Using a Web Feature Service

Using a Web Feature Service

by Chris Holmes and Mike Pumphrey of OpenGeo (http://opengeo.org/)

October, 2008

This document is an introduction/tutorial to the basic profile of the OGC Web Feature Service. It is not meant as a full-scale, exhaustive reference. For those who want the full details, they are encouraged to download the specification at http://www.opengeospatial.org/standards/wfs (http://www.opengeospatial.org/standards/wfs) .

1. Introduction

The Web Feature Service is officially defined as:

> [T]he OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services."
>
> -- from http://www.opengeospatial.org/standards/wfs

Since we're not assuming very much here in this tutorial, let's also briefly define GML:

> The Geography Markup Language (GML) is the XML grammar defined ... to express geographical features.
>
> --Wikipedia

That's enough official definitions for now. The Web Feature Service (hereafter WFS) refers to the sending and receiving of geospatial data through the medium of the World Wide Web (specifically HTTP). An important distinction must be made between WFS and Web Map Service (WMS), which refers to the sending and receiving of geographic information after it has been rendered as a digital image. One can think of WFS as the "source code" to the maps that one would view via WMS. The current version of WFS is 1.1. WFS version 1.0 is still used in places, and we will note where there are differences. However, the syntax will often remain the same.

For more information on WFS, the GeoServer project (http://geoserver.org) has a good page on WFS (http://geoserver.org/display/GEOSDOC/2.+WFS) .

2. GET versus POST requests

There are two main methods of requesting data from a WFS server: HTTP GET and HTTP POST. An HTTP GET request (hereafter GET) is a request for data encoded into a URL. It takes the form:

```
http://www.example.com/wfsserver?
        name1=value1&
        name2=value2&
        ...
```

The above request sends the site at `http://www.example.com/wfsserver` the key/value pairs of `"name1=value1", "name2=value2", etc`. These names and values refer to server-specific settings (details will be discussed below). A fair amount a data can be passed through a GET request, as there is no official limit to the length of a URL. But sending too much data through a GET request can become unwieldy, not to mention rather hard to read. The pro side of a GET request is that it is very compact, and can be sent via a web browser. POST requests, which involve custom clients and XML encoding, can be more powerful, but are not the easiest place to start the discussion. With this in mind, we will only be concerned with GET requests in this document. For those familiar with POST requests, every GET request here has an equivalent POST request (but the opposite is not true).

3. Operations

There are six operations a WFS can possibly implement:

- GetCapabilities
- DescribeFeatureType
- GetFeature
- GetGMLObject *(new in v1.1)*
- Transaction
- LockFeature

As we are describing Basic WFS in this document, we'll limit our discussion to the first three operations, **GetCapabilities**, **DescribeFeatureType**, and **GetFeature**. **GetGMLObject** is a very advanced feature, and the final two operations, **Transaction** and **LockFeature**, are for WFSes that let one edit their holdings. In this tutorial we stick to read-only operations.

3.1. GetCapabilities

It's necessary for clients who wish to connect to a WFS server to know what sort of functionality ("capabilities") is being offered by that server. The **GetCapabilities** operation is a request for this information.

A typical GetCapabilities request (using GET) would look like this:

```
http://www.example.com/wfsserver?
        service=wfs&
        version=1.1.0&
        request=GetCapabilities
```

(The actual request would be all on one line, with no line breaks, but our convention here is to show each key/value pair on its own line, for clarity.) Here there are three parameters being passed to our wfs server, "`service=wfs`", "`version=1.1.0`", and "`request=GetCapabilities`." At a bare minimum, it is required that a WFS request have these three parameters (service, version, and request). Occasionally, a WFS server will relax these requirements, but officially they are mandatory, so they should always be included. The *service* key tells the WFS server that a WFS request is forthcoming. The *version* key refers to which version of WFS is being requested. Note that there are only two version numbers officially supported: "1.0.0" and "1.1.0". Supplying a value like "1" or "1.1" will likely return an error. The *request* key is where the actual GetCapabilities operation is specified.

