then specifying the filter type, such as TYPE_EQUALSTO.

```csharp
private static Request newGetFeatureAttributeRequest()
{
    GetFeatureRequest pr = new GetFeatureRequest();
    pr.RequestID = "R1";

    Query[] querys = new Query[1];
    querys[0] = new Query();
    querys[0].TypeName = "Washington DC.DC Travel Agencies";
    String[] columns = {"ID", "Zip", "Address", "City", "STATE"};
```
//set the Logic Filter
LogicFilter lF = new LogicFilter();
lF.Type = LogicFilter.TYPE_AND;
//create the Filters for the Logic Filter
Filter[] aFs = new Filter[2];
//create an attribute filter
AttributeFilter aF = new AttributeFilter();
aF.Type = AttributeFilter.TYPE_EQUALSTO;
aF.AttributeValue = "Zip";
aF.LiteralValue = "20433";
aFs[0] = aF;
//create a Spatial Filter
SpatialFilter sF = new SpatialFilter();
sF.Type = SpatialFilter.TYPE_OVERLAPS;
sF.AttributeName = "Geoloc";
//set the envelope in the Spatial Filter
Point[] bounds = {GeometryUtils.newPoint(-77.040600, 38.899100), GeometryUtils.newPoint(-77.044900, 38.900600)};
sF.Envelope = GeometryUtils.newEnvelope(bounds, "EPSG:4269");
aFs[1] = sF;
lF.Filters = aFs;
//set the Logic Filter in the query
querys[0].Filter = lF;

pr.Querys = querys;

return pr;

XML Sample

The following example returns all of the Washington DC travel agencies that are within (DWithinType) a distance of 100 meters from the specified geometry. The geometry is specified using an envelope and the search area buffer is created using the Distance element. You must specify the attribute (column) names that you wish to be returned, such as ID, ZIP, Address, City, STATE, using the PropertyName element. The spatial DWithin filter is defined by using the PropertyName element (GeoLoc) to specify the attribute name the operation is to be performed upon. The attribute filter is then specified using the spatialOps element, such as DWithinType.

<_RequestParameters xsi:type="ns3:GetFeatureRequestType"
xmlns:ns3="http://www.mapinfo.com/wfs">
 <ns4:GetFeature maxFeatures="-1" outputFormat="GML3" service="WFS" version="4.0" xmlns:ns4="http://www.opengis.net/wfs">
   <ns4:Query typeName="Washington DC.DC Travel Agencies">
     <ns5:PropertyName xmlns:ns5="http://www.opengis.net/ogc">ID</ns5:PropertyName>
     <ns6:PropertyName xmlns:ns6="http://www.opengis.net/ogc">Zip</ns6:PropertyName>
     <ns7:PropertyName xmlns:ns7="http://www.opengis.net/ogc">Address</ns7:PropertyName>
</ns4:Query>
</ns4:GetFeature>
</_RequestParameters>
<ns8:PropertyName xmlns:ns8="http://www.opengis.net/ogc">City</ns8:PropertyName>
<ns9:PropertyName xmlns:ns9="http://www.opengis.net/ogc">STATE</ns9:PropertyName>
<ns10:Filter xmlns:ns10="http://www.opengis.net/ogc">
  <ns10:spatialOps xsi:type="ns10:DWithinType">
    <ns10:PropertyName>Geoloc</ns10:PropertyName>
    <ns11:_Geometry xsi:type="ns11:EnvelopeType" srsName="EPSG:4326" xmlns:ns11="http://www.opengis.net/gml">
      <ns11:pos dimension="2">-77.0449 38.8991</ns11:pos>
      <ns11:pos dimension="2">-77.0406 38.9006</ns11:pos>
    </ns11:_Geometry>
    <ns10:Distance uom="M" value="100"/>
  </ns10:spatialOps>
</ns10:Filter>
</ns4:Query>
</ns4:GetFeature>
</_RequestParameters>
This section describes WFS concepts and any differences in functionality to the standards. The information in this section is specific to using SOAP requests with the WFS. For information specific to using XML requests through HTTP, refer to Using WFS Capabilities Through HTTP Request in Chapter 4 on page 23.

Before you create an application that uses the WFS Web Service, it is helpful to understand basic concepts and how these concepts are implemented by the WFS.

