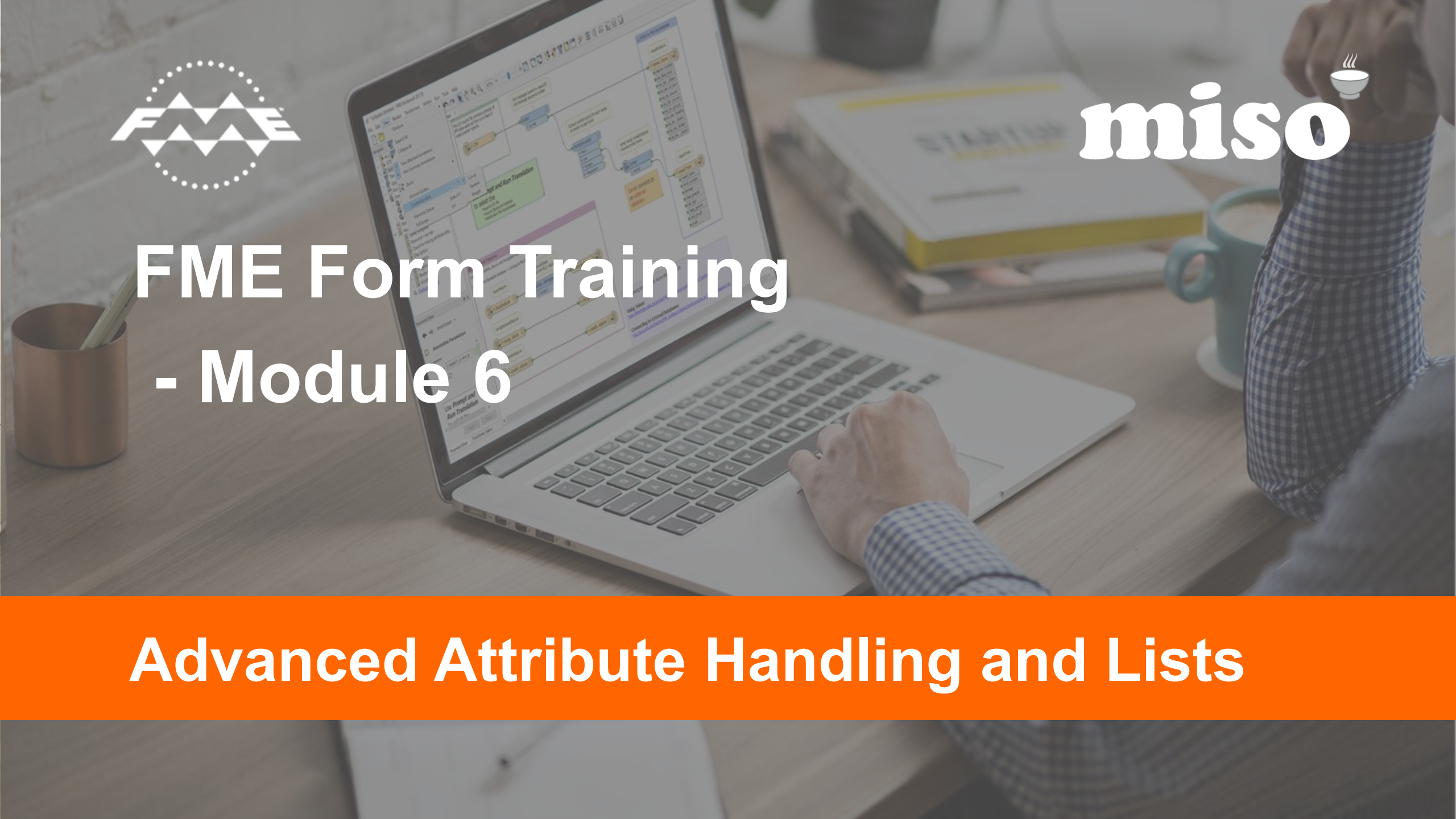
FME Form Training

Slide 2 = Module 6 - Advanced Attribute Handling and Lists

Slide 45 = Module 7 - Advanced Workflow Design



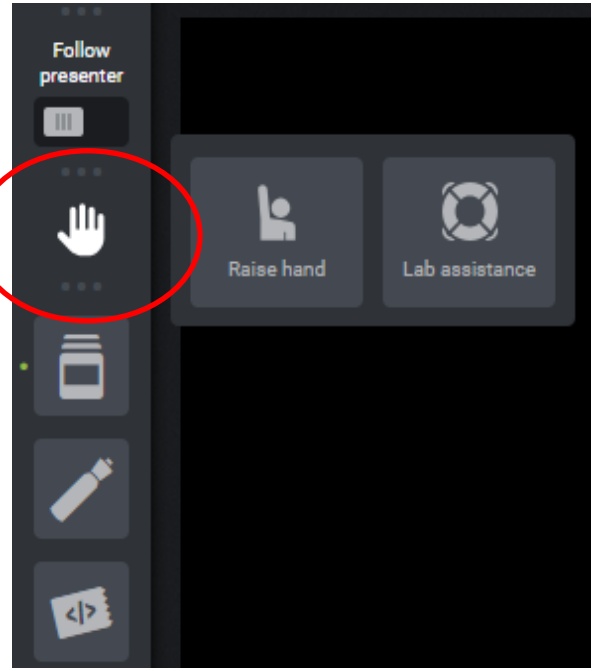
FME Form Training - Module 6

Advanced Attribute Handling and Lists

Training Environment

< keep '**Follow presenter**' on

< **Lab**



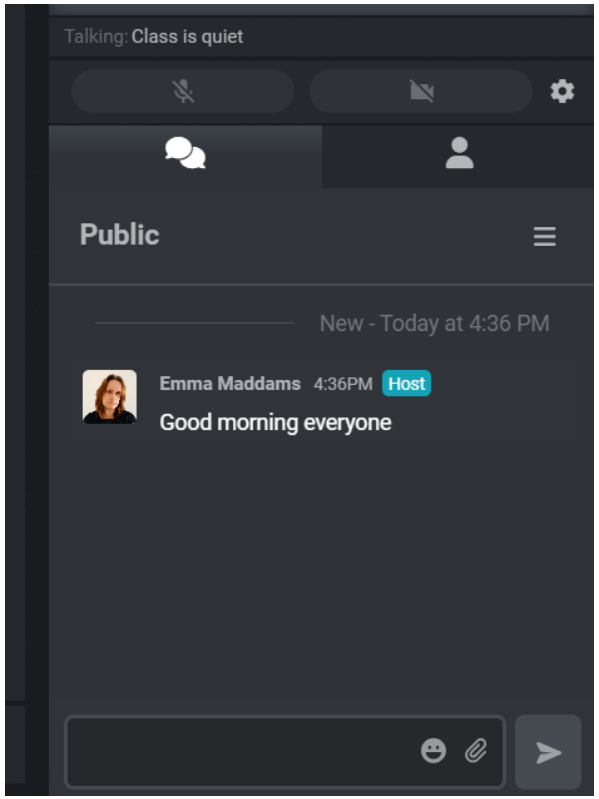
need help when using your Lab

Use either:

- 'Raise hand' or
- 'Lab Assistance'

Chat

- to everyone
- to trainer



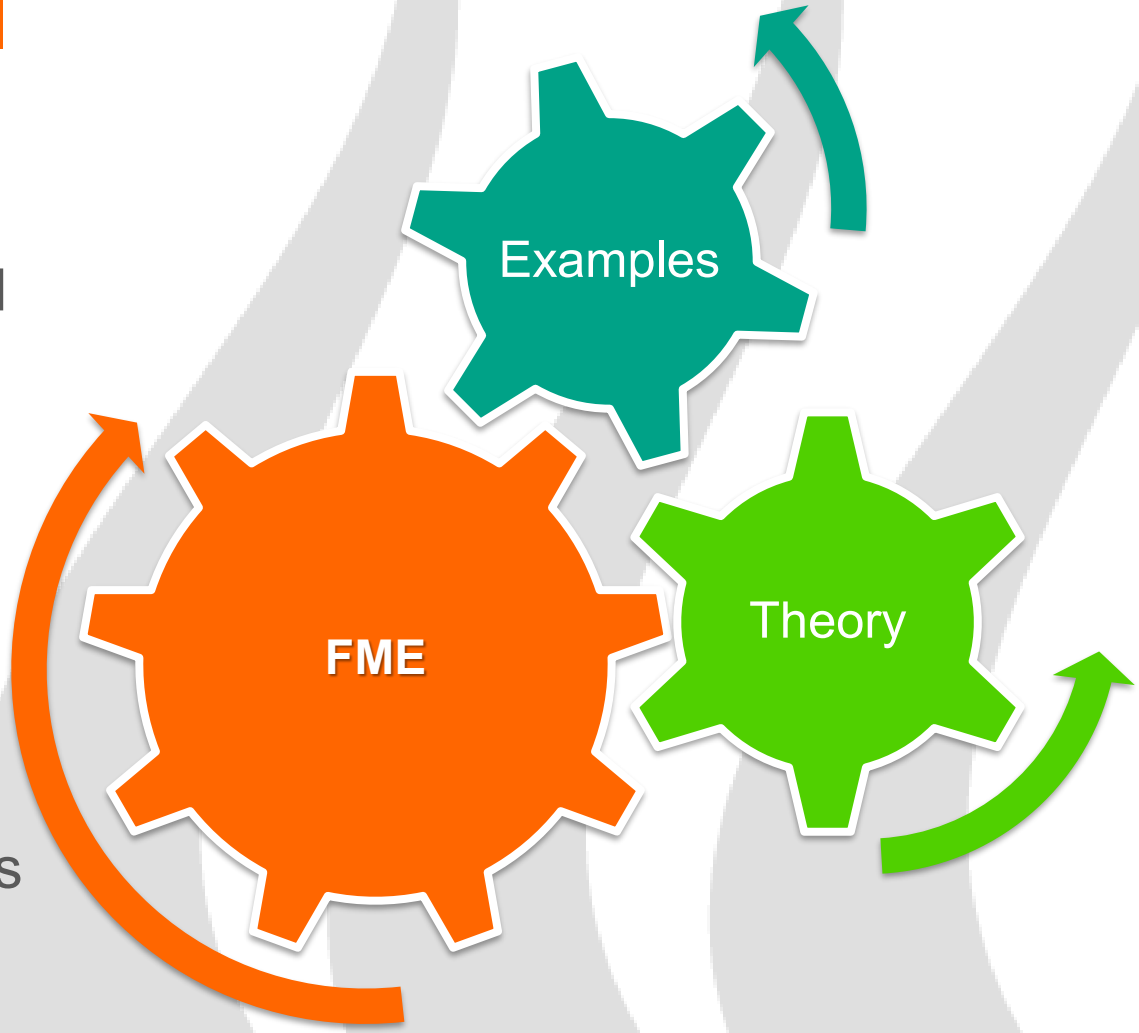
Training Environment

- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

What we'll be covering today…

Agenda

- Advanced Attribute Handling
  - Constructing Values using Text and Arithmetic Editors
  - Date/Time formatting
  - Functions
  - Null and Missing Values
  - Adjacent Feature Attributes
- Working with List Attributes
  - Generating and using List Attributes
  - Transformers for working with Lists



A little bit of background…

miso

- Based in Birmingham
- Platinum partner
- Spatial data experts
- Provide training and consultancy in FME

Safe Software

- Based in Canada
- Create FME product suite
- Continuously develop products

Advanced Attributes

Connect. Transform. **Automate**

Advanced Attributes - techniques are we going to cover

- **Construct** and **format** new attribute values using string manipulation, arithmetic calculations and date/time stamps.
- Efficiently set values based on conditional tests using **Conditional Values**
- Improve data quality by handling **Null and Missing Values** - reduce the likelihood of mistakes and errors in your data by identifying and addressing missing, empty, and null values.
- Extract insights from ordered data by creating attributes with values derived from **adjacent features**
- Manipulate **Lists** and extract information from them using list transformers

Constructing Attributes

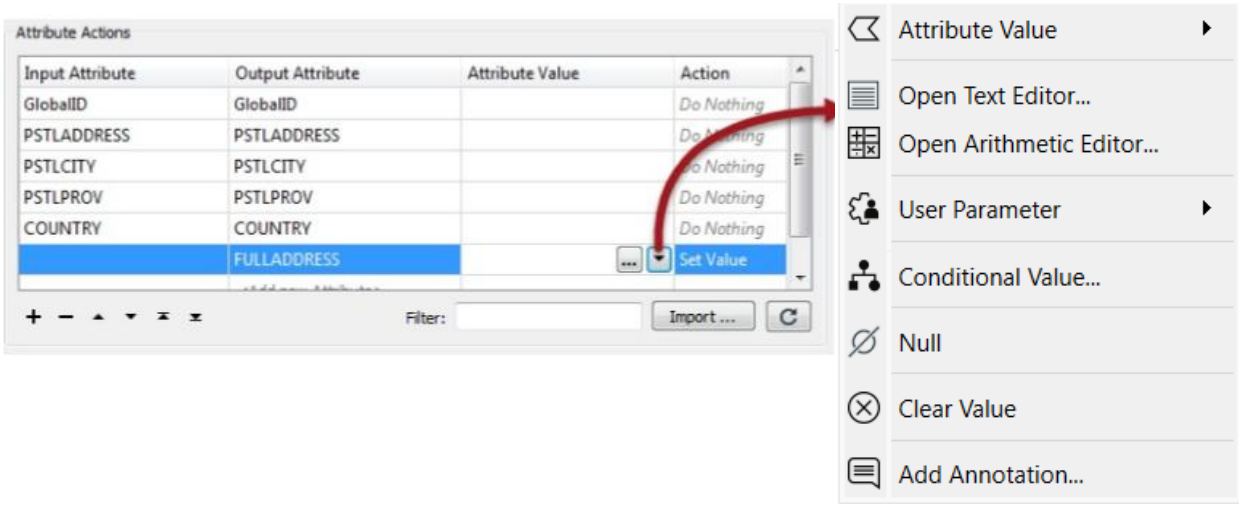
Connect. Transform. **Automate**

Constructing Values

Besides constant attribute values, FME allows you to construct new values using **string manipulation** and **arithmetic calculations**

The attribute or the parameter now no longer is a fixed value: it can be constructed from a mix of existing attributes and parameters.

The two main methods are the **Text Editor** and **Arithmetic Editor**.



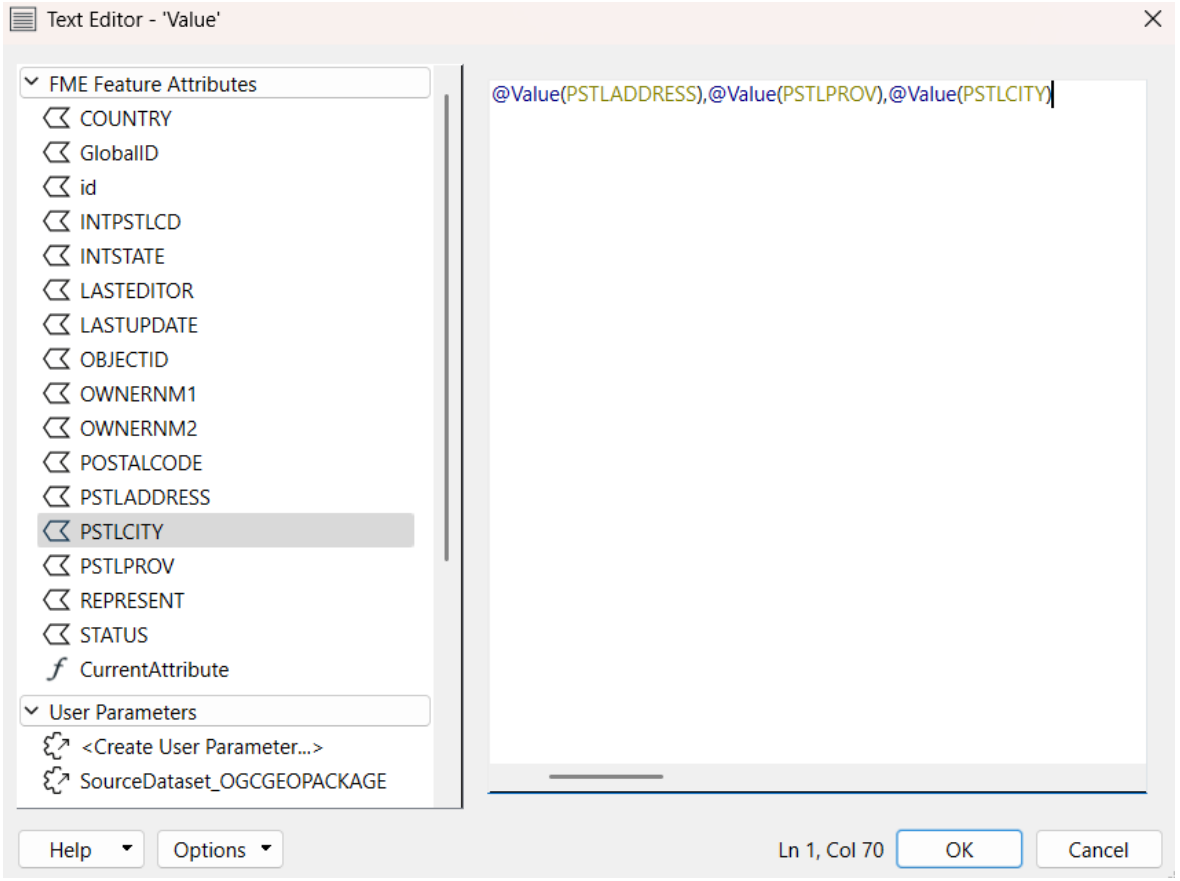
Constructing Values

Text Editor

It allows you to manipulate and construct text values.

It includes all the usual string-handling functionality such as concatenation, trimming, padding, and case changing.

Menu options on the left side of a dialog contain **String Functions** that can be used to manipulate the strings.

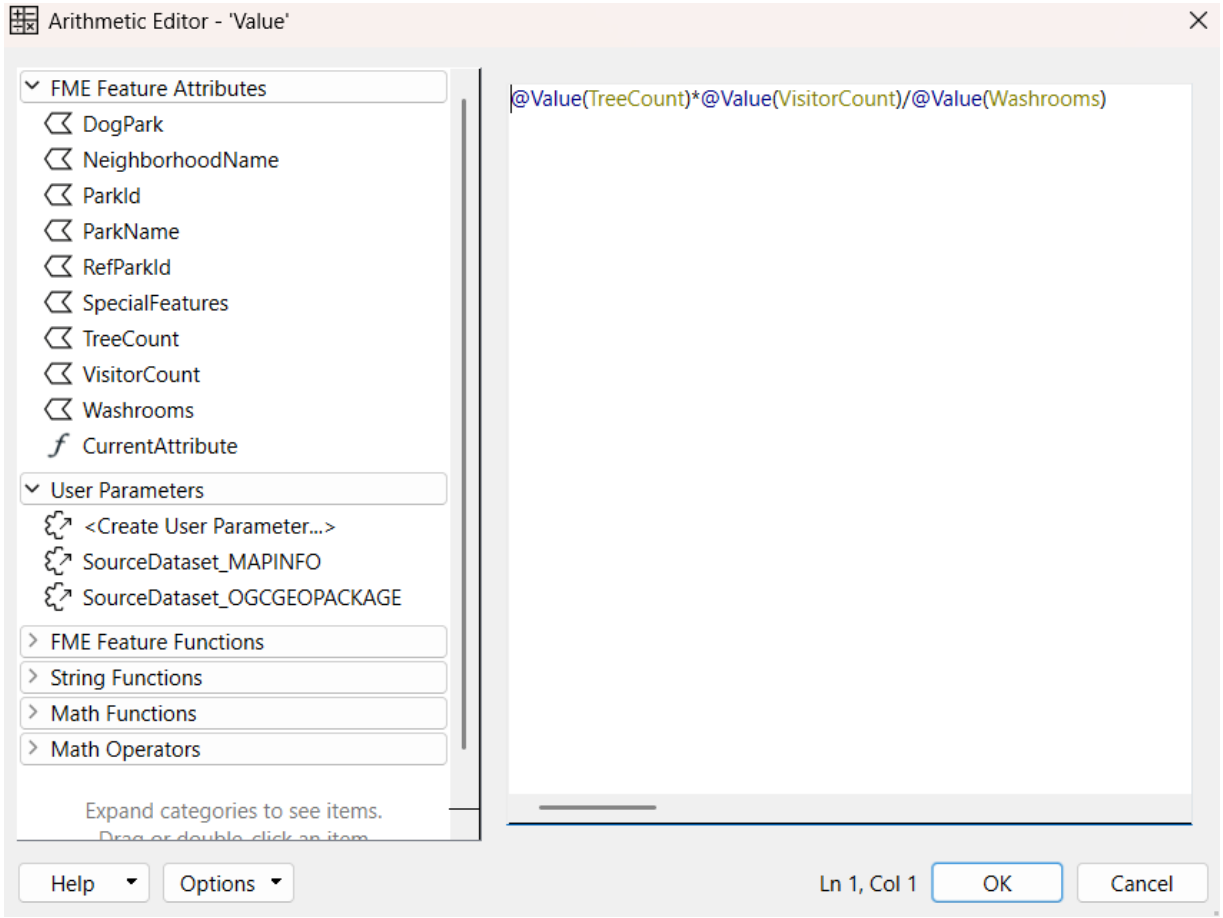


Constructing Values

Arithmetic Editor

Very similar to the Text Editor, however expects that the values/attributes are to be evaluated as an numeric expression (thus a numeric result).

There are various **Math Functions** and Math Operators available in the Menu Panel that Arithmetic Editor can include in expressions.



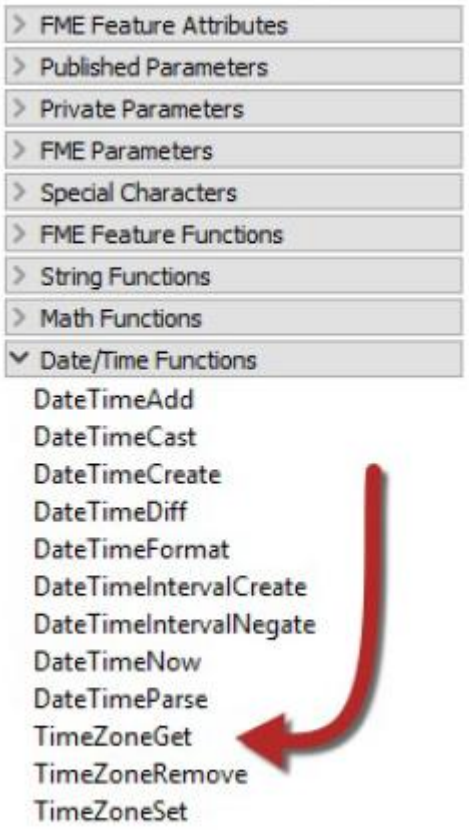
Constructing Values – Date/Time Functions

Date/Time Functions

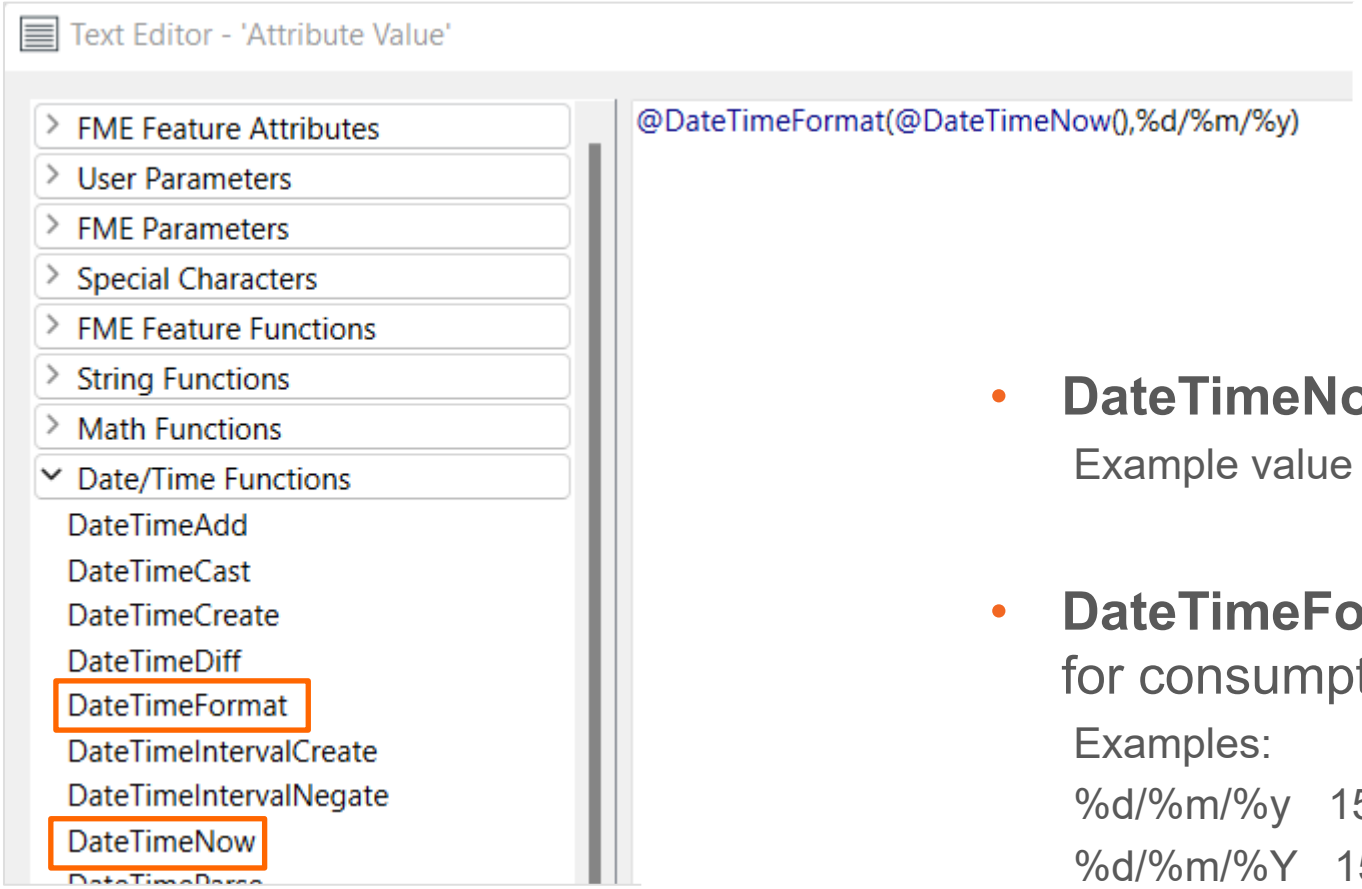
These functions allow you to calculate results for various date operations,

like:

- Determine the interval between two dates in a number of units,
- Add or subtract time from a date,
- Change a date from one structure to another.

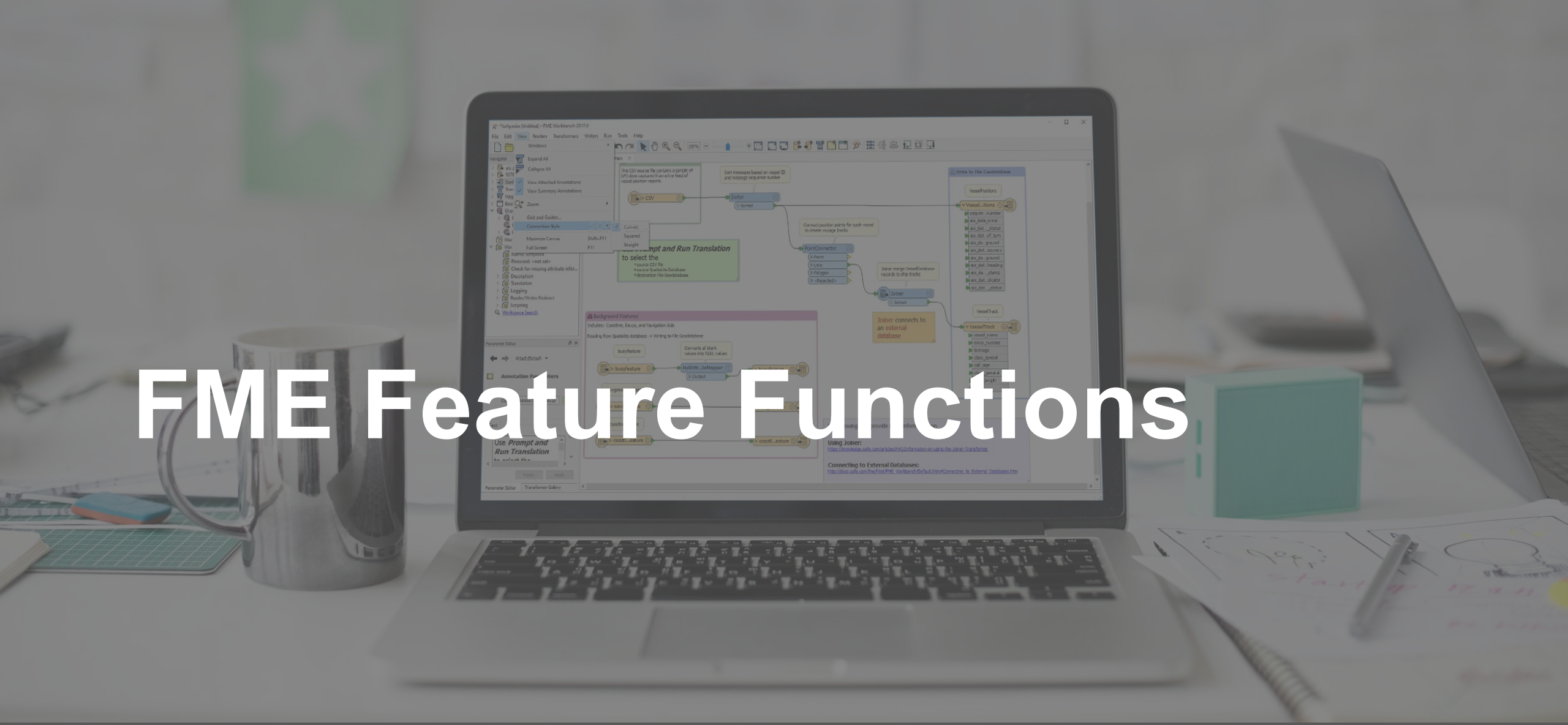


Constructing Values – Dates and Formatting



- **DateTimeNow** – Creates timestamp
  Example value returned: 20170203170000.1234567

- **DateTimeFormat** – Formats FME datetime strings for consumption by writers and transformers
  Examples:
  %d/%m/%y 15/06/22
  %d/%m/%Y 15/06/2022
  %d %B %Y 15 June 2022



FME Feature Functions

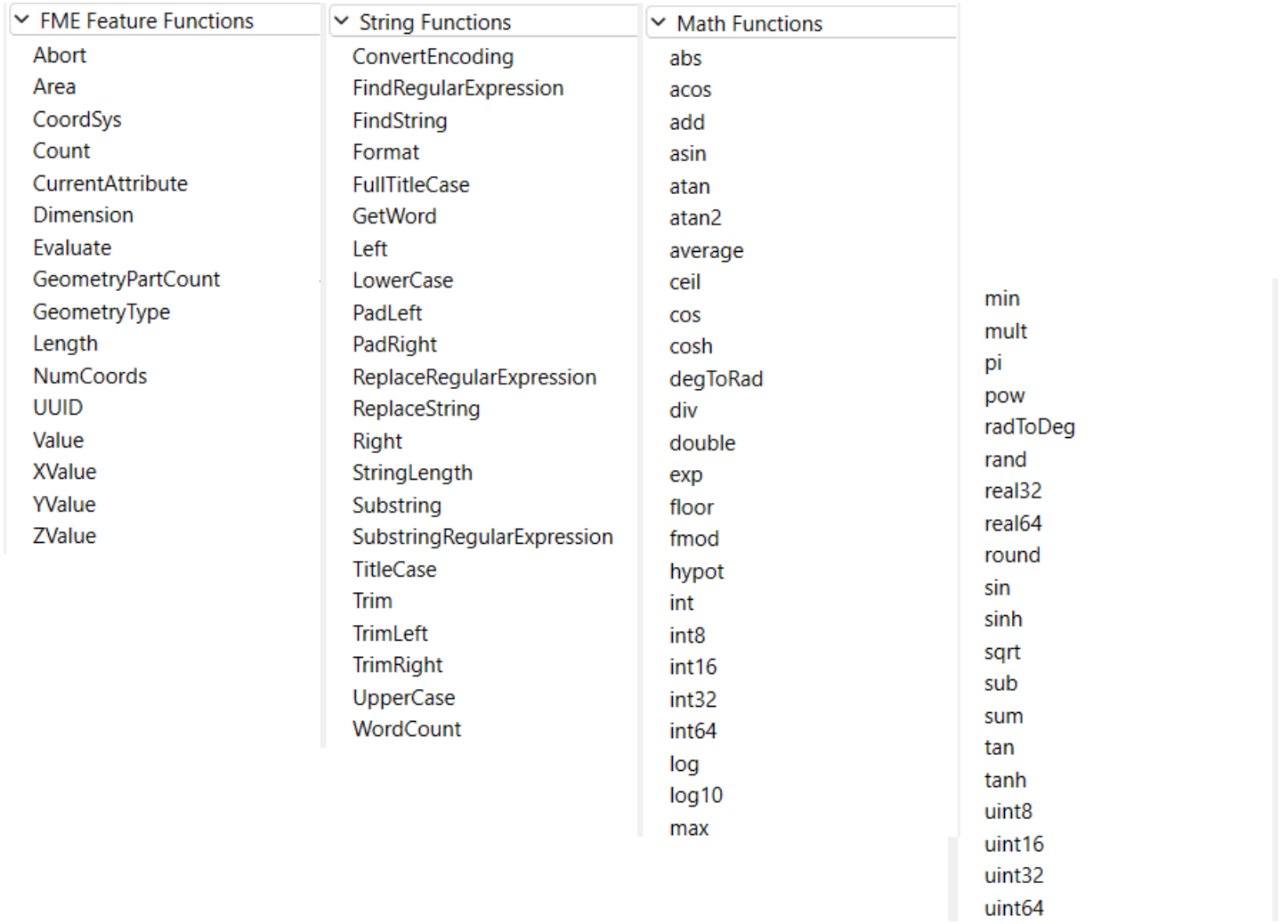
Connect. Transform. **Automate**

FME Feature Functions

Available in both, Text and Arithmetic Editors, are functions that reach into the very heart of FME's core functionality.