The Capabilities document that is returned is a long and complex chunk of XML, but very important, and so it is worth taking a closer look. (The 1.0.0 Capabilities document is very different from the 1.1.0 document discussed here, so beware.) There are five main components we will be discussing (other components are beyond the scope of this document.):

- **ServiceIdentification** - This section contains basic "header" information such as the Name and ServiceType. The ServiceType mentions which version(s) of WFS are supported.
- **ServiceProvider** - This section provides contact information about the company behind the WFS server, including telephone, website, and email.
- **OperationsMetadata** - This section describes the operations that the WFS server recognizes and the parameters for each operation. A WFS server can be set up not to respond to all aforementioned operations.
- **FeatureTypeList** - This section lists the available FeatureTypes. They are listed in the form "namespace:featuretype". Also, the default projection of the FeatureType is listed here, along with the resultant bounding box for the data in that projection.
- **Filter_Capabilities** - This section lists filters available in which to request the data. SpatialOperators (Equals, Touches), ComparisonOperators (LessThan, GreaterThan), and other functions are all listed here. These filters are not defined in the Capabilities document, but most of them (like the ones mentioned here) are self-evident.

Let's see this in action by using a real-world Capabilities document. The one we will use is from the WFS server here: http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi (http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi) . As the document is long, we will only examine certain areas.

(/pub/ogcnetwork/files/images/wfs_tut_01-getcap.png)

ServiceIdentification:

<ows:ServiceIdentification>

<ows:Title>USGS Framework Layer WFS</ows:Title>

```
<ows:Abstract>A WFS serving USGS framework layers.</ows:Abstract>

<ows:Keywords>

<ows:Keyword>WFS HTTP data feature spatial USGS framework hydrography governmental units roads gazetteer</ows:Keyword>

</ows:Keywords>

<ows:ServiceType>WFS</ows:ServiceType>

<ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>

<ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>

<ows:Fees>NONE</ows:Fees>

<ows:AccessConstraints>NONE</ows:AccessConstraints>

</ows:ServiceIdentification>
```

Here we see that the WFS server is named "USGS Framework Layer WFS" and that it responds with WFS versions 1.1.0 and 1.0.0.

ServiceProvider:

```
<ows:ServiceProvider>

<ows:ProviderName>GeoLeaders LLC</ows:ProviderName>

<ows:ProviderSite xlink:href="http://www.geoleaders.com"/>

<ows:ServiceContact>

<ows:IndividualName>Panagiotis (Peter) A. Vretanos (CubeWerx Inc.)</ows:IndividualName>

<ows:PositionName>Senior Developer</ows:PositionName>

<ows:ContactInfo>

<ows:Phone>

<ows:Voice>(703)491-9543</ows:Voice>

<ows:Facsimile>(703)491-0873</ows:Facsimile>

</ows:Phone>

<ows:Address>

<ows:DeliveryPoint>12052 Willowood Drive</ows:DeliveryPoint>

<ows:City>Lake Ridge</ows:City>

<ows:AdministrativeArea>Virginia</ows:AdministrativeArea>

<ows:PostalCode>22192</ows:PostalCode>

<ows:Country>USA</ows:Country>

<ows:ElectronicMailAddress>pvretano@cubewerx.com</ows:ElectronicMailAddress>

</ows:Address>
```

```
<ows:OnlineResource xlink:href="http://geoleaders.cubewerx.com"/>

</ows:ContactInfo>

</ows:ServiceContact>

</ows:ServiceProvider>
```

Here we see the name of the company is GeoLeaders LLC, with a contact person, Panagiotis (Peter) A. Vretanos, and other address information.

