

Article

← [FME Desktop \(/S/Topic/0TO4Q000000QL9uWAG/Fme-...\)](#)

How to Consume and Produce XML using Application Schemas with the XSD-Driven XML Writer

🕒 Aug 3, 2022 • Knowledge

Product Type

FME Desktop

FME Version

2022.0

Tutorial: [Tutorial: Getting Started with XML \(/s/article/tutorial-getting-started-with-xml\)](/s/article/tutorial-getting-started-with-xml) | **Previous:** [XML Writing with XMLTemplater \(/s/article/xml-writing-with-xmltemplater\)](/s/article/xml-writing-with-xmltemplater) | **Next:** [GML Reading \(/s/article/reading-xmlgml\)](/s/article/reading-xmlgml)

Introduction

In 2020, FME released a new XSD-Driven XML reader and writer. If you are new to XML, this [tutorial series \(https://community.safe.com/s/article/tutorial-getting-started-with-xml\)](https://community.safe.com/s/article/tutorial-getting-started-with-xml) describes how to read, process, and write XML in FME. XML Schema Definition, commonly abbreviated as XSD, was created to standardize the way elements are described in XML documents.

As you may be familiar, Extensible Markup Language or XML can be extremely versatile, making it appealing to many applications. However, XML can become complex, especially if documents are highly nested. XSD is one - of multiple - adopted method(s) for combatting this issue.

XSD was introduced in 2001, after being acknowledged by the World Wide Web Consortium (W3C) for being a beneficial way of structuring, defining, and restricting elements. Uniquely, in FME, XSD-Driven XML Reading/Writing gives you the advantage of interpreting XML elements as FME features.

Where is XSD-Driven XML Used?

XSD-Driven XML is beneficial if there is an application schema available, and the user wants to preserve that schema.

When FME accesses the XSD, XML elements become schema-driven, as opposed to data-driven. Though the XML reader is frequently used to convert XML to GIS, the reader itself can't support complex schemas (no schema support). Similarly, before, complex writing XML used to require the use of an XMLTemplater, but now XSD-Driven XML can solely comply with the respected schema. Note that for GML, it is usually preferable to use the GML reader/writer unless there is some aspect of the schema that is not supported, in which case XML XSD might help (such as for unsupported element or geometry types).

Format Similarities:

Format Differences:

<ul style="list-style-type: none"> • XSD is written with XML • XSD and XML are both object-oriented (flat schema potential) • Both formats satisfy to W3C recommendations • Both are human and machine-readable • Store a variety of structured information 	<ul style="list-style-type: none"> • XSD defines elements and structures, where XML describes data • XSD focuses on XML interpretation • XSD can be used for XML validation • XSD can restrict node values • XSD can be used to create custom data types • XML generally requires more configuration if an XSD is not present.
--	--

XSD Driven XML Writing Demo

The purpose of this exercise is to demonstrate writing XML using the new XSD Driven XML writer. In this scenario, we are creating a natural disaster alert.

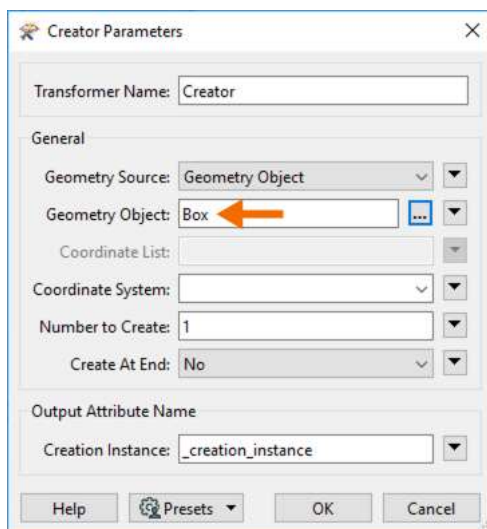
Step-by-step Instructions

1. Start FME Workbench

Open FME Workbench on your machine and open a blank workspace. Note if you are short on time, open the XSD-DrivenWriter-Begin.fmw then skip to step 4. This starting workspace has the demo data already created.

