



# FME Form Training

**Slide 2 = Module 1 - Getting Started with FME Form**

**Slide 56 = Module 2 - Managing Attributes and Filtering Features**

**Slide 107 = Module 3 - Working with Spatial Data**

**Slide 152 = Module 4 - Workflow Design**

**Slide 191 = Module 5 - Best Practice and Performance**



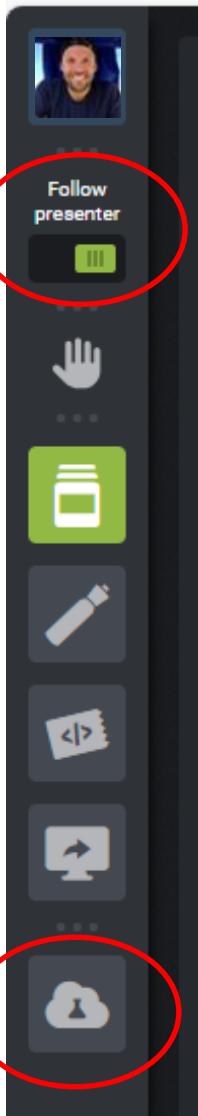
# FME Form Training

## - Module 1

miso

Getting Started with FME Form

# Training Environment



< keep 'Follow presenter' on

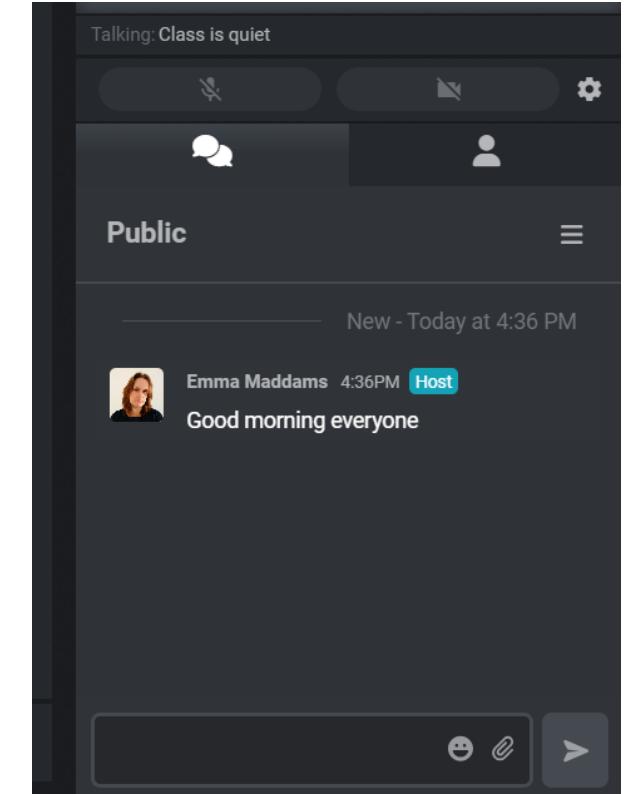
< Lab

need help when using your Lab  
Use either:

- o 'Raise hand' or
- o 'Lab Assistance'

## Chat

- o to everyone
- o 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

# A little bit of background...

---



miso

The logo consists of the word "miso" in a bold, orange, sans-serif font. Above the letter "o", there is a small gray icon of a bowl containing a liquid, with two orange wisps of steam rising from it.

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

Safe  
Software

The logo for Safe Software features the word "Safe" in black and "Software" in a larger, bold, orange sans-serif font.

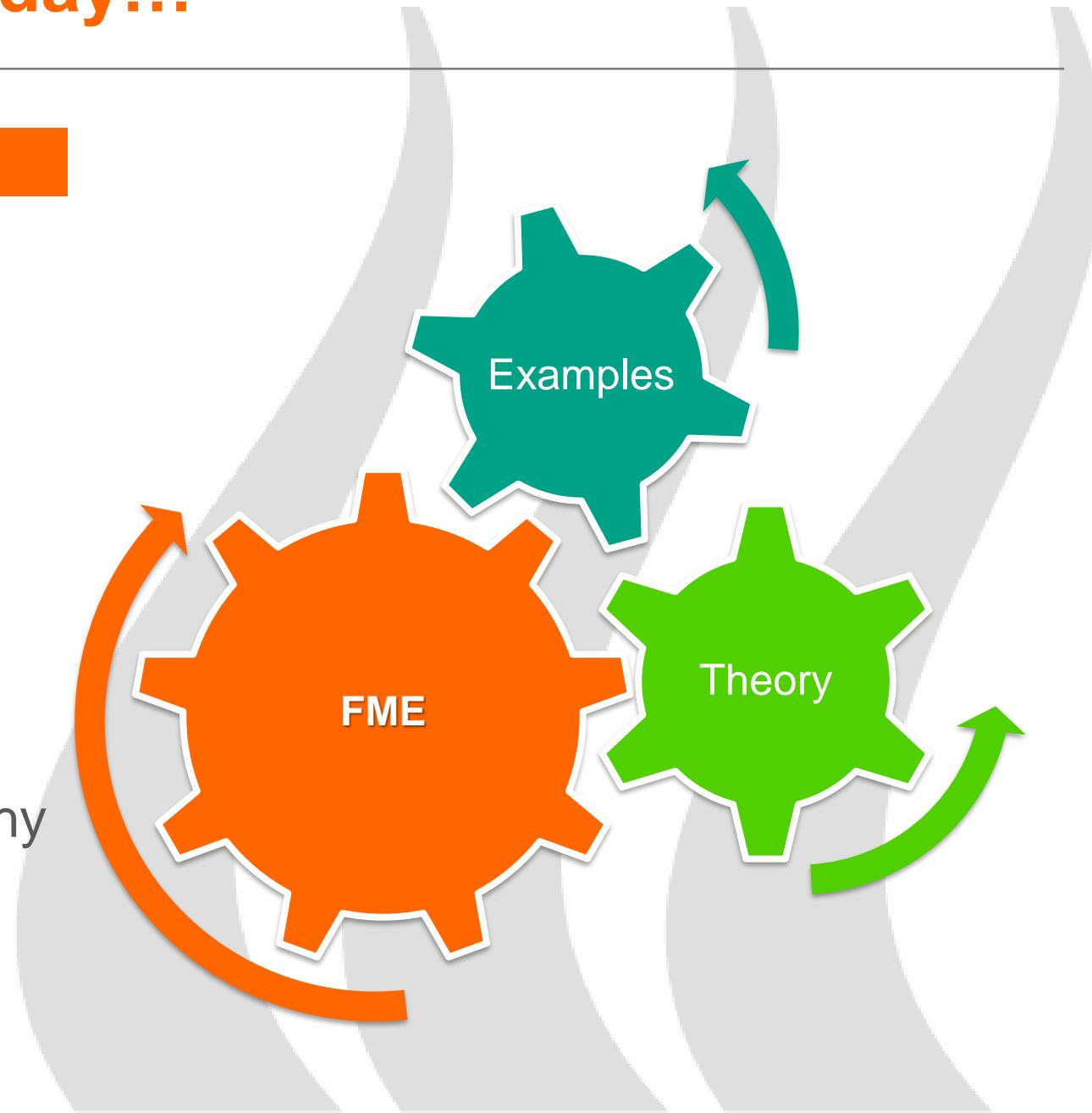
- Based in Canada
- Create FME product suite
- Continuously develop products

# What we'll be covering today...

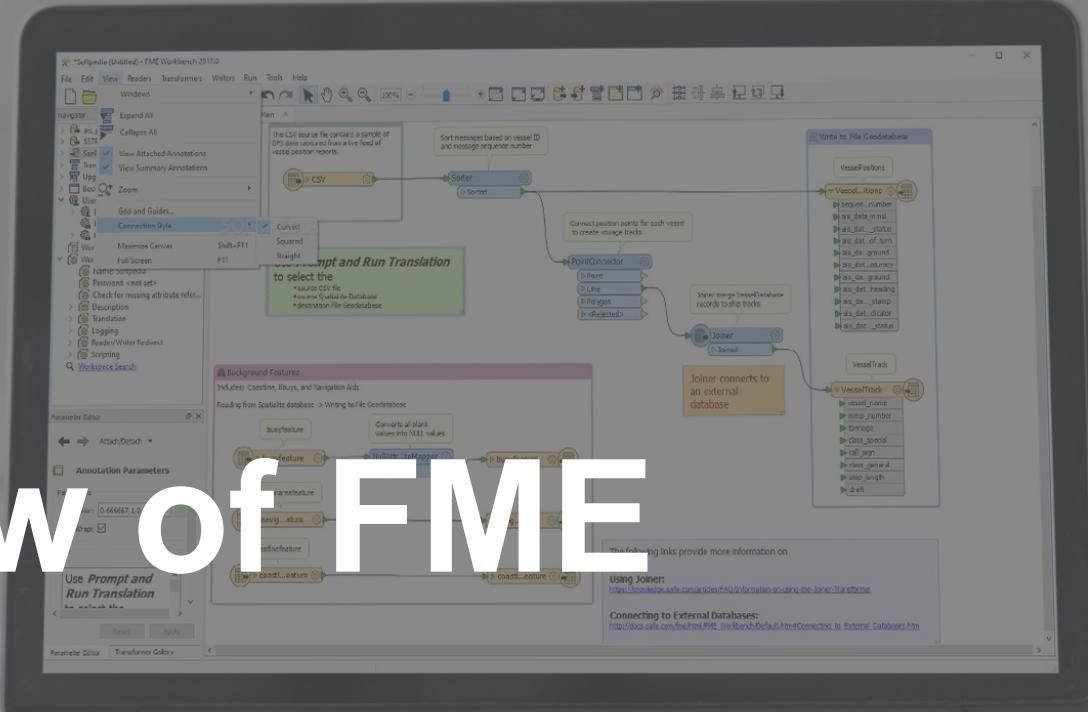
---

## Agenda

- Introductions
- FME Overview
- Inspecting Data
- Workbench Fundamentals
- Generating a Workspace
- Transformers
- FME Components and Hierarchy



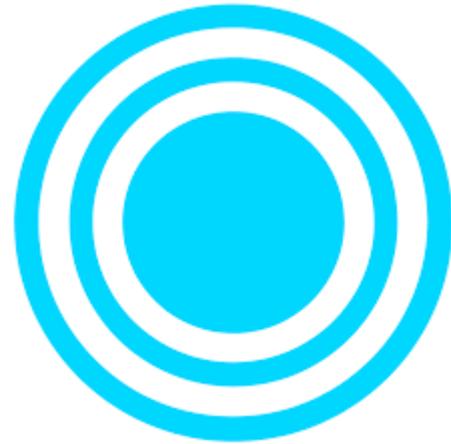
# Overview of FME



Connect. Transform. **Automate**

# FME Product Suite

---

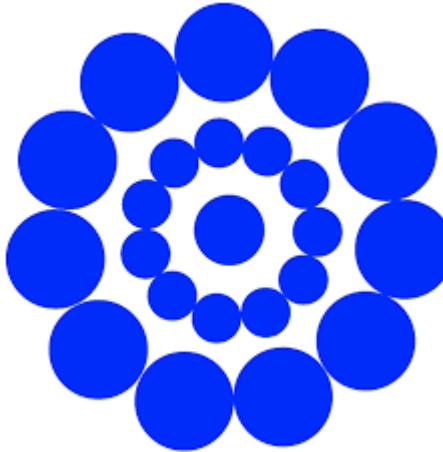


## FME Form

(formerly FME Desktop)

Authoring environment that enables you to create data integration workflows.

The FME Form Workbench is where these Workspaces are created and locally run.

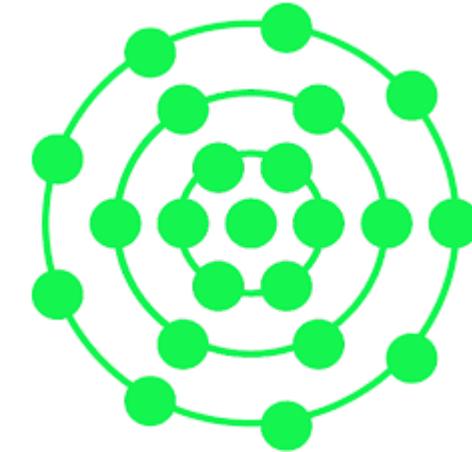


## FME Flow

(formerly FME Server)

The enterprise automation environment that enables you to run your data integration workflows on a schedule or in response to events or in real-time.

FME Flow enables users to integrate their data automatically and distribute it to any web, desktop, or mobile application. Users can set up event-driven workflows, send notifications to stakeholders, and provide self-serve data integration.



## FME Flow Hosted

(formerly FME Cloud)

FME Flow hosted in the cloud by Safe Software, where users pay only for the computing resources they use. It's scalable and uses a pay-as-you-go model.

Linux based; hosted on Amazon Web Services.

# How FME Can Help You



## Data Integration

Convert and transform data so you can access it easily from a single source.



## Application Integration

Create connections between various applications to allow for direct access to data.



## Data Transformation

Alter the structure, content, and characteristics of data to make it more useful for your needs.



## Data Conversion

Configure data for use in specific applications by converting the data format or model.



## Spatial Data

Integrate data with spatial data to gain new knowledge about location intelligence.



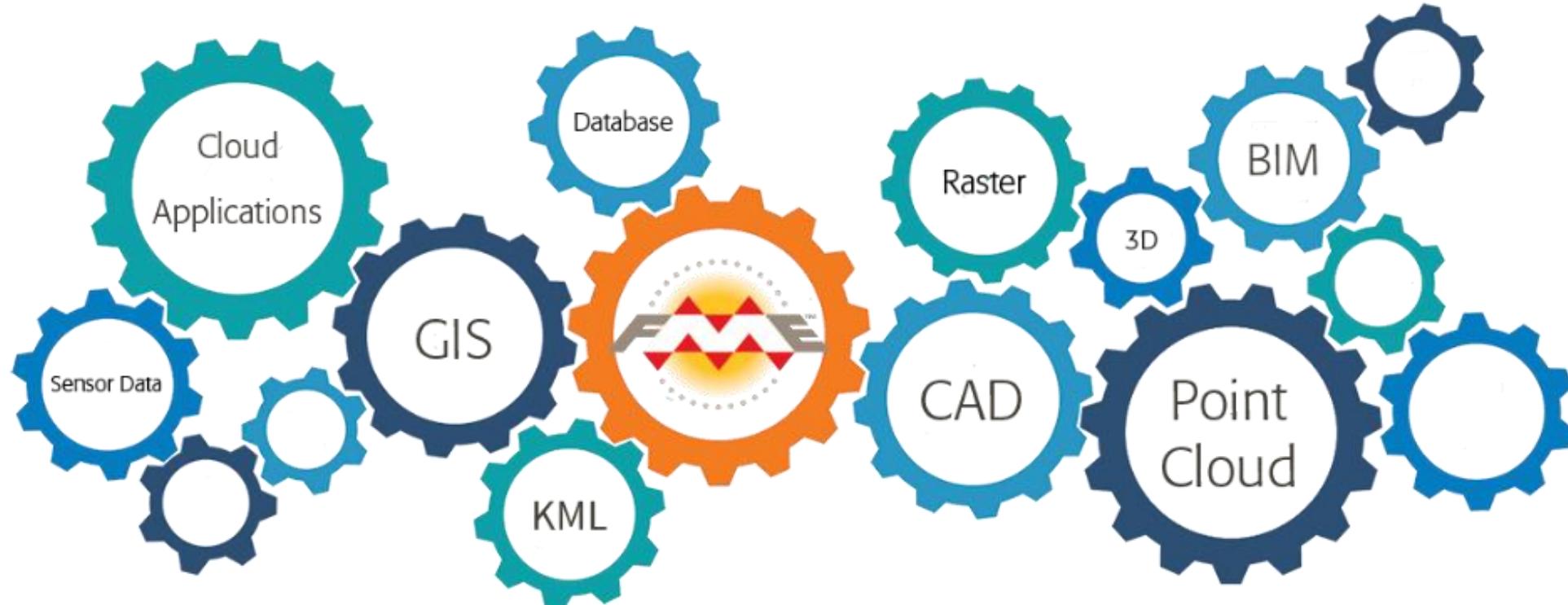
## Data Validation

Verify the quality of data to make better, more informed business decisions.