In this section:

- Concepts .................................................17
- Functionality Limitations ..............................21
- Exceptions and Error Handling ......................22
Chapter 3: Web Feature Service Fundamentals

Concepts

The MapInfo implementation of the Web Feature Service searches for a feature type from a specified hierarchy (enterprise and domain) of catalogs and categories. A feature is a grouping of similar data information, such as banks, schools, and airports, which are organized into referenced Content from a database or TAB files, similar to a yellow pages directory.

These catalogs, categories, and Content are defined using the Content Manager. The Content Manager allows the administrator to define new Content and their attribute information to be returned to the user. It also defines additional filters for each Content to provide the user the capability to further refine their search criteria. For more information on Contents and the Content Manager, please refer to the Content Manager Guide.

This section provides information about:

- Specifying Contents in the Request
- Understanding Enterprises and Domains
- Supported Filters

Specifying Contents in the Request

The Content to search could be point-of-interest (POI) data or boundaries, such as trade areas or standard geographical boundaries (states, provinces, zip boundaries). There are two ways to define Content to search in the request: define the top level of the hierarchy (directory) and the Contents separate, or define the entire hierarchy including the Content to be searched together.

Specify the Hierarchy Separately

The hierarchy of a Content can be specified using the directory of the POI selection criteria and the value of the POI property. The directory type specifies the top level of the hierarchy, while the value specifies the Content to be searched.

For example, the top level directory called Washington DC, can have many Contents including DC Travel Agencies and DC Banks. To do a search for the travel agencies, specify the top level, Washington DC, as the directory type and the Content to be searched, DC Travel Agencies, as the POI Content to be search.

Java Sample

```java
rv.setDirectoryType("Washington DC");
rv.setPOISearchCriteria("other", "DC Travel Agencies");
```

.NET Sample

```csharp
request.DirectoryType="Washington DC";
POIProperty[] prp = new POIProperty[1];
prp[0] = new POIProperty();
prp[0].POIPropertyNameType = POIPropertyNameType.other;
prp[0].Value = "DC Travel Agencies";
request.SearchCriteria = prp;
```
XML Sample
<POISelectionCriteria xsi:type="ns2:POIPropertiesType"
directoryType="Washington DC">
  <POIProperty xsi:type="ns2:POIPropertyType" name="other" value="DC Travel Agencies"/>
</POISelectionCriteria>

Specify the Hierarchy Together

The complete hierarchy can be defined in a single line. The hierarchy, including directory and
Content, can be defined using a "." to separate the different levels. For example, Washington DC.DC
Banks or Washington DC.DC Travel Agencies.

Java Sample
POIProperty poiProperty = new POIProperty(POIProperty.NAME_OTHER,
  "Washington DC.DC Child Day Care Services");

.NET Sample
POIProperty [] prp = {new POIProperty(POIPropertyNameType.other,
  "Washington DC.DC Travel Agencies")};

XML Sample
<POISelectionCriteria xsi:type="ns2:POIPropertiesType">
  <POIProperty xsi:type="ns2:POIPropertyType" name="other" value="Washington DC.DC Travel Agencies"/>
</POISelectionCriteria>

Understanding Enterprises and Domains

The concept of Enterprises and Domains have been created in Envinsa Location Platform to
organize data and users for a solution. You may want to set up groups of users who have access to
certain data sources. You may not want the users in these groups to have access to the information
in other groups. To do this, the Envinsa Location Platform uses the concept of Enterprises. When a
user account is created, it is prefixed with a token that assigns that user rights to a particular
Enterprise. This allows users to access the information only within the Enterprise to which they have
been assigned. For example, a user account created for Fred, MapInfo.Fred, defines Fred as a user
in the MapInfo Enterprise.

A domain is a repository that hosts all the Content Manager information. It is the highest level of the
tree-like structure that logically groups an organization’s data. It is composed of resources stored on
a file system, as XML files, or on an LDAP server as small pieces of XML strings. Domains are all
given a unique name in the scope of an Enterprise and all domains are accessible using this name
when they are registered with the Content Manager.