OperationsMetadata:

```
<ows:OperationsMetadata>

<ows:Operation name="GetCapabilities">

<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework&amp;"/>

<ows:Post xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework"/>

</ows:HTTP>

</ows:DCP>

<ows:Parameter name="AcceptVersions">

<ows:Value>1.1.0</ows:Value>

<ows:Value>1.0.0</ows:Value>

</ows:Parameter>

<ows:Parameter name="AcceptFormats">

<ows:Value>text/xml</ows:Value>

</ows:Parameter>

<ows:Parameter name="Sections">

<ows:Value>ServiceIdentification</ows:Value>

<ows:Value>ServiceProvider</ows:Value>

<ows:Value>OperationsMetadata</ows:Value>

<ows:Value>FeatureTypeList</ows:Value>

<ows:Value>SupportsGMLObjectTypeList</ows:Value>

<ows:Value>Filter_Capabilities</ows:Value>

</ows:Parameter>

</ows:Operation>

<ows:Operation name="DescribeFeatureType">
```

```
<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework&amp;"/>

<ows:Post xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework"/>

</ows:HTTP>

</ows:DCP>

<ows:Parameter name="TypeName">

<ows:Value>gubs:GovernmentalUnitCE</ows:Value>

...

</ows:Parameter>

<ows:Parameter name="outputFormat">

<ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>

<ows:Value>text/xml; subtype=gml/3.1.1/profiles/gmlsf/0</ows:Value>

<ows:Value>text/xml; subtype=gml/3.1.1/profiles/gmlsf/1</ows:Value>

<ows:Value>text/xml; subtype=gml/2.1.2</ows:Value>

<ows:Value>application/gml+xml; subtype=gml/3.1.1</ows:Value>

...

</ows:Parameter>

</ows:Operation>

<ows:Operation name="GetFeature">

<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework&amp;"/>

<ows:Post xlink:href="http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?DATASTORE=Framework"/>

</ows:HTTP>

</ows:DCP>

<ows:Parameter name="outputFormat">

<ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>

<ows:Value>text/xml; subtype=gml/3.1.1/profiles/gmlsf/0</ows:Value>

...
```

```
</ows:Parameter>

<ows:Parameter name="resultType">

<ows:Value>results</ows:Value>

<ows:Value>hits</ows:Value>

</ows:Parameter>

</ows:Operation>

<ows:Parameter name="service">

<ows:Value>WFS</ows:Value>

</ows:Parameter>

<ows:Parameter name="version">

<ows:Value>1.1.0</ows:Value>

<ows:Value>1.0.0</ows:Value>

</ows:Parameter>

<ows:Constraint name="srsName">

<ows:Value>EPSG:4326</ows:Value>

<ows:Value>CRS:84</ows:Value>

...

</ows:Constraint>

<ows:Constraint name="SupportsSOAP">

<ows:Value>TRUE</ows:Value>

</ows:Constraint>

</ows:OperationsMetadata>
```

(The ellipses indicate chunks of XML that are omitted because they would not add any clarity to our discussion. Feel free to type in the URL above to see the full list!) Here we see the different types of output formats supported by the WFS server. Among much else, there is also a long list (mostly omitted) and the various projections (here seen as ows:Constraint name="srsName") supported by the server.

FeatureTypeList:

```
<FeatureTypeList>

<Operations>

<Operation>Query</Operation>

</Operations>

<FeatureType>

<Name>gubs:GovernmentalUnitCE</Name>
```

```
<Title>Governmental Unit (County or Equivalent)</Title>

<DefaultSRS>EPSG:4269</DefaultSRS>

<ows:WGS84BoundingBox>

<ows:LowerCorner>-179.14221197 18.9108417</ows:LowerCorner>

<ows:UpperCorner>-66.94983061 71.35256069</ows:UpperCorner>

</ows:WGS84BoundingBox>

</FeatureType>

<FeatureType>

<Name>gubs:GovernmentalUnitMCD</Name>

<Title>Governmental Unit (Minor Civil Divisions)</Title>

<DefaultSRS>EPSG:4269</DefaultSRS>

<ows:WGS84BoundingBox>

<ows:LowerCorner>-179.14221197 18.9108417</ows:LowerCorner>

<ows:UpperCorner>-66.94982987 71.38990002</ows:UpperCorner>

</ows:WGS84BoundingBox>

</FeatureType>

<FeatureType>

<Name>gubs:GovernmentalUnitST</Name>

<Title>Governmental Unit (State or Territory)</Title>

<DefaultSRS>EPSG:4269</DefaultSRS>

<ows:WGS84BoundingBox>

<ows:LowerCorner>-179.14221197 18.9108417</ows:LowerCorner>

<ows:UpperCorner>-66.94983061 71.35256069</ows:UpperCorner>

</ows:WGS84BoundingBox>

</FeatureType>

...

</FeatureTypeList>
```

Here is the list of FeatureTypes (most omitted here). The first FeatureType listed is "gubs:GovernmentalUnitCE", with a title of "Governmental Unit (County or Equivalent)." Its default projection is EPSG:4269, and the bounding box is from approx. (-179.14 18.91) to (-66.95 71.35). We'll talk more about bounding boxes later on.