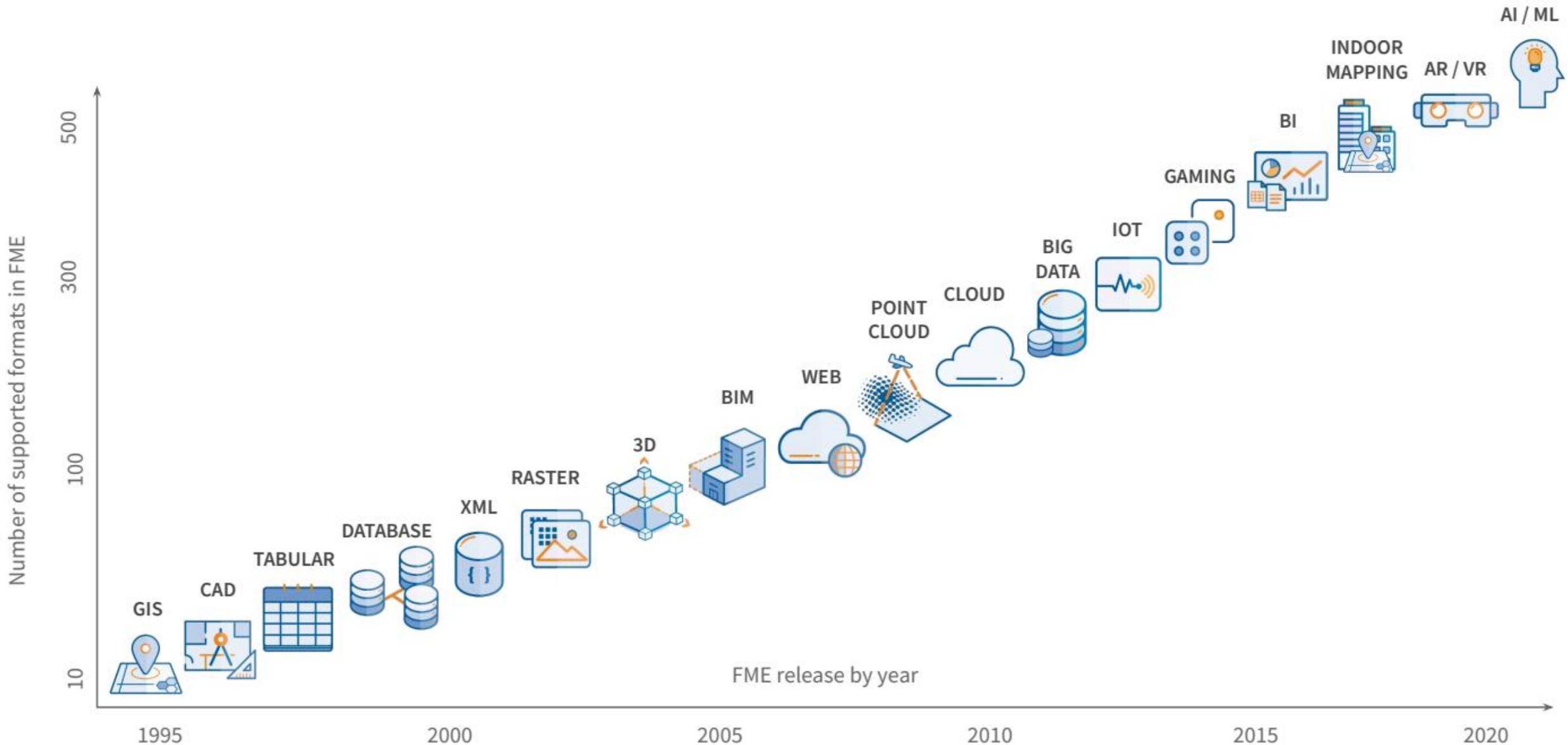
# Formats



FME supports over **450** file formats



# What Data Types Does FME Support?



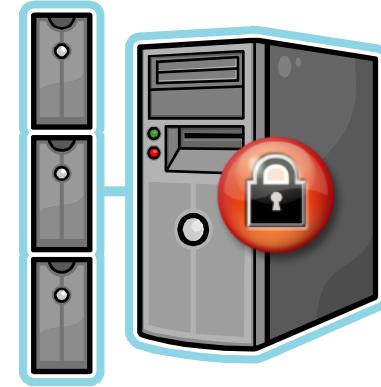
# FME Form Licensing

---



## Fixed License

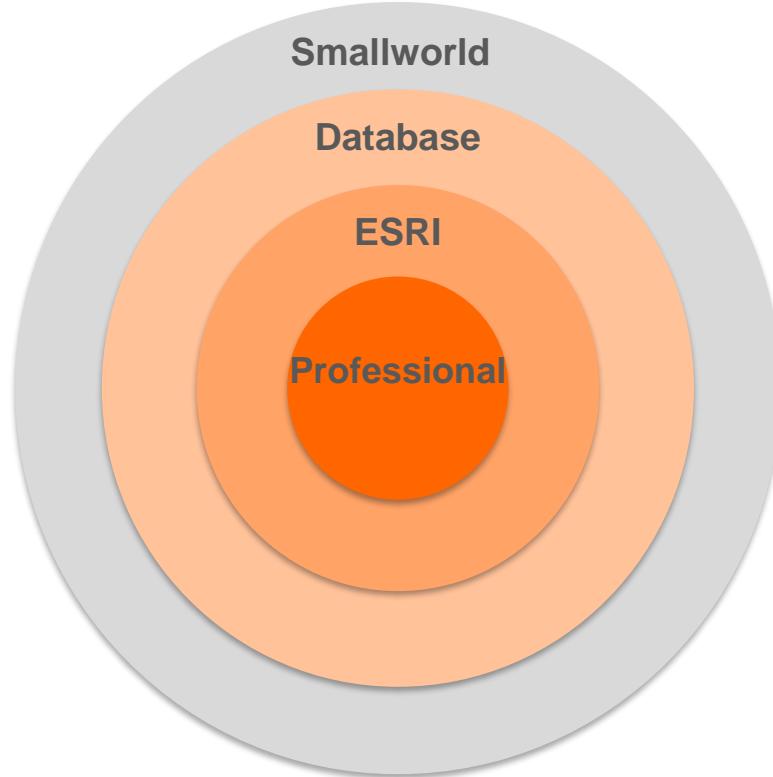
A fixed license allows you to run FME Form on a single computer



## Floating License

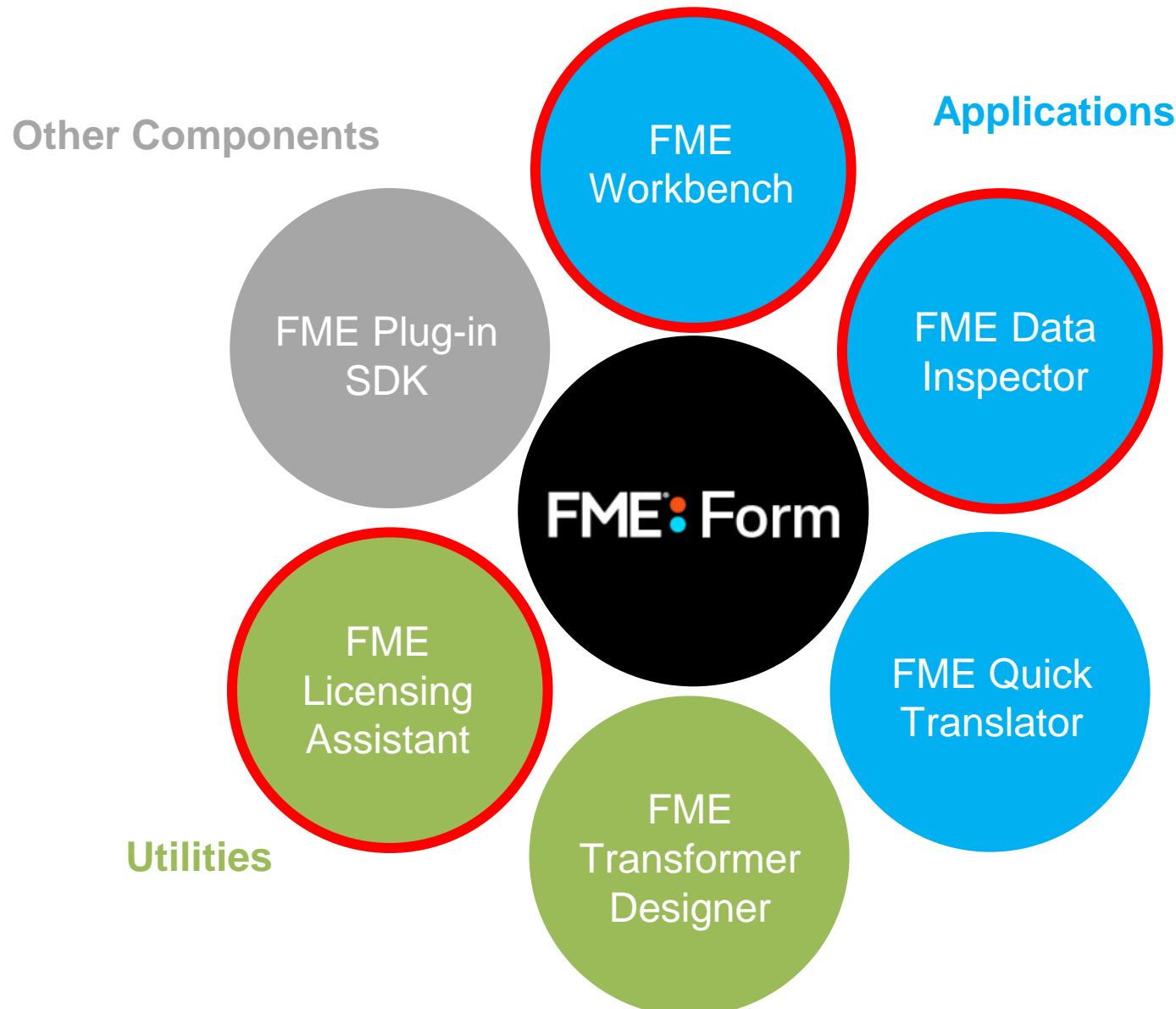
A floating license is shared by multiple people / computers

# Old FME Desktop Editions (pre 2023)



Professional Edition	ESRI Edition	Database Edition	Smallworld Edition
Good for reading & writing most data and applications  <i>ESRI SHP, MapInfo TAB, Geopackage, CSV, Excel, DWG/DXF, DGN, PostGIS, plus more</i>	Added abilities for those working with ESRI's ArcSDE	Write data to popular cloud and on-premises databases  <i>Amazon DynamoDB, IBM, Microsoft SQL, Oracle, SAP, Snowflake, Teradata, plus more</i>	Read and write data to GE Smallworld
	includes everything in the Professional Edition	includes everything in the ESRI Edition	includes everything in the Database Edition

# FME Form Components



# Terminology

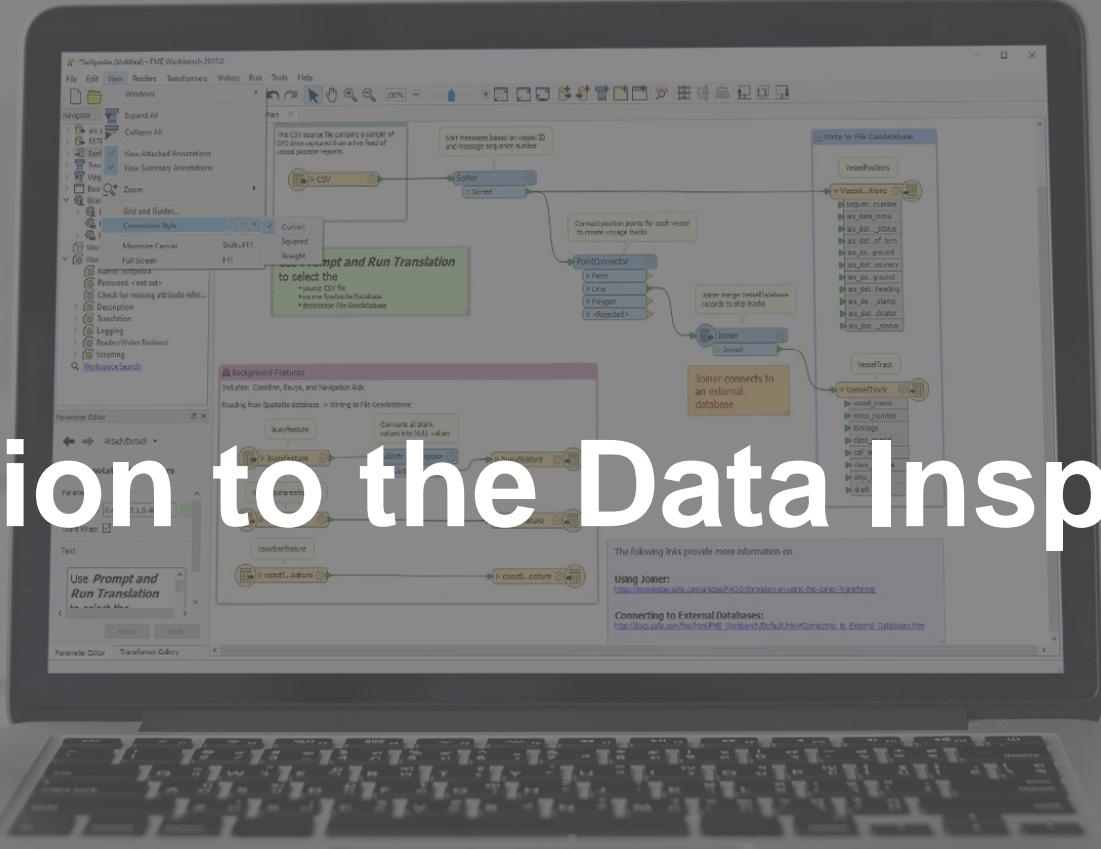


- **FME** - Feature Manipulation Engine
- **Workflow** – the sequence of processes through which a piece of work, data or something else passes from initiation to completion
- **Workbench** – name of the FME application in which a workflow is designed and then saved as a **workspace**.
- **Workspace** – process defined and saved as an **.fmw file** which can be run using the FME Form Workbench, FME Form .EXE or FME Server
- **Canvas** – where you graphically define a translation.
- **Connector** – line used to connect different items on the canvas.
- **Transformers** – the building blocks / tools used in FME Workbench.
- **Readers** – component in a translation that ‘reads from’ a ‘**source dataset**’
- **Writers** – component in a translation that ‘writes to’ an ‘**output dataset**’
- **Attribute** – field/column of data
- **Feature** – a ‘piece of data’ eg. a row or a record



# Introduction to the Data Inspector

Connect. Transform. **Automate**



# User Interface - The Data Inspector



The FME Data Inspector interface is divided into several main windows:

- Display Control Window**: Located on the left, it shows a tree view of layers and their properties. A red box highlights the "Parks [MAPINFO] (80)" layer.
- Map View Window**: The central window displays a map of Vancouver and North Vancouver, highlighting the "Parks" layer in green. A red box highlights the map area.
- Menu/Toolbar**: The top bar contains standard application menus (File, View, Camera, Tools, Window, Help) and various toolbar icons for file operations (Open, Add, Save As, Save Selected, Refresh, Stop, 2D, 3D, Table, Slideshow, Measure), zooming (Zoom Selected, Zoom Extents), selecting features (Select No Geometry, Filter, Mark), and other tools.
- Feature Information Window**: A red box highlights this window, which shows detailed information about the selected feature. It includes sections for Exposed Attributes and Unexposed Attributes, with a search bar at the bottom.
- Table View Window**: Located at the bottom, it displays a table of data from the "Parks [MAPINFO]" layer. The table includes columns for ParkId, RefParkId, ParkName, NeighborhoodName, VisitorCount, TreeCount, DogPark, Washrooms, and SpecialFeatures. A red box highlights the table area.

Below the table, there is a search bar and a status bar showing the current coordinates (X: 495194.0108 Y: 5456517.0801 UTM83-10) and unit (METER).

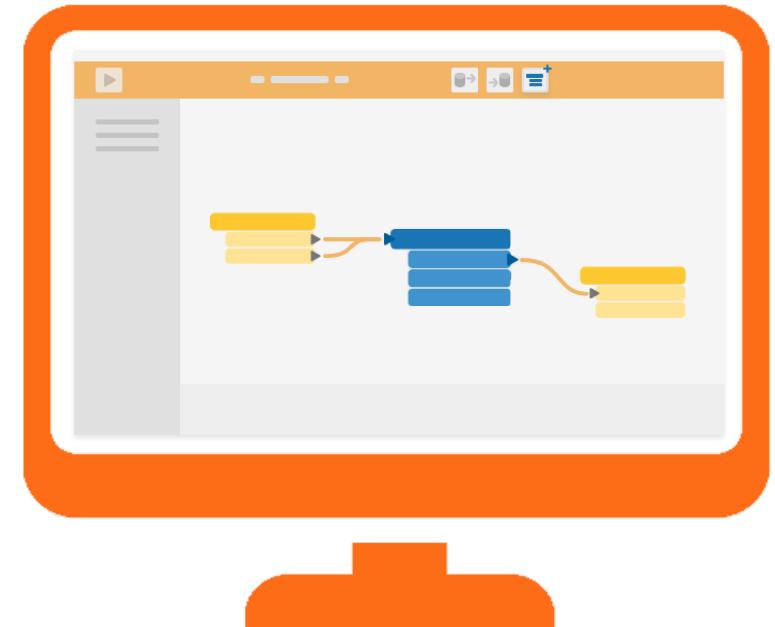
ParkId	RefParkId	ParkName	NeighborhoodName	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures
73	73	-9999	Nelson Park	West End	12868	8	Y	<missing>
74	74	206	Stanley Park	West End	26930	1205	N	Y
75	75	18	Devonian Harb...	Downtown	18459	10	N	Y
76	76	24	Marina Square	Downtown	11014	0	N	Y

# Demo – Data Inspector



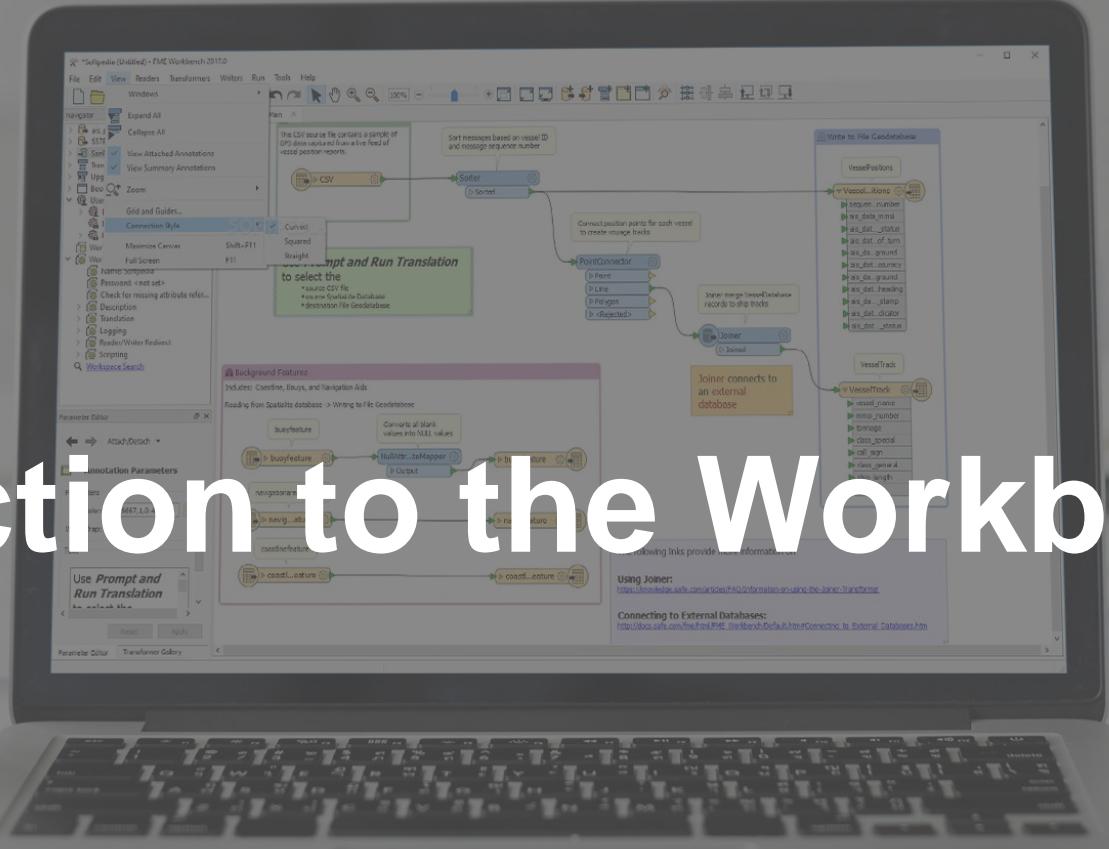
Open ‘Parks.shp’ dataset into Data Inspector to look at:

