XMLTemplater – Writing GML to a complex application schema

In this exercise we will read UN gazetteer data and use XMLTemplaters to write complex GML 3.2.1 data compliant with the INSPIRE Geographic Names application schemas.

XML Templater – Basic Feature / Dataset Example	
Scenario	FME user, national European mapping agency
Data	C:\FMEData\VARtraining\source\Europe_LL
Overall Goal	Generate Geographic Names INSPIRE GML
Demonstrates	XMLTemplater, XMLValidator

For this exercise we start with a prepared workspace that handles assembling the source data via the necessary source files and joins, and then transforming it to the INSPIRE schema using SchemaMapper. The XML part of this problem is how to take relational geographic names data and write it out as GML 3.2.1 data compliant with the INSPIRE Geographic Names application schemas. The instructions below will guide you through these steps:

1. Start with a prepared workspace:
C:\FMEData\VARtraining\XML\Ex6_INSPIREgeographicNames/GeoNames_shp2INSPIREgml_start.fmw

2. We will be working in the GML Generation Bookmark and start in the Feature XML bookmark. After the AttributeKeeper add a CoordinateExtractor. Accept the defaults. This will provide us with the _x, _y values we need. It is common in GML writing to use a coordinate or geometry extractor to generate geometry data for use in XMLTemplater.

3. After the CoordinateExtractor add an XMLTemplater to process your features. Set the output xml field to be '_features'. Use a template expression and copy and paste in the sample record in to serve as your template start:

   C:\FMEData\VARtraining\source\XML\GeographicNamesSample.gml

   Then insert {fme:get-attribute("")} commands for the following fields in the appropriate locations:
   - name_GeographicalName_featureID
   - name_GeographicalName_pointID
   - _x,_y
   - _uuid
   - name_GeographicalName_namespace
   - name_GeographicalName_language
   - name_GeographicalName_nativeValue

When you are complete, your feature template expression should look like:

```
<NamedPlace gml:id="{fme:get-attribute("name_GeographicalName_featureID")}"
xmlns="urn:x-inspire:specification:gmlas:GeographicalNames: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/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:x-inspire:specification:gmlas:GeographicalNames:3.0
../XSD/GeographicalNames.xsd">
<beginLifespanVersion>2010-01-01T12:00:00</beginLifespanVersion>
<geometry>
<gml:Point gml:id="{fme:get-attribute("name_GeographicalName_pointID")}"
srsName="urn:ogc:def:crs:EPSG::4258">
<gml:pos>{fme:get-attribute("_x")} {fme:get-attribute("_y")}</gml:pos>
</gml:Point>
</geometry>
<inspireId>
<base:Identifier>
<base:localId>{fme:get-attribute("_uuid")}</base:localId>
<base:namespace>{fme:get-
attribute("name_GeographicalName_namespace")}</base:namespace>
</base:Identifier>
</inspireId>
<localType>
<gmd:LocalisedCharacterString locale="en-GB">{fme:get-
attribute("typeLocal")}</gmd:LocalisedCharacterString>
</localType>
<name>
<GeographicalName>
<language>{fme:get-attribute("name_GeographicalName_language")}</language>
<nativeness>endonym</nativeness>
<nameStatus>standardised</nameStatus>
<sourceOfName/>
<pronunciation>
<PronunciationOfName/>
</pronunciation>
<spelling>
<SpellingOfName>
<text>{fme:get-attribute("name_GeographicalName_nativeValue")}</text>
<script>Latn</script>
</SpellingOfName>
</spelling>
</GeographicalName>
</name>
<type>Administrative unit</type>
</NamedPlace>
```

This will generate the xml we need for each Geographic Name record.