They are useful because they allow to build processing directly into attribute creation, instead of using a separate transformer.

Note: some functions return strings, others return numeric values; therefore the available functions vary depending on whether the text or arithmetic editor is being used.



Exercise 6.1

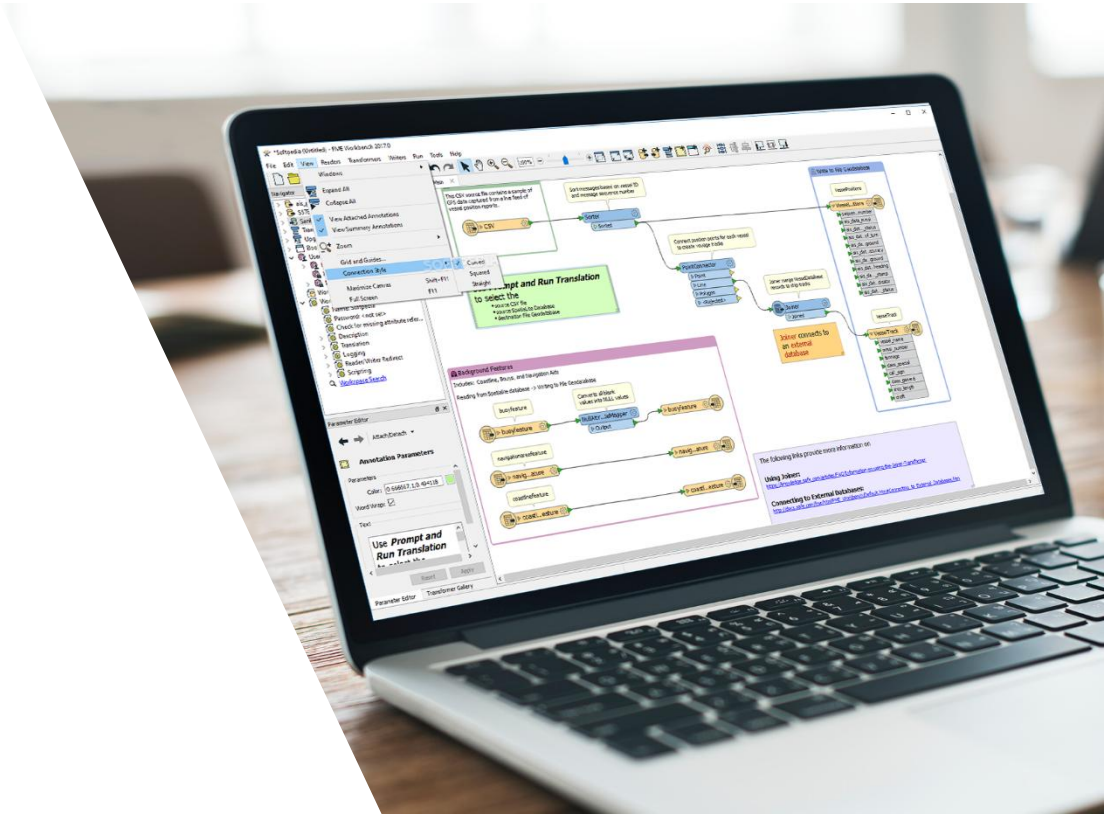
Constructing Attribute Values - Taxation Report

- 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.

Data - Building Footprints (AutoCAD DWG)
Zoning Data (MapInfo TAB)

Starting Workspace - C:\FMEModularData\Workspaces\6.01-AdvancedAttributes-ConstructingValues-Begin.fmw

Goal - Create a HTML taxation report for buildings in the city by constructing attributes using the arithmetic and text editors.



Conditional Values

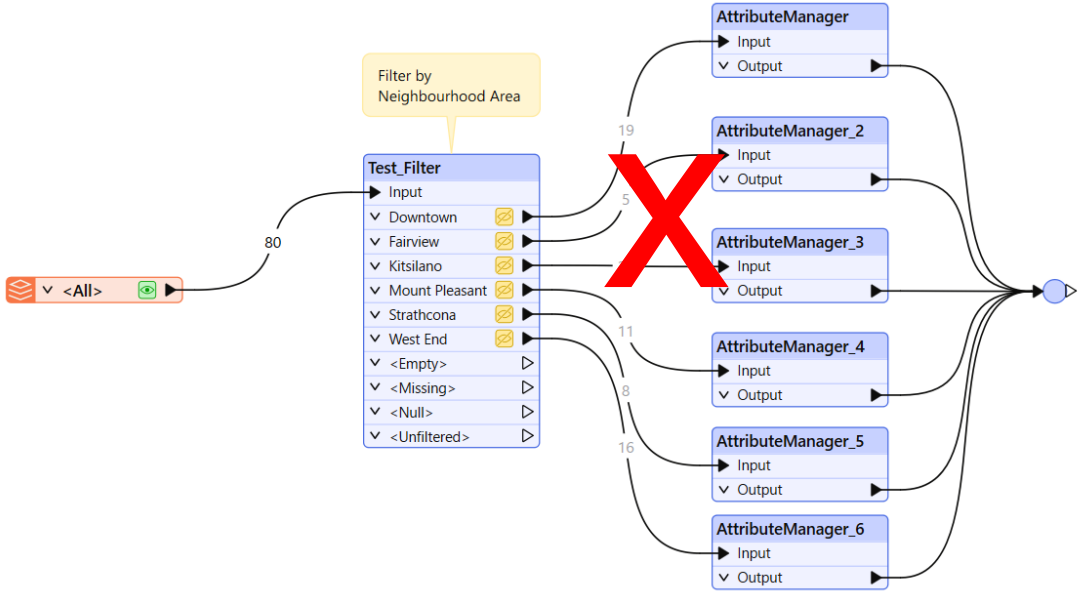
Connect. Transform. **Automate**

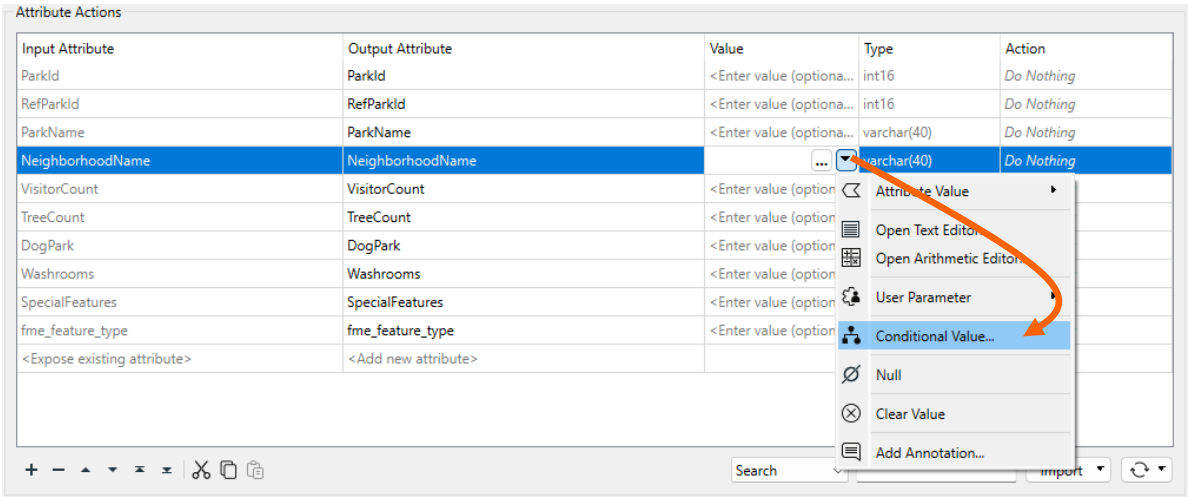
Conditional Attribute Values

Conditional Attribute Values are a tool that can be used to replace many existing transformers of the same type.

How?

Instead of creating a set of conditions and values as separate objects in a workspace (using Testers/TestFilters etc, and AttributeCreators), the author sets both condition and value inside a single transformer!



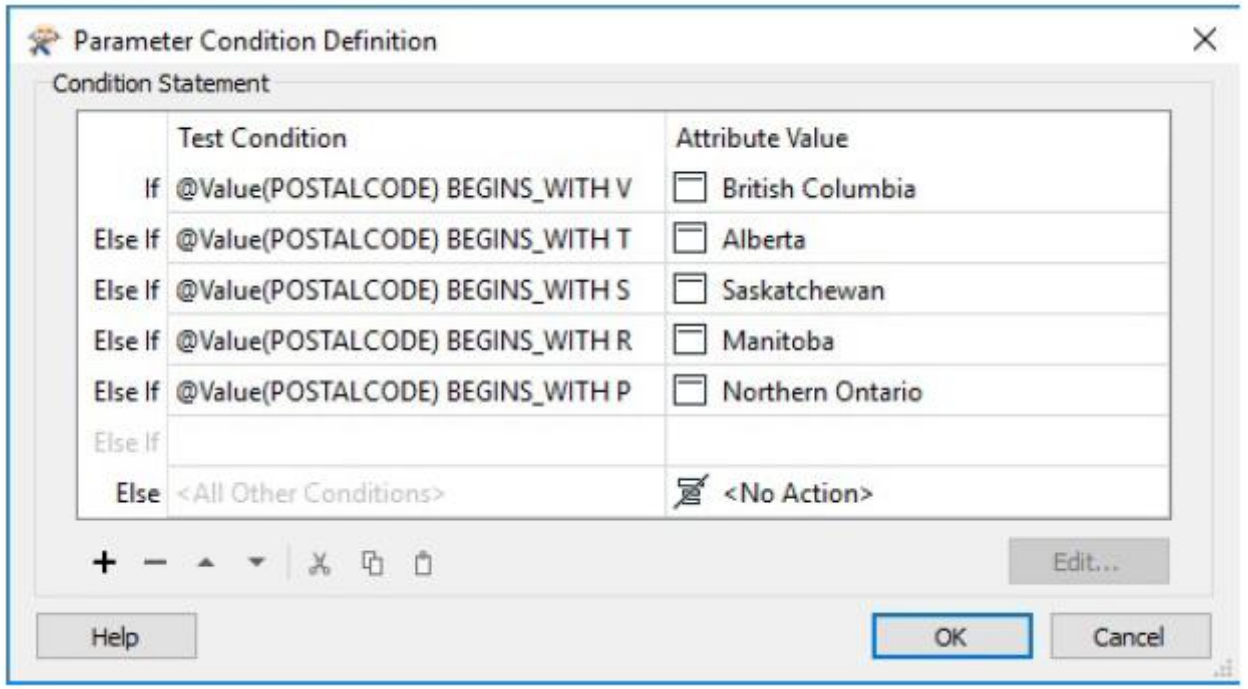


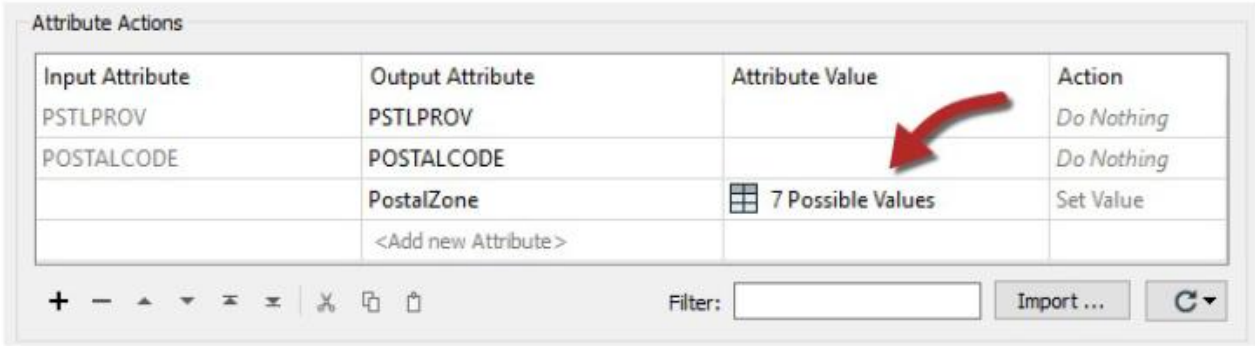
Conditional Attribute Values

A series of conditions (left) map to different values (right).

Workspace author is creating a new attribute called PostalZone. The values for PostalZone are conditional upon other attribute values.

Unlike the AttributeValueMapper, this dialog allows much more complex conditions than simple 1:1 mapping – thanks to full Test capabilities.





Exercise 6.2

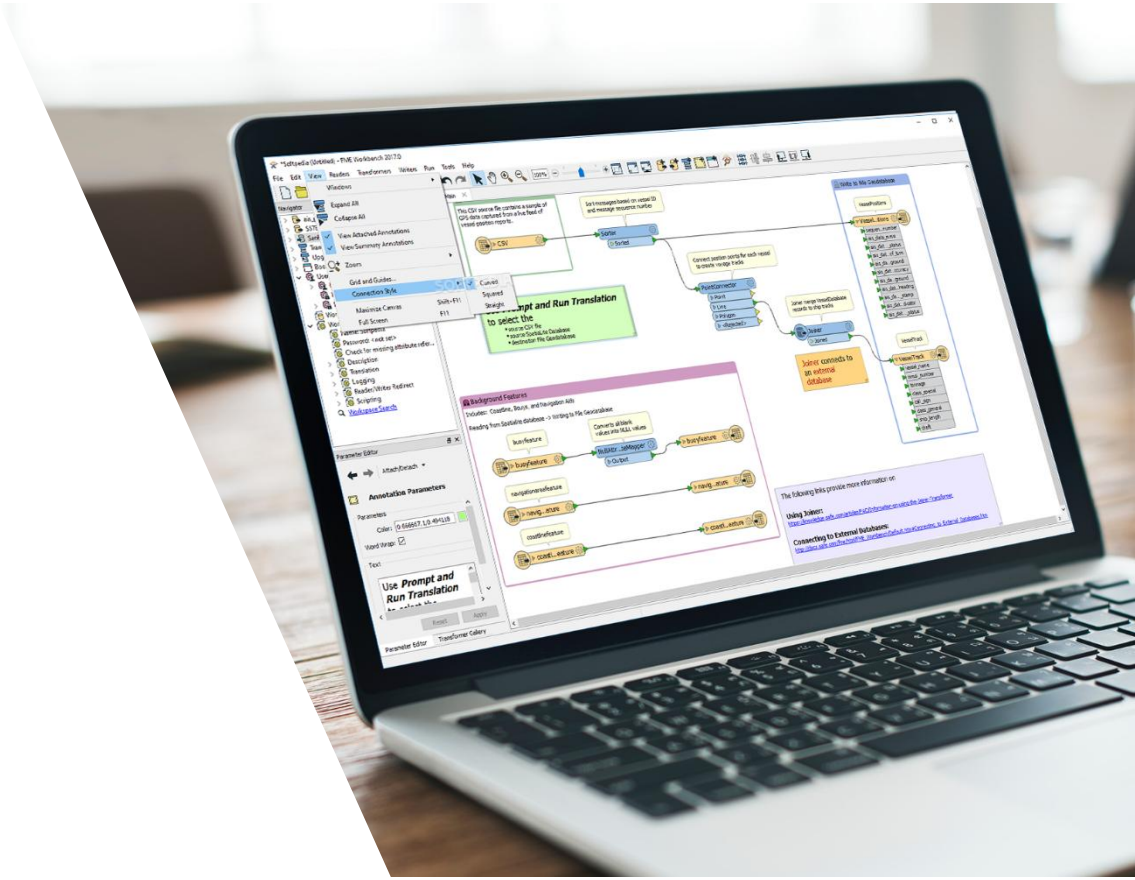
Conditional Attribute Values – Park Categories

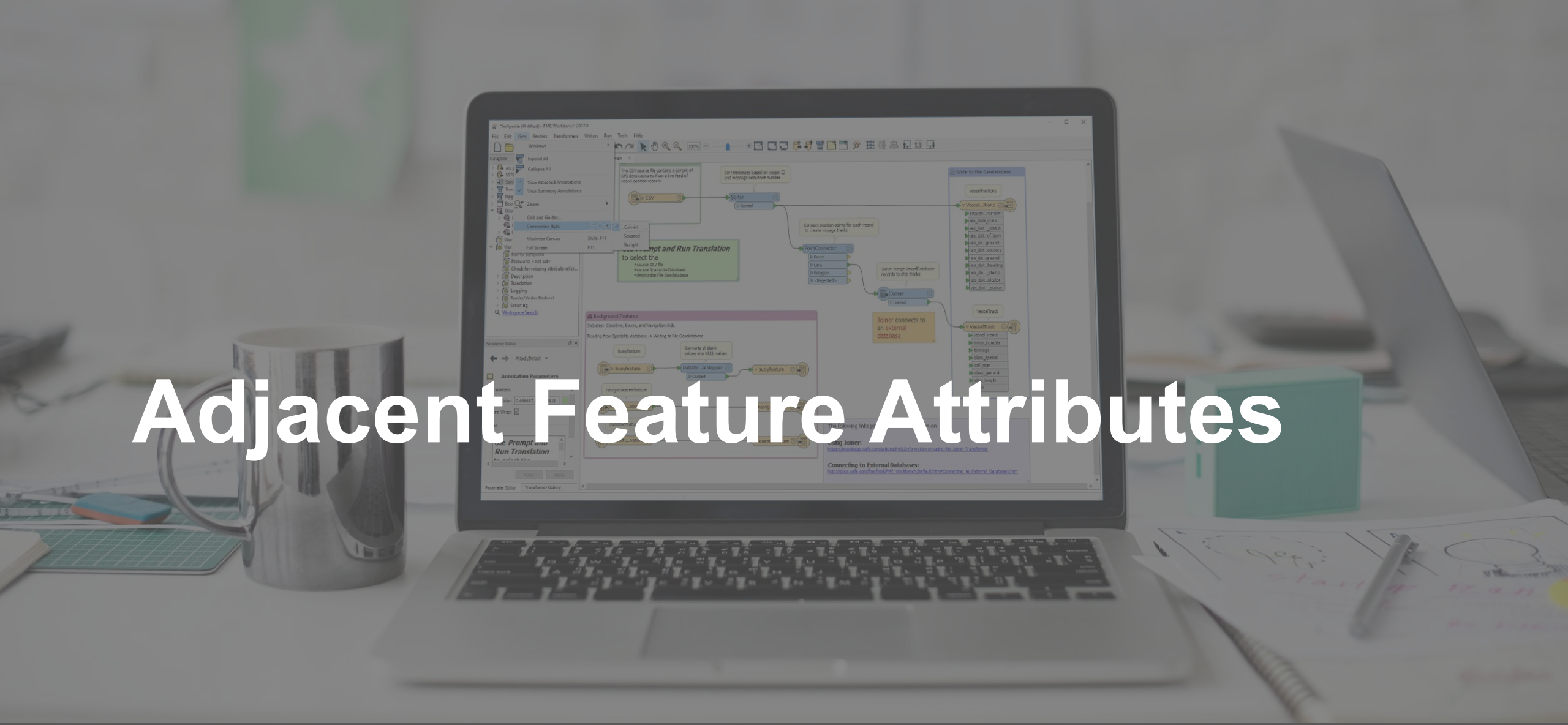
- The Council's Environment Directorate want to categorize the city Parks into 'premier', 'destination' or 'local' - based on their features.
- We'll use Conditional Attribute Values to assign the appropriate park category value based on meeting specific criteria.

Data - Parks (MapInfo TAB)

Start Workspace – None

Goal - Use Conditional Attribute Values to assign the appropriate park category value based on meeting specific criteria





Adjacent Feature Attributes

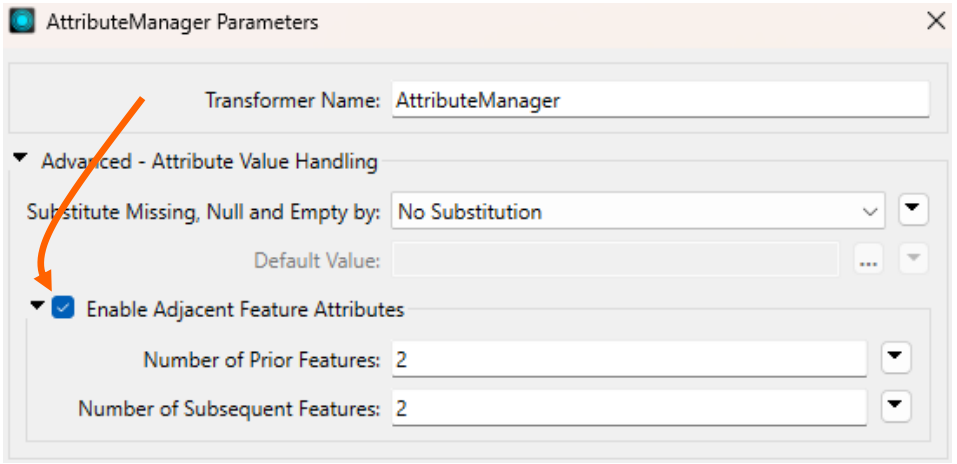
Connect. Transform. **Automate**

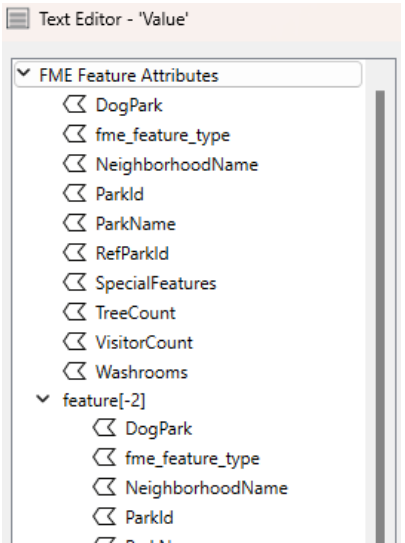
Multiple Feature Attributes

- Normally a feature in FME is self-contained, but in some cases the ability for a feature to access the attributes of other features is useful.

- The simplest way to make use of the attributes retrieved from prior/subsequent features is through the text or arithmetic editors, when the Adjacent Feature functionality is enabled.

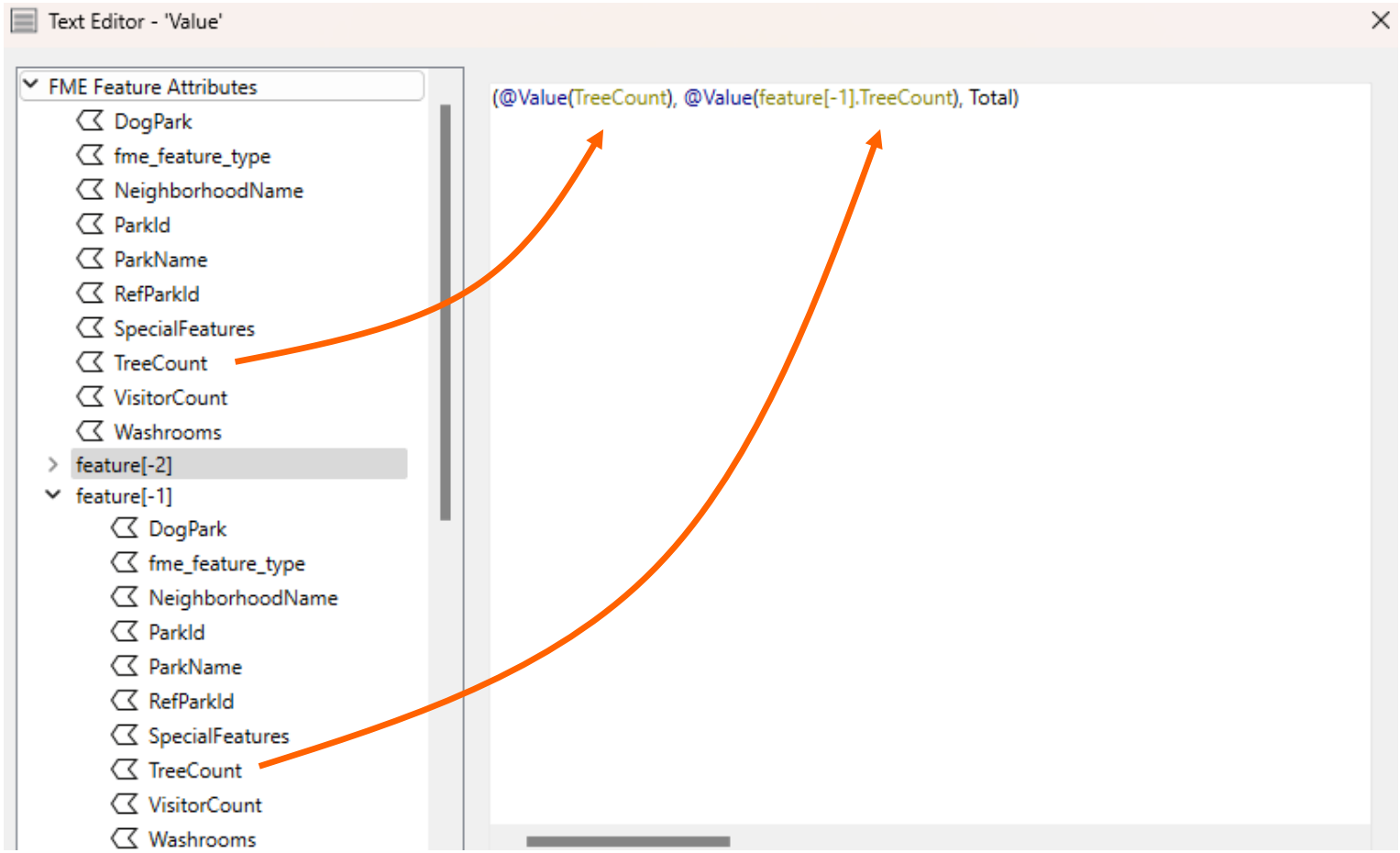
The Adjacent Feature functionality is activated by ticking the box '**Enable Adjacent Feature Attributes**' in the AttributeCreator or AttributeManager:





Multiple Feature Attributes

These attributes are then exposed for the previous and subsequent features which can be used as you would a normal User Attribute:



Exercise 6.3

Adjacent Feature Attribute - Precipitation Calculations

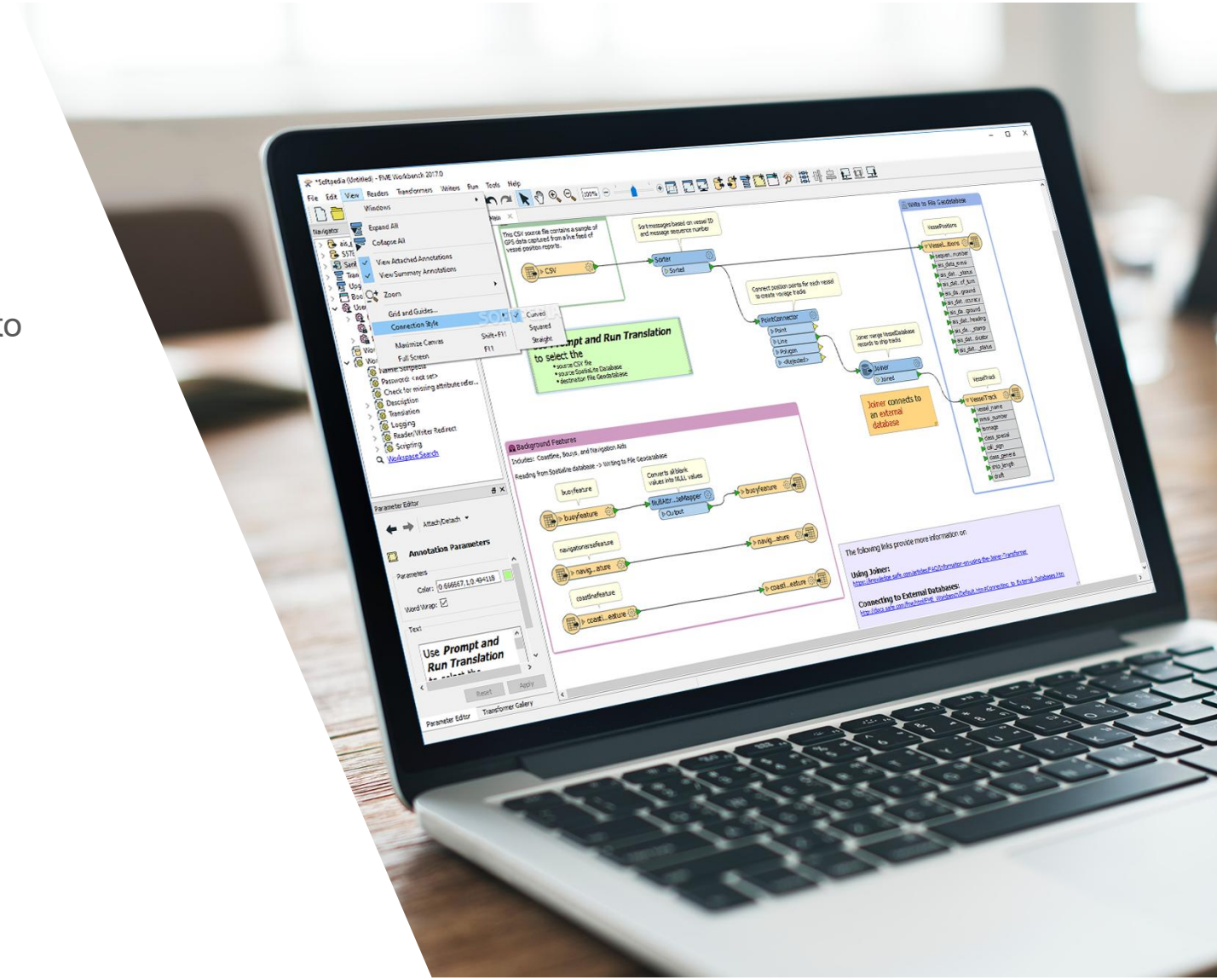
- You're working on a project mapping monthly precipitation (rainfall) in the city. Unfortunately, the numbers are a cumulative amount, and you wanted to map individual figures for each month.

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

Data - Precipitation Data (Microsoft Excel)

Starting Workspace – none

Goal - Calculate monthly precipitation using adjacent feature attributes



Null Attributes

Connect. Transform. **Automate**

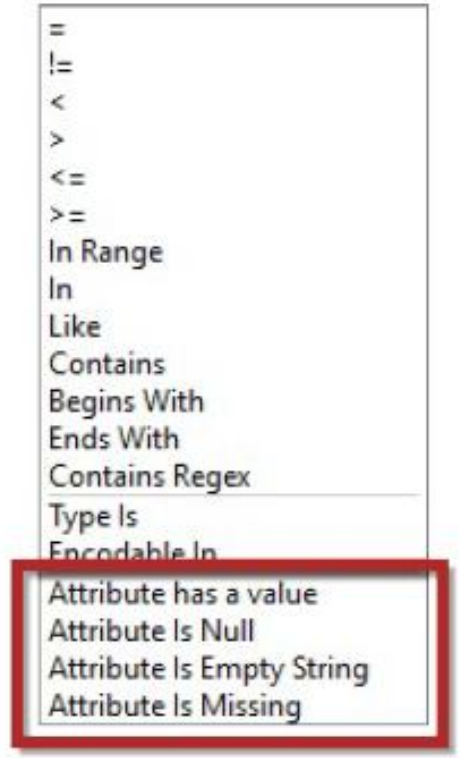
Null Attributes

A null attribute value is the equivalent of nothing.