- Format Gallery
- View map features
- Table view and Attributes
- Selecting features
- Feature Information window
  - Coordinate System
  - Attributes and *properties*
  - Geometry info
- Map Background colour
- Add Background mapping (STAMEN: Terrain)



# Introduction to the Workbench

Connect. Transform. **Automate**



# User Interface - Workbench



The FME Workbench interface is organized into several main components:

- Menu/Toolbar**: Located at the top, featuring a standard Windows-style menu bar (File, Edit, View, Readers, Transformers, Writers, Run, Tools, Help) and a toolbar with various icons for file operations (New, Open, Save, Run, Stop, Undo, Redo) and spatial tools (Select, Pan, Zoom In, Zoom Out).
- Navigator**: A tree-based panel on the left side showing the project structure. It includes sections for Parks [MAPINFO], Training [MAPINFO], Transformers (4), and Parameter Editor.
- Parameter Editor**: A panel below the Navigator showing parameters for selected objects. It includes tabs for Workspace and Work.
- Transformer Gallery**: A panel at the bottom left listing available transformers categorized by type (All, C, E, F, R, S).
- Canvas**: The central workspace where data flows are visualized as a network of nodes and connections. A specific flow is highlighted with red numbers (80, 8, 14) and arrows.
- Translation Log**: A panel at the bottom left displaying log messages from the translation process, including error and warning counts.
- Visual Preview**: A panel on the right showing a map preview of the transformed data, with a green polygon highlighting a specific area.

Menu/Toolbar

Navigator

Canvas

Parameter  
Editor

Transformer  
Gallery

Translation Log

Visual Preview

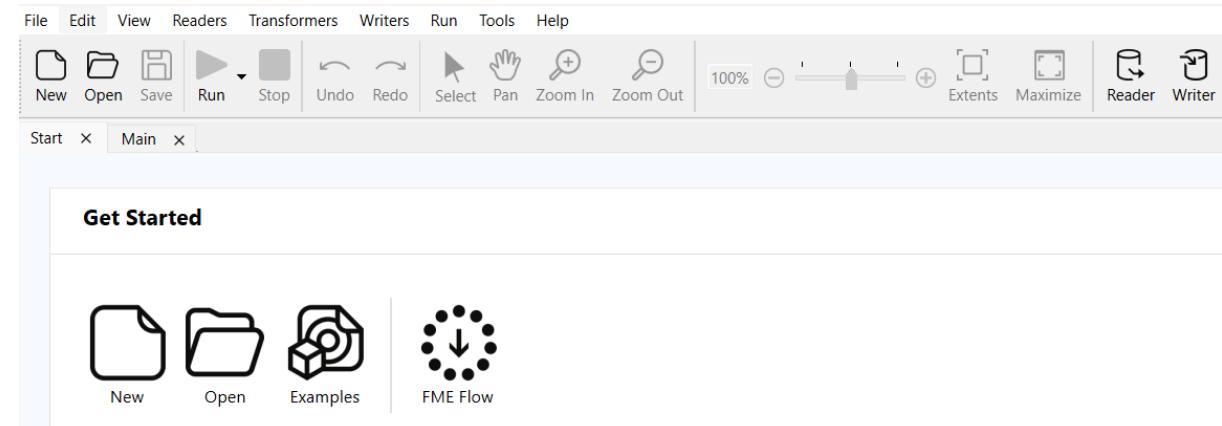
# Workspaces



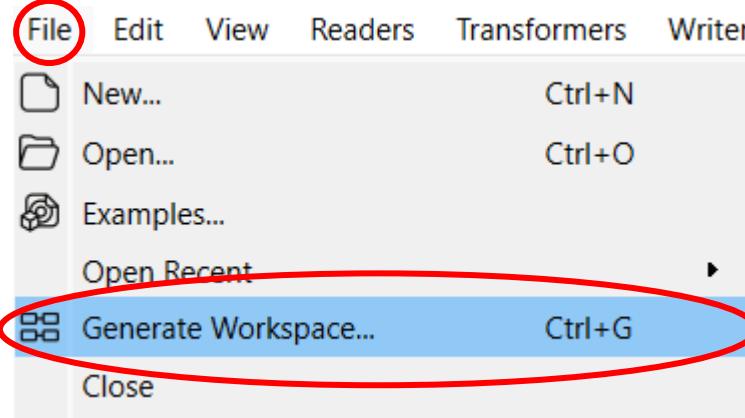
**Workspace** – a process flow defined and saved as an **.fmw** file

The **Workbench application** is used to:

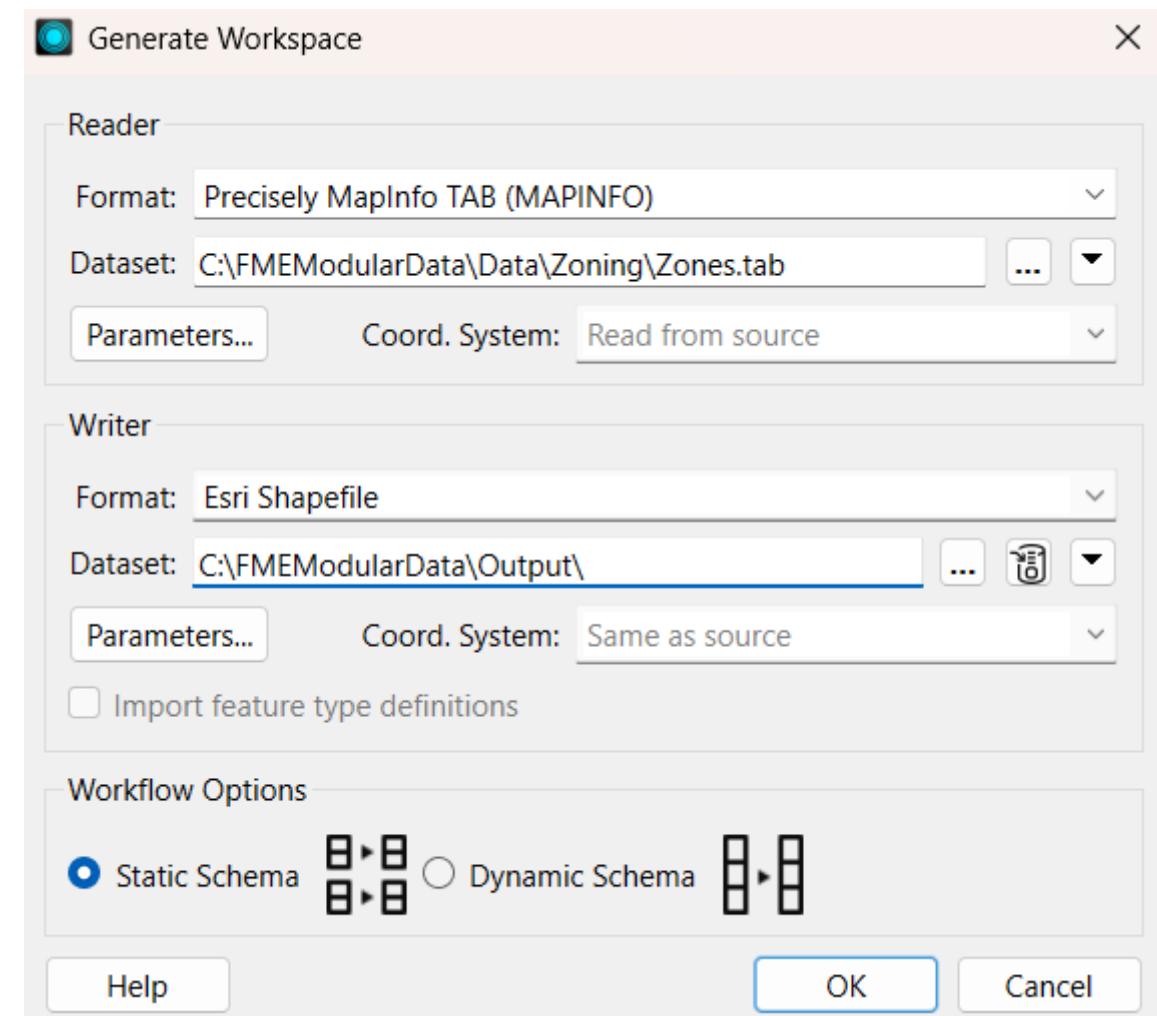
- **Open & Run** existing workspaces
- **Modify** existing workspaces
- **Create New** workspaces
  - from scratch (blank)
  - using Generate Workspace Wizard found in File > Generate Workspace



# Generate Workspace Wizard



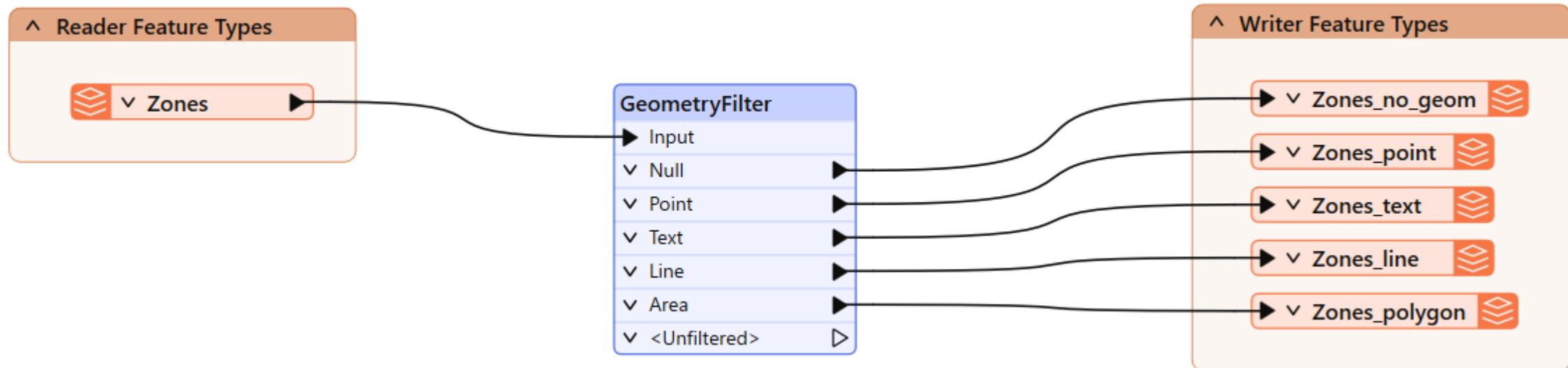
- the Generate Workspace wizard helps you set up a workspace faster – it sets up the readers and writers for us
- you simply need to specify the reader and writer format and dataset locations, and FME does the rest.



# Generate Workspace Wizard



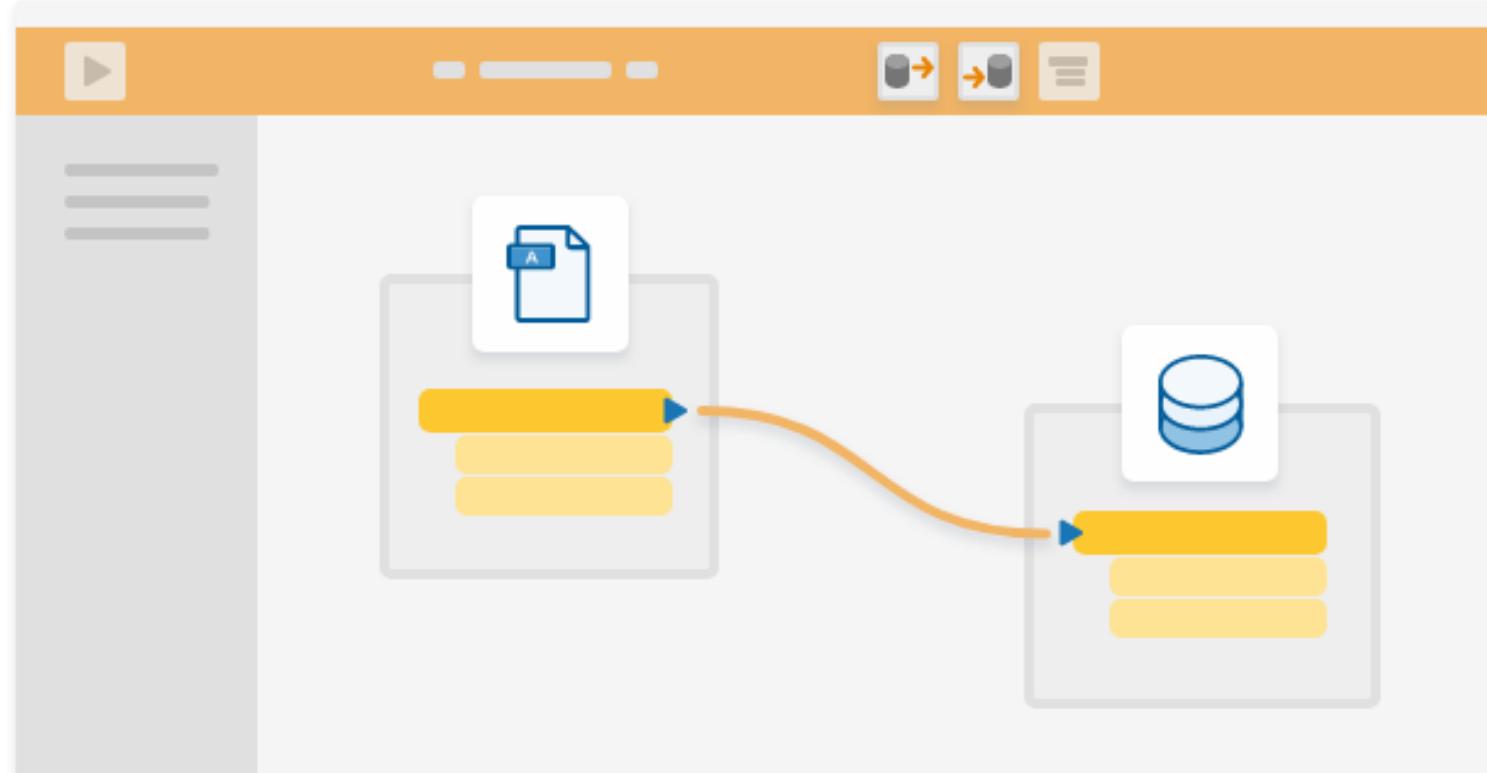
- It will automatically do its best to duplicate the reader schema on the writer, handling differences in data types or restrictions on geometry types or attribute name lengths.