Filter_Capabilities:

```
<Filter_Capabilities xmlns="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/ogc
http://schemas.opengis.net/filter/1.1.0/filterCapabilities.xsd">
```

```
<ogc:Spatial_Capabilities>

<ogc:GeometryOperands>

<ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>

<ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>

<ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>

<ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>

...

</ogc:GeometryOperands>

<ogc:SpatialOperators>

<ogc:SpatialOperator name="BBOX"/>

<ogc:SpatialOperator name="Equals"/>

<ogc:SpatialOperator name="Disjoint"/>

<ogc:SpatialOperator name="Intersects"/>

<ogc:SpatialOperator name="Touches"/>

<ogc:SpatialOperator name="Crosses"/>

<ogc:SpatialOperator name="Within"/>

<ogc:SpatialOperator name="Contains"/>

<ogc:SpatialOperator name="Overlaps"/>

</ogc:SpatialOperators>

</ogc:Spatial_Capabilities>

<ogc:Scalar_Capabilities>

<ogc:LogicalOperators/>

<ogc:ComparisonOperators>

<ComparisonOperator>LessThan</ComparisonOperator>

<ComparisonOperator>GreaterThan</ComparisonOperator>

<ComparisonOperator>LessThanEqualTo</ComparisonOperator>

<ComparisonOperator>GreaterThanEqualTo</ComparisonOperator>

<ComparisonOperator>EqualTo</ComparisonOperator>

<ComparisonOperator>NotEqualTo</ComparisonOperator>

<ComparisonOperator>Like</ComparisonOperator>
```

```
<ComparisonOperator>Between</ComparisonOperator>

<ComparisonOperator>NullCheck</ComparisonOperator>

</ogc:ComparisonOperators>

<ogc:ArithmeticOperators>

<ogc:SimpleArithmetic/>

<ogc:Functions>

<ogc:FunctionNames>

<ogc:FunctionName nArgs="1">MIN</ogc:FunctionName>

<ogc:FunctionName nArgs="1">MAX</ogc:FunctionName>

</ogc:FunctionNames>

</ogc:Functions>

</ogc:ArithmeticOperators>

</ogc:Scalar_Capabilities>

<ogc:Id_Capabilities>

<ogc:FID/>

</ogc:Id_Capabilities>

</Filter_Capabilities>
```

Finally, we see the available filters. We have geometries (point, linestring, polygon, envelope), spatial operators (BBOX, Equals, Disjoint,...), comparison operators (LessThan, GreaterThan, LessThanEqualTo,...), and two functions known as MIN and MAX.

3.2. DescribeFeatureType

We may wish to know more about the individual FeatureTypes before requesting data from them. This is the purpose of the operation entitled **DescribeFeatureType**. Specifically, DescribeFeatureType will request a list of features and attributes for the given FeatureType, or list the FeatureTypes available.

Let's say we want a list of FeatureTypes. The appropriate GET request would be:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=DescribeFeatureType
```

Note again the three required fields (service, version, and request). This will return the list of FeatureTypes, sorted by namespace.

If we wanted information about a specific FeatureType, the GET request would be:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=DescribeFeatureType&
                typeName=namespace:featuretype
```

The only difference between the two requests is the addition of "typeName=namespace:featuretype" where featuretype is the name of the FeatureType and namespace is the name of the namespace that FeatureType is contained in.

Let's look at an example of a DescribeFeatureType response. Once again, we will use the same WFS server: http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi (http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi) .

(This particular WFS server requires an additional parameter of "FRAMEWORK=datastore&" when making requests. If you don't believe it, go back up and look at the GetCapabilities document, under the OperationsMetadata area. You'll see it there.)

We will be asking for information on the "gubs:GovernmentalUnitCE" FeatureType. This is a collection of all the counties in the US, or equivalents in the case of states that don't have counties, like Louisiana and Alaska. The command we are using for this request is:

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=DescribeFeatureType&
                typeName=gubs:GovernmentalUnitCE
```

(/pub/ogcnetwork/files/images/wfs_tut_02-dft.png)

And the response (again, with header information omitted):

```
<xs:element name="GovernmentalUnitCE" type="gubs:GovernmentalUnitCEType" substitutionGroup="gml:_Feature"/>

<xs:complexType name="GovernmentalUnitCEType">

<xs:annotation>

<xs:documentation>

<cwmeta:Metadata>

<cwmeta:Title>Governmental Unit (County or Equivalent)</cwmeta:Title>

</cwmeta:Metadata>

</xs:documentation>

</xs:annotation>

<xs:complexContent>

<xs:extension base="gml:AbstractFeatureType">

<xs:sequence>

<xs:element name="unitId" type="fw:IdentifierPropertyType"/>

<xs:element name="typeAbbreviation" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="instanceName" type="xs:string"/>