To fully understand the concepts of Enterprises and Domains, which are the fundamentals of data
organization, please refer to the Content Manager Guide.
How Services use Enterprises and Domains

Services use both Enterprises and Domains when specifying the hierarchy of a Content. In order to access a specific Content, you need to know the Domain and Enterprise where the Content was created. Depending on the account privileges you have, you may or may not be able to access different Enterprises and Domains. To find out which Enterprise and Domains you have access to, or where your Content has been created, please see your Content Manager Administrator. To find more conceptual information on Content security and access refer to the Content Manager Guide.

The format for specifying a Content hierarchy with an Enterprise and a Domain is as follows:

\[<\text{Enterprise};\text{Domain}>//<\text{Content Path}>\]

For example, to specify a Content hierarchy for Washington DC.DC Travel Agencies accessible to the MapInfo Enterprise, Devel Domain:

MapInfo;Devel//Washington DC.DC Travel Agencies

The following rules apply when specifying Enterprises and Domains with services:

- Enterprise may have multiple Domains and one of which is default. An Enterprise which can be accessed by all users is called a public Enterprise. This Enterprise is defined using PUBLIC.
- A public Domain is a Domain which belongs to the public Enterprise. It is visible and accessible by all other Enterprises. The Envinsa default public Domain is the Content Manager sample domain, and can be specified as PUBLIC;SampleDomain. To access a Content, such as Washington DC.DC Travel Agencies, in the public Domain you should use the following convention:
  
  PUBLIC;SampleDomain//Washington DC.DC Travel Agencies

- An Envinsa domain is a PUBLIC domain by default, unless it's explicitly assigned to an Enterprise
- If a Domain is not specified in a request and the user's Enterprise is found, the default Domain of the user’s Enterprise is used. For example, MapInfo;SampleDomain//. If no default Domain is defined for the user’s Enterprise, the default public Domain is used. For example, PUBLIC;SampleDomain//.
- If a domain is not specified in a request and the user's Enterprise is NOT found, the default PUBLIC domain is used. For example, PUBLIC;SampleDomain//.

Supported Filters

Consumers of WFS would apply filters to selectively retrieve features from a WFS server based on certain conditions being met in the Get Feature Request. For for details about this capability, refer to Getting Available Features in Chapter 2 on page 12. These conditions are expressed in the form of filters, which are the XML equivalent of boolean expressions. A boolean expression either returns a "true" or a "false" for every feature in a Content. For example, consider the expression "Age > 30". Using this filter, only features with the age attribute greater than 30 are returned.

Filters can be a simple expression as in the previous example or they can be complex, formed by combining simple expressions using logical OR, AND, and NOT operators. For example:

Age > 30 AND Name like 'John'

Filters can also be spatial. A spatial filter can be used to search features based on their geographic attribute. For example, find all features that are within 10 miles of a given point.
Three types of filters are supported by the WFS: attribute, spatial, and logical.

**Note:** For SQL users, filters are the direct equivalent of the "where clause" in a SQL statement.

**Attribute Filter**

An AttributeFilter defines a non-spatial searching constraint or query using a comparison. The following comparison operators are supported. Examples illustrating how they may be used are provided.

- **EqualTo** – This filter can be used to find all of the features that are in a particular country subdivision.
- **NotEqualTo** – This filter can be used to find all of the features that are not in a particular country subdivision.
- **LessThan** – Define a search to find all features where a population is less than a given value. For example, Population < 100,000.
- **GreaterThan** – Define a search to find all features where a population is greater than a given value. For example, Population > 200,000.
- **LessThanOrEqualTo** – Define a search to find all features where a population is less than or equal to a given value. For example, Population <= 100,000.
- **GreaterThanOrEqualTo** – Define a search to find all features where a population is greater than or equal to a given value. For example, Population >= 200,000.
- **Like** – This filter can be used to define a search based on a pattern, such as find all features where the country name starts with A. For example, CountryName like A%.
- **Between** – This filter can be used to select features where the population is within a range of two values. For example, Population between 100,000 AND 200,000.
- **IsNull** – This filter can be used to select features that do not have any value set for an attribute, such as Date of Birth is null (or not set). For example, DATE_OF_BIRTH is null.

**Spatial Filter**

A Spatial Filter defines a spatial searching constraint or query, using a spatial operator. The following spatial operators are supported:

- **Overlaps** – The features to be returned may lie within, or partially within, the specified geometry. Boundaries that touch are considered overlapping.
- **DWithin** – The features to be returned lie within the specified distance from the geometry.
- **Within** – The features to be returned lie within the specified geometry. Boundaries can touch, but the inner object cannot have any points outside the containing boundary.