2. Add a Creator

Add a Creator transformer to the canvas. In the parameters set Geometry Object to Box. Leave all other parameters as default.



3. Add an AttributeCreator

Now we are going to create information content. Add an AttributeCreator, and connect it to the Creator. Double click the transformer to access the parameters. In the parameters, add the following attributes and corresponding values:

New Attribute:	Attribute Value:
.....	

language	en-US
category{0}	Safety
responseType{0}	Monitor
event	Flood
urgency	Expected
severity	Moderate
description	Test Message: Monitor flood levels on the Fraser River near Fort Langley.
certainty	Observed
contact	lizard@safe.com (mailto:lizard@safe.com).

AttributeCreator Parameters

Transformer Name:

> Advanced: Attribute Value Handling

Attributes To Create

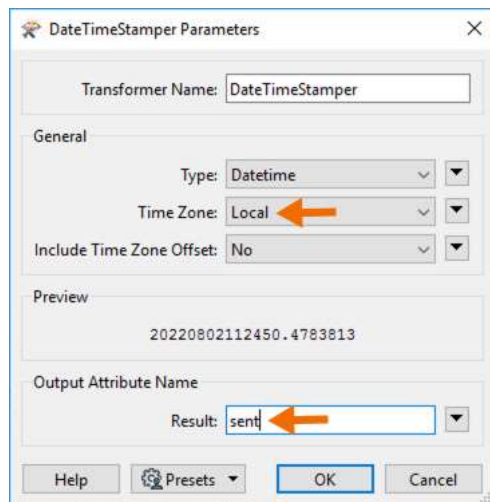
New Attribute	Attribute Value
language	<input type="checkbox"/> en-US
category{0}	<input type="checkbox"/> Safety
responseType{0}	<input type="checkbox"/> Monitor
event	<input type="checkbox"/> flood
urgency	<input type="checkbox"/> Expected
severity	<input type="checkbox"/> Moderate
description	<input type="checkbox"/> Test Message:... (MultiLine)
certainty	<input type="checkbox"/> Observed
contact	<input type="checkbox"/> lizard@safe.com

Filter Import...

Help Presets OK Cancel

4. Add a DateTimeStamper

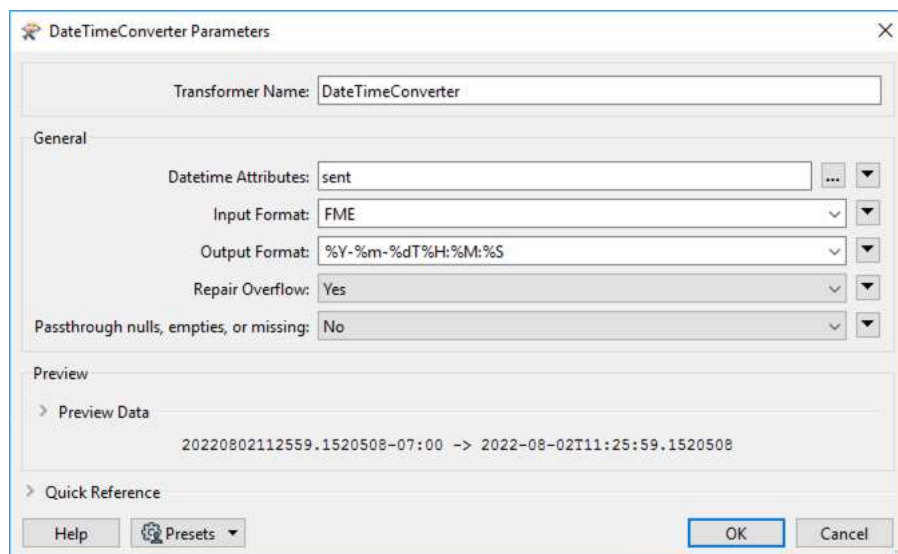
Next, add a DateTimeStamper and connect it to the AttributeCreator. In the parameters, change the Time Zone to Local and the Output Attribute Name to sent.