# Data Translations



Data Translations simply convert data from one format to another



.....we'll look at data *Transformation* later

# Demo – Create Workspace using Generate Wizard



- **Data Translation** convert data from one format to another:
  - convert 'Trees' dataset from ESRI Shapefile format to GeoPackage
- Create the workspace using the **Generate Workspace Wizard**
- Save workspace (creating .fmw file)
- Run workspace
- Log file created



# Exercise 1.1



# The Data Inspector and Workbench - Generate Workspace wizard

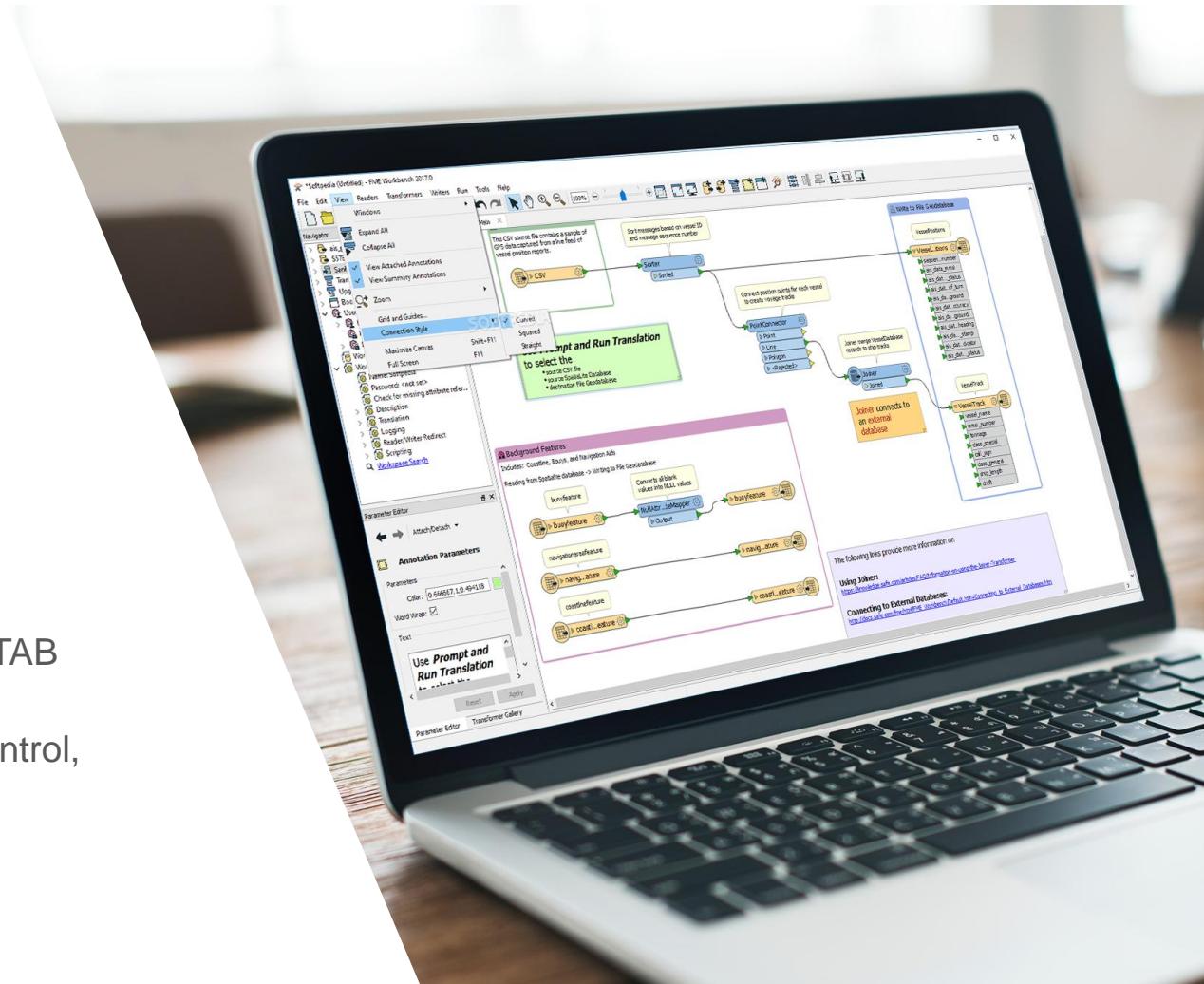
- You have just landed a job as a technical analyst in the GIS department of your local city.
  - On your first day, you've been asked to do a simple file format translation.

## Data - Zoning Data (MapInfo TAB)

## Neighborhoods (Google KML)

## Goals -

- Inspect the zone data before processing
  - Create a workspace to translate the zoning data from MapInfo TAB format to GeoJSON
  - Inspect the output of the translation - understand the display control, view multiple datasets together and add background maps



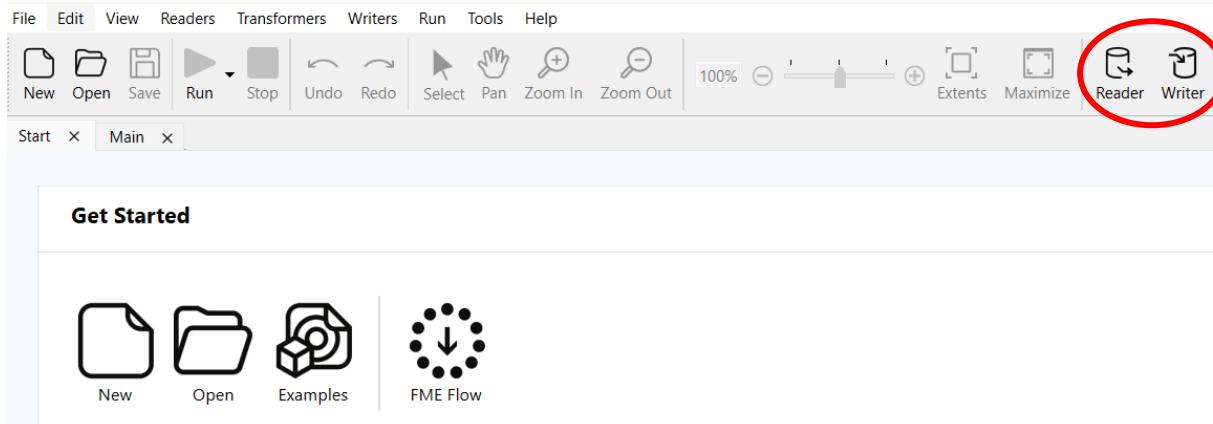
# Create Workspace from Blank



## Get Started



- Start with blank canvas
- Manually add the required **Readers and Writers**

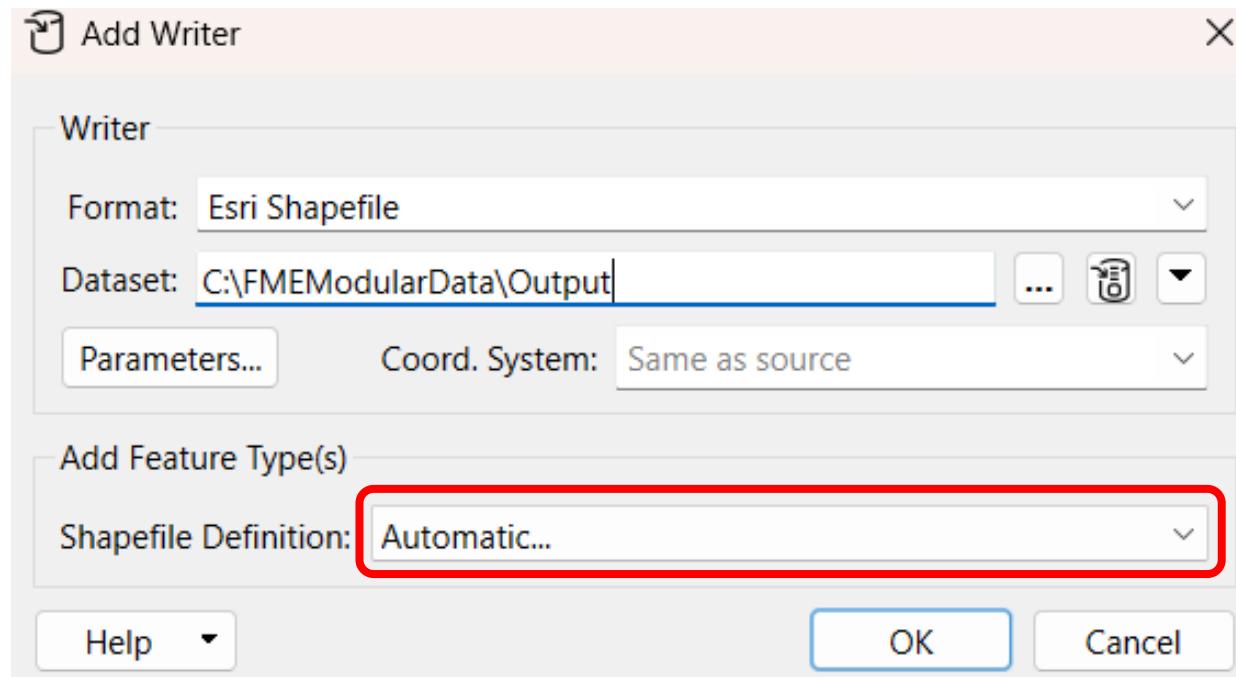


*We can actually add Readers / Writers in a number of ways!*

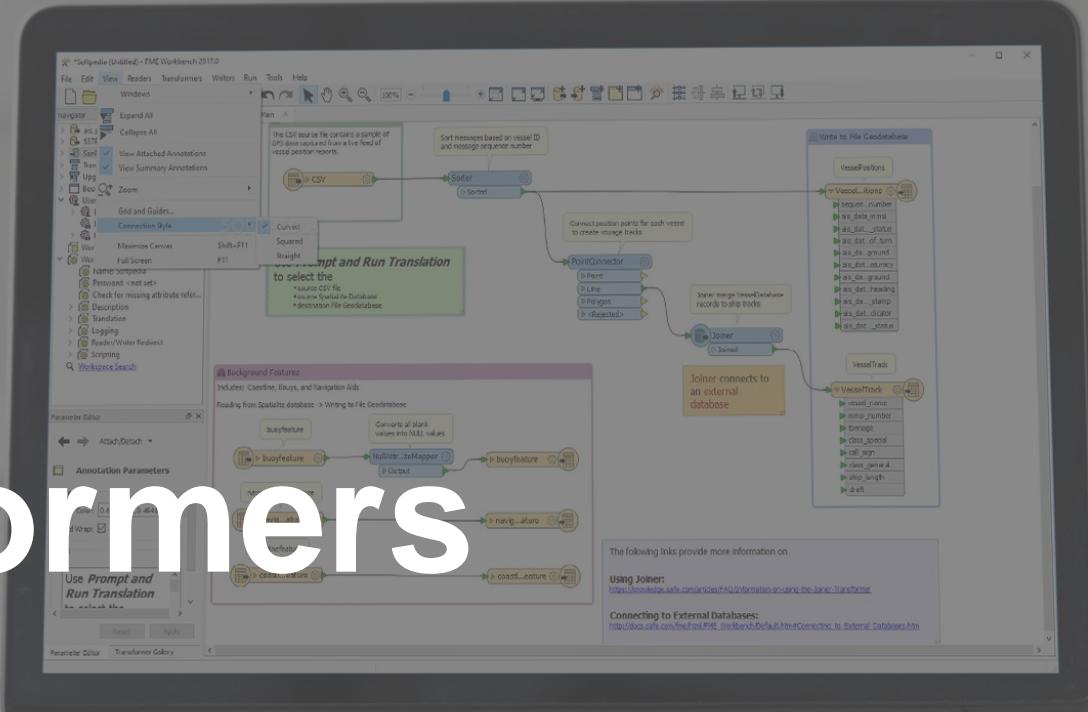
# Create Workspace from Blank



- specify **Automatic** for the Attribute/Table Definition on Writers  
*(don't worry about this for now, we look at the options in here properly in module 2)*



# Transformers



Connect. Transform. **Automate**

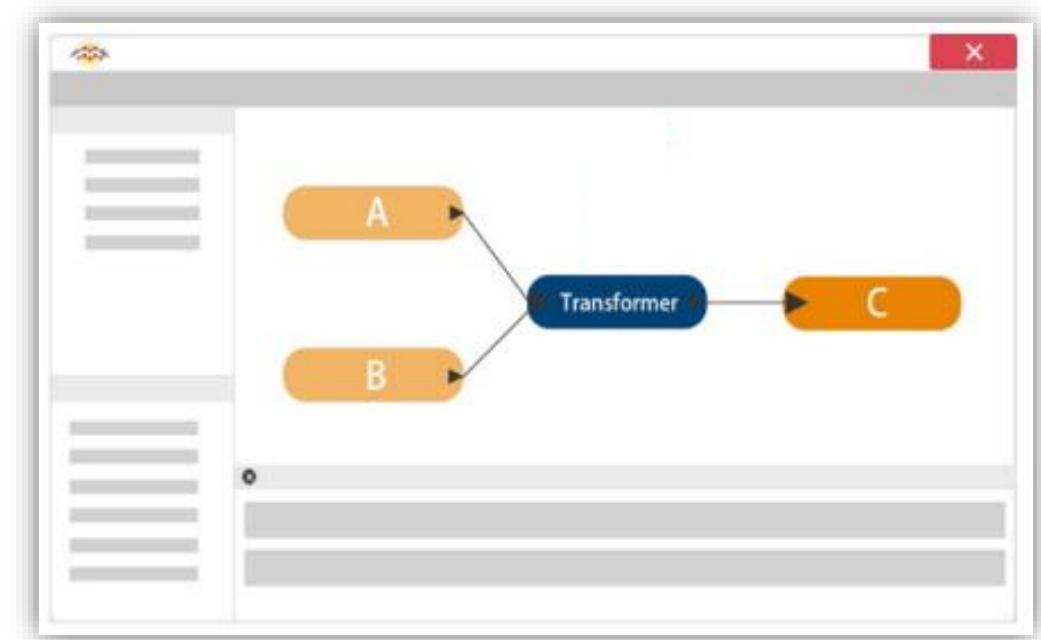
# Data Transformation



Two distinct types:

- **Structural Transformation** - changing the data model/schema; this includes the ability to merge data, divide data, reorder data, and define custom data structures: layer names, attribute names, attribute types
- **Content Transformation** - alter the content of a dataset; manipulation of a feature's geometry or attribute values

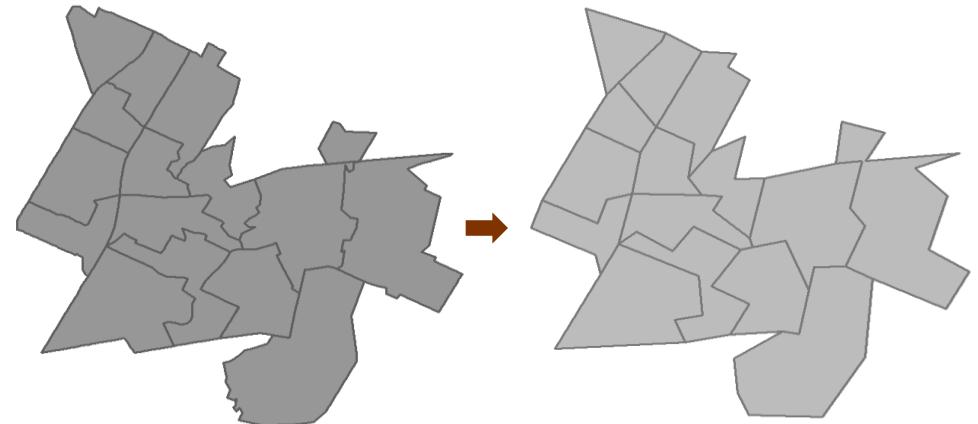
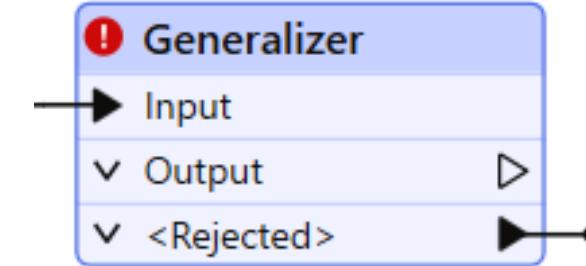
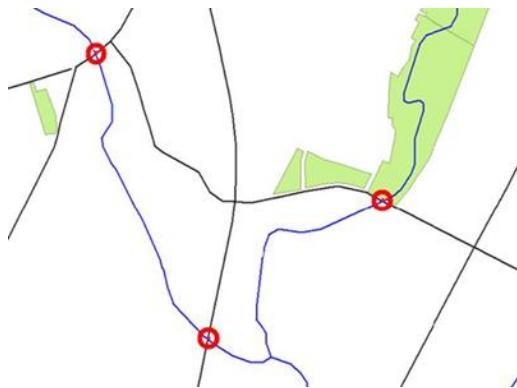
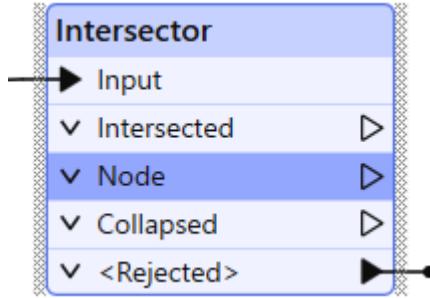
In FME you can manipulate and transform data exactly as needed by using any combination of FME's **Transformers**



# Transformer Examples



## Geometric transformation



*These Roads are intersected with Rivers  
To produce a point wherever a bridge would be*

# Transformer Examples



## Attribute transformation



Address1	Address2	City	Postcode
Rutland House	148 Edmund Street	Birmingham	B3 2FD

Concatenate

*Address1 + Address2 + City + Postcode*

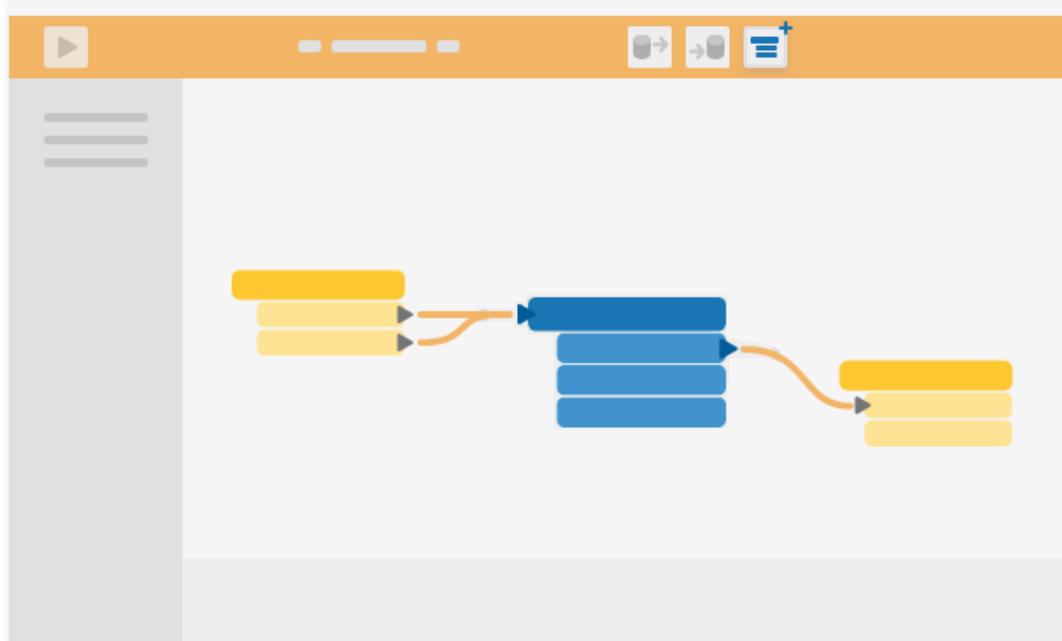
Concatenated Address

Rutland House, 148 Edmund Street, Birmingham, B3 2FD

# Transformers



There are many different Transformers to carry out different types of data operations



With over 500 transformers, FME possesses a lot of functionality, but how do you find the transformer you really need?

