FME® Server
OGC Web Services

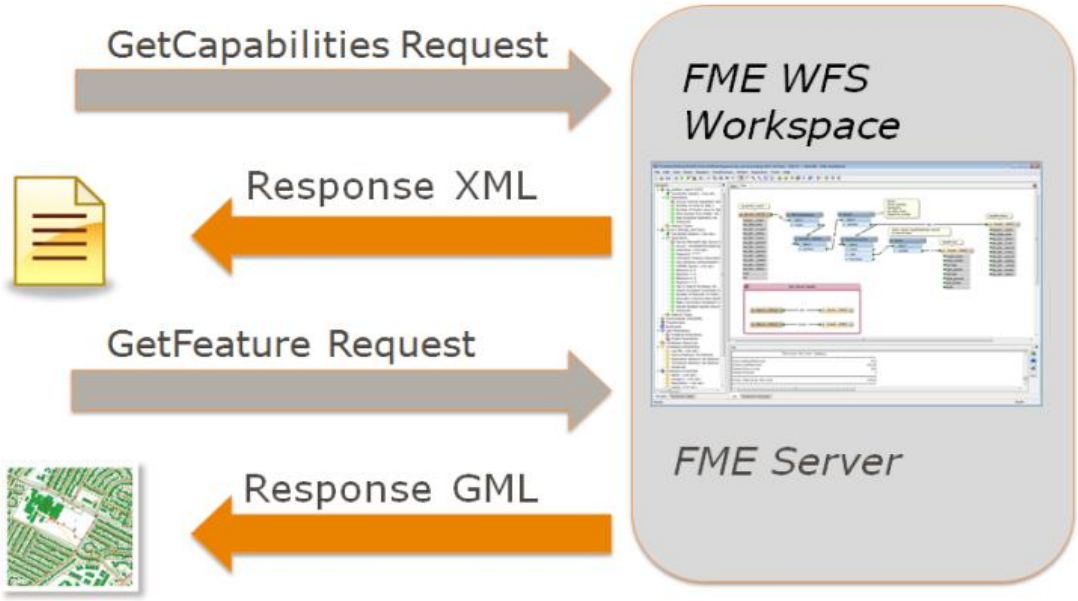
OGC Services – INSPIRE Focus

OGC Services – INSPIRE Focus

Introduction

With the new power of FME's GML generation, we need ways to share this data. Our new approach to supporting OGC web services accomplishes this. FME web services are now hosted by publishing a service broker workspace to the data streaming service on FMEServer. Instead of a workspace that just handles data conversion, the service broker workspace handles the web message traffic – accepting requests and generating responses according to the chosen service standard.

For WFS, this means the workspace accepts GET (a URL) or POST (XML encoding) requests (GetCapabilities, DescribeFeatureType and GetFeature) and generates the appropriate responses as XML or GML data streams. This approach also supports XML filters - something the old built in WFS could not do. This 'service broker' workspace just needs to be published to FME Server's data streaming service in order to function as a web service. Also, the workspace is configured to support FME Data Inspector as a WFS client.



FME workspace as web service broker for WFS

Any web service, such as WCS, WPS, WMTS or SOS could be supported by this FME service broker workspace approach. All that is required is to understand the web service protocol client / server requirements and configure accordingly. Additionally, any complex XML / GML that needs to be transmitted via WFS can be supported, whether INSPIRE*, AIXM, metadata/CSW etc.

New service types would require some setup / configuration to implement the message handling compatible with that service standard, but this approach gives the user control to configure, customize, update and extend their web services how they want. This can be applied to other types of REST based web services beyond OGC or INSPIRE applications.

The purpose of this training is to show how FME workspaces can be configured to act as a service broker for INSPIRE web services, via data streaming on FME Server.

* INSPIRE, or Infrastructure for Spatial Information in Europe, is a European Union directive that aims to create a spatial data infrastructure of EU data, which would be used for policy making across boundaries, especially environmental. The directive also aims to improve public access to spatial information. Open standards such as the Open Geospatial Consortium Web Feature Service (OGC WFS) are used to facilitate the exchange of INSPIRE data. For more information please visit the INSPIRE website.

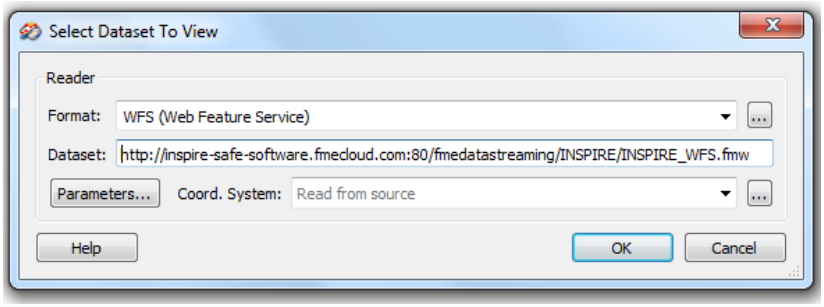
Exercise 1: INSPIRE WFS for FMEServer

PART A: Reading Cadastral Parcels from Safe's demo WFS

In this exercise, FME Data Inspector will be used to read cadastral parcel data from Safe's demo WFS. A XML filter will be used to demonstrate the ability to query the WFS.

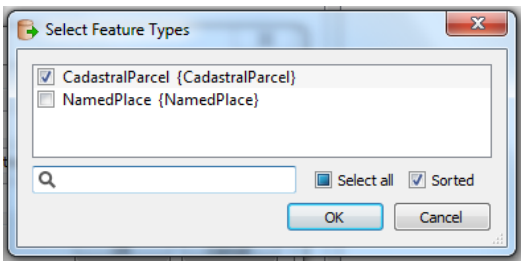
1. Start Data Inspector. Under Tools - FME Options set background maps to Stamen Maps and under parameters, select 'toner', to provide a suitable backdrop.

2. Open Dataset.
   Format: WFS (Web Feature Service)
   Dataset: http://inspire-safe-software.fmecloud.com:80/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw

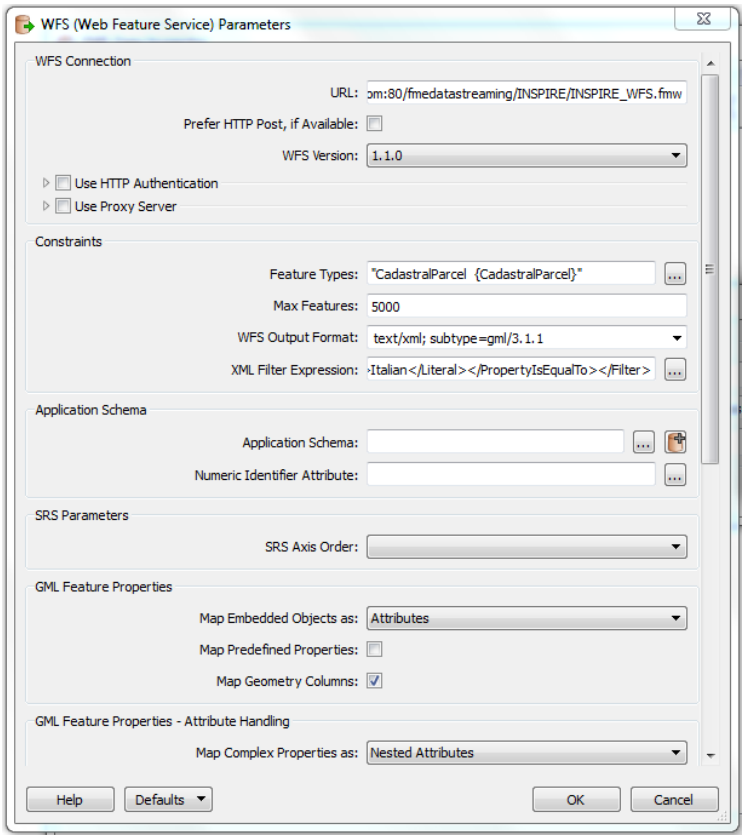


Select Dataset to View

3. Click on Parameters…
   Click the […] button to the right of Feature Types, and choose Cadastral Parcels. Note that this fires a 'GetCapabilities' request to INSPIRE_WFS.fmw, which hosts the WFS service on FME Server. Also, note that this server only allows the selection of one feature type at a time. If you see nothing here then you have a problem with the URL or your internet connection.



Select Feature Types



WFS Parameters – Cadastral parcels

Safe's INSPIRE demo requires no login, so ensure Use HTTP Authentication is unchecked. Make sure the XML filter is empty and search envelope is unchecked, and press OK.

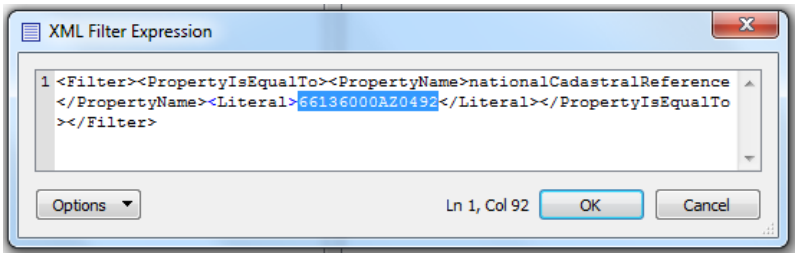
After about 10-15 seconds the parcels data from Perpignan in southwest France should appear. If your data appears off of Africa, then you set the SRS axis order incorrectly - should be blank. If nothing happens after more than a minute - hit stop and check all your settings, including search envelope and try again. If your settings yield no data, the client can sit and wait a long time before timeout.

4. Find a large parcel and click on it to find its local ID.

Cadastral Parcel read with the WFS Reader

5. One parcel has a local ID of 136000AZ0063. Copy this ID, add dataset again, and select a new WFS view with the same settings as before. Then under parameters, go into the XML Filters Expression under Constraints and paste the ID into an XML filter expression of the form:

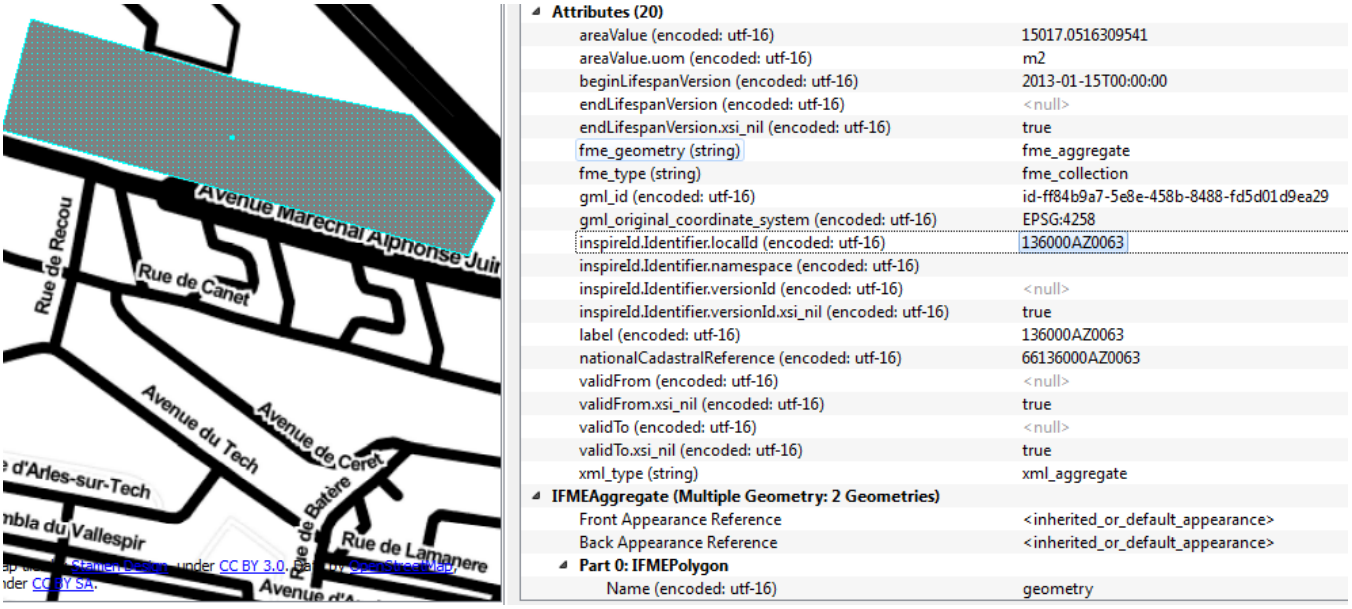
```
<Filter><PropertyIsEqualTo><PropertyName>inspireId.Identifier.localId</
PropertyName><Literal>136000AZ0063</Literal></PropertyIsEqualTo>
</Filter>
```



XML filter expression for cadastral parcel localId

Note that spelling must be exact. This will be parsed by the WFS workspace into _whereQuery = 'inspireId.Identifier.localId='136000AZ0063''.