**Logical Filter**

A Logical Filter is the combination of an Attribute Filter and a Spatial Filter using the logical operators And (AndType), Or (OrType), and Not (NotType).
Functionality Limitations

Envinsa Services extend service functionality beyond what is outlined by OpenLS and OpenGIS standards. As a result, there are differences in functionality from what is described in the standards. This section describes any extensions or limitations.

If using the OGC Web Feature Service, refer to Using WFS Capabilities Through HTTP Request in Chapter 4 on page 23.

Standards Limitations

The following OGC standard operations are not supported in the current Envinsa Web Feature Service implementation. These limitations apply to this release.

Transaction

A Web Feature Service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.

LockFeature

A Web Feature Service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serialize transactions are supported.

Feature Restrictions

The OpenGIS Web Feature Service Implementation Specification has required and optional implementations. The required section, called WFS Basic, is essentially a read-only service. The optional section is called WFS Transaction and allows for the addition, modification, and deletion of features in a feature type. The WFS implementation in Envinsa only implements WFS Basic at this time.

The following features are not supported in the current WebFeature service implementation. These restrictions apply to this release.

- Only GetCapabilities, DescriptionFeatureType, and GetFeature operations are currently supported.
- Handle in the GetFeature operation is not currently supported.
- Version in the outputFormat attribute is not currently supported in all the three Web Feature Service operations.
- No boundary geometry is returned in the DescribeFeatureType operation.
Exceptions and Error Handling

In the event that a WFS encounters an error while processing a request or receives an unrecognized request, it generates an XML document indicating that an error has occurred. The format of the XML error response is specified by, and must validate against, the exception response schema defined in section A.2 of OGC/WFS Specification (http://www.opengeospatial.org/specs/?page=specs).
Using WFS Capabilities Through HTTP Request

The WFS follows the OGC Web Feature Service standard, which defines how a client can retrieve geospatial data encoded in GML by submitting XML POST/GET requests through HTTP. Products that can access data from an OGC compliant WFS server will work with the Envinsa WFS, products such as MapInfo Professional and MapXtreme or other vendor software.

For a descriptions using SOAP requests, refer to Using WFS Capabilities Through SOAP Requests in Chapter 2 on page 7.

In this section:

- Supported Capabilities ...............24
- FeatureType and PropertyName Conventions ...............29
- Compliance with GML2 Geometries ...............30
- Turning off Security Authentication ...............30
**Supported Capabilities**

Envinsa’s OGC WFS allows a client to retrieve geospatial data encoded in GML. A client issues a Get Capabilities request to determine the list of feature types that a WFS offers, it issues a DescribeFeatureType request in order to obtain a description of one or more feature types, it then uses the description of the feature types to formulate a GetFeature operation.

The specific operations defined in the OGC WFS Specification are: GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, LockFeature, and Transaction. Envinsa includes a Basic OGC WFS: it is read only, and implements GetCapabilities, DescribeFeatureType, and GetFeature operations.

**Standards**

For details on the standards that the OGC WFS is compliant with, refer to the following documents at the [http://www.opengeospatial.org](http://www.opengeospatial.org) site:

- The OGC WFS Specification version 1.0.0
- Filer Encoding Specification version 1.0.0
- Level 0 Profile of GML3 from WFS version 0.0.10

**GetCapabilities**

The get capabilities operation is used to request a capabilities document from the OGC WFS. It issues an XML request to the WFS and gets a response that describes the capabilities of the WFS.

The supported spatial operators are: Overlaps, DWithin, and Within.

**POST Request**

The request schema is defined by the following XML Schema fragment:

```xml
<xsd:element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
<xsd:complexType name="GetCapabilitiesType">
  <xsd:attribute name="version" type="xsd:string" use="optional"/>
  <xsd:attribute name="service" type="xsd:string" use="required" fixed="WFS"/>
</xsd:complexType>
```

**GET Request**

The GET request contains the following:

<table>
<thead>
<tr>
<th>Input</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUEST=GetCapabilities</td>
<td>yes</td>
<td>Set to GetCapabilities for a get capabilities operation.</td>
</tr>
</tbody>
</table>
Chapter 4: Using WFS Capabilities Through HTTP Request

For example:

http://WFS_URL?REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.0.0

Where WFS_URL is the URL to the location of the WFS.

