FME Form Modules 6-7

Exercise Workbook

info@misoportal.com

0121 232 8000

Contents

6 Advanced Attribute Handling and Lists

6.1 Constructing Attribute Values

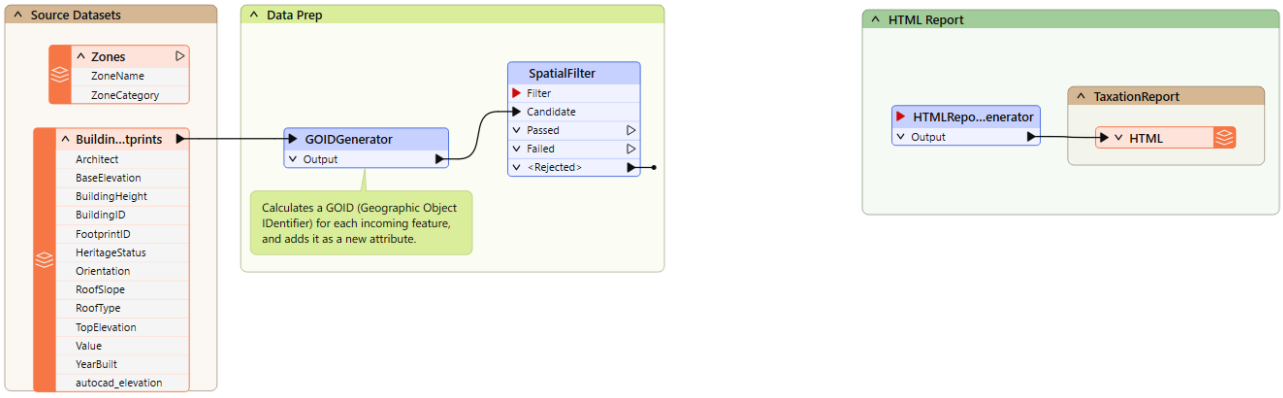
Demonstrates	Constructing values with the Arithmetic Editor and Text Editor. Use of the AttributeValueMapper transformer to define values. Use of the DateTimeFormat Function to format dates. Generate an HTML report and write it to an output file.
Overall Goal	Create a HTML taxation report for buildings in the city by constructing attributes using the arithmetic and text editors.
Data	Building Footprints (AutoCAD DWG), Zoning Data (MapInfo TAB)
Start Workspace	C:\FMEModularData\Workspaces\6.01-AdvancedAttributes-ConstructingValues-Begin.fmw
End Workspace	C:\FMEModularData\Workspaces\Complete\6.01-AdvancedAttributes-ConstructingValues-Complete.fmw

The annual property tax reports are due to be calculated, and it is decided to use FME to carry out the processing. A partial workspace already exists; you must finish it by calculating tax values for each building (based on size and zoning) and creating a HTML report of the results.

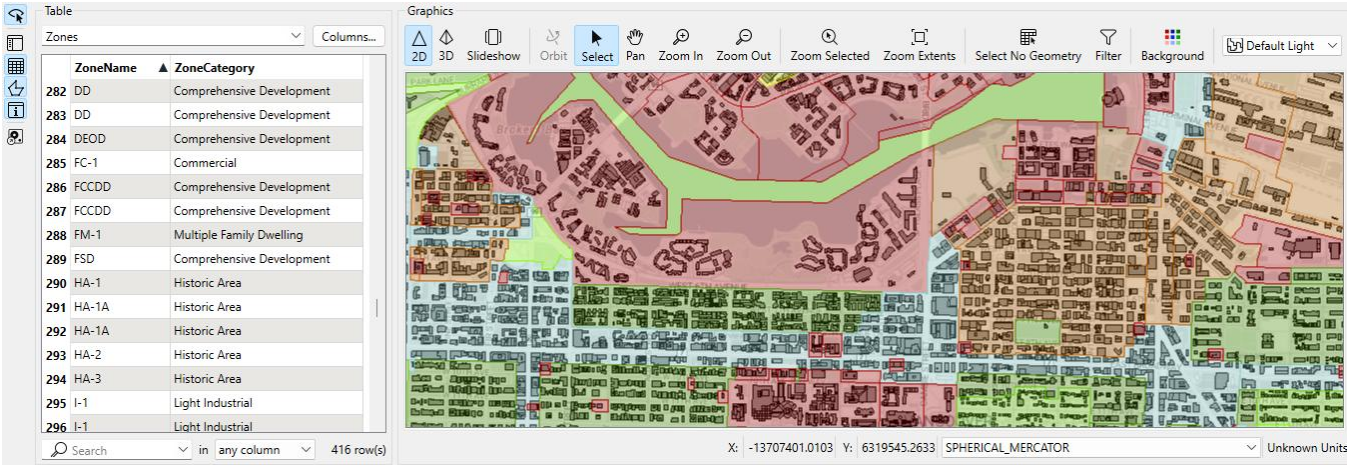
6.1.1 Open Workspace and Inspect the Source Datasets

A workspace already exists with the first part of our required workflow already setup.

Launch the FME Workbench, if it isn't open already. Within the Get Started section of the Workbench, click on Open Workspace. Navigate to and open C:\FMEModularData\Workspaces\6.01-AdvancedAttributes-ConstructingValues-Begin.fmw



Familiarize yourself with both source datasets, using the Visual Preview or Data Inspector tools.



Notice there are different zone categories, each will require a different tax multiplier to be applied:

Zone Category	Tax Multiplier
Comprehensive Development	0.9
Historic Area	1.1
Light Industrial	0.7
Multiple Family Dwelling	0.7
One Family Dwelling	1.2
Two Family Dwelling	1
Commercial	0.8
Industrial	0.5

We need to add a new *TaxMultiplier* attribute to the zones and populate it with the appropriate values. There are a couple of ways we could achieve this; manually defining the categories and values to apply (time consuming and cumbersome), using Conditional Values (we'll look at this technique in a later exercise), or using the AttributeValueMapper transformer.

6.1.2 Add an AttributeValueMapper transformer

The AttributeValueMapper compares attribute values to a lookup table and assigns new values where matches are found. Mapped values may be stored in a new attribute, or overwrite an existing attribute.

Add an AttributeValueMapper transformer and connect between the *Zones* feature type and the *SpatialFilter* Filter port.

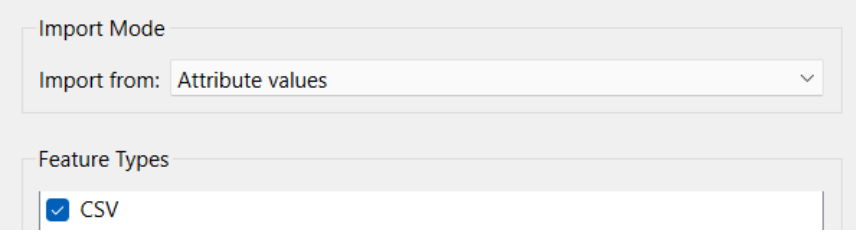
Within the transformer's parameters, set the Input Attribute to *ZoneCategory* and the Output Attribute to *TaxMultiplier.*

We could then manually define the Value Map Input and Output values, but luckily for us

we already have a lookup defined within a csv file which we can simply import.

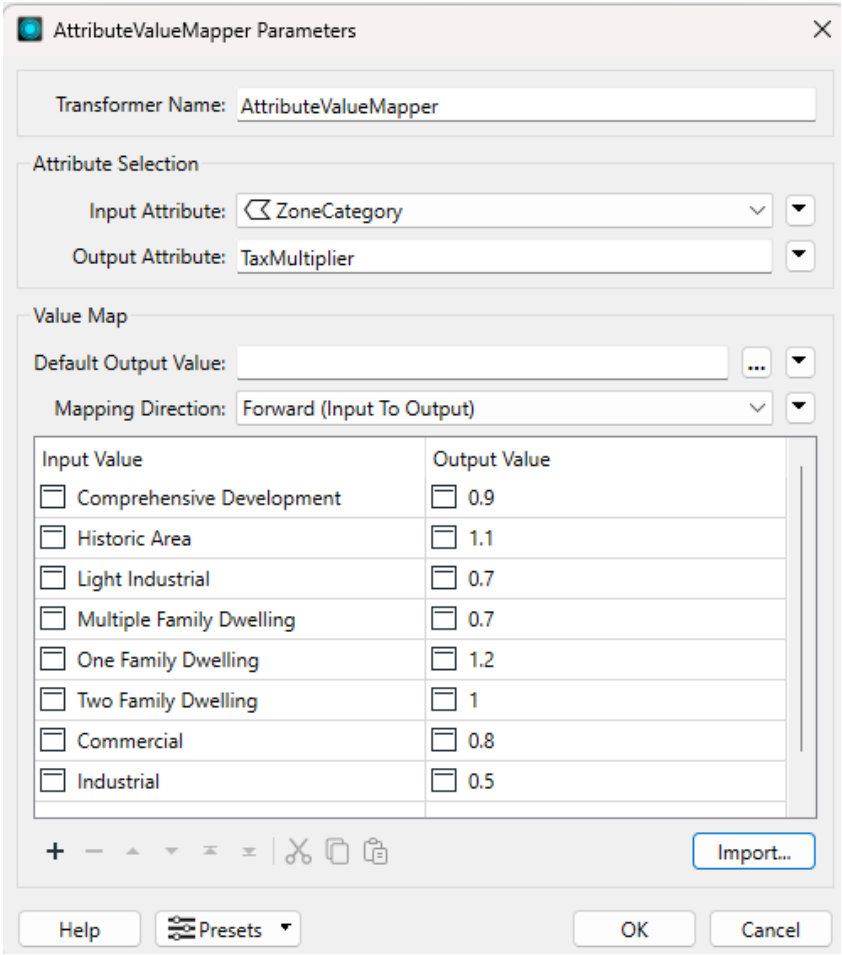
Click on the ***Import…* button**, set the Reader Format to *CSV (Comma Separated Value)*, then navigate to and select *C:\FMEModularData**Resources**\TaxationLookup.csv*

Click Next, then set the Import from: to *Attribute values*



Click Next, then set *Input Value*: to *zonetype* and Output Value to '*multiplier'*

Click Import.



We have now finished configuring the transformer parameters. Use *Run to This* on the

AttributeValueMapper, then examine the feature cache. The new TaxMuitplier attribute has been added and populated based on the zone category/multiplier lookup:

AttributeValueMapper_Output

	ZoneName	ZoneCategory	TaxMultiplier
283	DD	Comprehensive Development	0.9
284	DEOD	Comprehensive Development	0.9
285	FC-1	Commercial	0.8
286	FCCDD	Comprehensive Development	0.9
287	FCCDD	Comprehensive Development	0.9
288	FM-1	Multiple Family Dwelling	0.7
289	FSD	Comprehensive Development	0.9
290	HA-1	Historic Area	1.1

We now need to carry out the tax calculations for the buildings. For which will require the TaxMulitplier of the Zone the building falls within.

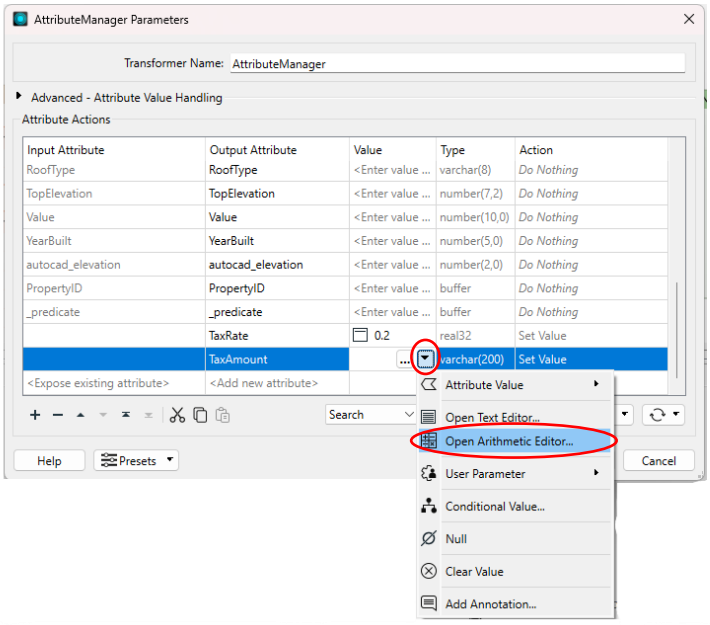
The SpatialFilter is already configured to perform the necessary spatial tests and merge attributes. Which importantly will obtain the TaxMultiplier value for each building footprint.

6.1.3 Add an AttributeManager transformer

To carry out our tax calculations, we'll use an AttributeManager transformer. So, place an AttributeManager and connect it to the SpatialFilter *Passed* output port.

The first task is to create a numeric value for the taxation rate. Create a new attribute called *TaxRate*. Set it to a fixed value of *0.2*

Add a second new attribute, this one called *TaxAmount*. Instead of setting a fixed value, click the drop-down arrow to the right and choose the option for Open Arithmetic Editor.



6.1.4 Calculate Tax Amount

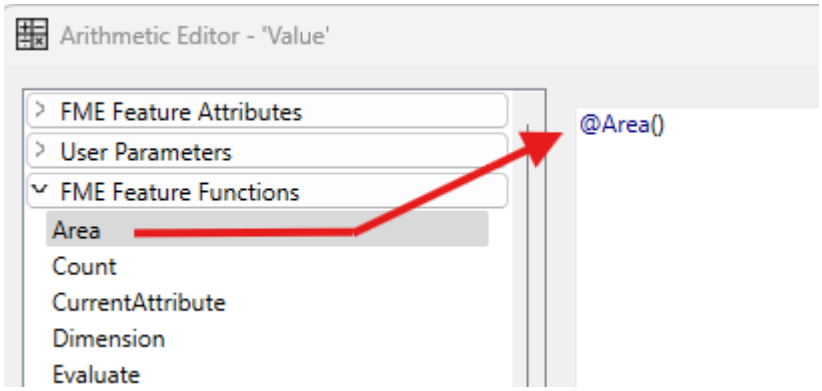
The calculation for the tax amount is:

Building Footprint x Tax Multiplier x Tax Rate = Tax Amount

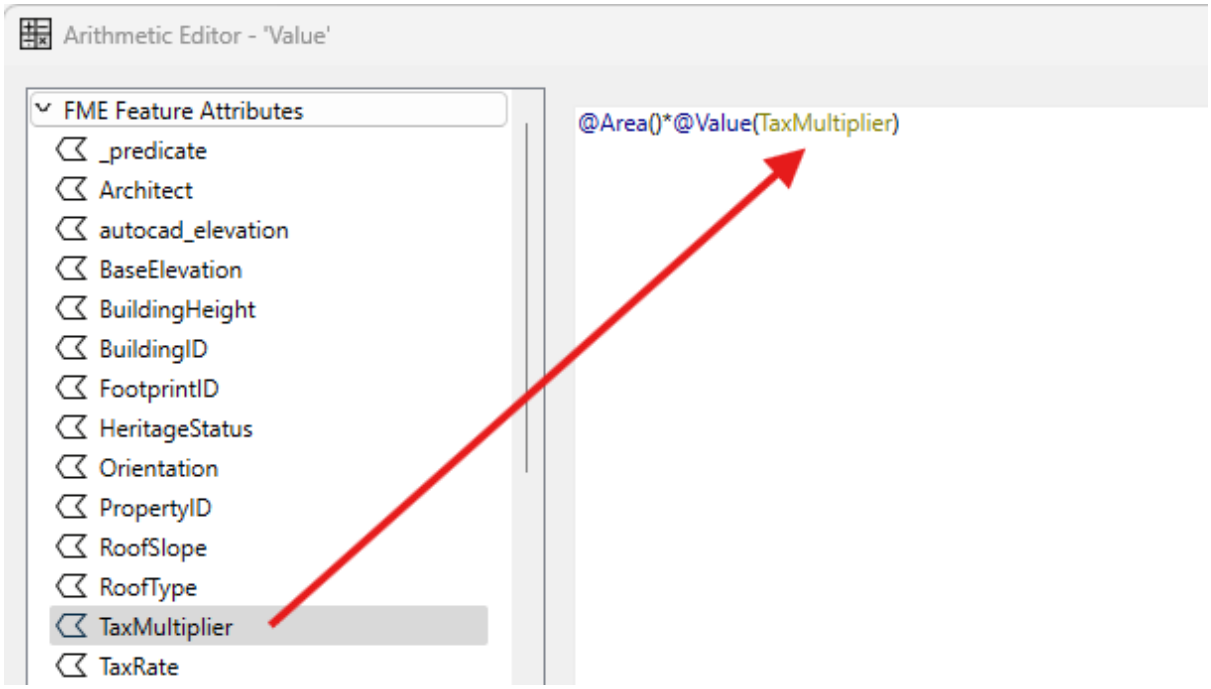
....where Building Footprint is the area of the building in square meters, Tax Multiplier is the value relating to the Zone Type, and Tax Rate is the fixed value we entered previously.

The result should also be rounded off to two decimal places.

So, start out by locating Area under FME Feature Functions and double-clicking it to add the area of the building footprint to the equation:

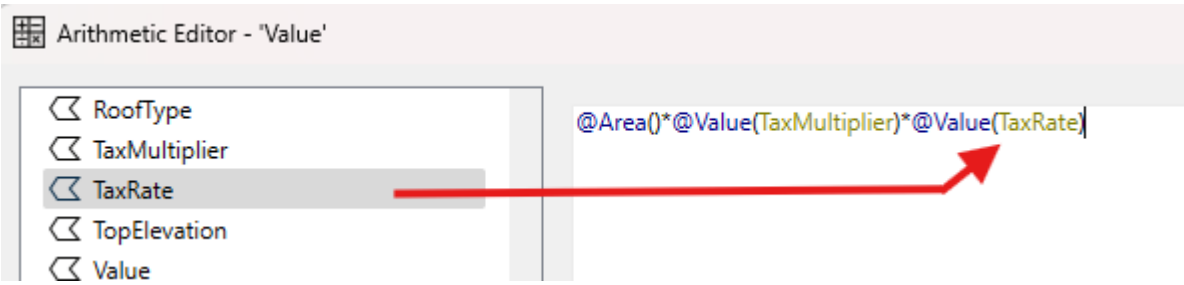


Add a multiplication symbol. It can be found under Math Operators, but it's easier to just type it. Next, locate the *TaxMultiplier* attribute under FME_Feature_Attributes and double-click it to add it to the equation:



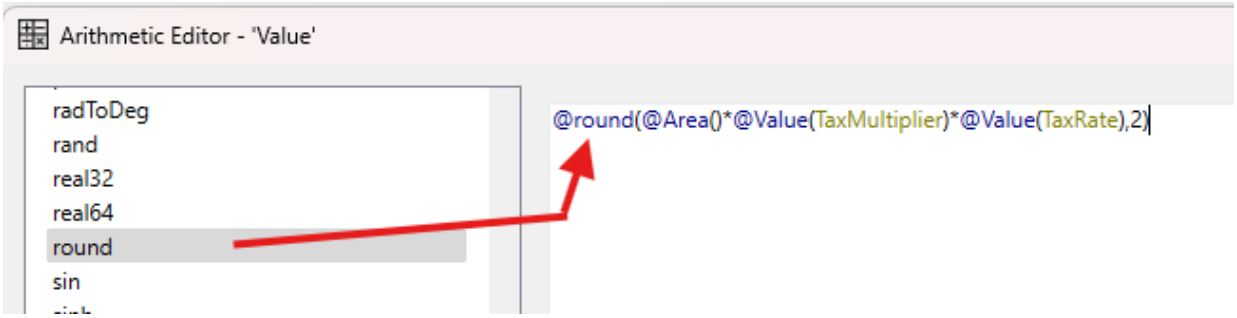
Add another multiplication symbol. Now locate the *TaxRate* attribute and double-click it

to add it to the equation:



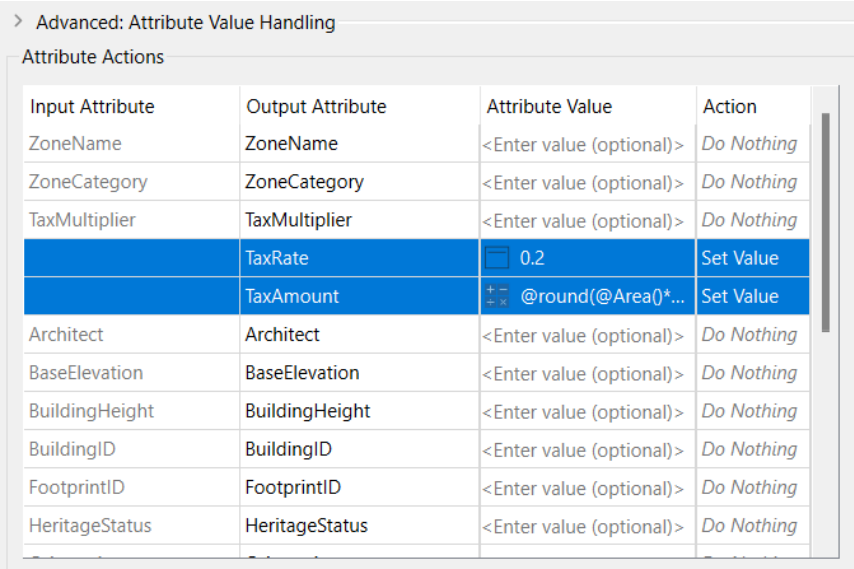
Finally, we need to round the result to two decimal places. Add the *round()* function to the equation. Like the multiplication sign, it can be found under the Math Function section but it is often easier to type manually

```
@round(@Area()*@Value(TaxMultiplier)*@Value(TaxRate),2)
```



Click OK to close this dialog.

Reorder the attributes to promote *TaxRate* and *TaxAmount* to higher in the order.



Run the translation and inspect the feature cache to ensure the results look correct.

Table

AttributeManager_Output

	ZoneName	ZoneCategory	TaxMultiplier	TaxRate	TaxAmount
20278	I-2	Light Industrial	0.7	0.2	141.94
20279	I-2	Light Industrial	0.7	0.2	13.52
20280	DEOD	Comprehensive Development	0.9	0.2	17.8
20281	DEOD	Comprehensive Development	0.9	0.2	8.28
20282	C-2B	Commercial	0.8	0.2	45.49
20283	C-2B	Commercial	0.8	0.2	7.66
20284	I-2	Light Industrial	0.7	0.2	129.92
20285	I-2	Light Industrial	0.7	0.2	106.32
20286	RT-8	Two Family Dwelling	1	0.2	14.07
20287	RT-8	Two Family Dwelling	1	0.2	10.68
20288	HA-2	Historic Area	1.1	0.2	8.14

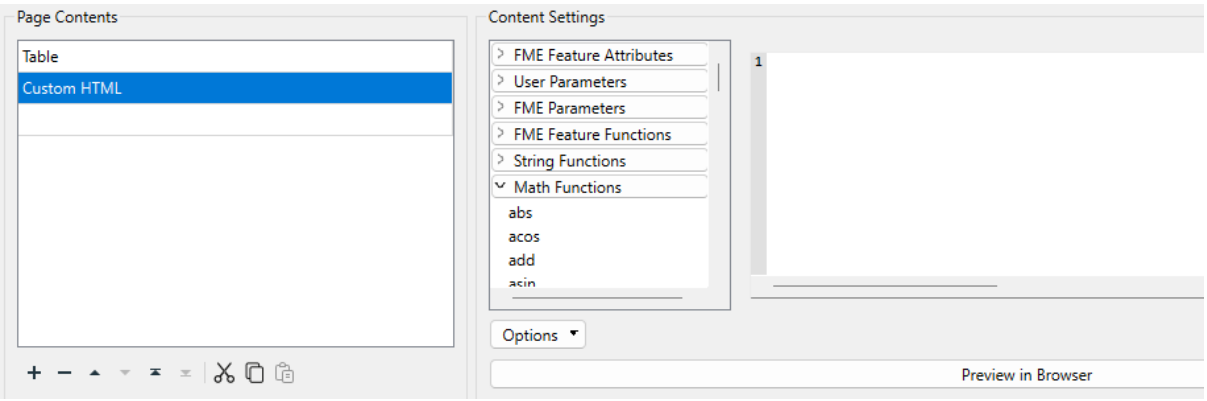
6.1.5 Generate Report

The final task is to create a report. The latter section of workspace already contains a partially edited HTMLReportGenerator transformer and HTML writer, so our task is simply to write a footer for the report.

Connect the AttributeManager to the HTMLReportGenerator's input port.

Open the parameters dialog for the HTMLReportGenerator. Under *Page Contents*, click below Table to create a new entry for *Custom HTML*:

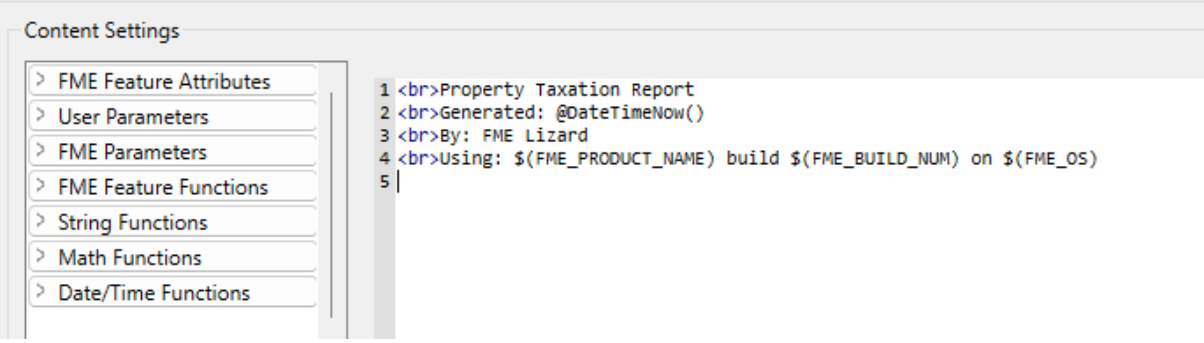
Then on the righthand side simply click anywhere and the view will then change and appear similar to this:



The right-hand side is now the Text Editor.

Using a combination of manual text entry, FME Feature Functions, and FME Parameters, enter the following text:

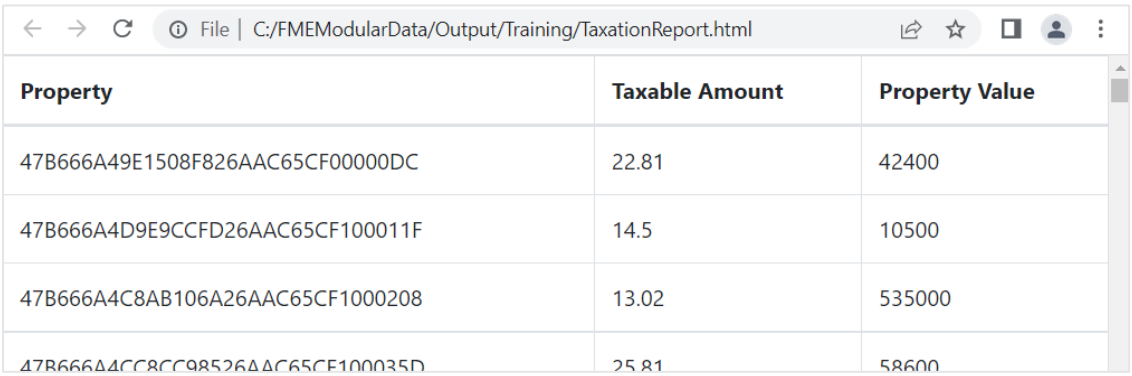
```
<br>Property Taxation Report
<br>Generated: @DateTimeNow()
<br>By: FME Lizard
<br>Using: $(FME_PRODUCT_NAME) build $(FME_BUILD_NUM) on $(FME_OS)
```



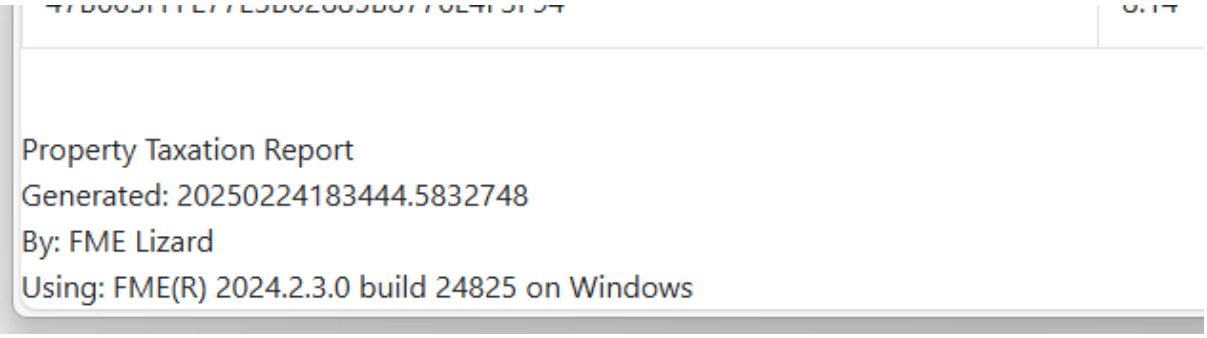
Note:
 is a HTML tag that represents the HTML equivalent of a newline character.

6.1.6 Run the Workspace and Examine Output

Run the workspace using *Run >Rerun Entire Workspace* (to ensure no incomplete caches are used). Open the HTML file that is created in a web browser. The report should look like this:



Scroll to the bottom to view properties:



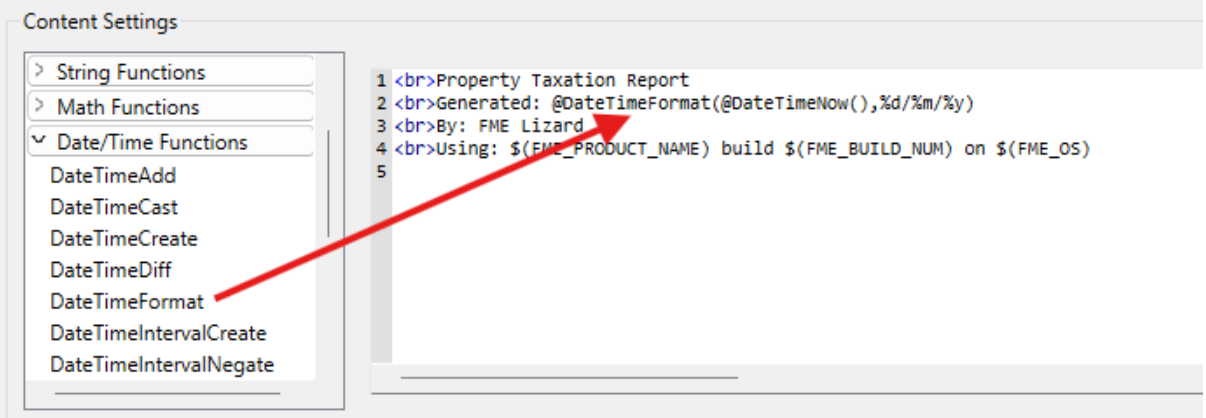
The *Generated* date stamp is there, but we need to format it.

6.1.7 Format Date

Return to the HTMLReportGenerator, then the Text Editor for *Custom HTML*

Use the *DateTimeFormat* function to set the format of the DateTimeNow()

```
<br>Generated: @DateTimeFormat(@DateTimeNow(),%d/%m/%y)
```



Rerun the workspace and examine the revised HTML output. The date should now be formatted as desired.

Congratulations

By Completing this exercise, you have learned how to:

- Construct numeric values with the Arithmetic Editor
- Construct strings with the Text Editor
- Use the AttributeValueMapper transformer to define values
- Use the DateTimeFormat Function to format dates
- Generate an HTML report and write it to an output file

6.2 Conditional Values

Demonstrates	Use of Conditional Attribute Values to map different attribute values based on tests/conditions.
Overall Goal	Use Conditional Attribute Values to assign the appropriate park category value based on meeting specific criteria
Data	Parks (MapInfo TAB)
Start Workspace	none
End Workspace	C:\FMEModularData\Workspaces\Complete\6.02-AdvancedAttributes-ConditionalValues-Complete.fmw

The Council's Environment Directorate want to categorize the city Parks into '*Premier*', '*Destination*' or '*Local*' - based on their features. We'll use Conditional Values to assign the appropriate park category value based on meeting specific criteria.

Conditional attribute values are great for when you need to map (or set) an attribute in relation to the value of an existing attribute, and when the conditions are more complex than can be handled in a simple AttributeValueMapper. In essence, conditional values are like a combination of TestFilter and AttributeCreators in the range of functionality that they include.

6.2.1 Launch FME Workbench and Create new Workspace from Blank

Launch the FME Workbench, if it isn't open already.
Within the Get Started section of the Workbench, click on Blank Workspace.

6.2.2 Add Reader

Use either the Reader button, or use Readers > Add Reader… from the menu bar.
The Add Reader dialog will open, in which define the Format and Dataset settings as follows:

Reader Format	Precisely MapInfo TAB (MAPINFO)
Reader Dataset	C:\FMEModularData\Data\Parks\Parks.tab

Into the Navigator panel a MapInfo Reader will be added and on the canvas there will be a single reader feature type for Parks.

6.2.3 Examine the Parks Data

Before we start designing our workflow, first examine the parks data within a Table view. This will display the Parks attributes and attribute values.