It is important to be precise in our terminology. Depends on the data environment, there are many ways to represent nothing:

- **Null** – an attribute has a value that indicates nothingness *– never been an entry*
- **Empty** – an attribute exists but has no value *– value deleted*
- **Missing** – an attribute doesn't exist *– i.e. two datasets merged and column only existed in one of them*

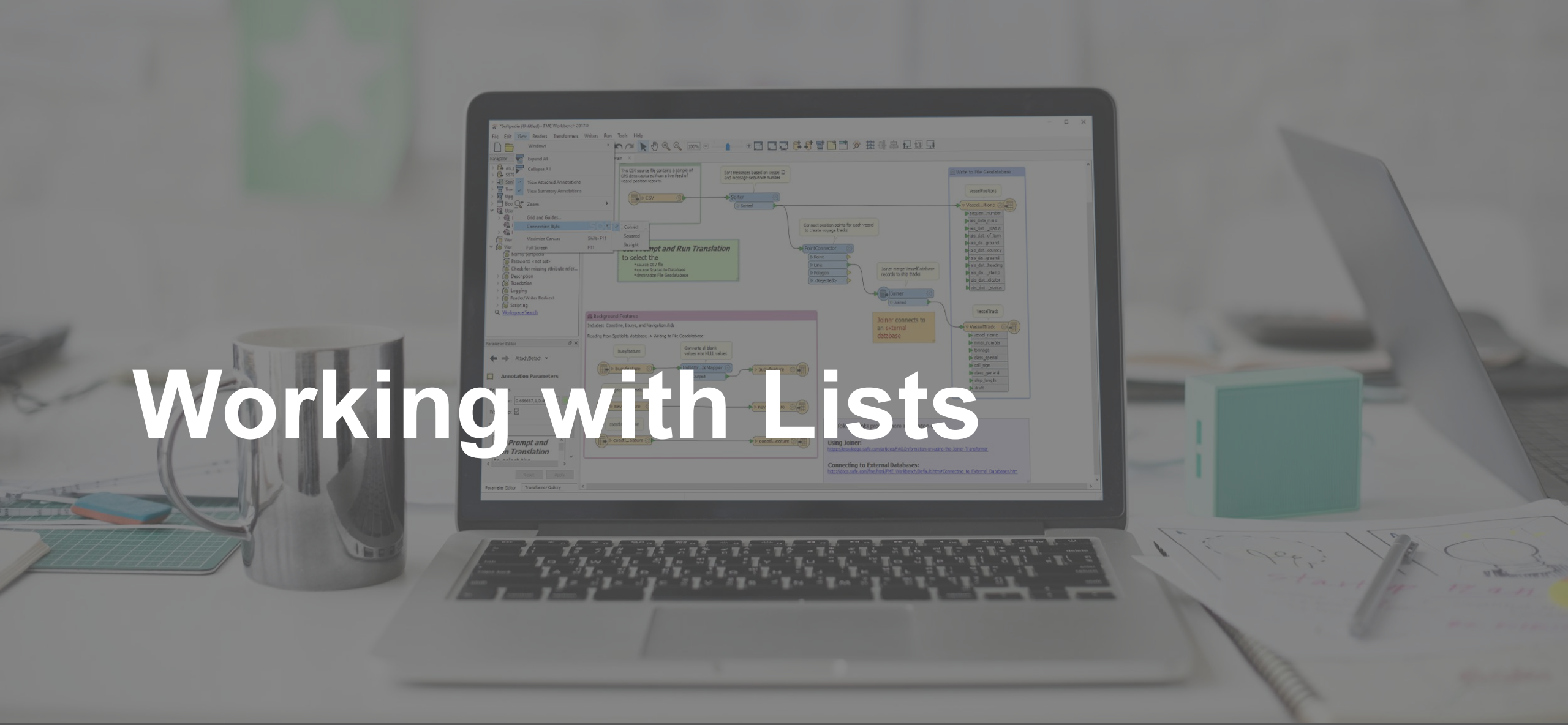
- **Not a Number** – a numeric attribute NaN
- **Zero** – a numeric attribute has a value of zero
- **-9999** – an attribute has a particular value that indicates nothingness



How Does FME Represent Nothing?

```
========================================================================
Logger: Feature is:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Feature Type: `Logger_LOGGED'
Attribute(encoded: windows-1252): `DogPark' has value `N'
Attribute(encoded: windows-1252): `NeighborhoodName' has value `West End'
Attribute(string)               : `ParkId' has value `72'
Attribute(string)               : `ParkName' is <null>
Attribute(string)               : `RefParkId' has value `-9999'
Attribute(string)               : `TreeCount' has value `8'
```

	ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures
1	1	-9999	<null>	Kitsilano	9406	10	N	<missing>	<missing>
2	4	-9999	<null>	Strathcona	9755	6	N	<missing>	<missing>
3	8	-9999	<null>	Mount Pleasant	8061	3	N	<missing>	<missing>
4	15	-9999	<null>	Kitsilano	7920	5	N	<missing>	<missing>
5	30	-9999	<null>	Strathcona	10636	7	N	<missing>	<missing>
6	40	-9999	<null>	Kitsilano	7399	10	N	<missing>	<missing>
7	64	-9999	<null>	West End	13136	7	N	<missing>	<missing>

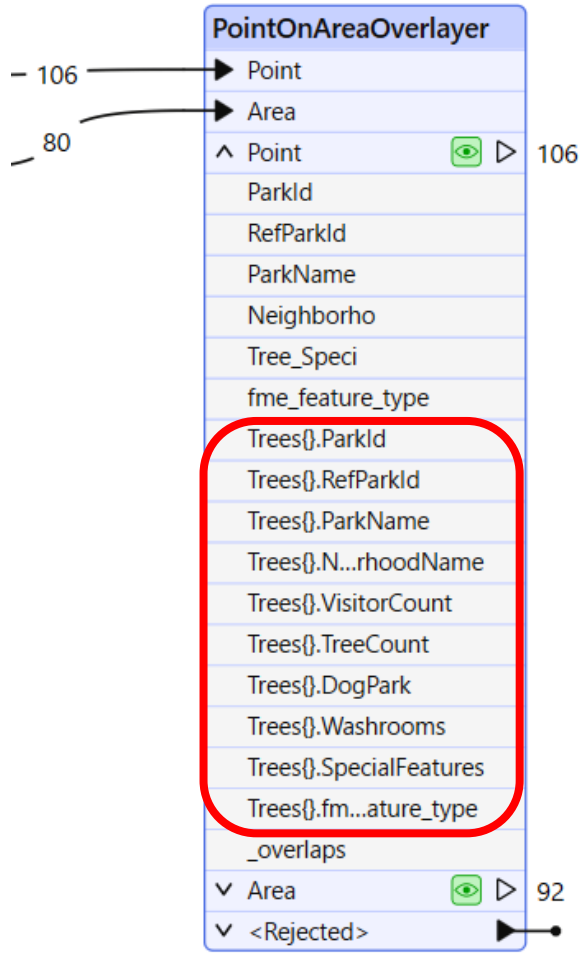


Working with Lists

Connect. Transform. **Automate**

List Attributes

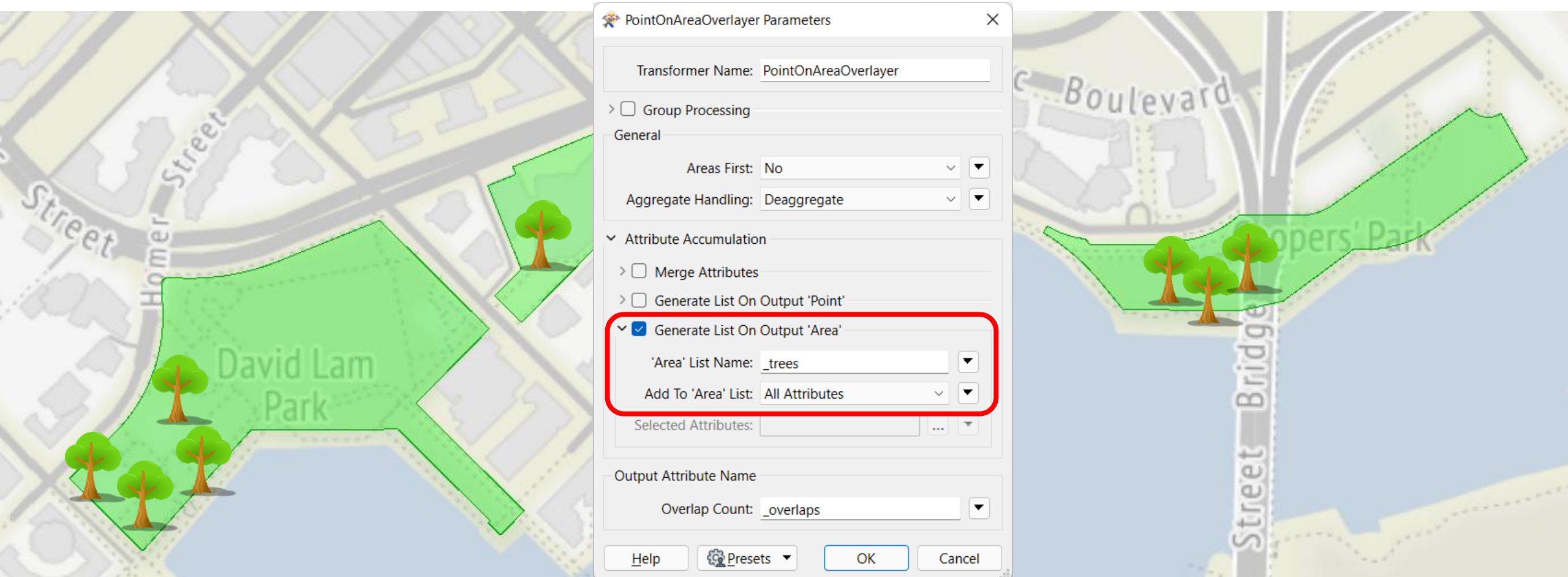
- FME's way of allowing **multiple values per attribute**
- List attributes are denoted using curly brackets after the list name. e.g. *_trees{}.CommonName*
- Where can you find them?
  - Some **Transformers** create a list as a result of their processing
  - Some **FME readers** represent recurring values in the original data source in a list eg:
    - XML
    - JSON
    - DGN format returns IGDS linkages in list form



List Attributes - example

There are multiple Trees (points) in each Park (polygon)

Using a PointOnAreaOverlayer transformer we can choose to Generate List output on our parks, returning multiple values (an entry for each tree feature) per attribute



PointOnAreaOverlayer Parameters

Transformer Name: PointOnAreaOverlayer

Group Processing

General

Areas First: No

Aggregate Handling: Deaggregate

Attribute Accumulation

Merge Attributes

Generate List On Output 'Point'

Generate List On Output 'Area'

'Area' List Name: _trees

Add To 'Area' List: All Attributes

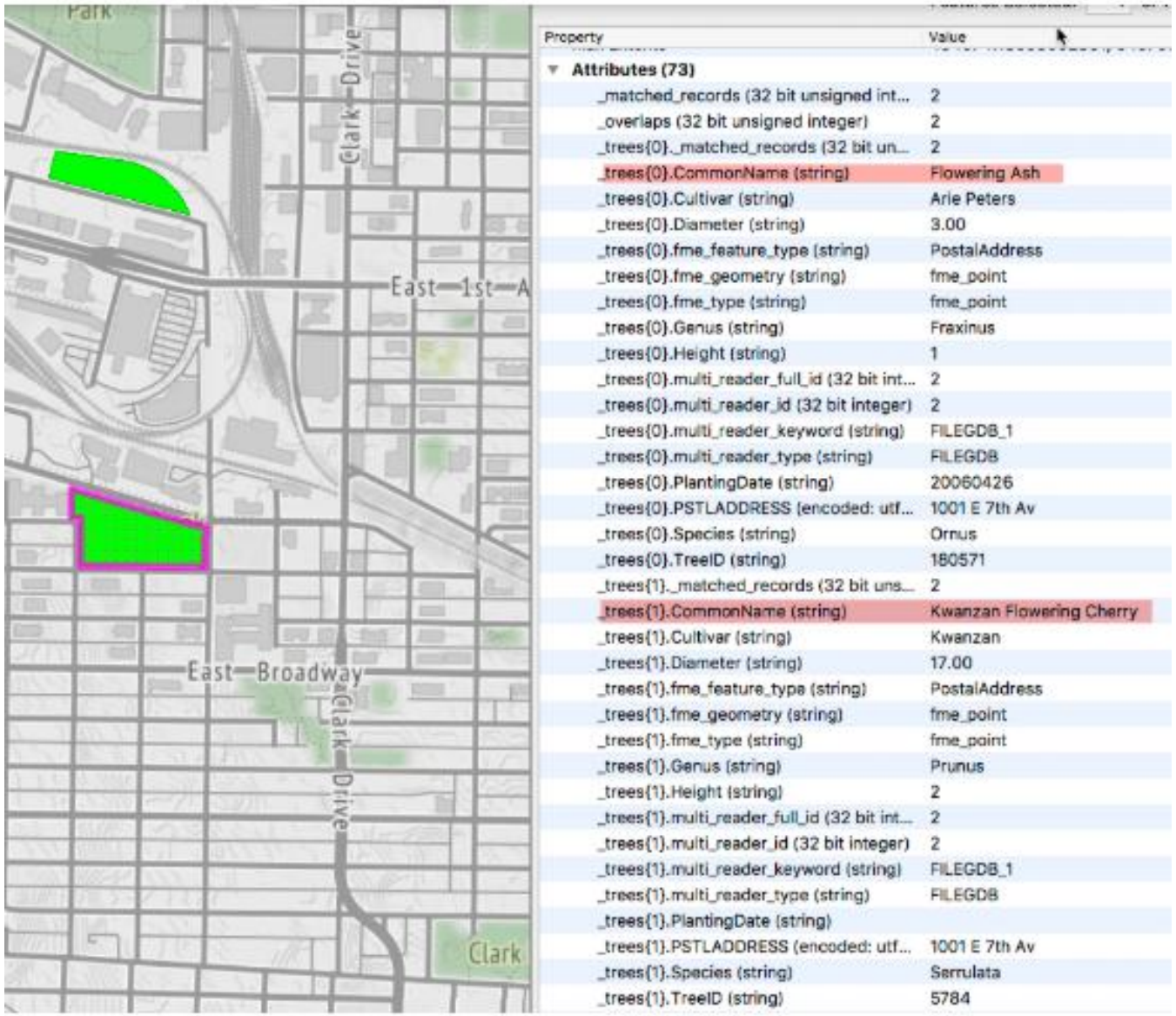
Selected Attributes:

Output Attribute Name

Overlap Count: _overlaps

Help | Presets | OK | Cancel

List Attributes - example



Property	Value
Attributes (73)	
_matched_records (32 bit unsigned int...	2
_overlaps (32 bit unsigned integer)	2
_trees{0}._matched_records (32 bit un...	2
_trees{0}.CommonName (string)	Flowering Ash
_trees{0}.Cultivar (string)	Arie Peters
_trees{0}.Diameter (string)	3.00
_trees{0}.fme_feature_type (string)	PostalAddress
_trees{0}.fme_geometry (string)	fme_point
_trees{0}.fme_type (string)	fme_point
_trees{0}.Genus (string)	Fraxinus
_trees{0}.Height (string)	1
_trees{0}.multi_reader_full_id (32 bit int...	2
_trees{0}.multi_reader_id (32 bit integer)	2
_trees{0}.multi_reader_keyword (string)	FILEGDB_1
_trees{0}.multi_reader_type (string)	FILEGDB
_trees{0}.PlantingDate (string)	20060426
_trees{0}.PSTLADDRESS (encoded: utf...	1001 E 7th Av
_trees{0}.Species (string)	Ornus
_trees{0}.TreeID (string)	180571
_trees{1}._matched_records (32 bit uns...	2
_trees{1}.CommonName (string)	Kwanzan Flowering Cherry
_trees{1}.Cultivar (string)	Kwanzan
_trees{1}.Diameter (string)	17.00
_trees{1}.fme_feature_type (string)	PostalAddress
_trees{1}.fme_geometry (string)	fme_point
_trees{1}.fme_type (string)	fme_point
_trees{1}.Genus (string)	Prunus
_trees{1}.Height (string)	2
_trees{1}.multi_reader_full_id (32 bit int...	2
_trees{1}.multi_reader_id (32 bit integer)	2
_trees{1}.multi_reader_keyword (string)	FILEGDB_1
_trees{1}.multi_reader_type (string)	FILEGDB
_trees{1}.PlantingDate (string)	
_trees{1}.PSTLADDRESS (encoded: utf...	1001 E 7th Av
_trees{1}.Species (string)	Serrulata
_trees{1}.TreeID (string)	5784

Lists are used to return attributes of each Tree within a Park;

- CommonName
- Cultivator
- Diameter
- Genus
- Height
- PlantingDate
- PSTADRDRESS
- Species
- TreeID

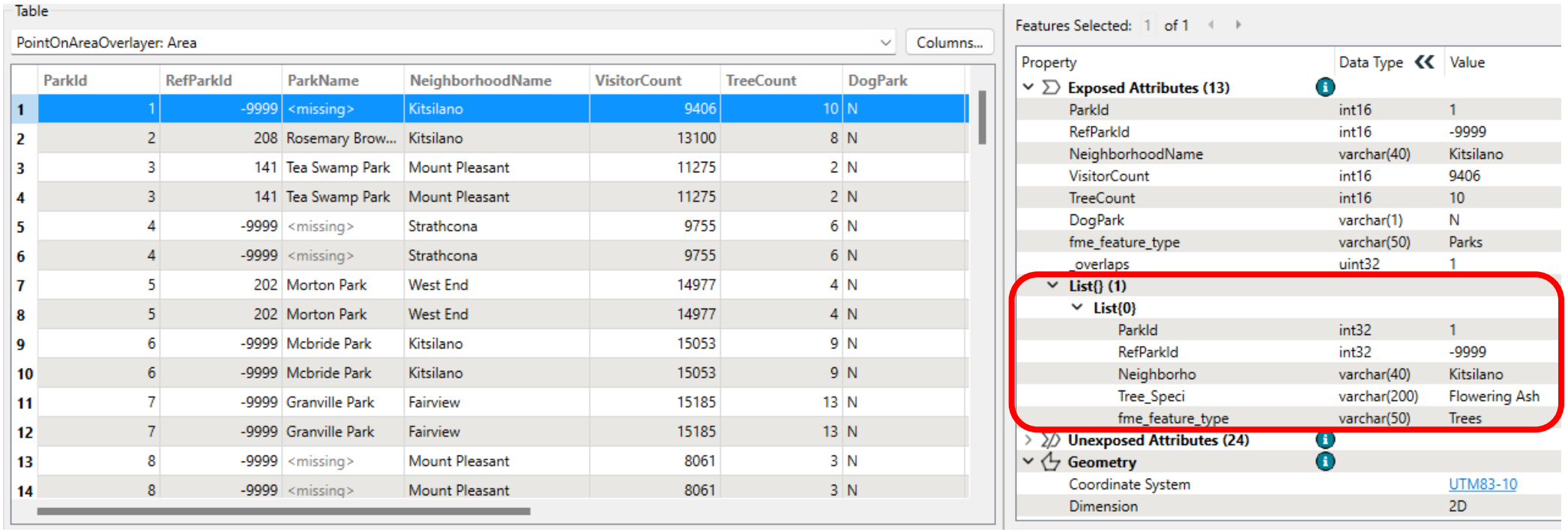
Where there are multiple trees within a park, multiple values are returned. The number inside the curly brackets represents the element's index inside the list:
{0}, {1}, {2}, etc

How Do I Inspect List Attributes

- List attributes are NOT included as attributes in Table View

- Instead, you inspect List attributes using the **Feature Information Window**

  (select the feature you want to inspect in either the Table or Map view)



Table

PointOnAreaOverlayer: Area — Columns...

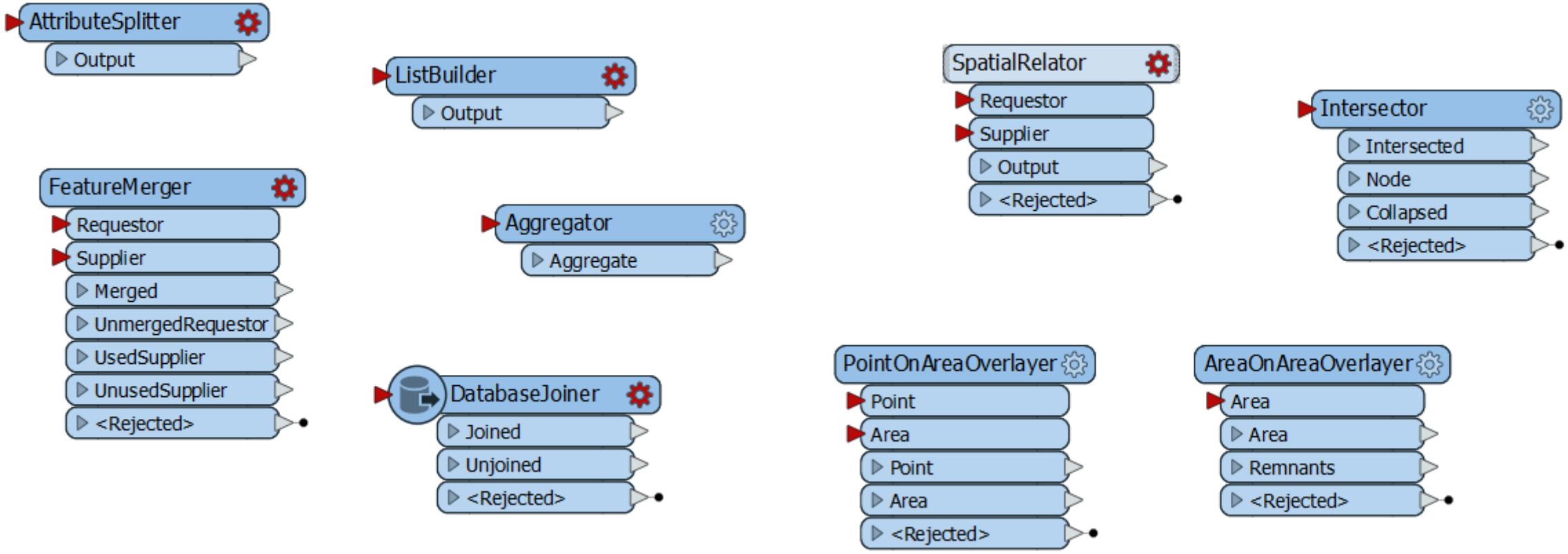
	ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark
1	1	-9999	<missing>	Kitsilano	9406	10	N
2	2	208	Rosemary Brow...	Kitsilano	13100	8	N
3	3	141	Tea Swamp Park	Mount Pleasant	11275	2	N
4	3	141	Tea Swamp Park	Mount Pleasant	11275	2	N
5	4	-9999	<missing>	Strathcona	9755	6	N
6	4	-9999	<missing>	Strathcona	9755	6	N
7	5	202	Morton Park	West End	14977	4	N
8	5	202	Morton Park	West End	14977	4	N
9	6	-9999	Mcbride Park	Kitsilano	15053	9	N
10	6	-9999	Mcbride Park	Kitsilano	15053	9	N
11	7	-9999	Granville Park	Fairview	15185	13	N
12	7	-9999	Granville Park	Fairview	15185	13	N
13	8	-9999	<missing>	Mount Pleasant	8061	3	N
14	8	-9999	<missing>	Mount Pleasant	8061	3	N

Features Selected: 1 of 1

Property	Data Type	Value
Exposed Attributes (13)		
ParkId	int16	1
RefParkId	int16	-9999
NeighborhoodName	varchar(40)	Kitsilano
VisitorCount	int16	9406
TreeCount	int16	10
DogPark	varchar(1)	N
fme_feature_type	varchar(50)	Parks
_overlaps	uint32	1
List{} (1)		
List{0}		
ParkId	int32	1
RefParkId	int32	-9999
Neighborho	varchar(40)	Kitsilano
Tree_Speci	varchar(200)	Flowering Ash
fme_feature_type	varchar(50)	Trees
Unexposed Attributes (24)		
Geometry		
Coordinate System		UTM83-10
Dimension		2D

Which transformers create lists?

Several (both spatial and non spatial), here are a few examples:



How Do I Extract List Attributes?

Several transformers are available to help you extract data from list attributes and use it in your workspace or written data:

Searching a list (**ListSearcher**) for a particular value

Counting the number of elements in a list (**ListElementCounter**)

Concatenating all the elements of a list into a single attribute (**ListConcatenator**)

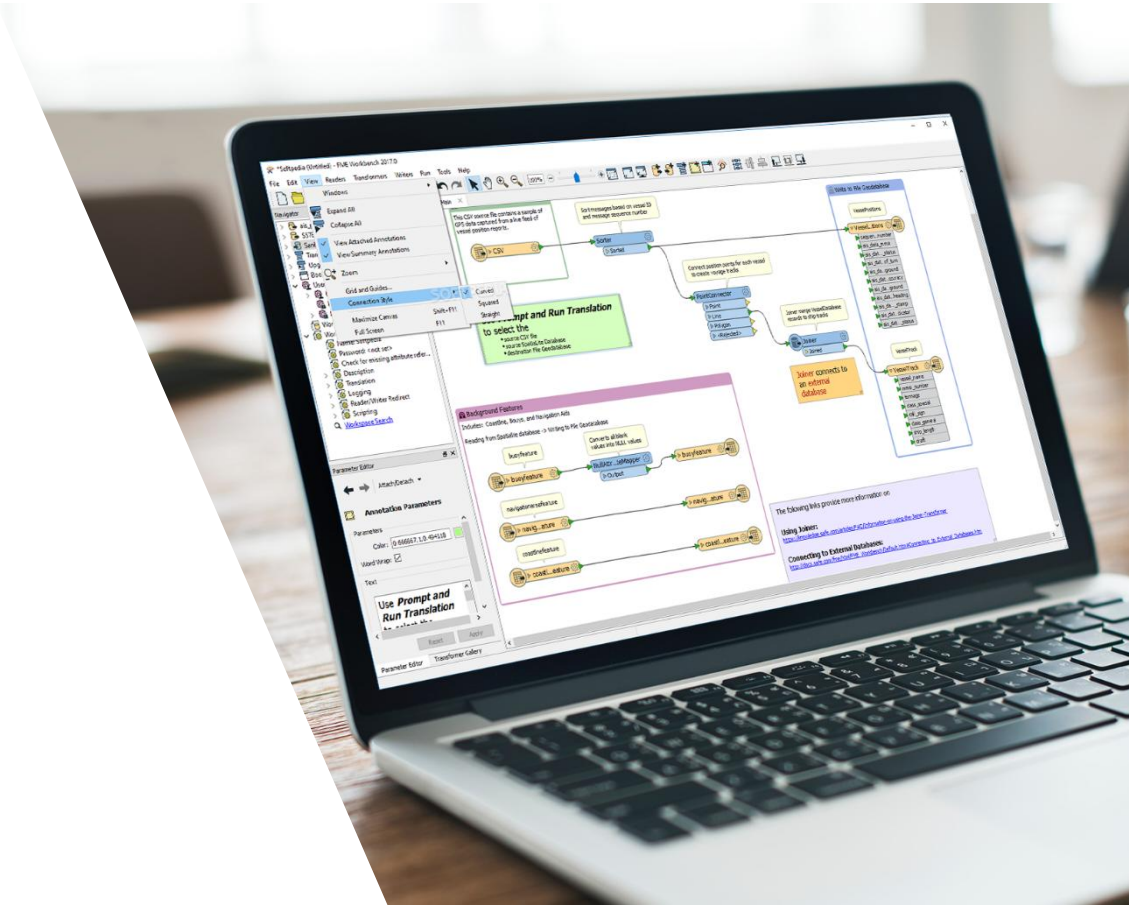
Exploding a list by creating a new feature for each element in the list and adding its attributes (**ListExploder**)

Find the occurrences of the values in your list and sort by my common (**ListHistogrammer**)

Exercise 6.4

Working with Lists

- The city have tree records stored in a GIS layer. There is a requirement to merge this tree information with the parks polygons.
- After performing a simple point on polygon overlay and generating list attributes, you'll explore ways in which the tree information could be extracted from the list attributes and added to the park features.



Data - Parks (MapInfo TAB), Trees (MapInfo TAB)

Starting Workspace - C:\FMEModularData\Workspaces\6.04-AdvancedAttributes-Lists-Begin.fmw

Goal –

- Create a copy of the park polygons which includes information about all the trees within each park, using List attributes.
- Investigate different ways in which the List attributes can be manipulated and values extracted.

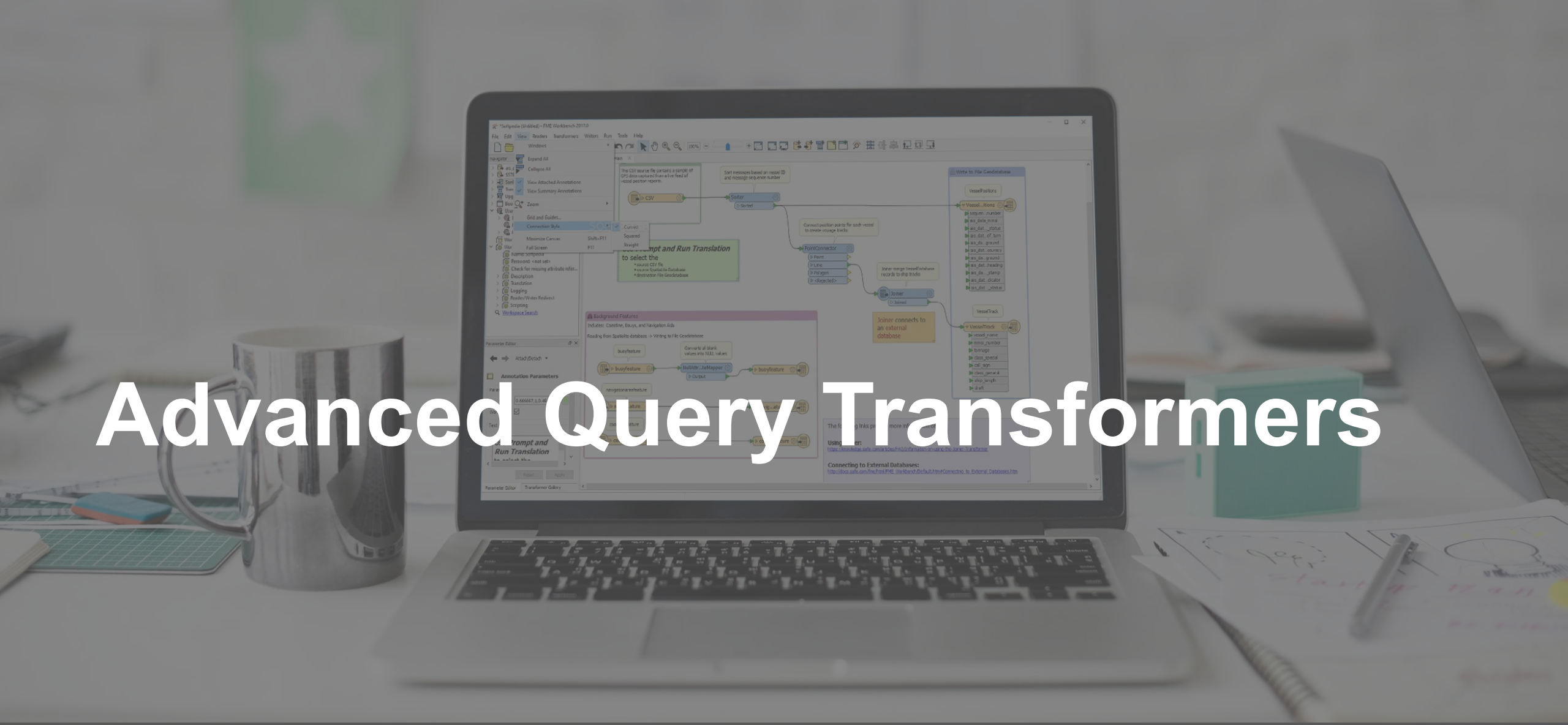
Benefits of lists

1. Generating lists for one-to-many relationships in database and/or spatial joins can preserve all the joined data.

2. List elements can easily be converted into tabular attributes, so you always have the option to store one or many list elements in a table.

3. Storing data in a structured indexed list prevents you from creating a large number of unnecessary table attributes, which can quickly become overwhelming.

Be careful! Just like when using attributes, you could end up with a ton of unnecessary data. It's easy to get lost in the data, so clean up your lists as you go!