- Although the transformer list can look a bit overwhelming, don't panic!
- The reality is that most users focus on 20-30 transformers.
- You don't need to know every single transformer to use FME effectively.

# Locating transformers



**FME Transformer Gallery**

Transformers are the building blocks to your FME data transformation workflows. Each transformer has a specific function, and when combined, can be used to create powerful integration workspaces.

Help Me Choose a Transformer

Select a Data Type  — or — Enter a Transformer

Clear Filters

Browse by Keyword

Showing 1 - 20 of 500 Transformers

Sort by Rank

Transformer	Rank
AttributeManager	1
Tester	2
AttributeCreator	3
FeatureMerger	4

Transformer Categories

- 3D
- Attributes
- Calculated Values

Clipper

Official Publisher - Safe Software

Category Spatial Analysis

Rank 33

Link [Help](#)

Description

Divides Candidate features using Clipper features, so that Candidates and parts of Candidates that are inside or outside of the Clipper features are output separately. Attributes may be shared between objects (spatial join).

Related Transformers

Transformer	Transformer
AreaOnAreaOverlayer	PointOnAreaOverlayer
Bufferer	PointOnLineOverlayer
Intersector	PointOnPointOverlayer
LineOnAreaOverlayer	SpatialFilter
LineOnLineOverlayer	ContainDetector

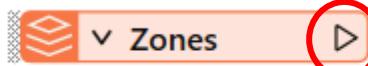
Transformer Gallery, Transformer Categories, Quick Add, Transformer Help, Transformer Searching, FME HUB

# Connecting Transformers



- Quick Connect

- Click on any output port to be connected
- Click again on the desired input destination port

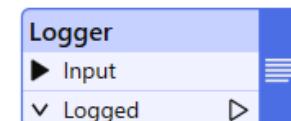


- Drag and Insert

You can "drag and insert" a transformer onto an existing connection. The connection will turn green.



- Manual Connect

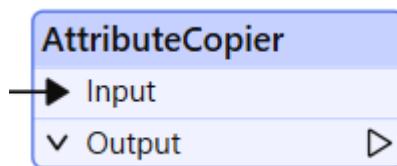


# Transformer Parameter Status



Every transformer contains at least one parameter

The properties button on a transformer is colour-coded to reflect the status of its parameters:



**Black** indicates that the default transformer parameters have been checked and amended as required, and the transformer is ready to use



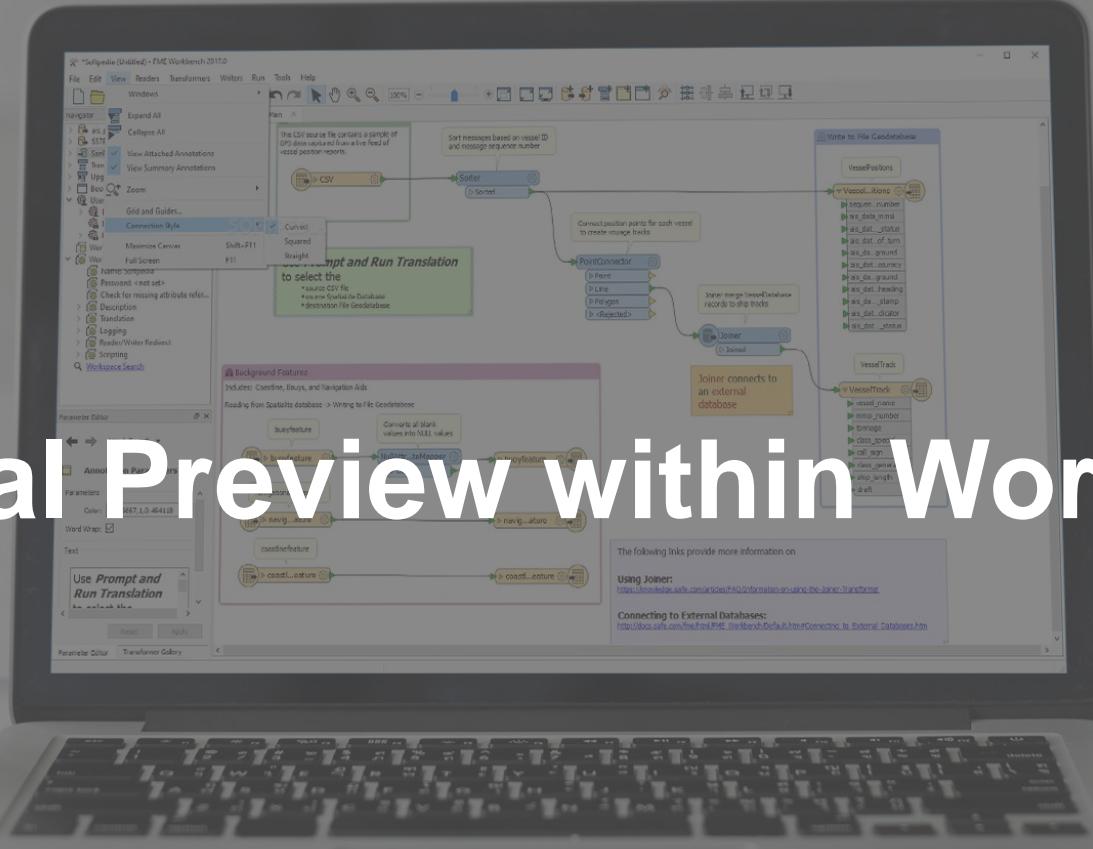
**yellow** indicates that the default parameters have not yet been checked. You can use a transformer that is in this state, but the workspace results may be unpredictable



**red** indicates that there is at least one setting for which FME cannot supply a default value. You must provide a value for the required parameter(s) before you can use the transformer

# Data Visual Preview within Workbench

Connect. Transform. **Automate**

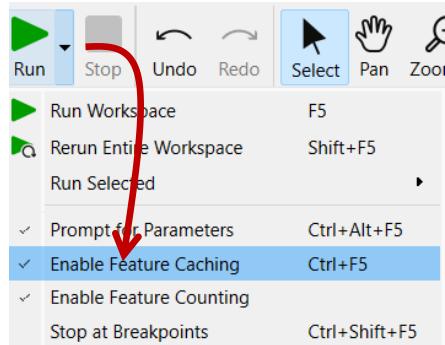


# Feature Caching



Feature caching is the process of storing data to allow it be easily accessed when it is required.

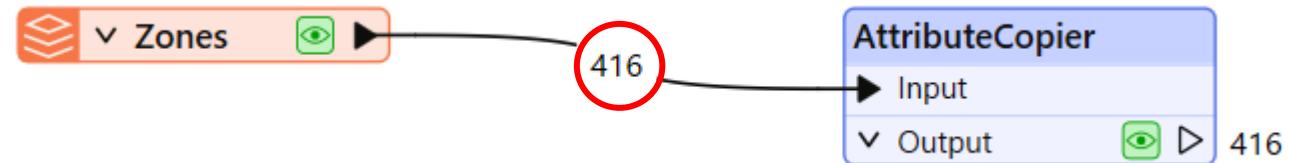
FME workspaces can be run with or without feature caching.



Once activated FME will store your data at each stage of your workflow.

Pros and cons:

- Your workspace will take longer to run
- Helps when building, testing and debugging workspaces
- Allows partial runs



# Workbench Visual Preview

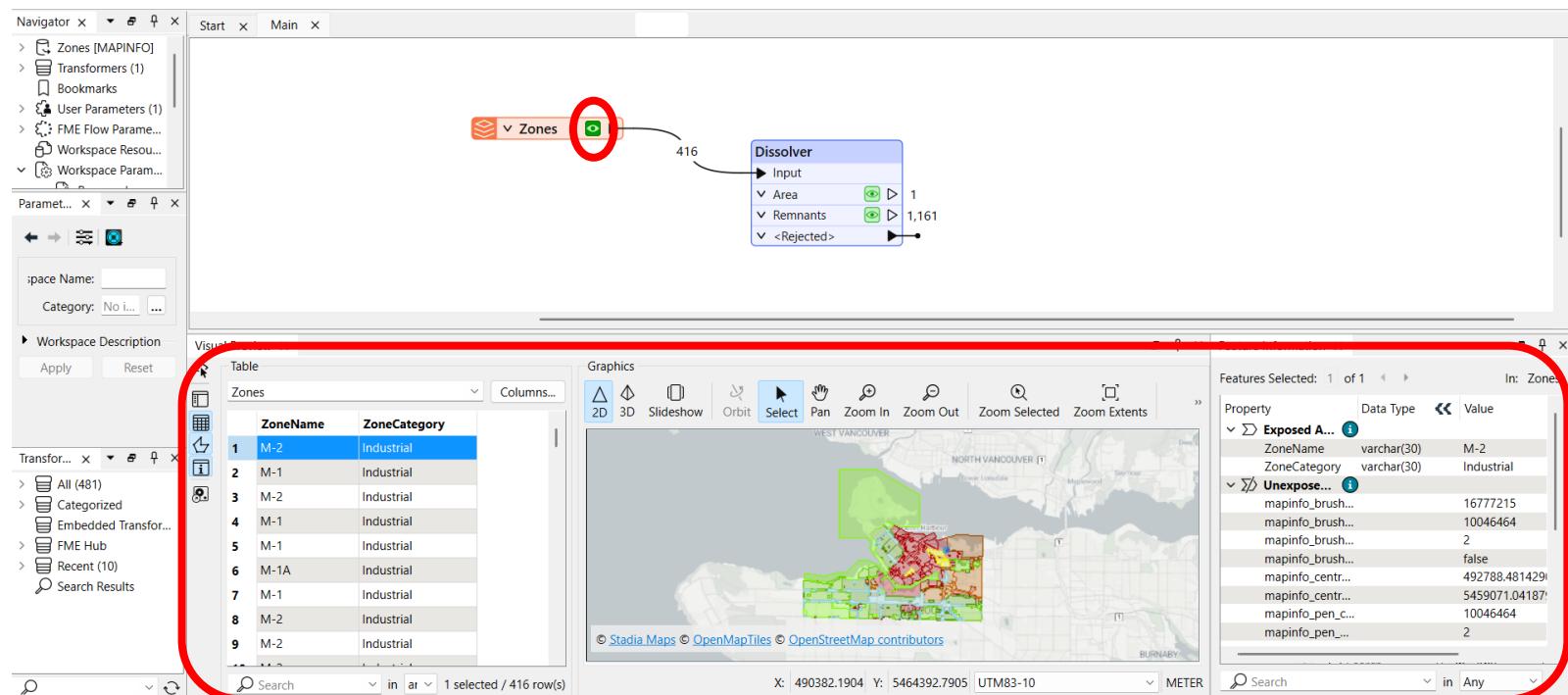


Within the Workbench the Visual Preview window contains tools for examining features at each point in the process.

Click on the ‘click to inspect cached features’ icon



- Automatic Inspect on Selection
- Display Control
- Table View
- Graphics View
- Feature Information window
- Open in Data Inspector



# Demo – New Workspace from Blank, Transformer Use and Visual Preview



- Create the workspace from Blank
  - Add Reader - MapInfo TAB, Zones
  - Set Attribute definition to Automatic
  - Manually add Connector line
- Use Visual Preview to examine the Zones data
  - the ZoneCategory value of ‘Comprehensive Development’ is going to be changed to ‘Development’
- Find and add Transformer
  - StringReplacer transformer - find it
  - Connect into workflow
  - Set transformer parameters
    - Attributes: ZoneCategory
    - Text to Replace: ‘Comprehensive’
    - Replacement text: *blank*
- Use Visual Preview to examine transformer output
- Add Writer - CSV



# Exercise 1.2



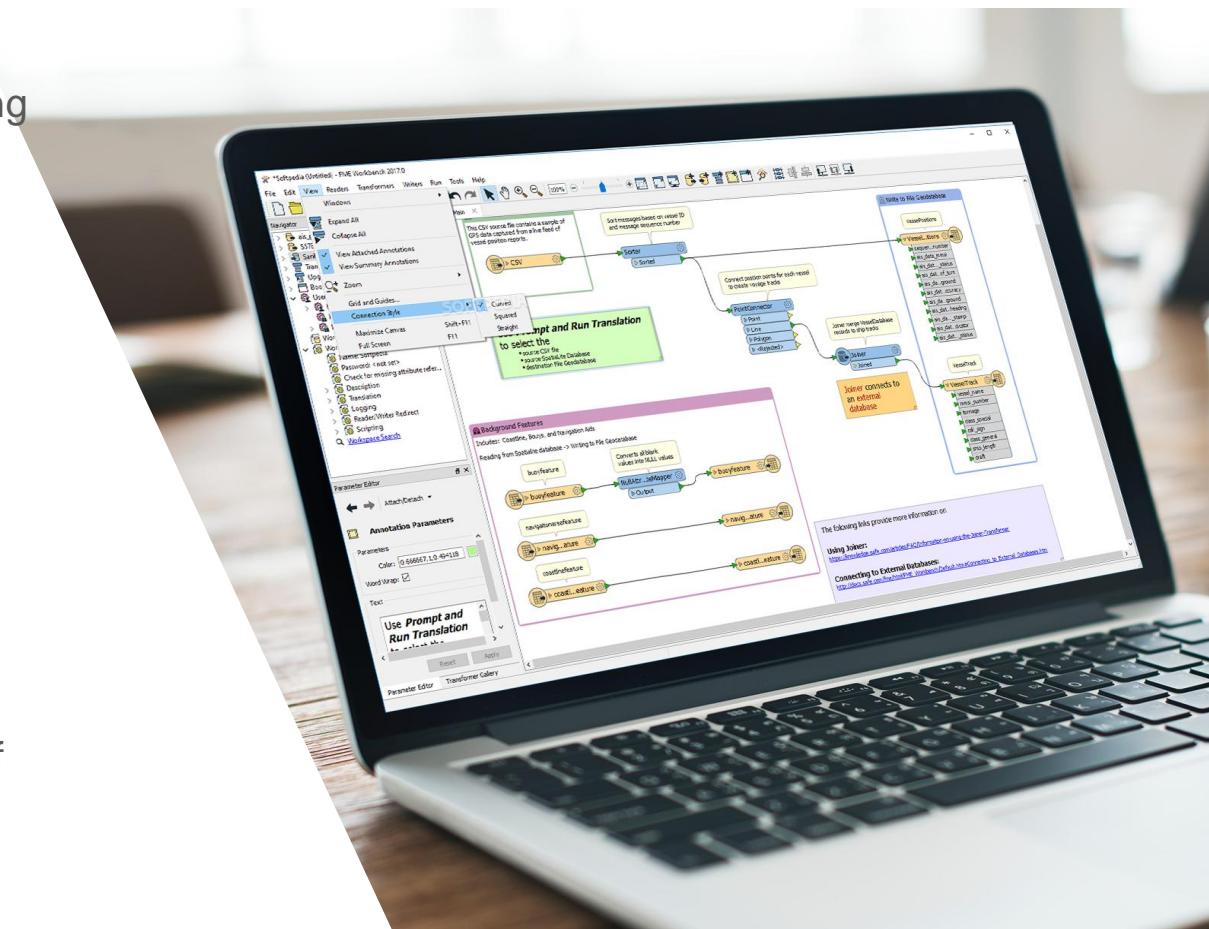
## Create a Workspace using Visual Preview and Using Transformers

- In the previous exercise, we inspected and translated some zoning data.
- Now we are going to create a new workspace (from a blank canvas) and use a transformer to merge the zones into new polygons based on their zone category.

### Data - Zoning Data (MapInfo TAB)

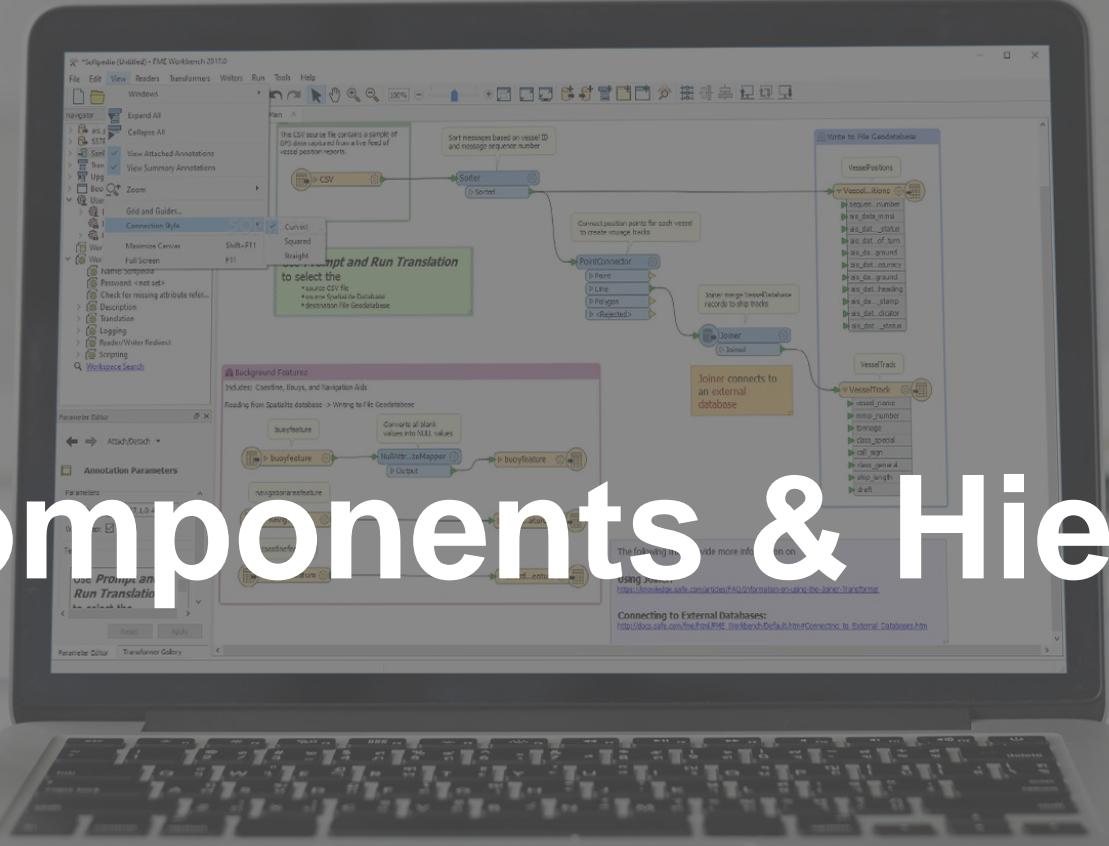
#### Goal :

- create a new workspace from a blank canvas, setting up required reader and writer.
- use a transformer to perform data transformation.
- use the Visual Preview tools to examine features at each stage of the workflow.