Table

Parks

	ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures
1	1	-9999	<missing>	Kitsilano	9406	10	N	<missing>	<missing>
2	2	208	Rosemary Brown Park	Kitsilano	13100	8	N	N	N
3	3	141	Tea Swamp Park	Mount Pleasant	11275	2	N	N	N
4	4	-9999	<missing>	Strathcona	9755	6	N	<missing>	<missing>
5	5	202	Morton Park	West End	14977	4	N	N	N

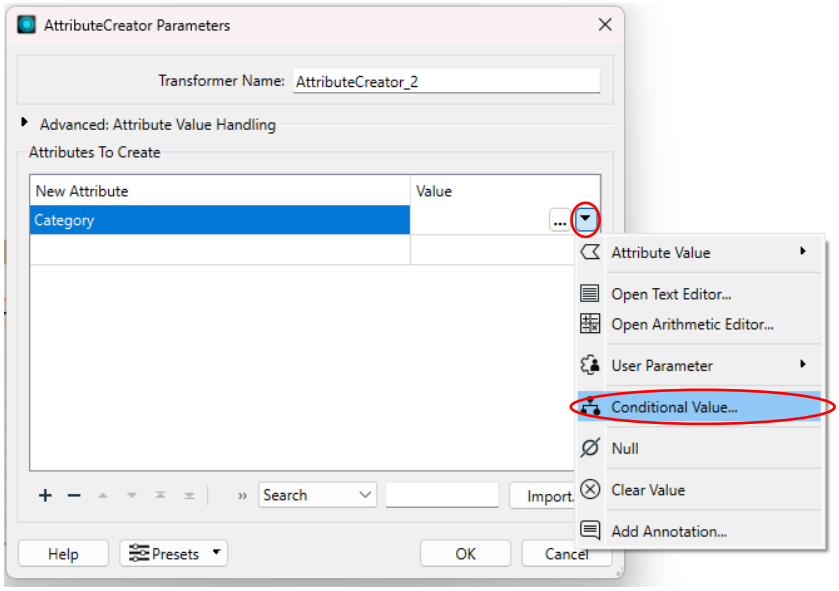
The parks need to be assigned a category value based on the following criteria:

Category	Criteria
Premier	Has Special Features and Washrooms Or the Visitor Count is greater than 20000
Destination	Has Special Features or Washrooms
Local	Everything else

6.2.4 Add an AttributeCreator

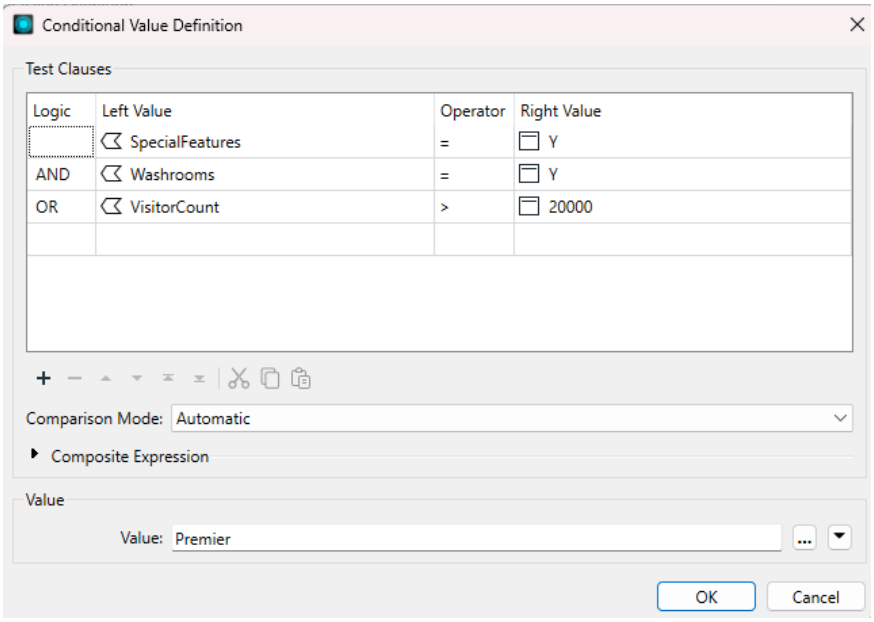
Add an AttributeCreator transformer and connect it to the Parks feature type. Within the transformers parameters set the New Attribute to be called *Category*.

Within the Attribute Value section use the dropdown menu arrow to select *Conditional Value....*



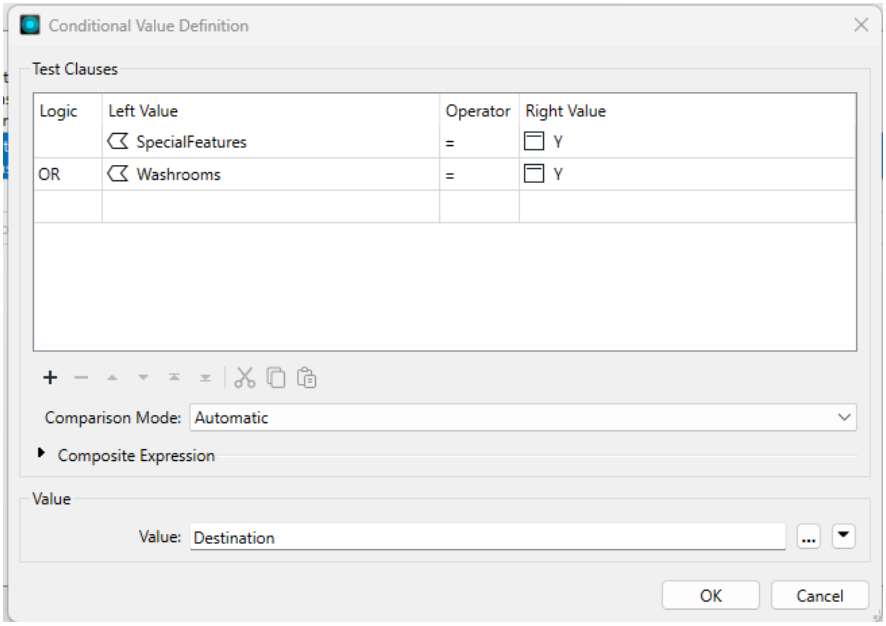
Set the first Test Condition; double-click on the '*If*' cell below '*Test Condition*'.
The Test Conditions dialog will open. In here we need to set multiple Test Clauses to filter features to meet the criteria for the Premier category.

And set the Attribute Value to *Premier*



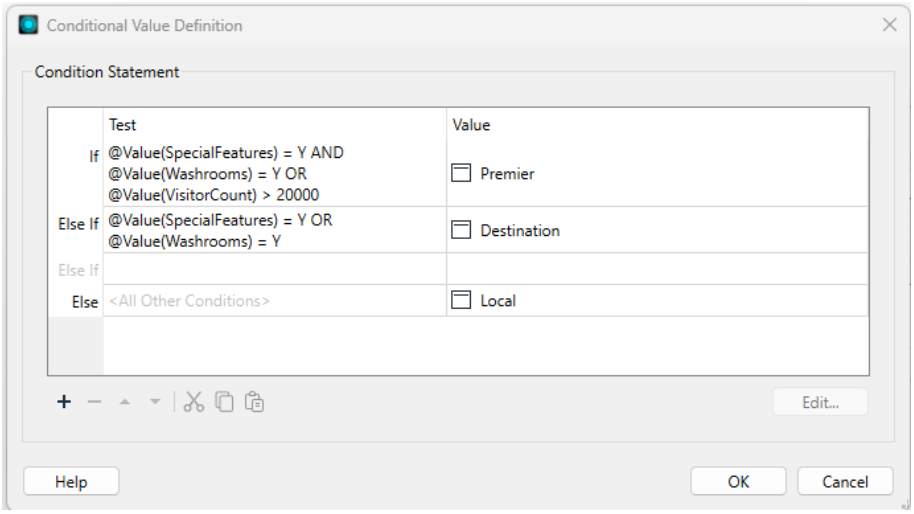
Click Ok to return to the Parameter Condition Definition dialog. Next, we need to set the Test Conditions for the Destination category. Double-click on the *Else If* cell below our first Test Condition.

The Test Conditions dialog will open. In here we need to set multiple Test Clauses to filter features to meet the criteria for the Destination category.
Also set the Attribute Value to *Destination*



Click OK to return to the Parameter Condition Definition dialog.

All other parks will be categorized as *Local*



Click OK and OK to confirm the changes and return to the canvas.

6.2.5 Run the Workspace and Examine results

Run the workspace. Then examine the output of the AttributeCreator within a Table View. A new attribute called Category has been created, and for each park feature it has been populated with a value based on the specified conditions.

Table

AttributeCreator_Output

	Category	ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures
56	Destination	56	14	Coopers' Park	Downtown	15493	9	Y	N	Y
57	Local	57	23	Helmcken Park	Downtown	14253	9	N	N	N
58	Local	58	29	Roundhouse Tu...	Downtown	12566	8	N	N	N
59	Premier	59	16	David Lam Park	Downtown	11660	5	N	Y	Y
60	Local	60	25	May & Lorne Br...	Downtown	10834	7	N	N	N
61	Local	61	-9999	Sunset Beach P...	West End	14832	6	Y	<missing>	<missing>
62	Premier	62	201	English Bay Bea...	West End	18022	17	N	Y	Y

Congratulations

By Completing this exercise, you have learned how to:

- Use Conditional Attribute Values to populate attributes based on conditions

6.3 Adjacent Feature Attributes

Demonstrates	Using Adjacent Feature Attributes
Overall Goal	Calculate monthly precipitation using adjacent feature attributes
Data	Precipitation Data (Microsoft Excel)
Start Workspace	none
End Workspace	C:\FMEModularData\Workspaces\Complete\6.03-AdvancedAttributes-AdjacentFeatureAttribute-Complete.fmw

You're working on a project mapping monthly precipitation (rainfall) in the city. Unfortunately, the numbers are a cumulative amount, and you wanted to map individual figures for each month. The dataset looks like so:

Month	Precipitation
Jan	168
Feb	273
Mar	387
Apr	476
May	541
Jun	595
Jul	631
Aug	668
Sep	719
Oct	840
Nov	1029
Dec	1191

Rather than reaching into your desk drawer for a calculator, you decide to use FME to do the calculations!

6.3.1 Generate Workspace

Launch the FME Workbench, if it isn't open already. Within the Get Started section of the Workbench, click on Generate Workspace.

Set the translation as follows:

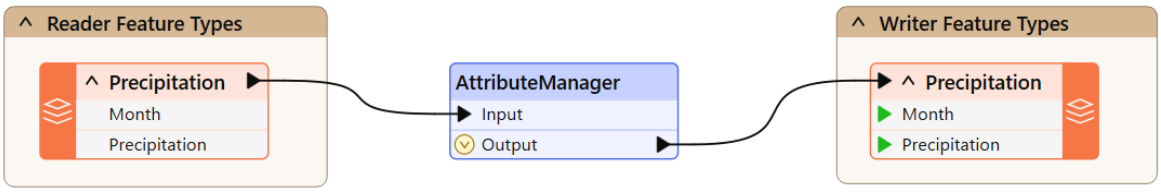
Reader Format	Microsoft Excel
Reader Dataset	C:\FMEModularData\Data\ElevationModel\Precipitation.xlsx
Writer Format	Microsoft Excel
Writer Dataset	C:\FMEModularData\Output\Training\MonthlyPrecipitationV2.xlsx

Check the parameters for the reader to ensure FME recognizes the headers at the top of each column.

6.3.2 Add an AttributeManager

To calculate precipitation for any given month, you need to subtract the previous month's cumulative total from the current month's cumulative total.

With FME you can use the Adjacent Feature Attribute functionality to fetch the previous month's number. So, place an AttributeManager transformer between the reader and writer feature types:

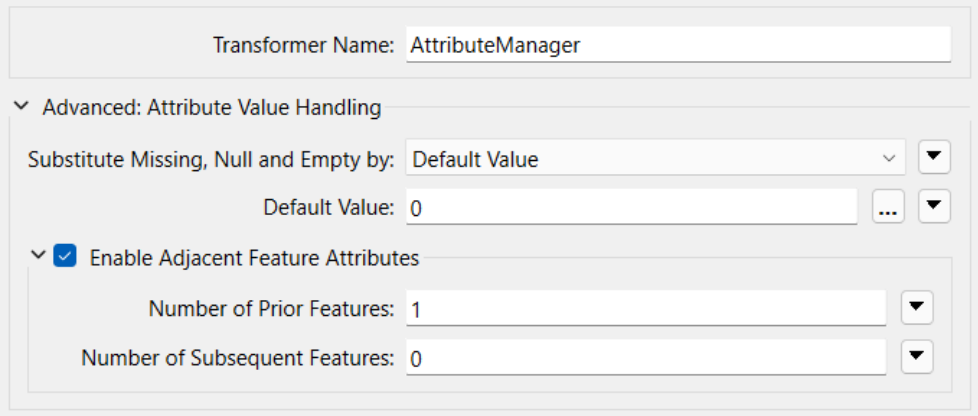


6.3.3 Set AttributeManager Parameters

Inspect the AttributeManager's parameters. Within the Advanced Attribute Handling section (expand using drop-down arrow), check the box marked *Enable Adjacent Feature Attributes*.

Enter *1* for the *Number of Prior Features*.
Enter *0* for Number of *Subsequent Features*

Then above, set the parameter *Substitute Missing, Null and Empty by:* to *Default Value* and enter *0* into the *Default Value*:



FME Lizard

The substitution parameter is important because the first feature to be processed can't have a prior feature, and the last feature to be processed can't have a subsequent one. Therefore, you always have to be careful about what you have set here.

In this exercise we're calculating a numeric value; therefore, it makes sense to use 0 (zero) as the default replacement.

Now let's create a new attribute and calculate the new precipitation value.
First create a new attribute called *MonthlyPrecipitation*

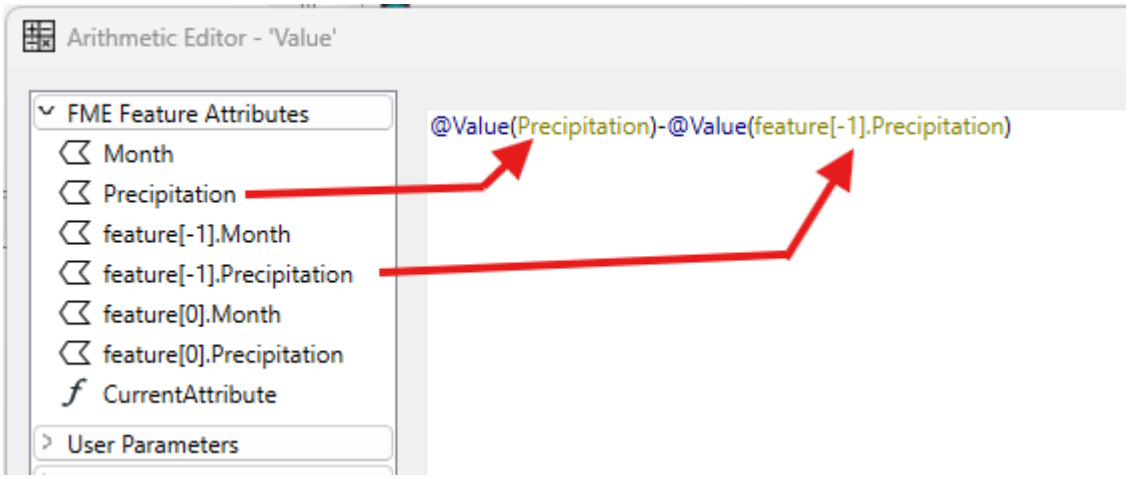
In the Attribute Value field for the *MonthlyPrecipitation* attribute, click the drop-down arrow and open the Arithmetic Editor.

In the arithmetic editor dialog use the menu on the left to select:

- The FME Feature Attribute Precipitation
- The Math Operator – (minus)
- The FME Feature Attribute Precipitation for feature[-1]

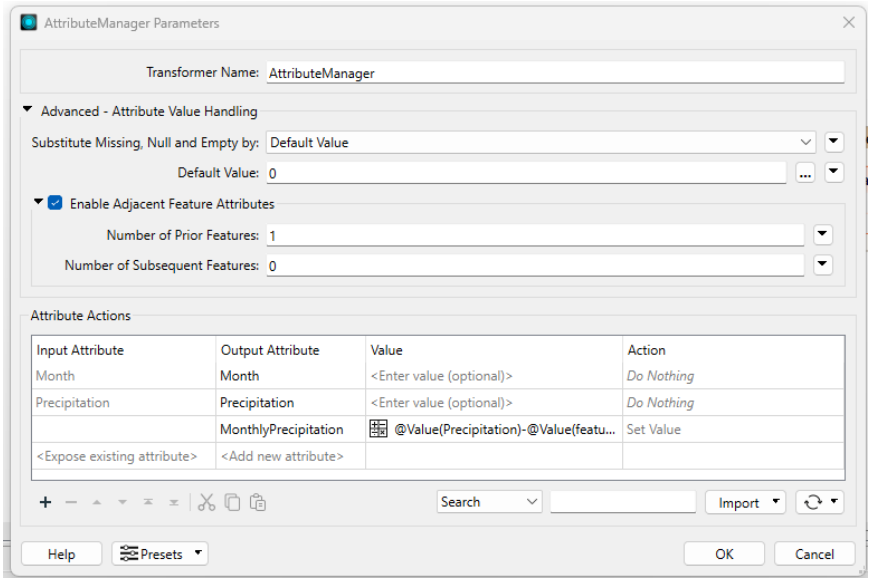
All of which should leave you with an expression looking like this:

```
@Value(Precipitation)-@Value(feature[-1]. Precipitation)
```



Now you can see why it was so important to set the substitution field because it's uncertain what result would occur from the above when *feature[-1].Precipitation* is missing!

Click OK to close the Arithmetic Editor dialog and then accept the parameter changes.



	Month	Precipitation	MonthlyPrecipitation
1	Jan	168	168
2	Feb	273	105
3	Mar	387	114
4	Apr	476	89
5	May	541	65
6	Jun	595	54
7	Jul	631	36
8	Aug	668	37
9	Sep	719	51
10	Oct	840	121
11	Nov	1029	189
12	Dec	1191	162

6.3.4 Save and Run Workspace

Save the workspace and then run it. Inspect the output:

FME Lizard

If the values in the output are literally "273-168", "387-273", etc., then you've used the string editor and not the arithmetic editor! If the values are all zero, then you need to make sure the AttributeManager action is set to Do Nothing for the Precipitation field, rather than Set Value.

Congratulations

By Completing this exercise, you have learned how to:

- Expose adjacent feature attributes
- Use adjacent feature attributes
- Handle missing values in attribute manipulation

6.4 Working with Lists

Demonstrates	Generating List attributes. Using transformers to extract from and manipulate Lists
Overall Goal	Create a copy of the park polygons that have attributes telling us what trees exist in each park, using List attributes. Investigate different ways in which the List attributes can be manipulated and values extracted.
Data	Full Trees (MapInfo TAB) Parks (Google KML)
Start Workspace	C:\FMEModularData\Workspaces\6.04-AdvancedAttributes-Lists-Begin.fmw
End Workspace	C:\FMEModularData\Workspaces\Complete\6.04-AdvancedAttributes-Lists-Complete.fmw

In this scenario, we want to create a copy of the park polygons that have attributes telling us what trees exist in each park. To do this, we will merge tree points with park polygons using the PointOnAreaOverlayer.

However, it isn't enough to simply copy attributes from each tree to their respective park polygons, because what if there is more than one tree per park? List attributes are perfect for this scenario: if more than one tree falls within the same park polygon, we can use a list attribute to store information for all trees. The PointOnAreaOverlayer allows us to generate a list that stores the values for all points overlaid on the area.

Once the list attribute has been generated, we'll then go on to explore manipulating lists using transformers.

First, we need a workspace that will overlay the trees and parks and generate the required list attribute.

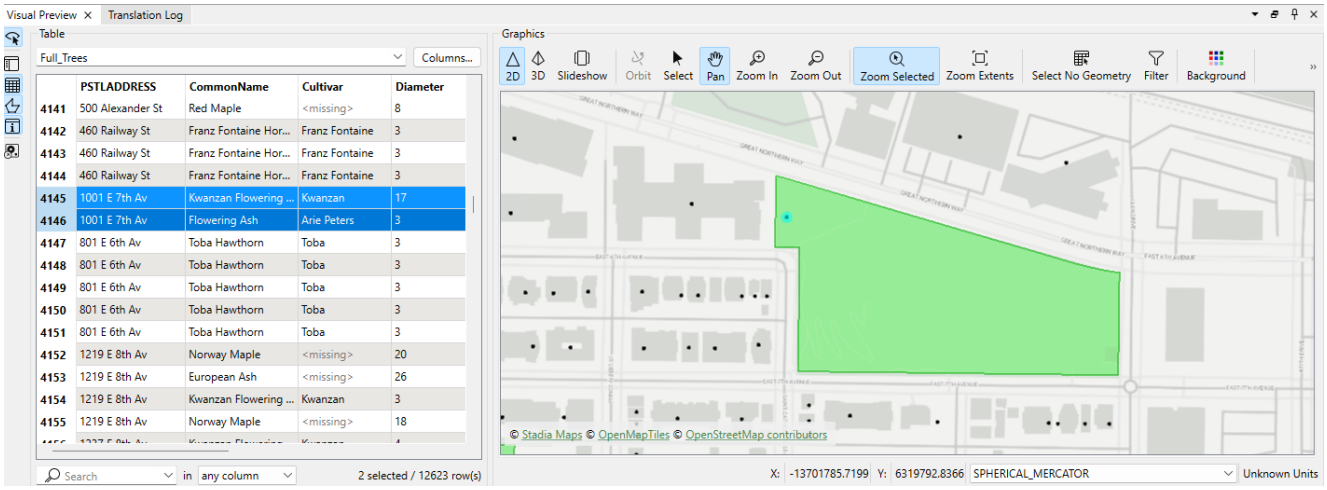
6.4.1 Open Workspace and Inspect the Source Datasets

A workspace already exists with the first part of our required workflow already setup.

Launch the FME Workbench, if it isn't open already. Within the Get Started section of the Workbench, click on Open Workspace. Navigate to and open "C:\FMEModularData\Workspaces\6.04-AdvancedAttributes-Lists-Begin.fmw"

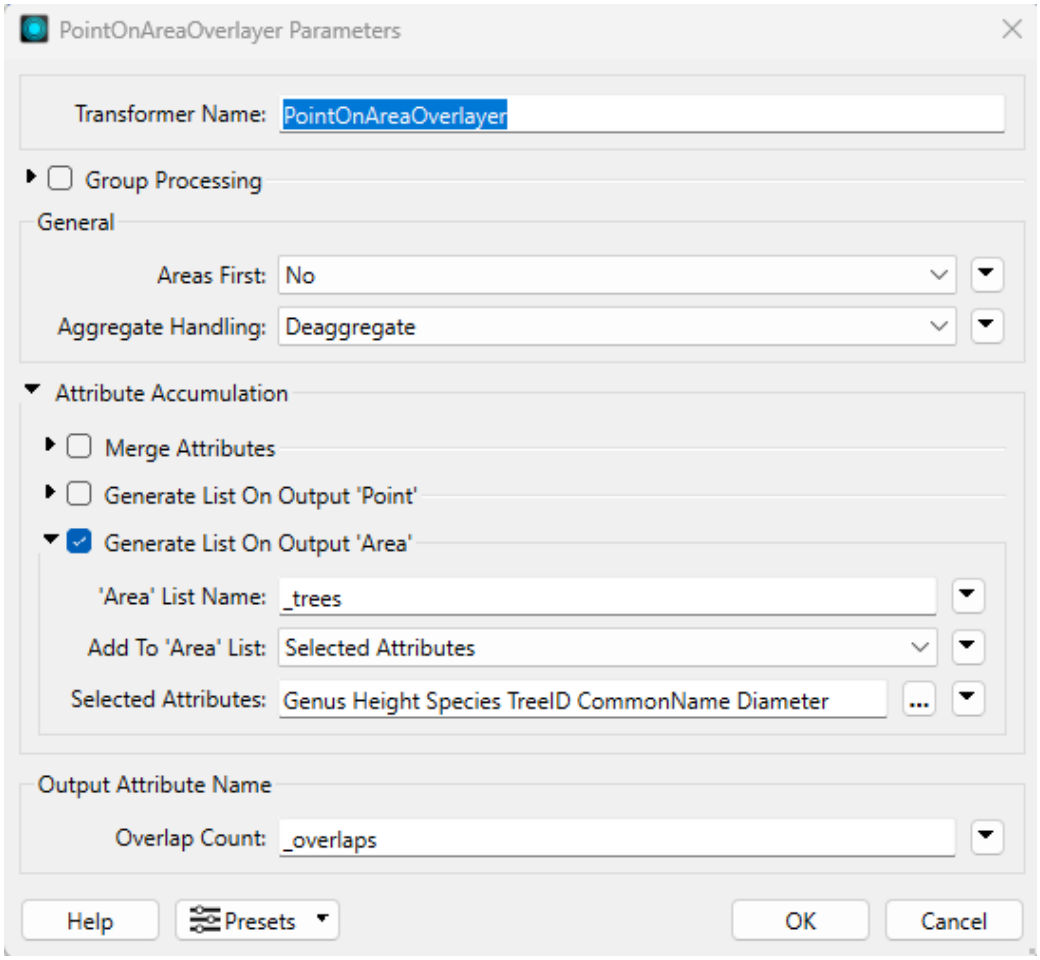
Familiarize yourself with both the tree and parks data, using the Visual Preview or Data Inspector tools.

Note how some parks contain multiple point features. For example, China Creek Park North contains two coincident trees, as shown in the Table View:



6.4.2 Configure the PointOnAreaOverlayer Parameters

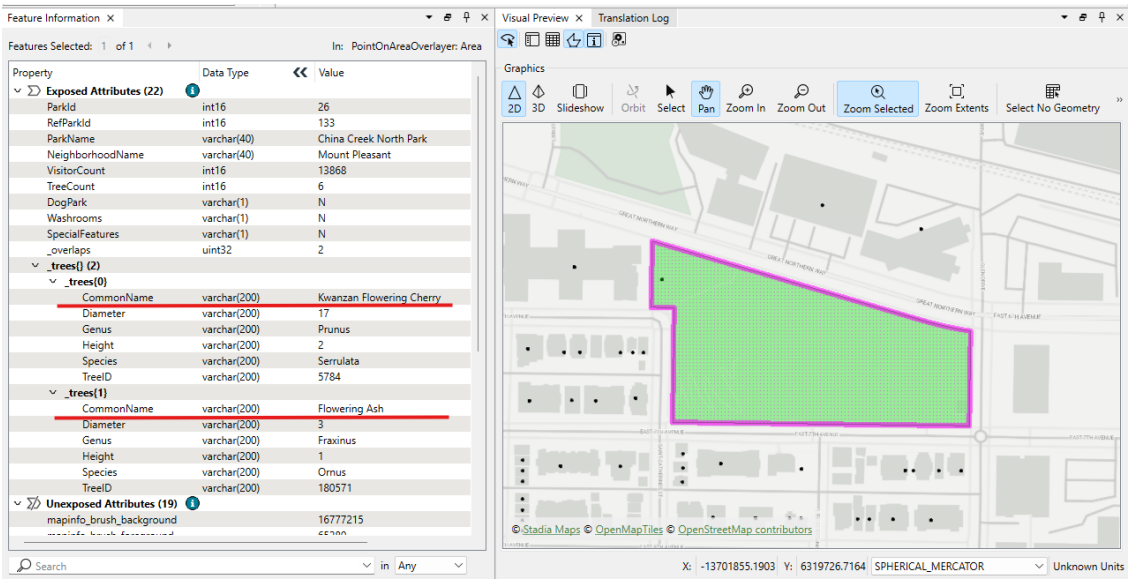
Double-click the PointOnAreaOverlayer to open its parameters. Check *Generate List* on Output '*Area*'. Name the list *_trees* and use *Selected Attributes* to select the following attributes (CommonName, Diameter, Genus, Height, Species and TreeID).



Click OK to close the parameters

6.4.3 Run the Workspace

Run the workspace and inspect China Creek Park North again. The Feature Information pane shows a list attribute, *_trees{}*, which contains information from both tree points in the park:



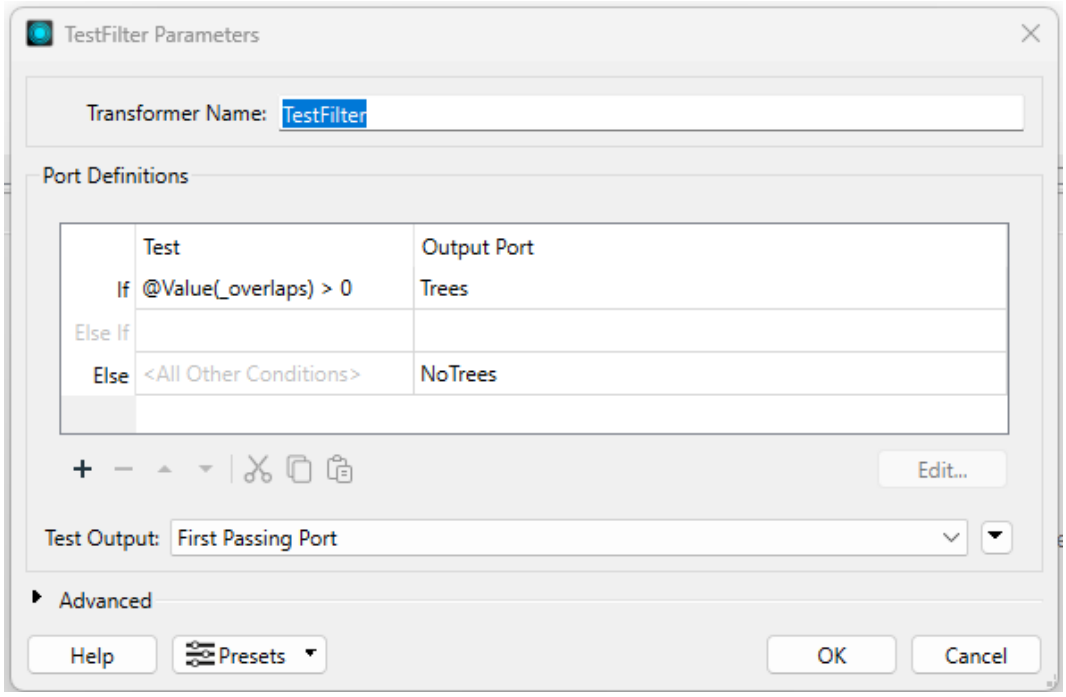
We successfully used the PointOnAreaOverlayer to create a list called *_trees{}*, which stores the values of multiple tree points that fall within each park polygon. Note how each numbered list element contains the same set of attributes: *CommonName, Diameter*, etc.

The next step is to process and analyze the data. For example, a TestFilter can be used to filter out parks that contain no tree points, and then list manipulation transformers can be used for tasks like the following:

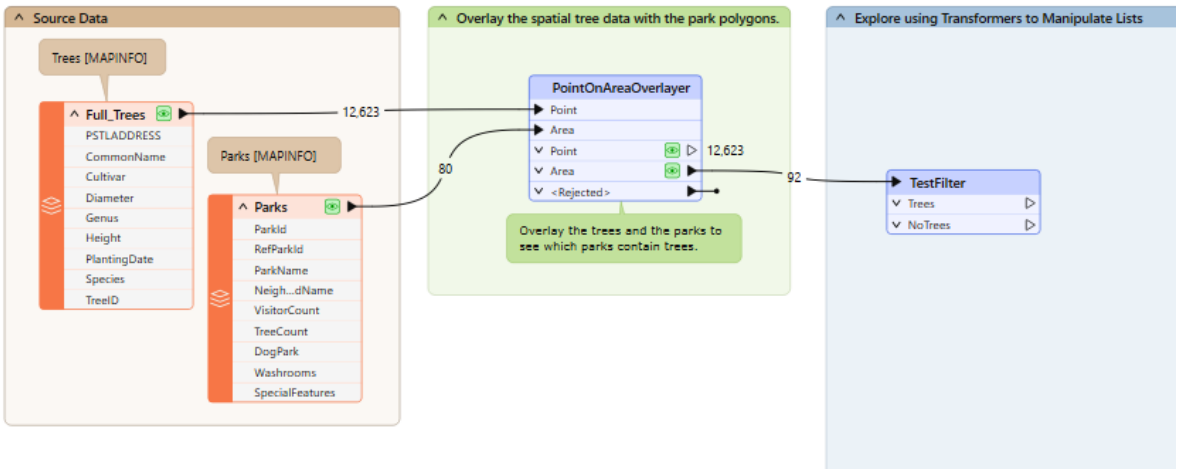
Operation	Transformer
Count the number of trees in each park	**ListElementCounter**
Find the maximum tree diameter	**ListSorter & ListIndexer**
Find the count of each species	**ListHistogrammer**
Create a list of species	**ListConcatentaor**
Find which parks have an Oak tree	**ListSearcher**
Create a table of park trees with the park name	**ListExploder**
Find the average tree height in a park	**ListSummer**
Find the minimum/maximum of tree diameters	**ListRangeExtractor**

6.4.4 Add a TestFilter transformer

Add a TestFilter transformer and connect it to the *Area* output port of the PointOnAreaOverlayer. Then configure the transformer to separate features into those that do and don't have tree overlaps:



Also add appropriate bookmarks and annotations.



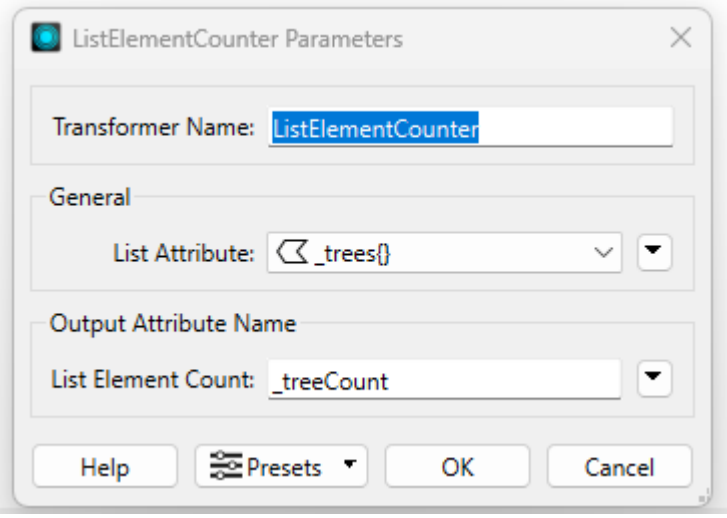
We'll now explore some of the list manipulation transformers.

6.4.5 Add a ListElementCounter transformer

The ListElementCounter counts the number of elements in a list. For example, we can use it to count the elements in the _trees{} list and store that number in the _treeCount attribute.

Add a ListElementCounter transformer to the canvas and connect it to the *Trees* output

port of the TestFilter.

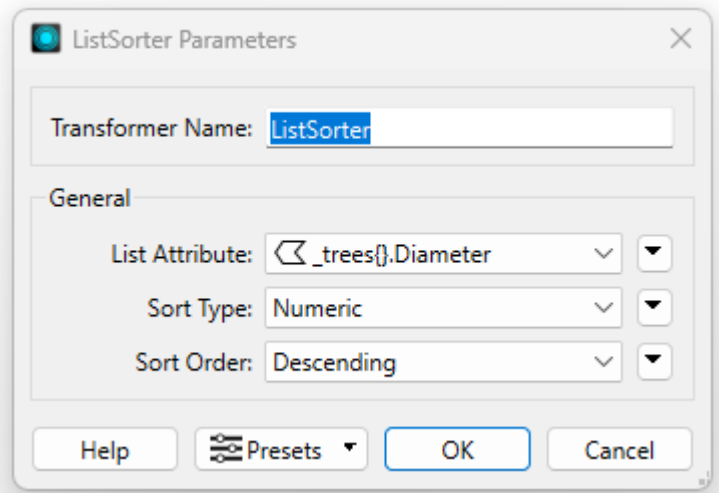


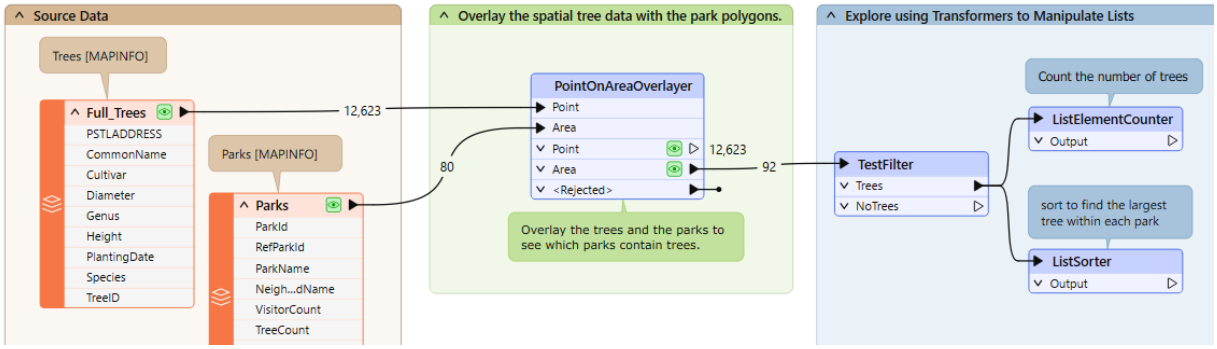
Since the number of elements in the list represents the number of tree points within a particular park, the result tells us how many trees are in each park.