4. Before we write all the features to our dataset we need to assemble them into a single list per feature type. Add a Listbuilder and leave the output as '_list'

5. Add another XMLTemplater, this time for the dataset. Enter the following template expression:

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:FeatureCollection gml:id="AT.HR.IT.0"
xmlns="urn:x-inspire:specification:gmlas:GeographicalNames: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/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:x-inspire:specification:gmlas:GeographicalNames:3.0
../XSD/GeographicalNames.xsd">
<gml:boundedBy>
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4258">
<gml:lowerCorner>7.0 35.0</gml:lowerCorner>
<gml:upperCorner>20.0 50.0</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<gml:featureMembers>
 {fme:get-xml-list-attribute("_list{}._features")}
 </gml:featureMembers>
</gml:FeatureCollection>
```

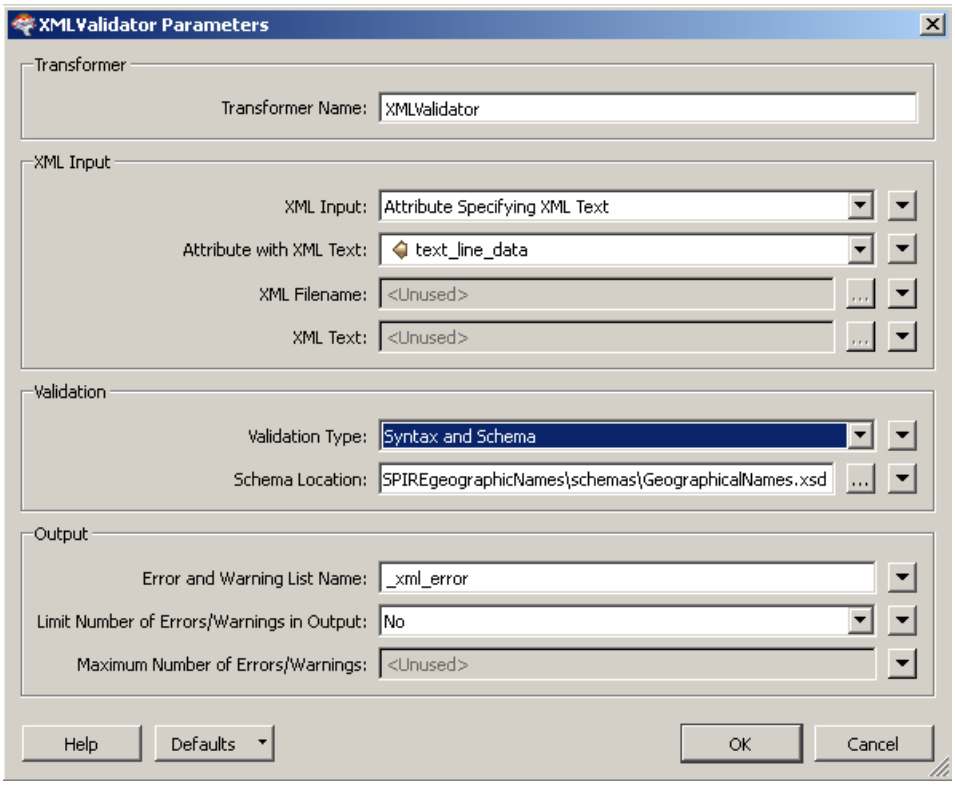
Set the output xml fields to be '_xml_dataset'

6. Add an XMLFormatter. Format the '_xml_dataset' field and set the output field to be 'text_line_data'. Also turn on clean-up of redundant namespace declarations.

7. Add an XMLValidator and validate the text_line_data field for syntax and application schema. Set the application schema path to:

   C:\FMEData\VARtraining\source\XML\INSPIRE.schemas\GeographicalNames.xsd