6. Click OK. The result will be a view containing only the parcel with local ID 136000AZ0063.

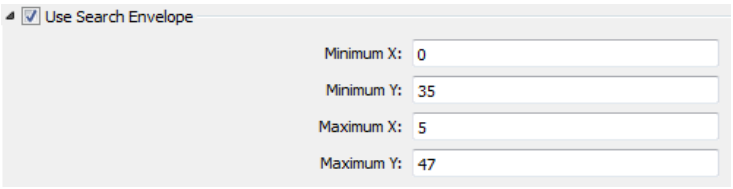


Cadastral parcel returned from using an XML filter expression

PART B: Reading Named Places from Safe's demo WFS:

Now, we will use Data Inspector to read named place data from the same WFS, utilizing spatial query and XML filter capabilities. The same format will be followed to read NamedPlace feature type.

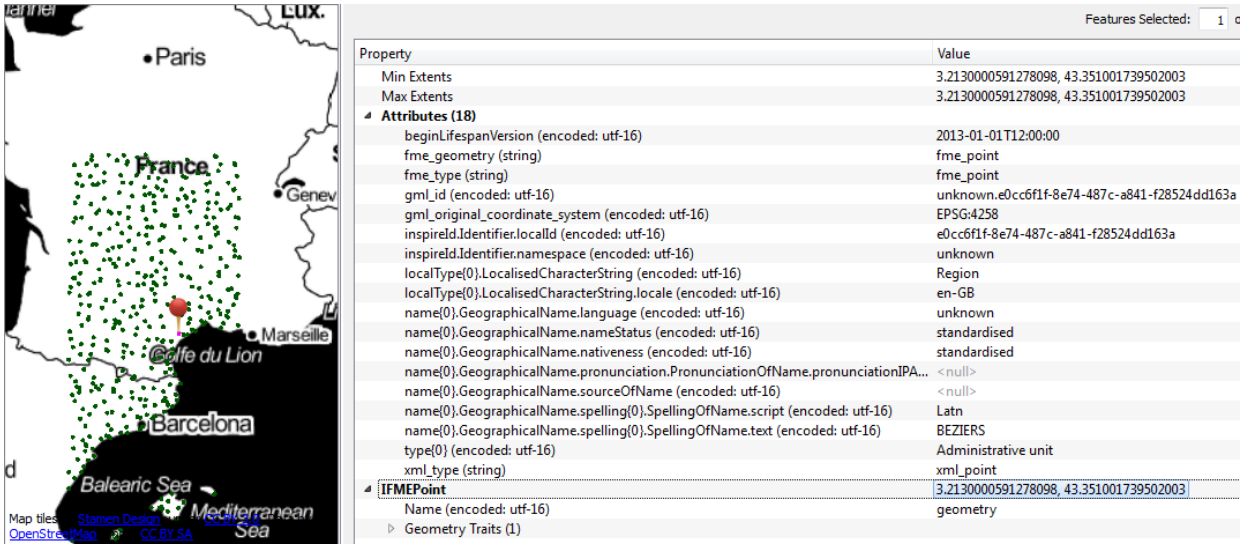
1. In Data Inspector, choose add a dataset with the WFS. Open the WFS parameters, click the […] button to the right of Feature Types, and choose Named Places only (uncheck Cadastral Parcels).



Search envelope for Named Place

2. This time we will set a spatial query for Named Places. Check Use Search Envelope and set Min X=0, Min Y=35, Max X=5, Max Y = 47. Make sure the XML filter expression is empty. Press OK twice. You should now see named place data for West France, which overlaps parcel data.

Note: If you forget to change the spatial filter, you will get an XML parser error since there is no data returned. To correct this, change the spatial filter to: Min X=0, Min Y=35, Max X=5, Max Y=47.



Named Place output using a spatial query

3. Finally, we select all the named places for Italy. Select add a new WFS dataset for Named Place, and set up this XML filter, in the reader parameters:

   ```
   <Filter><PropertyIsEqualTo><PropertyName>GeographicalName_language</Pro
   pertyName><Literal>Italian</Literal></PropertyIsEqualTo></Filter>
   ```

   Remember to uncheck the Use Search Envelope box or you will get an XML parser error.

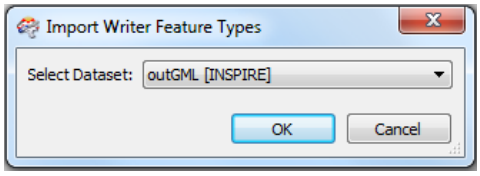
PART C: INSPIRE_WFS.fmw

For this exercise, we will add the CadastralParcel feature type to the INSPIRE_WFS-Start.fmw workspace. To do this, you basically just need to replicate the logic that is already there for NamedPlace. Note that the CadastralParcel custom transformer is already there to parse your filters and make the correct query against the database.

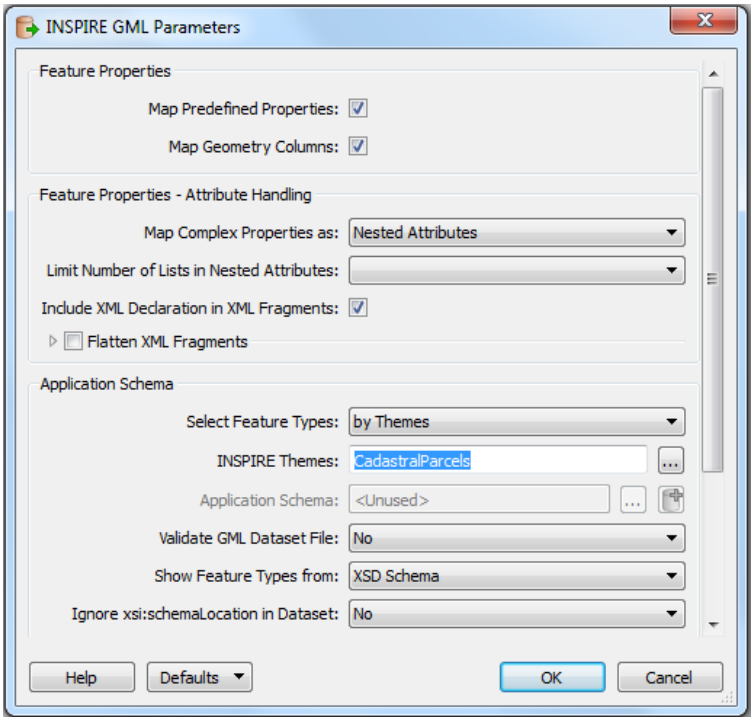
Note that this workspace requires FME 2014 or later to support the INSPIRE GML Writer. Presently, this is a basic example with only 2 feature types, spatial extents queries and one XML filter operation, though these can be extended easily by following the approach laid out in this workspace.

1. Open C:\FMEData2014\Resources\INSPIRE\OGCservices-INSPIREfocus\WFS\INSPIRE_WFS-Start.fmw. Add a CadastralParcel destination dataset. Go to Writer - Import Feature Types, select outGML [INSPIRE] as dataset, click OK. In the Import Writer Feature Types dialog, open the INSPIRE parameters.

   Under application schema, select feature types by themes, choose the CadastralParcel INSPIRE theme. Click OK three times. Choose the CadastralParcel feature type.

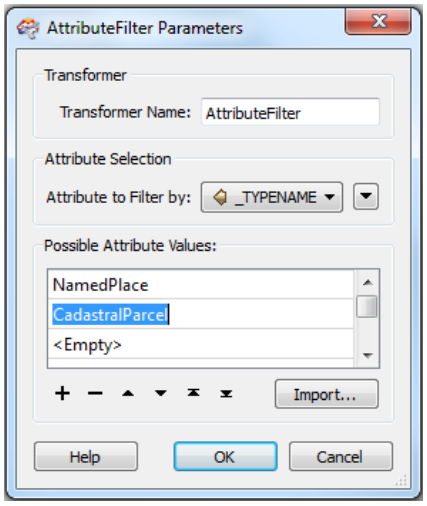


Import Writer Feature Types



Cadastral Parcel INSPIRE GML Writer Parameters

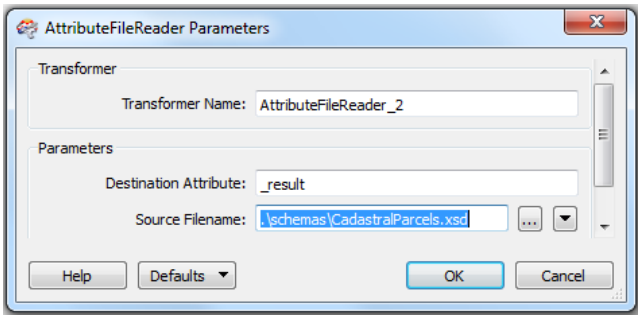
2. Add the CadastralParcel feature type name to all **AttributeFilters** (AttributeFilter in the Process GET Request bookmark, AttributeFilter_2 in the DescribeFeatureType bookmark, AttributeFilter_3 in the GetCapabilities bookmark).



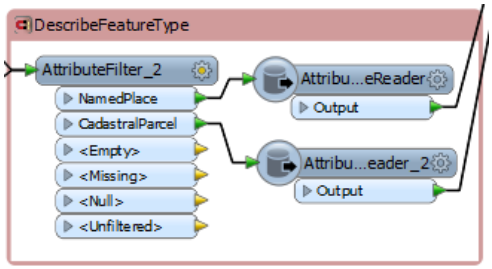
AttributeFilter parameters

3. Connect **AttributeFilter** CadastralParcel port to the **CadastralParcel** custom transformer, and enable the CadastralParcel transformer. Also, connect the output of **CadastralParcel** custom transformer to the **CadastralParcel output feature type** (INSPIRE Writer). This takes care of the GetFeature request.