<xs:element name="instanceAlternateName" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

```
<xs:element name="officialDescription" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="instanceCode" type="xs:string"/>

<xs:element name="codingSystemReference" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="effectiveDate" type="xs:date" minOccurs="0" maxOccurs="2"/>

<xs:element name="geometry" type="gml:SurfacePropertyType">

<xs:annotation>

<xs:documentation>

<cwmeta:Metadata>

<cwmeta:Title>3</cwmeta:Title>

<cwmeta:Abstract>3</cwmeta:Abstract>

<cwmeta:MetadataURL>3</cwmeta:MetadataURL>

</cwmeta:Metadata>

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="extended" type="fw:ExtendedAttributePropertyType" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="governmentalUnitType">

<xs:complexType>

<xs:simpleContent>

<xs:restriction base="gml:CodeType">

<xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>

</xs:restriction>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="typeDefinition" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="boundedBy" type="gml:ReferenceType" minOccurs="0" maxOccurs="unbounded">

<xs:annotation>

<xs:appinfo source="urn:x-gml:targetElement"/>

</xs:annotation>

</xs:element>
```

```
</xs:sequence>

</xs:extension>

</xs:complexContent>

</xs:complexType>
```

It's easy to get overwhelmed when reading XML, but there actually isn't that much information here. We learn that this FeatureType has thirteen elements or attributes (count 'em: unitId, typeAbbreviations, instanceName, instanceAlternateName, officialDescription, instanceCode, codingSystemReference, effectiveDate, geometry, extended, governmentalUnitType, typeDefinition, and boundedBy). We also see some attributes of these features (for example, the element called "instanceName" is of the type "string").

If we wanted to get information about all FeatureTypes on the WFS server, we would omit the "typeName=gubs:GovernmentalUnitCE" pair. This would return a large volume of data, and wouldn't really advance the discussion, but it is certainly possible.

3.3. GetFeature

All of this has been an introduction to the actual main course, as it were, of actually asking for and receiving the map data. This is the "source code" spoken about in the Introduction. The operation for this is called **GetFeature**. More so than the other operations, it is complex and powerful. Obviously, not all of its abilities will be discussed here.

The simplest way to run a GetFeature command is without any arguments.

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype
```

This syntax should be familiar from previous examples. The only difference is the "request=GetFeature."

You probably don't want to execute this command! Don't worry, you won't break anything (you're not making changes, after all), but a Feature is a complex beast, and we haven't specified which Feature we wanted, so our WFS server would return *all* of them. This can take forever to load, and has been known to crash browsers, and so is generally to be avoided. Instead, we want to specify some way of limiting the output. One way to do this is if we happen to know the name of the Feature. In this case, the GET request would be:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                featureID=feature
```

where we have added the addition parameter of "featureID=feature." Replace the word "feature" with the Feature you wish to retrieve.

If we don't know the name of the Feature, or if we don't care about a specific one, but wish to limit the amount of Features returned, there is the "maxFeatures" parameter.

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                maxFeatures=N
```

where N = the number of Features to return.

A question that may arise at this point is how the WFS server knows which N Features to return. The bad news is that it depends on the internal structure of the data, which may not be arranged in a very helpful way. The good news is that we have the ability to sort the features based on a certain attribute. (This is new as of 1.1.) We use the following syntax:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                maxFeatures=N&
                sortBy=property
```

where "`sortBy=property`" determines the sort. Replace criteria with the attribute you wish to sort by. The default is to sort ascending. Some WFS servers require sort order to be specified, even if ascending. If so, append a "+A" to the request. Want to sort descending instead? No problem. Just add a "+D" to the request, like so:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                maxFeatures=N&
                sortBy=property+D
```

It is not necessary to use sortBy with maxFeatures, but they can often complement each other.

All of that said, sometimes we don't want to narrow down the search by Feature, but instead by some attribute. In this case, instead of wanting all the attributes of one or more features, we want one or more attributes of the (one or more) Features. To do this, we use the "propertyName" tag in the form "propertyName=property." We can specify a single property, or multiple properties separated by commas. If we want a single property from all features, we use the following:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                propertyName=property
```

If we wanted a single property from just one feature:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                featureID=feature&
                propertyName=property
```

Or more than one property from a feature:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
```

```
                        typeName=namespace:featuretype&
                        featureID=feature&
                        propertyName=property1,property2
```

As usual, we can mix and match at will, depending on the data that we wish to retrieve.