   The XMLValidator should look as follows once configured correctly:

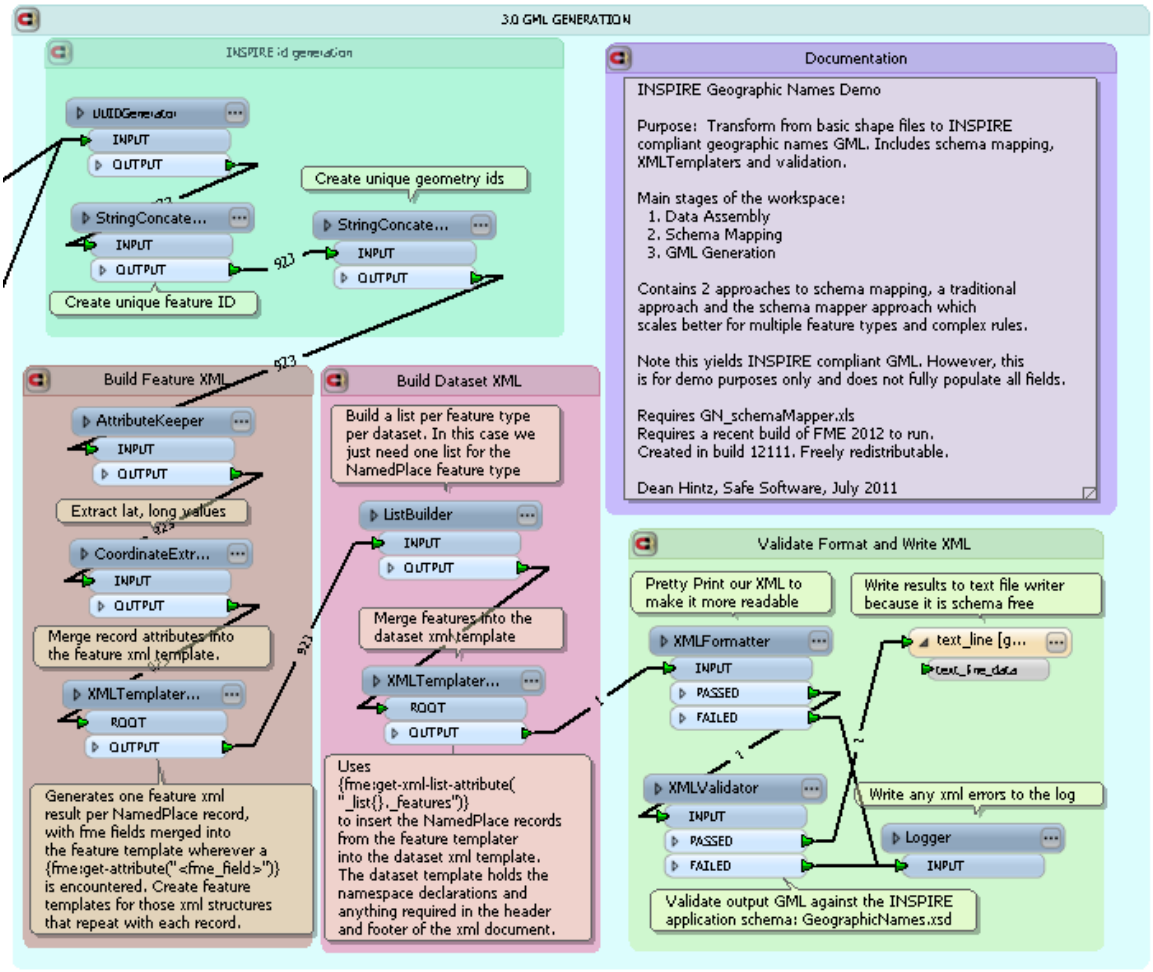


8. Add a text file writer and set the output path to be:

   C:\FMEData\VARtraining\result\XML\GeographicNames.gml

9. Save and run your workspace. Confirm that your XML validates both for syntax and against the INSPIRE application schemas. Examine the result in a text editor.

The GML Generation bookmark of your final workspace should look something like:

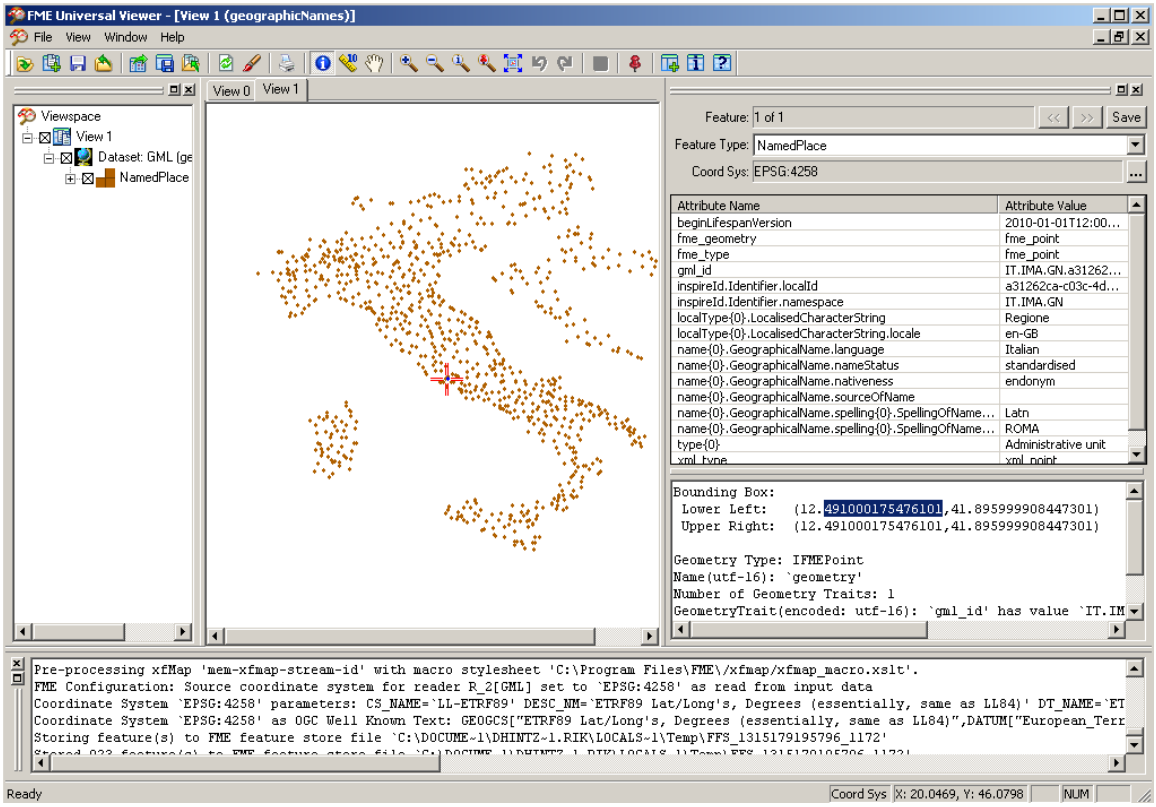


Finally, confirm that you have created valid INSPIRE GML by reading it with the GML reader. As of FME 2012 there is no need to manually specify the INSPIRE application schemas. FME ships with these and automatically recognizes and reads INSPIRE data.

Open in FME viewer using the GML reader:

C:\FMEData\VARtraining\result\XML\GeographicNames.gml

The result should look something like:



Congratulations. You have now:

- Generated INSPIRE compliant Geographic Names GML 3.2.1 data from UN gazetteer data using XML templates and verified success by reading it back in using the GML reader.

XML Transformers

- **XMLFeatureMapper** –make features with xfmap
- **XMLFormatter** – clean up XML formatting
- **XMLFragmenter** – extract XML fragments and generate features and attributes
- **XMLFlattener –** flatten XML into relational attributes
- **XMLTemplater** – template + FME values = XML
- **XMLValidator** – check syntax and schema
- **XMLUpdater –** update XML element values without changing the structure
- **XQueryExtractor** – extract XML elements
- **XQueryExploder** – create new features
- **XQueryUpdater** – update XML (replace values)
- **XSLTProcessor** – Process XML with xslt script
- **XMLNameSpaceDeclarer** – Add required namespaces

GeometryExtractor - essential for generating GML 3.2.1 geometry xml.

Formating and Validation: XMLFormatter and XMLValidator

- Good to format the XML before output.
- Makes it easier to review in case some of our concatenation processes interfere with white space
- Remove redundant namespace declarations, unneeded xsi:schemaLocations
- Check the XML for validity with the XMLValidator. Two settings:
    - Syntax Only
    - Syntax and Application schema (need xsd)

XMLSampleGenerator

- addresses last major gap in our XML workflow.
- Scan XSD and create sample XML templates based on user selection for node etc
- This can be used to serve as an XML feature or dataset template when you only have the schema xsd and no source XML data
- Very useful for INSPIRE where little data previously exists. It can be difficult to manually ‘read’ the xsd and manually create XML that is compliant (try it sometime!)