# FME Components & Hierarchy

Connect. Transform. **Automate**



# FME Components and hierarchy



The screenshot shows the FME Workbench 2024.2 interface with several components highlighted:

- Readers & Writers**: A red box highlights the "Readers" and "Writers" sections in the top menu bar.
- Feature Types**: A red box highlights the "Parks" and "city\_parks" feature types in the Navigator panel.
- Translation Log**: The bottom panel displays the Translation Log with 10 Errors, 5 Warnings, and 1 Information message. It shows the process ID, peak memory usage, and a note about renamed attribute names due to format constraints.

**Navigator Panel (Left):**

- Parks [MAPINFO]
- city\_parks [SHAPEFILE]
- Transformers
- Bookmarks
- User Parameters (2)
- FME Flow Parameters
- Workspace Resources
- Workspace Parameters
  - Password: <not set>
  - Name: <not set>
  - Description
  - Logging
  - Reader/Writer
  - Scripting
  - Translation
- Workspace Search...

**Main Panel (Center):**

- Parks
- city\_parks
  - ParkId
  - RefParkId
  - ParkName
  - Neighborhood
  - VisitorCount
  - TreeCount
  - DogPark
  - Washrooms
  - SpecialFeatures

**Translation Log (Bottom):**

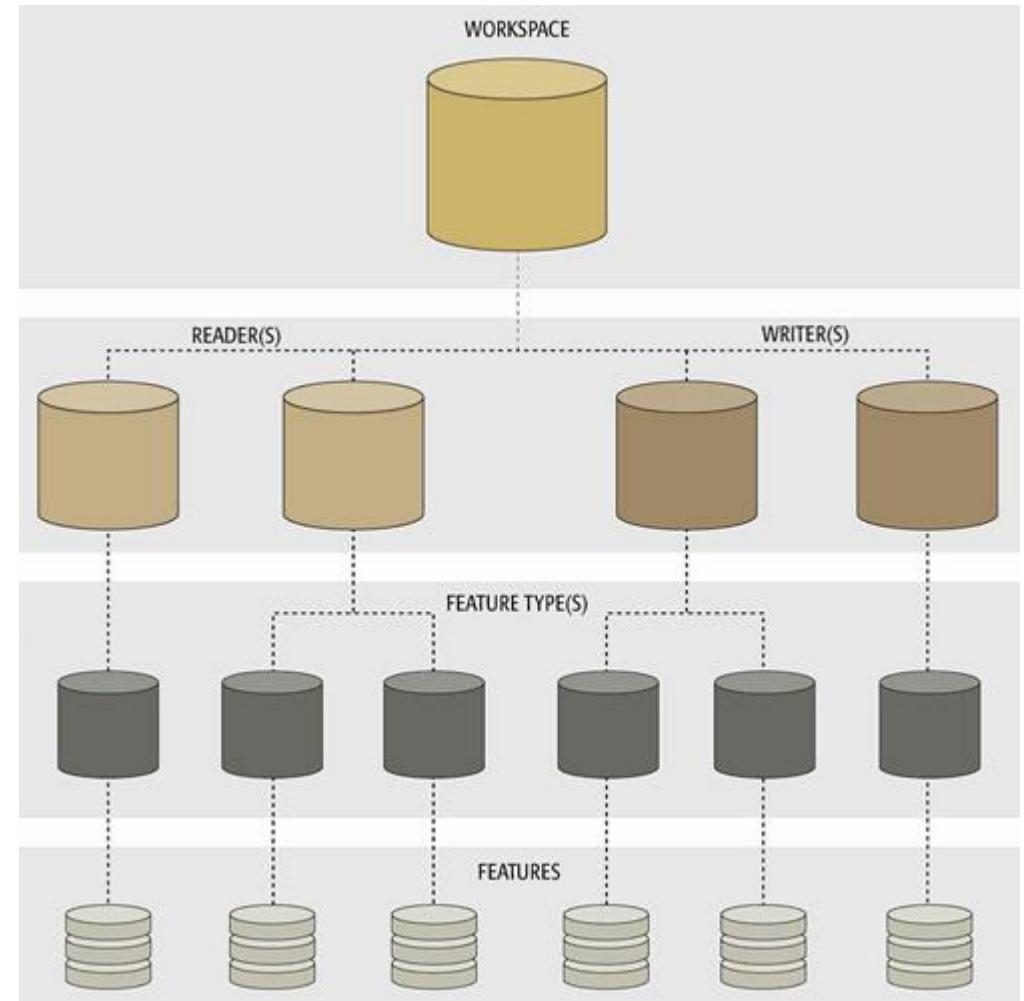
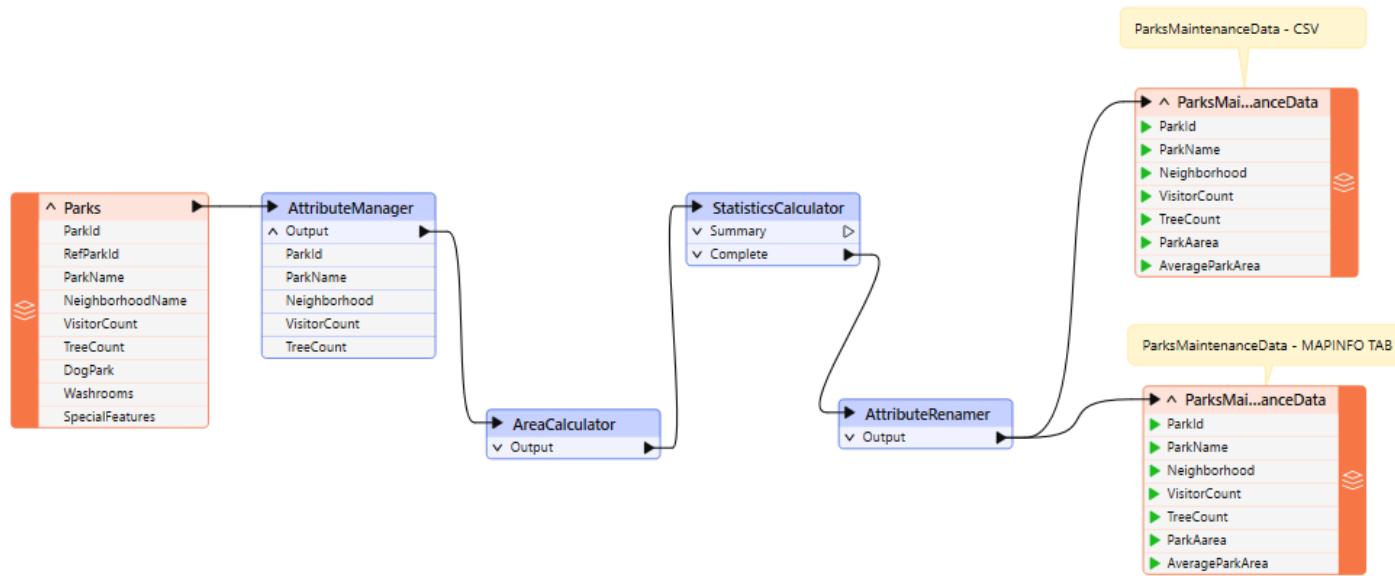
Transformer	Message
36	END - ProcessID: 11332, peak process memory usage: 49432 kB, current process memory usage: 49432 kB
37	SHAPEFILE Writer: Following Attribute name(s) were renamed based on format constraints for invalid characters...
38	- Renamed 'NeighborhoodName' to 'Neighborhood'
39	- Renamed 'SpecialFeatures' to 'SpecialFeatures'
40	- Renamed 'VisitorCount' to 'VisitorCount'

# Translation components



The key components in an FME translation:

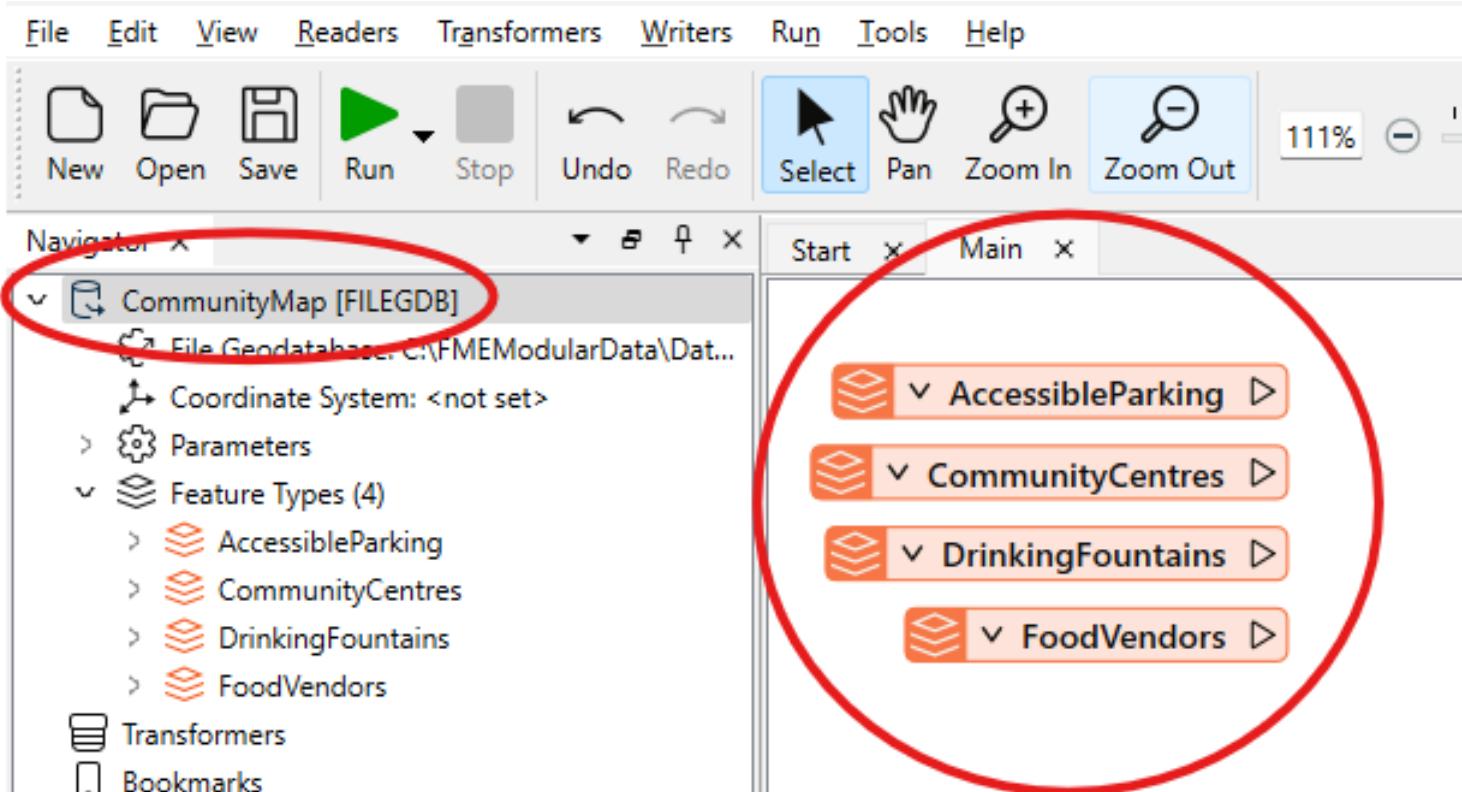
- Workspace
- Readers and Writers
- Feature Types
- Features