4. Connect CadastralParcel output from **AttributeFilter_2** in the DescribeFeatureType bookmark to a new **AttributeFileReader_2**. In the **AttributeFileReader_2** parameters, set the source filename to ...\schemas\CadastralParcels.xsd. This takes care of the DescribeFeatureType request.



AttributeFileReader_2 parameters

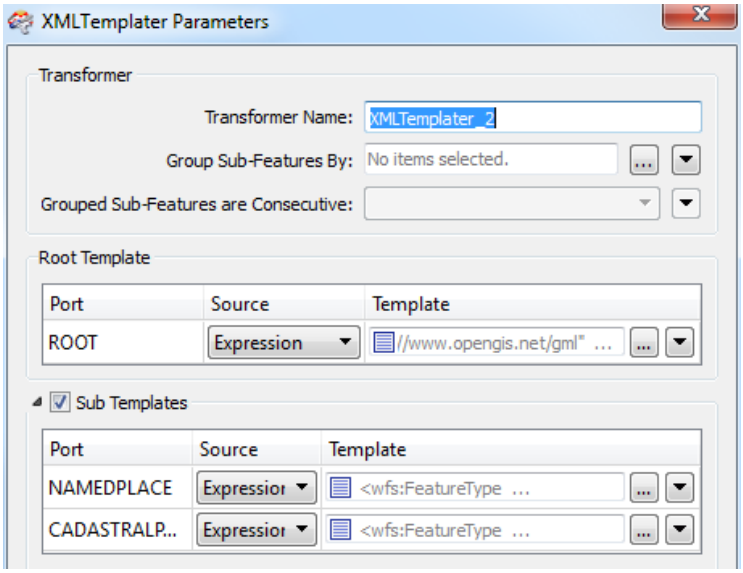


DescribeFeatureType Workflow

5. Add CADASTRALPARCEL subtemplate to **XMLTemplater_2**. Define the Cadastral Parcels GetCapabilities Template as follows:

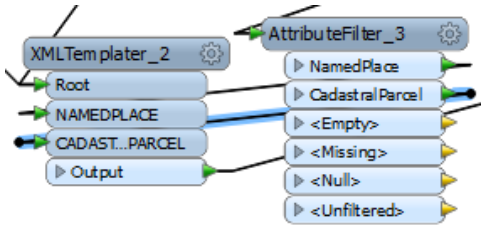
```
<wfs:FeatureType
xmlns="urn:x-inspire:specification:gmlas:CadastralParcels:3.0"
xmlns:base="urn:x-inspire:specification:gmlas:BaseTypes:3.2"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
updateSequence="0" version="1.1.0"
xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
<wfs:Name>{fme:get-attribute("theme")}</wfs:Name>
<wfs:Title>{fme:get-attribute("theme")}</wfs:Title>
<wfs:Abstract/>
<wfs:DefaultSRS>EPSG:4326</wfs:DefaultSRS>
<wfs:OutputFormats>
<wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
</wfs:OutputFormats>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-20 20</ows:LowerCorner>
<ows:UpperCorner>60 80</ows:UpperCorner>
</ows:WGS84BoundingBox>
</wfs:FeatureType>
```

6. In **XMLTemplater_2**, ROOT template:
   Right after:
   `{fme:process-features("NAMEDPLACE")}`
   Add in:
   `{fme:process-features("CADASTRALPARCEL")}`



XMLTemplater_2 parameters

7. Connect the CadastralParcel output of **AttributeFilter_3** to the CADASTRALPARCEL input of **XMLTemplater_2**. This completes the support for the GetCapabilities request.



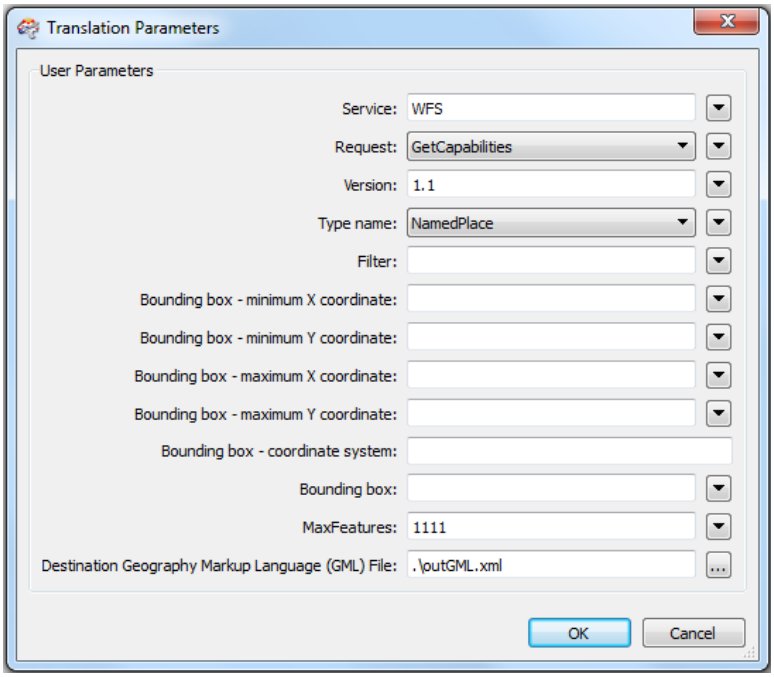
CadastralParcel output to CADASTRALPARCEL input

8. Duplicate the NamedPlace **Counter** and **Tester** transformers. In-between the **CadastralParcel custom transformer** and the INSPIRE GML Writer, add in the Counter and Tester.

9. Examine the data flow for each primary WFS request type: GetCapabilities, DescribeFeatureType, and GetFeature. Focus on GET as this is easier to follow.

10. 'Run with prompt' and see the behaviour of the workspace with different request types. This will show how the output is an XML document for GetCapabilities, an XSD for DescribeFeatureType and a GML document for GetFeature.

11. Examine the output files in an XML/text editor for each of the above request types. XML output goes to outText.xml, GML output goes to outGML.xml.



Run with prompt parameters

12. Open the output GML in Data Inspector using the INSPIRE reader.

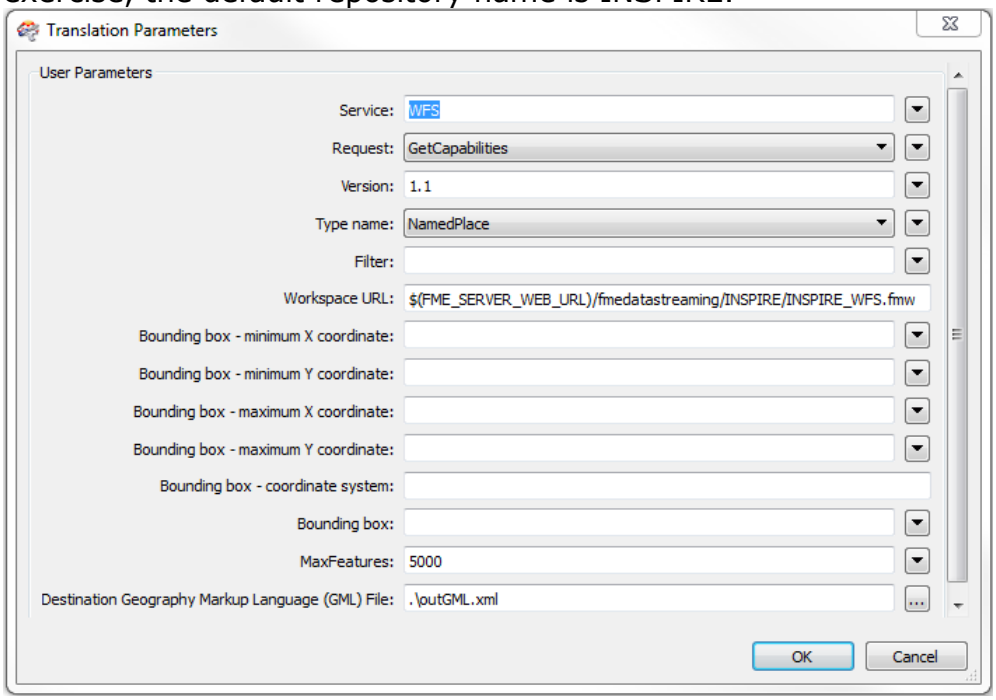
13. Validate your output using an XMLValidator and the relevent INSPIRE XSDs. XMLValidator.fmw is a workspace set up for XML syntax and schema validation. All is needed is for the XML input file and the application schema file to be specified in the XMLValidator transformer.

NOTE: If at any point you cannot complete your configuration of CadastralParcels, you can always start at step 8 with the completed workspace: INSPIRE_WFS.fmw.

PART D: Publishing your WFS to FMEServer

In this demo, INSPIRE_WFS.fmw will be published to FME Server, creating your own WFS service on FME Server.

1. Modify workspace URL parameters so that GetCapabilities is configured to point to your FMEServer and the correct repository is defined. For this exercise, the default repository name is INSPIRE.



User friendly parameters

2. Update WFS request parameters as needed (default theme, max features etc.)

3. Set INSPIRE GML writer default parameters such as axis order, SRS etc.

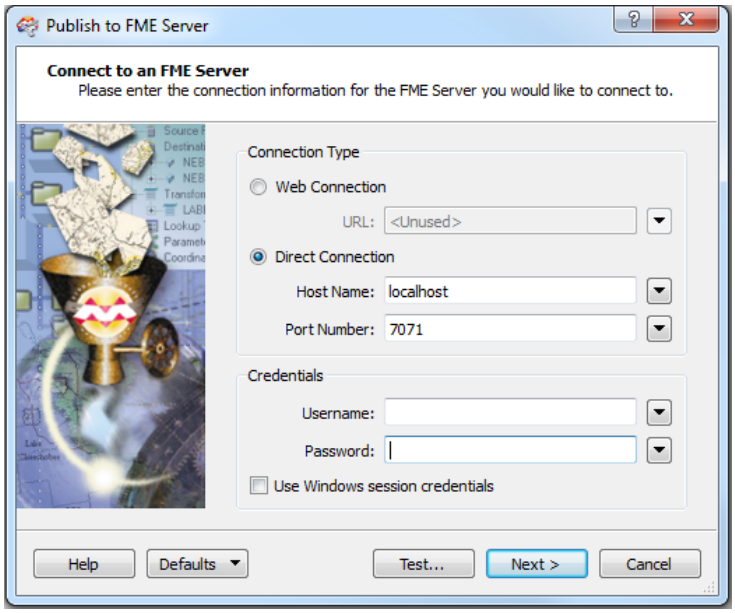
4. Test run workspace locally using each of GetCapabilities, DescribeFeatureType and GetFeature request types to make sure correct output is generated.

5. Prepare for publishing by changing request back to GetCapabilities and removing any filters before saving. The goal is to have user friendly default parameter values.