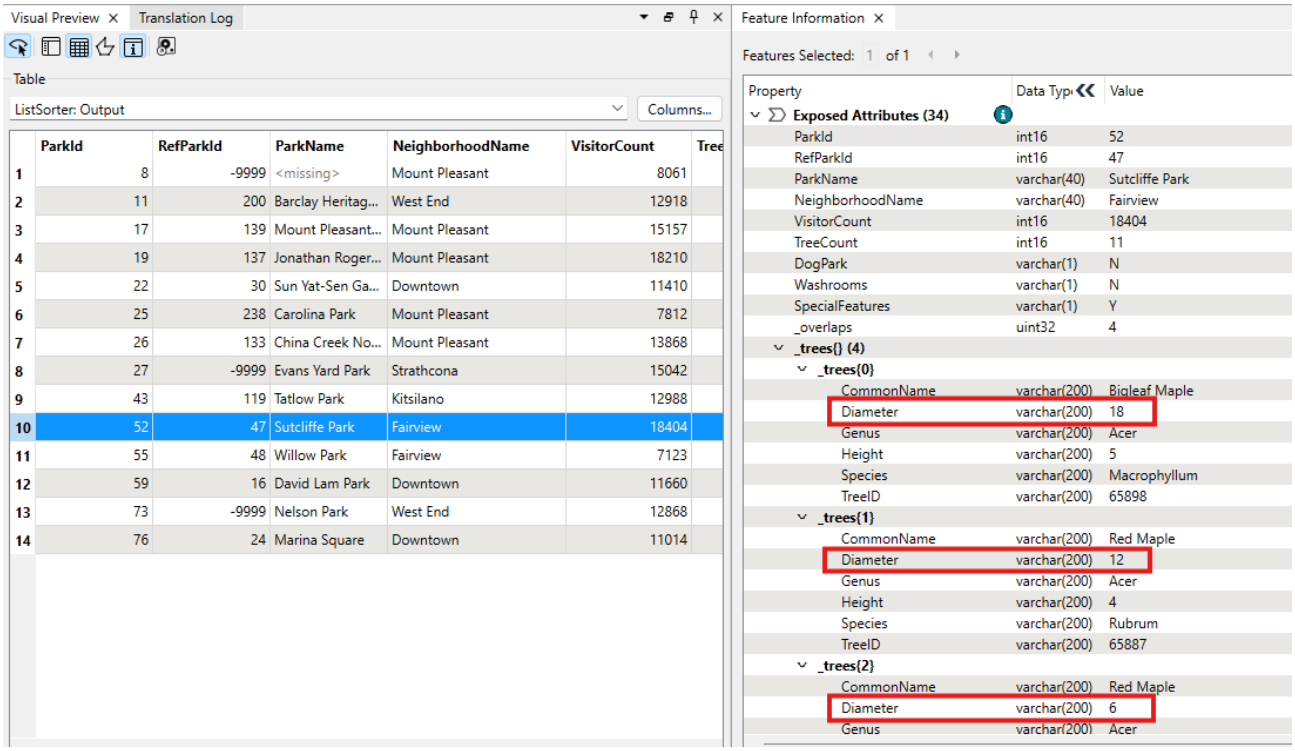
6.4.6 Add a ListSorter transformer

Sorting lists alphabetically or numerically makes it easy to pick off certain characteristics from a list. For example, we will sort the _trees{}.Diameter list attribute to find the largest tree.

Add a ListSorter transformer to the canvas and connect it to the *Trees* output port of the TestFilter.





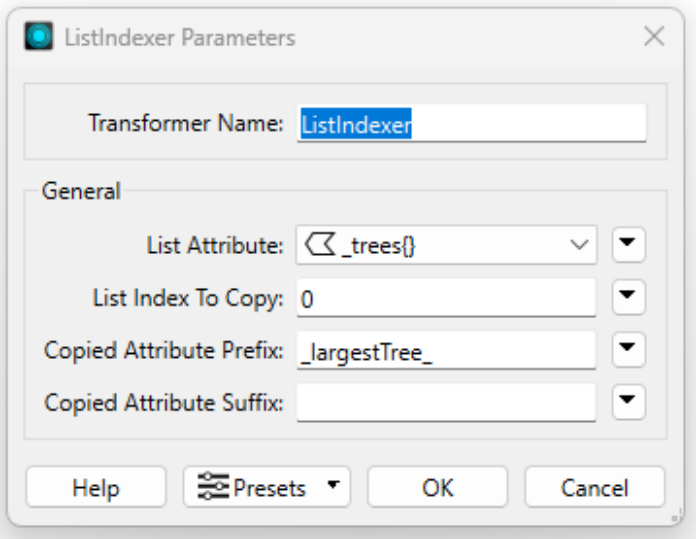


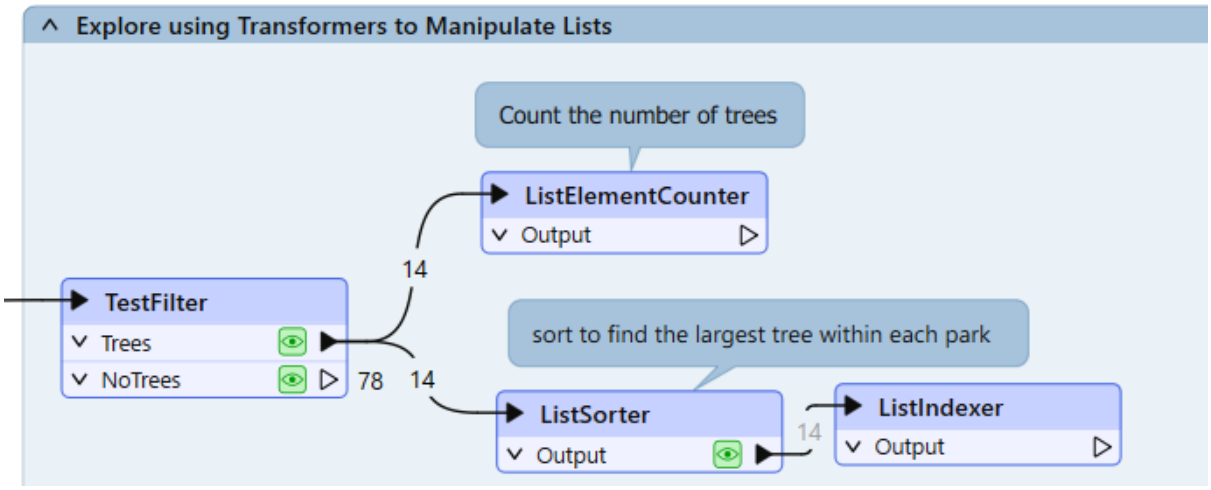
6.4.7 Add a ListIndexer transformer

The ListIndexer turns selected lists into distinct features. It only explodes the element(s) at a given index, e.g. {1}. This transformer is often placed after another transformer that arranges the list in a particular order before choosing the index – for example, the ListSorter or ListSearcher.

In this exercise the ListIndexer can be used to pick the largest tree in the *_trees{}* list. First, the ListSorter is used to sort the list by the *_trees{}.Diameter* attribute. Next, the ListIndexer is going to be used to get the tree at index *0*. The result is stored in a set of new attributes with the prefix *_largestTree_*.

Add a ListIndexer transformer to the canvas and connect it to the output port of the ListSorter transformer.





The details of the largest tree for each park are then extracted as new attributes on the parks dataset:

Table

ListIndexer_Output

Columns...

		ParkName	_largestTree_CommonName	_largestTree_Diameter	_largestTree_Genus	_largestTree_Height	_largestTree_Species	_largestTree_TreeID
1	2	<missing>	Chanticleer Pear	6	Pyrus	3	Calleryana	196868
2	4	Barclay Heritag...	Schwedler Norway Maple	26	Acer	4	Platanoides	58777
3	6	Mount Pleasant...	Worplesdon Sweetgum	8	Liquidambar	3	Styraciflua	130369
4	1	Jonathan Roger...	Kwanzan Flowering Cherry	19	Prunus	3	Serrulata	6098
5	5	Sun Yat-Sen Ga...	Persian Ironwood	4	Parrotia	1	Persica	208530
6	1	Carolina Park	Eastern Redbud	4	Cercis	1	Canadensis	204133
7	2	China Creek No...	Kwanzan Flowering Cherry	17	Prunus	2	Serrulata	5784
8	35	Evans Yard Park	Aristocrat Pear	6.5	Pyrus	4	Calleryana	157977

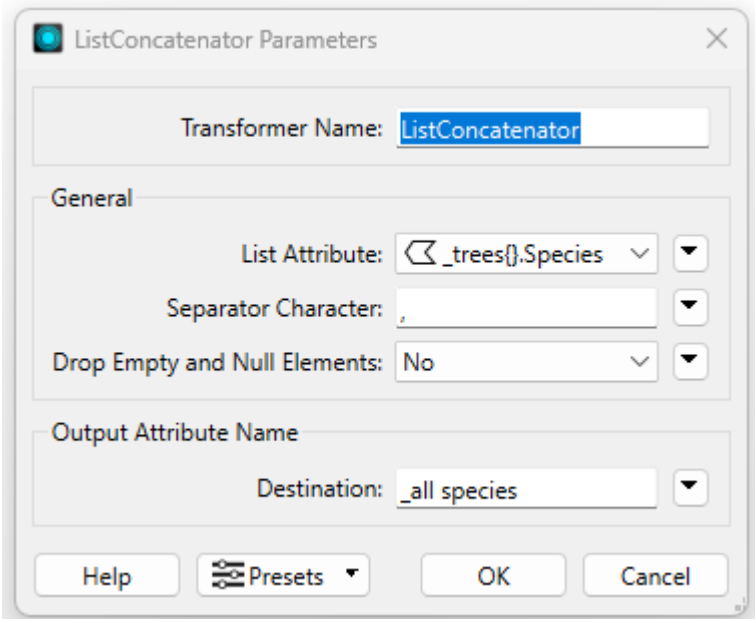
6.4.8 Add a ListConcatenator transformer

The ListConcatenator strings the elements of a list together into a single attribute value. In the parameters, you can specify which character to use as a separator (e.g. a comma or a newline character) and the name of the resulting attribute.

Let's create a new attribute on our parks called *_all species* and then populate it with all

the species (comma-separated) within each park

Add a ListConcatenator transformer to the canvas and connect it to the *Trees* output port of the TestFilter.



This new attribute will be added and populated with all the _tree{}.Species list elements.

Table

ListConcatenator_Output

	e	_all species
1	2	Calleryana,Calleryana
2	4	Platanoides,Platanoides,Platanoides,Platanoides
3	5	Styraciflua,Styraciflua,Styraciflua,Styraciflua,Styraciflua,Styraciflua
4	1	Serrulata
5	5	Persica,Persica,Persica,Persica,Persica
6	1	Canadensis
7	2	Serrulata,Ornus

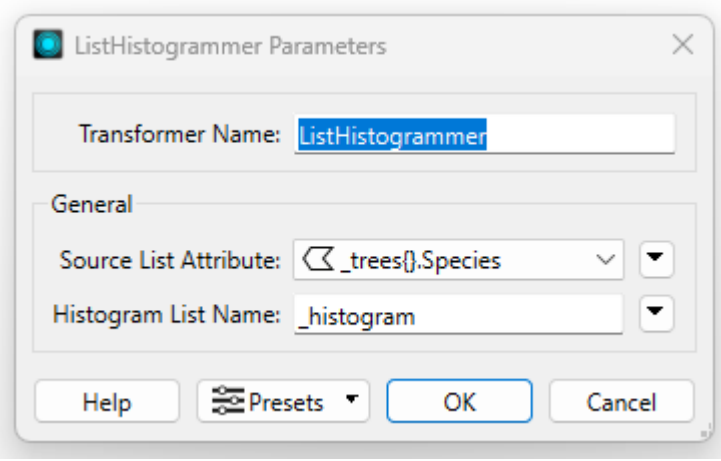
However, it might be more meaningful to remove duplicate species values. Instead, we could have first used a ListHistogrammer or used a ListDuplicateRemover.

6.4.9 Add a ListHistogrammer transformer

The ListHistogrammer builds a histogram from the values in a list and returns these in a new list. The new list is sorted so the value with the most occurrences will be first.

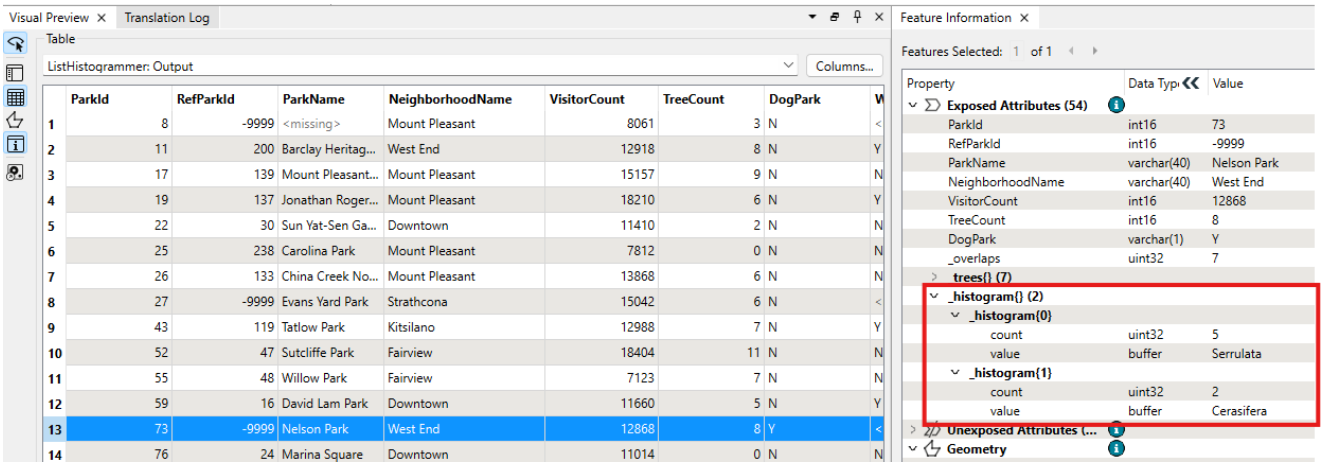
We can use the ListHistogrammer to find out how many of each species are in each park.

Add a ListHistogrammer transformer to the canvas and connect it to the *Trees* output port of the TestFilter.



With these parameters set, the transformer will count how many of each tree name exists in the *_tree{}* list. It derives the *_histogram{}* list and outputs it.

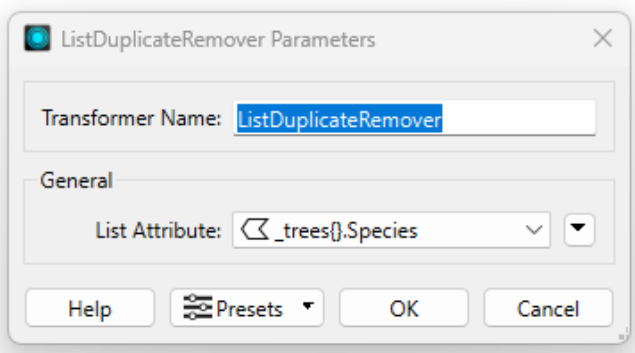
Note how the resulting list includes *.count* and *.value* attributes:



6.4.10 Add a ListDuplicateRemover transformer

The ListDuplicateRemover cleans up a list by removing elements with duplicates of a particular attribute value.

Add a ListDuplicateRemover transformer to the canvas and connect it to the *Trees* output port of the TestFilter. Configure its parameters to remove duplicate Species for each park.



Compare the output of the ListDuplicateRemover to the original for a couple of parks with multiple occurrences of the same species. (e.g. Barclay Heritage Square, Sutcliffe Park, Nelson Park)

6.4.11 Add a ListExploder transformer

The ListExploder is in the top 30 most frequently used FME transformers because of its power to turn lists into distinct features. It explodes a list by creating a feature for each element. Most formats do not have the ability to write list structures (XML and JSON being notable exceptions), so exploding lists can be useful when elements need to be preserved.

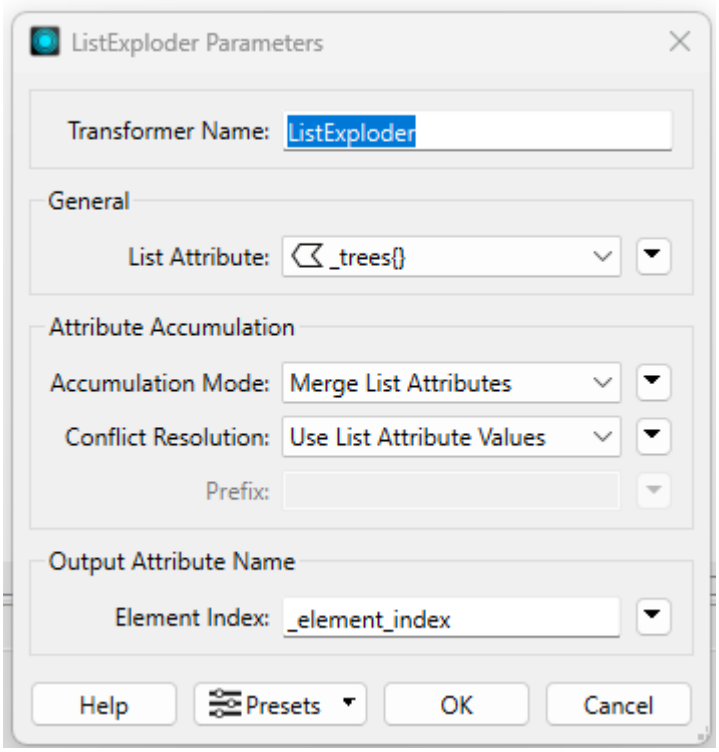
The ListExploder parameters allow you to specify the list that is to be turned into distinct features. Optionally, an "element index" attribute can be added to each feature to store the position of the element in the original list.

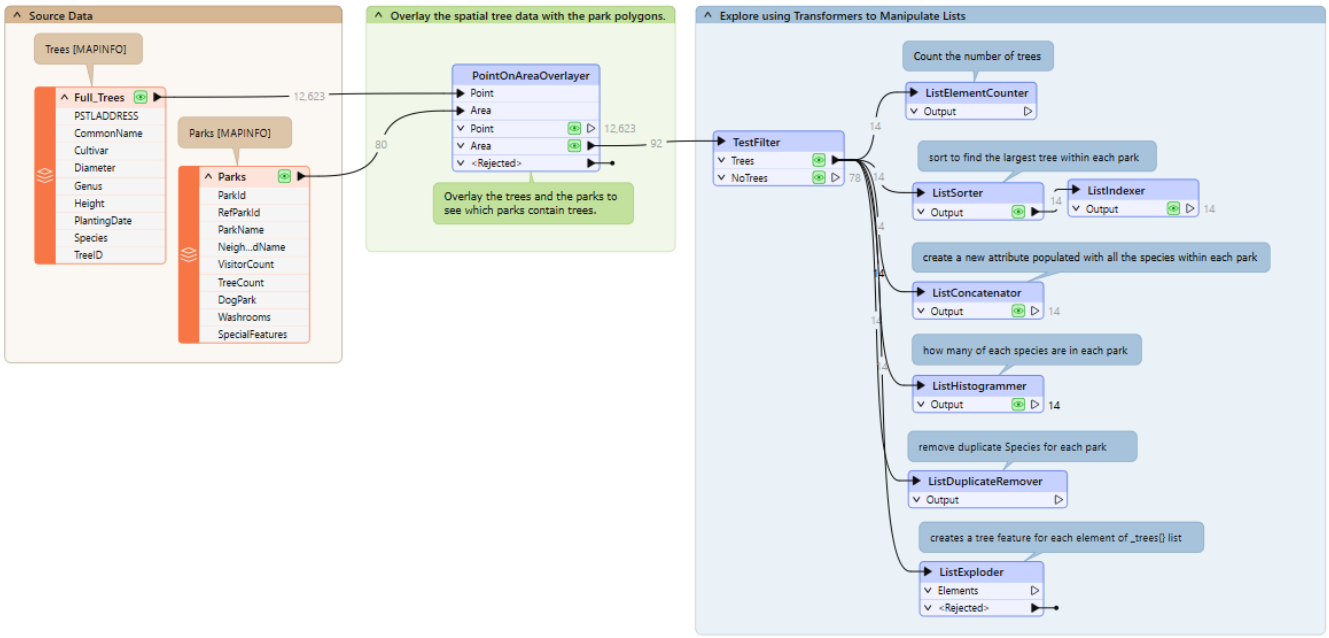
The “Attribute Accumulation” section is where you choose how to merge or preserve the incoming attributes. "Accumulation Mode" has three options:

- *Merge List Attributes* - retain all attributes and add the exploded list attributes.
- *Prefix List Attributes* - retain all attributes and add a prefix to the exploded list attributes. This mode is a good choice for retaining all attributes and avoiding attribute name conflicts.
- *Only Use List Attributes* - preserve the flattened list attributes and remove all original attributes.

"Conflict Resolution" helps resolve problems when an exploded list attribute ends up with the same name as one of the original attributes. You can choose to either preserve the original attribute or give the exploded attribute precedence. If you use Prefix List Attributes, there is no need for conflict resolution.

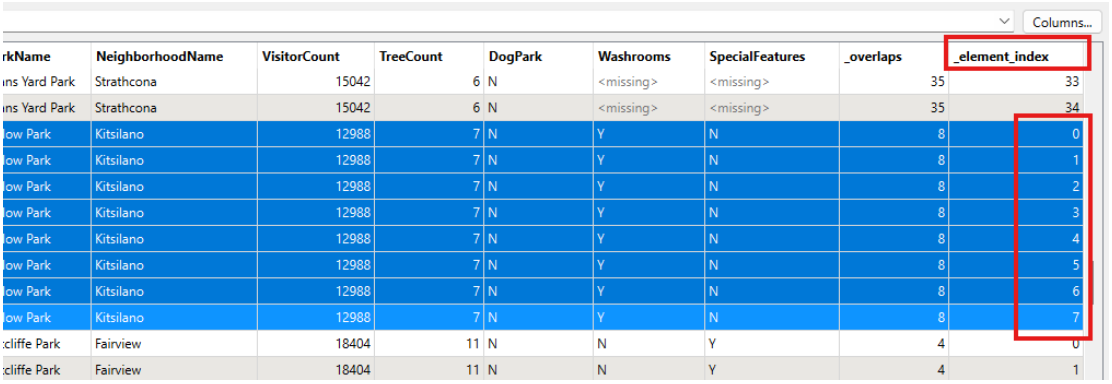
When we explode the *_trees{}* list into distinct features, note how the features outputted by the ListExploder keep their respective park attributes, as well as the attributes of that particular tree:





The ListExploder also adds a new *_element_index* attribute to indicate the position of this element in the original list, starting at '0' and going up.

Run the workspace and inspect the output to see how the trees are represented in the new table. Note that 14 features go into the transformer and 84 come out, which indicates that the list has successfully been exploded into distinct features.



rkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures	_overlaps	_element_index
ıns Yard Park	Strathcona	15042	6	N	<missing>	<missing>	35	33
ıns Yard Park	Strathcona	15042	6	N	<missing>	<missing>	35	34
low Park	Kitsilano	12988	7	N	Y	N	8	0
low Park	Kitsilano	12988	7	N	Y	N	8	1
low Park	Kitsilano	12988	7	N	Y	N	8	2
low Park	Kitsilano	12988	7	N	Y	N	8	3
low Park	Kitsilano	12988	7	N	Y	N	8	4
low Park	Kitsilano	12988	7	N	Y	N	8	5
low Park	Kitsilano	12988	7	N	Y	N	8	6
low Park	Kitsilano	12988	7	N	Y	N	8	7
cliffe Park	Fairview	18404	11	N	N	Y	4	0
cliffe Park	Fairview	18404	11	N	N	Y	4	1

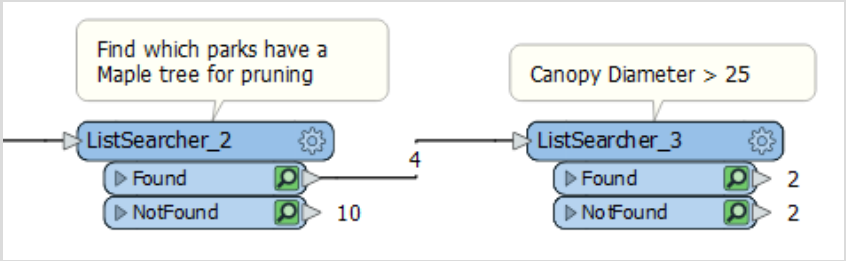
There are even more transformers for working with lists, but we don't have time to cover them all now. Check out ListSearcher, ListSummer, ListRangeExtractor and ListBuilder transformers if you get some free time.

FME Lizard

Python can be useful for performing more advanced list manipulation. There is a correlation between Python lists and FME lists. Consider the following list of tree names:

```
# Python list:
_trees = ["Fernleaf Beech", "Aristocrat Pear", "European Beech", "Pin Oak"]
# FME list attribute:
_trees{0} = Fernleaf Beech
_trees{1} = Aristocrat Pear
_trees{2} = European Beech
_trees{3} = Pin Oak
```

For example, say we want to search a list for elements that meet two conditions: it is a Maple tree and it has a canopy diameter greater than 25. We could chain together two ListSearchers to check each condition:



This indicates which parks have Maple trees with canopies greater than 25, but it doesn't output a list of all of the qualifying Maple trees. We can achieve this more advanced scenario with Python using the PythonCaller.

Two calls help pass lists in and out of Python. Start with feature.getAttribute():

```
_treesName = feature.getAttribute('_trees{}.CommonName')
```

Then perform the desired list manipulation, and then finish with feature.setAttribute():

```
feature.setAttribute('_trees{}.CommonName', _treesList)
```

See Python and FME Basics for a tutorial on working with Python in FME: https://community.safe.com/s/article/python-and-fme-basics

Congratulations

By Completing this exercise, you have learned how to:

- Create a list attribute using a transformer.
- View a list attribute in the Feature Information Window
- Manipulate lists and extract information from them using common list transformers.

7 Advanced Workflow Design

7.1 Using Parameters

Demonstrates	Use of FME Parameters Creation and use of User Parameters ParameterFetcher transformer
Overall Goal	Create a workspace where end-users are able to input metadata values for use within the workflow at the time of running the workspace
Data	Parks (MapInfo TAB)
Start Workspace	C:\FMEModularData\Workspaces\7.01-AdvancedWorkflow-Parameters-Begin.fmw
End Workspace	C:\FMEModularData\Workspaces\7.01-AdvancedWorkflow-Parameters-Complete.fmw

In this example, imagine that you are a GIS technician working for the city. The team responsible for maintaining parks has a workspace that translates their data from the source MapInfo TAB format to Google KML. It also writes a file of XML metadata to show who translated the data and when.

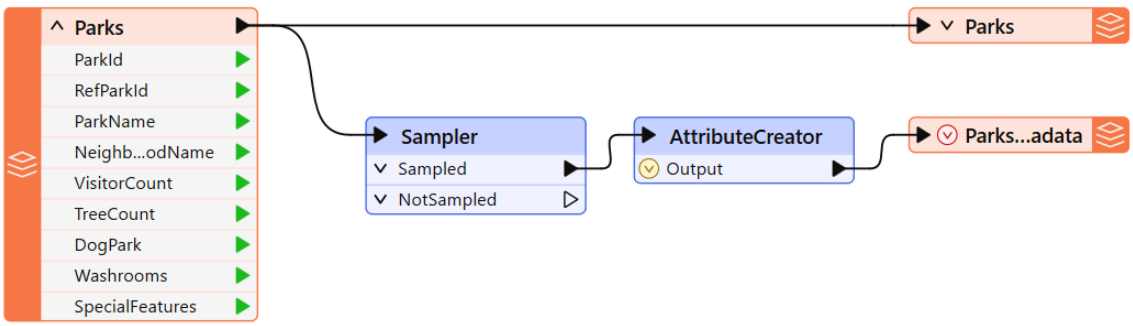
At the moment there are a number of problems they face:

- The XML output is not particularly well formatted
- The date attribute is being rejected by an online XML validator
- All of the XML metadata fields are hard-coded in an AttributeCreator transformer. This is quite inconvenient (especially when they want to run the workspace on FME Server!)

You have been assigned to help solve these problems. At least one of these requires you to create user parameters to take the place of hard-coded values.

7.1.1 Launch FME Workbench and Open Workspace

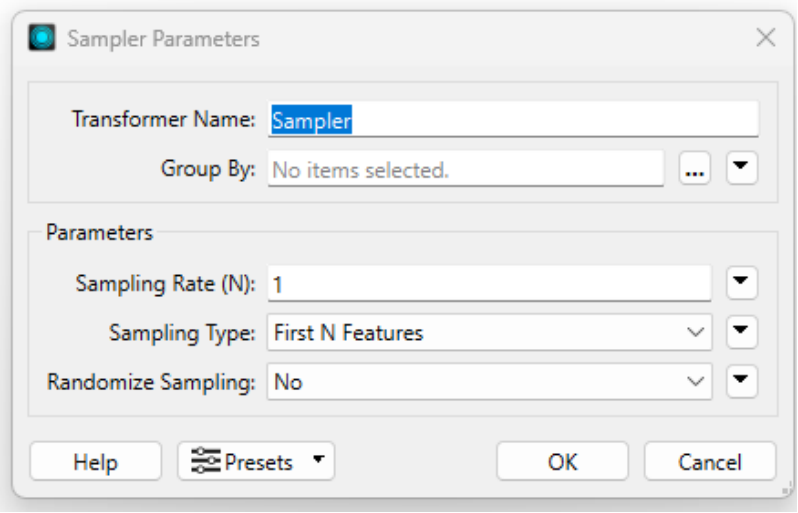
Launch the FME Workbench, if it isn't open already. Then open the workspace C:\FMEModularData\Workspaces\7.01-AdvancedWorkflow-UseOfParameters-Begin.fmw



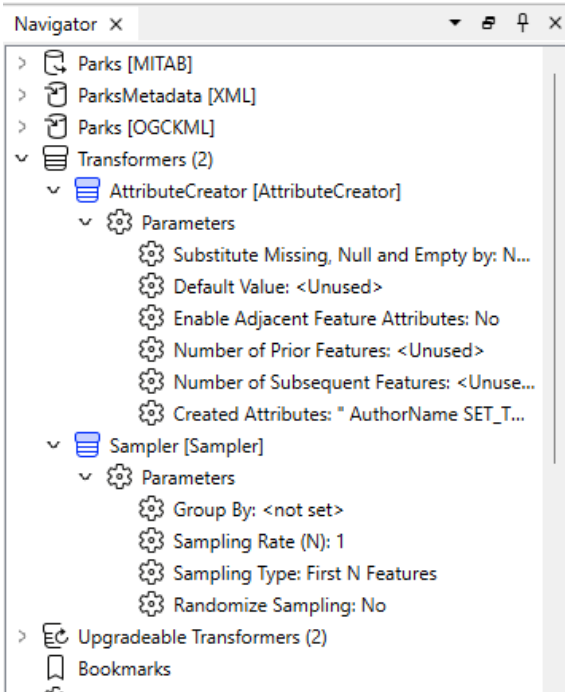
The metadata part of the translation consists of the two transformers and an XML writer feature type.

The Sampler transformer ensures that only one record is written to the output metadata, by discarding all but one feature, and the AttributeCreator creates a set of attributes to write to the metadata.

Check the parameters for each transformer in turn. These are FME parameters, set by the workspace author and not available to the end-user. Here, for example, are the parameters for the Sampler transformer:

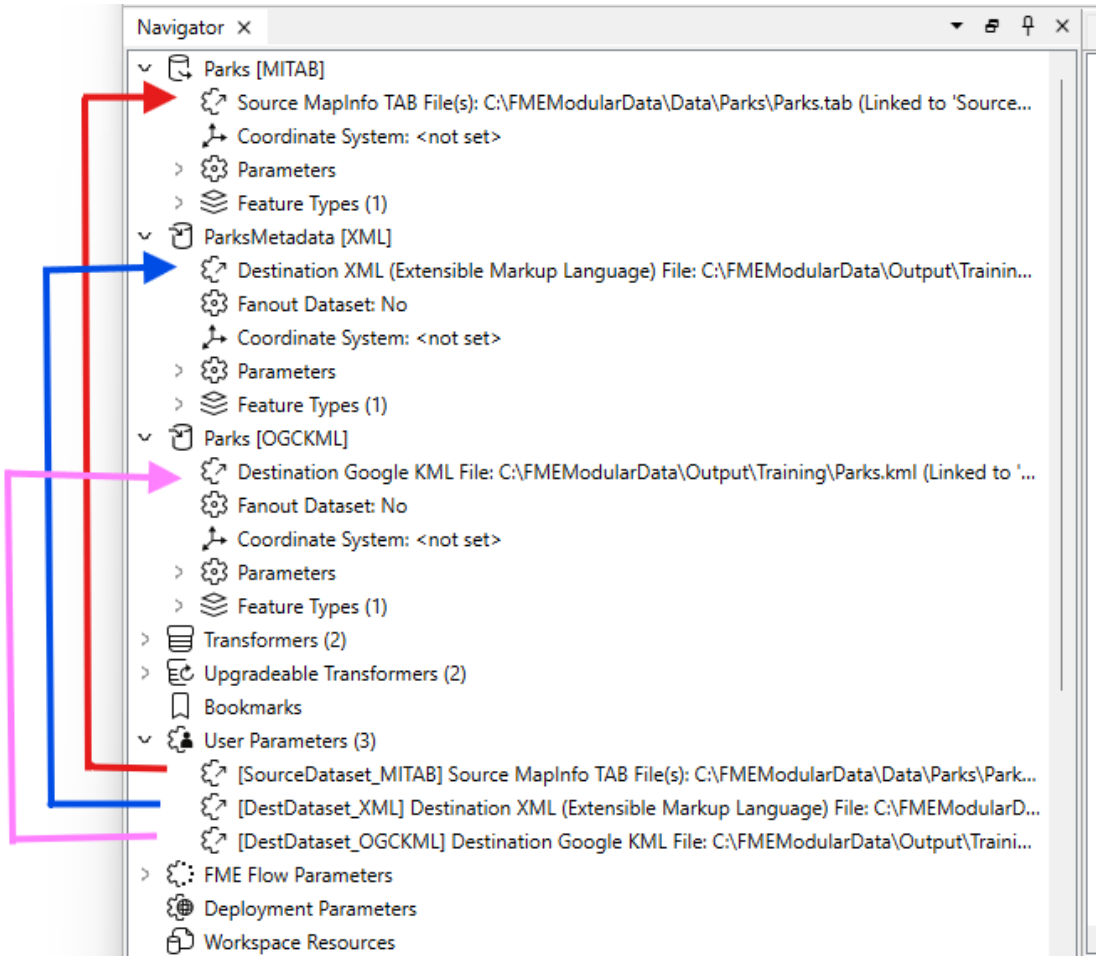


You can also find these parameters in the Parameter Editor window, the transformers' Parameters Dialog, and under the Transformers section of the Navigator window.