# Readers, Writers and their Feature Types

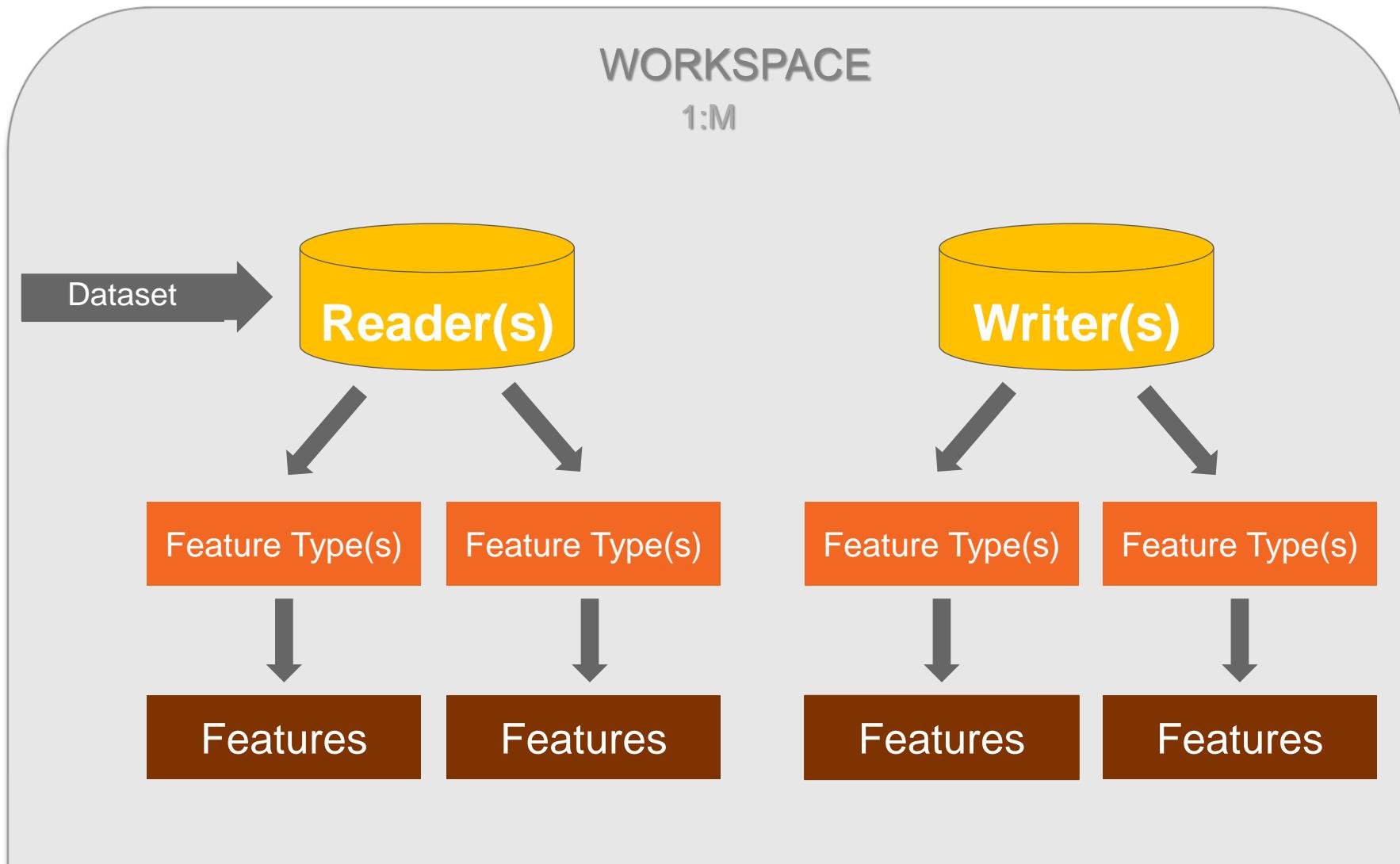


Each Reader/Writer will have one or more Feature Types

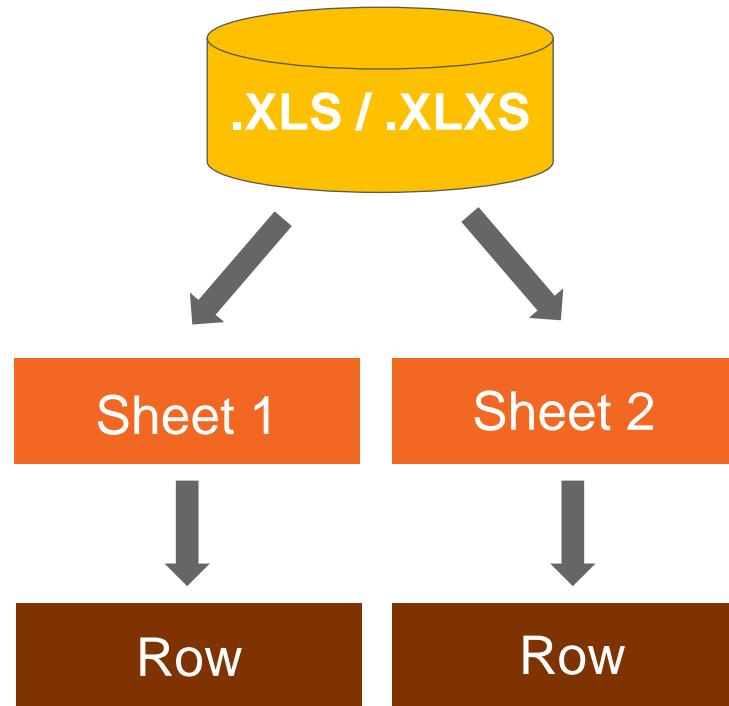


This ESRI Geodatabase  
Reader has four Feature Types

# FME components and hierarchy



# EXAMPLE: Excel



Reader / Writer

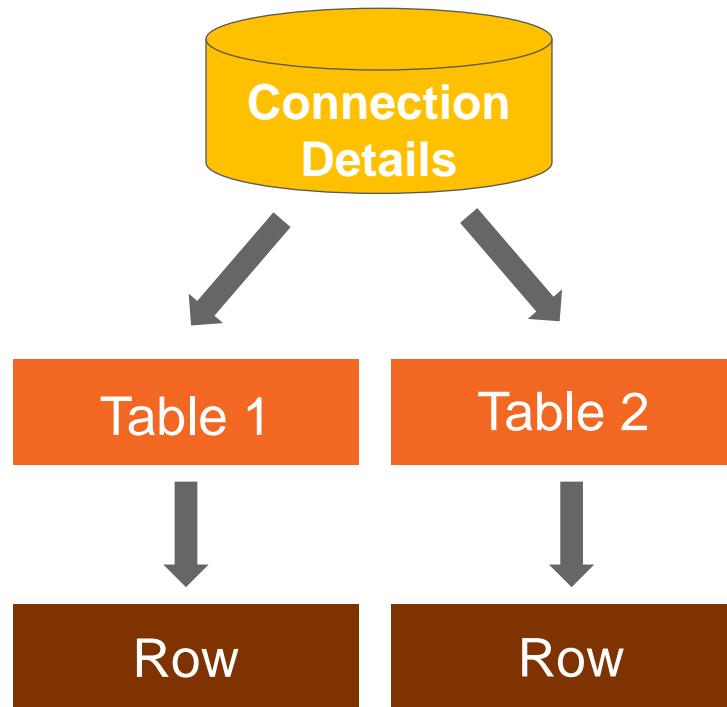
Feature Type

Features

PublicArt.xlsx -

	B	C	D	E
1	Title	Longitude	Latitude	
2	The Belonging Action	-123.1100977	49.28378068	
3	The Belonging Action	-123.110029	49.28373217	
4	China Gate	-123.1032823	49.27975613	
5	Moving Pictures	-123.1250401	49.27705517	
6	Suan Phan: Abacus	-123.1055799	49.27977613	
7	Untitled (light work)	-123.1180593	49.28855627	
8	Untitled (light work)	-123.1178331	49.28839592	
9	Scopes of Site	-123.127174	49.29058951	
10	Untitled	-123.1128704	49.28022166	
11	Untitled	-123.1128704	49.28022166	
12	Untitled	-123.1178337	49.28573477	
13	Untitled	-123.1182869	49.28571828	
14	Untitled	123.127174	49.29058951	

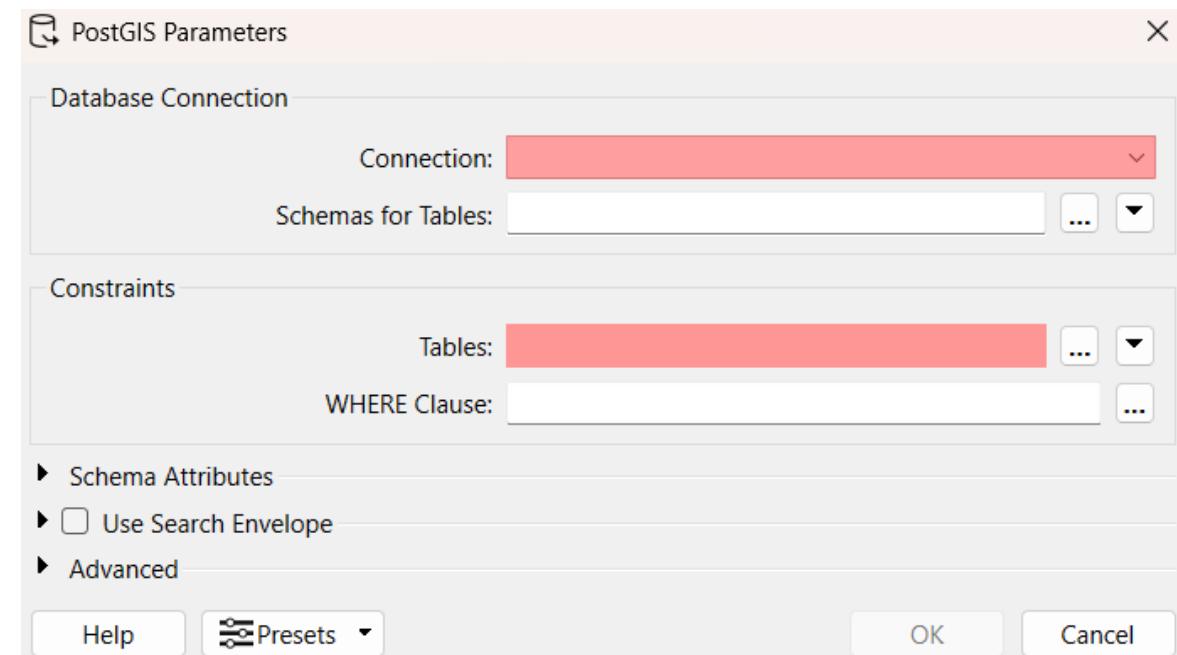
# EXAMPLE: Database



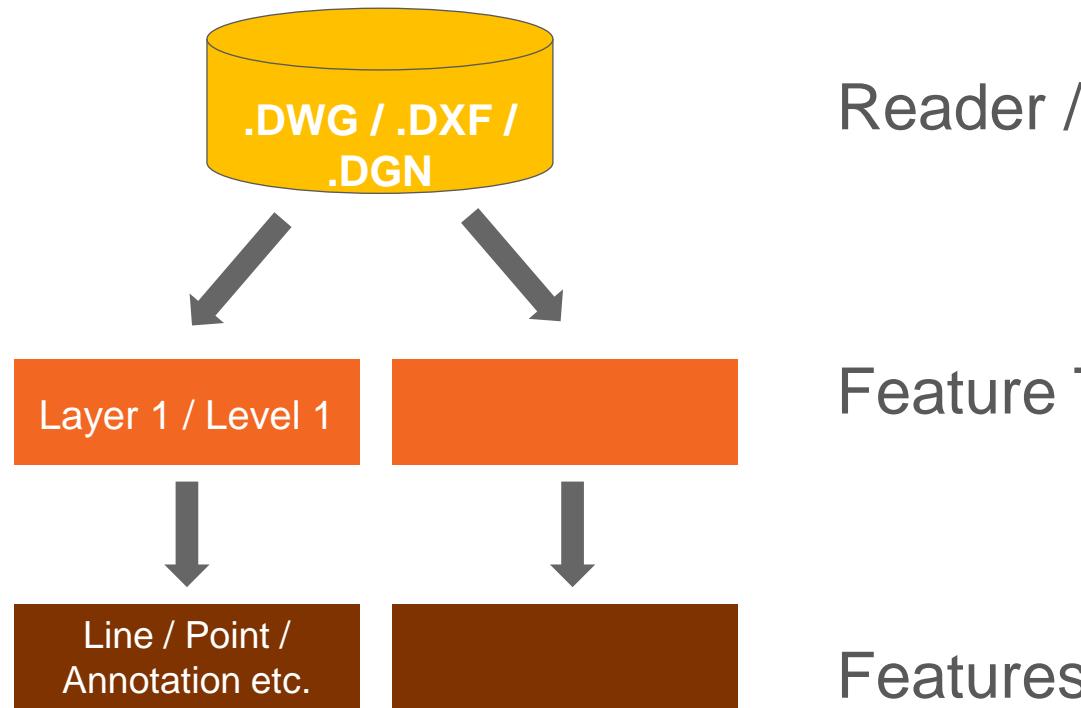
Reader / Writer

Feature Type

Features



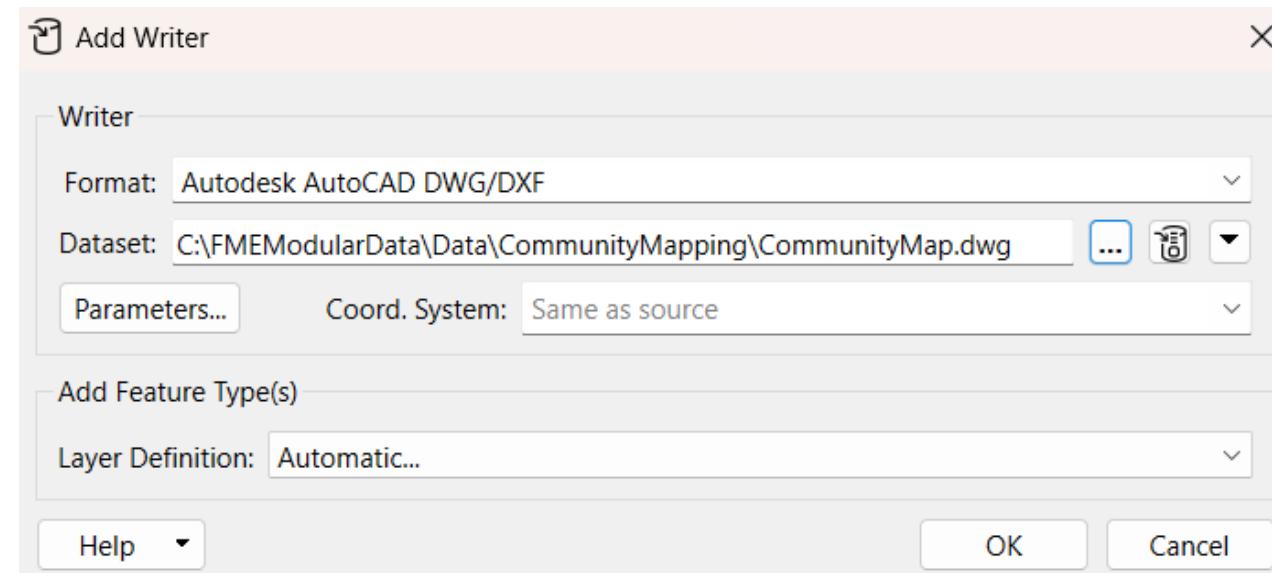
# EXAMPLE: CAD files



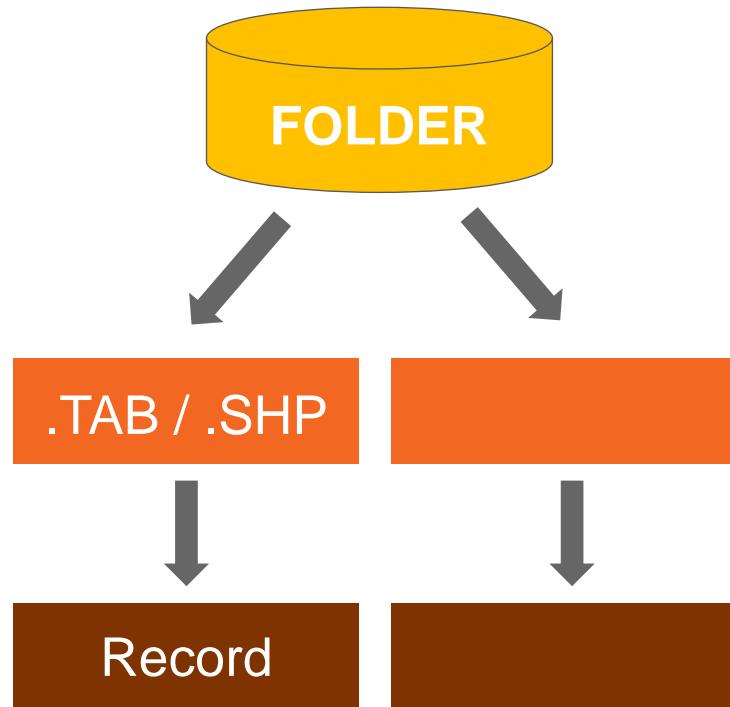
Reader / Writer

Feature Type

Features



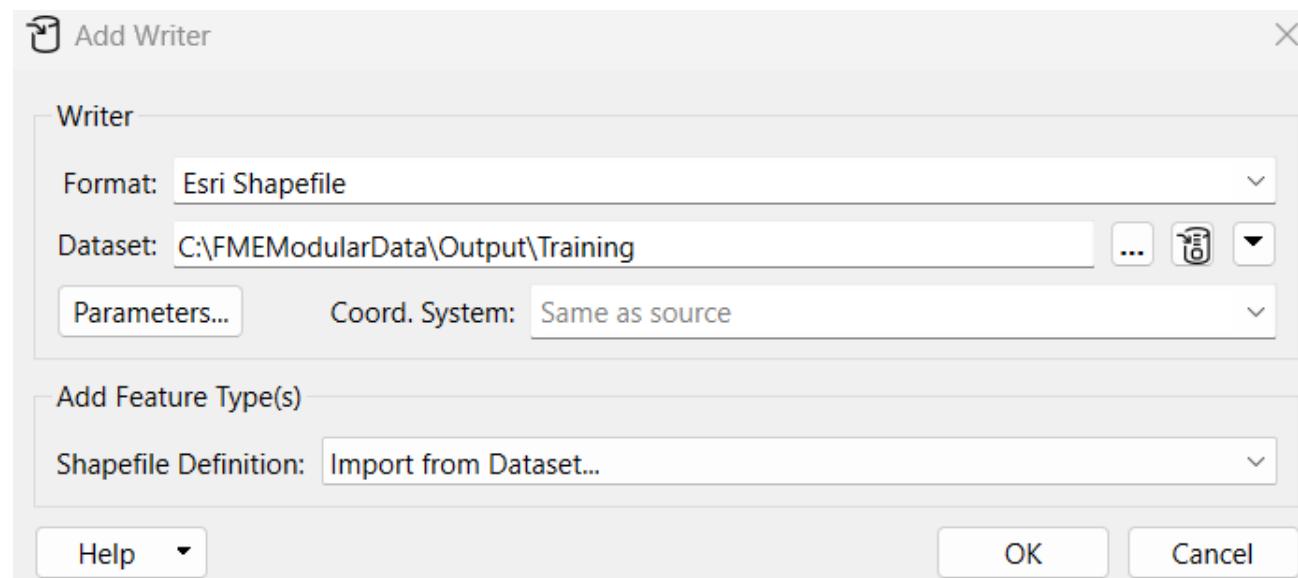
# EXAMPLE: MapInfo TAB, ESRI SHP, CSV



Reader / Writer

Feature Type

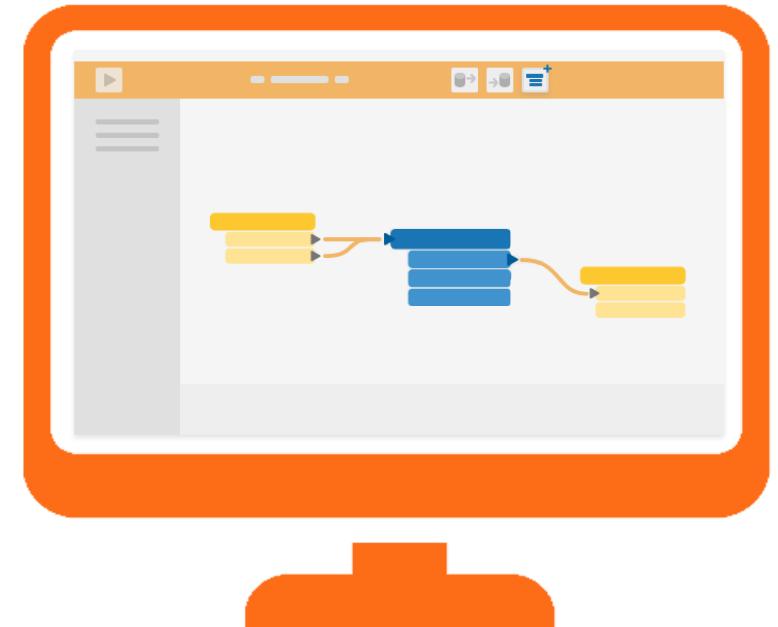
Features



# Demo – Readers/Writers and Feature Types



- Data Inspector:
  - Open the ‘CommunityMapping’ ESRI Geodatabase
  - Examine the Feature Types & Features
  
- Workbench:
  - Add a Reader for ‘CommunityMapping’ ESRI Geodatabase
  - Examine Reader in Navigator panel
  - Examine Feature Types on Canvas