The **DateTimeStamper Parameters** dialog box is shown. The **Transformer Name** is `DateTimeStamper`. Under the **General** tab, **Type** is `Datetime`, **Time Zone** is `Local` (indicated by an orange arrow), and **Include Time Zone Offset** is `No`. The **Preview** section shows the timestamp `20220802112450.4783813`. The **Output Attribute Name** section shows **Result** is `sent` (indicated by an orange arrow). At the bottom are **Help**, **Presets**, **OK**, and **Cancel** buttons.

4. Convert to ISO Datetime Format

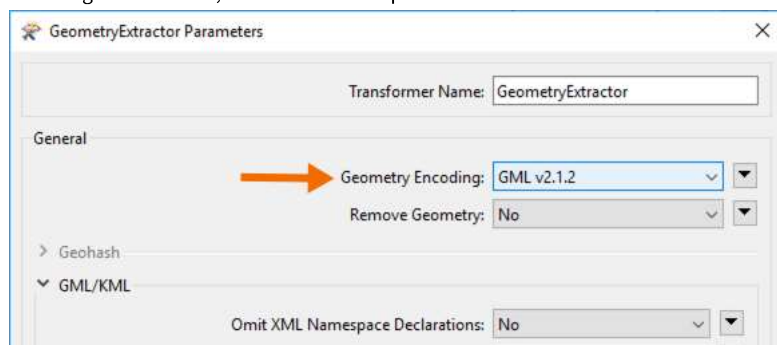
For the XML document to be validated, the datetime value needs to conform with the ISO datetime format. Add a `DateTimeConverter` and connect it to the `DateTimeStamper`. In the parameters, set the `Datetime Attributes` to `sent`. Then for `Input Format` select `FME`, and for `Output Format` select `%Y-%m-%dT%H:%M:%S` (ISO datetime).



The **DateTimeConverter Parameters** dialog box is shown. The **Transformer Name** is `DateTimeConverter`. Under the **General** tab, **Datetime Attributes** is `sent`, **Input Format** is `FME`, **Output Format** is `%Y-%m-%dT%H:%M:%S`, **Repair Overflow** is `Yes`, and **Passthrough nulls, empties, or missing** is `No`. The **Preview** section shows the conversion: `20220802112559.1520508-07:00 -> 2022-08-02T11:25:59.1520508`. At the bottom are **Help**, **Presets**, **OK**, and **Cancel** buttons.

5. Add a GeometryExtractor

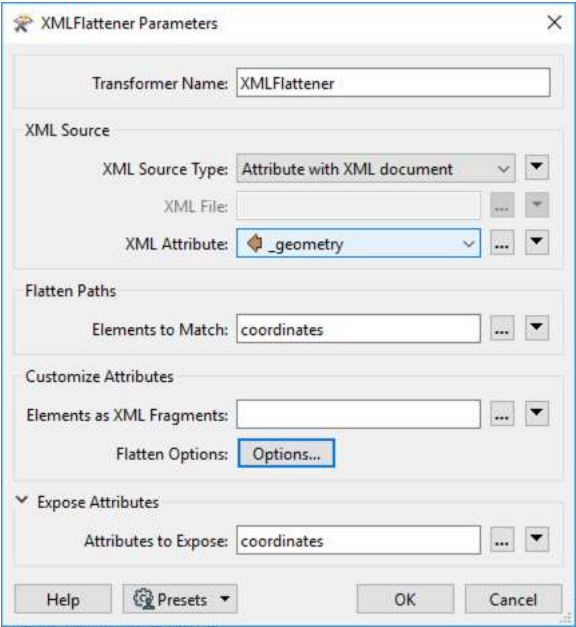
One limitation with the XSD XML reader / writer is that there is no geometry support out of the box - any geometry handling needs to be managed as xml fragments. To this end, we are going to start generating geometry XML using the `GeometryExtractor`. We use `GML 2.1.2` since it uses an XML coordinate syntax similar to what is required for this application schema. Add a `GeometryExtractor` and connect it to the `DateTimeConverter`. In the parameters, set the `Geometry Encoding` to `GML v2.1.2`; there are no other parameters to set.