7.1.2 Examine the existing User Parameters

Within the Navigator panel examine the existing User Parameters. There are three, one for the Parks MITAB Reader Source, one for the ParksMetadata XML Writer Destination and one for the Parks KML Writer Destination:



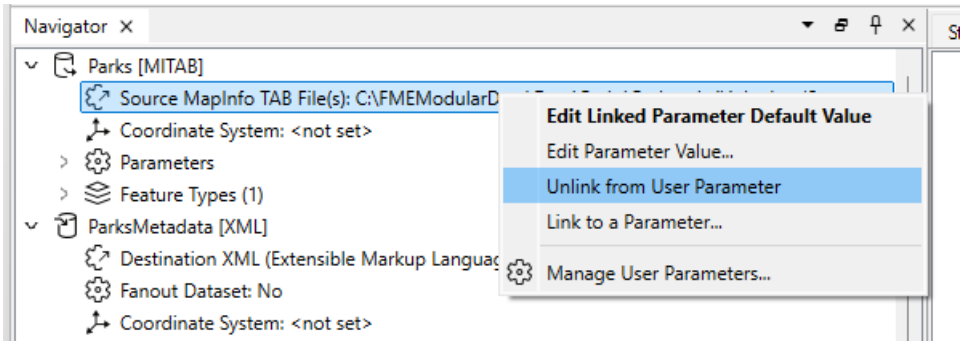
By default, when adding new readers and writers User Parameters are automatically created for the source and destinations.

In this scenario it's not appropriate for end users to alter the source and destinations. So we will convert these User Parameters into just FME Parameters

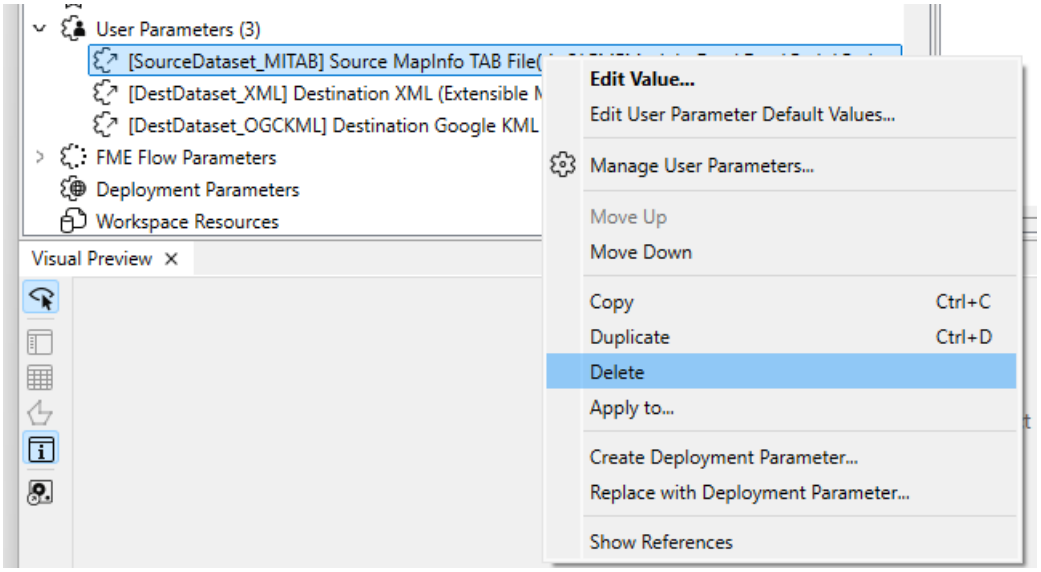
7.1.3 Convert Reader & Writer User Parameters to FME Parameters

We can convert the reader/writer source and destinations parameters from User to FME by either:

- Right-clicking on the Source/Destination parameter and choosing *Unlink from User Parameter:*



- Or, within the User Parameters section, right-click on each and choose to *delete* them:

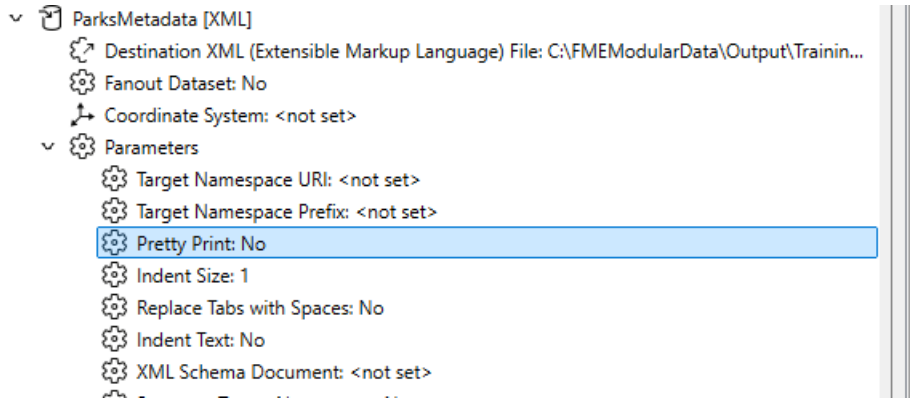


Make sure all three existing User Parameters are unlinked/deleted (Parks MITAB Reader Source, ParksMetadata XML Writer Destination and Parks KML Writer Destination).

Their parameter cogs should now be green in colour (no longer purple), reflecting that they are now FME Parameters, not User Parameters.

7.1.4 Change XML Writer FME Parameter

An FME parameter called Pretty Print controls the style of the XML file being written:



To ensure the output is always well-formatted, we should set this parameter to Yes - *but we won't create a user parameter from it because we don't want the end-user to change it.*

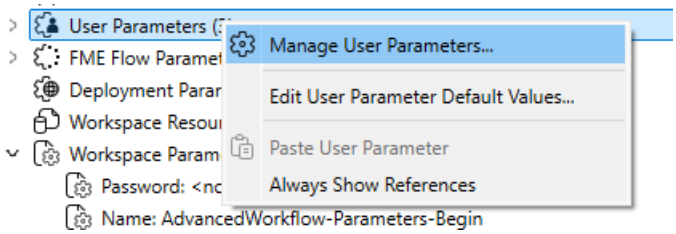
In the Navigator window locate the XML writer, expand the parameters list, and locate the parameter labelled *Pretty Print*.
Double-click it. In the dialog that opens, change the value to *Yes* and then click OK to close the dialog.

We have now - as a workspace author - changed an FME parameter.

7.1.5 Create a User Parameter

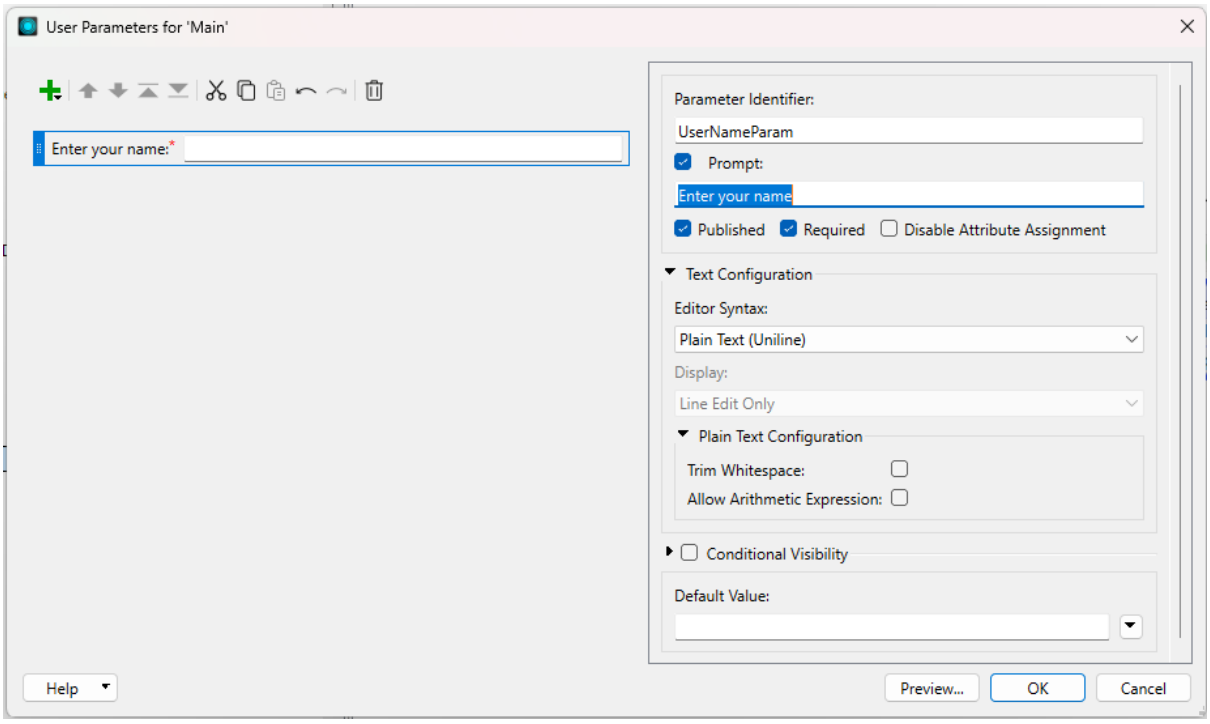
The output schema has three variable attributes: username, user company (organization), and user email. We should create a user parameter for each of these to allow the end-user to enter that information.

Firstly, locate the User Parameters section of the Navigator window, right-click on it, and choose the option to *Manage User Parameters…:*



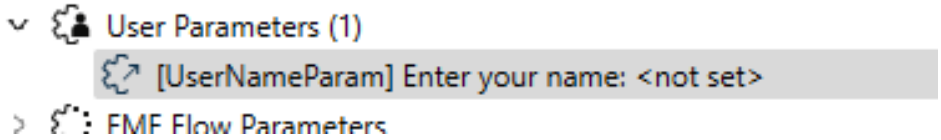
Click on the '+' button to add a new parameter, and choose *Text* as the type.

Each parameter needs a Parameter Identifier, so call this one *UserNameParam*
Now enter a Prompt, such as *"Enter your name"*



This is a required value that the user must define (we don't want them to be able to leave it unpopulated), so ensure '*Required*' is ticked.

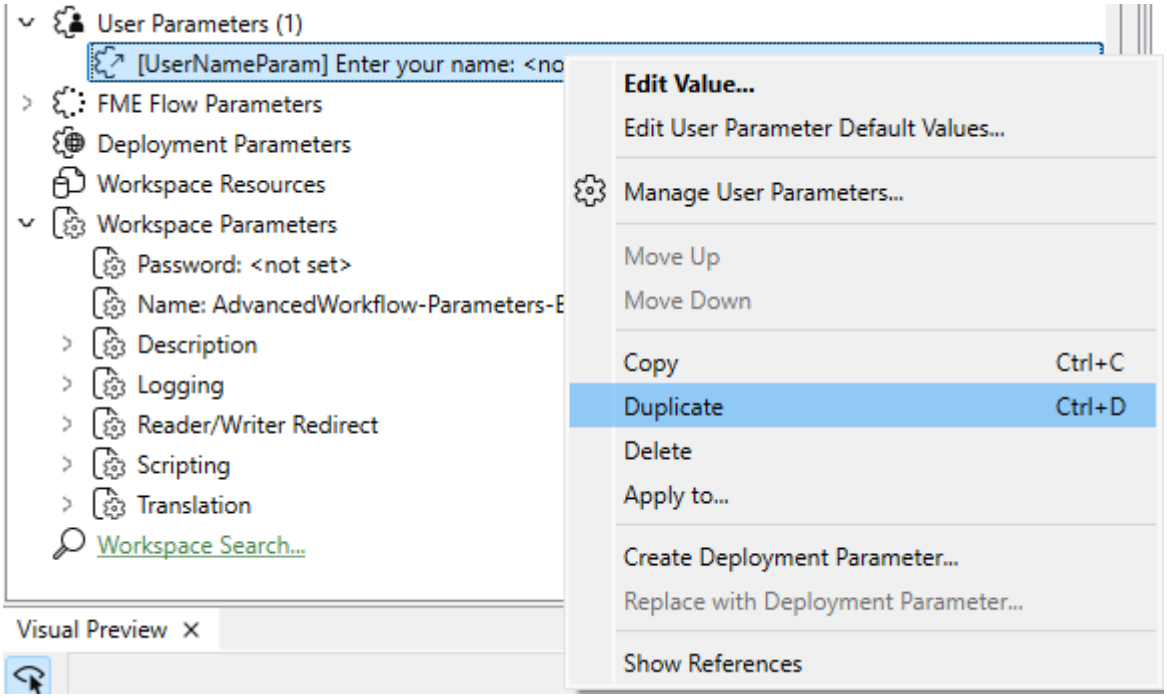
Click *OK* to close the dialog and create the parameter, which now appears in the Navigator window.



7.1.6 Create two more User Parameters

The quickest way to create the other two required parameters (UserMailParam and UserCompanyParam) is to duplicate the UserNameParam parameter.

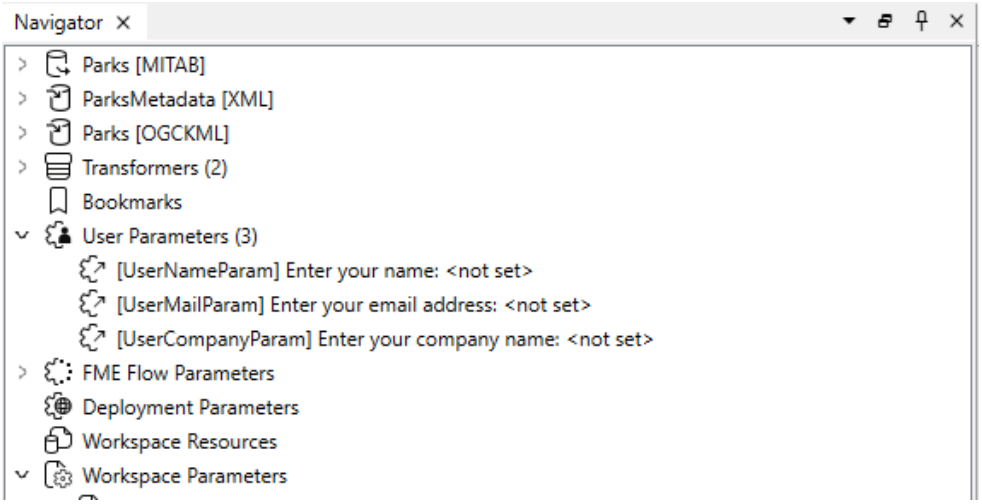
So, right-click on the UserNameParam parameter and choose the option to Duplicate:



A settings dialog for the duplicate parameter will open. Call it *UserMailParam* and set the prompt to "*Enter your email address*".
Then click OK to apply.

Repeat the duplication process, this time creating a parameter called *UserCompanyParam* with the prompt "*Enter your company name*."
Then click OK to apply.

When done the Navigator window looks like this:

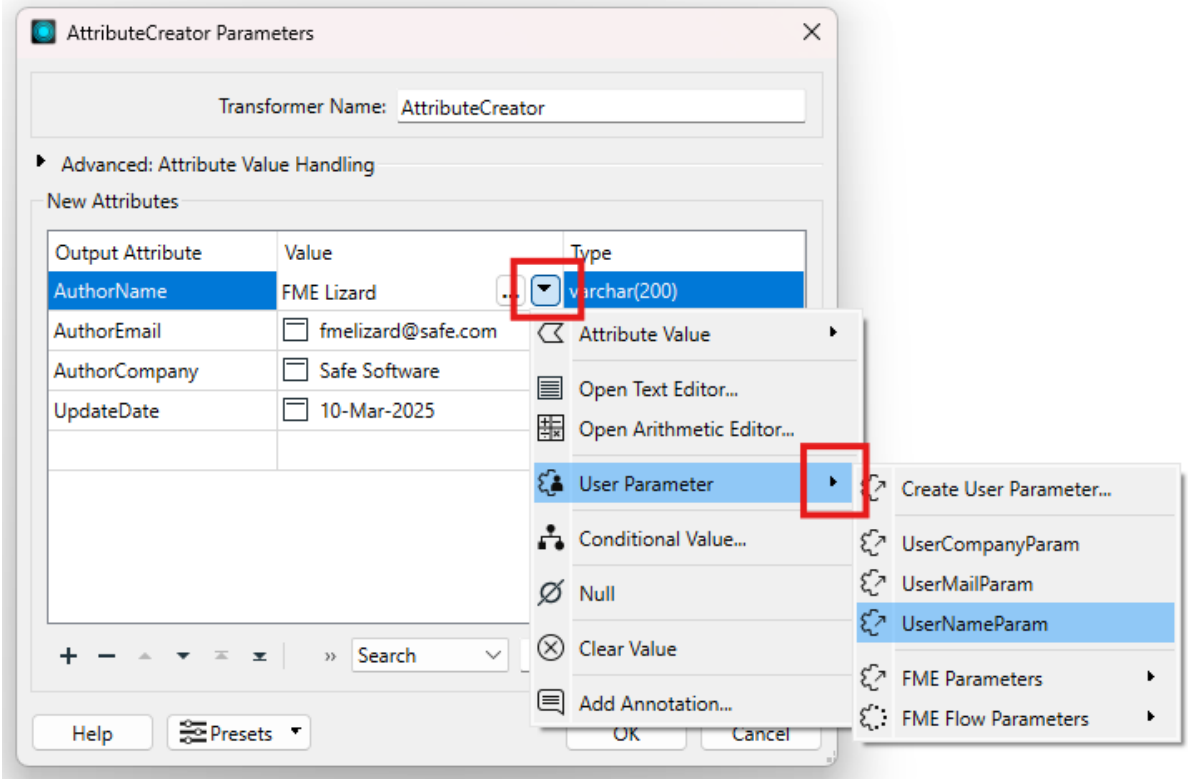


Each of the user parameters we've just defined provides values that need to go into attributes in the writer schema. There are a number of ways to extract the value for such a purpose, and we'll use a different way for each parameter, just to illustrate the different methods.
END OF PART 1

PART 2

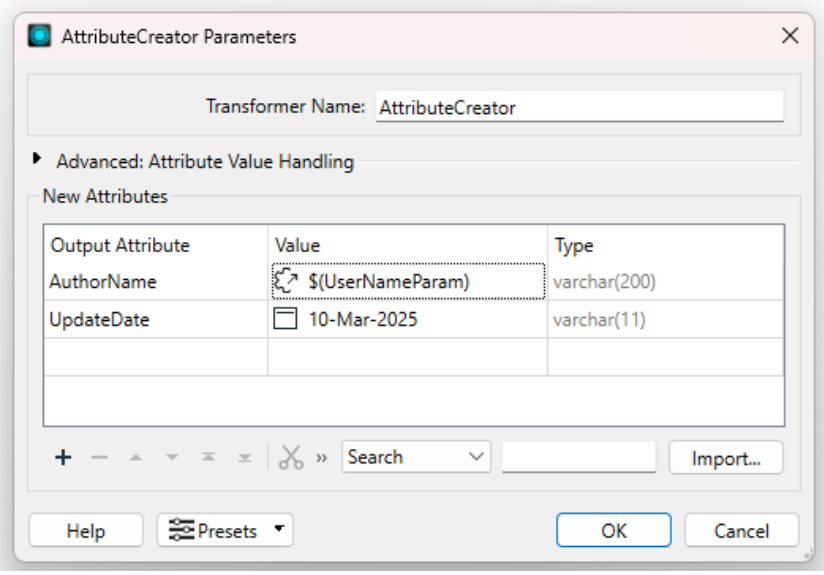
7.1.7 Using a User Parameter - method 1 - within a transformer

So, firstly locate the parameters for the AttributeCreator (either the Parameter Editor window or AttributeCreator Parameters dialog). This transformer is what currently creates the attributes for the output.
Click the Attribute Value field for the *AuthorName* attribute. Click on the drop-down arrow, then select *User Parameter* > *UserNameParam*.



Once done the value field will change to a special icon and show the parameter that was chosen:

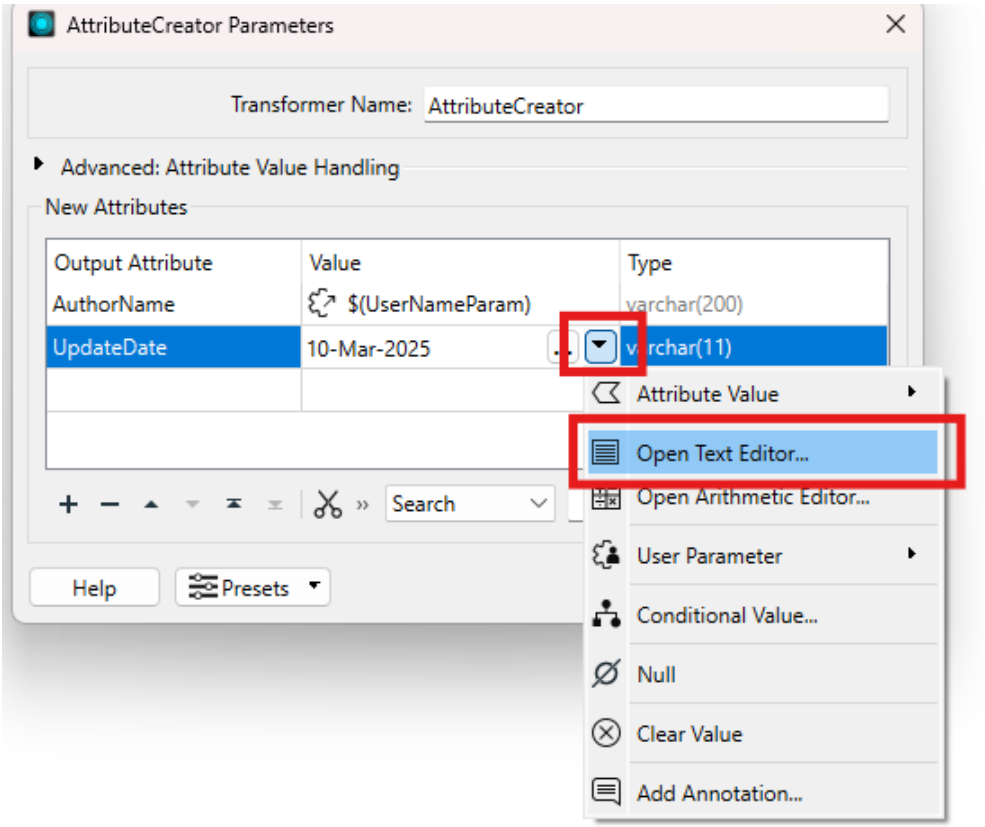
While here, click on the *AuthorEmail* and *AuthorCompany* attributes, and press the minus button to delete them (so that we can demonstrate dealing with these a different way).



7.1.8 Fix the Data Attribute

Looking at the AttributeCreator, we can see that the date field is being entered as a fixed value. Although not a user parameter as such, it's evident that the user must be setting this manually at runtime.
Additionally, the date structure does not conform to an ISO standard, which is why the output fails XML validation.

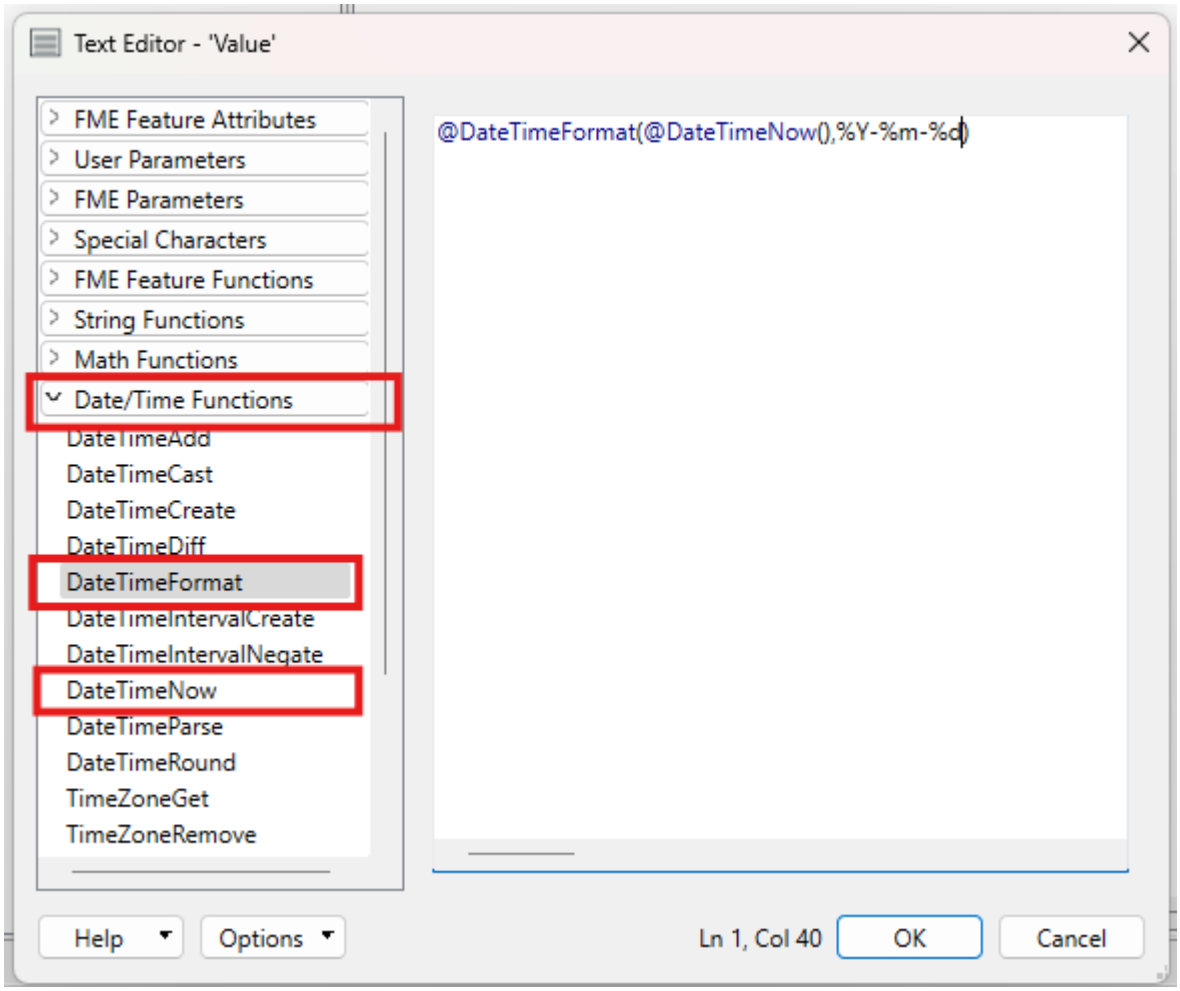
Let's fix these issues. First click on the drop-down arrow next to the UpdateDate Attribute Value field, then choose Open Text Editor:



In the text editor remove any existing content and replace it with:

```
@DateTimeFormat(@DateTimeNow(),%Y-%m-%d)
```

This uses FME Date/Time functions to return today's date in a structure matching the ISO date standard.



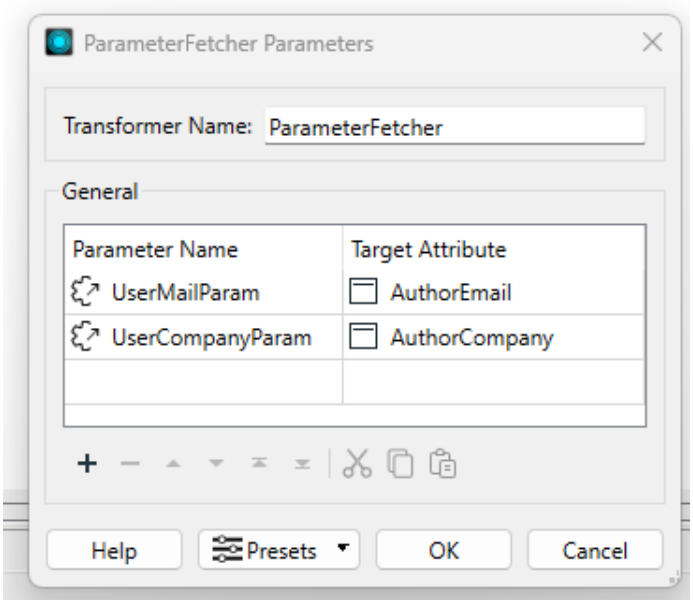
Click OK, and OK again to close the AttributeCreator

7.1.9 Using a User Parameter – method 2 – with a ParameterFetcher

A second way to extract the value from a user parameter is with a ParameterFetcher transformer.

Place a ParameterFetcher transformer (after the AttributeCreator is fine). Inspect the parameters.

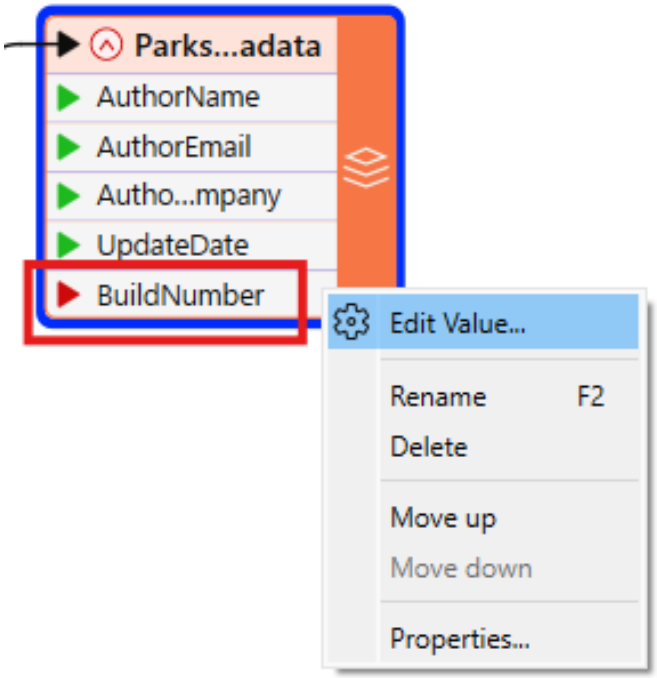
Select *UserMailParam* as the parameter to fetch. Then, enter *AuthorEmail* as the name of the target attribute.
Next, select *UserCompanyParam* and enter *AuthorCompany* as the target attribute:



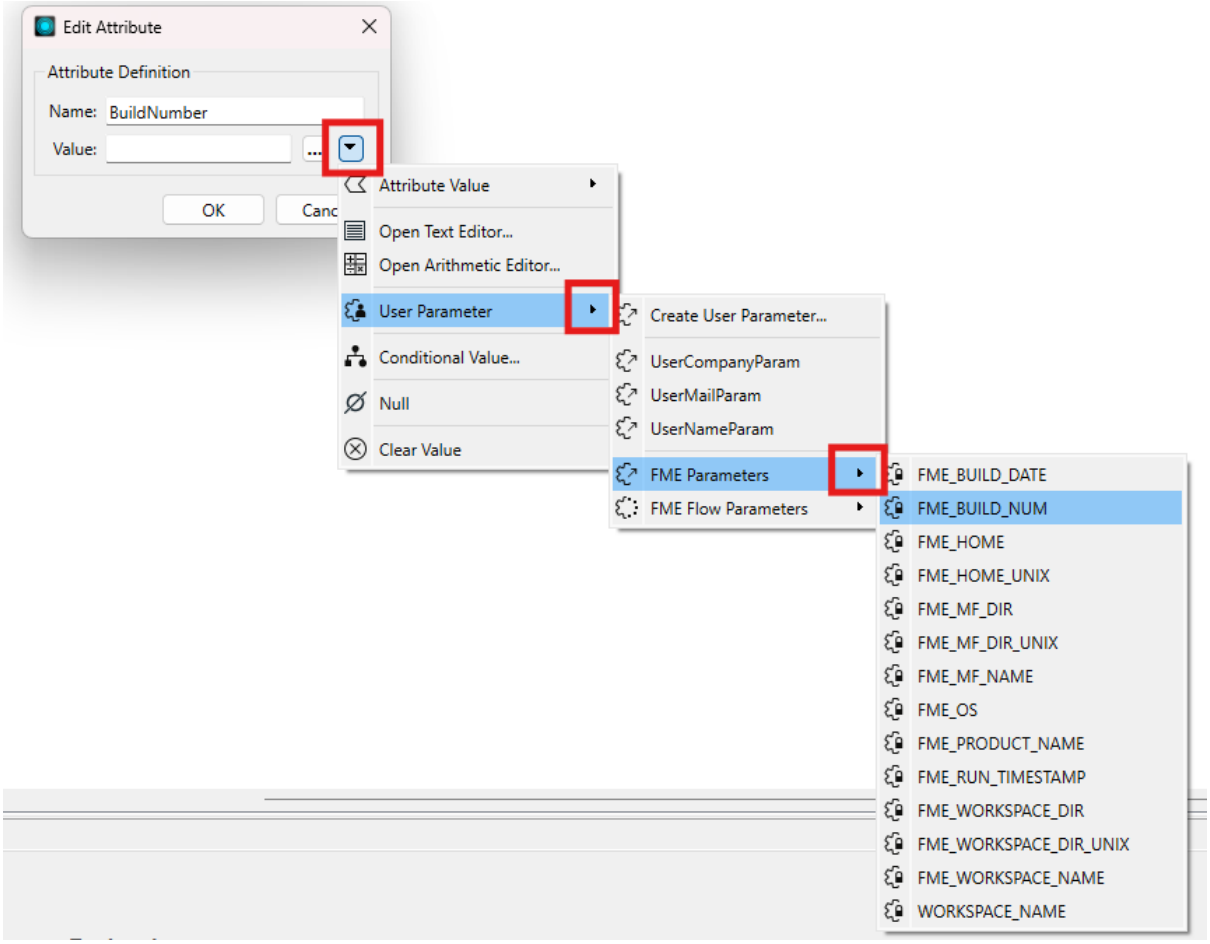
7.1.10 Using a User Parameter – method 3 – on a Feature Type

Did you notice that the list of parameters available includes many FME-related system parameters? These are particularly useful for use with FME Server and for writing metadata.