All of these permutations so far have centered on parameters of a non-spatial nature, but what about spatial relationships? We've said nothing about the geometry of the Features – mainly because they are quite cumbersome to work with. But querying for Features based on location is vital to any discussion of geospatial data, so let's not put it off any longer. While there are very limited tools available in a GET request for spatial queries (much more is available in POST requests, using Filters) one of the most important can be used. This is known as the "bounding box" or BBOX. The BBOX allows us to ask for only such Features that are contained (or partially contained) inside a box of the coordinates we specify. The form of the bbox query is "bbox=a1,b1,a2,b2" where a, b, c, and d refer to coordinates.

This brings up a bit of a thorny issue. Notice that the syntax wasn't "bbox=x1,y1,x2,y2" or "bbox=y1,x1,y2,x1". The reason the coordinate-free "a,b" syntax was used above is because the order depends on the coordinate system used. To specify the coordinate system, append "srsName=CRS" to the WFS request, where CRS is the coordinate reference system. As for which corners of the bounding box to specify (bottom left / top right or bottom right / top left), that appears to not matter, as long as the bottom is first. So the full request for returning Features based on bounding box would look like this:

```
http://www.example.com/wfsserver?
            service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=namespace:featuretype&
                bbox=a1,b1,a2,b2
```

Now let's look at some examples using our now-familiar WFS server (http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi). Before, we looked at the "gubs:GovernmentalUnitCE" FeatureType. If we were out of our minds, or wanted to crash our browser, we would type in the following:

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE
```

This would return an enormous amount of data, as we would receive the coordinates for every single polygon on every single Feature. Let's only return the first feature:

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
                maxFeatures=1
```

(/pub/ogcnetwork/files/images/wfs_tut_03-gf1.png)

And the response (this one will be a bit long, but still quite incomplete):

```
<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.0">

<gubs:unitId>

<fw:Identifier>

<fw:identifier>e64b1841-57d3-4f6a-af04-bc011024ad98</fw:identifier>

<fw:idAuthority>US Census Bureau</fw:idAuthority>

</fw:Identifier>

</gubs:unitId>

<gubs:instanceName>Kauai</gubs:instanceName>

<gubs:instanceCode>007</gubs:instanceCode>

<gubs:effectiveDate>2005-11-07</gubs:effectiveDate>

<gubs:geometry>

<gml:Polygon srsName="EPSG:4269">

<gml:exterior>

<gml:LinearRing>

<gml:posList srsDimension="3">-159.4020556 22.2326597 0 -159.4019946 22.2326127 0 -159.40194973 22.23252363 0 -159.40176473 22.2323727 0 -159.4016346 22.23238263 0 -159.40141673 22.23233777 0 -159.40121573
22.2322597 0 -159.4011106 22.23217177 0 -159.40098566 22.2319397 0 -159.40093666 22.23156963 0 -159.4008376 22.23136163 0 -159.4009916 22.23121357 0 -159.4009896 22.2310217 0 -159.4011856 22.2308087 0
-159.4010966 22.23066463 0 -159.40112866 22.2304787 0 -159.40127366 22.2302917 0 -159.40133166 22.22947677 0 -159.40107773 22.22931077 0 -159.40081766 22.22937857 0 -159.40055066

...

</gml:posList>

</gml:LinearRing>

</gml:exterior>

<gml:interior>

<gml:LinearRing>

<gml:posList srsDimension="3">-160.08855366 22.0021807 0 -160.0881446 22.00190557 0 -160.08755566 22.0018867 0 -160.08728073 22.00142657 0 -160.08707866 22.0011877 0 -160.08703073 22.00102457 0 -160.08681373
22.0006637 0 -160.08662873 22.00007663 0 -160.08666173 21.99976963 0 -160.0867606 21.99961763 0 -160.08668273 21.99925057 0 -160.08693466 21.99851877 0 -160.08696466 21.99800063 0 -160.08660466 21.99745057 0
-160.08539673 21.99646657 0 -160.08375373 21.99622357 0 -160.08311773 21.9961457 0 -160.08195373 21.99583057 0 -160.08109566 21.9960237 0 -160.0799656 21.99638063 0

...

</gml:posList>

</gml:LinearRing>

</gml:interior>

</gml:Polygon>
```

```
</gubs:geometry>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>OBJECTID</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>1344</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_DATASETID</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>0381af74-802c-4aba-b2ec-fd75c5d56d03</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_DATADESCR</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>Counties derived from US Bureau of Census polygons</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_ORIGINATOR</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>
```

```
<fw:value>US Bureau of Census</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>DATA_SECURITY</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Integer</fw:type>

<fw:value>5</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>DISTRIBUTION_POLICY</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>E4</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>POPULATION2000</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>60165</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>AREASQKM</fw:name>
```

```
<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>0.00000000</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>FCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>61200</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STATE_FIPSCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>15</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STATE_NAME</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>Hawaii</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>
```