The **GeometryExtractor Parameters** dialog box is shown. The **Transformer Name** is `GeometryExtractor`. Under the **General** tab, **Geometry Encoding** is `GML v2.1.2` (indicated by an orange arrow), and **Remove Geometry** is `No`. The **Geohash** section is collapsed. The **GML/KML** section shows **Omit XML Namespace Declarations** is `No`. At the bottom are **Help**, **Presets**, **OK**, and **Cancel** buttons.

6. Flatten XML

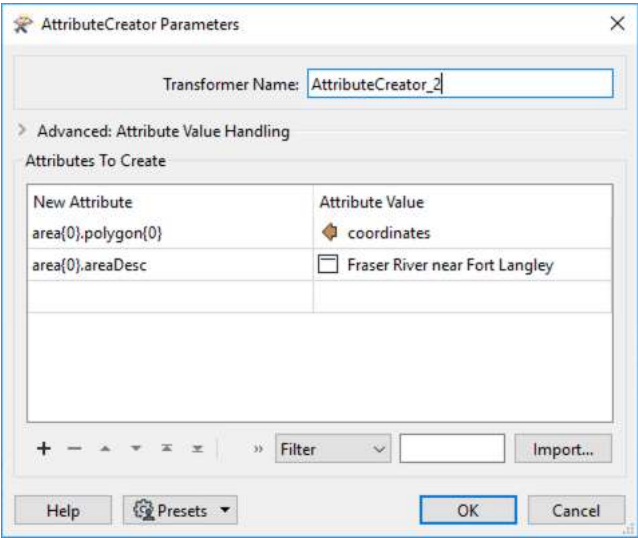
We are going to use an XMLFlattener to flatten the content of XML element(s) into feature attributes. Add an XMLFlattener to the canvas and connect it to the GeometryExtractor. In the parameters, set the XML Source Type to Attribute with XML document, then select `_geometry` as the XML Attribute. For Elements to Match, type in `coordinates`. Expand Expose Attributes, then for Attribute to Expose type in `coordinates`. Make sure to set the XML Source Type as Attribute with XML document, and the XML Attribute to `'_geometry'` (the Destination Geometry Attribute we created in the GeometryExtractor).



7. Add Area Properties

Add another AttributeCreator to the canvas and connect it to the XMLFlattener. We will use this transformer to add area properties to our emergency alert. In the parameters, we will define two new attributes to store area coordinates and description.

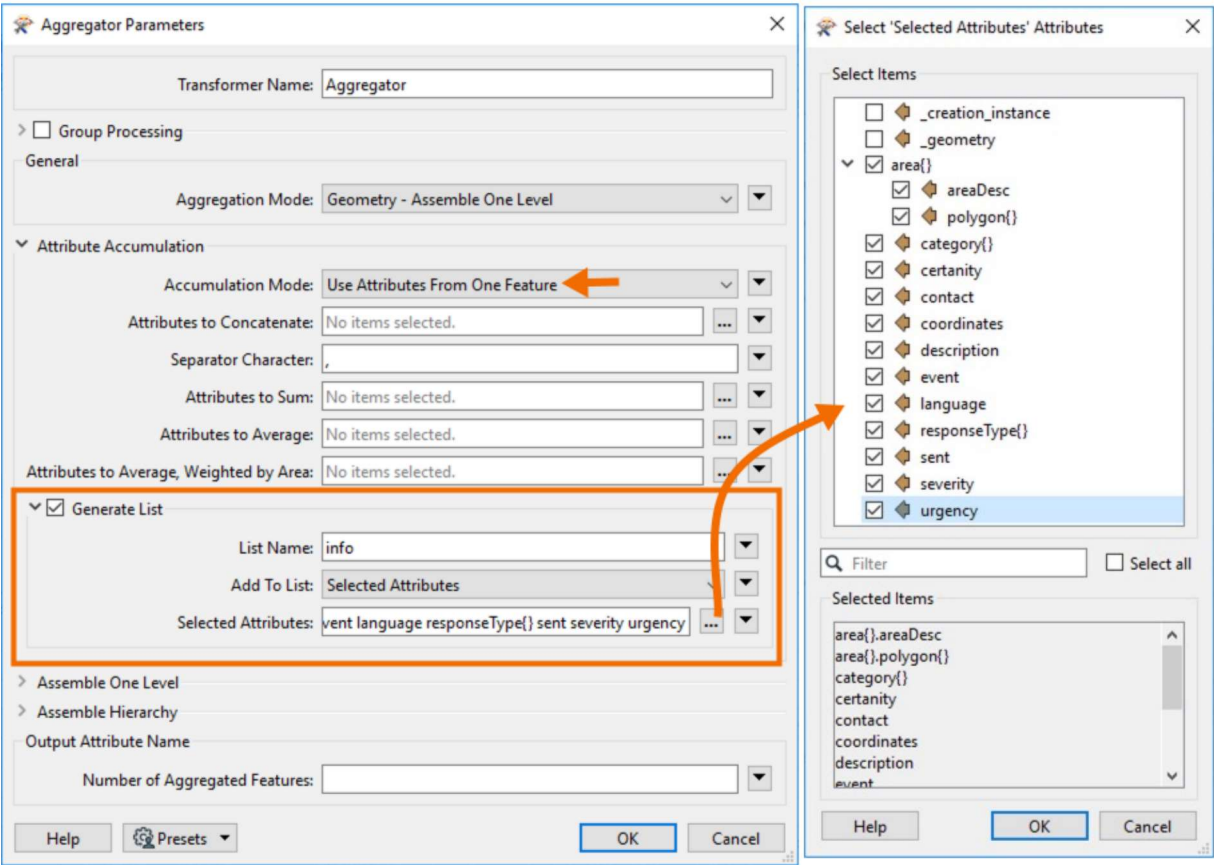
New Attribute:	Attribute Value:
area{0}.polygon{0}	coordinates (attribute)
area{0}.areaDesc	Fraser River near Fort Langley