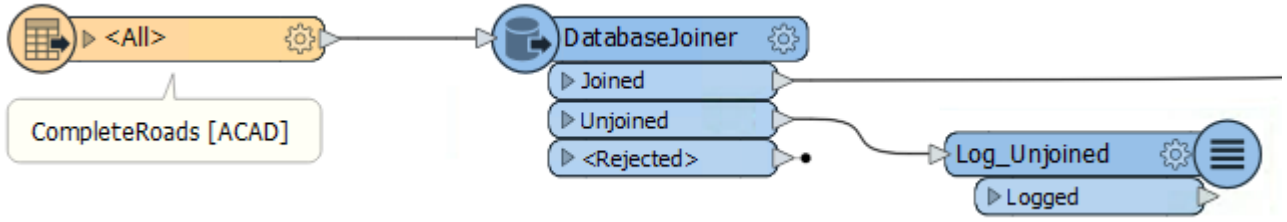


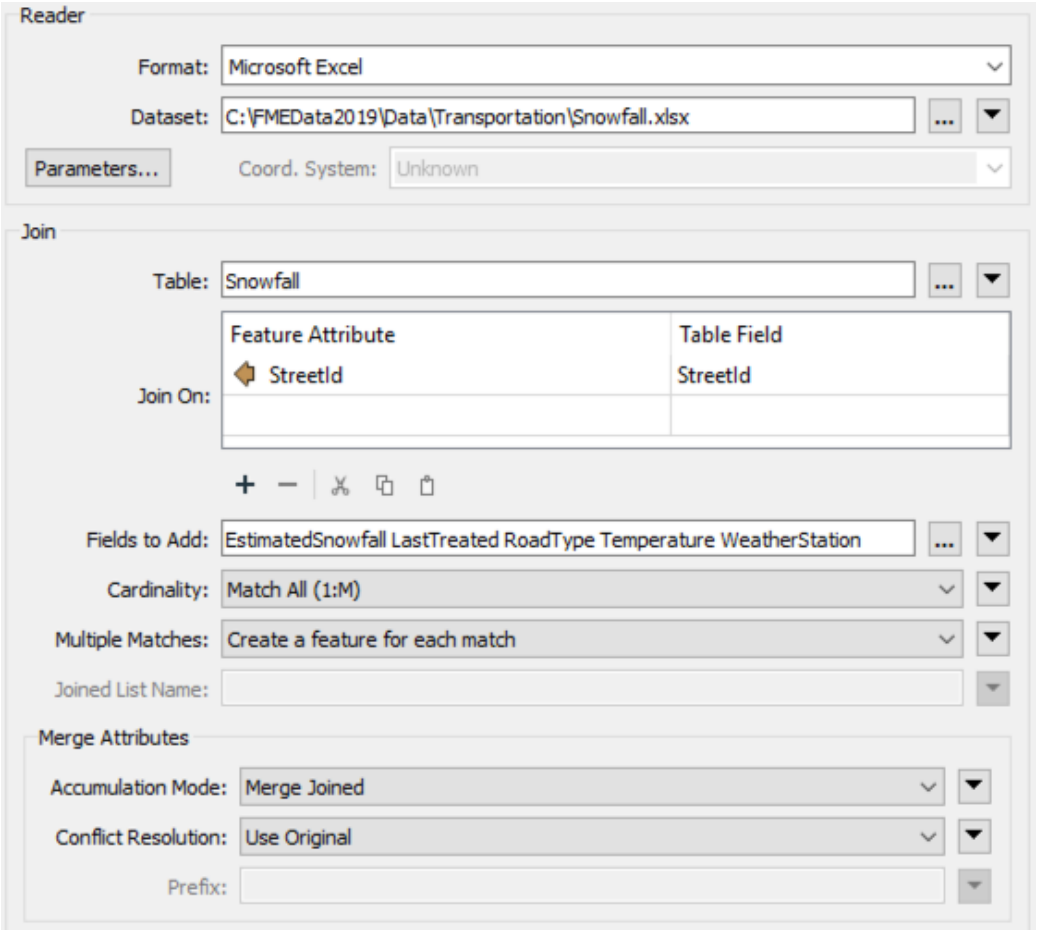
Advanced Query Transformers

Connect. Transform. **Automate**

The DatabaseJoiner Transformer

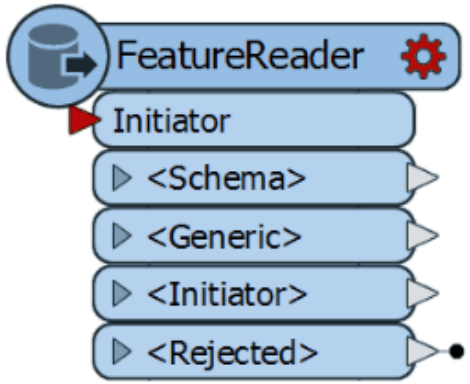
- Merges 1 or more streams of data with records from an external Database
- i.e here we're merging with Snowfall Data in an external excel table

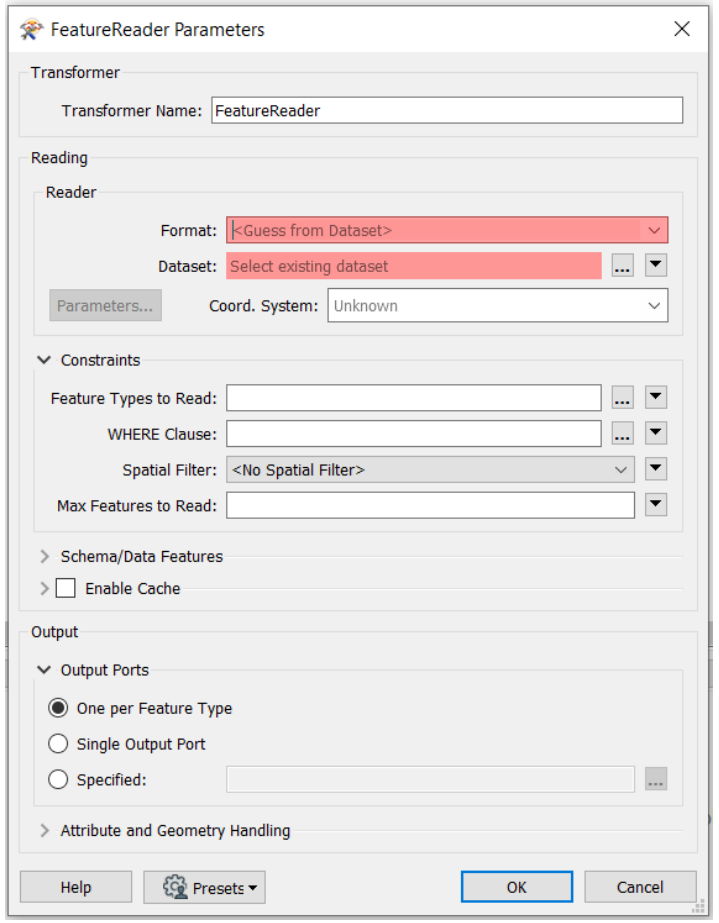




The FeatureReader Transformer

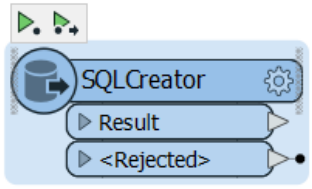
- A complete read is done for each feature that enters the ***Initiator*** port.
- Can specify a WHERE clause
- Can use a spatial filter

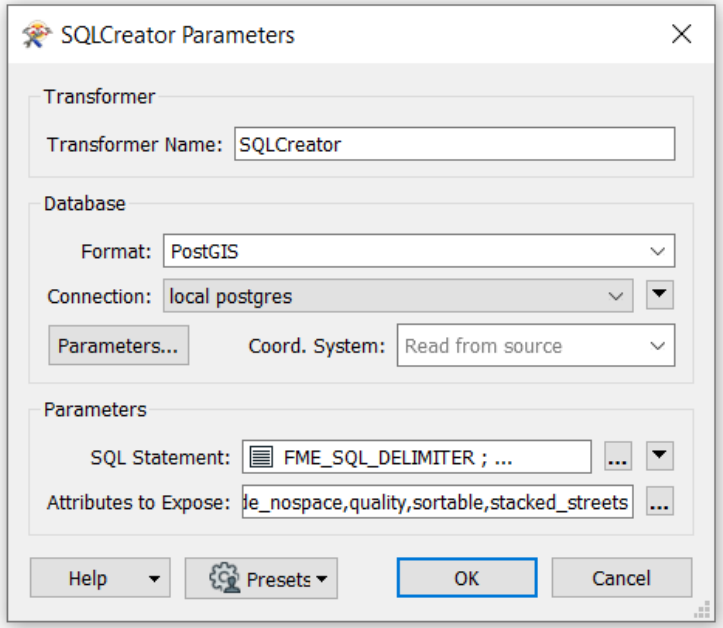


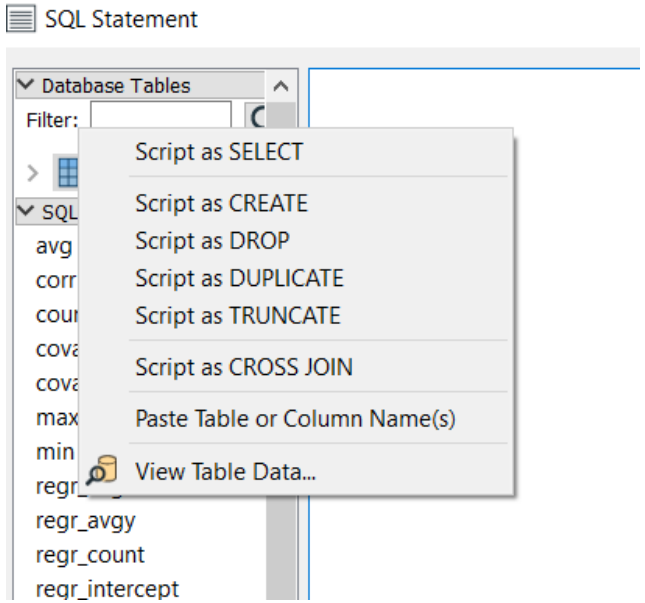


SQL Creator

- Generates FME features from the results of a SQL query executed once against a database.
- Databases you want to connect to can be defined in your Database Connections from the FME Options window
- Shortcuts when writing your SQL statement – right click table

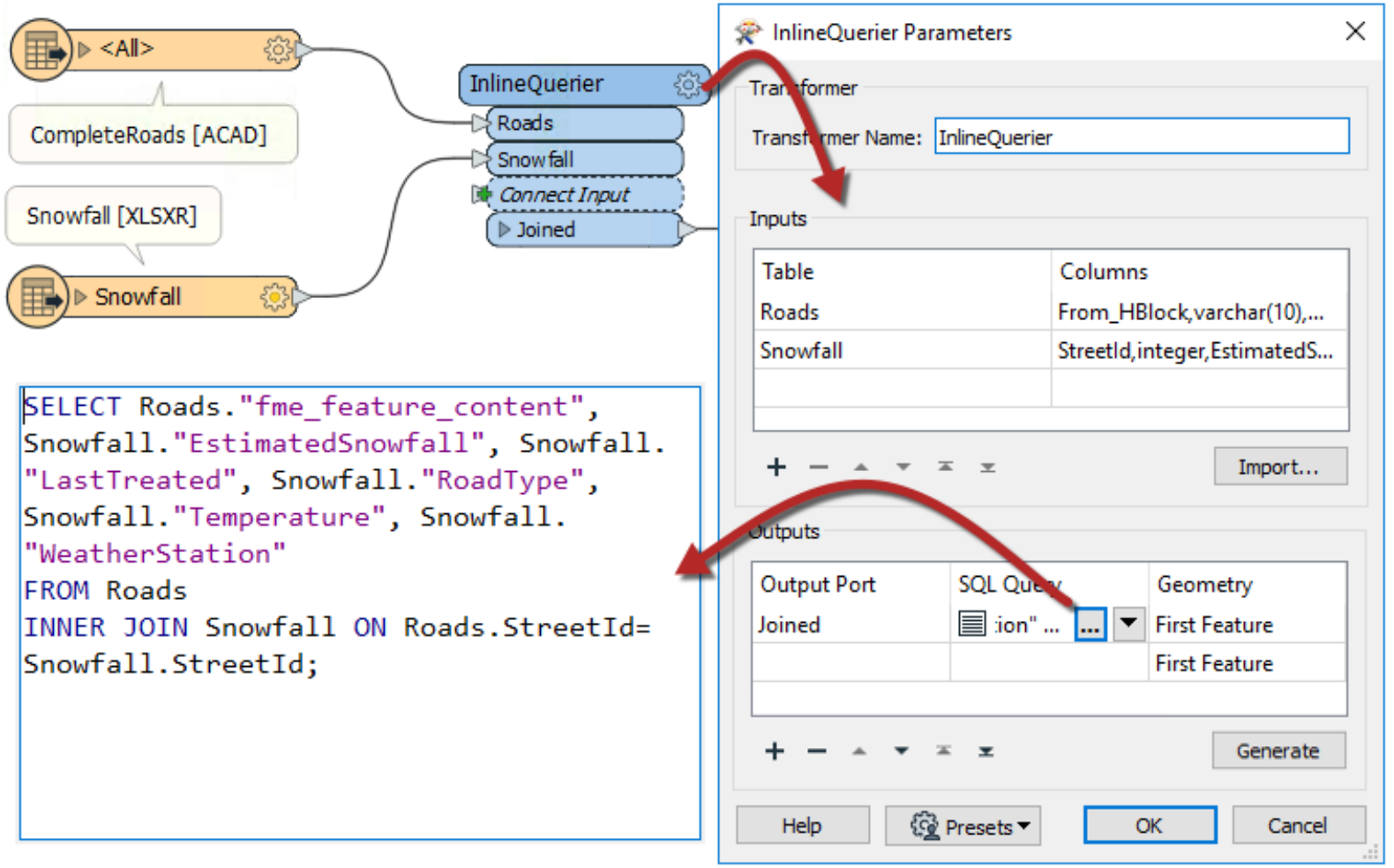






The InlineQuerier Transformer

- Generates a temporary database – allowing SQL commands



Resources

Transformer Gallery

Community pages and forums

Knowledge Base

Each Other


Thanks for watching!
Any questions?

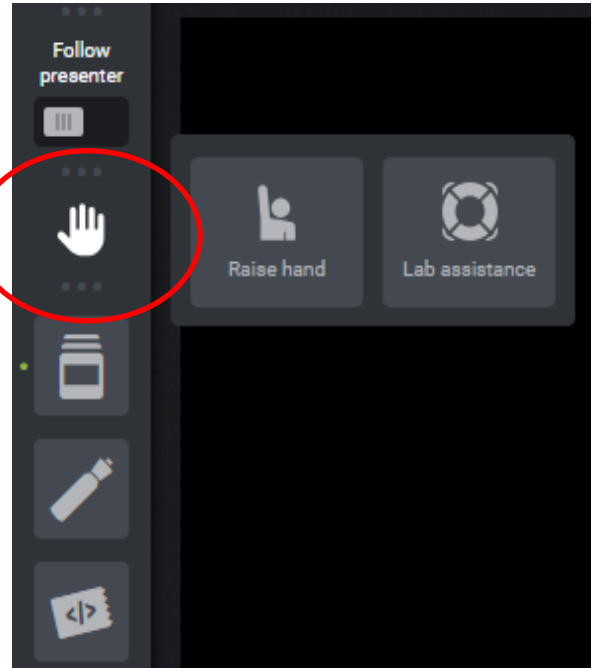
FME Form Training - Module 7

Advanced Workflow Design

Training Environment

< keep '**Follow presenter**' on

< **Lab**



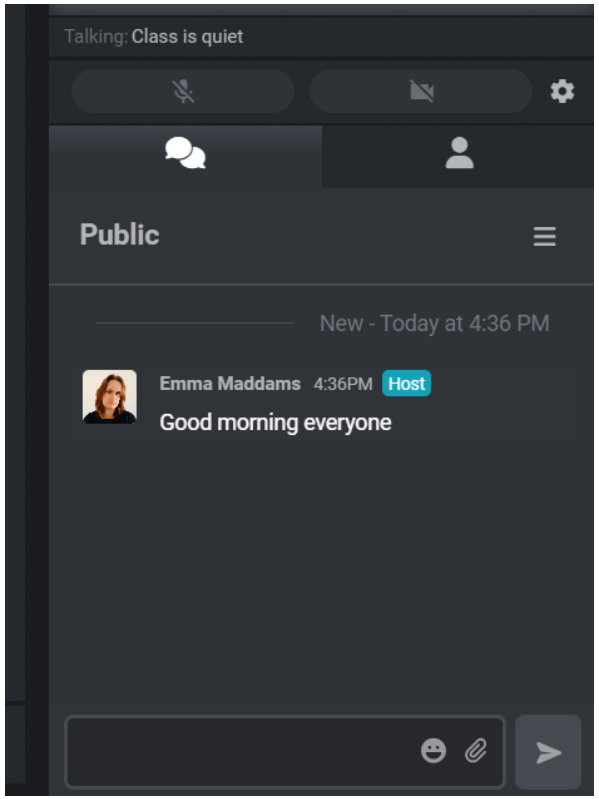
need help when using your Lab

Use either:

- 'Raise hand' or
- 'Lab Assistance'

Chat

- to everyone
- to trainer



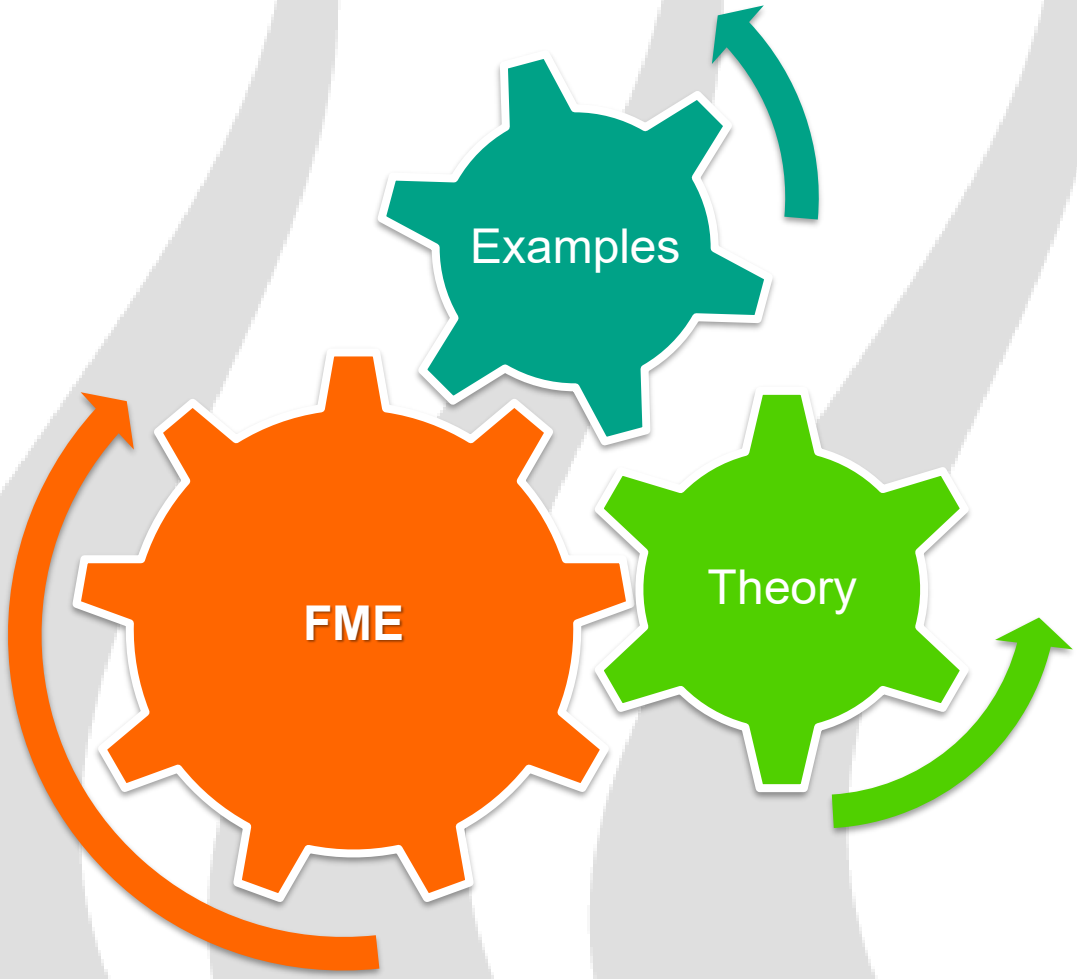
Training Environment

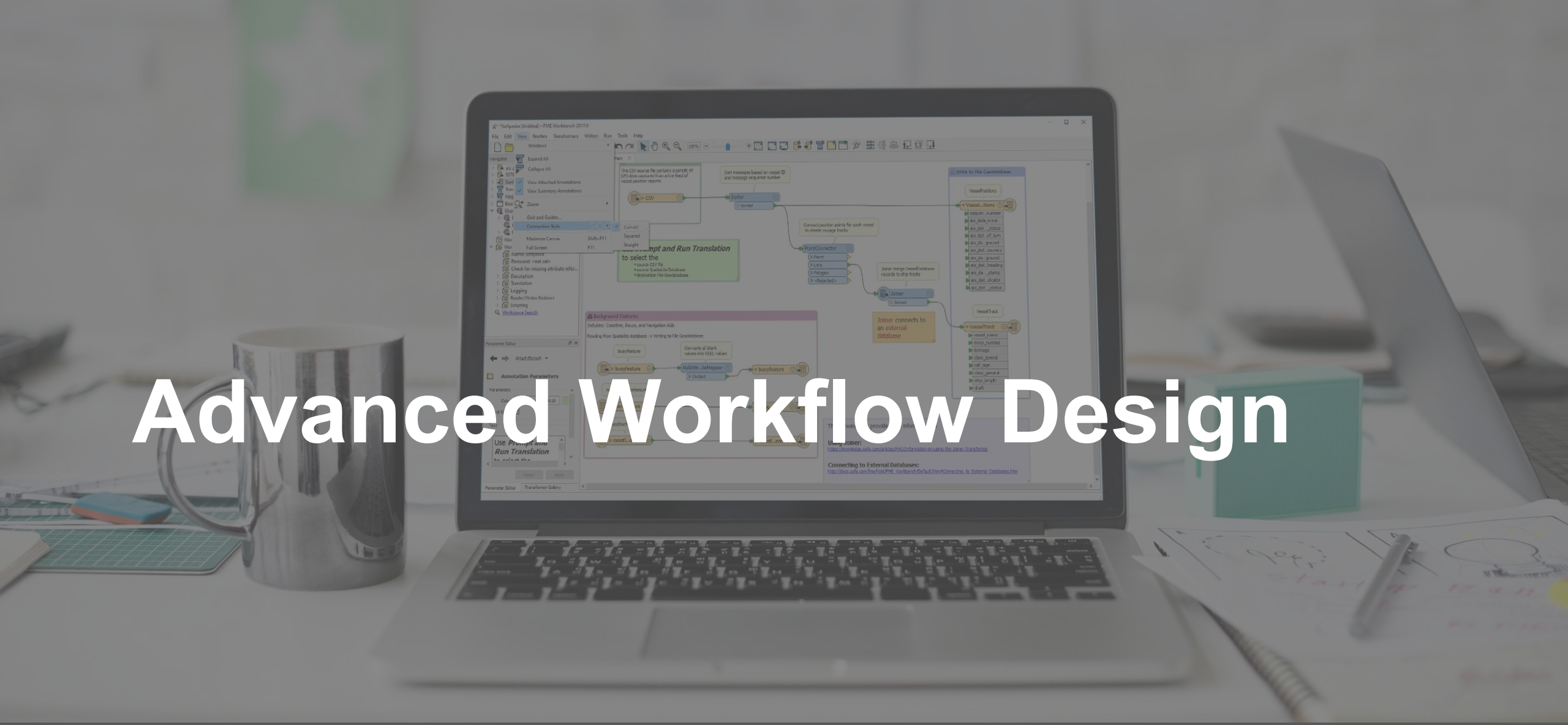
- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

What we'll be covering today…

Agenda

- Parameters
  - Parameter Types
  - Creating & Managing Parameters
  - Published Parameters
- Generic Reader/Writer
- Dynamic Translations
- Processing Bulk Data
  - Batch Processing with Workspace Runner





Advanced Workflow Design

Connect. Transform. **Automate**

What techniques are we going to cover

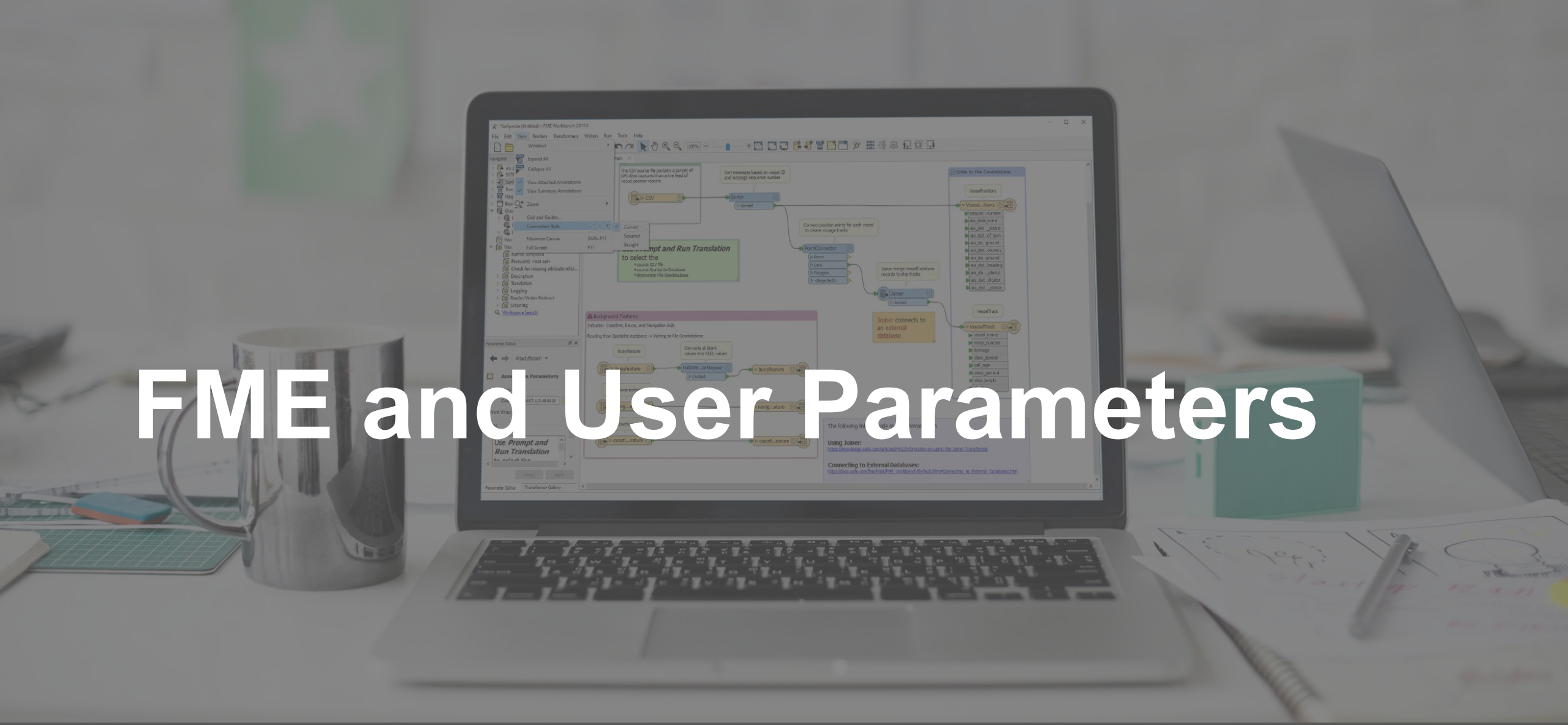
Build reusable, versatile self-serve workflows:

- Utilise **Published Parameters** to directly manage the parameters in your workspace and to create end-user controlled workspaces. Ideal for creating user friendly workspaces that can be shared with FME Desktop users or deployed to FME Server and Server Apps

- Create dynamic workspaces that can handle data in any format and a changing schema.
  - Use a **Generic Reader/Writer** to allow FME workspaces to be free from format restrictions.
  - **Dynamic translations** - how to create a workspace that is truly flexible
  - Using an **external schema source** to control the structure of the output data

Process Bulk Data:

- **Batch processing** multiple input data sources with **WorkspaceRunner**.
- Activating **Parallel processing** to improve performance by running multiple actions at once as a set of separate processes



FME and User Parameters

Connect. Transform. **Automate**

Parameters

Parameters control how FME operates: how data is read in, transformed and written out using various FME components

Workspace parameters are the key to giving the end-user control over self-serve workspaces.

Parameter Types

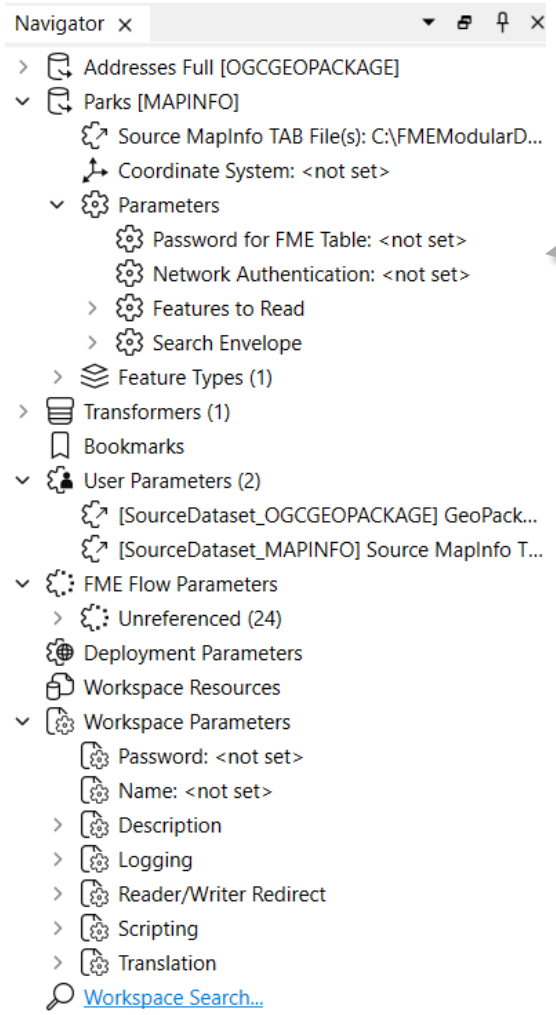
There are two types of FME users:

- **Workspace Authors** – people who design and create a workspace.
- **Workspace Users** – people who use/run a workspace, might have little knowledge of FME.