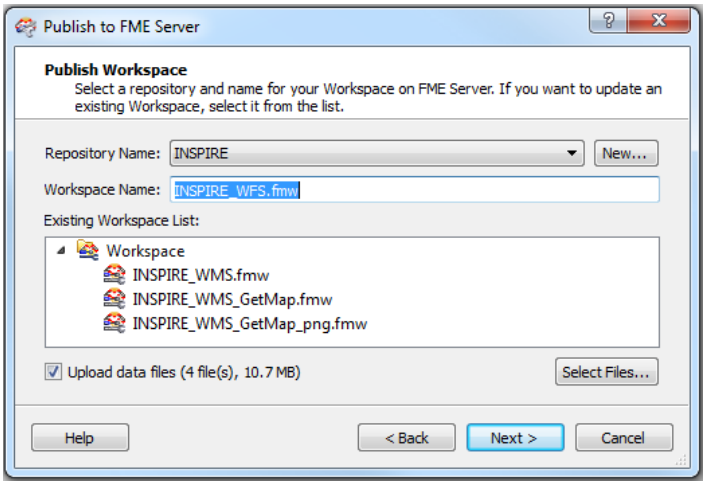
6. Publish to FME Server data streaming service. Select Publish to FME Server. Enter the details for your local server, for which the default name is localhost. Enter any required authentication info. Click Next.

   For this exercise use INSPIRE as the repository name. Upload all supporting resources by choosing Select Files. Ensure all of source_get.txt, INSPIREdb.sdf, GeographicalNames.xsd, and CadastralParcel.xsd are selected. Click Next.

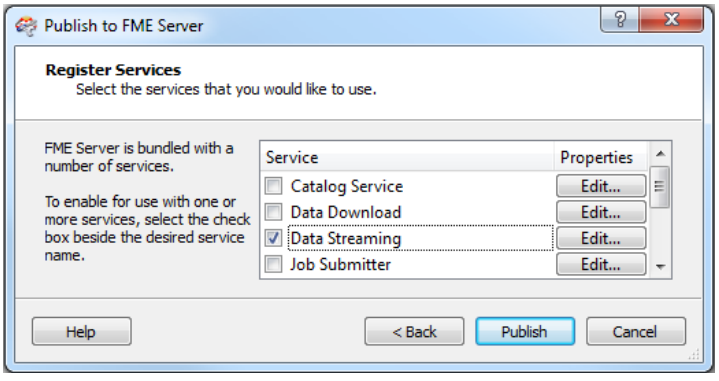
   For registered services, select only Data Streaming. Then click the Edit… make sure both the Text File writer and GML writer are published outputs. Click OK, then Publish.



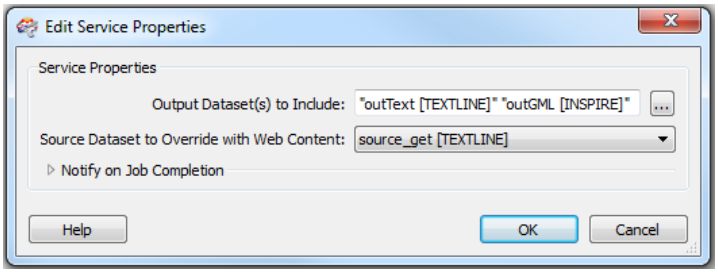
FME Server Connection



Naming and Upload of Resources



Registered Services

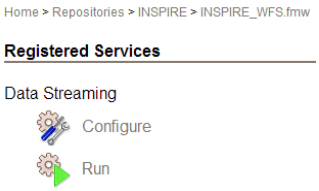


Published Outputs

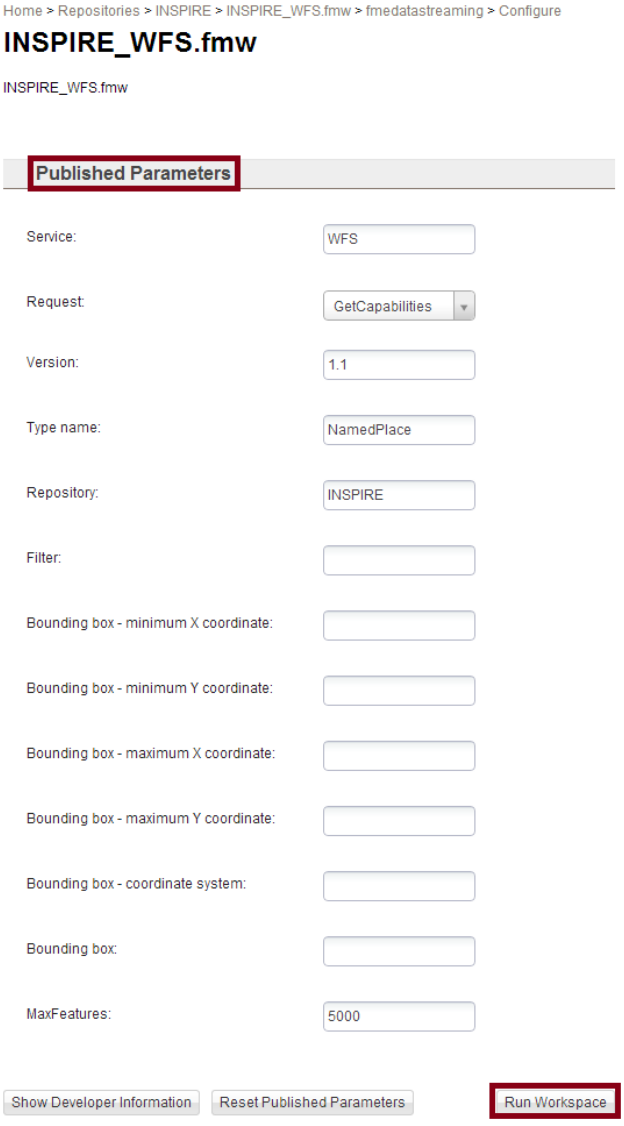
7. Go to your FME Server by entering **localhost** as the path. You will be required to enter a provided username and password. Under the INSPIRE repository, verify that the workspace was published correctly.

   Click on the workspace and click Run to make sure it works ok. A GetCapabilities document should be output if that was what you set as your default parameters.

8. Then click on the workspace again, but this time select Configure. Here you can alter various parameters, then run the workspace. At the bottom of the page, click Show Developer Information. Copy and paste the direct URL example into a web browser to ensure that the server can be called from an external client.



Run or Configure Workspace



Configuration Parameters

9. Finally, try out your WFS from Data Inspector, by adding a WFS dataset and using the core part of the path as the dataset path:

   ```
   http://<Host>/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw
   ```

   Where <Host> corresponds to the name of your FME Server

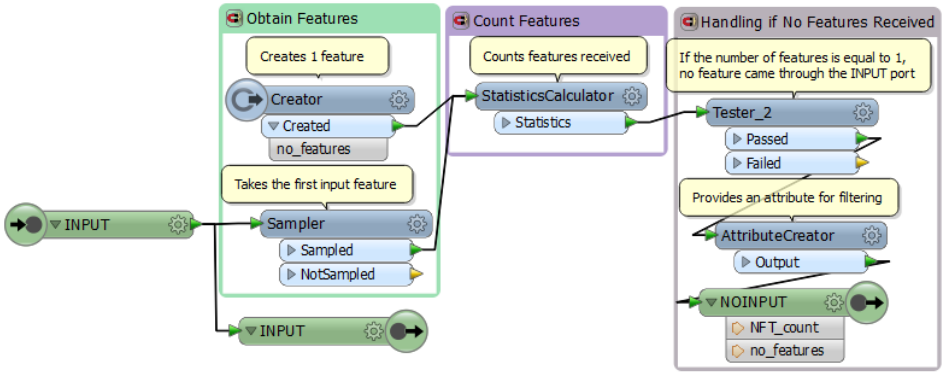
   Note that if you paste the above URL into a web browser and you correctly set your default WFS request type to GetCapabilities, then this URL should return a GetCapabilities response. Note that you will need to set your user and password authentication in your browser and Inspector settings.

   Example GetCapabilities response:

   ```
   <wfs:WFS_Capabilities xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:
   wfs="http://www.opengis.net/wfs" xmlns:ows="http://www.opengis.net/ows"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.
   opengis.net/gml" updateSequence="0" version="1.1.0" xsi:schemaLocation=
   "http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
   <!--  Service Identification Section  -->
   <ows:ServiceIdentification>
   <ows:Title>
   WFS Service Compliant with OGC WFS Specification Version 1.1.0
   </ows:Title>
   <ows:Abstract>
   An OGC WFS Service created by Safe Software's FME Server
   </ows:Abstract>
   <ows:ServiceType>WFS</ows:ServiceType>
   <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
   …
   ```

PART E: Bonus WFS Exercise

This WFS does not produce valid output if no features are returned from the FeatureReaders. This can easily happen if the constraints defined by the WFS parameters are too restrictive. For example, if there is no data in the extents or XML filter(s) specified. Thus, for a GetFeature request with such filters, Inspector will simply yield an XML parser error, or worse just stall and eventually time out.



NoFeaturesTester Workflow

To avoid this, use a NoFeatureTester custom transformer and StringConcatentor to build an XML error message stating something to the effect:

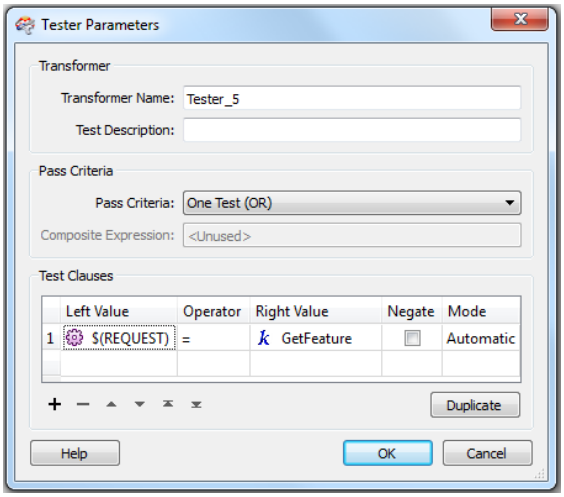
```
<error>No features for provided WFS GetFeature request parameters</error>
```

This way, the WFS client knows why there is no valid data returned. Also, this is a simple illustration as to how flexible and customizable this workspace based approach to providing web services can be.

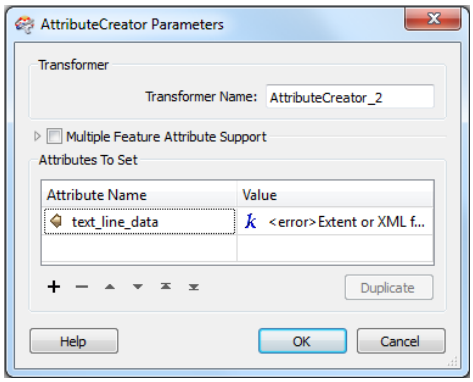
1. Connect the Passed outputs, from the MaxFeatures **Counter** transformers of both feature types, to a **NoFeaturesTester**.