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STCO_FIPSCCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>15007</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:governmentalUnitType/>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

Yes, XML is never pretty, but at least it's comprehensive. Let's take a look at the information we've gleaned from getting this feature. The information at the beginning states that this is data from the US Census Bureau, and that the county we are dealing with is Kauai, one of the Hawaiian islands. (The fact that Kauai came up first depends on, as mentioned previously, the internal structure of the data, of which we know nothing.) The feature, which is a polygon, is in the projection EPSG:4269 (details about projections are way beyond the scope of this document, but it will be only mentioned that EPSG:4269 can be thought of as longitude/latitude coordinates). What follows is the actual geometry of the Feature, a long list of coordinate points for both an "exterior" and an "interior." One can think of the "exterior" geometry as the outer shape of the figure, while the "interior" are the holes that aren't a part of the figure, much like how the hole isn't exactly part of a doughnut, even though it is inside of the doughnut's "exterior." After the geometry is listed in all its gory details, what follows is a list of attributes (POPULATION2000=60165, STATE_NAME=Hawaii, etc.)

If we wanted to sort our Features by alphabetical order, it looks like `sortBy=gubs:instanceName+A` would seem like the right sort option.

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
                maxFeatures=1&
                sortBy=gubs:instanceName+A
```

But before we see the output, let's say that we don't care about the geometry (which the above request would provide), but instead were just looking for the first County name in the big list. We'd want to only output the "gubs:instanceName", by appending the "propertyname=gubs:instanceName":

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
                maxFeatures=1&
                sortBy=gubs:instanceName+A&
                propertyName=gubs:instanceName
```

(/pub/ogcnetwork/files/images/wfs_tut_05-gf3.png)

And the response (excluding headers, as usual):

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.1394">

<gubs:instanceName>AAAAA</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

Of course, it's the county named "AAAAA". I've never heard of it myself, so let's find out what state it's in:

```
        http://www.example.com/wfsserver?
                    DATASTORE=Framework&
                        service=wfs&
                        version=1.1.0&
                        request=GetFeature&
                        typeName=gubs:GovernmentalUnitCE&
                        maxFeatures=1&
                        sortBy=gubs:instanceName+A&
                        propertyName=STATE_NAME
```

(/pub/ogcnetwork/files/images/wfs_tut_06-gf4.png)

And the response:

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.1394">

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STATE_NAME</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>Alaska</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

Ahh, yes, Alaska, we should have known.

Want a list of all the county names in the US in alphabetical order? (no, you don't, but this is just an example):

```
        http://www.example.com/wfsserver?
```

```
DATASTORE=Framework&
    service=wfs&
    version=1.1.0&
    request=GetFeature&
    typeName=gubs:GovernmentalUnitCE&
    sortBy=gubs:instanceName+A&
    propertyName=gubs:instanceName
```

(/pub/ogcnetwork/files/images/wfs_tut_07-gf5.png)

We won't show the full response here, as it's rather long (there are over 3000 counties in the US). But looking through the list, I notice that there are multiple counties named Delaware. I'm originally from Delaware County, Pennsylvania, and so was keen to find out which feature corresponded to my home county.

```
...

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.879">

<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.2564">

<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.1338">

<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.2683">

<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.3007">
```

```
<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.2976">

<gubs:instanceName>Delaware</gubs:instanceName>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

...
```

We've reached a bit of a limit with what we can query in a GET request. A POST request would be able to use Filters to query for Features whose county name is Delaware, and whose state name is Pennsylvania. We don't have that kind of granularity, so we would need to query each one of these features to see the state name. For the first one, the request would be:

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
                featureID=CWFID.GOVUNIT_CE.0.879&
                propertyName=STATE_NAME
```

Can you figure out the featureID of the Delaware County in Pennsylvania?

Now, let's try some examples with spatial relationships. We know just about the longitude and latitude of where Delaware County, PA is located (it's around 40 N, -75 W, or, for this projection, (-75,40)) so I know that if I requests features within a small bounding box in that area, the WFS server should return only that county (If even a small piece of another county is contained within our box, it will be returned.). Let's flex our BBOX muscles.

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
                bbox=-75.5,39.85,-75.49,39.86
```