In light of these two roles, we can say there are two different types of parameter:

- **FME Parameters** – for Authors to use
- **User Parameters** – for FME Users to use

Parameter Types



FME Parameters – for Authors

Users are not expected to set these

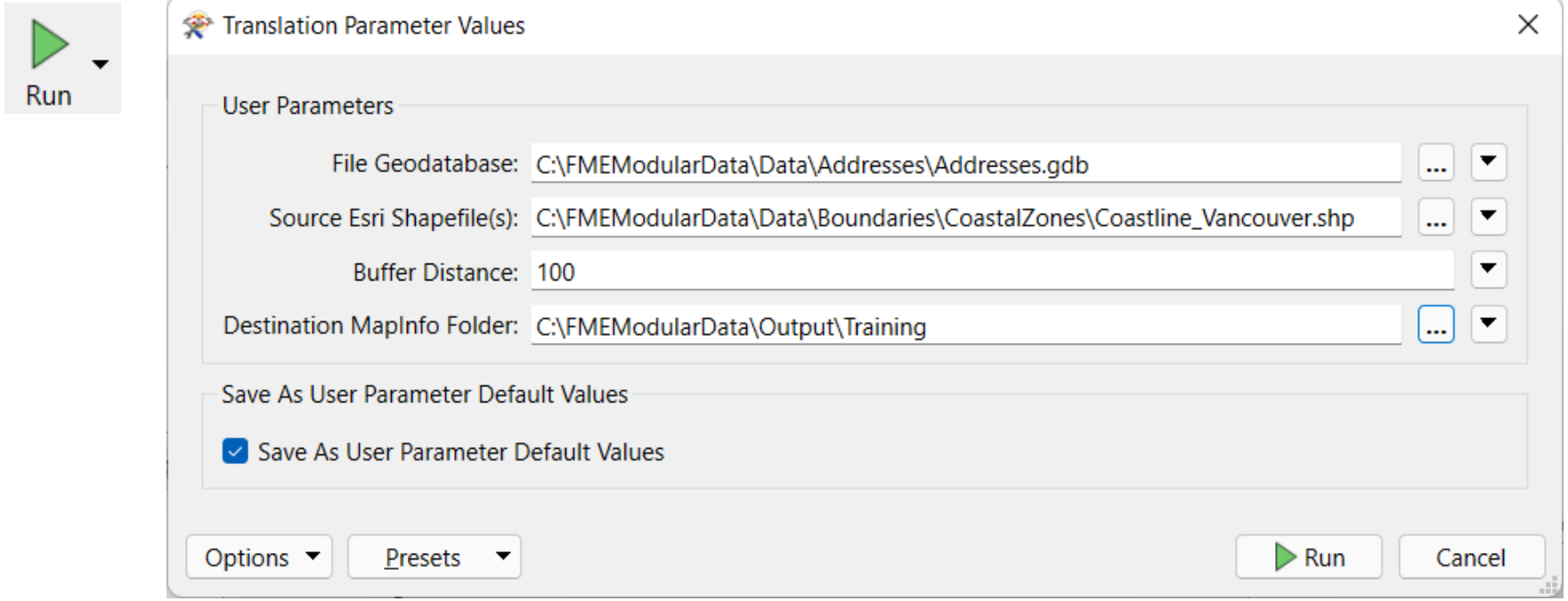
User Parameters – for FME Users

These are created by the Author, but for the User to set when running the workspace

Parameter Types

Published User Parameters

- appear in the Translation Parameter Values **Prompt** dialog when a user **Runs a workspace**



- also utilised when authoring workspaces for use in **FME Server**. Where they are especially important for self-serve workspaces and Server Apps

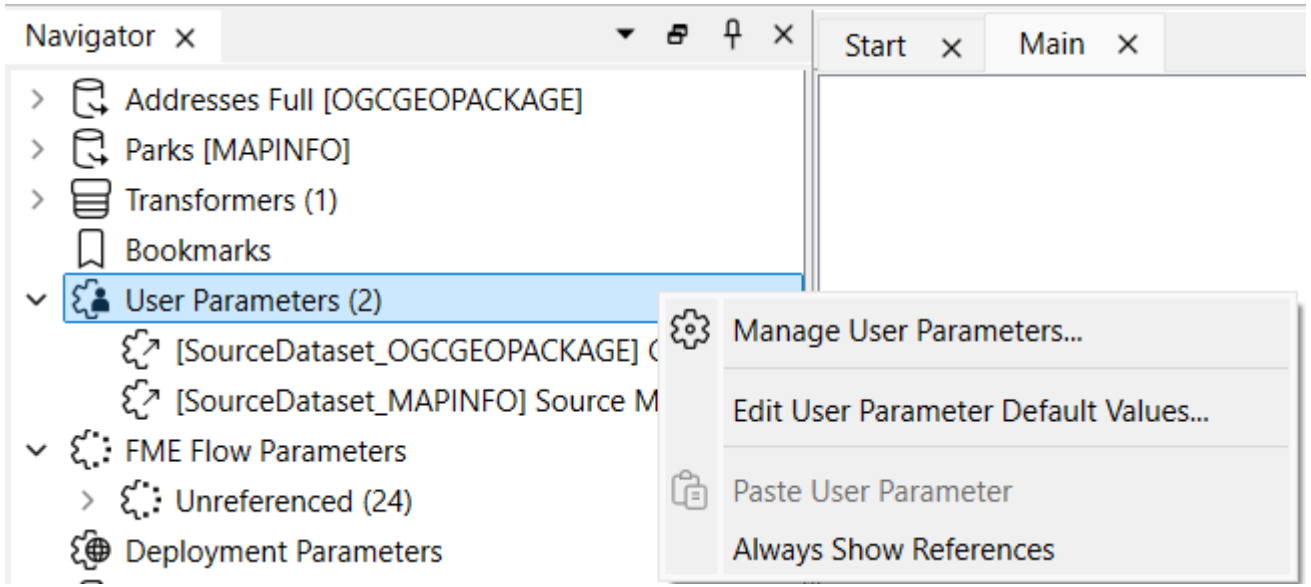
Managing Parameters

Connect. Transform. **Automate**

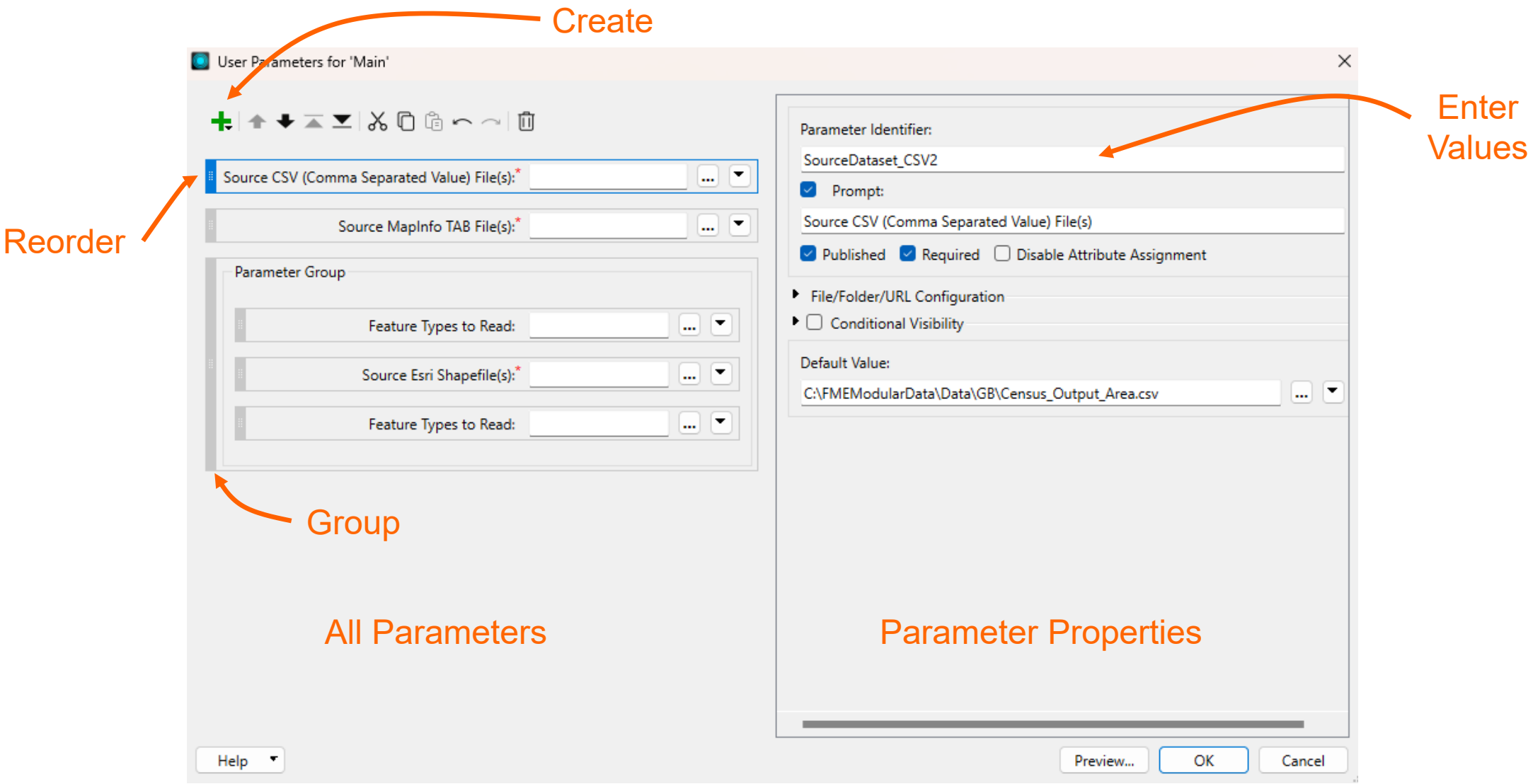
Manage Parameters

- You can add, configure, reorder, and group parameters in the **Parameter Manager** dialog.

- You can access the Parameter Manager by right-clicking User Parameters in the Navigator and choosing Manage User Parameters...

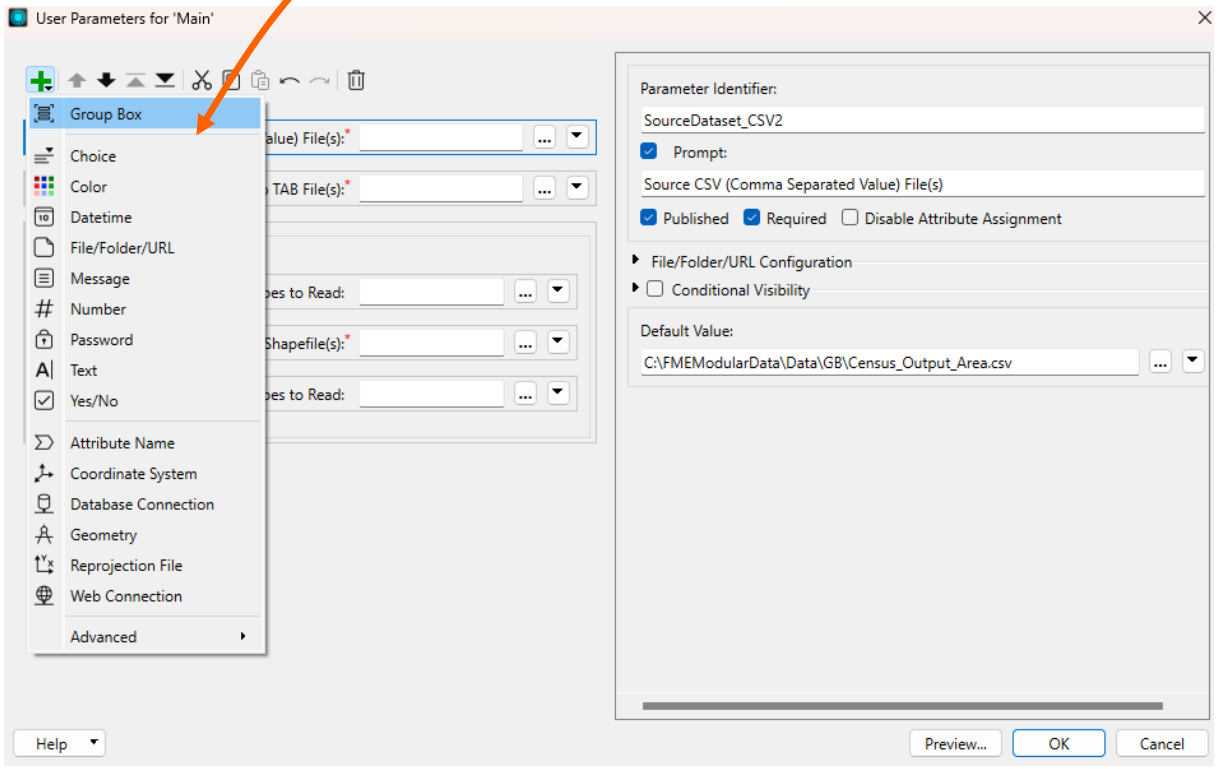


Parameter Manager Overview



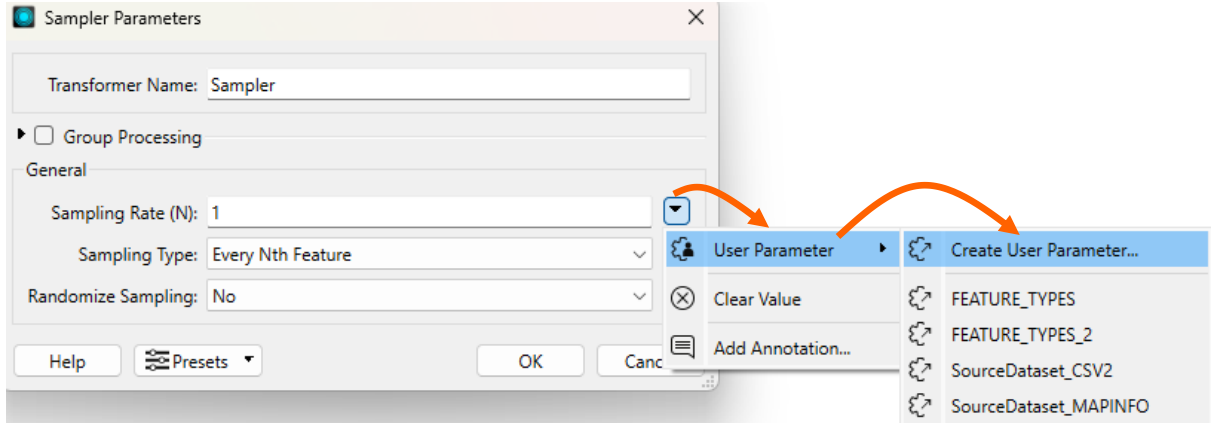
Creating User Parameters

- Within the **Parameter Manager**



- Within **Transformers**

Transformer parameters can also be published directly in transformer dialogs:



Choose the Correct Parameter Type

There are many types of parameters, mainly related to the data type required. As a workspace author, you must choose the appropriate parameter based on the following questions:

- What choice do you want the end-user to be presented with?
- What data type is required to supply the correct value to an FME parameter?

Choose the Correct Parameter Type

Choice	Choose one or more values
Color	Select a colour value
Datetime	Enter a date and time
File/URL	Choose the names and paths of existing files or folders, an output file or folder, or a URL
Message	Display a message at runtime – *useful for adding display text (instructions or info) for users running workspaces on FME Server*
Number	Enter an integer or floating-point number. The number can be bounded or unbounded based on the specified Configuration
Password	Enter a password
Text	Enter a text string

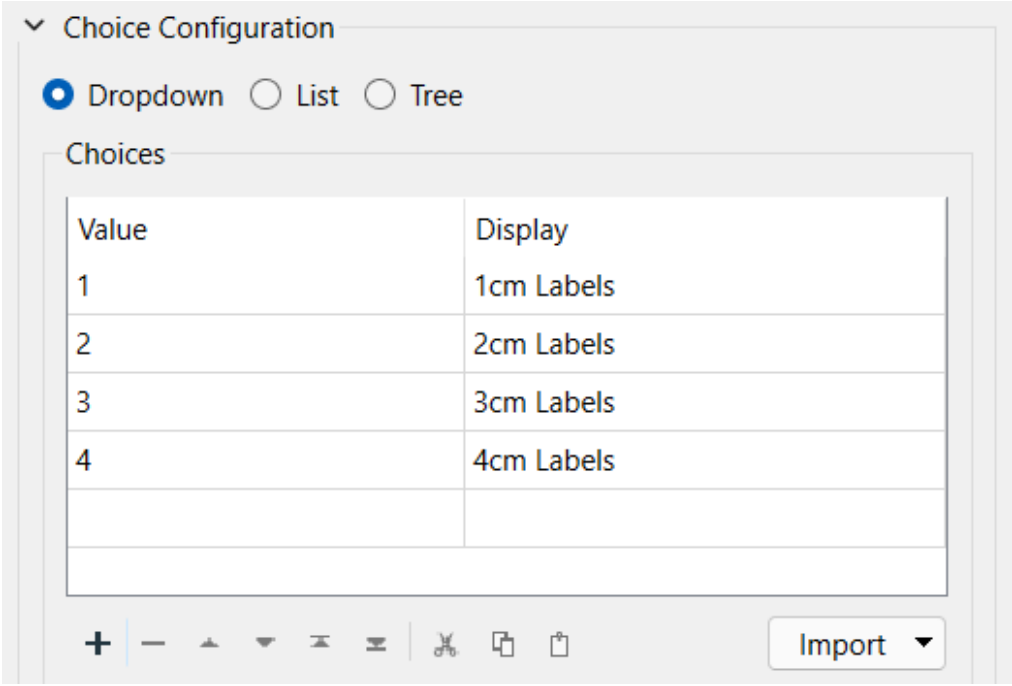
Yes/No	Select or unselect a checkbox
Attribute Name	Choose feature type attributes or items from a comma-delimited or space-delimited attribute list
Coordinate System	Choose a coordinate system
Database Connection	Choose a database connection
Geometry	Enter a geometry as a set of spatial coordinates in GeoJSON
Reprojection File	Choose a grid shift file for reprojecting data
Web Connection	Choose a web connection
Scripted	Write a Python or Tcl script that assigns the value of a parameter to the workspace at runtime

Parameter Type ‘Choice’

A notable parameter type for a **self-serve** setup is **Choice**

It lets you present the user with several different kinds of choices, including text, numbers, or a set of display values with a different value returned to FME based on a lookup table.

The user is presented with the list of (more intuitive) options on the right, but the value returned to FME will be the (more useful) number on the left.



On FME Server, this parameter type is very useful for Data Download tasks. That's because many of the values we want to pass to FME (e.g. format or coordinate system names) - have an abbreviated format that wouldn't make sense if presented to the user. A Choice parameter allows us to provide a user-friendly list of options that hides a more complex response to the server.

Parameter Options

While the exact configuration varies by parameter type, the shared options include:

- **Parameter Identifier**: a short, unique name for the parameter. This name is used to reference the parameter in values and menus. e.g. *$(myIdentifier)*
- **Prompt**: the prompt for the parameter that appears when an end-user runs the workspace on Desktop or Server. *use audience appropriate language, not geek speak!*
- **Published**: if checked, the parameter is created as a published parameter. If unchecked, the parameter is created as a private parameter.
- **Required**: if checked, the workspace does not run unless the user inputs a value. If unchecked, an input value is not required for this parameter.
- **Disable Attribute Assignment**: This setting controls whether the end-user can select an attribute to provide values to a user parameter.
- **Default Value**: choose the default value that will be used.

Exercise 7.1

Using Parameters

- The team responsible for maintaining parks has a workspace that translates their data from MapInfo TAB format to Google KML. It also writes a file of XML metadata to show who translated the data and when.
- However, the team are not experienced FME users and need to be able to easily enter any required values for each translation.

Data - Parks (MapInfo TAB)

Starter Workspace - C:\FMEModularData\Workspaces\7.01-AdvancedWorkflow-UseOfParameters-Begin.fmw

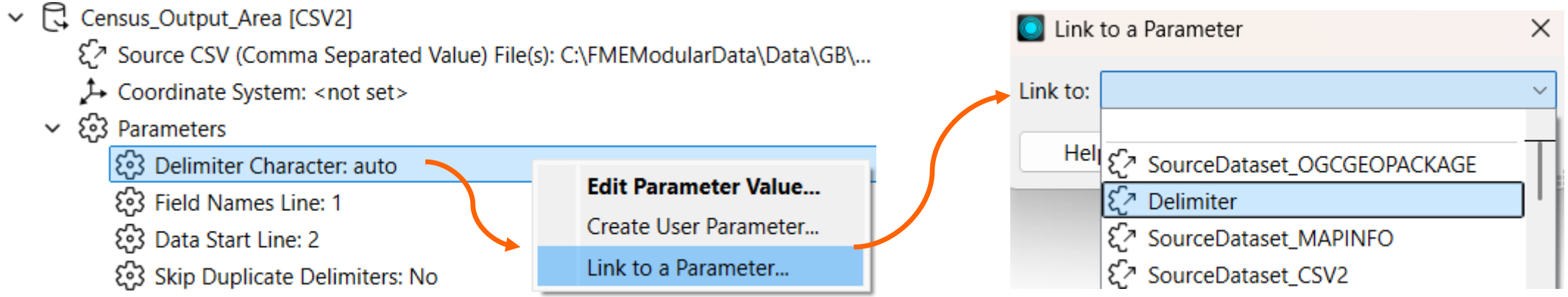
Goal - Create a workspace where end-users are able to input metadata values for use within the workflow at the time of running the workspace

Linking User Parameters

Sometimes a workspace author needs to link as user parameter to elsewhere in the workspace.

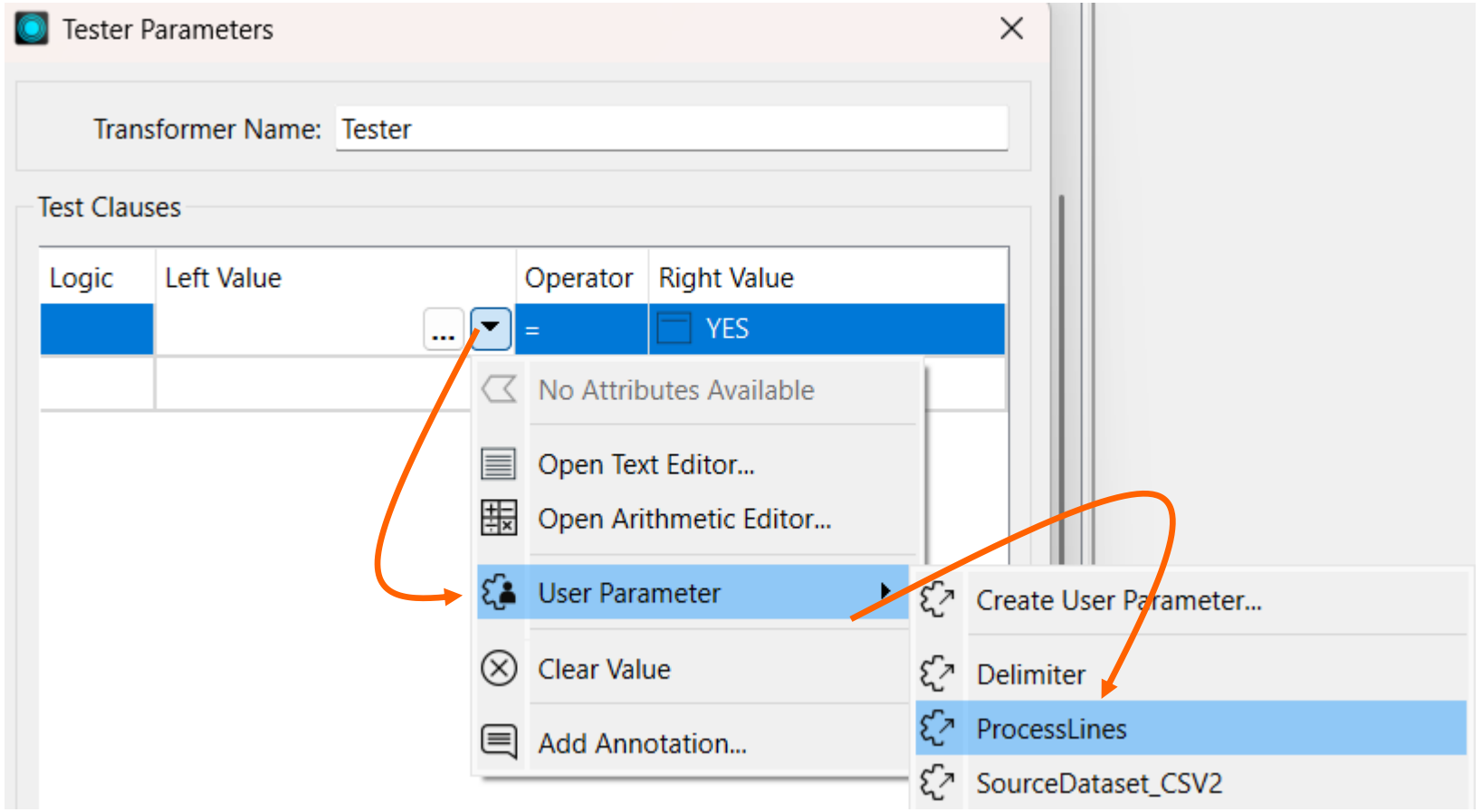
Right-click it and choose **User Parameter > Link to User Parameter**.

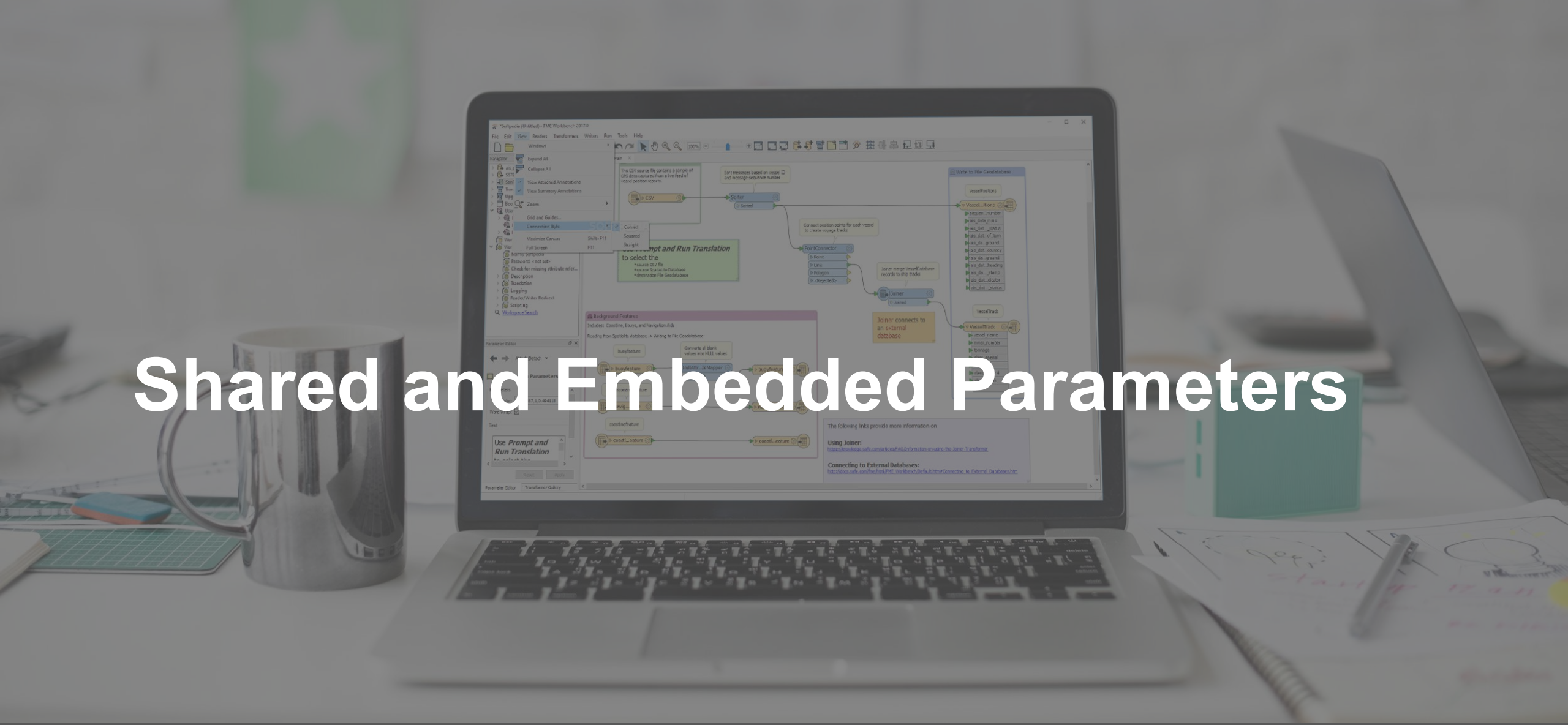
Here we are linking the Delimiter Character parameter of a CSV reader to a Text user parameter named *Delimiter*



Linking User Parameters

You can also link user parameters **directly in transformer** dialogs using the drop-down arrow and choosing User Parameter:





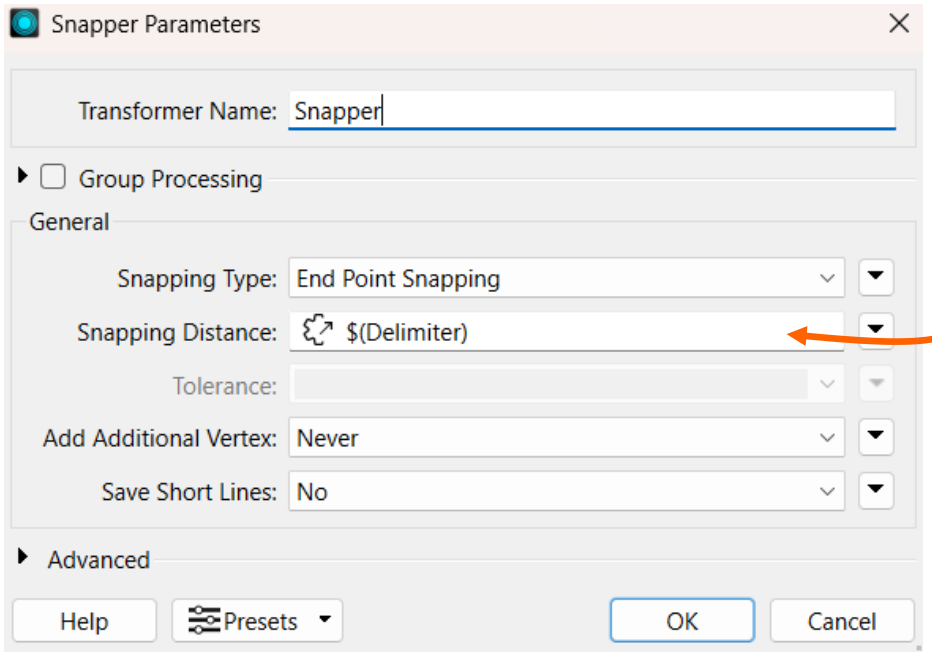
Shared and Embedded Parameters

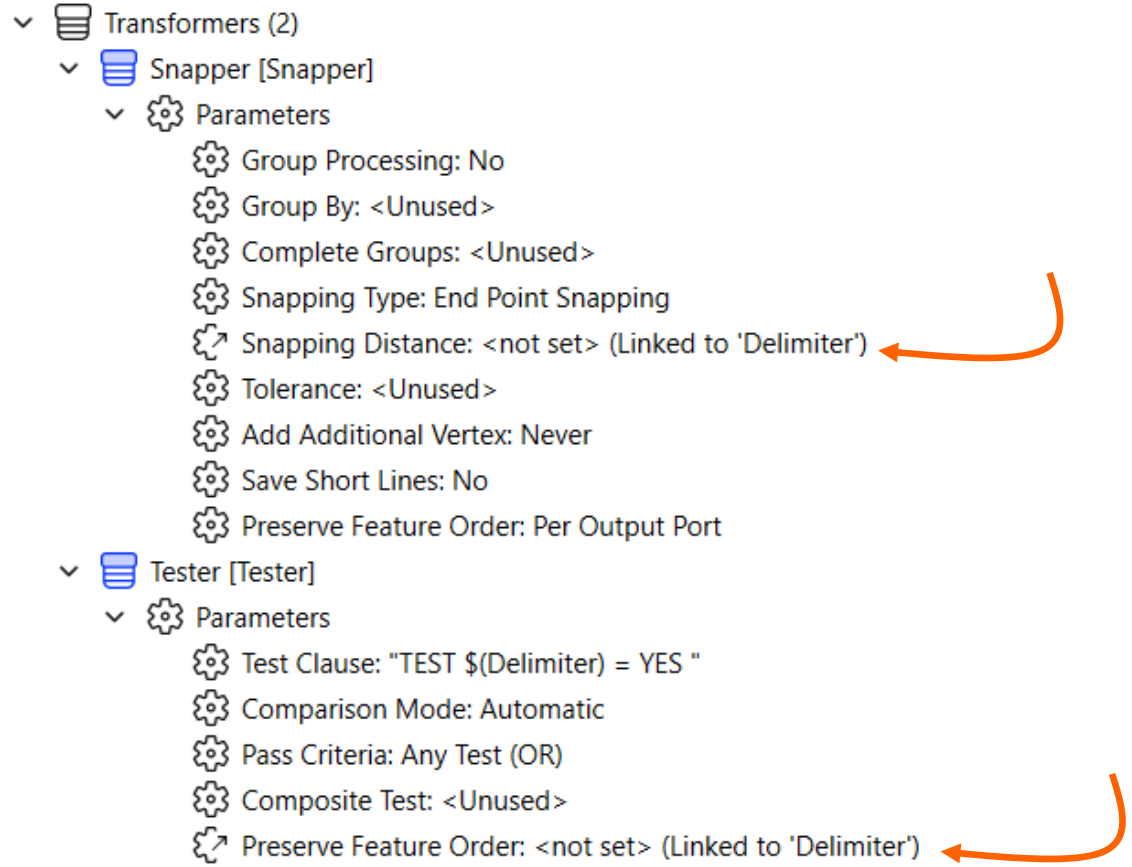
Connect. Transform. **Automate**

Shared Parameters

There is no limit on the number of times a User parameter can be linked to an FME parameter.