On the ParksMetadata feature type locate the *BuildNumber* attribute. Right click on that attribute and select the *Edit Value…* option:

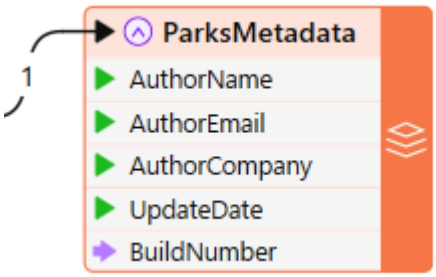


In the dialog that opens, you can enter a fixed (constant) value, but in our case, we'll click on the drop-down arrow, select *User Parameters*, then *FME Parameters*, and then select *FME_BUILD_NUM*:



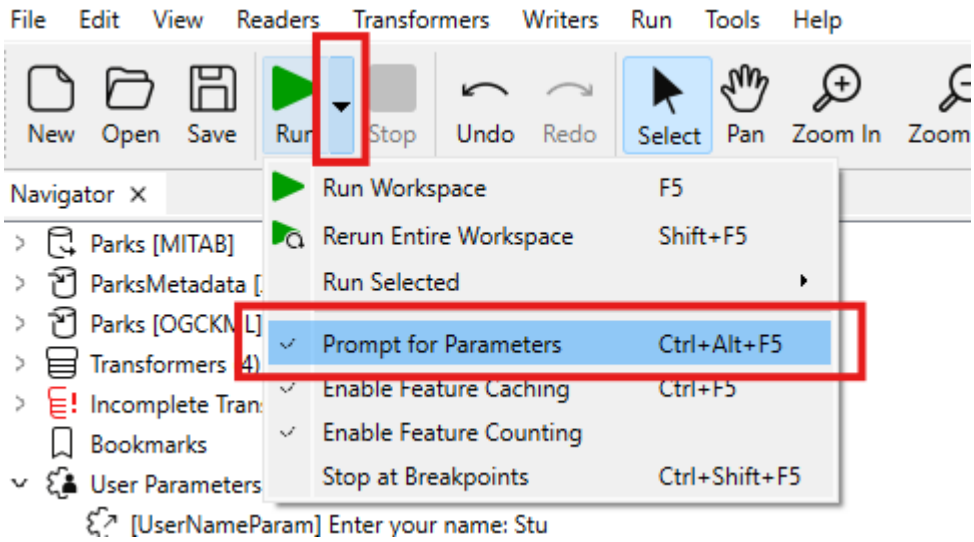
Click OK to close the dialog.

The feature type will highlight the attribute with a light red rightward-pointing arrow to show it has an attribute value set before being written.

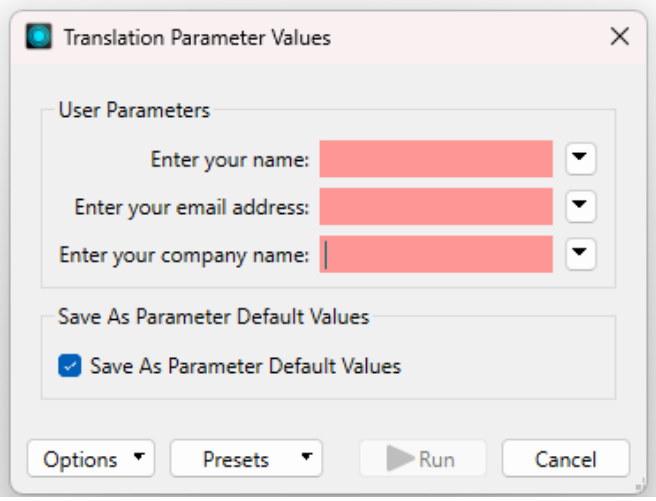


7.1.11 Save and Run the Workspace

Save the workspace and then – as if you were the end-user – run it. Be sure to set the Prompt option on the toolbar first:



When prompted enter your details into the parameters that have been created, then click Run.



Notice that the BuildNumber parameter we created isn't in the prompt. This is because it is an FME-specific private parameter that the user doesn't need to change.

Locate and open the XML file to ensure the contents have been inserted as expected:

```
▼<fme:xml-tables xmlns:fme="http://www.safe.com/xml/xmltables" xmlns:xsi="http://www
  ▼<fme:ParksMetadata-table>
    ▼<fme:ParksMetadata>
       <fme:AuthorName>FME Lizard</fme:AuthorName>
       <fme:AuthorEmail>training@misoportal.com</fme:AuthorEmail>
       <fme:AuthorCompany>miso</fme:AuthorCompany>
       <fme:UpdateDate>2025-03-10</fme:UpdateDate>
       <fme:BuildNumber>24825</fme:BuildNumber>
     </fme:ParksMetadata>
   </fme:ParksMetadata-table>
 </fme:xml-tables>
```

Advanced

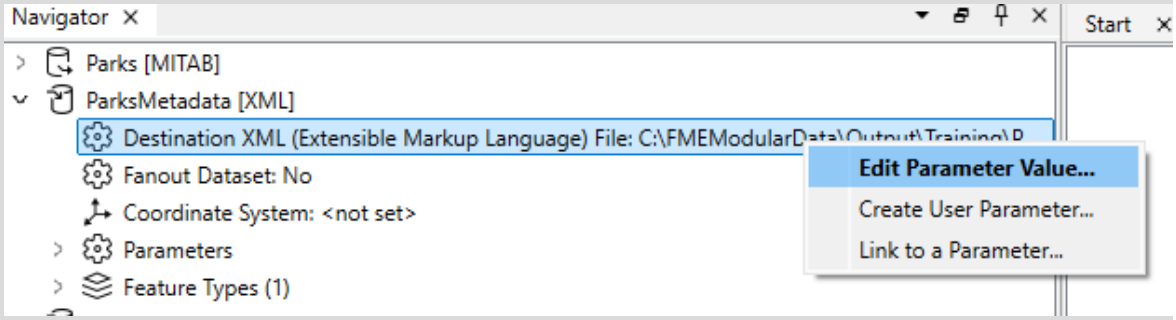
We previously converted destination parameters on both the Writers to be FME parameters, instead of User parameters. This was because there was no requirement for end-users to modify the output locations.

However, they now want to the ability to define the output folder location for the results. Both the KML and XML outputs need to be written to the same folder.

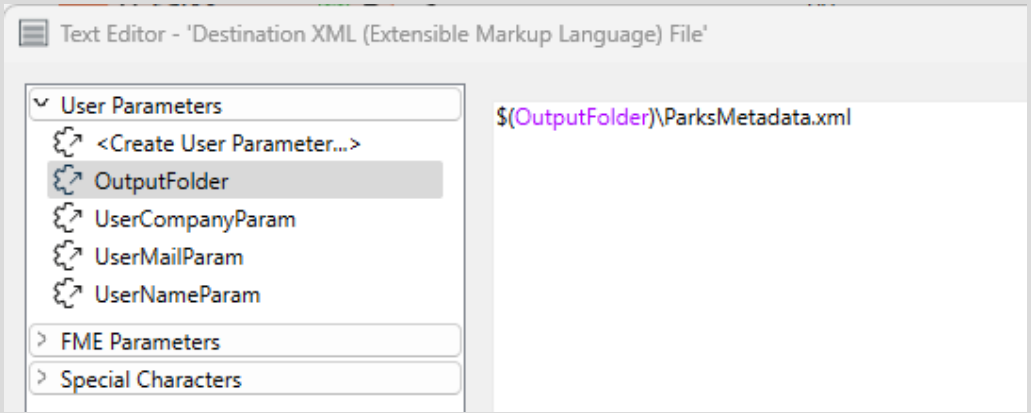
You need to make an '*OutputFolder*' parameter, then utilise it on both the writer destination parameters. So, when the end-user runs the workspace, they only need to specify the desired output folder once.

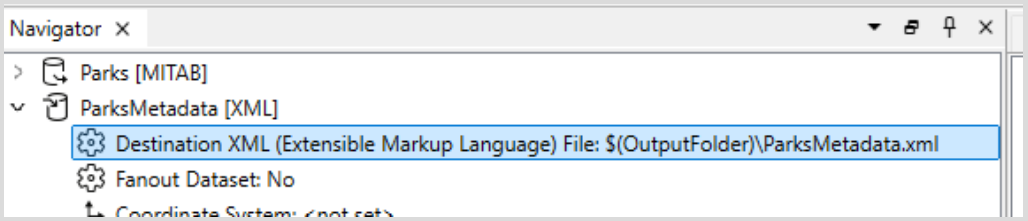
Create a new user parameter of type **File/URL**.
Set the Name to '*OutputFolder*'
In File/URL Configuration set *Items to Select* as '*Folders*' and *Access Mode* to '*Write*'
For the Default Value, browse to *C:\FMEModularData\Output\Training*

In the Navigator panel locate the Destination XML parameter. Right-click on it and choose *Edit Parameter Value...*

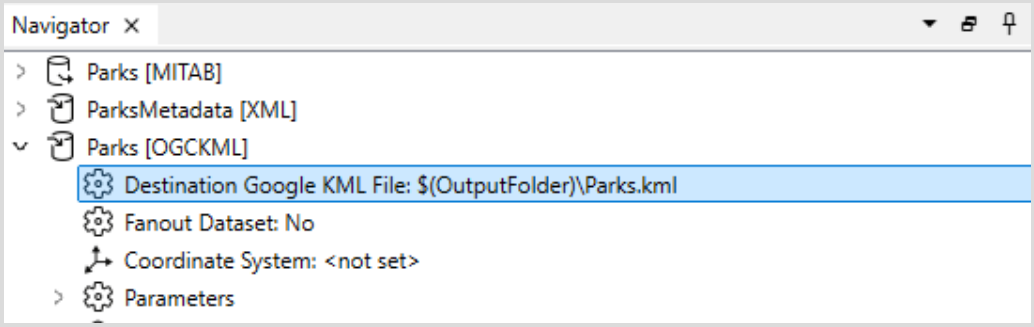


Then use the Text Editor to modify the destination value to utilise the *OutputFolder* parameter, followed by the xml file name:





Repeat the process on the Destination KML parameter:



When you run the workspace, you will be prompted to specify a folder, into which both output datasets will be written.

Congratulations

By Completing this exercise you have learned how to:

- Convert User parameters to FME parameters
- Modify an FME parameter
- Create a Text type User parameter
- Duplicate an existing parameter
- Use a User parameter in a regular transformer
- Use a User parameter in a ParameterFetcher transformer
- Use a User parameter in an Edit Value on a Feature Type

And you may have also learned how to:

- Embed/Concatenate User parameters inside an FME parameter
- Create a File/URL type User parameter

7.2 Generic Writer, plus more User Parameters

Demonstrates	Using a Generic Writer Creating and using Choice User Parameters Using a Choice (with alias) User Parameter to define coordinate systems Publish the Feature Types to Read parameter
Overall Goal	Create a workspace to translate Community Mapping data to a format of the end- user's choice and zip it
Data	Community Mapping (ESRI File Geodatabase)
Start Workspace	none
End Workspace	C:\FMEModularData\Workspaces\Complete\7.02-AdvancedWorkflow-GenericWriter-Complete.fmw

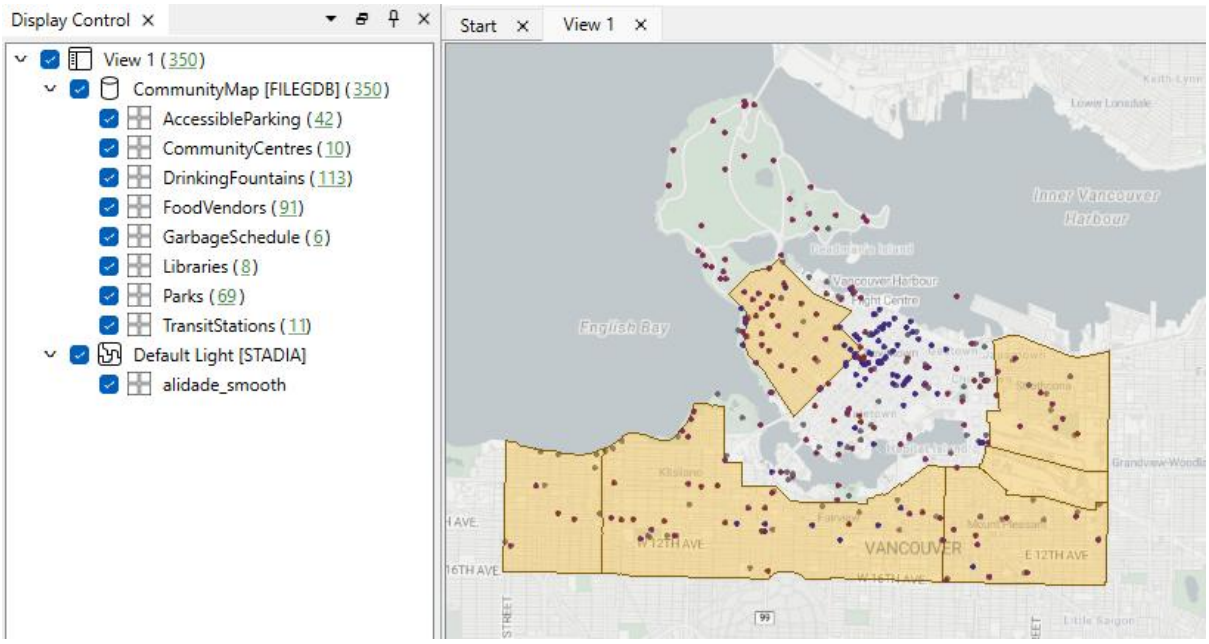
As resident FME expert you are often asked to translate data between formats. You realize that it would be much simpler if you created a workspace to do this - regardless of format - and let the end-users carry out the translation themselves. In the future, this would make an excellent use for an FME Server Data Download service, but for now, we'll let the users simply run the workspace in FME Workbench.

7.2.1 Examine Input Data

Use the Data Inspector to examine the Community Mapping (ESRI File Geodatabase)

Reader Format	ESRI Geodatabase (File Geodb Open API)
Reader Dataset	C:\FMEModularData\Data\CommunityMapping\CommunityMap.gdb

The Community Mapping dataset contains multiple Feature Types:



7.2.2 Launch FME Workbench and Create new Workspace from Blank

Launch the FME Workbench, if it isn't open already.
Within the Get Started section of the Workbench, click on Blank Workspace.

7.2.3 Add Reader

Use either the Reader button, or use Readers > Add Reader… from the menu bar.
The Add Reader dialog will open, in which define the Format and Dataset settings as follows:

Reader Format	ESRI Geodatabase (File Geodb Open API)
Reader Dataset	C:\FMEModularData\Data\CommunityMapping\CommunityMap.gdb
Workflow Options	Single Merged Feature Type

By selecting the single merged feature type option, we will have a workspace that is nice and compact, plus it will allow the user to select which tables they want to read from the source.

Click OK to close the dialog and add the reader.

7.2.4 Add Writer

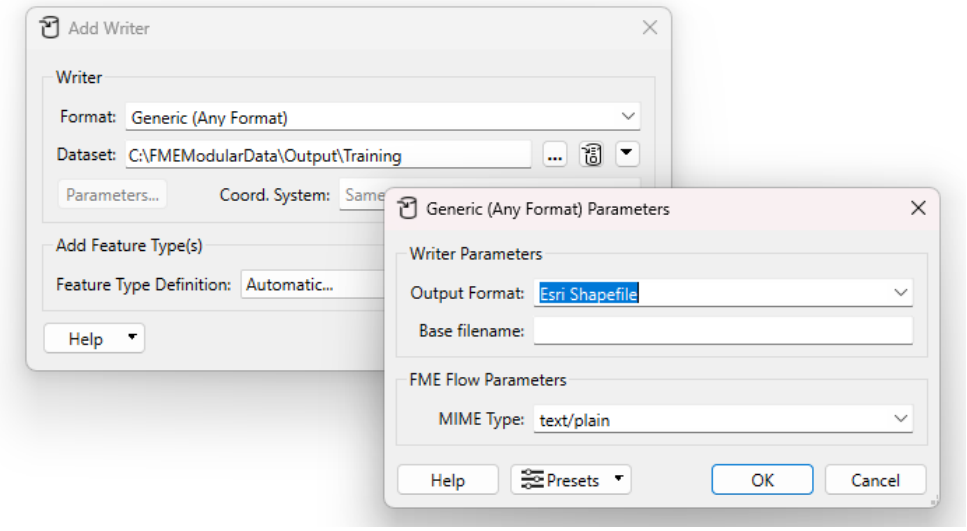
Select Writers > Add Writer from the menu bar and add a Writer:

Writer Format	Generic (Any Format)
Writer Dataset	None
Writer Parameters	Output Format: ESRI Shapefile
Add Feature Types	Feature Type Definition: Automatic

You don't have to select an output location, but for ease latter we will specify the following folder: C:\FMEModularData\Output\Training

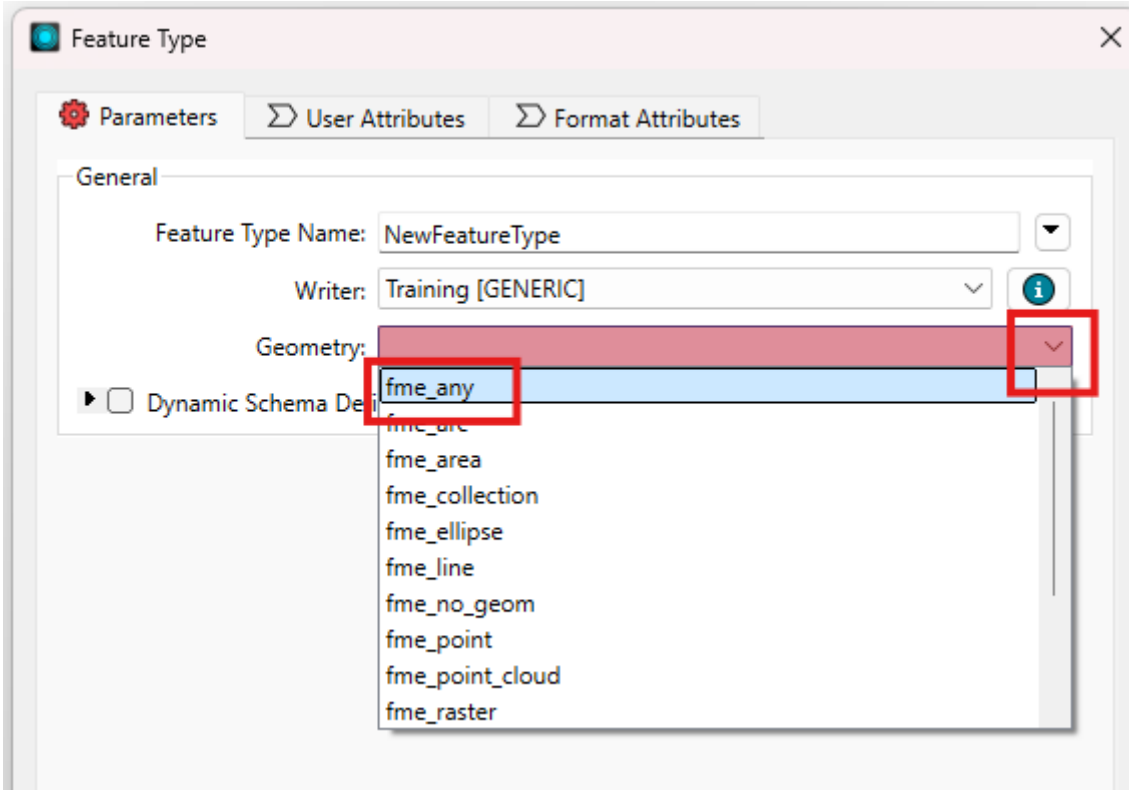
Open the parameters dialog and set an original output format; such as *ESRI Shapefile*.

In the "Add Feature Types" section of the dialog, select *Automatic* for feature type definitions:



Click OK, and the Feature Type Properties dialog for the new writer will open automatically.

Set the Geometry field to *fme_any*. This will allow any data to be written to this feature type:

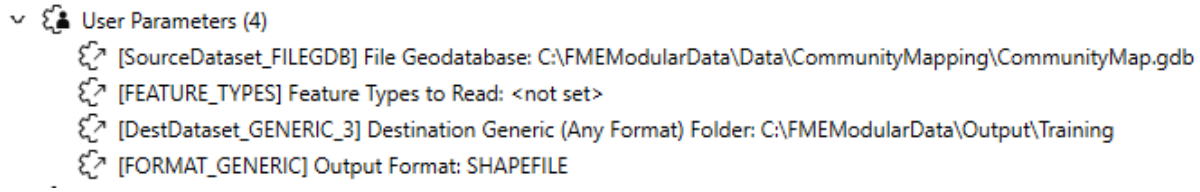


Click OK to close the dialog and to add the new feature type.

Connect it to the source feature type. When you make the connection the attribute schema will automatically be updated to match the connected reader feature type:

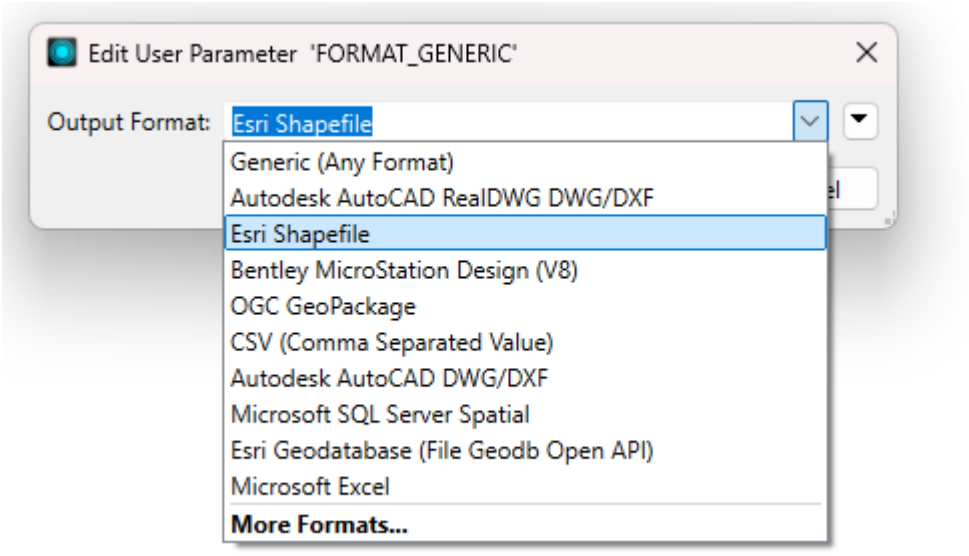
7.2.5 Check the User Parameters

Look in the Navigator window at the user parameters that were created automatically with the reader and writer:



One of the automatically created parameters is called *Feature Types to Read*. This is very useful because when the user runs the workspace, they will be prompted to select which tables to read from the source Geodatabase.

The *Output Format* parameter is interesting. Double-click on it as if you were going to set a value. Notice that the "More Formats...." option in the drop-down list opens up the full FME formats list:



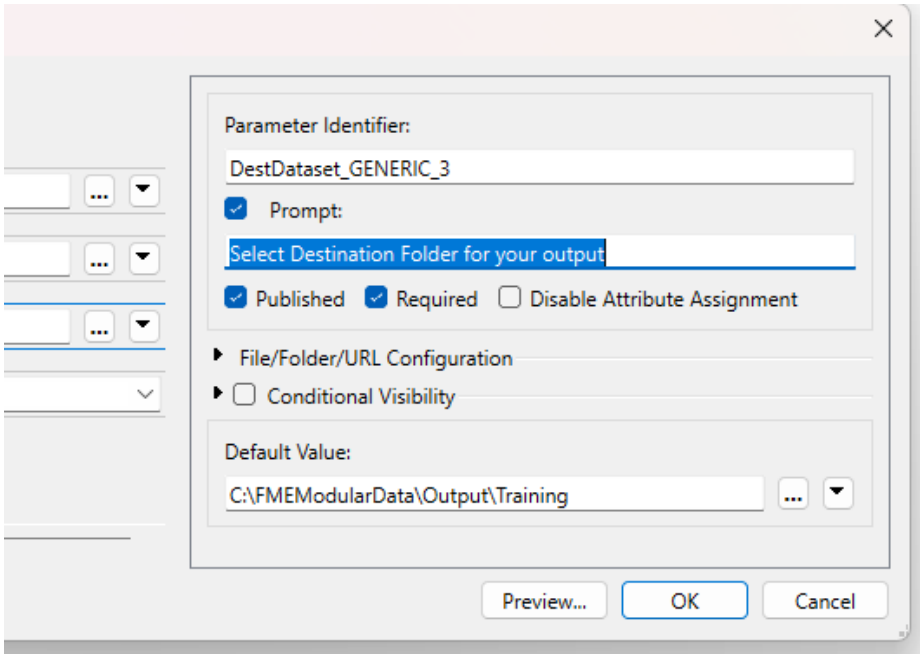
It wouldn't be fair to the end-user to expose so many formats when they don't really need to see or select most of them. It would be better to restrict this list.
So, delete this user parameter, and later we'll create a new - more restrictive - one.

7.2.6 Modify Prompt of Existing User Parameter

The Prompt value is what the end user sees when initiating the workspace run. Edit the Prompt value to help the end user understand what the parameter means.

Right-click on User Parameters and selecting *Manage User Parameters…*

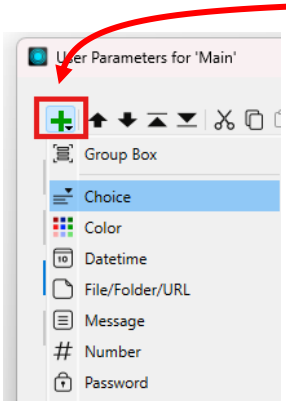
For the User Parameter *Destination Generic (Any Format) Folder*, modify the wording of the Prompt to become: *Select Destination Folder for your output*



7.2.7 Create a Choice User Parameter

We'll now setup a Choice User Parameter to enable users to choose from a predefined list of output formats.

Add a new User Parameter by clicking on the Insert button, then choose the option *Choice*

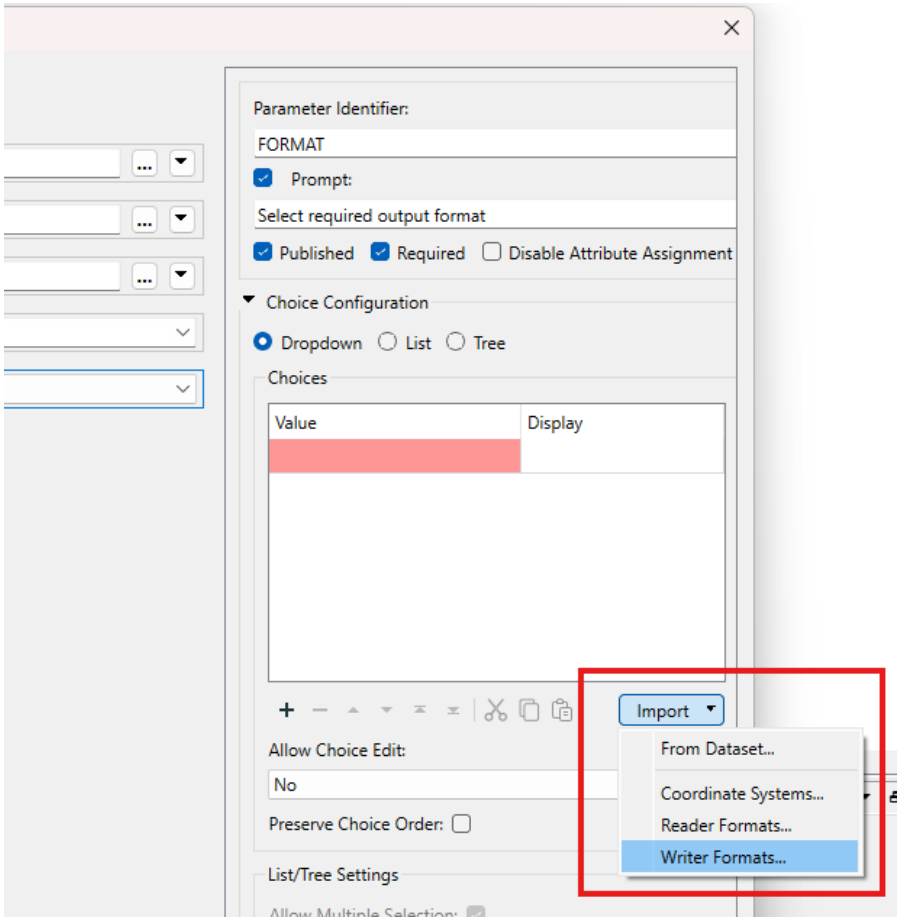


Set the following for the new User Parameter:

Parameter Identifier	FORMAT
Prompt	Select required output format

Published	Yes (checked)
Required	Yes (checked)
Choice Configuration	Drop Down

For the Choice Configuration values, click on *Import*. Then select *Writer Formats…*



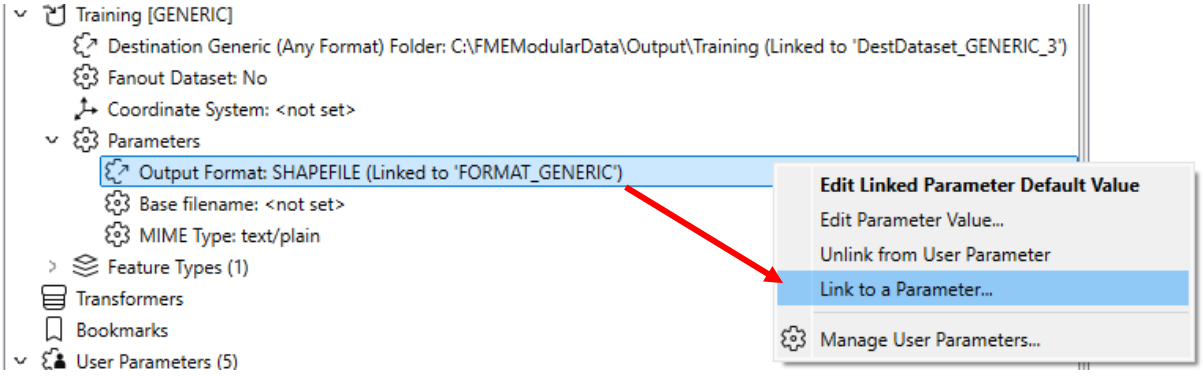
Select a handful of the most common spatial formats such as *ESRI Shapefile, AutoCAD DWG*, and *MapInfo TAB*. Then click OK.

Then click OK twice more until all the dialogs are closed. The new User Parameter will now be listed in the User Parameters section of the Navigator panel.

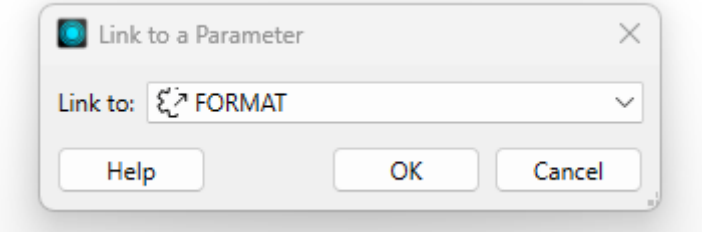
7.2.8 Link Format Parameter

The newly created *Format* User parameter needs to be linked to the Generic Writer's *Output Format* FME parameter.

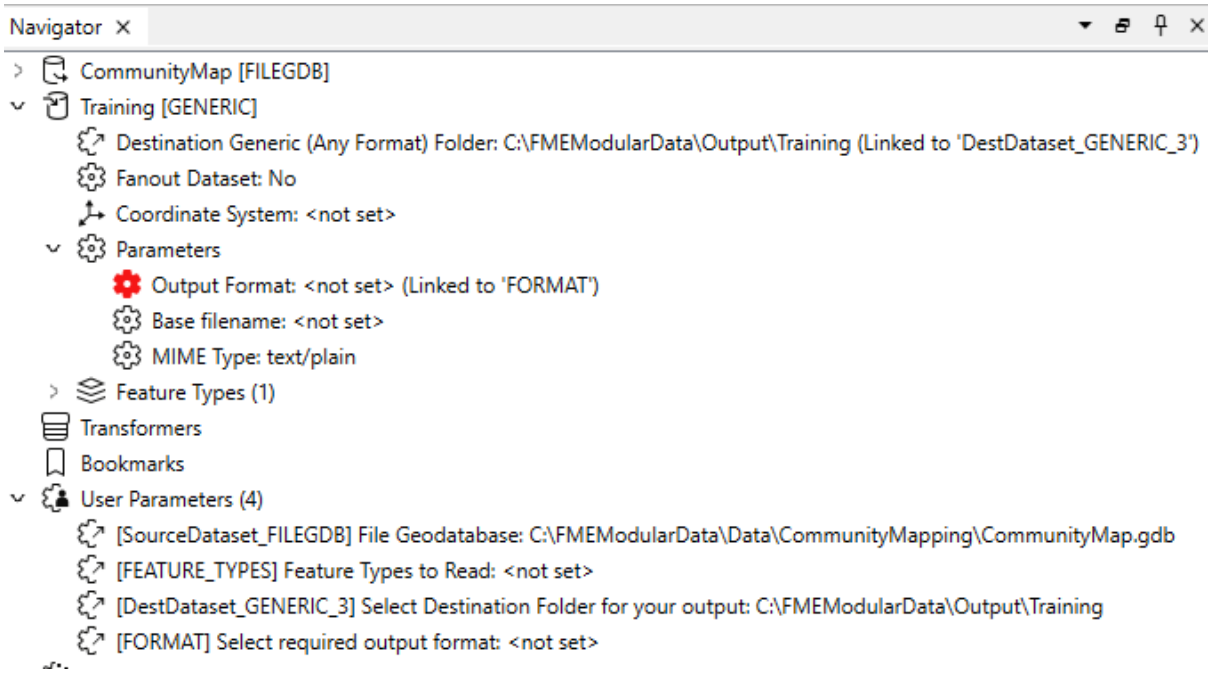
Now, in the Navigator window, expand the Parameters for the Generic Writer. Locate the *Output Format*: parameter. *Right-click* it and choose *Link to User Parameter*.



Select the newly created *FORMAT* parameter and click OK:



The Writer Parameters and User Parameters will update to this:



Don't worry that the parameter cog is red, this is because no value is currently set. But it will get chosen/defined at workspace run-time.

7.2.9 Create User Parameter for Coordinate System Choice

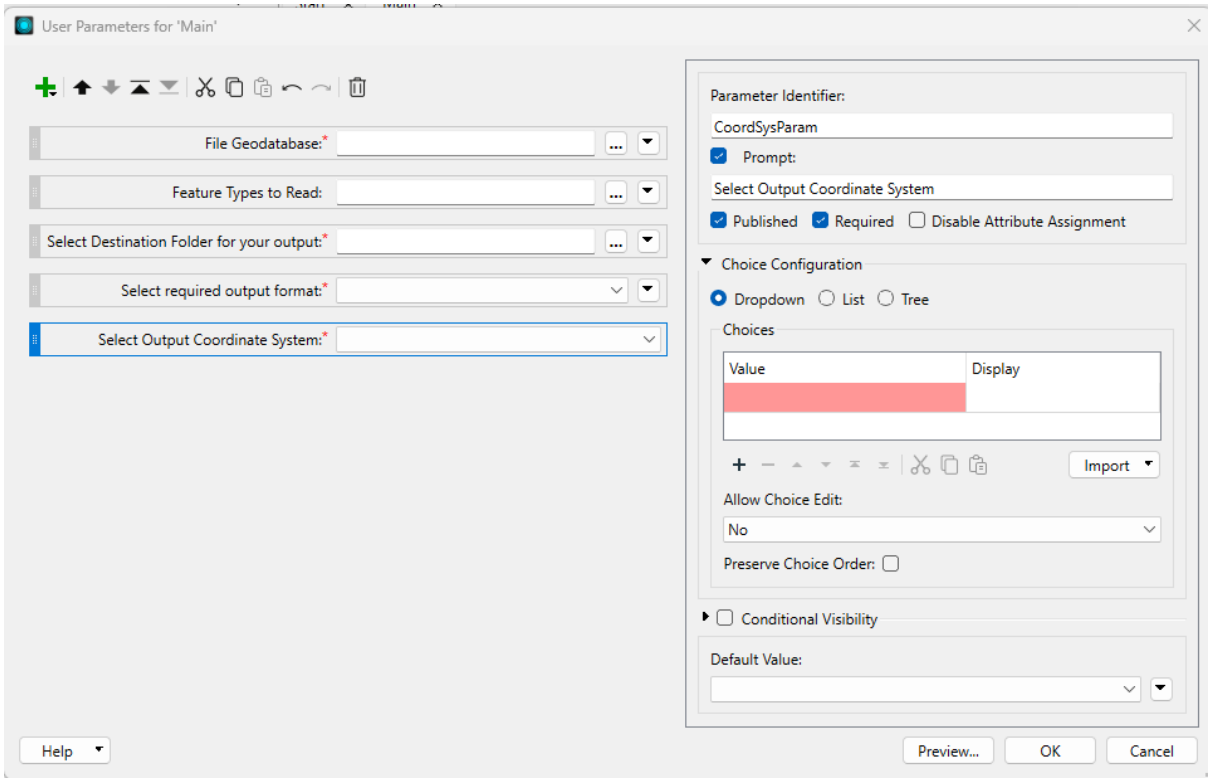
Another requirement, is an ability to set the output coordinate system. However, if you simply publish the writer's coordinate system parameter – try it and see – then there will be a problem. The parameter will allow the end-user to select ANY coordinate system supported by FME.