Java Code Sample

A GetCapabilities request provides information about what we can get from the WFS server. For example, if the response returns a feature type called World, then the DescribeFeatureType method can be invoked to get the feature type description of World.

The following sample obtains the capabilities from the server.

```java
public void getCapabilities() {
    String SERVICE_URL = "http://www.mapinfo.com/webfeature";
    String request = "<?xml version="1.0" ?> " + 
    "<GetCapabilities version="1.0.0" service="WFS"/>";
    try {
        URL url = new URL(SERVICE_URL);
        URLConnection conn = url.openConnection();
        HttpURLConnection httpConn = (HttpURLConnection) conn;
        httpConn.setRequestProperty("Content-Type","text/xml; charset=utf-8");
        httpConn.setRequestMethod( "POST" );
        httpConn.setDoOutput(true);

        OutputStream out = httpConn.getOutputStream();
        out.write( request.getBytes() );
        out.close();

        InputStreamReader ireader = new InputStreamReader(httpConn.getInputStream());
        BufferedReader in = new BufferedReader(ireader);
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
```

The OGC WFS supports HTTP Basic Authentication.
DescribeFeatureType

Use the DescribeFeatureType request to obtain a schema description for features serviced by the OGC WFS. The schema description describes how feature instances are expected to be encoded (described) on input and how feature instances are generated on output.

**Note:** A describe feature type SOAP request made to the WFS Web Service returns an instance to a schema to describe the feature type.

**POST Request**

The following sample calls the DescribeFeatureType method to get the description of feature type called World. The response is an XML Schema. The request schema is defined by the following XML Schema fragment:

```xml
<xsd:element name="DescribeFeatureType"
    type="wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
    <xsd:sequence>
        <xsd:element name="TypeName" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="xsd:string" use="required" fixed="1.0.0"/>
    <xsd:attribute name="service" type="xsd:string" use="required" fixed="WFS"/>
    <xsd:attribute name="outputFormat" type="xsd:string" use="optional" default="XMLSCHEMA"/>
</xsd:complexType>
```

**GET Request**

The GET request contains the following:

<table>
<thead>
<tr>
<th>Input</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE=WFS</td>
<td>yes</td>
<td>Fixed to WFS.</td>
</tr>
<tr>
<td>VERSION=1.0.0</td>
<td>yes</td>
<td>Fixed to 1.0.0. This is the version of the Web Feature Service standard implemented in this release. Regardless of the value specified, the service defaults to 1.0.0.</td>
</tr>
<tr>
<td>REQUEST=DescribeFeatureType</td>
<td>yes</td>
<td>Set to DescribeFeatureType for a describe feature type operation.</td>
</tr>
<tr>
<td>OUTPUTFORMAT=XMLSCHEMA</td>
<td>no</td>
<td>Default value is XMLSCHEMA.</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>no</td>
<td>A comma separated list of feature types. If no value is specified that is interpreted as all feature types</td>
</tr>
</tbody>
</table>

For example:
Chapter 4: Using WFS Capabilities Through HTTP Request

http://WFS_URL?SERVICE=WFS&VERSION=1.0.0&REQUEST=DescribeFeatureType&TYPENAME=World

Where WFS_URL is the URL to the location of the WFS.

Java Code Sample

The following example uses the DescribeFeatureType method to get the feature type description of World.

```java
public void getDescribeFeatureType() {
    String SERVICE_URL = "http://www.mapinfo.com/webfeature";
    String request =  "<?xml version="1.0" ?> " +
    "<DescribeFeatureType version="1.0.0" service="WFS" " +
    "xmlns:ogc="http://www.opengis.net/ogc" xmlns="http://
www.opengis.net/wfs" " +
    "xmlns:miwfs="http://www.mapinfo.com/wfs/" " +
    "xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" " +
    "xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.0.0/WFS-
basic.xsd"> " +
    "<TypeName>miwfs:World</TypeName> </DescribeFeatureType>";
    try {
        URL url = new URL(SERVICE_URL);
        URLConnection conn = url.openConnection();
        HttpURLConnection httpConn = (HttpURLConnection) conn;
        httpConn.setRequestProperty("Content-Type","text/xml;charset=utf-8");
        httpConn.setRequestMethod( "POST" );
        httpConn.setDoOutput(true);
        OutputStream out = httpConn.getOutputStream();
        out.write( request.getBytes());
        out.close();

        InputStreamReader ireader = new InputStreamReader(httpConn.getInputStream());
        BufferedReader in = new BufferedReader(ireader);
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }catch (Exception ex ) {
        System.out.println(ex.getMessage());
    }
}
```