The **value** obtained from the User parameter **can be used as many times as is required.**

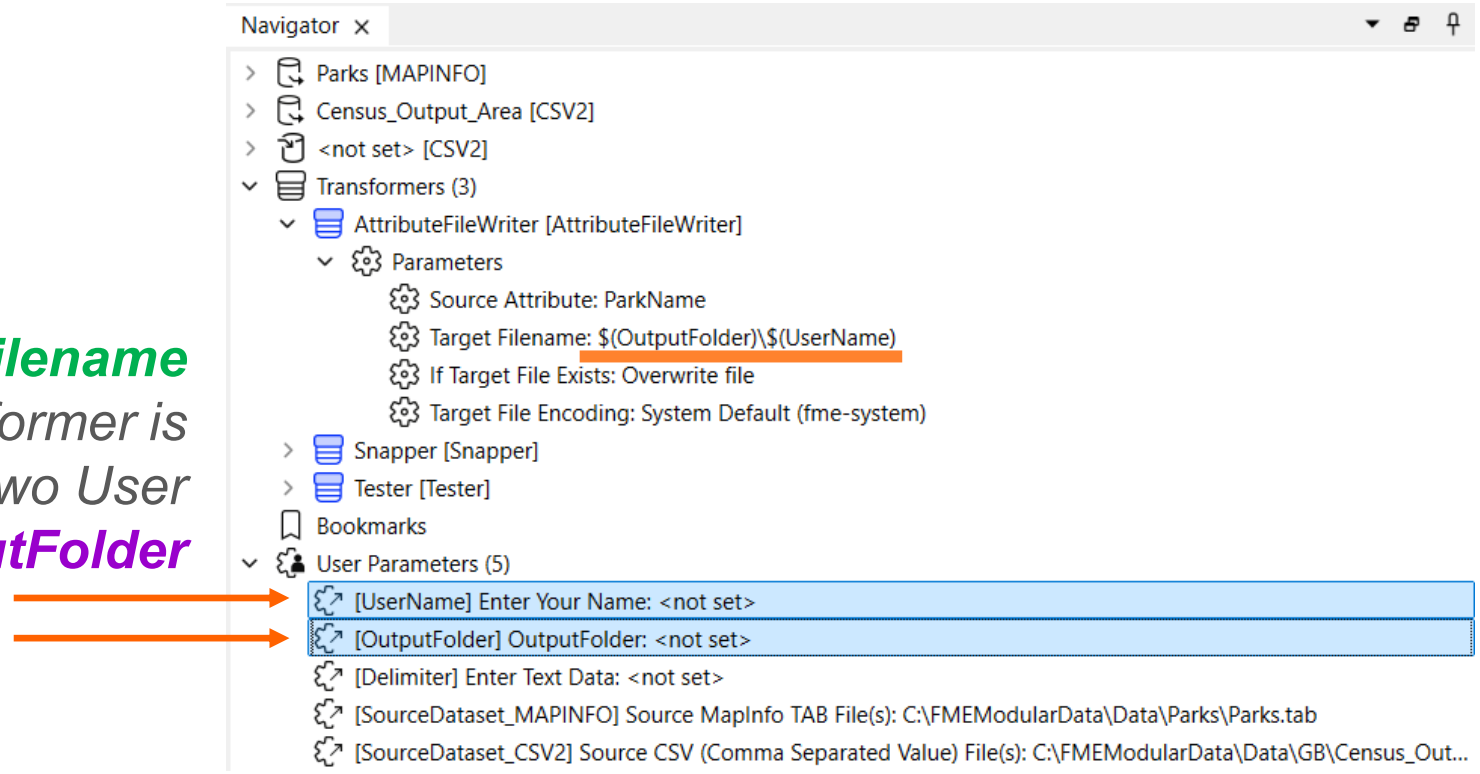


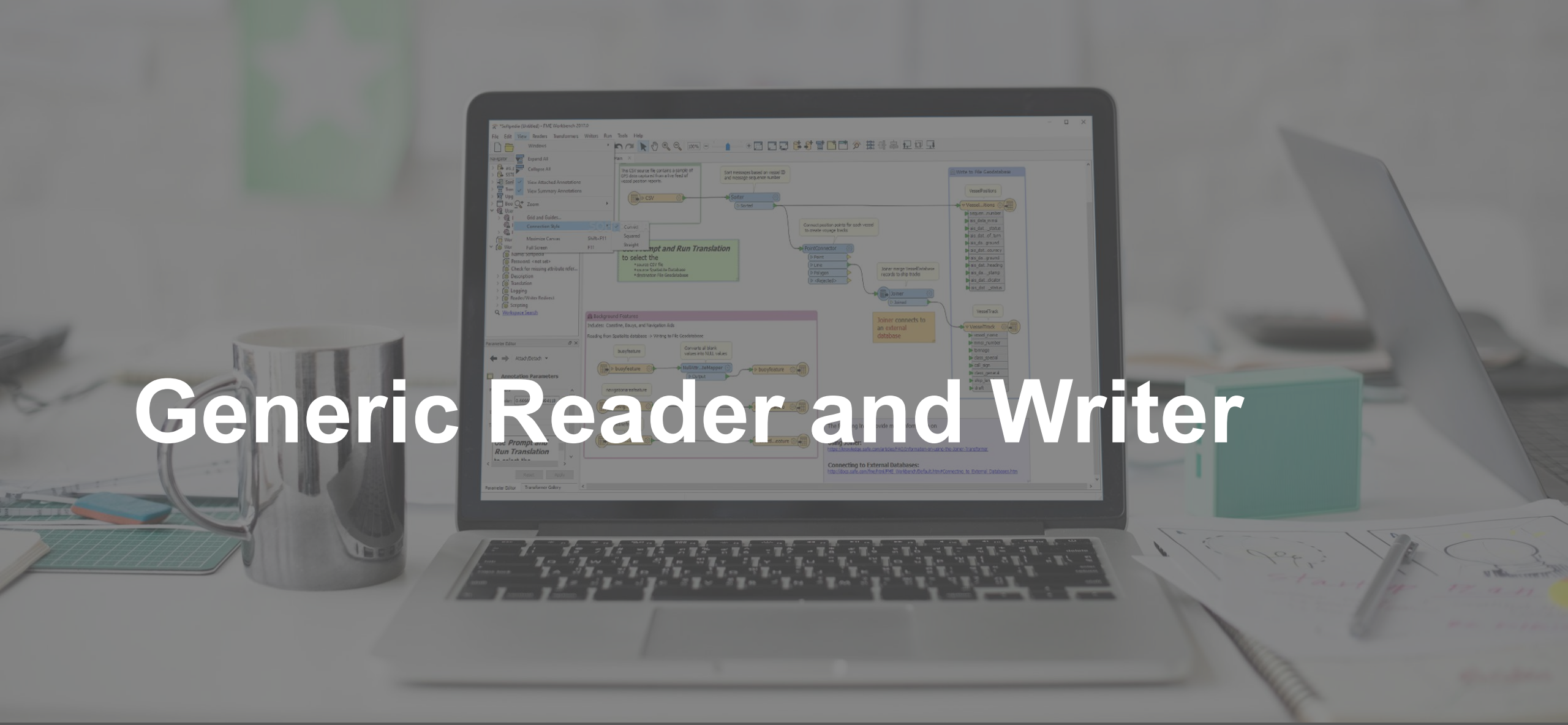


Embedded Parameters

When **a parameter value is constructed from multiple User parameters**, we call it **Embedded parameter**.

*The FME parameter called **Target Filename** on this AttributeFileWriter transformer is constructing a value by using two User parameters: **UserName** and **OutputFolder***





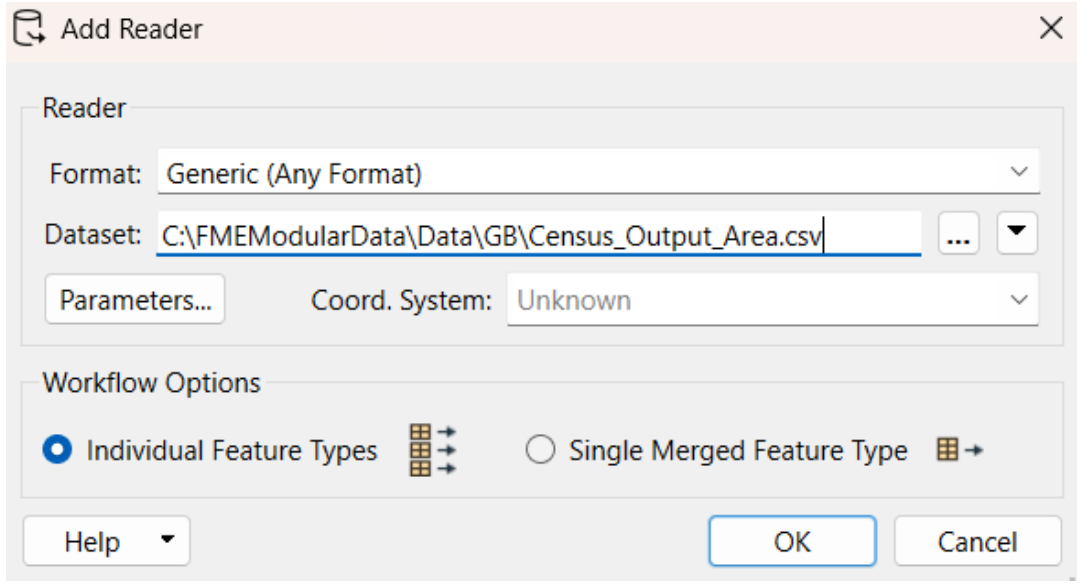
Generic Reader and Writer

Connect. Transform. **Automate**

Generic Reader & Writer

The Generic Reader and Writer allow FME workspaces to be free from format restrictions.

They are capable of reading/writing almost any format because they are not tied to a specific one.

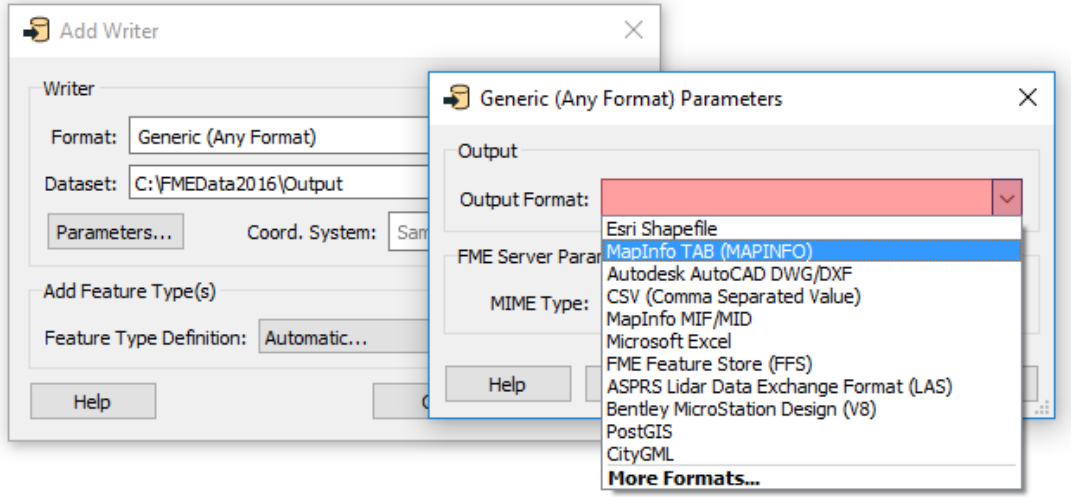


Note!

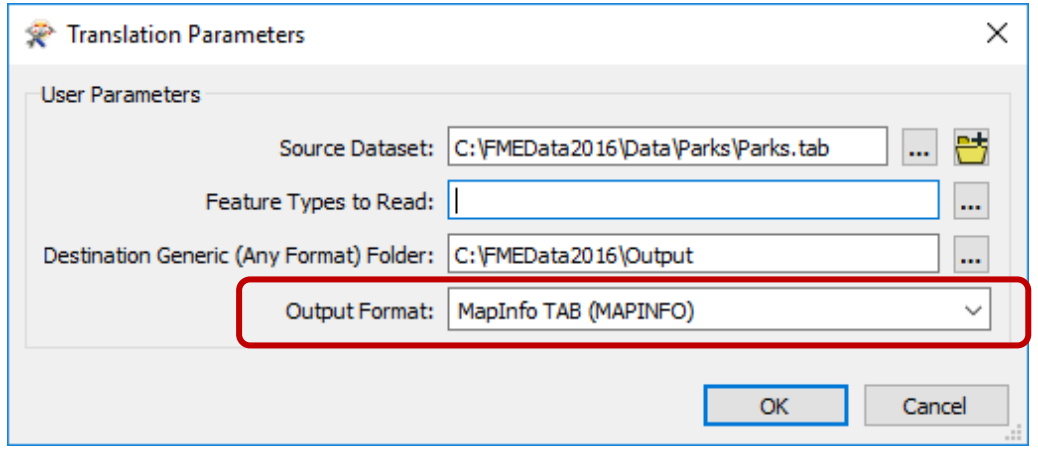
- It only works with file-based formats (like GIS files, CAD files, MS Office documents, CSVs) and not databases or web-based formats.
- Think about your attributes/schemas on your varying input – remember that Reader/Writer is not only focusing on file format – the structure of a file is also part of the reading/writing!

The Generic Writer

- When setting up a **Generic Writer**, a format has to be chosen – *this is just to initially add the writer*

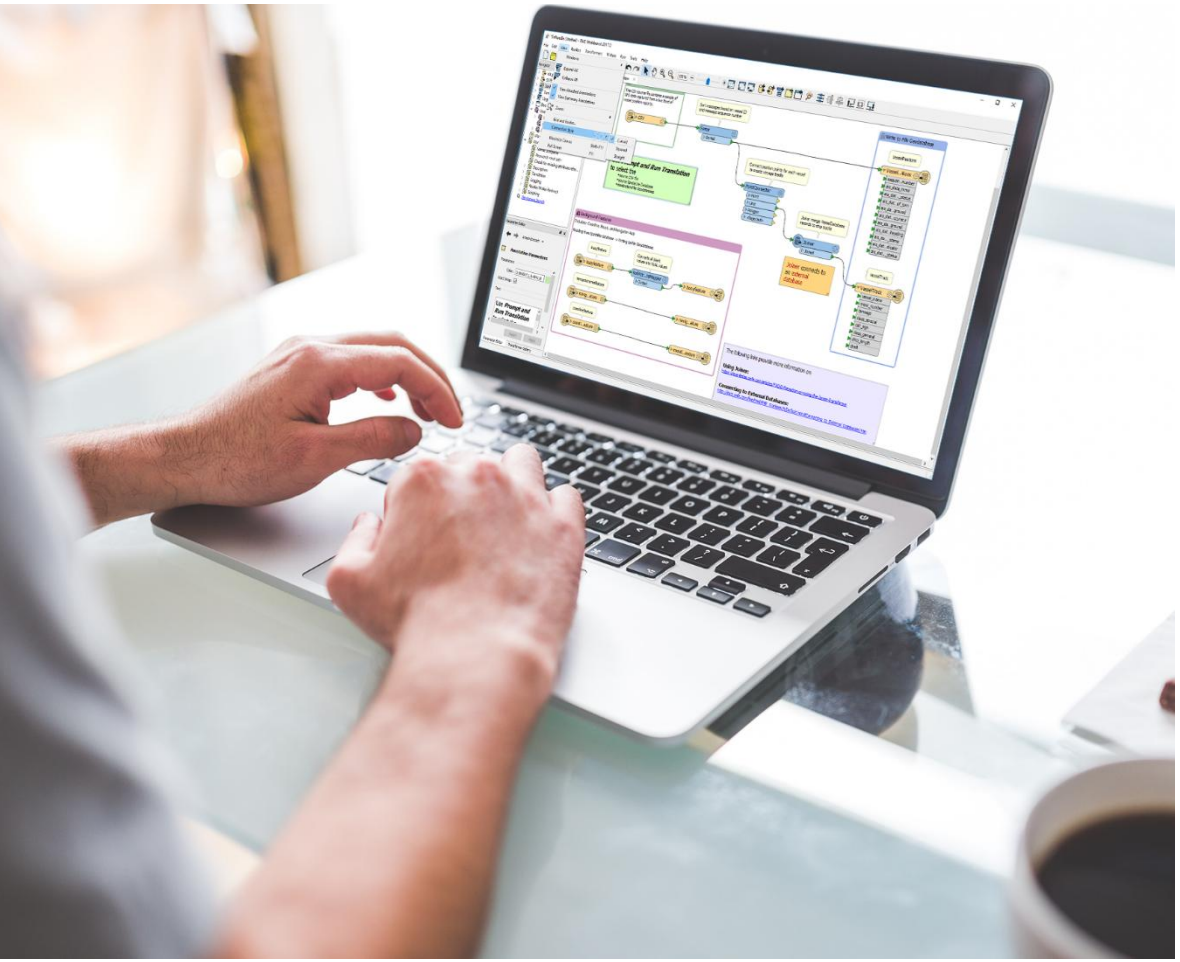


- Each time the workspace is run, the user will be prompted to choose the output format required, using a new User Parameter



Note: the Feature Type definition and Attribute Schema are defined within the workspace (like normal) – you'll see why this is important later!

Exercise 7.2



Generic Writer - plus more Parameters fun

- As resident FME expert you are often asked to translate data (particularly the community map) 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.

Data - Community Mapping (Esri File Geodatabase)

Starting Workspace - none

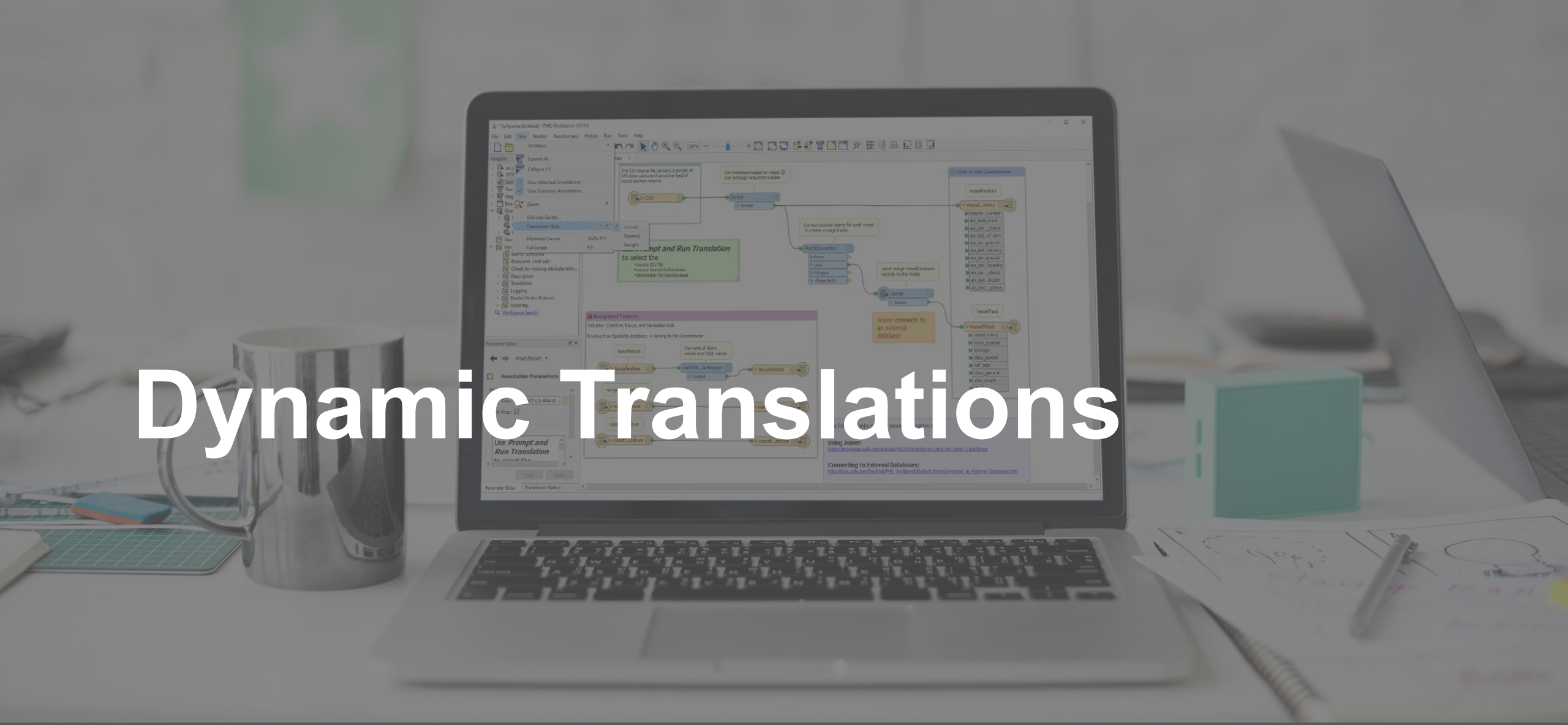
Goal - Create a workspace to translate Community Mapping data to a format of the end-user's choice and zip it

Generic Reader/Writer

In Exercise 7.2, Generic Writer **->** Each dataset being written ended up with the same attribute schema!

The Generic Reader and Writer allow FME workspaces to be free from format restrictions (file-based formats) *– which is really useful, especially when combined with User Parameters*

However, it would be even more useful if a workspace was capable of dynamically handling schema *– i.e. using the schema of the source dataset when writing out*



Dynamic Translations

Connect. Transform. **Automate**

Dynamic Translations

Dynamic Translations are a way to create “**schema-less**” **workspaces**

A Dynamic Translation does not reflect either the source or destination schema. It’s a universal layout that is designed to handle data regardless of the schema used by the source data.

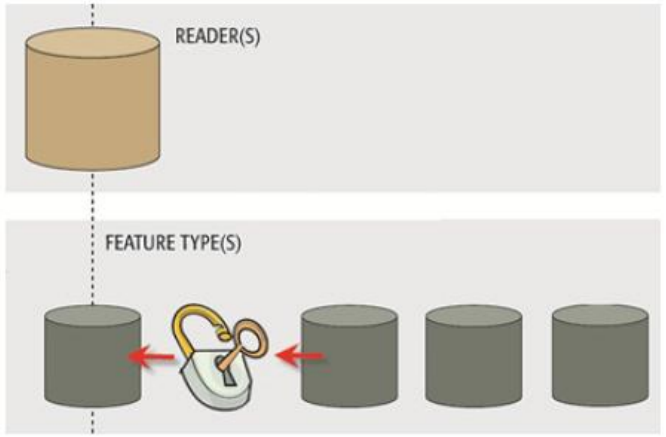
Keep in mind, **schema** to be a trinity of:

1. Feature types
2. Attributes
3. Geometry type

Dynamic Translations

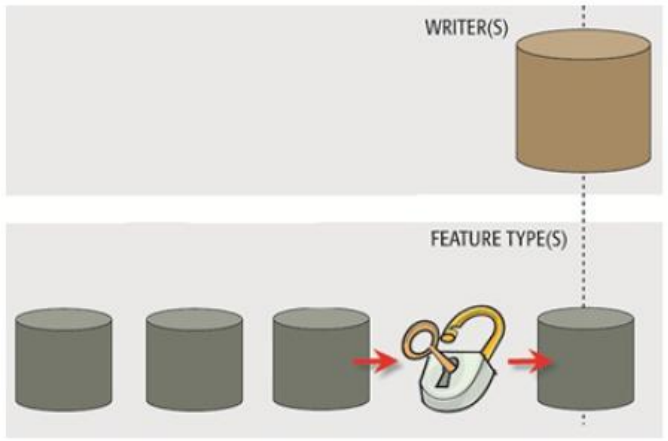
Dynamic Readers

- All Feature Types are read by the workspace, regardless of whether they are yet defined.
- Data is read regardless of attributes or geometry type.



Dynamic Writers

- All Feature Types are written to the destination dataset, regardless of whether they have been defined.
- All attributes and geometries are also written, regardless of whether they too have been predefined in a writer feature type.



Dynamic Translations

Dynamic Translations can be setup using two different methods:

1. Using **Generate Workspace Wizard**

2. On **Reader or Writer**

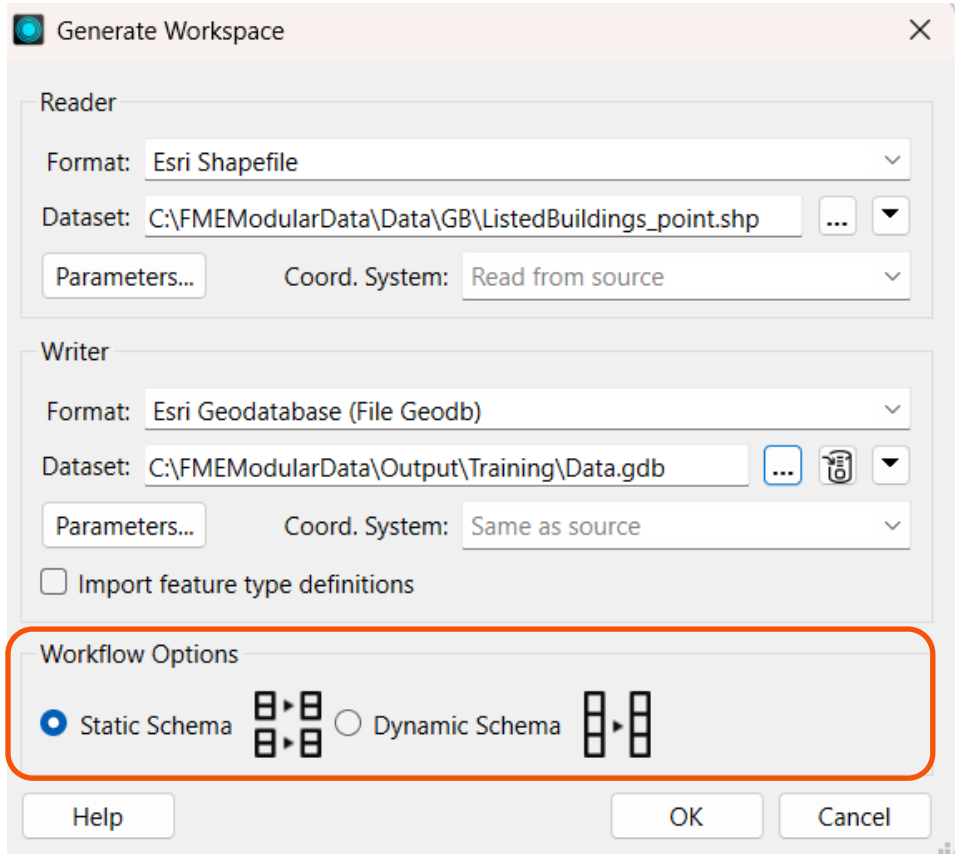
Dynamic Translations – Generate Workspace Wizard

Creating a Dynamic Translation using Generate Workspace Wizard

When creating a workspace using the ‘Generate’ wizard, there are two workflow options:

- Static Schema = schema is read from the source data. Each reader feature type will be connected to its corresponding writer feature type.

- Dynamic Schema = One merged feature type will be connected to one writer feature type. The schema is not replicated on the workspace which breaks the dependence on the source and destination schema.

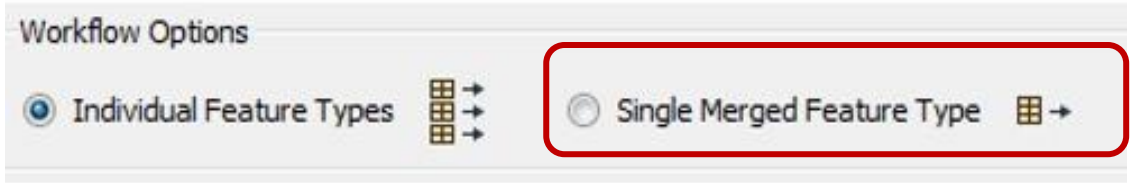


Dynamic Translations – on Reader/Writer

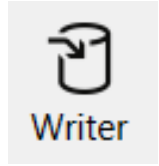
It is also possible to create a workspace where only the Readers are dynamic, or only the Writers:

Dynamic Reader Only

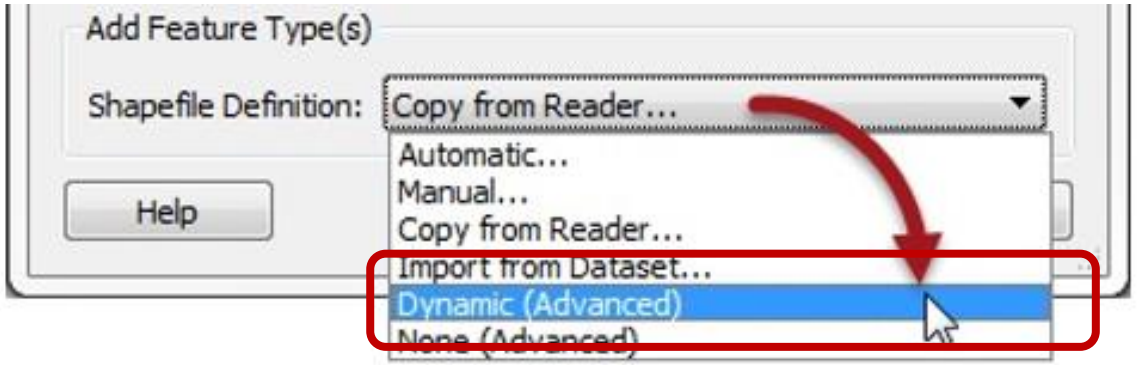
Simply use the **Single Merged Feature Type** in the ‘Workflow Options’ of the Reader.



Dynamic Writer Only

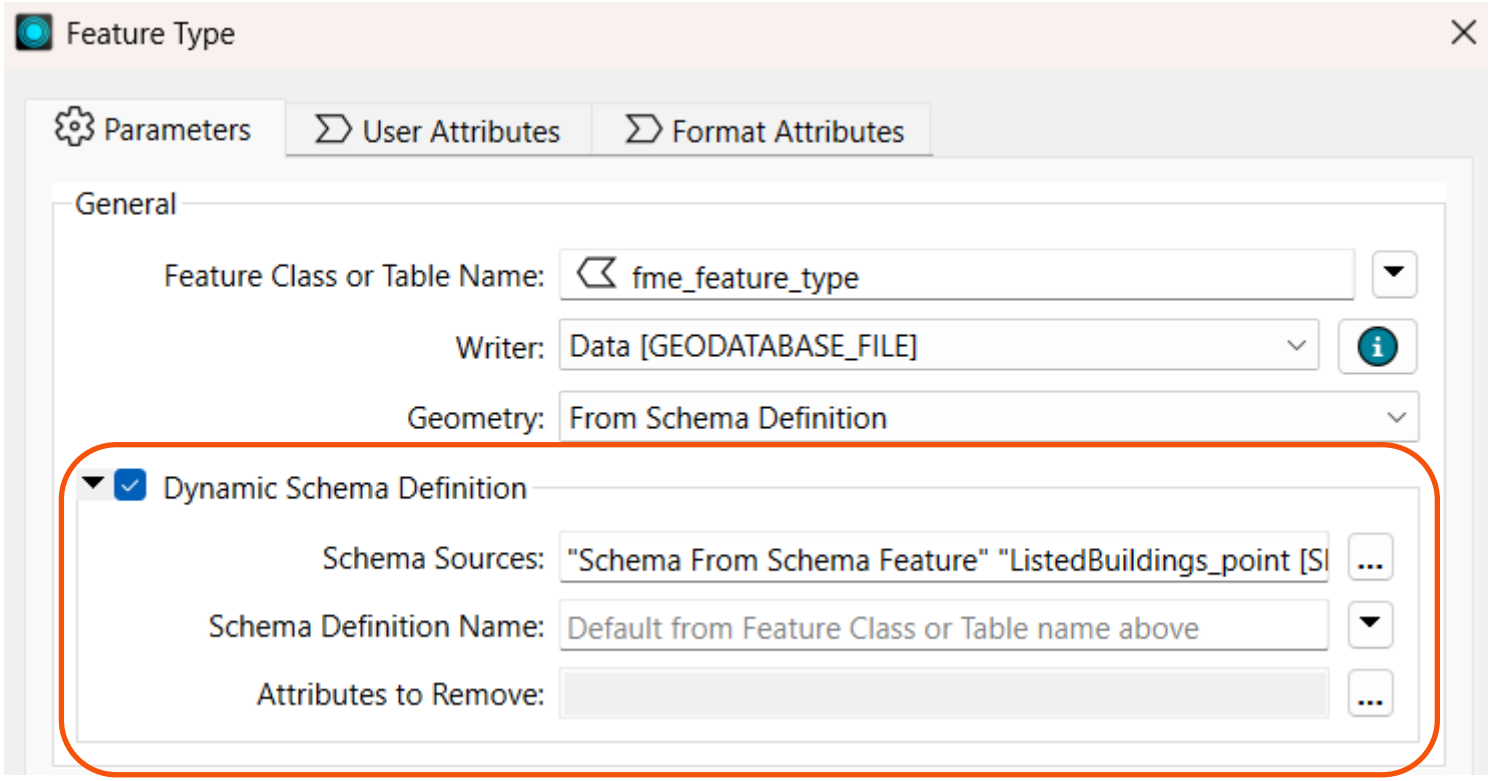


Use the **Dynamic (Advanced)** ‘File Definition’ option of the Writer.



Dynamic Schema Sources

By **default** the writer schema in a dynamic translation is defined not in the workspace, but by the source dataset.