This is not necessarily very useful. Since the data is located in Vancouver, BC, it makes little sense for the user to be able to reproject it to (for example) NZMG (a New Zealand coordinate system).

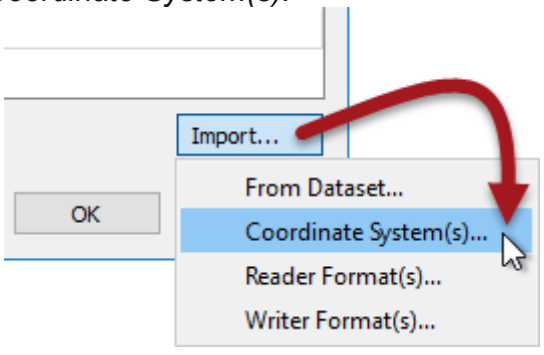
It would be preferable if the parameter only allowed the end-user to select a coordinate system from a smaller list.

Now create a new user parameter; right-click User Parameters > Manage User Parameters..., and add a new parameter and set the Type to be ***Choice***.

Parameter Identifier	*CoordSysParam*
Prompt	*Select Output Coordinate System*
Published	Yes (checked)
Required	Yes (checked)
Choice Configuration	Drop Down



Click on the button labelled *Import*, and choose *Coordinate System(s):*

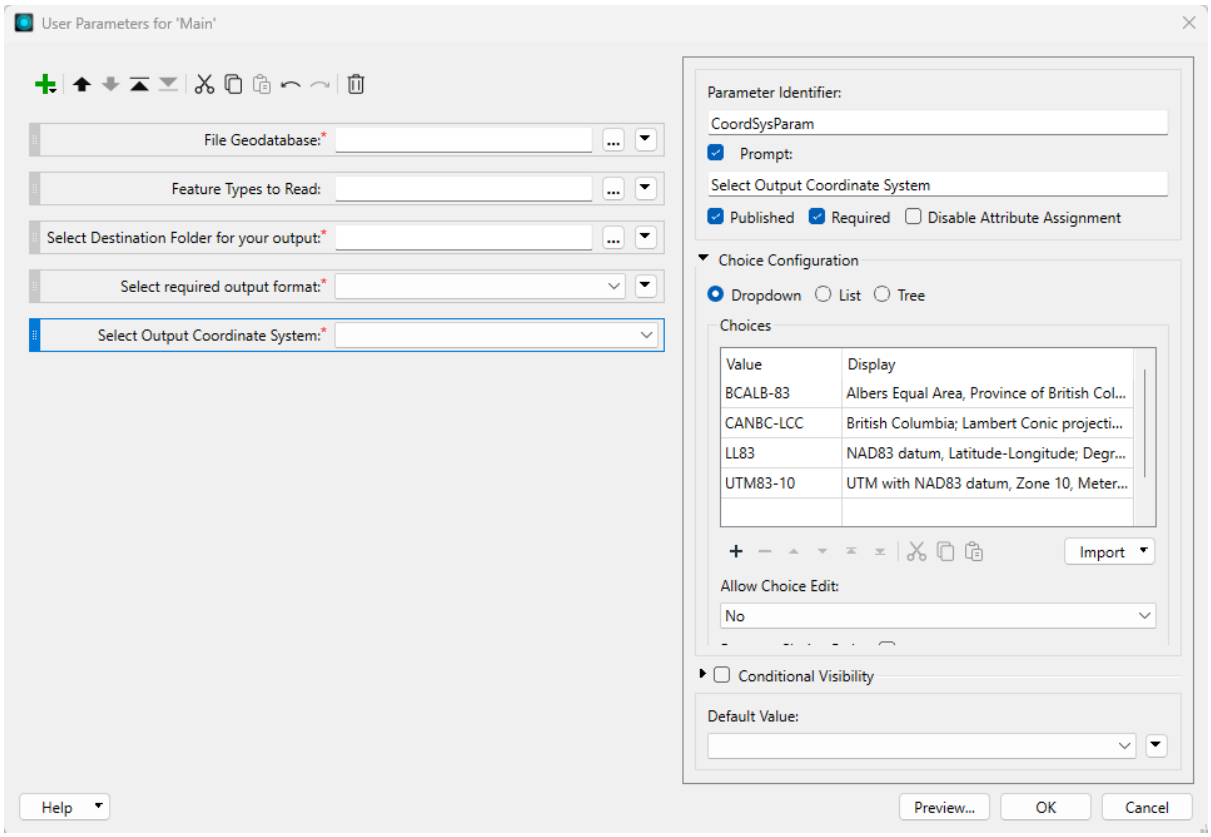


This opens a list of coordinate systems that we can import as values in our user parameter.

Locate and put a checkmark in the box for the following coordinate systems:

- UTM83-10
- BCALB-83
- LL83
- CANBC-LCC

Then click OK to close this dialog. You will be returned to the configuration dialog and find that names and values have been automatically entered for these coordinate systems:

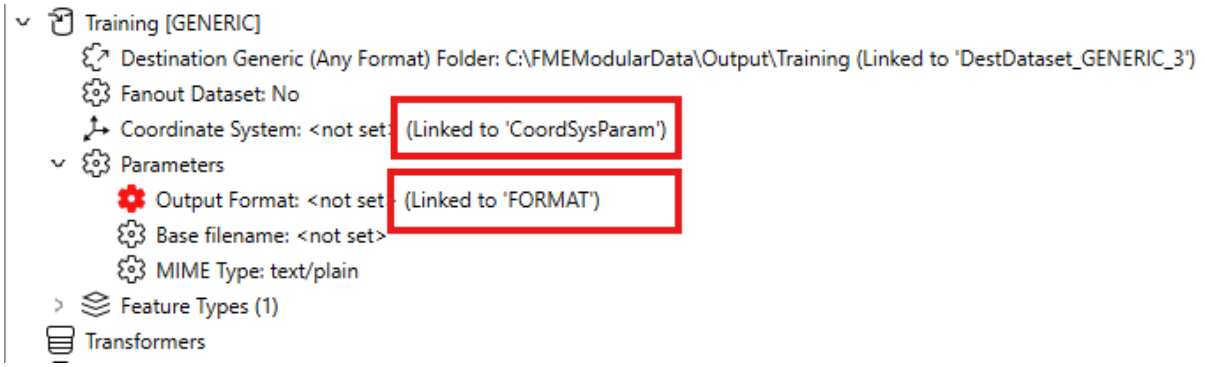


The right-hand side shows what the user is prompted to select, the left-hand side what the value fed to FME will be.

Click OK and then OK again to close the remaining dialogs and create the user parameter.

7.2.10 Link Coordinate System Parameter

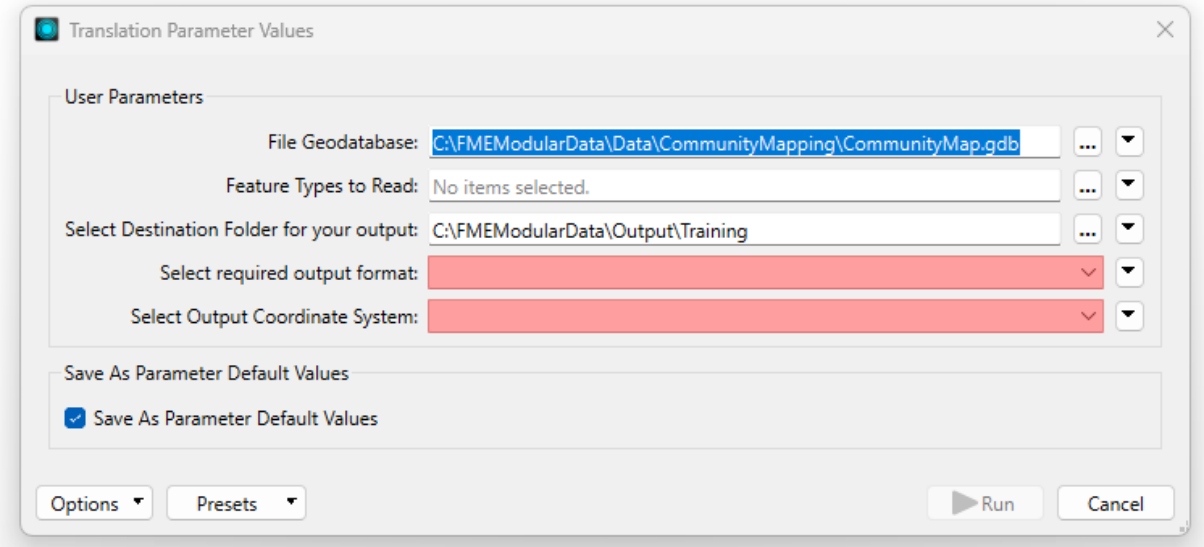
Now we have the user's selection, but we still have to apply it to the FME parameter on the Writer.
Locate the writer's coordinate system parameter, right-click on it, and choose *Link to User Parameter*.
When prompted, select the newly created *CoordSysParam* and click OK.



Now when the workspace is run the user is prompted to select a coordinate system, and that system's short name value is passed to FME.

7.2.11 Save and Run the Workspace

Save the workspace and then – as if you were the end-user – run it. (ensure the *Prompt for User Parameters* option is enabled on the Run menu).

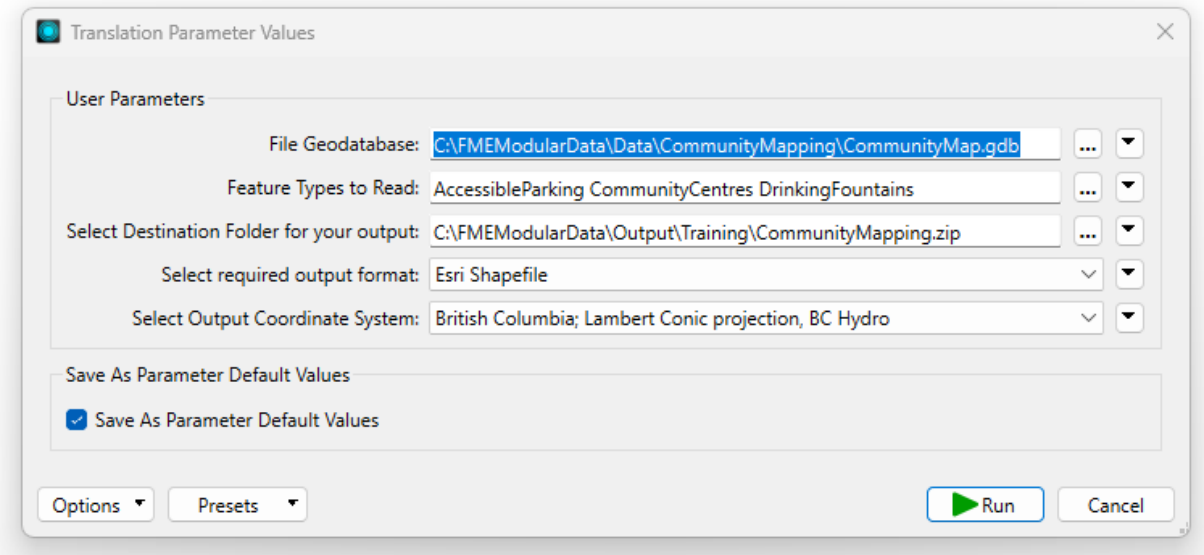


When prompted, select:

- A number of Feature Types to Read (include at least three)
- An output format and coordinate system
- Then for Destination Folder, we could simply select a folder location (e.g. C:\FMEModularData\Output\Training). But instead, we will create a zipped folder to contain all of our files. Type the following at the end of C:\FMEModularData\Output\Training: “CommunityMapping.zip”

```
\CommunityMapping.zip
```

This will create a zipped file called CommunityMapping



7.2.12 Examine the Output

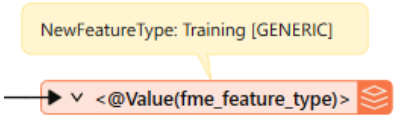
All the features of the chosen source geodatabase 'Feature Types to Read' (AccessibleParking, CommunityCentres, DrinkingFountains etc) have been written to the same single output feature type.

In this scenario it would be more useful to output the features based on their original feature type (AccessibleParking, CommunityCentres, DrinkingFountains etc)

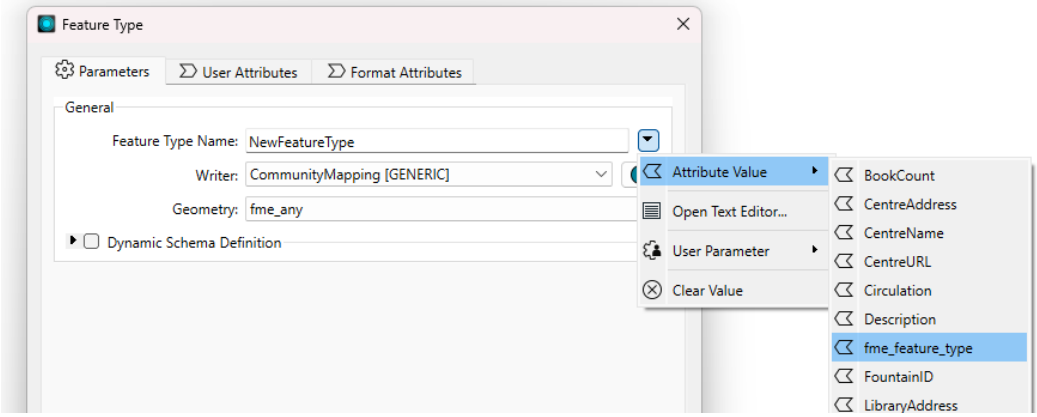
7.2.13 Set Fanout on Feature Type

What we can do here is to output the features based on their original table.
To do this we need to know where they came from, and that is obtained from a format attribute called *fme_feature_type*.

Inspect the properties for the Writer Feature Type.



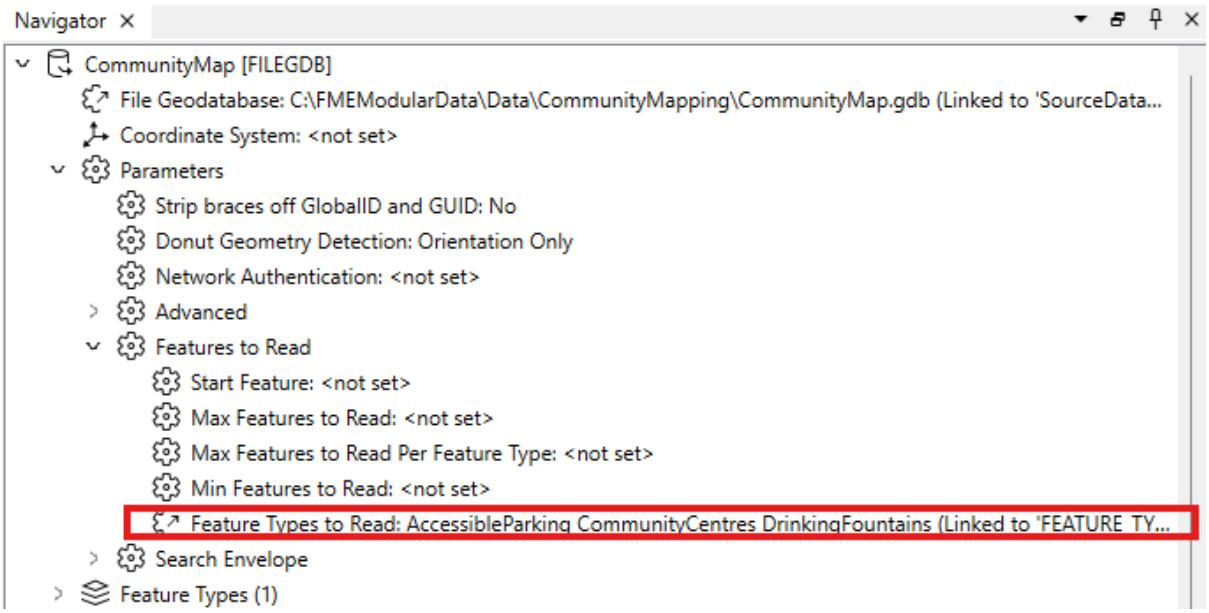
Set a Fanout by using the drop-down arrow and choosing *fme_feature_type* as the attribute supplying the feature type name.



7.2.14 Feature Types to Read parameter

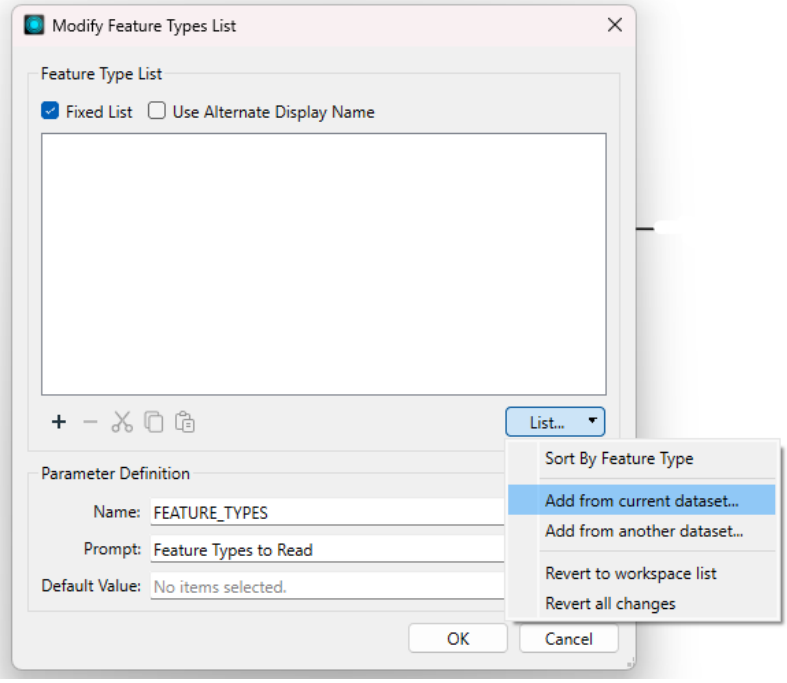
There's one final amendment to make. It's been decided that the Garbage Schedule data should not be available to end-users to output.

Locate the *Feature Types to Read* parameter in the CommunityMap Geodatabase Reader in the Navigator panel:

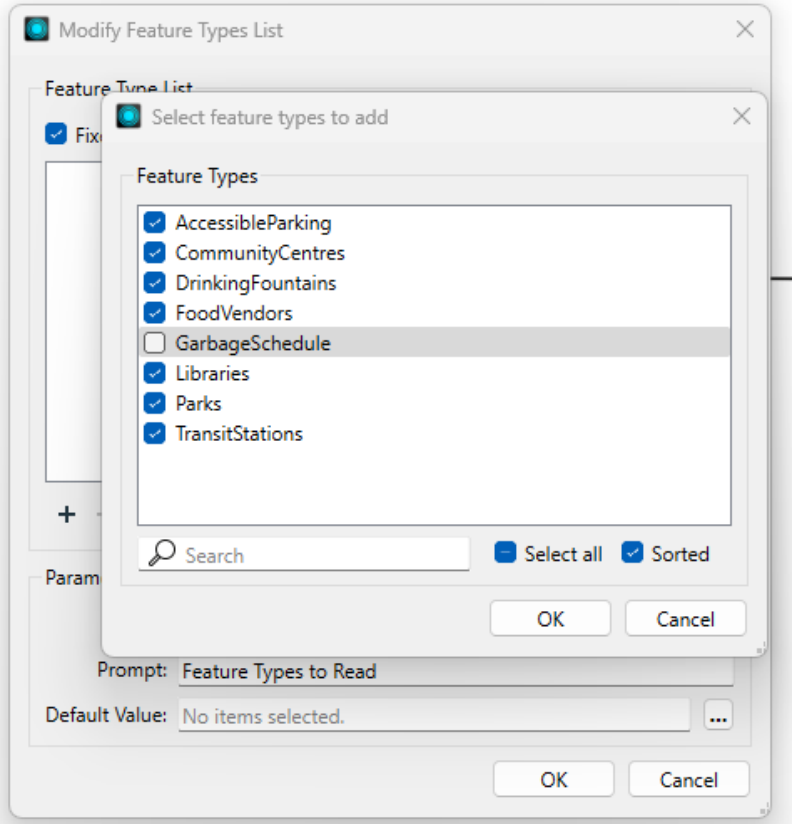


Right-click on it and choose *Edit Feature Types to Read Definition...*

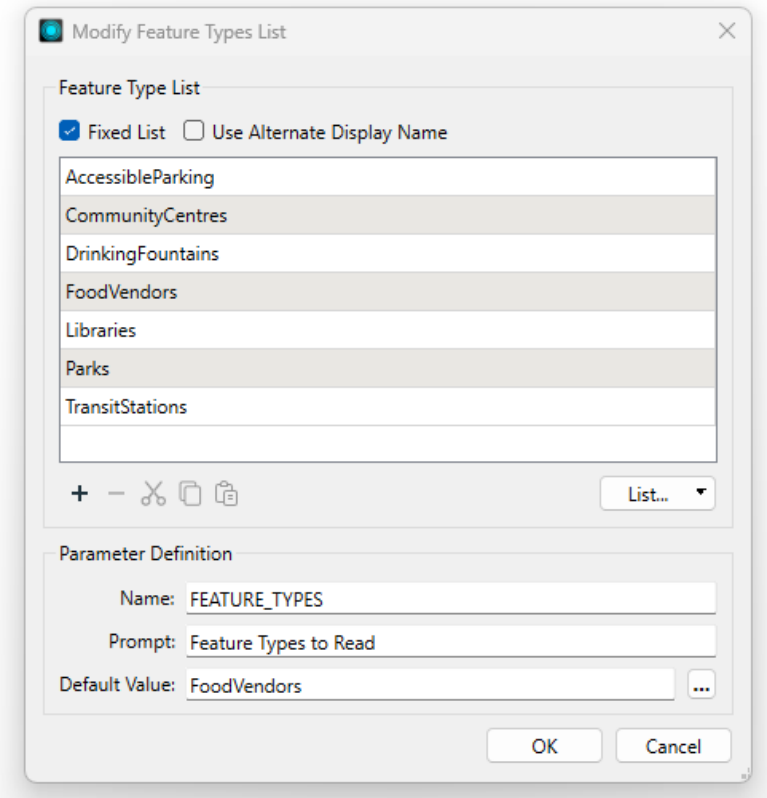
A dialog will open, in which tick *Fixed List*. (Notice here is also an option to set alternative/display names for the feature types, which can be handy in some circumstances). Then click on the *List*…button and choose *add from current dataset....*



Select all feature types **except** for *GarbageSchedule*:



Click OK. Then set a Default Value (doesn't matter which, as the end-user will change as needed).



End-users will now only be able to choose to read feature types from the specified list (i.e. not Garbage Schedule).

7.2.15 Resave and Rerun the Workspace

Resave the workspace and then rerun the workspace, using Run > Rerun Entire Workspace.

7.2.16 Examine the Output

The results zip file will contain separate MapInfo TAB or ESRI Shapefiles for each feature type selected at workspace run-time (due to the Fanout function).

A combination of a Generic Writer and User Parameters has enabled end-users to choose a desired output format at the time of running the workspace (along with choosing the coordinate system, destination folder and the source feature types to include).

Now you have a solution that almost anyone will be able to open and run for themselves. Also, if you published the workspace to FME Server it would automatically create a web page that matched the different user parameters.

FME Lizard

Did you notice that FME handled the different geometry types and output the files with the geometry as part of the name? It's a Shapefile format thing. FME can never – and will never – write more than one geometry type to the same Shapefile .shp file

*The one drawback with the output is that **each output dataset has all of the attributes of all of the source tables**. To avoid that you would need to use a dynamic translation, as we shall see shortly.....*

7.2.17 Examine the Attributes

Examine the attributes of the outputs datasets and compare to the original source feature types. Attributes of each source feature type:

CommunityMap [FILEGDB] - AccessibleParking

	Description	Location	Regulation	OBJECTID
1	Parking meter	North side 100...	<null>	1
2	2 hour zone	East side 700 H...	<null>	2
3	Parking meter	West side 700 ...	<null>	3

CommunityMap [FILEGDB] - CommunityCentres

	CentreName	CentreAddress	CentreURL	OBJECTID
1	Mount Pleasant	1 Kingsway	http://vancouve...	1
2	Strathcona	601 Keefer St	http://vancouve...	2
3	West End	870 Denman St	http://vancouve...	3

CommunityMap [FILEGDB] - DrinkingFountains

	FountainID	Location	Maintainer	OBJECTID
1	1	...	Engineering	1
2	2	...	Engineering	2
3	3	...	Engineering	3

CommunityMap [FILEGDB] - FoodVendors

	VendorID	VendorName	VendorDescription	OBJECTID
1	1	Aussie Pie Guy	Australian Pies	1
2	3	Chou Chou Crepes	French Crepes	2
3	4	Culver City Salads	Salads	3

CommunityMap [FILEGDB] - Libraries

	LibraryName	LibraryAddress	LibraryURL	BookCount	UserVisits	Circulation	OBJECTID	LibraryID
1	Strathcona	592 E Pender St	http://www....	17394	89007	49025	1	BVA020
2	Carnegie	401 Main St	http://www....	26689	418069	184466	2	BVA002
3	Central Branch	350 W Georgia St	http://www....	1188303	1958528	1751704	3	BVA003

CommunityMap [FILEGDB] - Parks

	ParkName	ParkAddress	ParkURL	OBJECTID
1	Alexandra Park	1755 Beach Av	http://www.city....	1
2	Almond Park	3600 W 12th Av	http://www.city....	2
3	Andy Livingsto...	89 Expo Boulevard	http://www.city....	3

CommunityMap [FILEGDB] - TransitStations

	StationName	OBJECTID
1	Waterfront	1
2	Burrard	2
3	Granville	3

CommunityMap [FILEGDB] - GarbageSchedule

	Zone	Subzone	Schedule	NumAddresses	OBJECTID
1	Purple	North	Monday	1245	1
2	Red	North	Wednesday	2827	2
3	Purple	South	Thursday	60	3

Congratulations

By Completing this exercise, you have learned how to:

- Use Generic Writer
- Remove pre-linked parameters
- Create a Choice User parameter that prompts for a writer format
- Link a User parameter to a Writer's FME parameter
- Use a Choice parameter to define Coordinate Systems
- Manage the Feature Types to Read parameter
- Use a feature type Fanout on fme_feature_type to return features to their original layer
- Use a .zip extension to create zipped output datasets

7.3 Dynamic Translation

Demonstrates	Using the Dynamic Schema workflow option
Overall Goal	Create a dynamic workspace to translate any Geodatabase dataset to a format of the end-user's choice
Data	Community Mapping (Esri File Geodatabase)
Start Workspace	none
End Workspace	C:\FMEModularData\Workspaces\Complete\7.03-AdvancedWorkflow-DynamicTranslation-Complete.fmw

In the previous exercise, a workspace was generated to translate a Geodatabase dataset into a number of formats using the Generic Writer.

However, that workspace had a limitation with the output attributes (every output dataset got all of the source table attributes), and you also feel it would be useful if that workspace could handle any source Geodatabase, not just the community maps dataset.

So, let's create a new workspace to handle that scenario.

7.3.1 Generate Workspace using Dynamic Schema workflow option

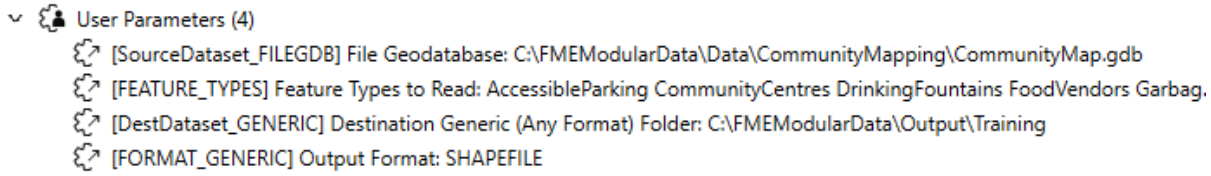
Launch the FME Workbench, if it isn't open already. Within the Get Started section choose the ***Generate*** wizard option.

Reader Format	ESRI Geodatabase (File Geodatabase Open API)
Reader Dataset	C:\FMEModularData\Data\CommunityMapping\CommunityMap.gdb
Writer Format	Generic (Any Format)
Writer Dataset	C:\FMEModularData\Output\Training
Writer Parameters	Output Format: ESRI Shapefile
Workflow Options	Dynamic Schema

The dynamic schema parameter is important in order to handle all aspects of the source schema for a variety of datasets.

Inspect the newly created workspace. There is one reader feature type and one writer feature type.

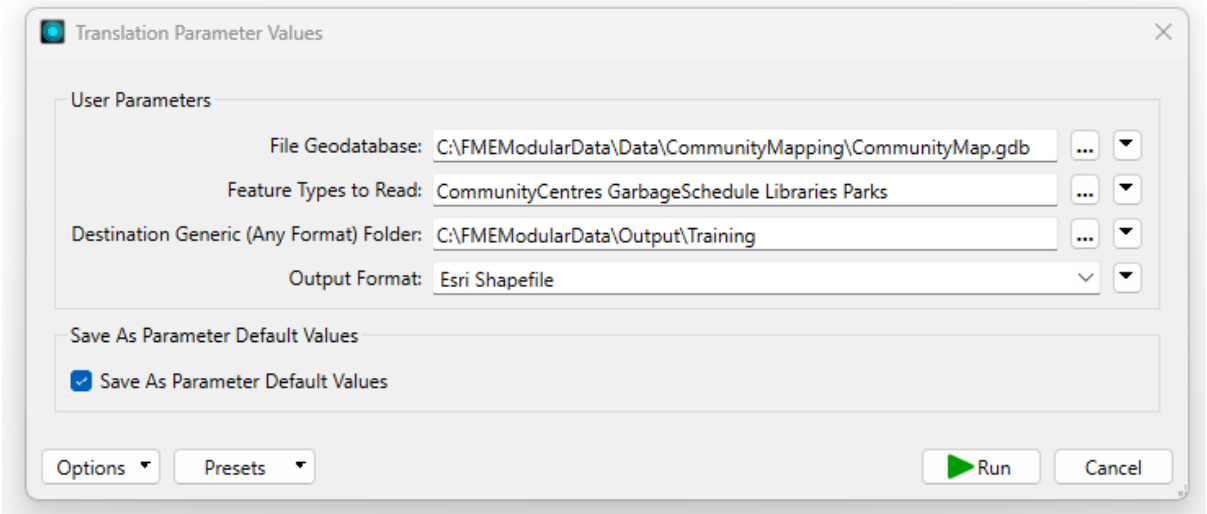
The reader feature type shows a list of attributes, but the writer feature type doesn't. It is, however, labelled <*Dynamic*>.
There will be a user parameter for the Feature Types to Read and the output format.



7.3.2 Run the Workspace

Run the workspace (use the option Rerun Entire Workspace).

When prompted, select a number of source Feature Types (CommunityCentres, GarbageSchedule, Libraries, Parks etc).
Also choose an output format.



The workspace will run to completion. Check the output to ensure it is all correct.

Each output dataset will only contain attributes specific to its source feature type. *See step 7.2.17 of the previous exercise for details of attributes within each feature type.*

We can see that a Dynamic workspace is capable of handling any source schema and writing it out to a new dataset just as it was in the source data.

Congratulations

By Completing this exercise, you have learned how to:

- Create a Dynamic workspace
- Use a Generic Writer in a dynamic workspace

7.4 Dynamic Schema Handing

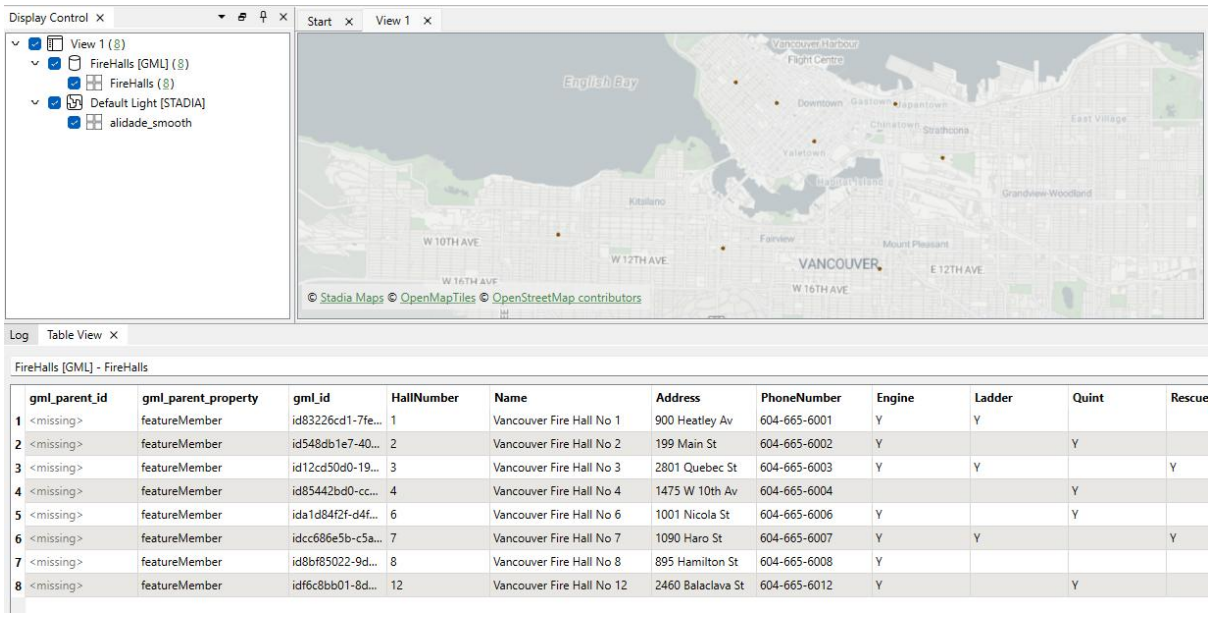
Demonstrates	Creating a Dynamic workspace with multiple readers. Make use of schema in another dataset using add a Resource Reader. Manage Schema Source of writer feature types. Add & remove hard-coded attributes in a dynamic workspace.
Overall Goal	Create a workspace to process two data sources and dynamically handle the schemas; including make use of a schema from another dataset.
Data	Firehalls (GML) Parks (MapInfo TAB) *Schema for Parks = CommunityMap (ESRI Geodatabase)*
Start Workspace	none
End Workspace	C:\FMEModularData\Workspaces\Complete\7.04-AdvancedWorkflow-DynamicSchema-Complete.fmw

For emergency response planning purposes, a new workspace is required to process two source datasets: Parks (MapInfo) and Firehalls (GML). There is an added complication, the workspace needs to handle schemas dynamically. This includes using another dataset to define the schema for the Parks.