(/pub/ogcnetwork/files/images/wfs_tut_09-gf7.png)

And the response:

```
<gml:featureMember>

<gubs:GovernmentalUnitCE gml:id="CWFID.GOVUNIT_CE.0.2976">

<gubs:unitId>

<fw:Identifier>
```

```
<fw:identifier>bd049cc5-7e13-4914-a1fd-550e49f3c8ea</fw:identifier>

<fw:idAuthority>US Census Bureau</fw:idAuthority>

</fw:Identifier>

</gubs:unitId>

<gubs:instanceName>Delaware</gubs:instanceName>

<gubs:instanceCode>045</gubs:instanceCode>

<gubs:effectiveDate>2006-01-09</gubs:effectiveDate>

<gubs:geometry>

<gml:Polygon srsName="EPSG:4269">

<gml:exterior>

<gml:LinearRing>

<gml:posList>-75.27647499 39.976955 -75.27783399 39.976256 -75.27912599 39.975591 -75.27942799 39.975436 -75.28030299 39.974991 -75.28022999 39.974888 -75.28019199 39.974835 -75.28

...

</gml:posList>

</gml:LinearRing>

</gml:exterior>

</gml:Polygon>

</gubs:geometry>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>OBJECTID</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>495</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_DATASETID</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>
```

```
<fw:value>0381af74-802c-4aba-b2ec-fd75c5d56d03</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_DATADESCR</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>Counties derived from US Bureau of Census polygons</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>SOURCE_ORIGINATOR</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>US Bureau of Census</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>DATA_SECURITY</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Integer</fw:type>

<fw:value>5</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>
```

```
<fw:name>DISTRIBUTION_POLICY</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>E4</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>POPULATION2000</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>552643</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>AREASQKM</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>0.00000000</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>FCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">Number</fw:type>

<fw:value>61200</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>
```

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STATE_FIPSCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>42</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STATE_NAME</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>Pennsylvania</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:extended>

<fw:ExtendedAttribute>

<fw:authority>US Census Bureau</fw:authority>

<fw:name>STCO_FIPSCCODE</fw:name>

<fw:type codeSpace="urn:fgdc:hydro:dataTypes:">CharacterString</fw:type>

<fw:value>42045</fw:value>

</fw:ExtendedAttribute>

</gubs:extended>

<gubs:governmentalUnitType/>

</gubs:GovernmentalUnitCE>

</gml:featureMember>

Or, if we wanted, we could look for only counties in the graticule of 38 N, 122 W:

```
http://www.example.com/wfsserver?
            DATASTORE=Framework&
                service=wfs&
                version=1.1.0&
                request=GetFeature&
                typeName=gubs:GovernmentalUnitCE&
```

```
bbox=-122.99,38,-122,38.99
```

(/pub/ogcnetwork/files/images/wfs_tut_10-gf8.png)

We would receive data on the counties of Mendocino, Contra Costa, Marin, Napa, Sonoma, Solano, Lake, Colusa, and Yolo, all in California.

4. WFS Clients

It should be obvious that just because one *can* retrieve WFS through a web browser doesn't mean that this is something that one *should* do. In reality, one would retrieve the data through a WFS client. There are a few to choose from.

uDig (udig.refractions.net (http://udig.refractions.net)) – Desktop-based and open source, it is based of Geotools and written in Java. Very fully featured for both WFS and WMS, although it doesn't currently support GML3 (the newest version of GML)

QGIS (*www.qgis.org (http://www.qgis.org)*) – Also desktop-based and open source, it is based on Qt, and provides similar functionality to uDig.

ArcGIS Desktop (http://www.esri.com/products/#1 (http://www.esri.com/products/#1)) – This is not free or open but is OGC standard compliant and is very popular in the proprietary world.

A larger list of clients can be found here: http://geoserver.org/display/GEOSDOC/Clients (http://geoserver.org/display/GEOSDOC/Clients) .

5. Final Words

There is much that can be asked for, and even more received, when dealing with map data through the protocol of WFS. We have only scratched the surface of what is possible, but hopefully this is enough to give a flavor of how WFS works. For further information, please read the full WFS specification (the link is given at the beginning of this document). A future document will also include more Advanced WFS topics. Thanks!

Comments

the maps that one would view

Submitted by amelia (not verified) on Wed, 2013-04-10 07:40.

the maps that one would view via WMS. The current version of WFS is
1.1. WFS version 1.0 is still used in places, and we will note where
there are differences.