2. Connect the NOINPUT port from the **NoFeaturesTester** to **Tester_5**.

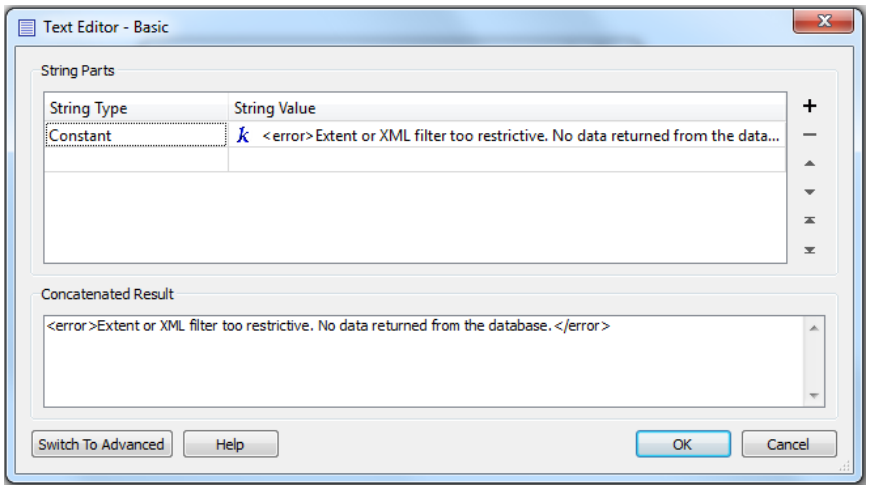
3. Connect the Passed **Tester_5** port to an **AttributeCreator_2** transformer.



Tester_5 parameters

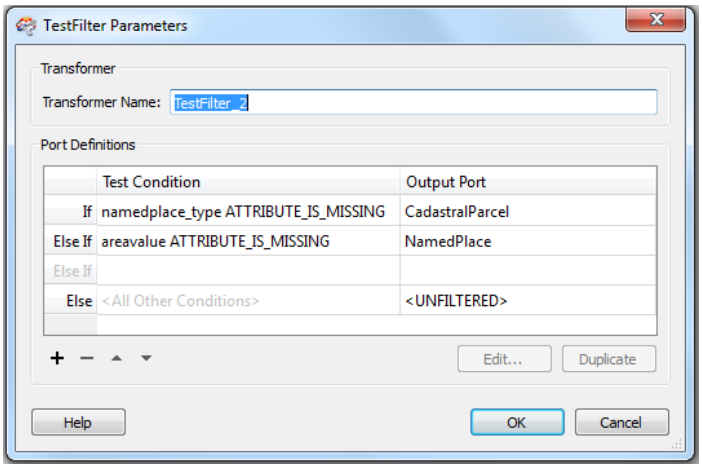


AttributeCreator_2 parameters



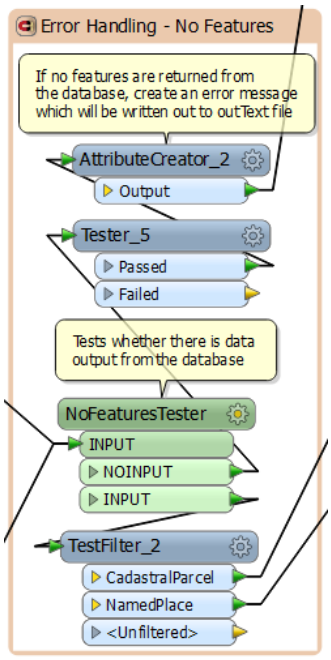
AttributeCreator_2 Text Editor

4. Connect the INPUT port from the **NoFeaturesTester** to **TestFilter_2**.



TestFilter_2 parameters

The error handling should look like:



OGC WMS for INSPIRE

WMS Background

The principles explored within the WFS exercise above apply equally to other web services. Let's examine how we can build a WMS service from scratch.

Building a new web service requires 3 main components:

- FME workspace that provides data to the client via FME Server's data streaming service.
- GetCapabilities XML document that tells the client what services and data layers are available.
- FME workspace that will act as the service broker for the web service. It needs to accept service requests and transmit responses according to the service specification requirements.

One fundamental difference between this exercise and the WFS one is that for WMS we use one workspace to do the message handling and a separate one for the data streaming. The reason we have to do this here is simple. To date, FME only allows one type of MIME encoding per workspace. For WFS this isn't a limitation since both the service messages and the GML results are XML which are encoded as application/xml. However, WMS is different. The GetCapabilities response is XML, while the GetMap response is typically image/png or image/jpg. So these two responses cannot coexist within the same workspace, at least not as of FME 2014.

To work around this we instead use 2 workspaces. One workspace simply listens for GetCapabilities requests and transmits the appropriate response in the form of application/XML documents. The other workspace parses GetMap requests and transmits the actual WMS data responses in the form of image/png data. The key to getting this to work is to edit the GetCapabilities document so that each service request calls the appropriate workspace.

Exercise 2: WMS Service Broker Workspace - GetCapabilities

1. Make a backup copy of the GetCapabilities.xml, and then open C:\FMEData2014\Resources\INSPIRE\OGCservices-INSPIREfocus\WMS\GetCapabilities.xml in an XML or text editor of your choice.

2. Examine the different Request type defined, including GetCapabilities and GetMap. Note how the URLs used point to different workspaces published to FME Server's data streaming service.

```
<Request>

  <GetCapabilities>
     <Format>application/vnd.ogc.wms_xml</Format>
        <DCPType>
          <HTTP>
            <Get>
               <OnlineResource
               xlink:href="http://localhost/fmedatastreaming/INSPIRE/INSPIRE_WMS.fmw"
               xlink:type="simple" xmlns:xlink="http://www.w3.org/1999/xlink"/>
            </Get>
          </HTTP>
        </DCPType>
   </GetCapabilities>
   <GetMap>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/jpeg</Format>
        <Format>image/svg+xml</Format>
        <DCPType>
           <HTTP>
               <Get>
                  <OnlineResource
                       xlink:href="http://localhost/fmedatastreaming/INSPIRE/INSPIRE_WMS_GetMap_p
                       ng.fmw" xlink:type="simple" xmlns:xlink="http://www.w3.org/1999/xlink"/>
               </Get>
           </HTTP>
        </DCPType>
    </GetMap>
```

WMS GetCapabilities response document – section showing request responses

3. Edit your **GetCapabilities.xml**. If you are running on a local FME Server then leave this as is. If you want to run on an external FME Server then you need to change all occurrences of **localhost** to the desired FMEServer **host name**. Contact your instructor if you are unsure. Save your changes.

Next we need to set up the GetCapabilites and GetMap workspaces

4. For the GetCapabilities workspace, open a new, blank workspace. Add a Text File reader:
   Reader format: Text File
   Reader dataset: GETrequest.txt

5. Add a Text File writer as follows:
   Writer format: Text File
   Writer dataset: messageResponse.xml

6. Edit your reader and writer settings. Make sure your reader has 'Read whole file at once' set to 'yes'. Also, make sure the MIME type is set to application/xml.

Next we need to create workspace parameters so that key terms from the client's GET request can be parsed.

Essentially what happens is as follows. For the following request URL:

```
http://localhost/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.1.1
```

By examining the URL closely, we can see the following parameters: SERVICE, REQUEST and VERSION.

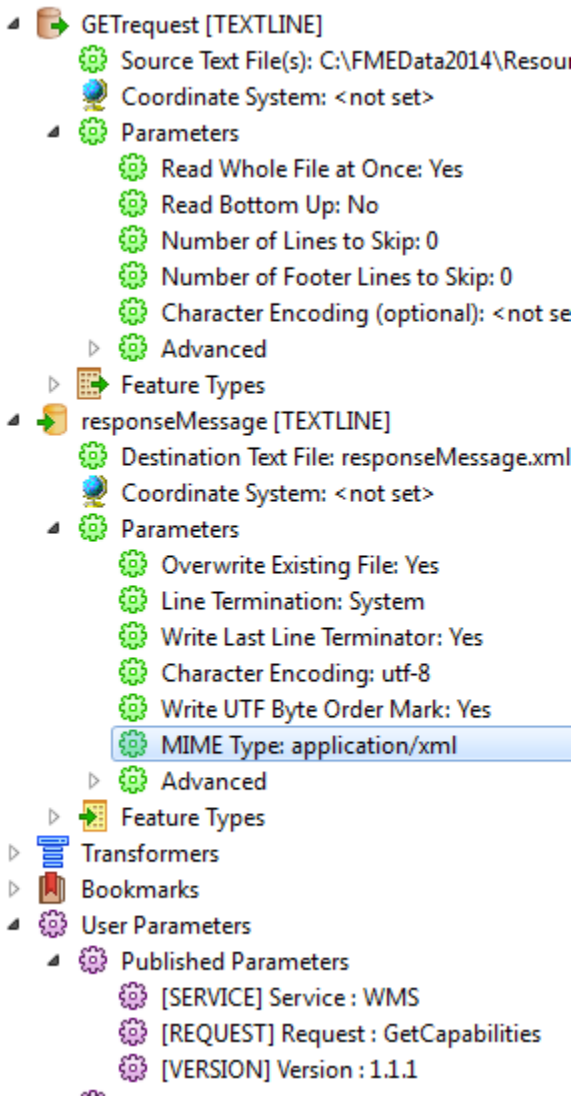
7. Create workspace level user parameters and defaults as follows:

   SERVICE = WMS

   REQUEST = GetCapabilities (this way the service defaults to a GetCapabilities even if no request is specified.)

   VERSION = 1.1.1

   We could add version handling here so that the correct format of GetCapabilities.xml is returned depending on what the client asks for: WFS 1.1, 2.0 etc.

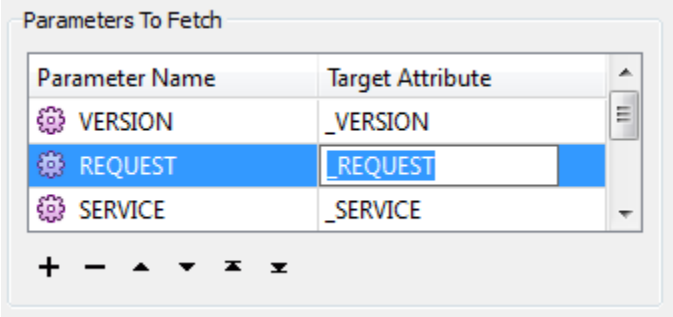


WMS workspace user parameters

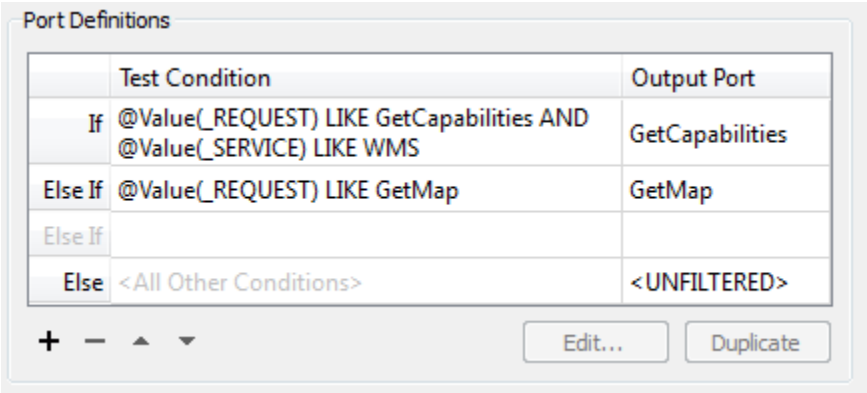
Also, it is worth noting from the navigator's view of the parameters that the text file reader is set to read the whole file at once. This is less important for URLs as these are usually one line, but would be important if the input was HTTP POST.