GetFeature

The GetFeature operation retrieves features and specified information about the feature from the WFS server. An XML document that contains the result set is returned to the client.
The returned FeatureCollection follows the OGC Compliance Profile. This is different from the FeatureCollection returned from a SOAP request to the WFS (Getting Available Features on page 12), which is defined by MapInfo.

Limitations:

- The GET method is not supported.

**POST Request**

The XML encoding of a GetFeature request is defined by the following XML Schema fragment:

```xml
<xsd:element name="GetFeature" type="wfs:GetFeatureType"/>
<xsd:complexType name="GetFeatureType">
  <xsd:sequence>
    <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:string" use="required" fixed="1.0.0"/>
  <xsd:attribute name="service" type="xsd:string" use="required" fixed="WFS"/>
  <xsd:attribute name="handle" type="xsd:string" use="optional"/>
  <xsd:attribute name="outputFormat" type="xsd:string" use="optional" default="GML2"/>
  <xsd:attribute name="maxFeatures" type="xsd:positiveInteger" use="optional"/>
</xsd:complexType>
<xsd:element name="Query" type="wfs:QueryType"/>
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle" type="xsd:string" use="optional"/>
  <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
  <xsd:attribute name="featureVersion" type="xsd:string" use="optional"/>
</xsd:complexType>
```

**Java Code Sample**

The following example calls GetFeature to retrieve data from the feature type called World.

```java
public void getFeature() {
    String SERVICE_URL = "http://www.mapinfo.com/webfeature";
```

```xml
<xld:element name="GetFeature" type="wfs:GetFeatureType"/>
```

```xml
<xsd:complexType name="GetFeatureType">
  <xsd:sequence>
    <xsd:element ref="wfs:Query" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:string" use="required" fixed="1.0.0"/>
  <xsd:attribute name="service" type="xsd:string" use="required" fixed="WFS"/>
  <xsd:attribute name="handle" type="xsd:string" use="optional"/>
  <xsd:attribute name="outputFormat" type="xsd:string" use="optional" default="GML2"/>
  <xsd:attribute name="maxFeatures" type="xsd:positiveInteger" use="optional"/>
</xsd:complexType>
```

```xml
<xsd:element name="Query" type="wfs:QueryType"/>
```

```xml
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:element ref="ogc:PropertyName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle" type="xsd:string" use="optional"/>
  <xsd:attribute name="typeName" type="xsd:QName" use="required"/>
  <xsd:attribute name="featureVersion" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Java Code Sample

The following example calls GetFeature to retrieve data from the feature type called World.