8. Add an Aggregator

Next, we will need to generate lists for the nested elements in the document. Add an Aggregator to the canvas and connect it to the AttributeCreator_2. In the parameters, set the Aggregation Mode to Geometry - Assemble One Level. Next, set the Accumulation Mode to Use Attributes From One Feature. Lastly,

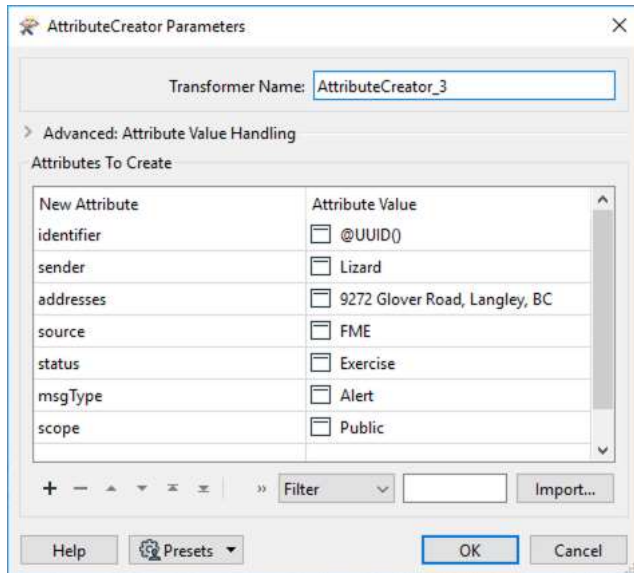
we are going to generate a list. Enable Generate List by clicking the checkbox, then set the List Name to info. Click on the ellipsis for Selected Attributes and click Select all, then un-select _creation_instance and _geometry.



9. Define Root Alert Content

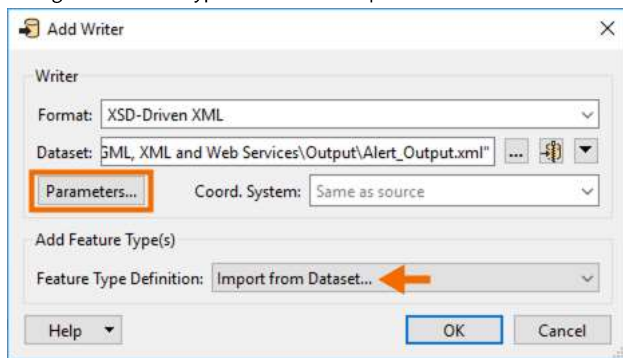
Before writing out, add a final AttributeCreator to define the root alert content. Use the table below to add the alert property values (feel free to have fun with the values):