Also note the MIME encoding.

8. Next add a ParameterFetcher so that we can use these values within the workspace. Select each of SERVICE, REQUEST and VERSION and use the default target attribute names as below:



9. Add a **TestFilter**. We will use this to check which request and service types the user has requested. Set up these **test conditions**:
   If @Value(_REQUEST) LIKE GetCapabilities AND @Value(_SERVICE) LIKE WMS; Output Port = GetCapabilities
   Else If @Value(_REQUEST) LIKE GetMap; OutputPort = GetMap

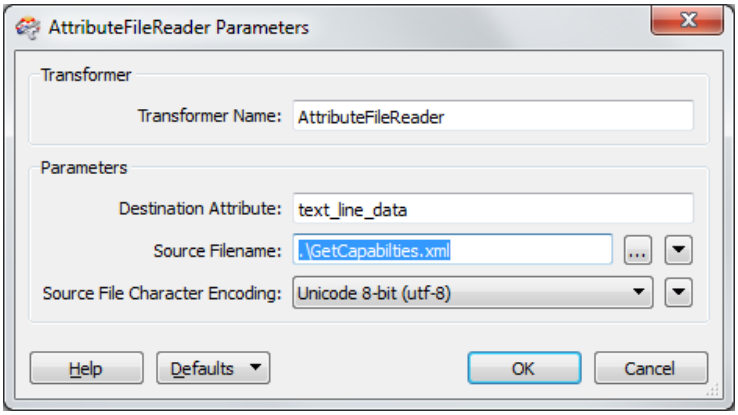


   Connect the GetMap and Unfiltered ports to a Logger so we can log unexpected inputs.

10. Add an **AttributeFileReader** to pull in the content of GetCapabilities.xml for streaming to the client.

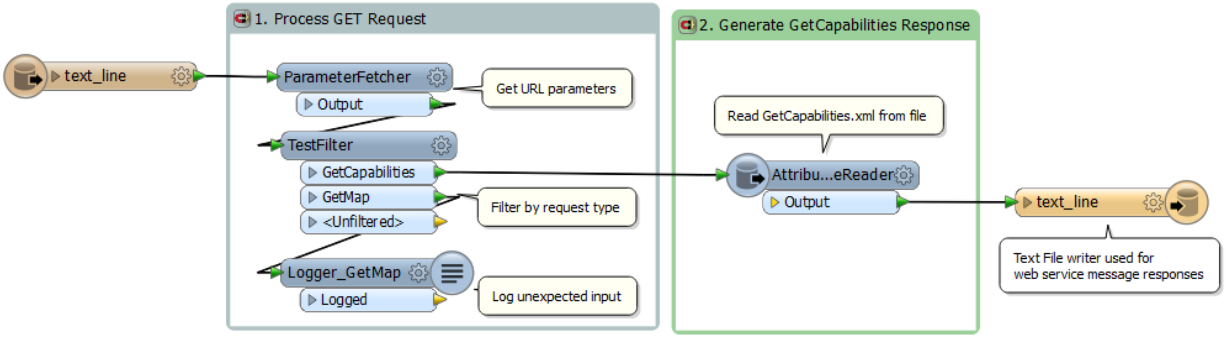
Set **Destination Attribute** = text_line_data

Source Filename = .\GetCapabilities.xml



11. Connect the output of the **AttributeFileReader** to the text_line destination feature type.

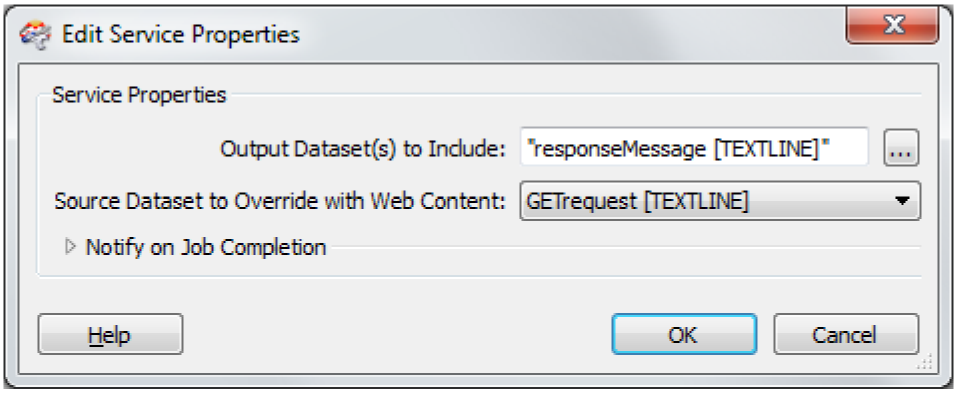
    Your workspace should now look something like this:



WMS service broker & message handling workspace

12. Publish your workspace to FME Server.
    - Make sure your published parameters are set to valid default values before you publish.
    - Select File > Publish to FME Server
    - In the wizard under 'Connect to FME Server', enter localhost (or whatever the Server hostname is), and any required authentication info.

- Next, on the 'Publish workspace' frame, make sure INSPIRE_WMS.fmw is selected. Then click New under the repositories and define a new repository called 'INSPIRE' or select it if it already exists.
- Click on 'select files' and make sure Getrequest.txt and GetCapabilities.xml are selected
- Under 'Register Services' make sure Data Streaming service is checked. Click on 'Edit' beside Data Streaming



Data streaming service properties during service registration

- Click 'Publish'

You should see the following message:

Successfully published to FME Server as 'INSPIRE_WMS.fmw'.
=-
Publish Summary
=-
Server Host : localhost
Server Port : 7071
Username : author
Repository : INSPIRE
Target Name : INSPIRE_WMS.fmw
Resources Uploaded :
\\bester\pserv\Projects\INSPIRE\Training\FMEUC2014_training\OGCservices-INSPIREfocus\WMS\GetCapabilties.xml
\\bester\pserv\Projects\INSPIRE\Training\FMEUC2014_training\OGCservices-INSPIREfocus\WMS\GETrequest.txt
Services Registered : Data Streaming
=-

Log for successful workspace publication to FME Server

WMS GetCapabilities Bonus Exercise:

Right now the workspace should work fine so long as it gets valid input from the client according to the OGC WMS standard. However, as with any standard, OGC standards are interpreted slightly differently by different vendors. Sometimes requests are encoded or composed differently.

To cover the potential of server / client miscommunication, add an error handling section to address those cases where the service, request or version parameters do not match what is expected or supported.

Hint: use the GetMap and Unfiltered outputs from the TestFilter, and add an XMLTemplater that contains something like:

```
<error>
    <unexpectedInput>
       <warning>Unexpected Input Received from WFS client</warning>
       <request>{fme:get-attribute("_REQUEST")}</request>
       <service>{fme:get-attribute("_SERVICE")}</service>
       <version>{fme:get-attribute("_VERSION")}</version>
    </unexpectedInput>
</error>
```

Sample warning from INSPIRE_WMS.fmw for unexpected input

Note that warnings such as this should be viewable in the client's log. Also, it is often useful to troubleshoot OGC request problems using a web browser. At a minimum, this message will display in any web browser when there is a problem with a request, such as if you ask for something unsupported such as SERVICE = WCS.

You have now published the main service broker workspace for your WMS service.

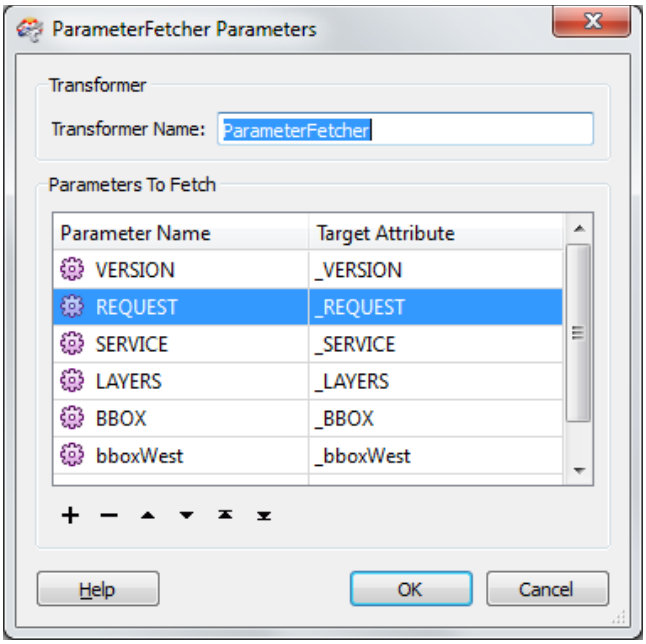
Exercise 3: WMS Get Map Data Streaming

Now we need to configure and publish the workspace that handles the GetMap requests and streams back the appropriate raster images. To do this we will build up the workspace from scratch with the benefit of some custom transformers.

1. Open INSPIRE_WMS_GetMap_Start.fmw

2. Add a **Text File Reader**. You can select the same source dataset as before: Getrequest.txt. Here this is a placeholder that allows the workspace to run since the parameter values in this case come via the workspace parameters.

3. Add a **ParameterFetcher** after the text_line input. Configure it as follows:



   (Or as a shortcut just copy the ParameterFetcher and TestFilter from your INSPIRE_WMS.fme workspace, and then add the new parameters.)

4. Add a **TestFilter** transformer after the ParameterFetcher. Configure it as before with these **test conditions**:

```
If @Value(_REQUEST) LIKE GetCapabilities AND @Value(_SERVICE) LIKE WMS; Output
Port = GetCapabilities
Else If @Value(_REQUEST) LIKE GetMap; OutputPort = GetMap
```

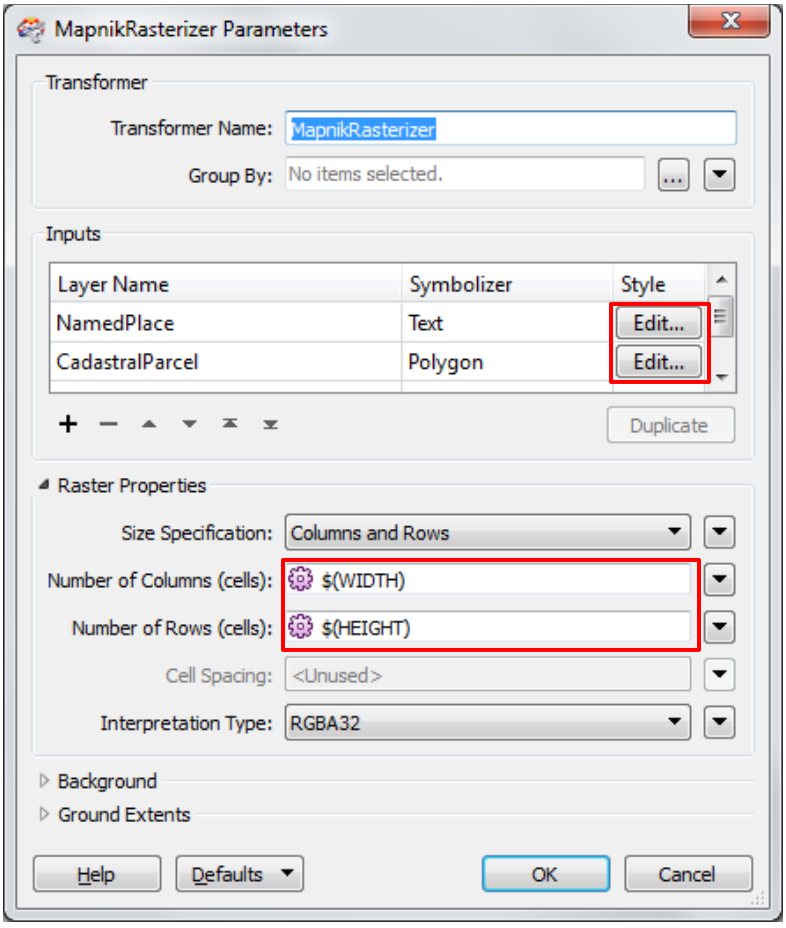
5. This time, connect a **Logger** to the GetCapabilities output, to capture unexpected inputs, since here we expect GetMap requests only.