7.4.1 Examine the Datasets

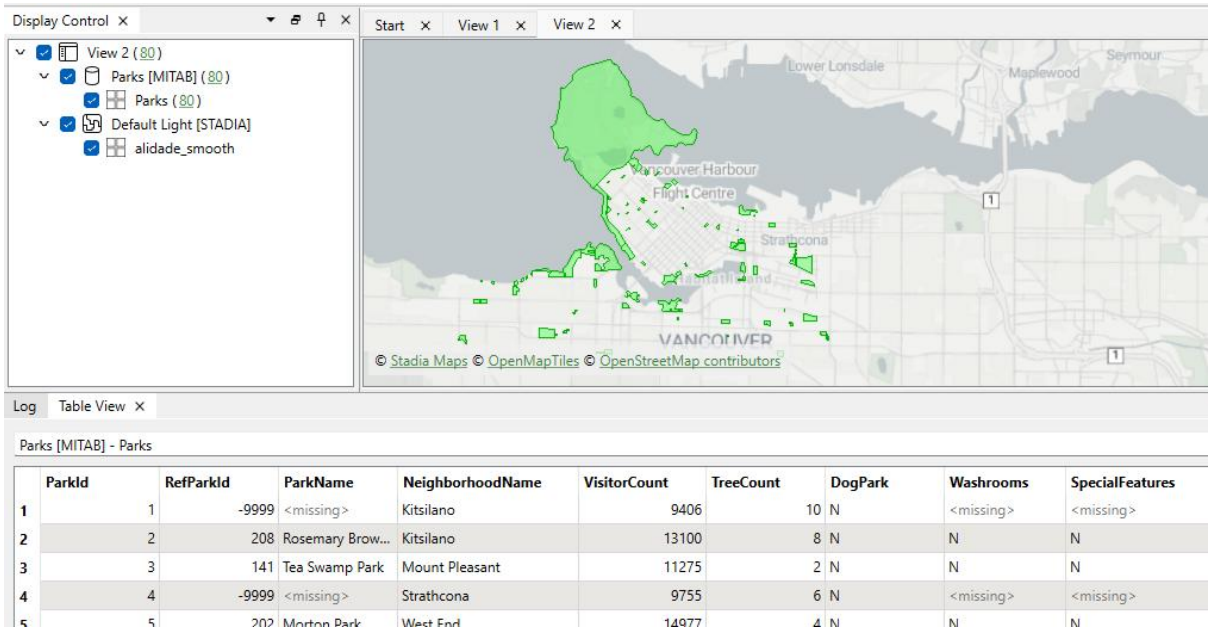
Use the Data Inspector to examine both Firehalls (GML) and Parks (MapInfo TAB).

Reader Format	OGC GML (Geography Markup Language)
Reader Dataset	C:\FMEModularData\Data\Emergency\FireHalls.gml



	gml_parent_id	gml_parent_property	gml_id	HallNumber	Name	Address	PhoneNumber	Engine	Ladder	Quint	Rescue
1	<missing>	featureMember	id83226cd1-7fe...	1	Vancouver Fire Hall No 1	900 Heatley Av	604-665-6001	Y	Y		
2	<missing>	featureMember	id548db1e7-40...	2	Vancouver Fire Hall No 2	199 Main St	604-665-6002	Y		Y	
3	<missing>	featureMember	id12cd50d0-19...	3	Vancouver Fire Hall No 3	2801 Quebec St	604-665-6003	Y	Y		Y
4	<missing>	featureMember	id85442bd0-cc...	4	Vancouver Fire Hall No 4	1475 W 10th Av	604-665-6004			Y	
5	<missing>	featureMember	ida1d84f2f-d4f...	6	Vancouver Fire Hall No 6	1001 Nicola St	604-665-6006	Y		Y	
6	<missing>	featureMember	idcc686e5b-c5a...	7	Vancouver Fire Hall No 7	1090 Haro St	604-665-6007	Y	Y		Y
7	<missing>	featureMember	id8bf85022-9d...	8	Vancouver Fire Hall No 8	895 Hamilton St	604-665-6008	Y			
8	<missing>	featureMember	idf6c8bb01-8d...	12	Vancouver Fire Hall No 12	2460 Balaclava St	604-665-6012	Y		Y	

Reader Format	Precisely MapInfo TAB (MAPINFO)
Reader Dataset	C:\FMEModularData\Data\Parks\Parks.tab

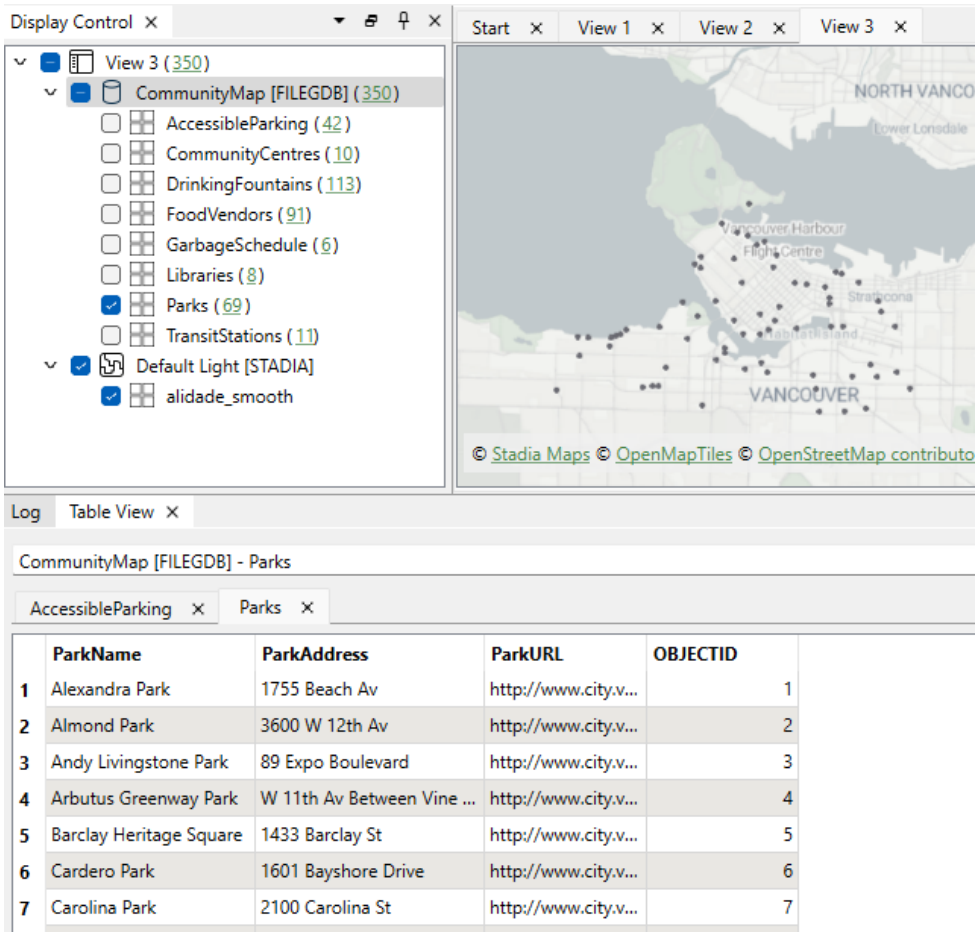


	ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures
1	1	-9999	<missing>	Kitsilano	9406	10	N	<missing>	<missing>
2	2	208	Rosemary Brow...	Kitsilano	13100	8	N	N	N
3	3	141	Tea Swamp Park	Mount Pleasant	11275	2	N	N	N
4	4	-9999	<missing>	Strathcona	9755	6	N	<missing>	<missing>
5	5	202	Morton Park	West End	14977	4	N	N	N

There is an added complication in that the Parks data needs to match an existing schema – and not as it exists within this MapInfo TAB dataset.

Also use the Data Inspector to examine the *Parks* layer (feature type) within the CommunityMap Geodatabase:

Reader Format	Esri Geodatabase (File Geodb Open API)
Reader Dataset	C:\FMEModularData\Data\CommunityMapping\CommunityMap.gdb



	ParkName	ParkAddress	ParkURL	OBJECTID
1	Alexandra Park	1755 Beach Av	http://www.city.v...	1
2	Almond Park	3600 W 12th Av	http://www.city.v...	2
3	Andy Livingstone Park	89 Expo Boulevard	http://www.city.v...	3
4	Arbutus Greenway Park	W 11th Av Between Vine ...	http://www.city.v...	4
5	Barclay Heritage Square	1433 Barclay St	http://www.city.v...	5
6	Cardero Park	1601 Bayshore Drive	http://www.city.v...	6
7	Carolina Park	2100 Carolina St	http://www.city.v...	7

7.4.2 Create Workspace using Dynamic Schema workflow option

Launch the FME Workbench, if it isn't open already. Within the Get Started section choose the ***Generate*** wizard option.

Reader Format	OGC GML (Geography Markup Language)
Reader Dataset	C:\FMEModularData\Data\Emergency\FireHalls.gml
Writer Format	OGC GeoPackage
Writer Dataset	C:\FMEModularData\Output\Training\EmergencyResponsePlanning.gpkg
Workflow Options	Dynamic Schema

The dynamic option is chosen so that we can use a schema other than that of the dataset being translated...

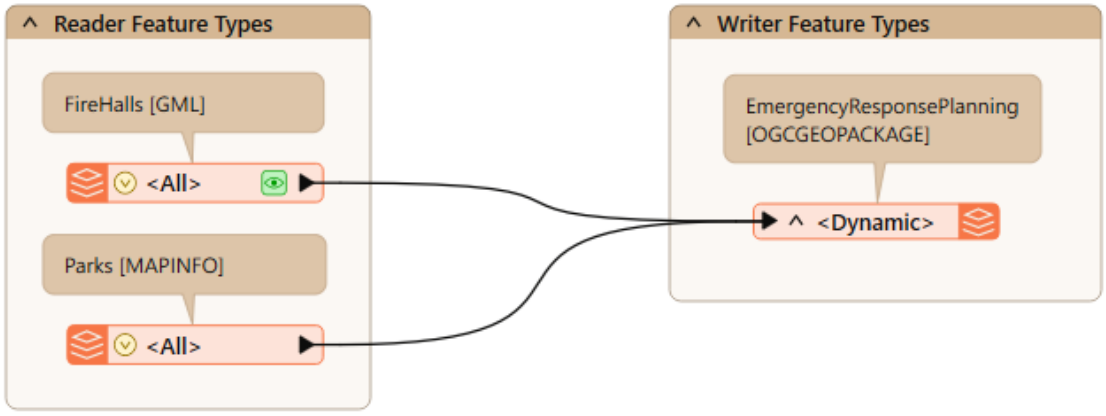
7.4.3 Add another Reader

Now let's add a Reader for the new parks data, using the following details:

Reader Format	MapInfo TAB (MAPINFO)
Reader Dataset	C:\FMEModularData\Data\Parks\Parks.tab

Workflow Options	Single Merged Feature Type

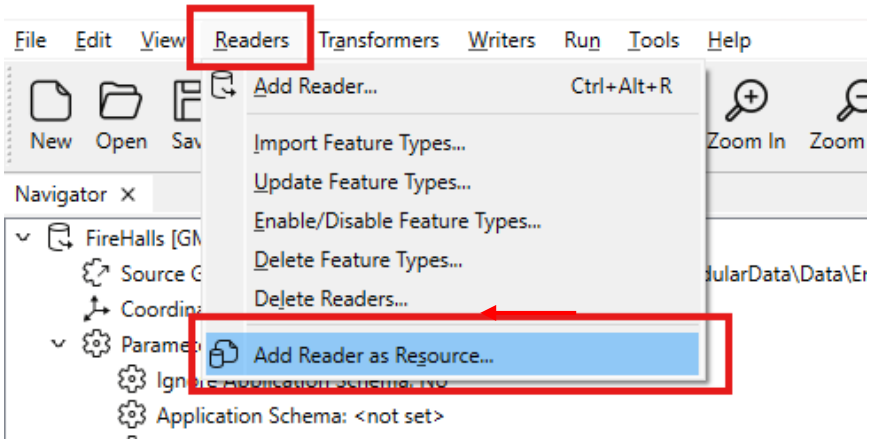
Connect the new reader feature type up to the existing writer feature type, and add annotation for the reader feature types for easier recognition:



7.4.4 Add a Resource Reader

Now we need to reference the required schema to apply to the Parks (MapInfo) dataset.

From the menu bar use *Readers > Add Reader as Resource*



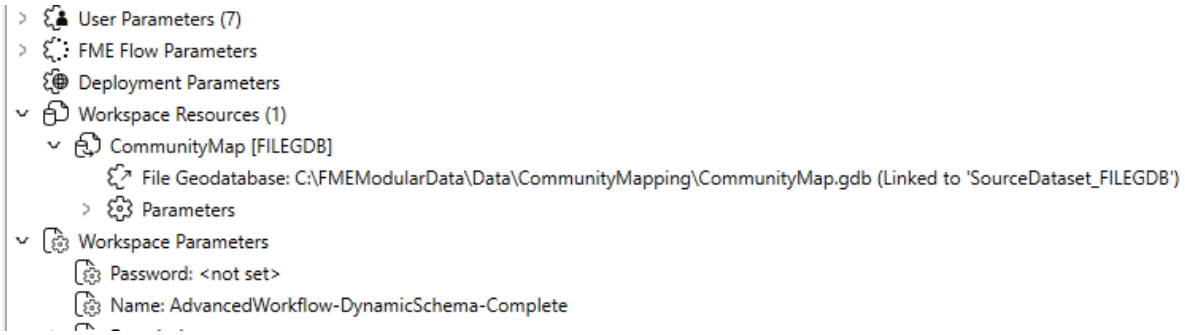
and, when prompted, enter the following details:

Reader Format	Esri Geodatabase (File Geodb Open API)
Reader Dataset	C:\FMEModularData\Data\CommunityMapping\CommunityMap.gdb

FME Lizard

If, in this dialog, you see the parameter for "Individual Feature Types/Single Merged Feature Types" then you chose "Add Reader" not "Add Reader as Resource". Click Cancel and pick the correct menu entry this time!

Click OK, and the reader is added as a *Workspace Resource*:



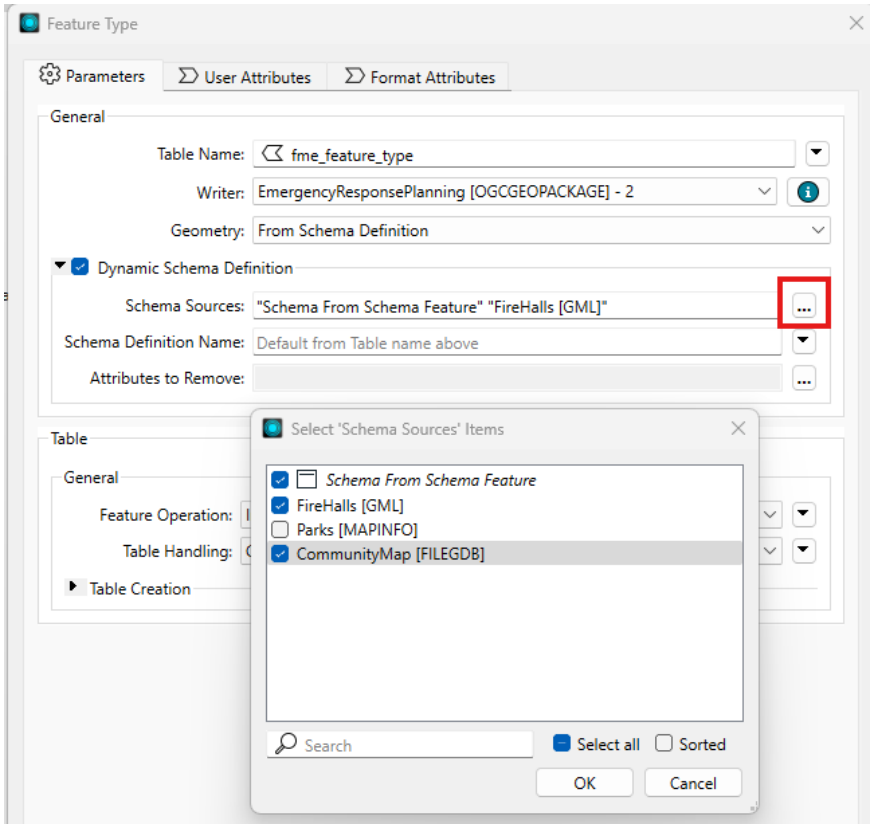
7.4.5 Adjust Dynamic Parameters

Now we need to make sure that the resource schema is being used for Parks.FireHalls will use the incoming FireHalls data schema.

Inspect the properties for the writer feature type. Click the *Schema Sources browse button*.

Put a *checkmark* against both *CommunityMap* [FILEGDB] and *FireHalls [GML]*

Ensure *Parks* [MAPINFO] is *NOT* checked:



Inspect the other dynamic parameters. For some formats it might be necessary to set Geometry types/definition. Since we are writing both points and polygons to a Geopackage, we'll need to do exactly that!

For this scenario change the Geometry parameter to: *First Feature Defines Geometry Type*

But notice there is also the geometry option *From Schema Definition*. However, that wouldn't work for Parks, as the parks features in the MapInfo dataset are polygons, but the park features in the Geodatabase (schema source) are points.

Accept the parameter changes to close the dialog.

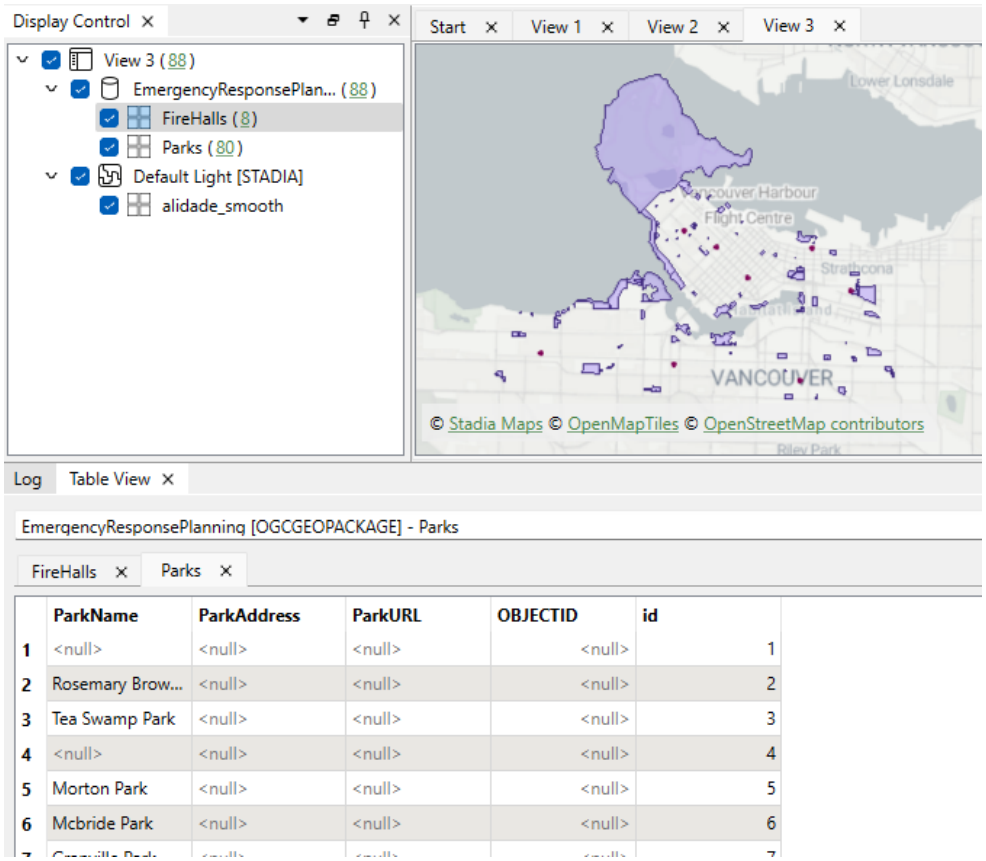
7.4.6 Save and Run the Workspace

Save the workspace and then Re-Run it (ensuring the entire workspace is re-run).

7.4.7 Examine the Output

Inspect the output EmergencyResponsePlanning.gpkg. There should be two layers: FireHalls and Parks.

The Parks dataset should have the schema from the CommunityMapping Geodatabase (not the MapInfo TAB Parks dataset), including attributes for ParkName, ParkAddress, and ParkURL (even if there is no data to fill some fields yet):

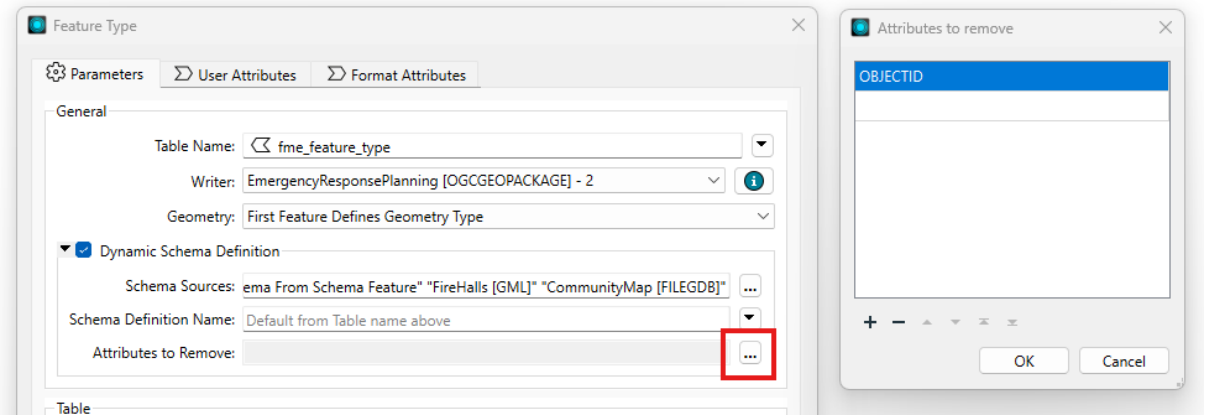


	ParkName	ParkAddress	ParkURL	OBJECTID	id
1	<null>	<null>	<null>	<null>	1
2	Rosemary Brow...	<null>	<null>	<null>	2
3	Tea Swamp Park	<null>	<null>	<null>	3
4	<null>	<null>	<null>	<null>	4
5	Morton Park	<null>	<null>	<null>	5
6	Mcbride Park	<null>	<null>	<null>	6

Notice that it also has OBJECTID, which came from the Geodatabase and we don't really need.

7.4.8 Delete Attribute

Revisit the writer feature type parameters. Click the *ellipse button* for *Attributes to Remove* and type in *OBJECTID* into the first row of the dialog that opens. *You won't be able to select it from the drop-down list because it comes from a resource reader, not a real reader:*

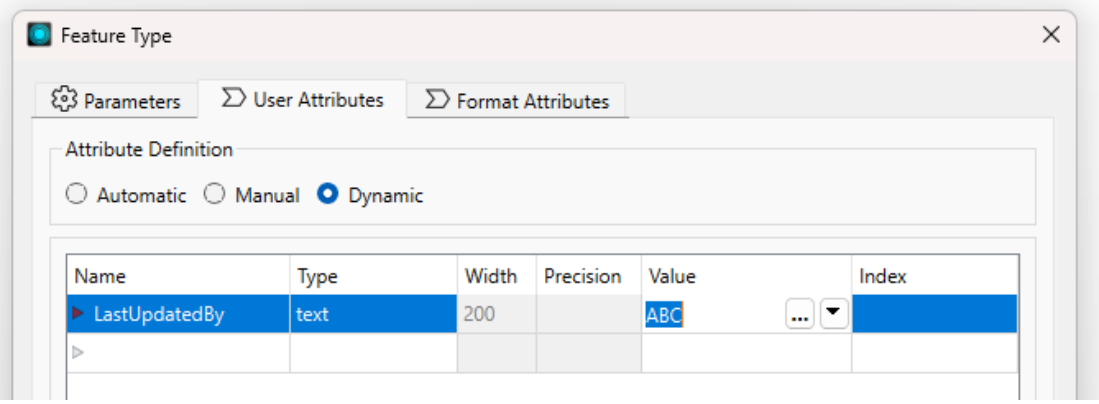


Click OK to close this dialog, but don't close the feature type dialog yet....

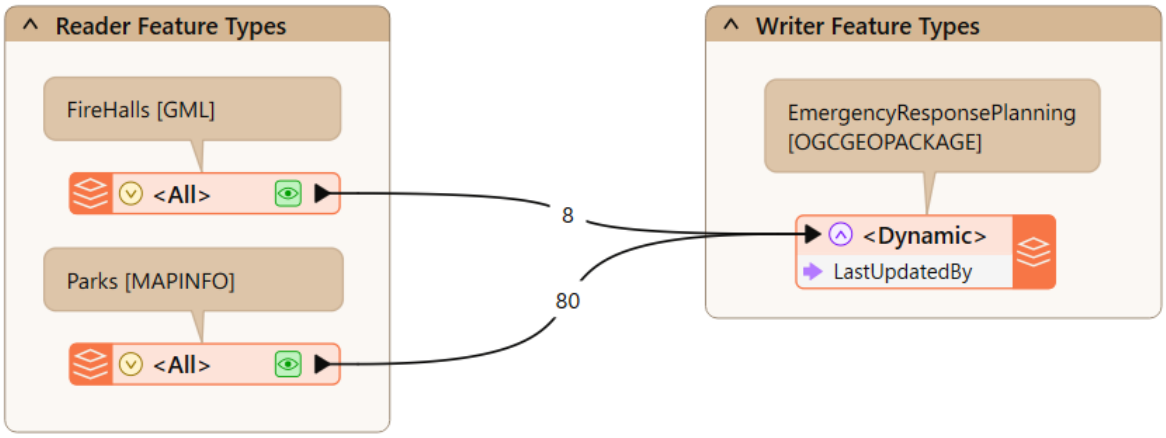
7.4.9 Add Attribute

A last-minute request is to add an attribute – *LastUpdatedBy* – to all tables in the output.

So, click on the User Attributes tab and add this new attribute. Make it a 30-character string.
We could create a Text User Parameter to enable the end-user to enter their initials at the time of running the workspace, to populate the *LastUpdatedBy* attribute. But for now, we'll just enter a static value. Enter your initials in the Value cell:



As you can see, there is no need to change the Attribute Definition mode - it should stay as Dynamic.



7.4.10 Resave, Rerun and Examine the Output

Resave the workspace and then Re-Run it (ensuring the entire workspace is re-run).

Inspect the output.
Notice that *OBJECTID* will not appear as an attribute. *LastUpdatedBy* does appear - on both Parks and Firehalls – and will be populated with the initials you entered as a static value.

Congratulations

By Completing this exercise you have learned how to:

- Create a Dynamic workspace with multiple readers
- Add a Resource Reader to make use of schema in another dataset
- Manage Schema Source of writer feature types
- Add & remove hard-coded attributes in a dynamic workspace

7.5 Dynamic Workspace Table-Based Schema

Demonstrates	Use the Schema (from Table) reader to read a schema from a spreadsheet
Overall Goal	Generate a new Community Mapping dataset using a table-based schema
Data	Firehalls (GML) Parks (MapInfo TAB) Zones (MapInfo TAB) *Schema Table = CommunityMapSchema (Microsoft Excel)*
Start Workspace	C:\FMEModularData\Workspaces\7.05-AdvancedWorkflow-TableBasedSchema-Begin.fmw
End Workspace	C:\FMEModularData\Workspaces\Complete\7.05-AdvancedWorkflow-TableBasedSchema-Complete.fmw

In a previous exercise, you created a new emergency response planning dataset using a dynamic schema. At the time only two tables were defined, but now another one is required, and the service area wants you to update the workspace.

Rather than having to make changes each time they add more datasets, you believe that you can simply create an Excel spreadsheet containing the schema definition. That way the service area team can edit it themselves and do the same for all future updates.

7.5.1 Inspect Spreadsheet

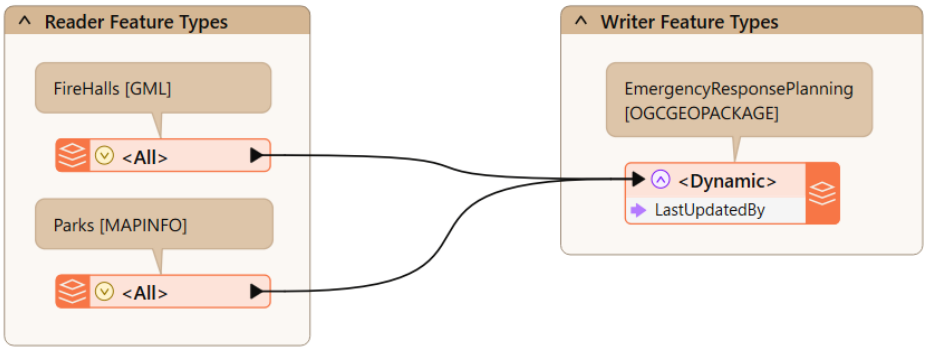
Open and examine the spreadsheet below in the Data Inspector:
C:\FMEModularData\Resources\CommunityMapSchema.xlsx

CommunityMapSchema [XLSXR] - Sheet1

	Layer	FieldName	DataType	Geometry	AttrOrder
1	FireHalls	HallNumber	fme_char(30)	fme_point	1
2	FireHalls	Name	fme_char(30)	<missing>	2
3	FireHalls	Address	fme_char(30)	<missing>	3
4	FireHalls	PhoneNumber	fme_char(30)	<missing>	4
5	FireHalls	LastUpdatedBy	fme_char(30)	<missing>	5
6	Parks	ParkName	fme_char(30)	fme_polygon	1
7	Parks	ParkAddress	fme_char(50)	<missing>	2
8	Parks	ParkURL	fme_char(100)	<missing>	3
9	Parks	LastUpdatedBy	fme_char(30)	<missing>	4
10	Zones	ZoneName	fme_char(30)	fme_polygon	1
11	Zones	ZoneCategory	fme_char(30)	<missing>	2
12	Zones	LastUpdatedBy	fme_char(30)	<missing>	3

7.5.2 Launch FME Workbench and Open Workspace

Launch the FME Workbench, if it isn't open already. Then open the workspace C:\FMEModularData\Workspaces\7.05-AdvancedWorkflow-TableBasedSchema-Begin.fmw



7.5.3 Delete CommunityMap Resource Reader

Because we are using a spreadsheet to define our output schemas, the CommunityMap Resource Reader is no longer needed. Locate it in the Navigator window, right-click on it, and choose *Delete*.

When prompted, click *Yes* to confirm that all references relating to this dataset will also be removed.

7.5.4 Add Excel File as Reader Resource

Now, from the toolbar, select *Readers* > ***Add Reader as Resource***. In the dialog that opens choose:

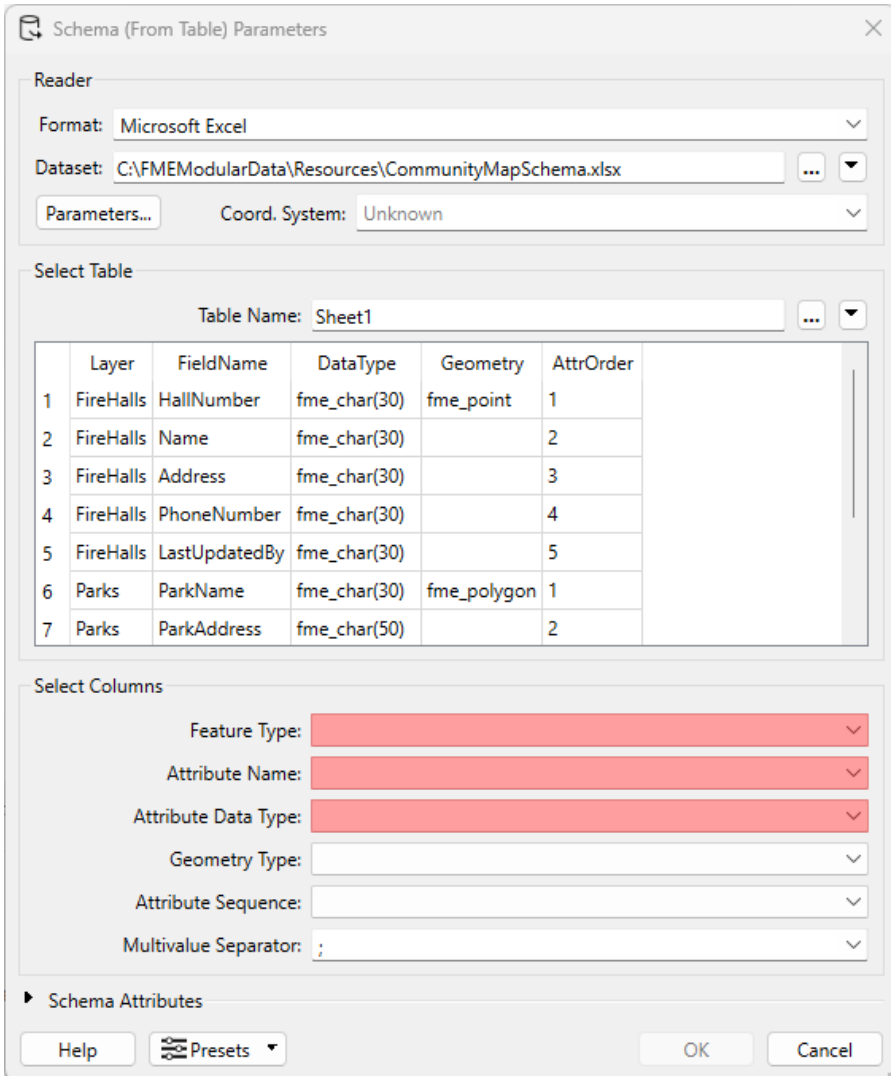
Reader Format	Schema (From Table)
Reader Dataset	C:\FMEModularData\Resources\CommunityMapSchema.xlsx

* NB: Be sure to use the *Schema (From Table)* format and not the Excel format! *
Click the *Parameters*... button (OK will be greyed-out until you do). This dialog is where we can define how the table maps to the required schema.