New Attribute:	Attribute Value:
identifier	@UUID()
sender	Lizard
addresses	9272 Glover Road, Langley, BC
source	FME
status	Exercise
msgType	Alert
scope	Public

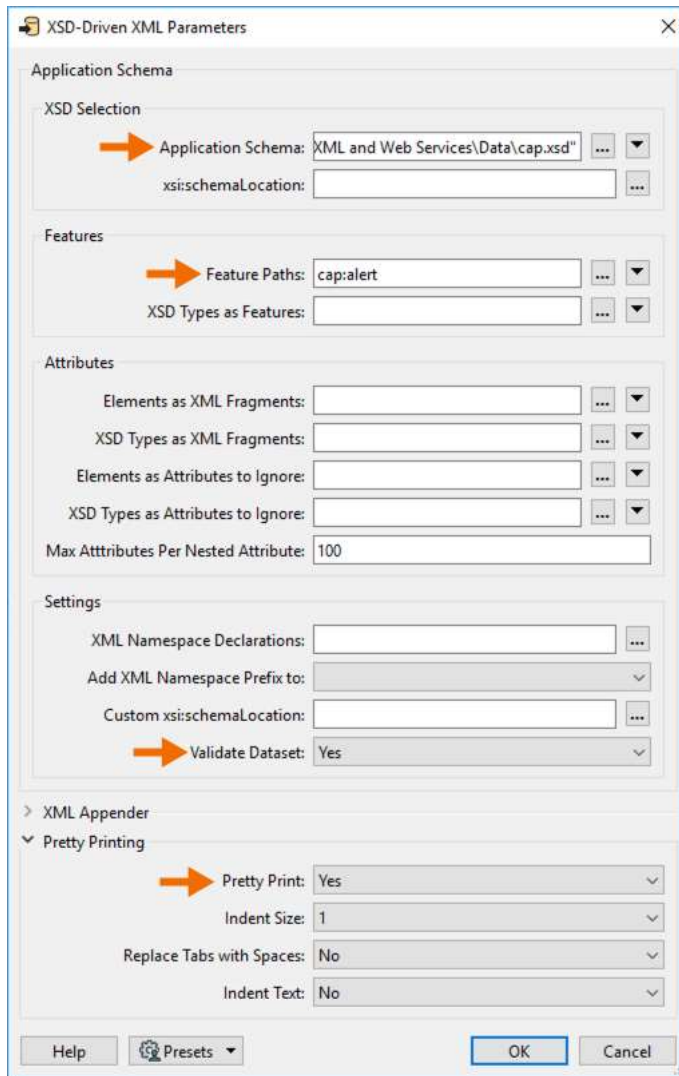


10. Add an XSD-Driven XML Writer

Now the data is ready to be written out. Add an XSD-Driven XML writer to the canvas, and browse to an output folder. Name the dataset Alert_Output.xml, then change the Feature Type Definition to Import from Dataset. Before clicking OK, open the Parameters.