6. Connect an **AttributeFilter** after the TestFilter. Filter on the _LAYERS field. Define possible values as NamedPlace and CadastralParcel.

7. Connect the **NamedPlace** custom transformer to the NamedPlace output port of the AttrbuteFilter and the **CadastralParcel** custom transformer to the CadastralParcel output.

8. Connect the outputs of the NamedPlace and CadastralParcel custom transformers to a **MapnikRasterizer**.

9. Configure it as below. Add inputs for NamedPlace and CadastralParcel. Set NamedPlace to Text symbolizer and CadastralParcel to Polygon.
Under Raster Properties, set Columns to the WIDTH parameter and Rows to the HEIGHT parameter.



MapnikRasterizer configuration

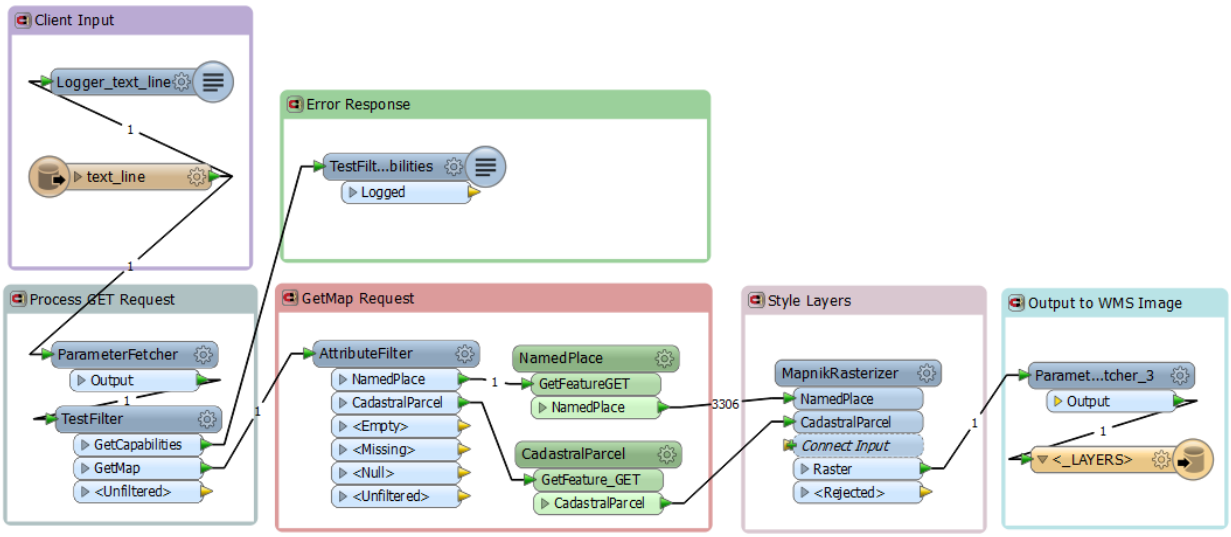
10. Edit the **MapnikRasterizer** Text symbolizer for NamedPlace.
    Set the text properties text field to:
    name_geographicalname_spelling_spellingofname_text
    Set the color to anything you wish.

11. Edit the **MapnikRasterizer** Polygon symbolizer for CadatralParcel. Select any color you like.

12. Copy the **ParameterFetcher** and connect it to the output of the MapnikRasterizer.

13. Add **a PNGRaster Writer**. Set it to fanout on _LAYERS. Set the output directory to be '.'. Set the output coordinate system to be: LL84.

    Your INSPIRE_WMS_GetMap.fmw workspace should now look something like:

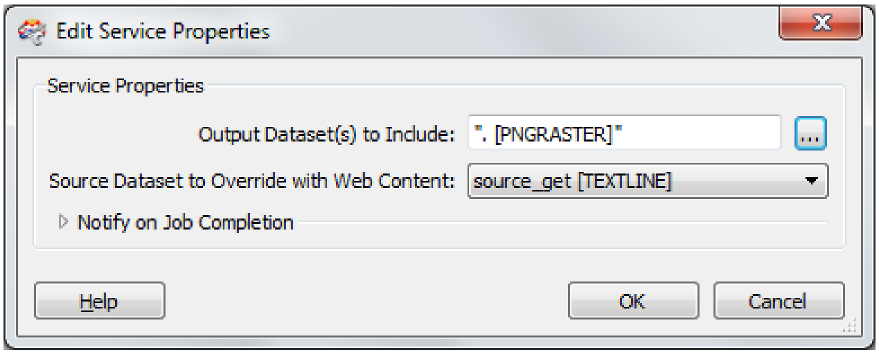


Completed INSPIRE_WMS_GetMap.fmw data streaming workspace

14. Test run your workspace using 'File > Run with prompt'. This is always a good way to preview a workspace before you publish it to FME Server.

15. **Publish** your workspace to FME Server.

- Make sure your published parameters are set to valid default values before you publish.
- Select File > Publish to FME Server
- In the wizard under 'Connect to FME Server', enter localhost (or whatever the Server hostname is), and any required authentication info.
- Next, on the 'Publish workspace' frame, make sure INSPIRE_WMS_GetMap_png.fmw is selected. Then select the repository called 'INSPIRE' (created last exercise).
- Click on 'Select Files' and make sure Getrequest.txt and INSPIREdb.sdf are selected.
- Under 'Register Services' make sure Data Streaming service is checked. Click on 'Edit' beside Data Streaming



Data streaming service properties during service registration

- Click 'Publish'

You should see the following message:

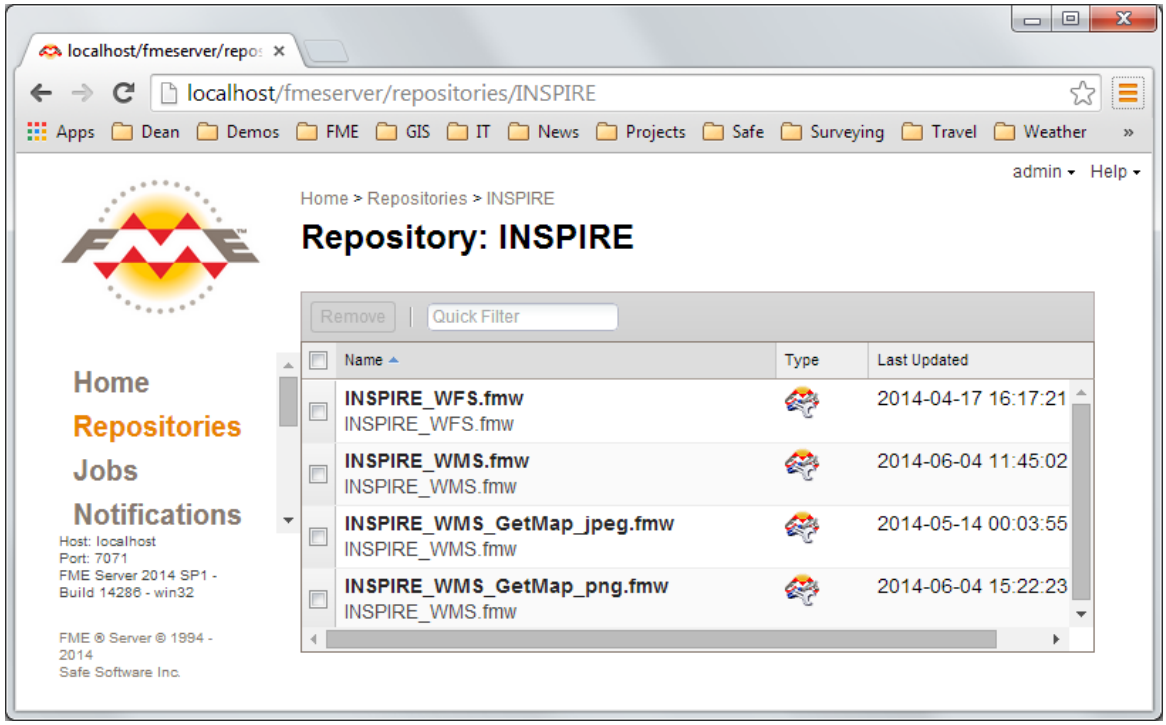
Successfully published to FME Server as 'INSPIRE_WMS_GetMap_png.fmw'.
=-
Publish Summary
=-
Server Host : localhost
Server Port : 7071
Username : author
Repository : INSPIRE
Target Name : INSPIRE_WMS_GetMap_png.fmw
Resources Uploaded :
\\bester\pserv\Projects\INSPIRE\Demos\WebServices\WMSservice\WMS_SDF\INSPIREdb.sdf
\\bester\pserv\Projects\INSPIRE\Demos\WebServices\WMSservice\WMS_SDF\source_get.txt
Services Registered : Data Streaming
=-

Log for successful workspace publication to FME Server

Exercise 4: Testing your WMS Workspaces on FME Server

Now that we have published the supporting workspaces to FME Server, it's time to test it out.

1. First, go to FME Server and verify that everything is published as it should be. Go to the **INSPIRE** repository and confirm that you have INSPIRE_WMS.fmw and INPIRE_WMS_GetMap_png.fmw published.