Set the Reader parameters as follows:

Format	Microsoft Excel
Dataset	C:\FMEModularData\Resources\CommunityMapSchema.xlsx
Table Name	Sheet1

The first row should get used as the field names. If this is not the case, then click the parameters button above and set the values correctly:



Next, select the appropriate fields to match to the required parameters:

Feature Type	Layer
Attribute Name	FieldName
Attribute Data Type	DataType
Geometry Type	Geometry
Attribute Sequence	AttrOrder

Click OK to close the dialog and OK again to add the resource reader.

FME Lizard says ….

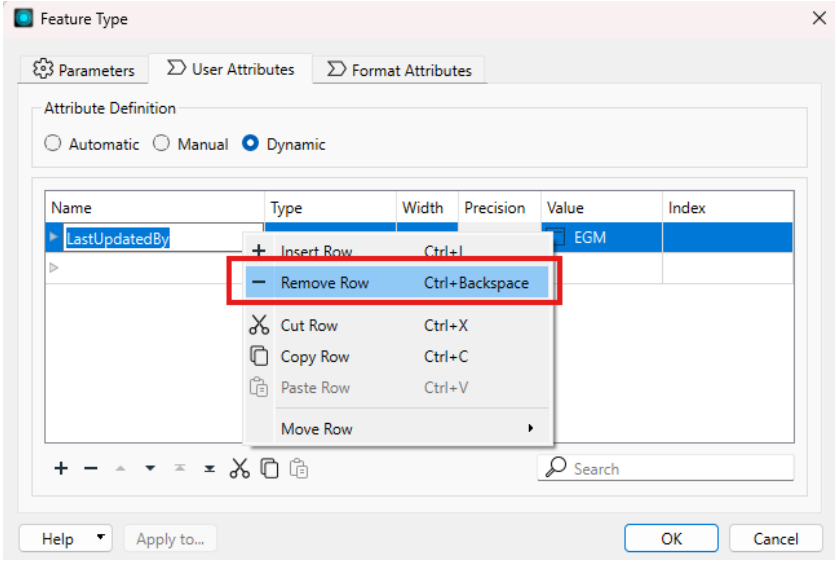
The fields in such a lookup table can be given any name you desire. In this example, they have been given similar names as the parameter to which they are being applied, but only to make the relationship as obvious as possible. Some of them (Geometry Type and Attribute Sequence) are even optional and don't have to be part of the

schema definition if not needed.

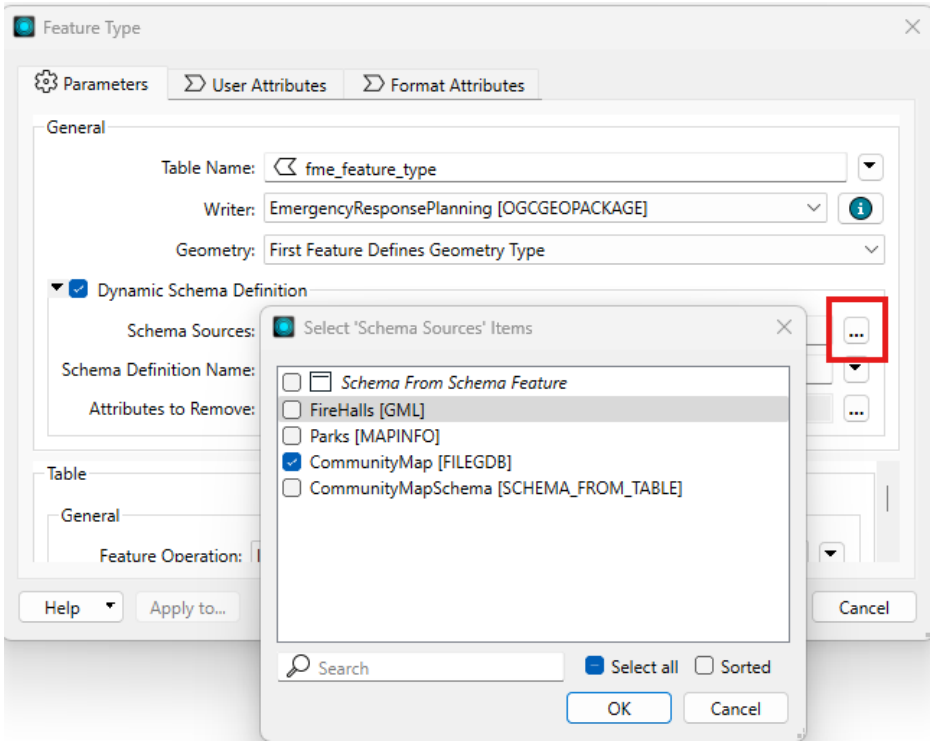
7.5.5 Set Dynamic Parameters

Now inspect the parameters for the writer feature type.

Under the *User Attributes* tab remove the *LastUpdatedBy* attribute, as we've added this to the spreadsheet definition for each type and no longer need it in here.



In the *Parameters* tab click the *Schema Sources* edit *button*.
Uncheck FireHalls and *check CommunityMapSchema [SCHEMA_FROM_TABLE]:*



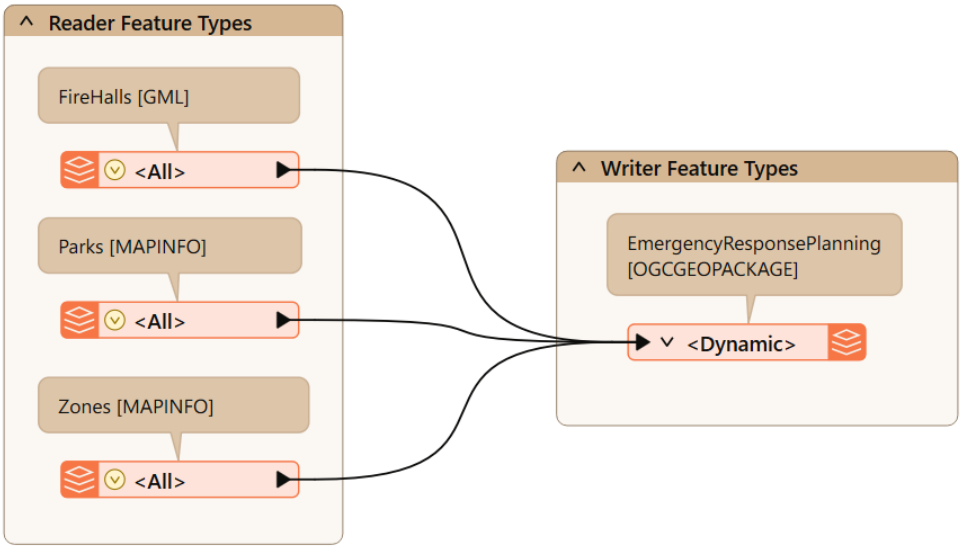
Accept the changes to these parameters.

7.5.6 Add a Reader

Add another Reader:

Reader Format	MapInfo TAB (MAPINFO)
Reader Dataset	C:\FMEModularData\Data\Zoning\Zones.tab
Workflow Options	Single Merged Feature Type

Once added, connect its reader feature type to the dynamic writer feature type.



7.5.7 Save and Run Workspace

Save the workspace and then run it.

Inspect the output. Notice that all three feature types have been written and that their attribute schema matches what was defined in the Excel spreadsheet.

EmergencyResponsePlanning [OGCGEOPACKAGE] - FireHalls

	HallNumber	Name	Address	PhoneNumber	LastUpdatedBy	id
1	1	Vancouver Fire Hall No 1	900 Heatley Av	604-665-6001	<null>	1
2	2	Vancouver Fire Hall No 2	199 Main St	604-665-6002	<null>	2
3	3	Vancouver Fire Hall No 3	2801 Quebec St	604-665-6003	<null>	3

EmergencyResponsePlanning [OGCGEOPACKAGE] - Parks

	ParkName	ParkAddress	ParkURL	LastUpdatedBy	id
1	<null>	<null>	<null>	<null>	1
2	Rosemary Brow...	<null>	<null>	<null>	2
3	Tea Swamp Park	<null>	<null>	<null>	3

EmergencyResponsePlanning [OGCGEOPACKAGE] - Zones

	ZoneName	ZoneCategory	LastUpdatedBy	id
1	M-2	Industrial	<null>	1
2	M-1	Industrial	<null>	2
3	M-2	Industrial	<null>	3

Congratulations

By Completing this exercise, you have learned how to:

- Use the Schema (from Table) reader to read a schema from a spreadsheet

7.6 Batch Processing with the Workspace Runner

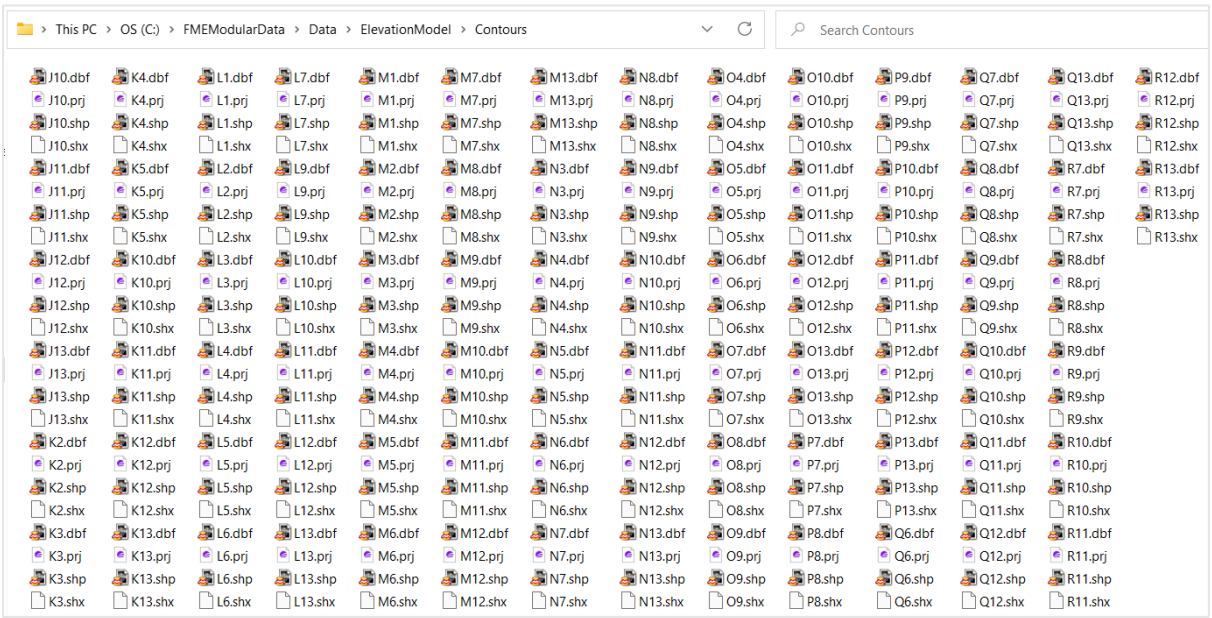
Demonstrates	Using a Directory and File Pathname reader to generate source dataset paths (one feature per dataset path) Using a WorkspaceRunner transformer to perform batch processing
Overall Goal	Use a WorkspaceRunner transformer in conjunction with the Directory and File Pathname reader to perform batch processing of numerous tiles of contour data
Data	Contours (ESRI Shapefiles)
Start Workspace	C:\FMEModularData\Workspaces\7.06-AdvancedWorkflow-WorkspaceRunnerChild-Begin.fmw
End Workspace	C:\FMEModularData\Workspaces\Complete\7.06-AdvancedWorkflow-WorkspaceRunnerParent-Complete.fmw

The WorkspaceRunner transformer runs a different workspace from within another workspace. In conjunction with the Directory and File Pathname reader, it can be used as a device for batch processing.

In this example, a colleague has a workspace that processing contour data; creating a new attribute and applying styling specific for the use in MapInfo Pro before writing back out to tiles. However, they have encountered issues when trying to process the entire contour data holding at once. You need to employ batch processing with a WorkspaceRunner to process the numerous tiles of contour data.

7.6.1 Examine the Input Contour Tile data

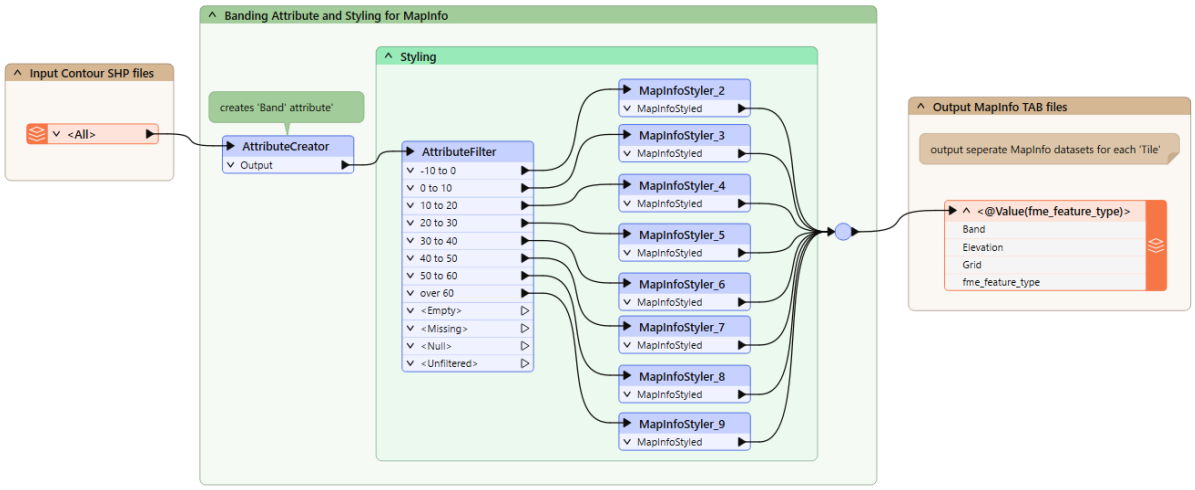
Open and examine one or two tiles of the contour data within the Data Inspector from C:\FMEModularData\Data\ElevationModel\Contours:



Reader Format	Esri Shapefile
Reader Dataset	C:\FMEModularData\Data\ElevationModel\Contours

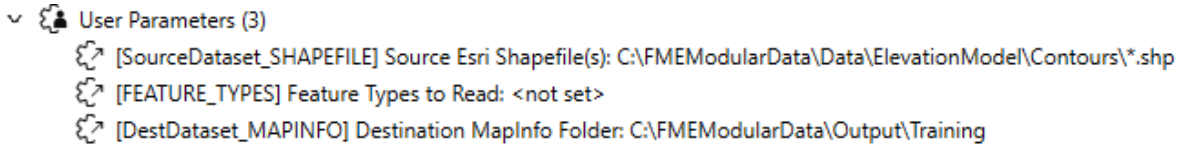
7.6.2 Launch FME Workbench and Open Workspace

Launch the FME Workbench, if it isn't open already. Then open the workspace C:\FMEModularData\Workspaces\7.06-AdvancedWorkflow-WorkspaceRunnerChild-Begin.fmw



This workspace includes an ESRI Shapefile reader, with a Single Merged Feature Type. Along with a MapInfo TAB writer using Fanout on feature type. The workflow is already taking care of the attribute creation and styling ready for MapInfo Pro users.

Check the existing User Parameters. Importantly the ESRI Shapefile reader Source parameter is a User parameter – this will be needed later!:



You can test run this workspace to ensure it works if you like. We will be using another workspace to run this workspace. Close the workspace.

7.6.3 Create a New Workspace

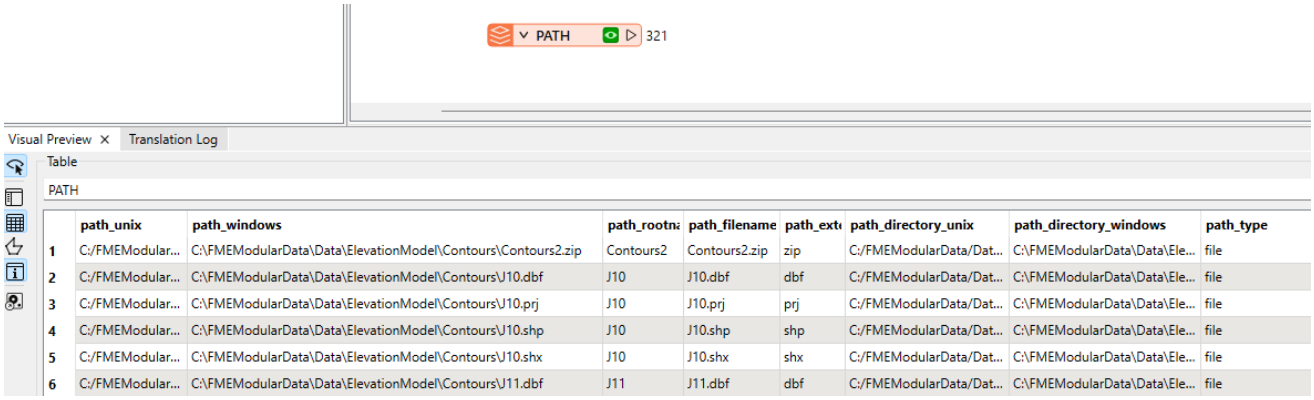
In a blank workspace, add a *Directory and File Pathnames* reader (as you would add a normal reader) and set as follows:

Reader Format	Directory and File Pathnames
Reader Dataset	C:\FMEModularData\Data\ElevationModel\Contours
Workflow Options	Single Merged Feature Type

By specifying the folder containing the contour tiles we wish to process, the Directory and File Pathnames reader will generate the source dataset path for each input file (tile).

7.6.4 Examine the Results of the Directory and File Pathnames reader

Use Run Just This on the feature type, then examine the feature cache in a Table view within Visual Preview:



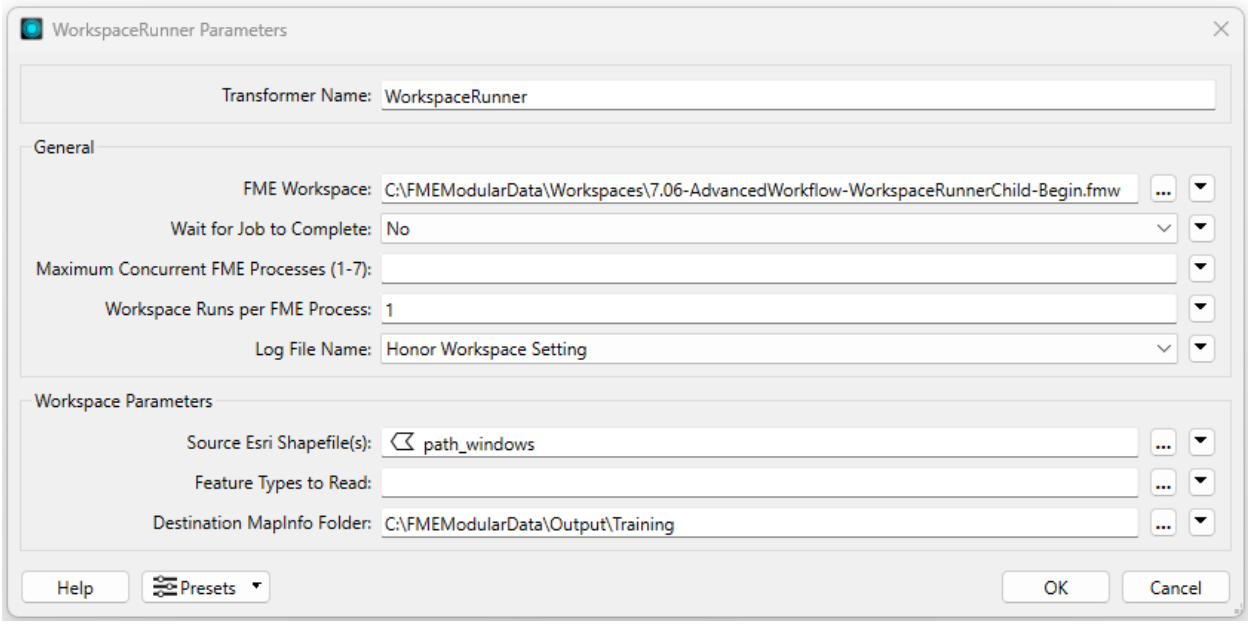
	path_unix	path_windows	path_rootna	path_filename	path_exte	path_directory_unix	path_directory_windows	path_type
1	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\Contours2.zip	Contours2	Contours2.zip	zip	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file
2	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\J10.dbf	J10	J10.dbf	dbf	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file
3	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\J10.prj	J10	J10.prj	prj	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file
4	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\J10.shp	J10	J10.shp	shp	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file
5	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\J10.shx	J10	J10.shx	shx	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file
6	C:/FMEModular...	C:\FMEModularData\Data\ElevationModel\Contours\J11.dbf	J11	J11.dbf	dbf	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file

7.6.5 Add a WorkspaceRunner transformer

Add a WorkspaceRunner transformer to the canvas and connect it to the Directory and File Pathnames reader's feature type. The WorkspaceRunner will be used to initiate our original workspace (C:\FMEModularData\Workspaces\7.06-AdvancedWorkflow-WorkspaceRunnerChild-Begin.fmw).

Set the parameters of the WorkspaceRunner as follows:

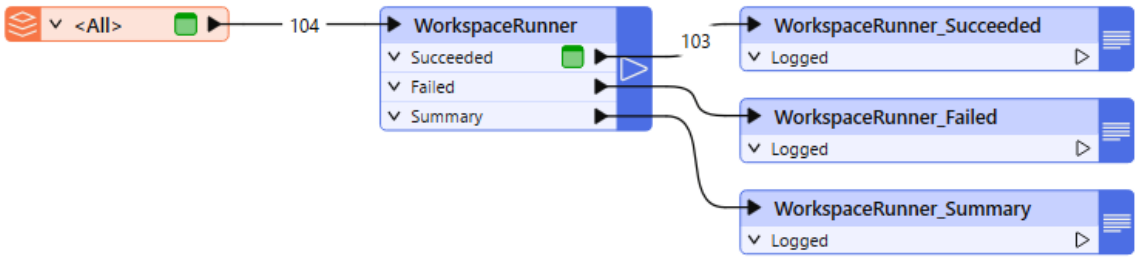
FME Workspace	C:\FMEModularData\Workspaces\7.06-AdvancedWorkflow-WorkspaceRunnerChild-Begin.fmw
Wait for Job to Complete	No
Source ESRI Shapefile(s)	This is the important User parameter we looked at earlier. Change this to use the attribute *path_windows*



As the WorkspaceRunner does its stuff, a number of FME processes are started - one for each file. The WorkspaceRunner can be set to wait for each process to complete before starting the next, in which case there would only be one new process at a time. There is a parameter in the WorkspaceRunner for setting the Maximum Number of Concurrent Processes. This limits the number of concurrently executing workspaces to the value specified. If specified, this parameter value must be an integer in the range 1-7. The specified value includes the process executing WorkspaceRunner. For example, if the value is set to 7 then WorkspaceRunner can only start 6 additional FME processes concurrently.

7.6.6 Connect Loggers

To log any errors that might arise with batch processing, right click on the WorkspaceRunner and click *Connect Loggers*.

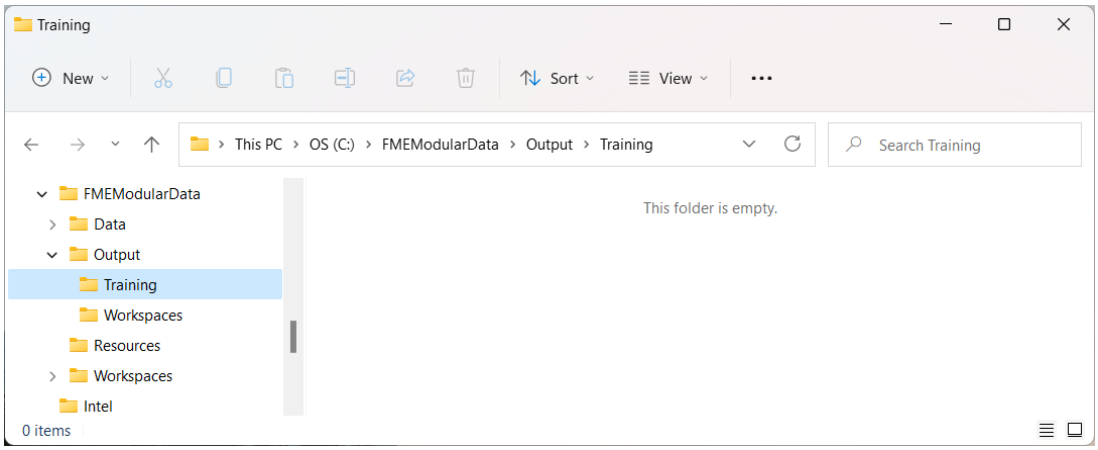


7.6.7 Save the Workspace

Save the workspace into C:\FMEModularData\Output\Workspaces giving it a meaningful name.

7.6.8 Get Ready

Before we initiate the processing, we will get a few views set up, so that we can observe the batch processing in action:
Open a File Explorer and navigate to the output destination folder:
C:\FMEModularData\Output\Training



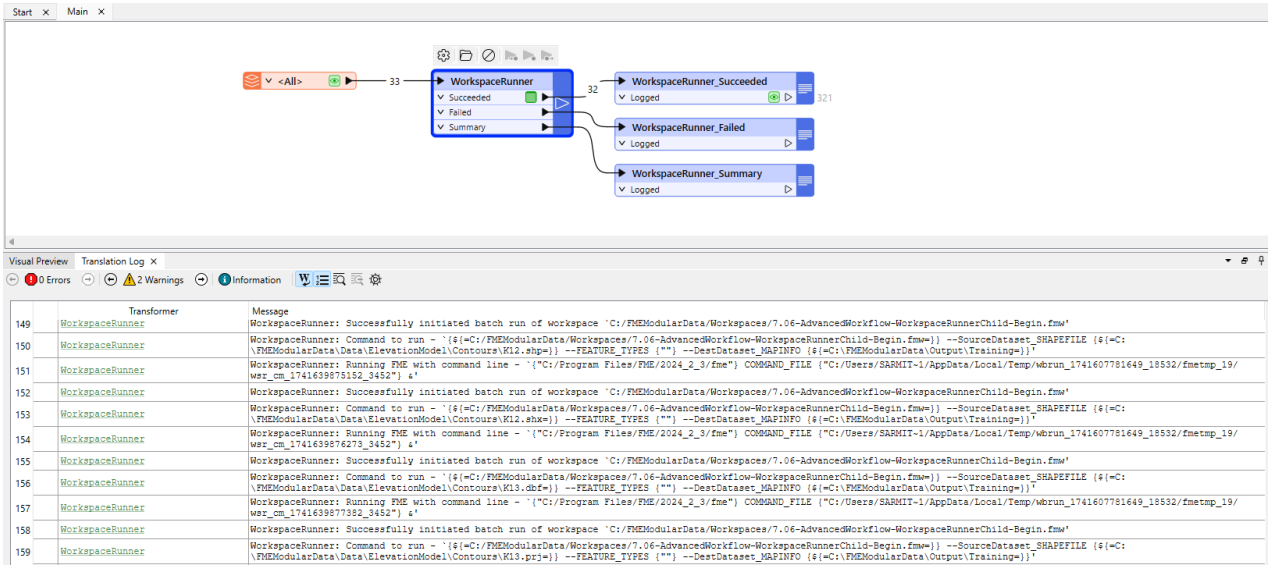
We'll be able to watch the output files being written when the batch processing commences.

Also, open Task Manager from the Windows Start. The Task Manager will show the number of executing fme.exe processes running.

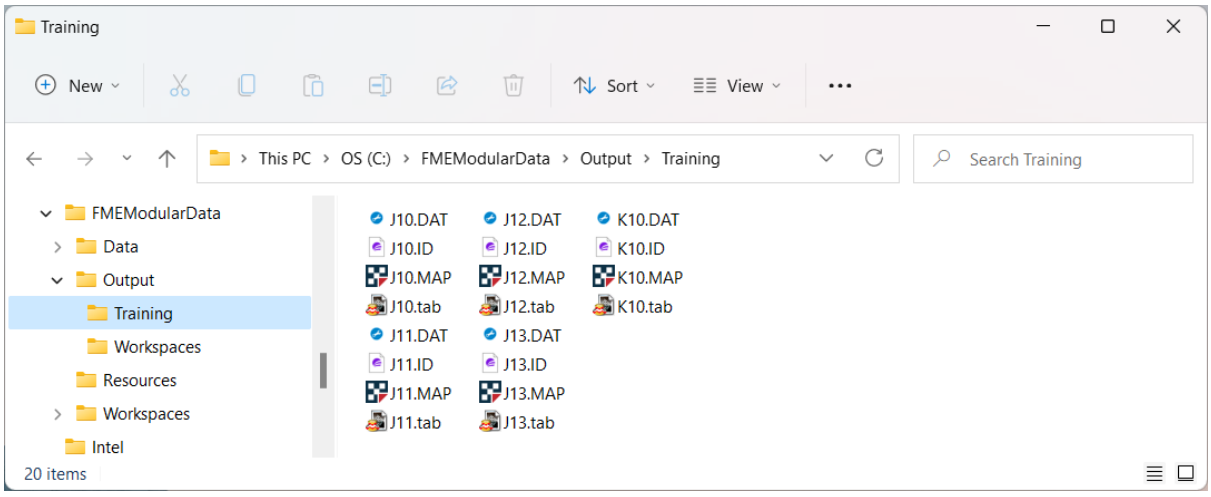
7.6.9 Run the Workspace

Run the workspace.

As each feature pass from the Directory and File Pathnames reader feature type into the WorkspaceRunner transformer, the WorkspaceRunner triggers the Child workspace to run. Watch the feature counts pass from the WorkspaceRunner to the Logger as each process is initiated.

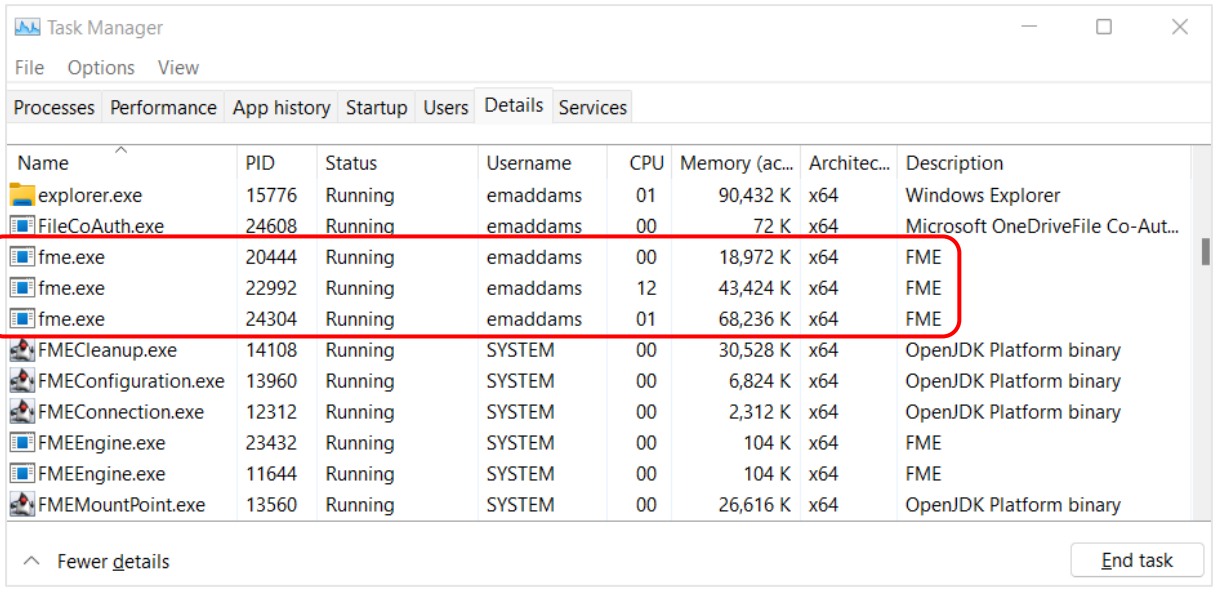


Within File Explorer you'll see a new MapInfo TAB dataset for each tile/process:



7.6.10 View the Task Manager

Within the Task Manager click on the *Details* tab. Here it will show the number of executing *fme.exe* processes (workspaces) running:



7.6.11 Examine Results in Data Inspector

Once all the contour tiles have been processed, view the written MapInfo TAB datasets using the Data Inspector. They will include the new attribute of 'Band' and styling ready for MapInfo Pro:

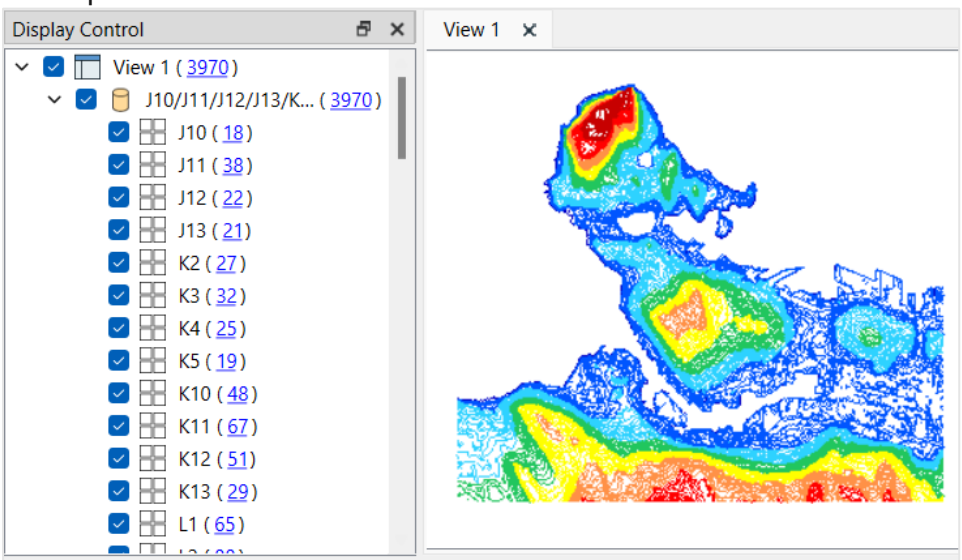


Table View

'L12/L13/M1/M2/M3/M4/M5/M6/M7/M8/M9/M10/M11/M12/M13/N3/N4/N5/N6/N7/N8/N9/N10/N11/N12

	Band	Elevation	Grid
1	0 to 10	1	J10
2	0 to 10	2	J10
3	-10 to 0	0	J10
4	0 to 10	3	J10
5	10 to 20	17	J10

Usage Notes

FME Desktop and FME Server

Publishing to FME Server: Publishing a workspace that includes the WorkspaceRunner transformer is not recommended. The transformer will try to start an FME outside of FME Server to run the workspace. When using FME Server for workspace chaining use FMEServerJobSubmitter transformer or FME Server Automations.

Only use the WorkspaceRunner transformer when running workspaces with FME Desktop.

Congratulations

By Completing this exercise, you have learned how to:

- Use the Directory and File Pathnames reader to generate source dataset paths
- Use batch processing with the WorkspaceRunner transformer to run a workspace on multiple input data sources simultaneously

Got any questions?

We’re happy to help!

0121 232 8000

info@misoportal.com