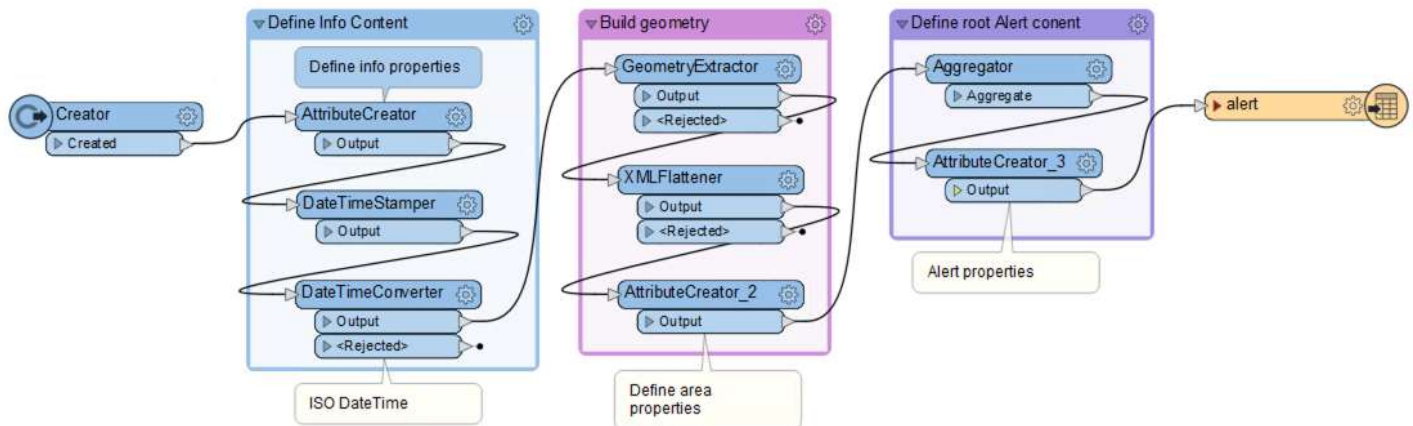


In the parameters, set the Application Schema to ...\\cap.xsd. Then for Feature Paths, click on the ellipsis and select cap:alert. This is also a beneficial time to validate the XML output. Set Validate Dataset to Yes. Also, enable Pretty Print to Yes. Click OK twice.



In the Import Writer Feature Types dialog, browse to the cap.xsd dataset, then click OK. You may need to switch the file type to All Files (*) in your file browser to see the cap.xsd file.

Connect the alert writer feature type to the AttributeCreator_3.



11. Run and Inspect

Run the workspace then input the output using Notepad++ or similar text editor. Another method of validating the output is by reading it back in using FME Data Inspector.


```
<?xml version="1.0" encoding="UTF-8"?>
<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <cap:identifier>13b0fcc4-fe46-4b9c-9aa3-09ad924bb00c</cap:identifier>
  <cap:sender>Lizard</cap:sender>
  <cap:sent>2021-02-22T13:52:41.7695242</cap:sent>
  <cap:status>Exercise</cap:status>
  <cap:msgType>Alert</cap:msgType>
  <cap:source>FME</cap:source>
  <cap:scope>Public</cap:scope>
  <cap:addresses>9272 Glover Road, Langley, BC</cap:addresses>
  <cap:info>
    <cap:language>en-US</cap:language>
    <cap:category>Safety</cap:category>
    <cap:event>flood</cap:event>
    <cap:responseType>Monitor</cap:responseType>
    <cap:urgency>Expected</cap:urgency>
    <cap:severity>Moderate</cap:severity>
    <cap:certainty>Observed</cap:certainty>
    <cap:description>Test Message:
Monitor flood levels on the Fraser River near Fort Langley.</cap:description>
    <cap:contact>lizard@safe.com</cap:contact>
    <cap:area>
      <cap:areaDesc>Fraser River near Fort Langley</cap:areaDesc>
      <cap:polygon>-122.63,49.16 -122.51,49.16 -122.51,49.21 -122.63,49.21 -122.63,49.16</cap:polygon>
    </cap:area>
  </cap:info>
</cap:alert>
```

Validation

The purpose of validation is to ensure documents comply with XML syntax and XSD standards. Typically, to validate XML files in FME, the process is enabled in the writer (Validate Dataset: Yes), or include an XMLValidator transformer in your workspace. If the XML is not valid, the log errors will be briefly reported, including details such as the line, column number, and a brief description of the error. Common errors include missing IDs, incorrectly formatted date fields, missing elements, and namespace elements. Remember, for both reading and writing XML, element order matters!

Unfamiliar with XML error notation? XML error messages come from the Apache Xerces parsing library (open source). Let's understand the partial example below:

```
ERROR | ... for content model '(identifier,name,phoneNum?,email?,gender+,references?,note*)
```

Elements in the model list that have no special trailing notation are 'identifier' and 'name' - these are required properties. Optional items in the content model are followed by a '?', like phoneNum and email. The remaining notations denote multiples (*), lists (*) or restricted domain (picklist) (+) data types. For example, gender could have the following picklist values: Female, Male or Other.