FME Server view of INSPIRE OGC repository

2. Click on INSPIRE_WMS.fmw, then on 'Configure', and then on 'Show Developer Information'

   You should see URL samples such as:

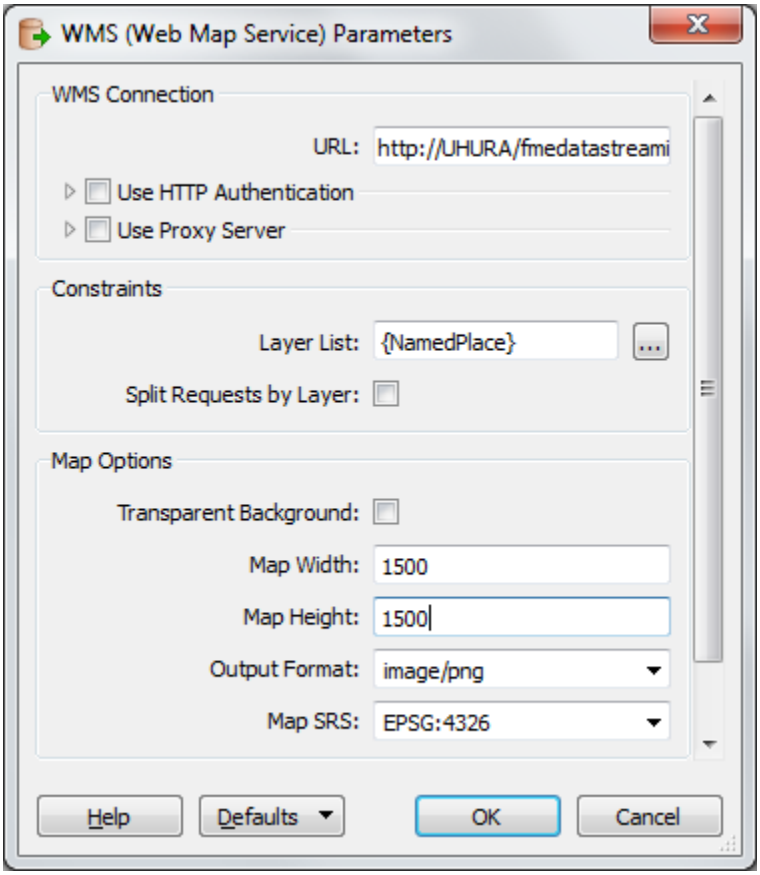
   Direct Url Example

   ```
   http://UHURA/fmedatastreaming/INSPIRE/INSPIRE_WMS.fmw?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.1.1
   ```

3. Copy this URL, open Data Inspector, select Open Dataset, and choose **WMS**.

4. Paste the URL into the dataset field:
   http://UHURA/fmedatastreaming/INSPIRE/INSPIRE_WMS.fmw?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.1.1

   Click on the Layer List […], and select one layer such as **NamedPlace**



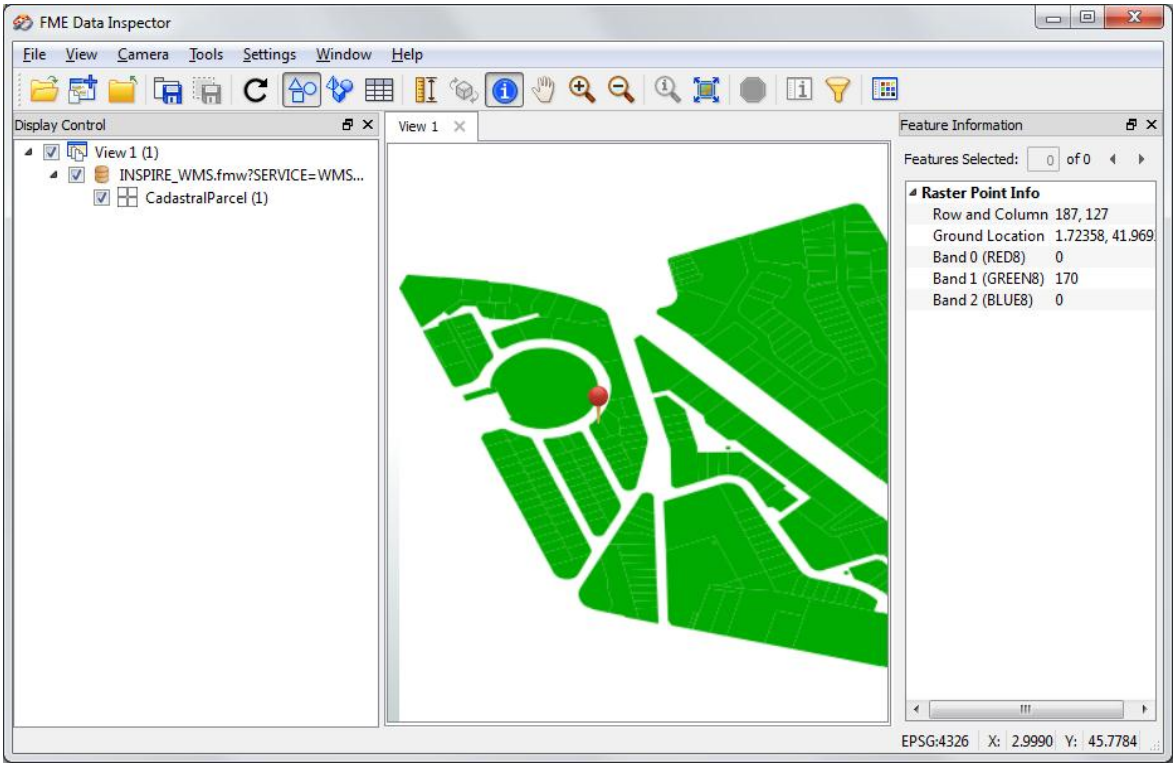
WMS Reader parameters

Make sure background maps are turned off as this can interfere. Click OK and see what happens.

If you have any problems examine the log carefully. You can always click on any of the URLs embedded in the log to try opening them in your web browser. This will often help you determine if you are missing some key connection element such as a login.

Without any spatial extents the labels on NamedPlace will not show up well. Try entering the following extents: xMin=0, yMin=33, xMax=12, yMax=45.

You may have to experiment with the extents to get the results you want. Note that if you choose extents that are too restrictive or result in 0 features returned, you will likely just get a WMS reader error message.



Data Inspector WMS client to INSPIRE_WMS.fmw deployed on FME Server

Congratulations! You have just implemented your own WMS on FME Server simply by configuring a workspace- without any coding.

WMS Bonus Exercise:

Option A:
Add logic to your INSPIRE_WMS_GetMap_png.fmw workspace to handle 0 features returned. You could detect 0 features coming back from the database and then generate an appropriate warning. Or you could try to detect that the result from MapnikRasterizer is not a raster. Then you could return a default or blank raster so that you always generate a valid raster, result regardless of the request.

Option B:
Add another layer to your WMS. You can publish virtually anything you want. You will need to add the corresponding layer name, FeatureReader and styling to the INSPIRE_WMS_GetMap_png.fmw workspace. You will also need to make sure you reproject everything to LL84 it comes from some other reference. Or you could simply add a label layer to the parcels.

Appendix A. WFS – Workspace Documentation

INSPIRE_WFS.fmw

OVERVIEW

This workspace comprises an INSPIRE WFS when published to the FME Server data streaming service.

The local server url is:
http://localhost/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw

Safe Software's WFS demo server url is:

http://inspire-safe-software.fmecloud.com/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw

It also supports writing INSPIRE GML using the XSD enabled INSPIRE GML Writer available as of FME 2014. It accepts GetCapabilities, DescribeFeatureType and GetFeature requests, either as GET URL or POST XML. Also, it supports Filter expressions.

When you send an HTTP POST to FME Server, it automatically overrides the input of this workspace with the post body. Note this means you need to have only one input and you assume that it is HTTP POST WFS XML. If there is a POST, then the POST body is the WFS request, such as the WFS GetFeatures XML with the embedded XML filter query. The workspace supports FME Viewer and FME Data Inspector WFS clients. It should support third party WFS clients as well.

Setup:

1. Modify workspace paths to make sure source database is accessible by FeatureReaders in the theme data extraction custom transformers (CadastralParcels and NamedPlace).

2. Update workspace parameters as needed (default theme, max features etc.)

3. Set GML writer default parameters such as axis order, SRS etc.

4. Test run workspace using each of GetCapabilities, DescribeFeatureType and GetFeature request types to make sure correct output is generated

5. Prepare for publishing by changing request back to GetCapabilities and removing any filters before saving. The goal is to have user friendly default parameter values.

6. Publish to FME Server data streaming service. Upload all supporting resources: source_get.txt etc. Make sure both Text File writer and GML writer are published outputs. Note xsds are no longer need to be uploaded as they are included with the INSPIRE writer. If you wish to use a draft or newer schema than what is delivered with FME you will need to make sure these are uploaded.

USAGE NOTES:

Make sure you publish with a non-xml source so that by default the XML test will fail.

Also make sure parameter Filter = "" and Request = GetCapabilities by default

GeoServer uses TypeNames and FME Viewer client uses TypeName

If getting an error message when making a request, run the following WFS diagnostics. First, try to make a simple request, such as setting max features to a low number to see whether the WFS is responding. Then, if the WFS is responding, examine the syntax of the desired request to check for errors.

OUTSTANDING

Currently only handles WFS 1.1 / GML 3.1.1.

Only handles queries to one theme at a time.

Need to add support for WFS 2.0. This should be straight forward - just need to update template headers and use GML 3.2.1 geometry extraction. Needs to handle POST extents query

Limited error handling in the workspace. Only able to output a specific error message if no data is received from the database. Bad feature requests may output unhelpful messages, just yields an XML error. We should detect this and return a valid GML or XML error feature. For example, if the XML filter has a syntax error, log may return an XML parser error, when it would be more descriptive to state that it does not have a not valid name.

Cannot currently make POST requests for CadastralParcels, only can make POST request for NamedPlace

Reading Safe's INSPIRE WFS

Sample Requests and Filters

Use this URL:

http://inspire-safe-software.fmecloud.com:80/fmedatastreaming/INSPIRE/INSPIRE_WFS.fmw

Backup / development Server:

http://inspire-safe-software.fmecloud.com:80/fmedatastreaming/INSPIRE/INSPIRE_WFSdev.fmw

*** Only query one theme at a time NamedPlace or CadastralParcels but not both at once***

Sample GetFeature request:

http://inspire-safe-software.fmecloud.com:80/fmedatastreaming/INSPIRE/INSPIRE_WFSdev.fmw?SERVICE=WFS&VERSION=1.1.0&REQUEST=GetFeature&TYPENAME=CadastralParcel&MAXFEATURES=1111

CADASTRAL PARCELS QUERIES:

Parcels XML FILTERS:

<Filter><PropertyIsEqualTo><PropertyName>inspireId.Identifier.localId</PropertyName><Literal>136000AZ0063</Literal></PropertyIsEqualTo></Filter>

(Generates: `_whereQuery' has value `inspireId.Identifier.localId='136000AZ0063'')

<Filter><PropertyIsEqualTo><PropertyName>gml_id</PropertyName><Literal>id-ff84b9a7-5e8e-458b-8488-fd5d01d9ea29</Literal></PropertyIsEqualTo></Filter>

<Filter><PropertyIsEqualTo><PropertyName>nationalCadastralReference</PropertyName><Literal>66136000AZ0492</Literal></PropertyIsEqualTo></Filter>

Parcels SPATIAL QUERY:

0 to 5, 40 to 50 (W France)

GEOGRAPHIC NAMES QUERIES:

NamedPlace XML FILTERS:

<Filter><PropertyIsEqualTo><PropertyName>GeographicalName_language</PropertyName><Literal>Italian</Literal></PropertyIsEqualTo></Filter>

<Filter><PropertyIsEqualTo><PropertyName>geographicalname_spellingofname_text</PropertyName><Literal>Roma</Literal></PropertyIsEqualTo></Filter>

NamedPlace Spatial query:

0 to 5, 40 to 50 (W France)

0 to 25, 42 to 48 (should cover both Italy and S France so results overlap with parcels)

Can also get schemas from:

http://inspire.ec.europa.eu/schemas/cp/3.0/CadastralParcels.xsd etc.

Safe Software values its customers' opinions very highly.
For questions and concerns, please use the general feedback page at: **www.safe.com/contact**, or email the Training Manager directly at: **training@safe.com**.

4102-01-A