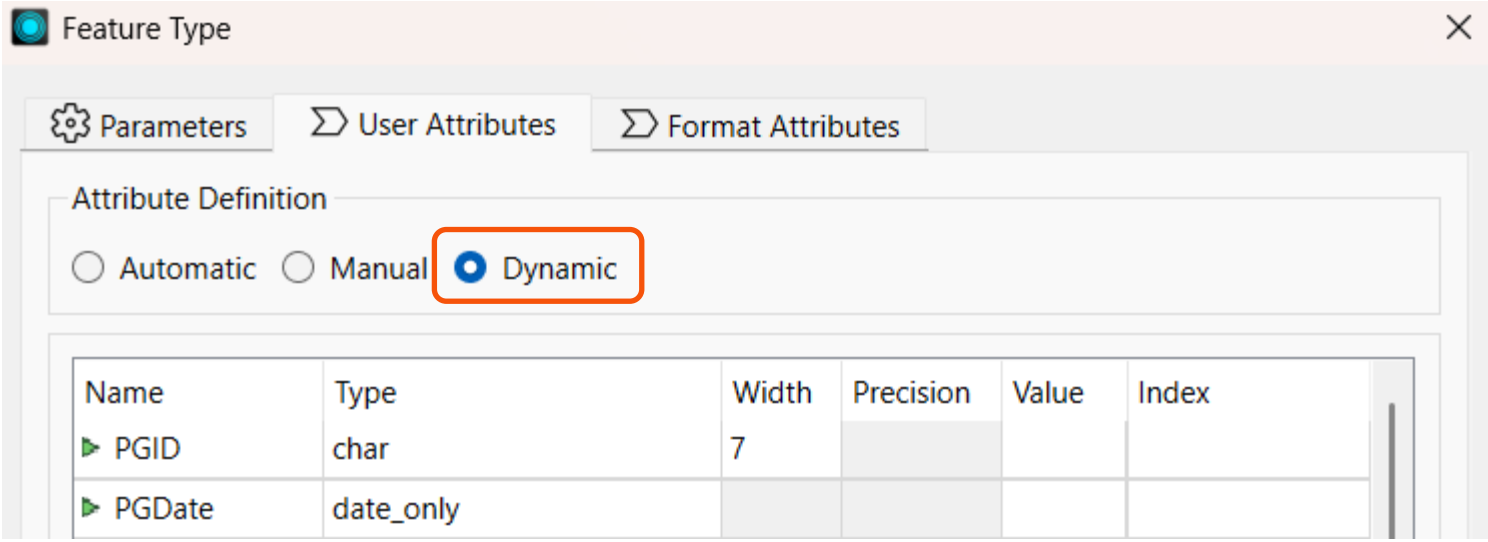


The **Schema Sources** parameter defines from where the destination schema is going to be obtained.

…we'll look at this later

Dynamic Translations – Attribute Definition

In an Dynamic translation, by default the Attribute Definition is blank, as the attributes will be defined by a source schema



Automatic vs Dynamic

- **Automatic attributes** take their definition from whatever is connected to them from the workflow.
- **Dynamic attributes** are different. If the source dataset parameter is changed, the attribute definition comes from whatever source data gets read, regardless of what is connected to it.

Exercise 7.3

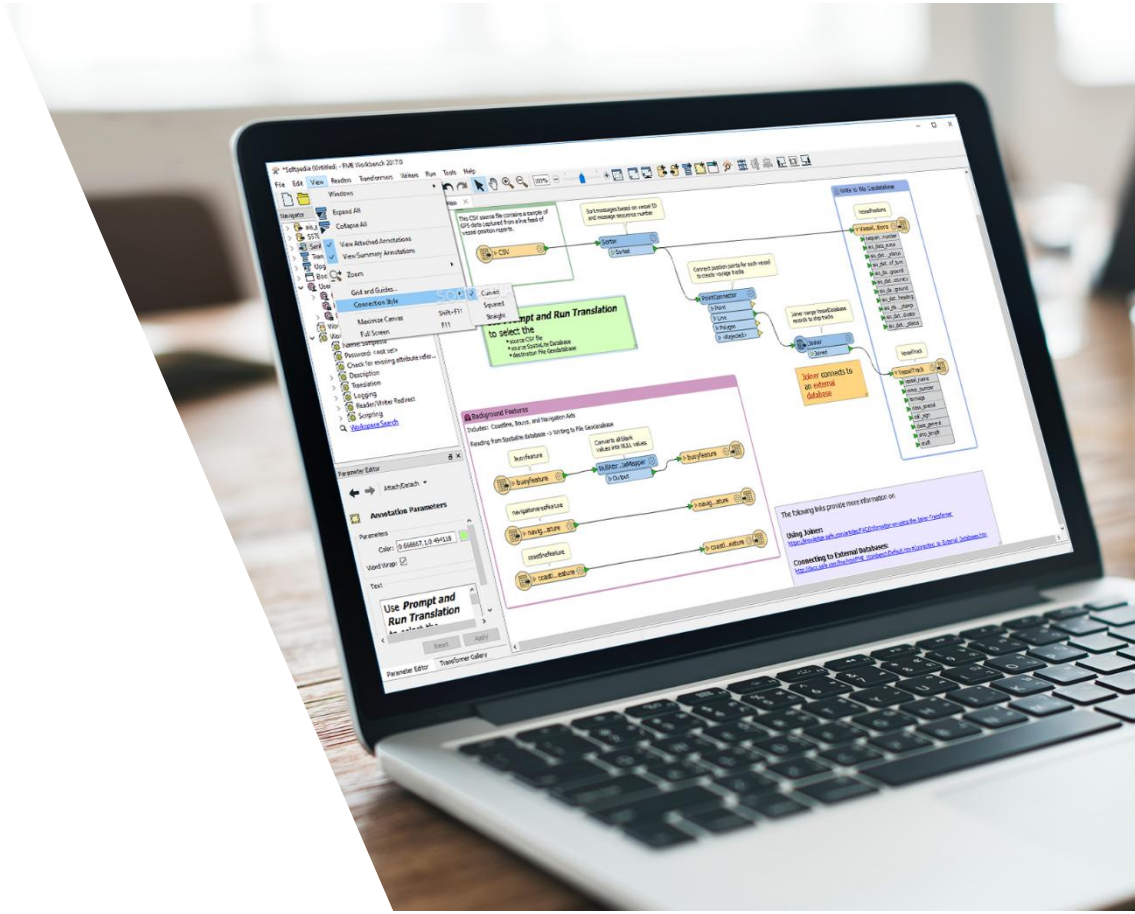
Dynamic Translations

- 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.

Data - Community Mapping (ESRI File Geodatabase)

Starting Workspace - none

Goal - Create a dynamic workspace to translate any Geodatabase dataset to a format of the end-user's choice

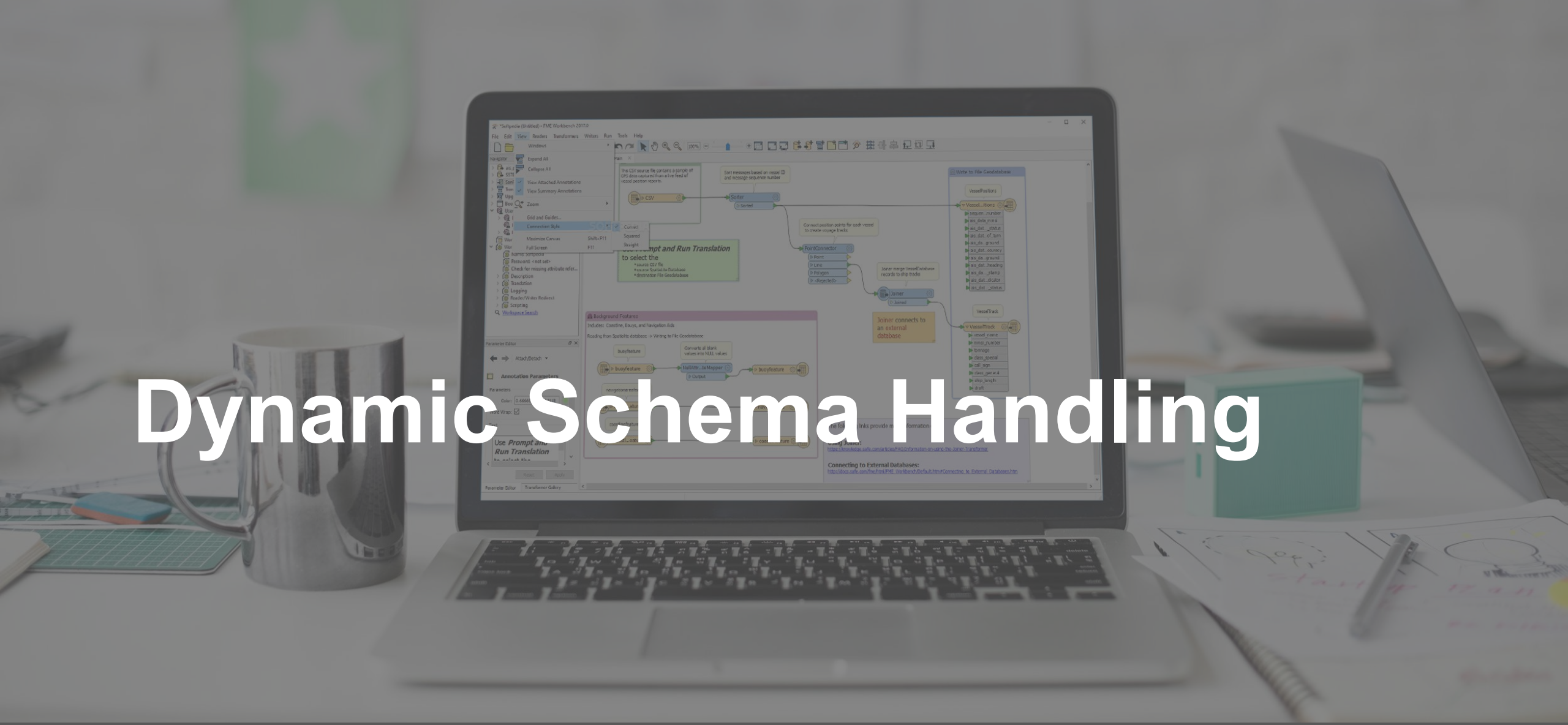


Dynamic Translations

Generic vs Dynamic

It's important to understand the differences between generic and dynamic translations, and when to use each of these tools. This table is quite useful as a reference guide.

	I know the format	I don't know the format	
I know the feature types	Static	Generic + Static	**I know the attributes**
I don't know the feature types	Fanout	Generic + Fanout	**I know the attributes**
I know the feature types	Dynamic	Generic + Dynamic	**I don't know the attributes**
I don't know the feature types	Dynamic	Generic + Dynamic	**I don't know the attributes**



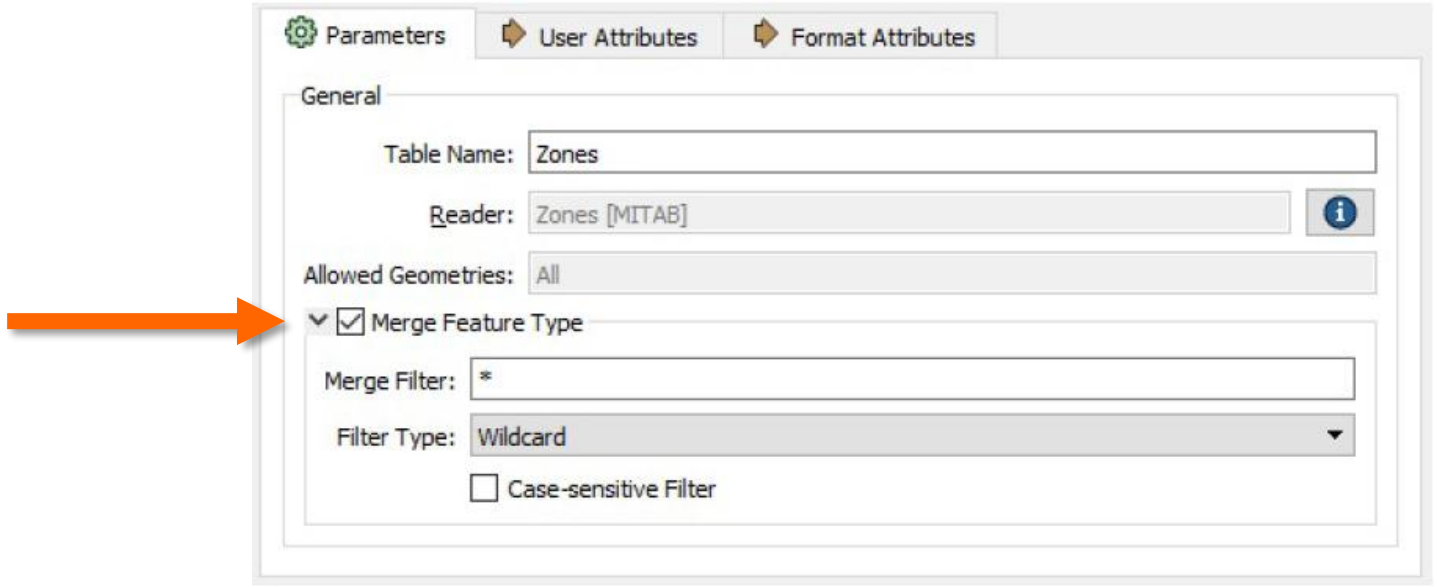
Dynamic Schema Handling

Connect. Transform. **Automate**

Schema Handling in Dynamic Translations

We've looked at creating Dynamic Reader and Writers so now we will look in more depth about the schema handling.

On the **Reader**, simply use a **Single Merged Feature Type**, easy:



Schema Handling in Dynamic Translations

However, on the **Writer,** the parameters are a bit more complex:

The three components of the schema all have to be set in different ways:

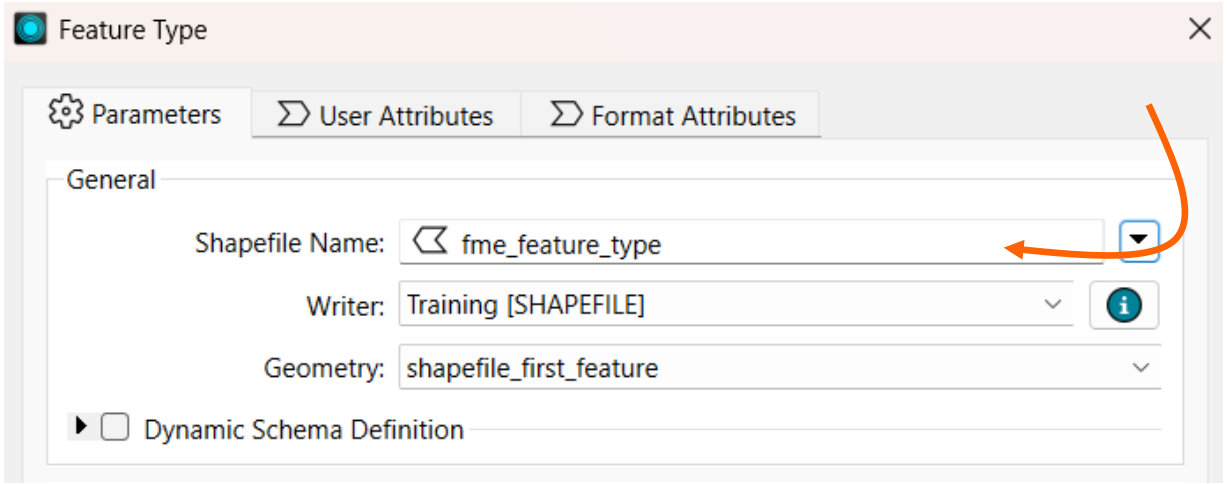
1. Feature type
2. Geometry
3. Attributes

Dynamic Schema Handling - Feature Types

In a dynamic translation a **Fanout** is set automatically for the output feature type name

By default **fme_feature_type** attribute is used

All data is written to the same feature type as it came from, and we get an output that is a duplicate of the input.



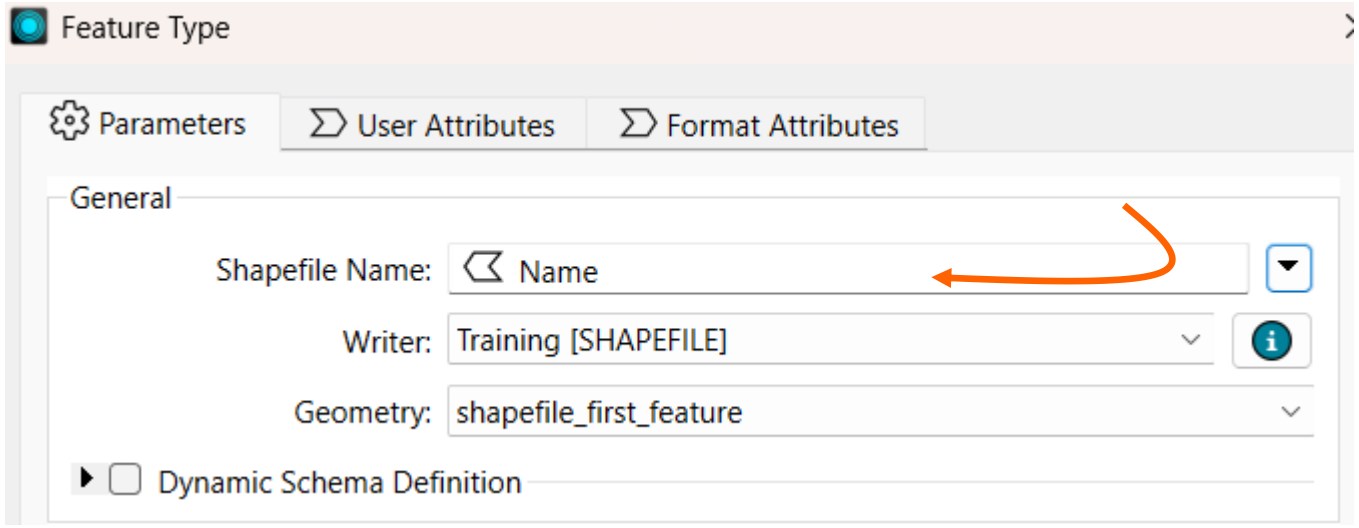
Dynamic Schema Handling - Feature Types

Note!

Changing the output feature type name is **not as simple as a normal fanout**.

A **dynamic workspace** fetches that definition from somewhere else (the schema source).

- e.g. here author is using Name to supply the name of the layer to create



The feature type name chosen must match a layer that exists in the source schema dataset.

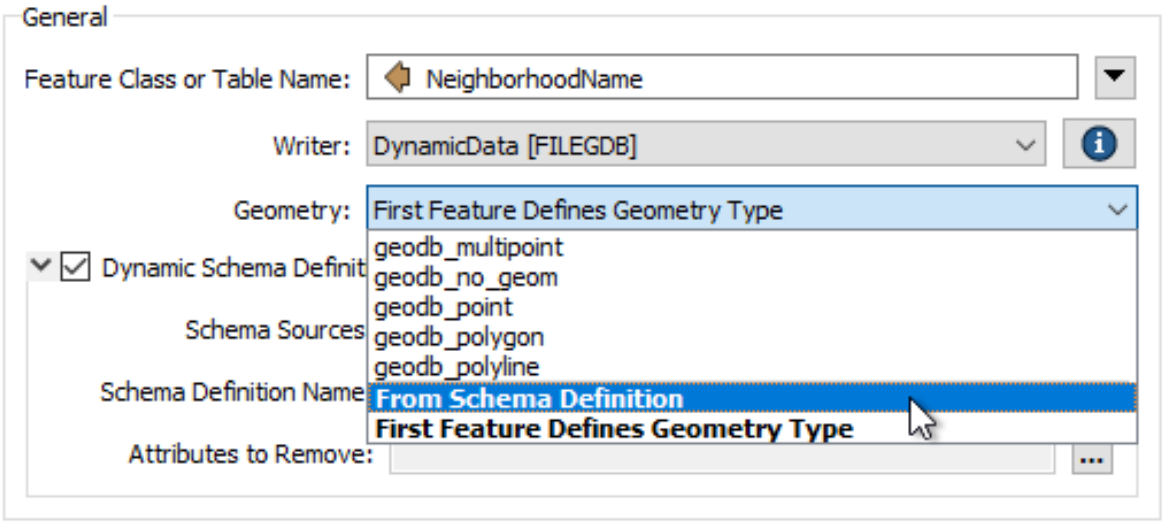
Failure to do so will lead to the data being dropped!

Dynamic Schema Handling - Geometry

In a standard translation you can select the geometry type within the Writer Feature Type.

In a Dynamic workspace, the geometry type can be specified in 3 ways:

- **From Schema Definition** - the permitted geometry is defined by the schema source dataset
- **First Feature Defines Geometry** – useful if the author does not need to know in advance what geometry is being processed or what geometry the schema permits
- **Manually Fixed Geometry Definition** - this will override the geometry defined by the chosen schema source



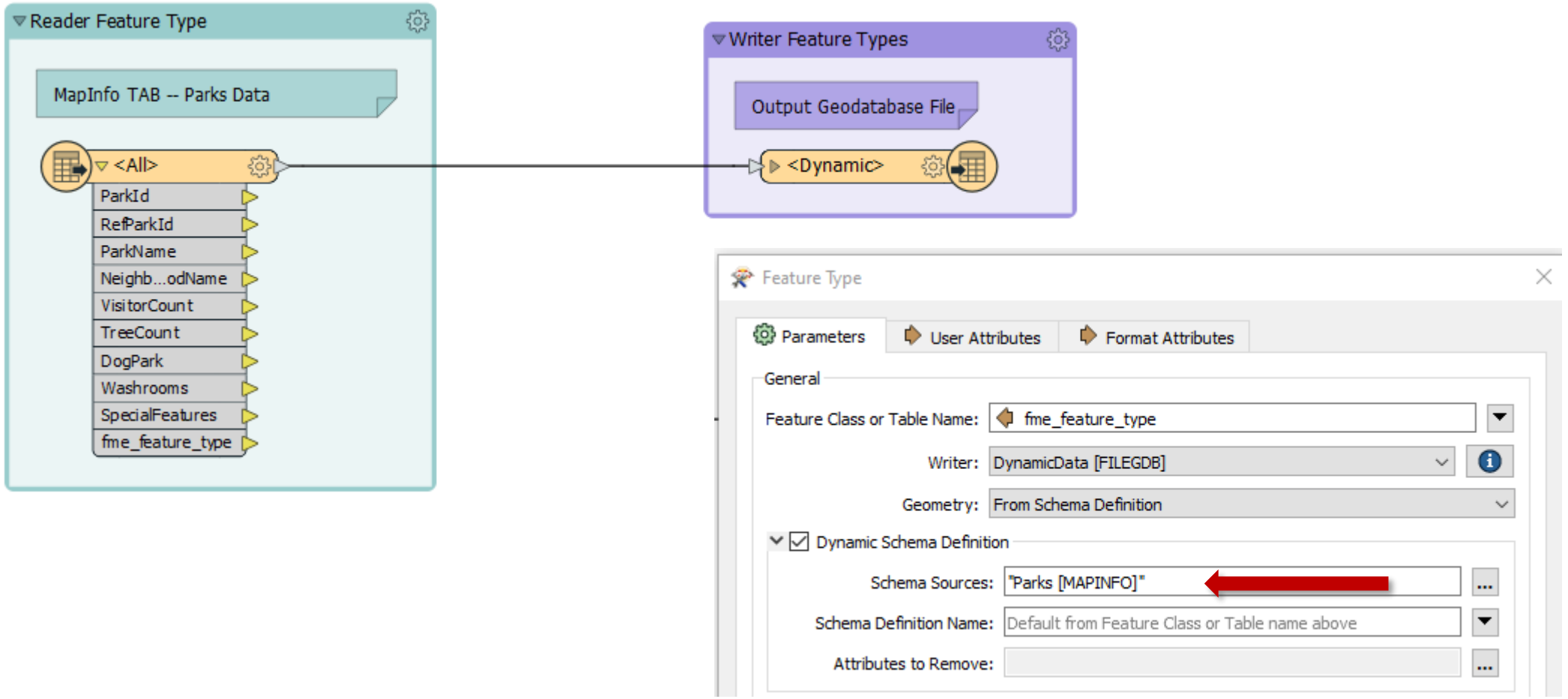
WARNING:
If the geometry of the data to be written is different from that schema, and the destination format does not support multiple geometry types, then features would be dropped instead of written.

Dynamic Schema Source

Connect. Transform. **Automate**

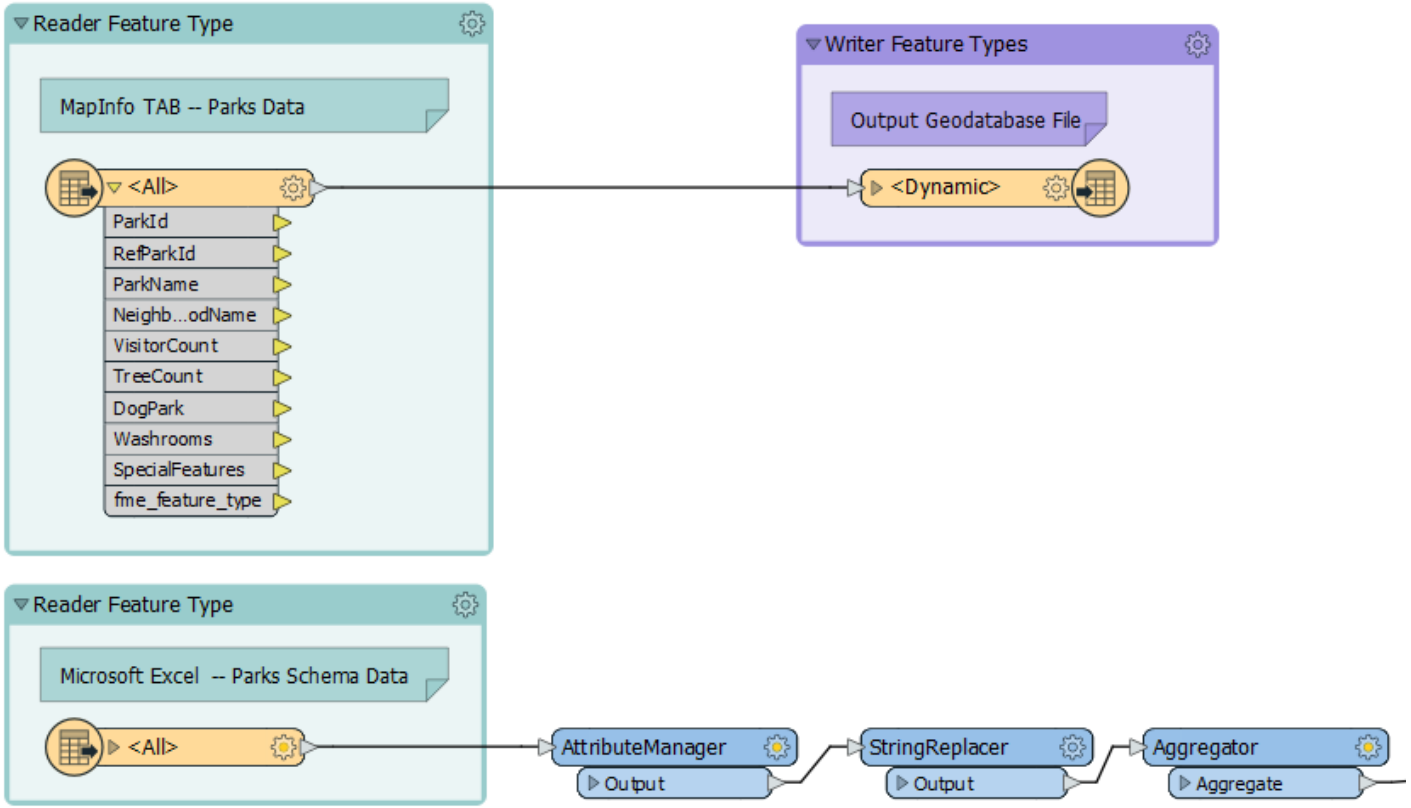
Dynamic Schema Sources – Same as Source

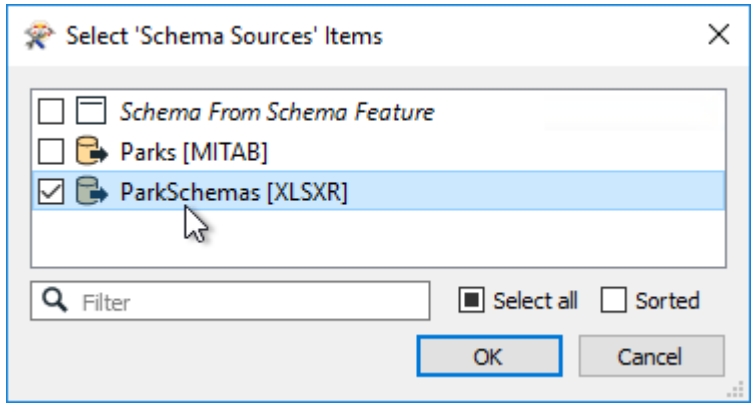
In this example, we have a straight translation to the Parks Geodatabase file so the schema source is set to the Parks MAPINFO schema source



Dynamic Schema Sources – From Another Reader

Here the workspace author is again converting parks data from MapInfo to a Geodatabase. The workspace also includes another Reader (Parks Schema Excel). The author has chosen the Excel dataset as the required structure for the output dataset.





Dynamic Schema Sources – Resource Reader

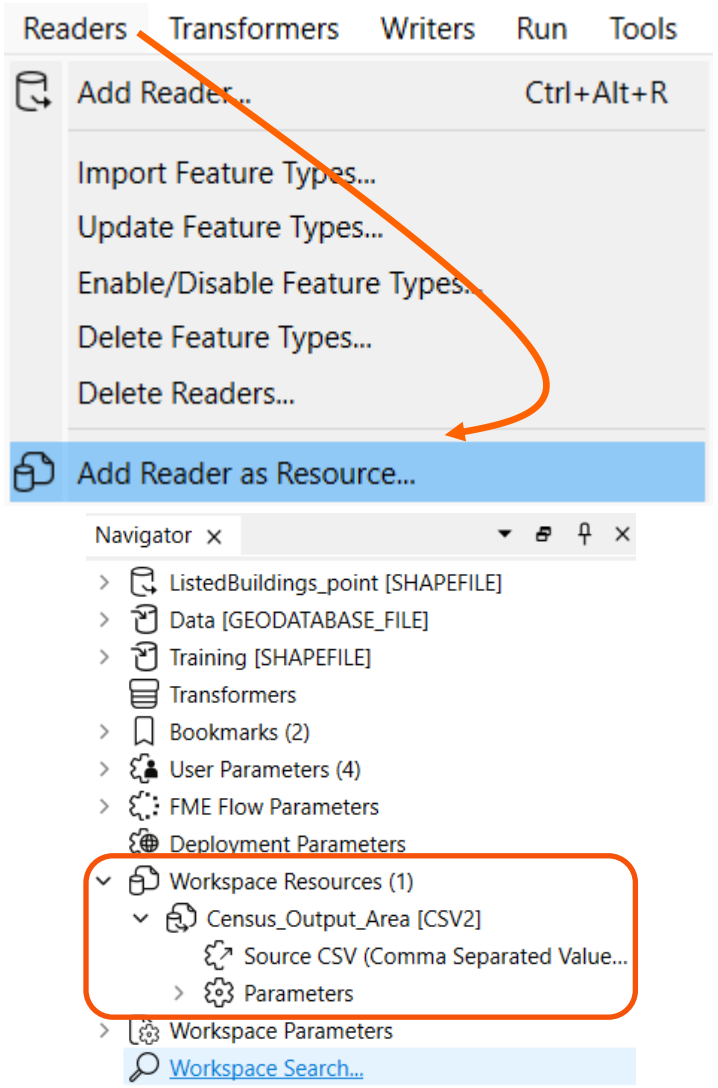
The Schema Sources parameter can be set to point to any Reader as a source of dynamic schema.

However, in most cases all we need from the dataset is the schema, not the data.

This is where **Resource Readers** can be used.

A Resource Reader **returns the schema of a dataset, but no data**.

Once available in the Navigator Panel, the Resource Reader can be used as the source of a schema for a dynamic Writer.