The next examples are specific to XML validation. The error below will appear if you are missing an element.

```
ERROR |XML Validation: Error in '6. EmergencyAlters_CAP1.2 XML\capOut.xml' on line 26, column 13: 'element
'certainty' is not allowed for content model '(language?,
category+,event,responseType*,urgency,severity,certainty,audience?,eventCode*,effective?,onset?,expires?,senderName?,
headline?,description?,instruction?,web?,contact?,parameter*, resource*,area*)
```

This message may seem misleading at first. It sounds like the element 'certainty' is not allowed. This is not quite accurate. What it really means is that in this case 'certainty' is not allowed in that order, because a required element that precedes it is missing. In this case, the severity property is missing and so when the parser encounters the certainty element it flags this warning.

We see a similar problem below:

```
ERROR |XML Validation: Error in '6. EmergencyAlters_CAP1.2 XML\capOut.xml' on line 27, column 13: 'element 'source'
is not allowed for content model
'(identifier,sender,status,msgType,source?,scope,restriction?,addresses?,code*,note?,reference?,incidents?,info*)
```

Here, 'msgType' is missing and so when the validator encounters this 'source' element the above warning is generated.

Like mentioned earlier, XML can be highly nested, which means both parent and child elements need to be included to access information throughout the document. Often this is modeled by the presence of parent and child XML ids such as xml_parent_id and xml_id. It is important to check the structure of your output to make sure it is correct. You may have orphaned features and depending on the schema this may or may not generate a warning. However, when you look at the XML output, you may notice features at the root level that really belong within a parent.

One good way to understand this better for a given schema is to find a valid sample XML dataset and read this with FME and send some features to the log with a Logger. The resulting features will show the parent/child ids that result from FME serializing the nested structure into FME features. Once your XSD-Driven XML document is successfully validated, turning off validation can improve writing performance.

Considerations

After completing this tutorial you will have successfully written a disaster alert using CAP1.1 XSD. As you just witnessed, no XMLTemplator or XMLValidator was required in our workspace. The XSD-Driven XML format reader/writer can be used to optimize workflows and replaces a more manual approach to working with XML schemas.

Continue to the next article: [GML Reading \(/s/article/reading-xmlgml\)](/s/article/reading-xmlgml)

First Published Date

2/24/2021, 6:32 PM

Last Published Date

8/3/2022, 9:46 PM


Formats and Applicatio...
(/s/topic/0TO4Q000000...)

FME Desktop
(/s/topic/0TO4Q000000...)

Sort by:

Latest Posts ▾




 [LizAtSafe \(/s/profile/0050c00000CeGV0AAN\)](/s/profile/0050c00000CeGV0AAN) (Employee) published a new version of this Knowledge.
[August 3, 2022 at 9:46 PM \(/s/feed/0D54Q000009ggOyYSAU\)](/s/feed/0D54Q000009ggOyYSAU)

1 view

 Like  Comment

[Log In to Comment](#)

 [trentatsafe \(/s/profile/005a000000CdnWnAAJ\)](/s/profile/005a000000CdnWnAAJ) (Employee) published a new version of this Knowledge.
[July 29, 2022 at 2:26 PM \(/s/feed/0D54Q000009fwaUDSAY\)](/s/feed/0D54Q000009fwaUDSAY)

 Like  Comment

[Log In to Comment](#)

Follow

[Files \(1\) \(/s/relatedlist/ka14Q000001DWxyQAG/AttachedContentDocuments\)](/s/relatedlist/ka14Q000001DWxyQAG/AttachedContentDocuments)



2b. Using the XSD Driven XML Writer

Aug 3, 2022 • 65KB • zip

[View All](#)

Related Articles

Converting from XML (Simple XML Reading Example) (/s/article/tutorial-getting-started-with-xml-reading)



[Register / Log In \(/s/login/\).](/s/login/)



Safe Software respectfully acknowledges that we live, learn and work on the traditional and unceded territories of the Kwantlen, Katzie, and Semiahmoo First Nations.