```java
public void getFeature() {
    String SERVICE_URL = "http://www.mapinfo.com/webfeature";
```
"xsi:schemaLocation="http://www.opengis.net/wfs../wfs/1.0.0/WFS-basic.xsd">" + "<Query typeName="miwfs:World"/>
</GetFeature>»;
try {
    URL url = new URL(SERVICE_URL);
    URLConnection conn = url.openConnection();
    HttpURLConnection httpConn = (HttpURLConnection) conn;
    httpConn.setRequestProperty("Content-Type","text/xml;charset=utf-8");
    httpConn.setRequestMethod("POST");
    httpConn.setDoOutput(true);
    OutputStream out = httpConn.getOutputStream();
    out.write( request.getBytes());
    out.close();
    InputStreamReader ireader = new InputStreamReader(httpConn.getInputStream());
    BufferedReader in = new BufferedReader(ireader);
    String inputLine;
    while ((inputLine = in.readLine()) != null)
        System.out.println(inputLine);
    in.close();
} catch (Exception ex) {
    System.out.println(ex.getMessage());
}

FeatureType and PropertyName Conventions

Feature type names and property names must be expressed in GML or XML as elements or attributes, so they must be valid element and attribute names. This means that they must be a qualified name (QName) as defined in the XML specifications of W3C. The QName definition is:

NCName::=(Letter|'_') (NCNameChar)*
NCNameChar::=Letter|Digit|'.'|'-'|'_'|CombiningChar|Extender
QName::=(Prefix':')?LocalPart
Prefix::=NCName
LocalPart::=NCName

The previous version of the Envinsa WFS used a semi-colon (;) and two back slashes (//) to separate Enterprise, Domain, and Content names. Also, spaces were permissible in the Content name and Property name. These characters and spaces are not permissible QName characters. Therefore, the following substitutions are made for the Envinsa OGC WFS.

• semi-colon (;) – replaced by “_ES_”
• two back slashes (//) – replaced by “_DS_”
• spaces – replaced by “_SP_”

These convention rules are defined in a property file called minameconvention.prop in the WebFeature\resources\admin directory where WFS is installed. This file defines the character convention rules by value pairs. The default values are:

;=_ES_
Compliance with GML2 Geometries

According to the OGC WFS specification, feature types must be described using GML and XML Schema. OGC releases a profile called the "Level 0 Profile of GML3 for WFS" to define how a GML application schema can be tested for compliance. The Envinsa OGC WFS follows this profile.

The Envinsa OGC WFS service supports GML2 output formats only. The profile refers to GML3 geometry definitions. The following table matches these geometry definitions to GML2.

<table>
<thead>
<tr>
<th>GML3</th>
<th>GML2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointPropertyType</td>
<td>PointType</td>
</tr>
<tr>
<td>CurvePropertyType</td>
<td>LineStringType</td>
</tr>
<tr>
<td>MultiCurvePropertyType</td>
<td>MultiLineStringType</td>
</tr>
<tr>
<td>LinearRingPropertyType</td>
<td>LinearRingType</td>
</tr>
<tr>
<td>RingPropertyType</td>
<td>N/A</td>
</tr>
<tr>
<td>SurfacePropertyType</td>
<td>PolygonType</td>
</tr>
<tr>
<td>MultiSurfacePropertyType</td>
<td>MultiPolygonType</td>
</tr>
</tbody>
</table>

Turning off Security Authentication

The OGC Web Feature Service standard does not define a security model, but the Envinsa WFS does provide mechanisms of specifying user and password using HTTP Basic authentication.

To be OGC compliant, authentication has been turned off for the WFS.

The OGC WFS and WFS WebService share the same security model. By default, security is turned on for the OGC WFS. Turn security off to send OGC WFS standard requests. Otherwise, username and password have to be sent through Basic Authentication. (Basic Authentication is an HTTP standard way to send username and password.)
The configuration of the Web Feature Service contains information for setting default values and defining custom layers for returning points of interest.

Use the Enterprise Manager to configure this service. There are two categories of configuration settings: connection to the Content Manager and preferences for the get capabilities operation. The configuration for the connection to the Content Manager should not be changed.

In this section:

- Configuration Preferences ........................................32
## Configuration Preferences

The GetCapabilities section contains the service parameters for the GetCapabilities operation of the Web Feature Service. The values of these parameters are presented in the response to the GetCapabilities request.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>An abstract name for the Web Feature Service.</td>
<td>No Abstract</td>
</tr>
<tr>
<td>AccessConstraints</td>
<td>Defines constraints for accessing the Web Feature Service.</td>
<td>No Constraints</td>
</tr>
<tr>
<td>Fees</td>
<td>Defines keywords associated with the Web Feature Service.</td>
<td>No Keyword</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the Web Feature Service (e.g., WFS).</td>
<td>WFS</td>
</tr>
<tr>
<td>serviceURL</td>
<td>This preference sets the return string for the end point of WFS service. The form should be [http://[host ip]:[port]/WebFeature/servlet/WebFeature]. If this preference is not set, proxy URL will be returned instead.</td>
<td>Null</td>
</tr>
<tr>
<td>Title</td>
<td>An expanded Name for the Web Feature Service (e.g., Envinsa Platform Web Feature Service).</td>
<td>MapInfo Web Feature Service 4.1</td>
</tr>
</tbody>
</table>