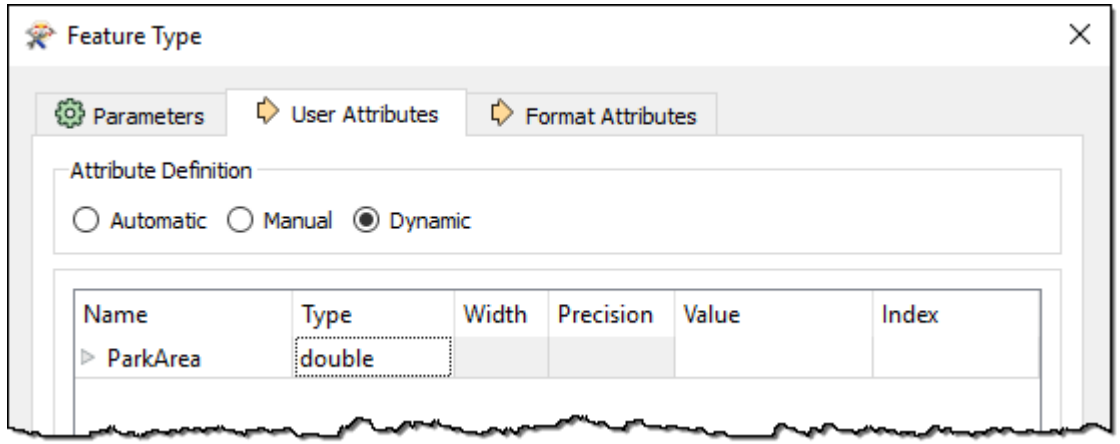


Adding or Deleting Attributes from Dynamic Schema

You can add and delete hard-coded attributes in a dynamic workspace:

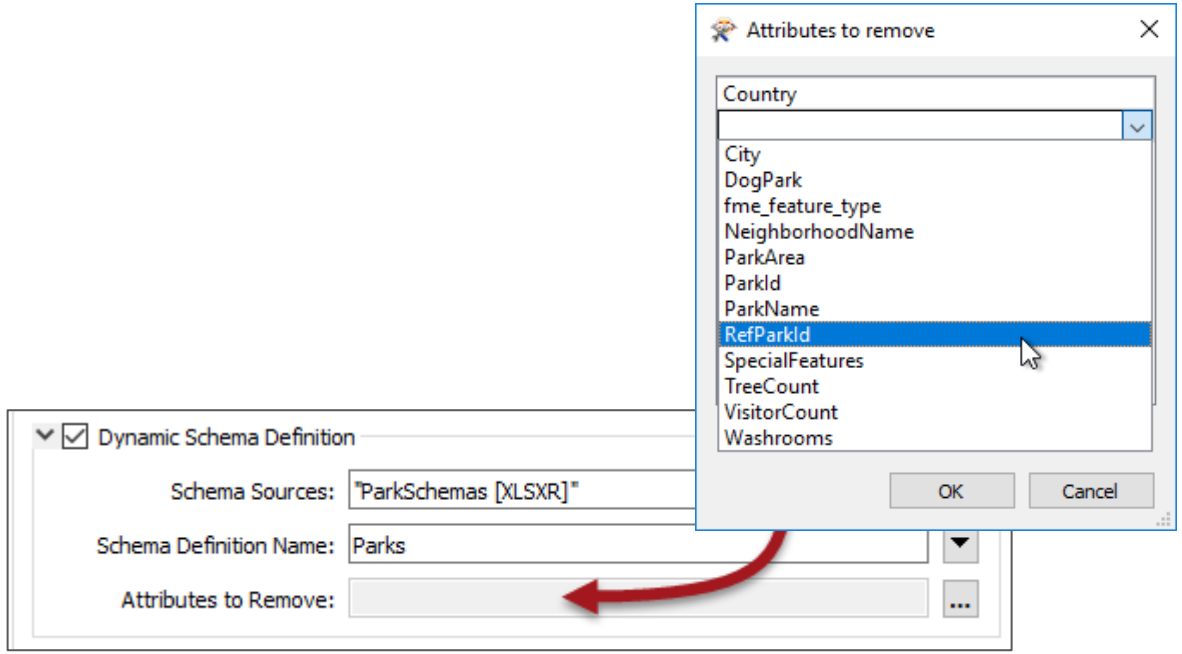
Adding a New Attribute

To add a new attribute to all output on a dynamic feature type, editing the feature type definition to add that attribute.

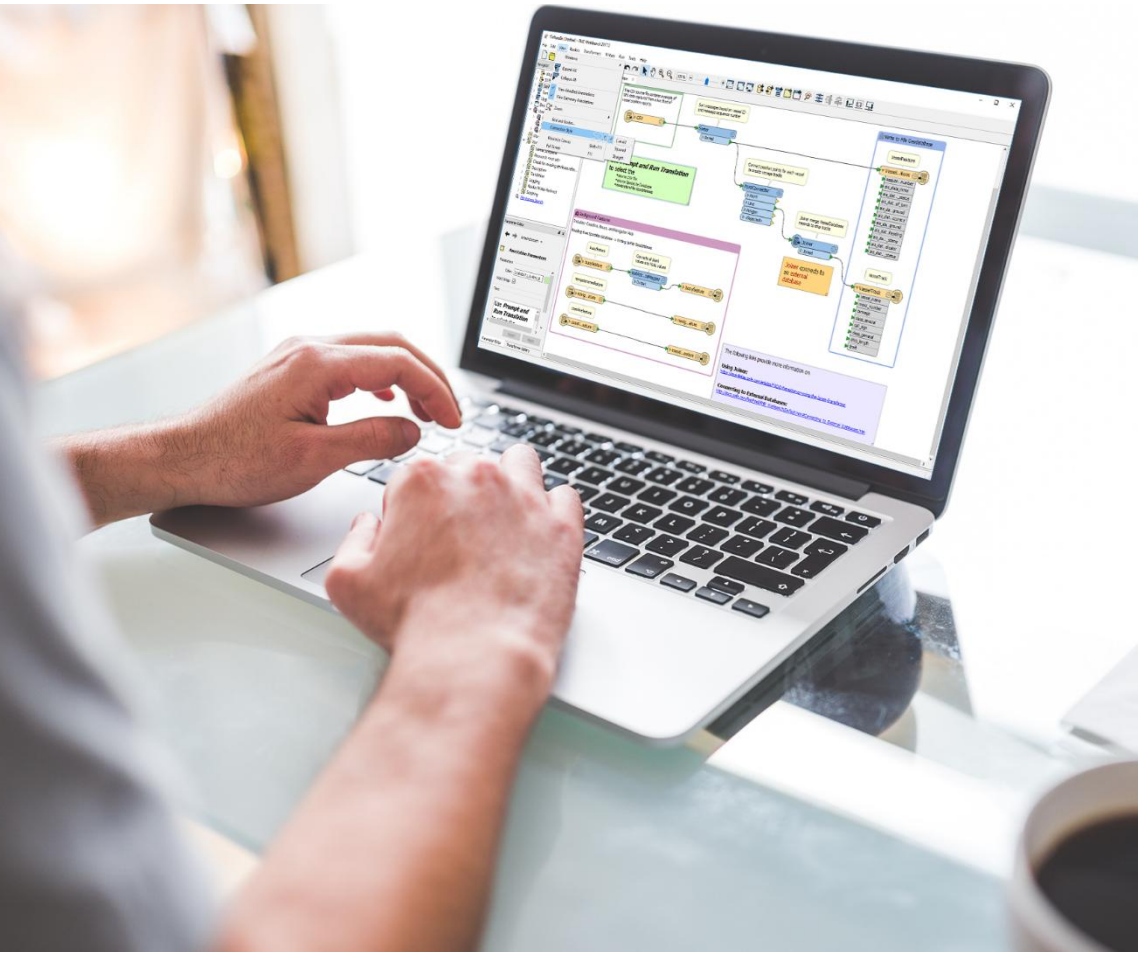


Deleting Attributes

It is possible to delete attributes through the dynamic Schema Definition dialog:



Exercise 7.4



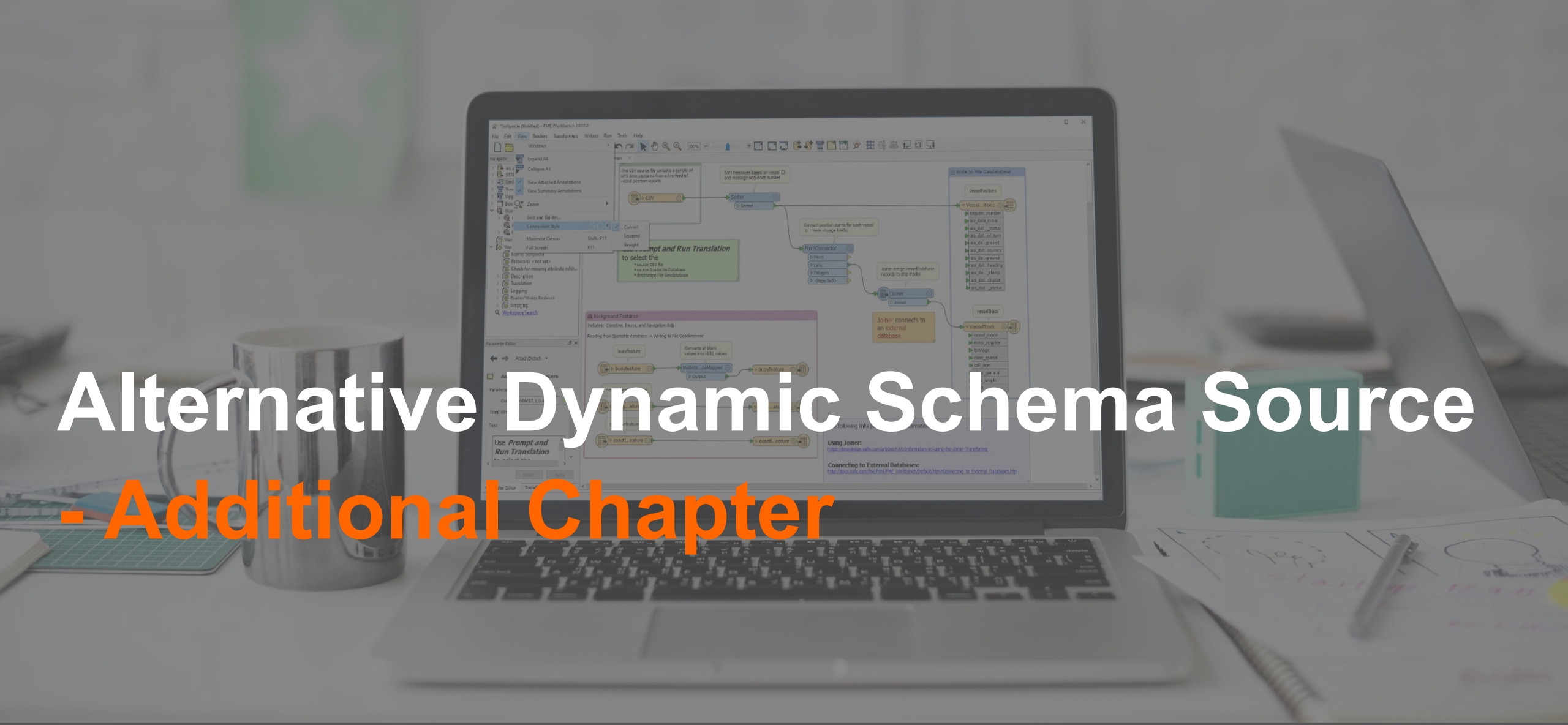
Dynamic Schema Handling

- 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.

Data -
- Firehalls (GML)
- Parks (MapInfo TAB)
- *Schema for Parks = CommunityMap (ESRI Geodatabase)*

Starting Workspace – none

Goal - Create a workspace to process two data sources and dynamically handle the schemas; including make use of a schema from another dataset.



Alternative Dynamic Schema Source

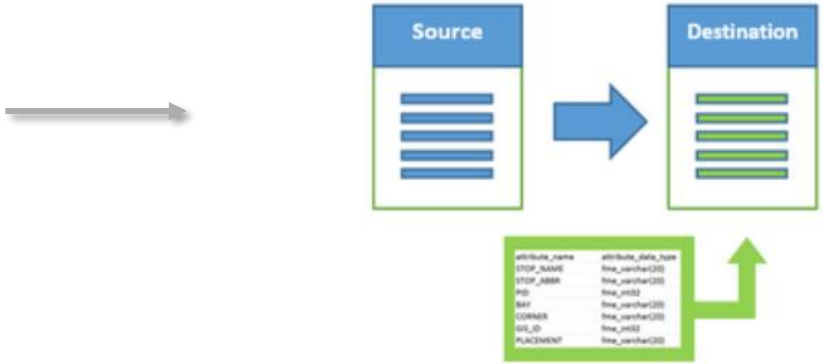
- Additional Chapter

Connect. Transform. **Automate**

Alternative Dynamic Schema Sources

There are two other scenarios for providing the output schema:

- **Tabled-Based Schemas** - A schema can come from a lookup table (text file or spreadsheet) in which definitions are stored



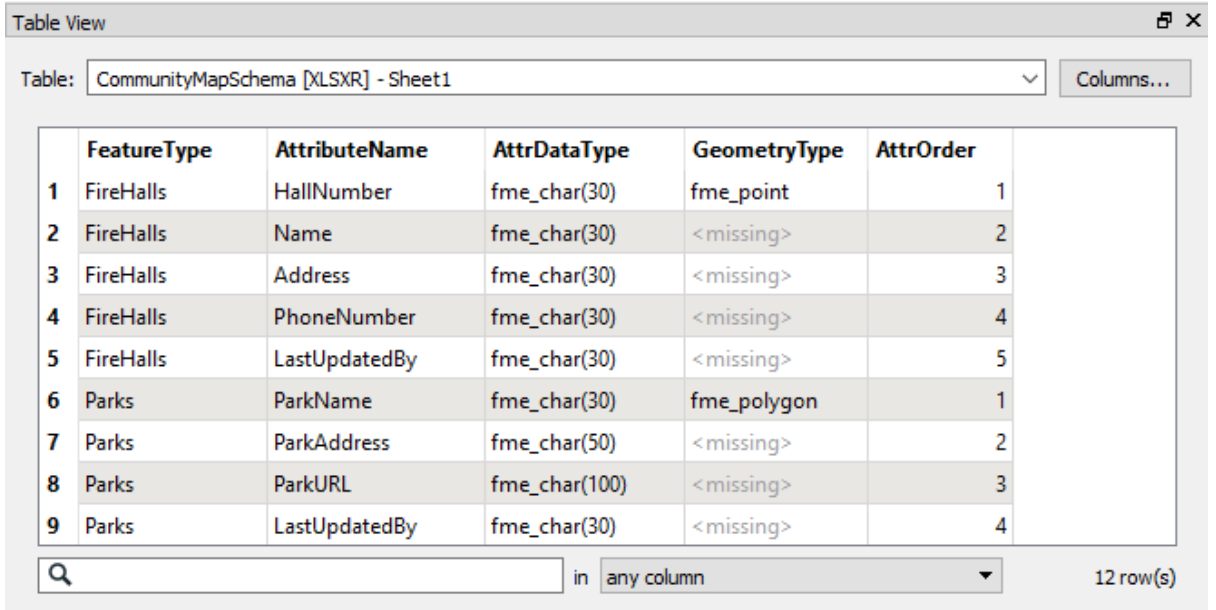
- **Constructed Attribute Schemas** - A schema can be defined dynamically as a list of attributes in a workspace

Table-Based Schemas

Table-Based Schemas

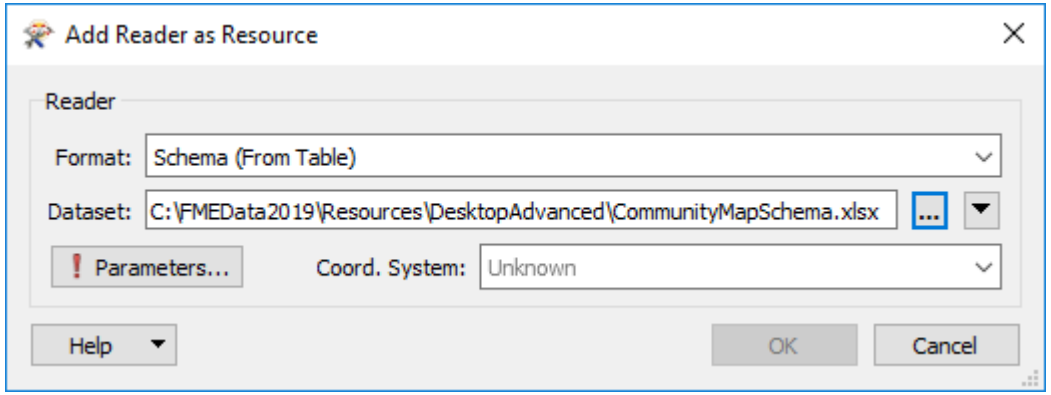
This type of workflow is useful when maintaining a schema outside of the FME environment.

The output schema is stored as an external text file or spreadsheet.



	FeatureType	AttributeName	AttrDataType	GeometryType	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

In FME they would use this schema by adding a **Resource Reader**. The format of the Resource Reader would be Schema (From Table).



Bonus Exercise 7.5

Dynamic Workspace - Table-Based Schema

- 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.

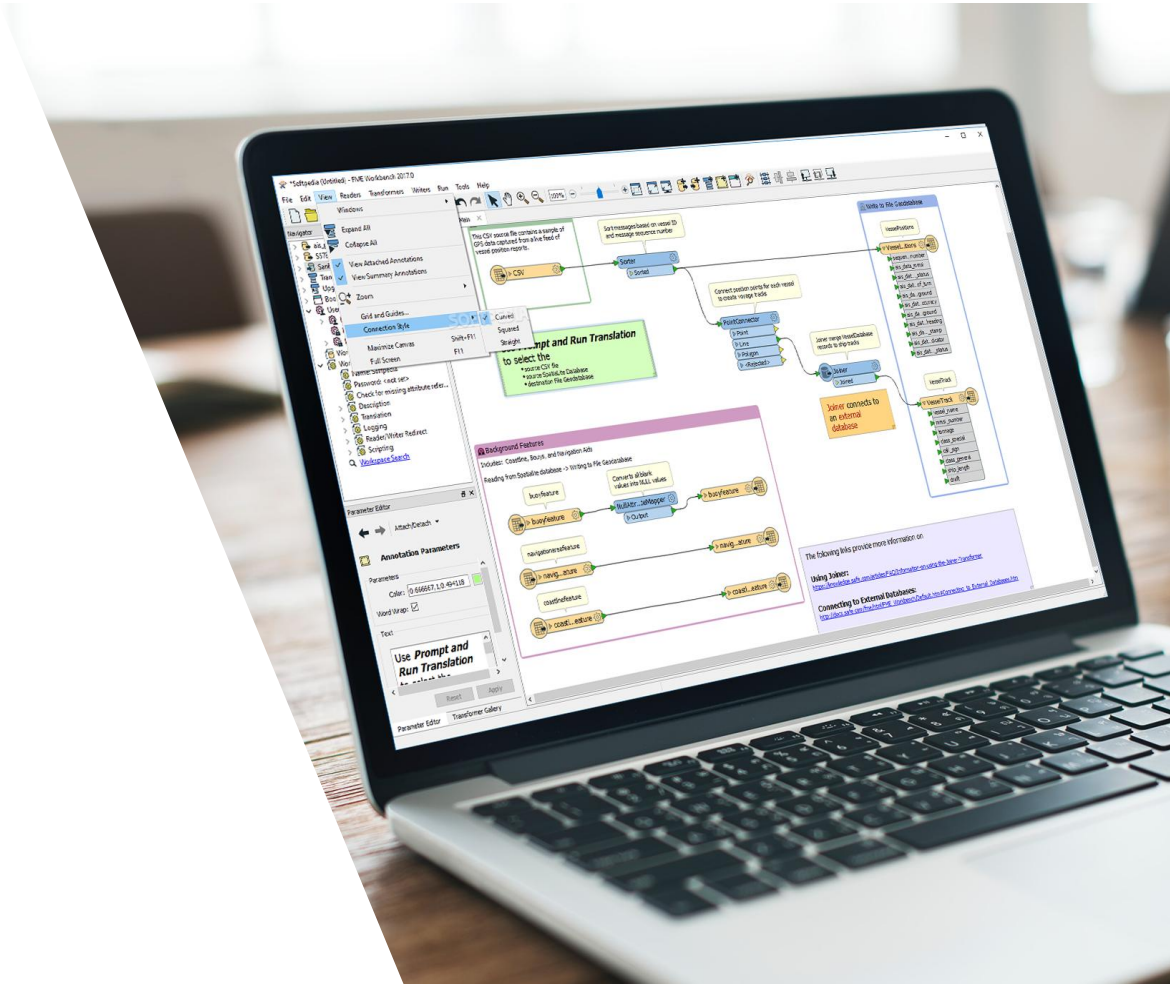
- Rather than having to make changes each time they add more datasets, you believe that you can simply create an Excel spreadsheet containing the schema definition. That way the service area team can edit it themselves and do the same for all future updates.

Data –

Firehalls (GML)
Parks (MapInfo TAB)
Zones (MapInfo TAB)
Schema Table = CommunityMapSchema (Microsoft Excel)

Starter Workspace: C:\FMEModularData\Workspaces\7.05-AdvancedWorkflow-TableBasedSchema-Begin.fmw

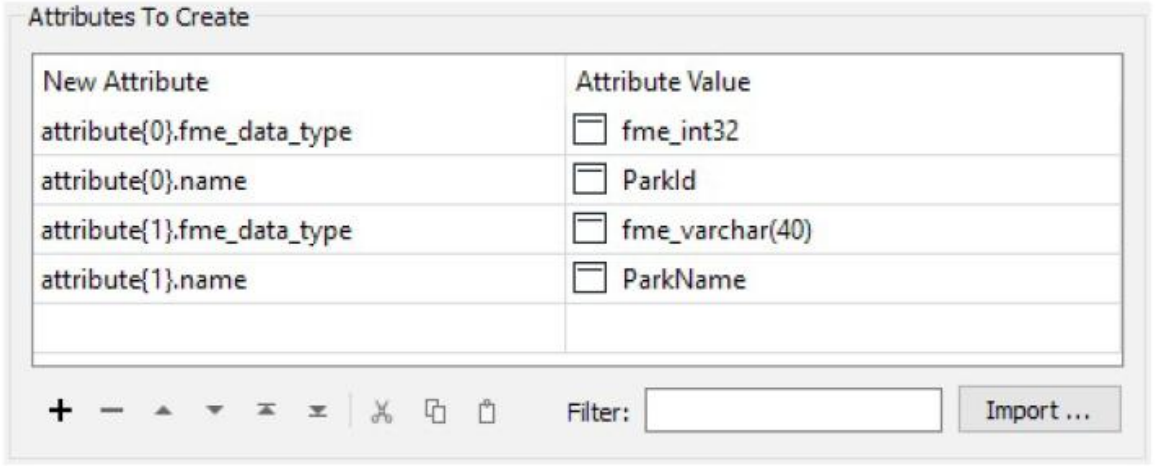
Goal - Generate a new Community Mapping dataset using a table-based schema



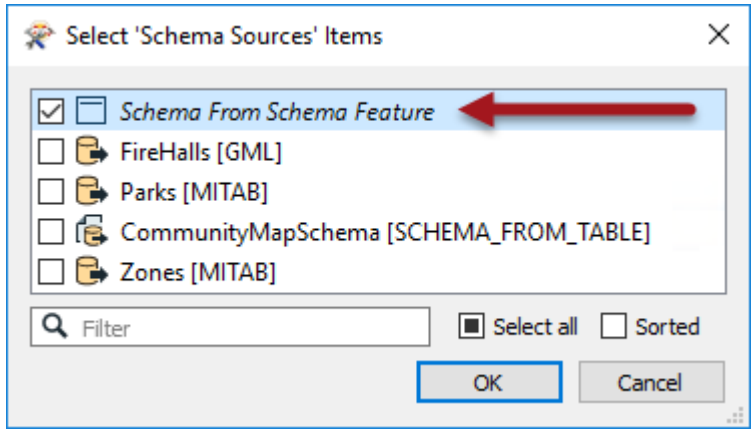
Constructed Attribute Schemas

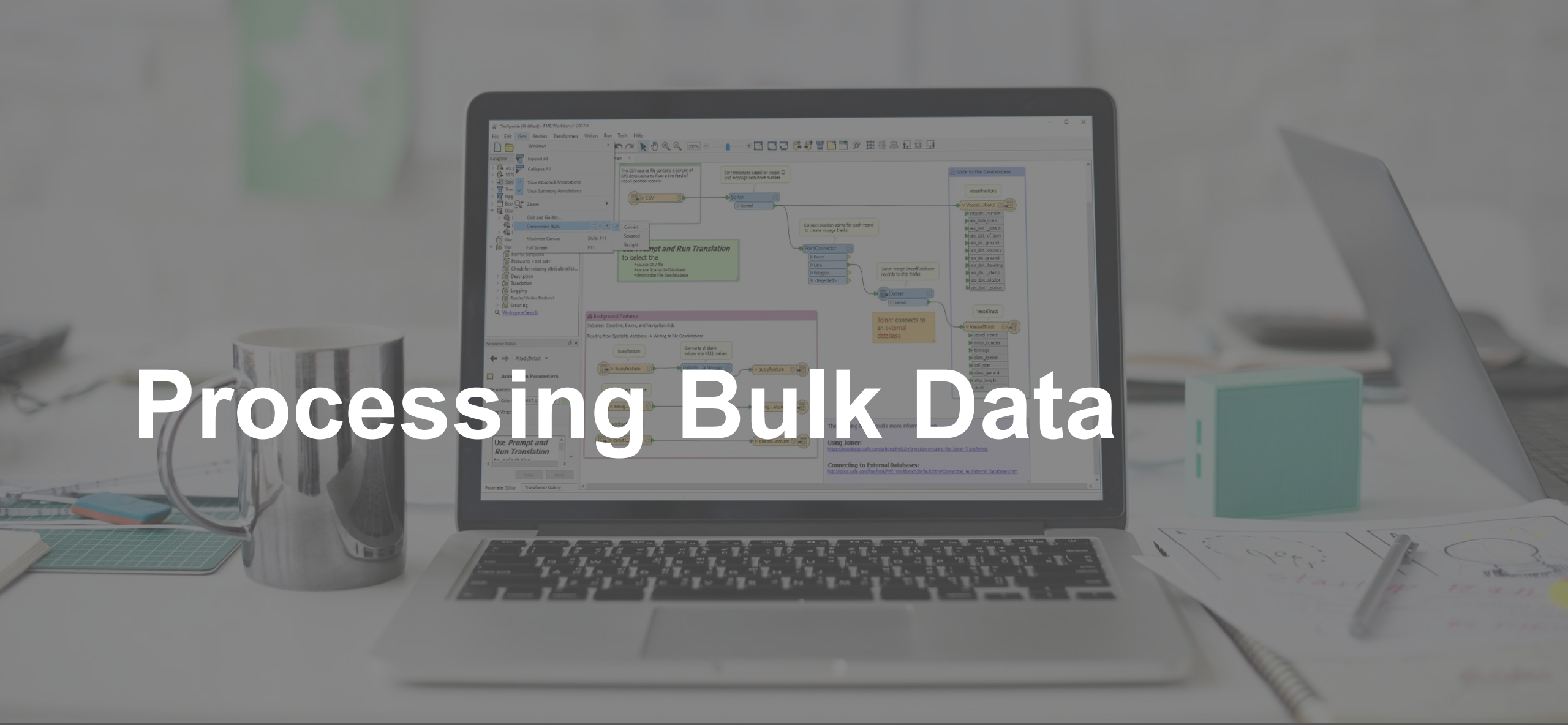
Constructed Attribute Schemas

A single feature will contain a List. The schema is defined by using attributes in this List.



The Writer is told to use this schema in preference to any others by selecting it as the ‘Source Schema’.





Processing Bulk Data

Connect. Transform. **Automate**

Processing Bulk Data

Batch Processing

Batch processing is the action of processing multiple source datasets in the same workspace. By splitting each dataset into a separate process, performance can be improved.
FME Desktop Batch Processing techniques:

- **Batch Processing Using the WorkspaceRunner**
- **Batch Deploy Wizard** - lets you create batch files to save and execute outside of FME Workbench

Parallel Processing

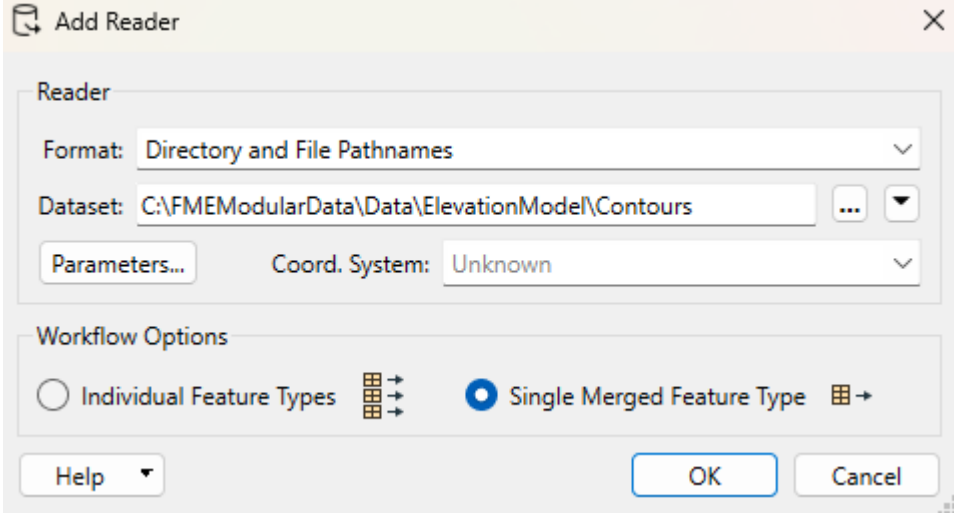
You can process data in bulk with FME Desktop **through custom transformer** parallel processing.
Parallel processing runs on features grouped by attribute values, similar to Group-Based transformers. Custom transformers using parallel processing assign each group to a separate process.

Connect. Transform. **Automate**

Directory and File Pathnames Reader

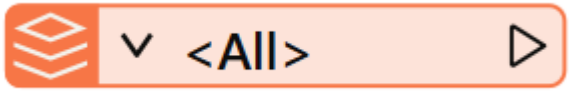
The **Directory and File Pathnames Reader** produces a feature for each file and/or folder contained in the specified folder.

The feature produced contains the full pathname, the folder, and the basename of the file or folder in attributes, which can then be used by any transformers to operate on pathnames



Why is this useful?

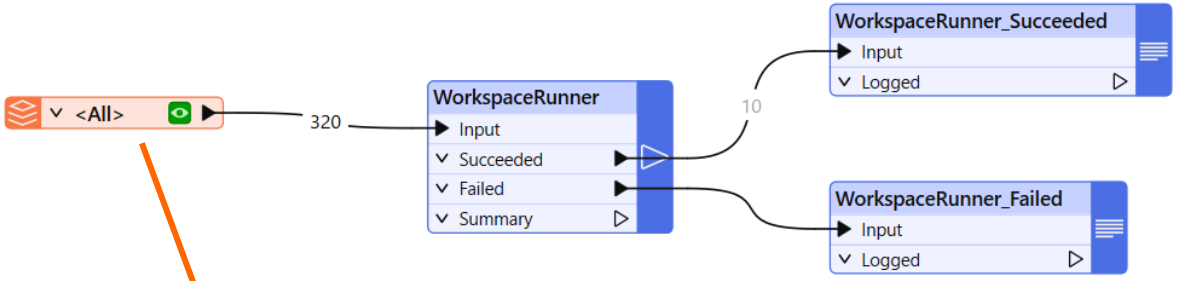
The **Directory and File Pathnames** reader used in conjunction with a **FeatureReader** is **great for batch processing data**!



Batch Processing – Using WorkspaceRunner

The **WorkspaceRunner** is a transformer that runs another FME workspace.
It can be used to run multiple workspaces simultaneously, in a form of batch processing - *and is especially effective used in conjunction with the Directory and File Pathname reader.*

Each feature passed into WorkspaceRunner will trigger a workspace to run (and process the detailed file/chunk)



Visual Preview | Translation Log

Table

<All>

	path_unix	path_windows	path_rootname	path_filename	path_extension	path_directory_unix	path_directory_windows	path_type	fme_feature_type
1	C:/FMEModular...	C:\FMEModularDa...	J10	J10.dbf	dbf	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
2	C:/FMEModular...	C:\FMEModularDa...	J10	J10.prj	prj	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
3	C:/FMEModular...	C:\FMEModularDa...	J10	J10.shp	shp	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
4	C:/FMEModular...	C:\FMEModularDa...	J10	J10.shx	shx	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
5	C:/FMEModular...	C:\FMEModularDa...	J11	J11.dbf	dbf	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
6	C:/FMEModular...	C:\FMEModularDa...	J11	J11.prj	prj	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
7	C:/FMEModular...	C:\FMEModularDa...	J11	J11.shp	shp	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
8	C:/FMEModular...	C:\FMEModularDa...	J11	J11.shx	shx	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
9	C:/FMEModular...	C:\FMEModularDa...	J12	J12.dbf	dbf	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
10	C:/FMEModular...	C:\FMEModularDa...	J12	J12.prj	prj	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
11	C:/FMEModular...	C:\FMEModularDa...	J12	J12.shp	shp	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH
12	C:/FMEModular...	C:\FMEModularDa...	J12	J12.shx	shx	C:/FMEModularData/Dat...	C:\FMEModularData\Data\Ele...	file	PATH

Exercise 7.6

Batch Processing with the Workspace Runner

- A colleague has a workspace that processes contour data tiles; 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.

Data – Contours (ESRI Shapefiles)

Starting Workspace - C:\FMEModularData\Workspaces\7.06-AdvancedWorkflow-WorkspaceRunnerChild-Begin.fmw

Goal – Use a WorkspaceRunner transformer in conjunction with the Directory and File Pathname reader to perform batch processing of multiple tiles of contour data

Resources

Transformer Gallery

Community pages and forums

Knowledge Base

Each Other


Good luck with FME