# File based v Folder based datasets

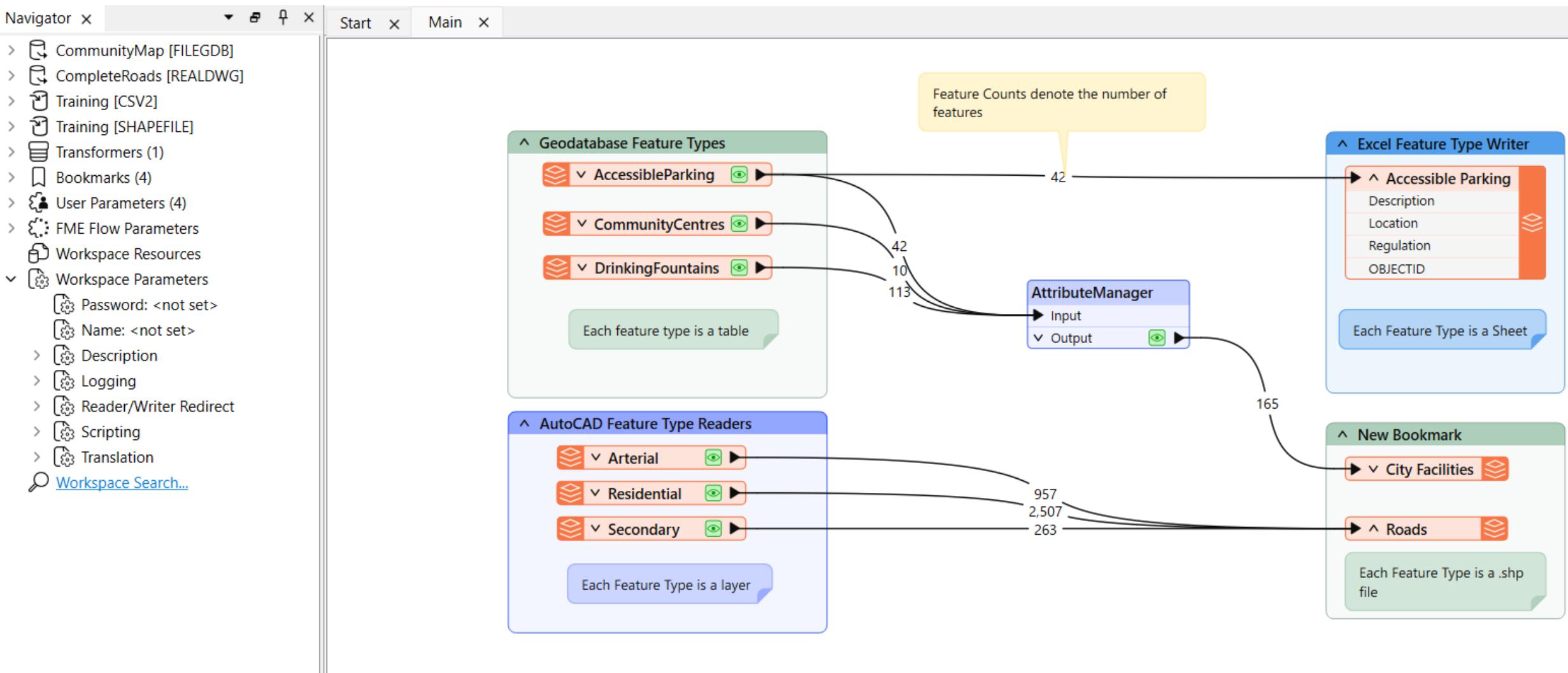


	ESRI SHP	MapInfo TAB	CSV	AutoCAD	KML	OGC GeoPackage	Excel	ESRI Geodatabase	Database	Raster
Reader/Writer	Folder	Folder	Folder	DWG File	KML File	GPKG File	XLS File	GDB File	Connection Details	Raster Folder
Feature Type	SHP File	TAB File	CSV File	Layer	Layer	Layer	Worksheet	Dataset/Table	Table	Raster File
Feature	Record	Record	Row	Object	Record	Record	Row	Record	Row	Raster File

# Multiple Readers & Writers



A workspace with multiple readers and writers might look like this



# Exercise 1.3

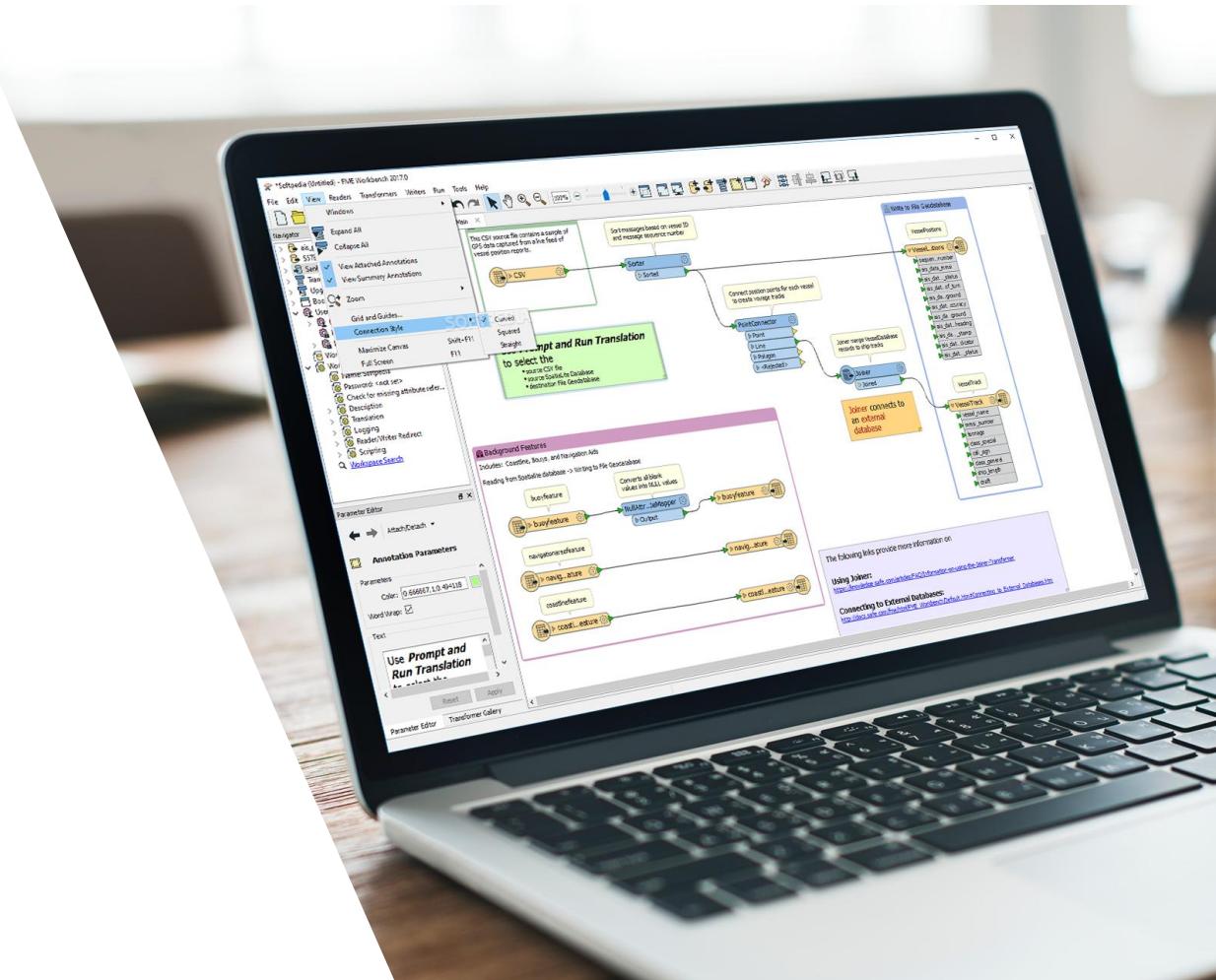


## FME Components and Hierarchy

- Understand the differences in FME hierarchy regarding the Readers, Writers and Feature Types of different formats
- Learn the difference between folder based and file based formats

**Data - CommunityPlanMap (Bentley MicroStation Design V8)**

**Goal - Create a workspace to translate from DGN to ESRI SHP, working with multiple feature types**



Thanks for watching!

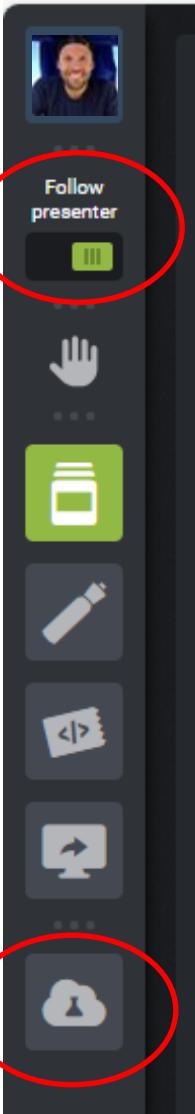
Any questions?



# FME Form Training - Module 2

Managing Attributes and Filtering Features

# Training Environment



< keep 'Follow presenter' on

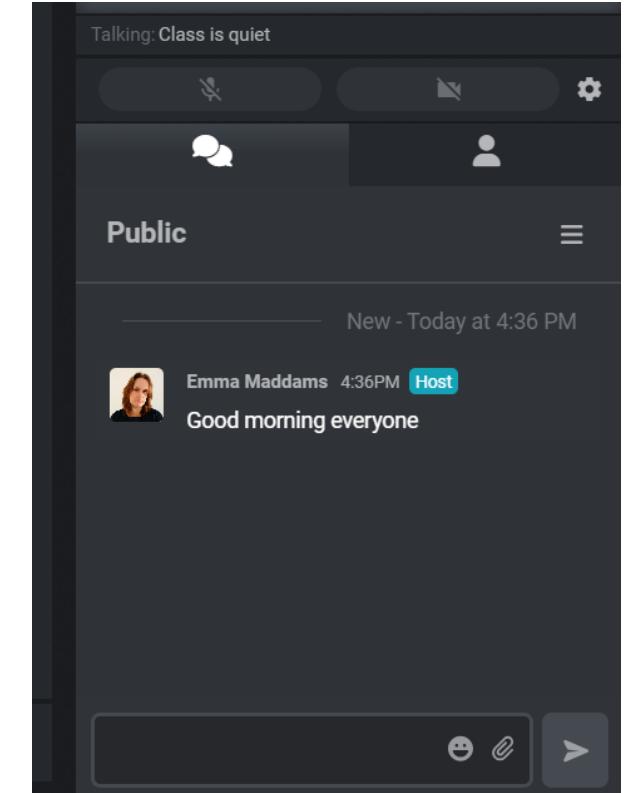
< Lab

need help when using your Lab  
Use either:

- 'Raise hand' or
- 'Lab Assistance'

## Chat

- to everyone
- to trainer



# Training Environment

---



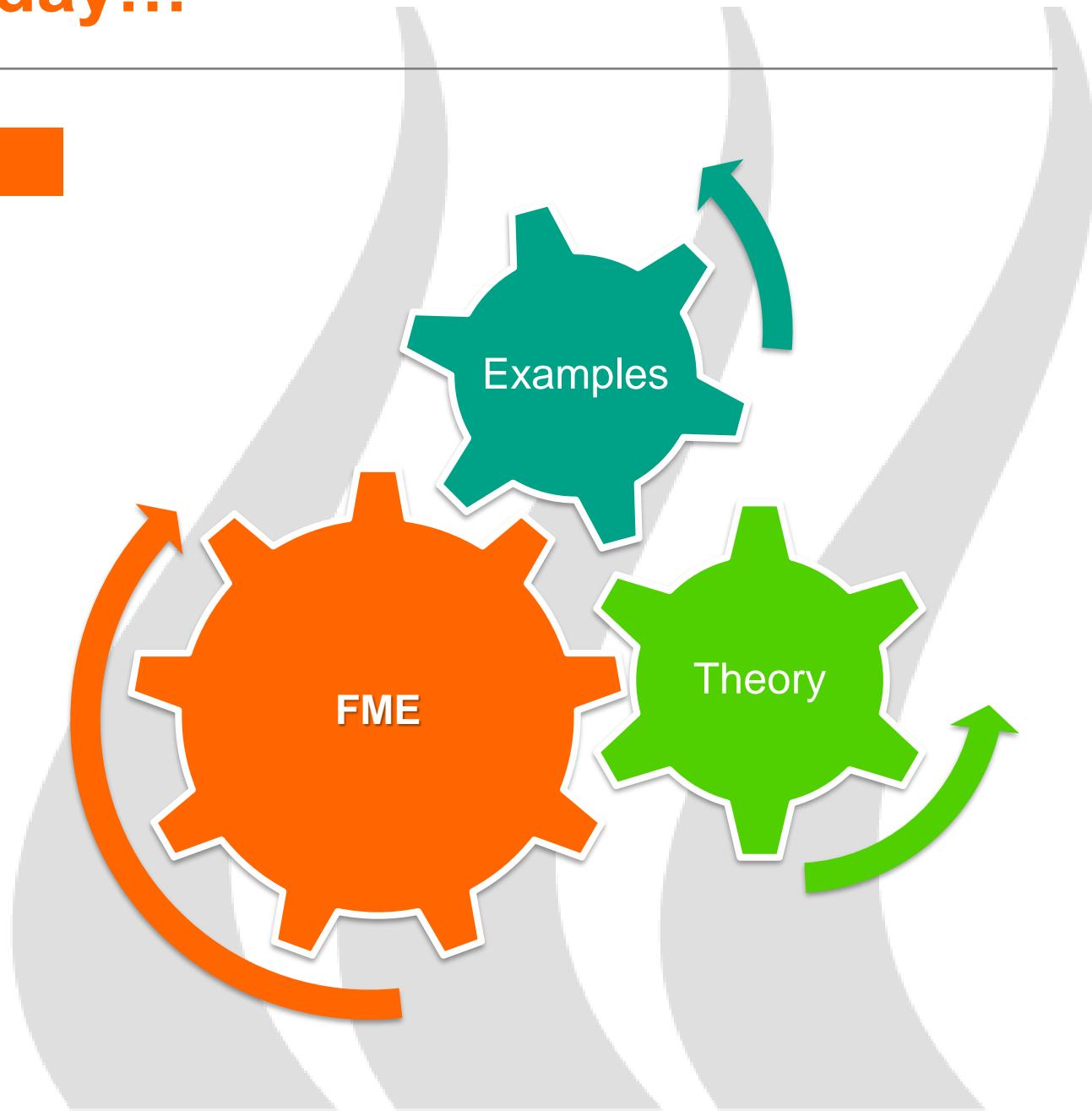
- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

# What we'll be covering today...

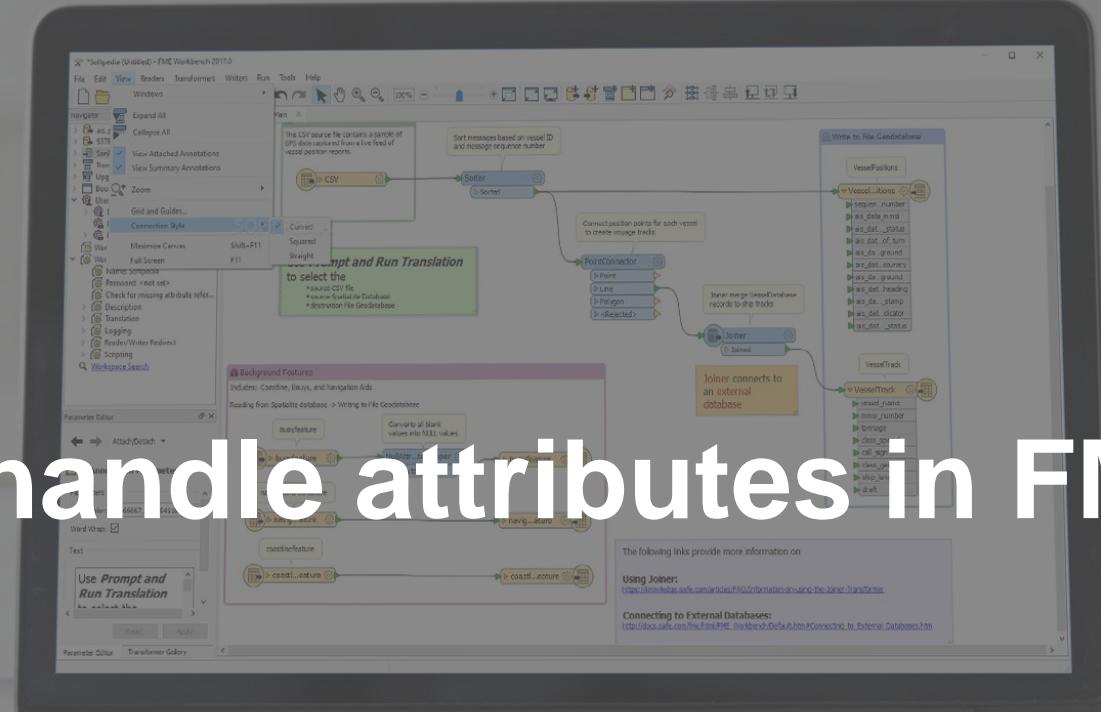
---

## Agenda

- Schema Mapping
- Managing Attributes
- Format Attributes
- Lists
- Filtering Features
- Key-based Joins



# How do I handle attributes in FME?

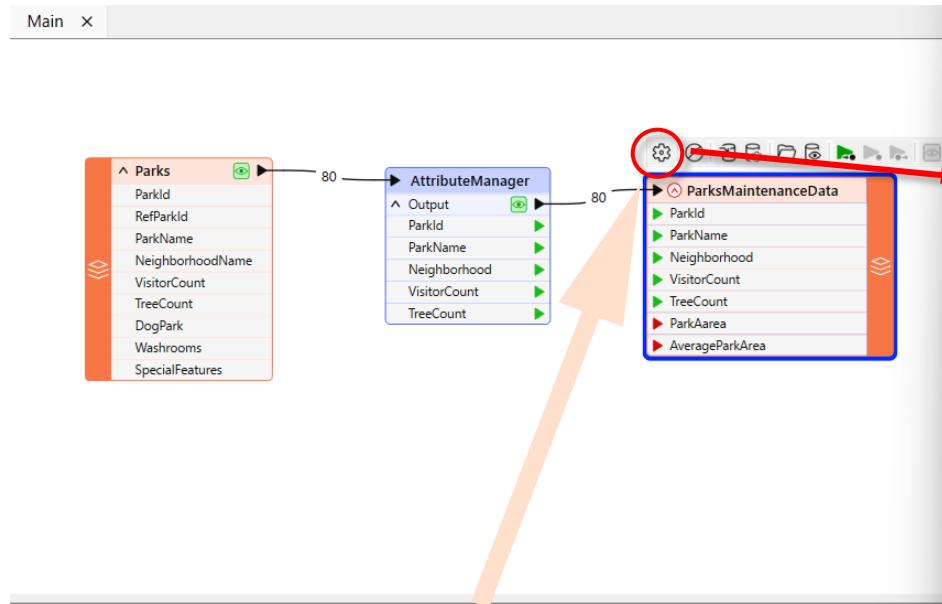


Connect. Transform. **Automate**

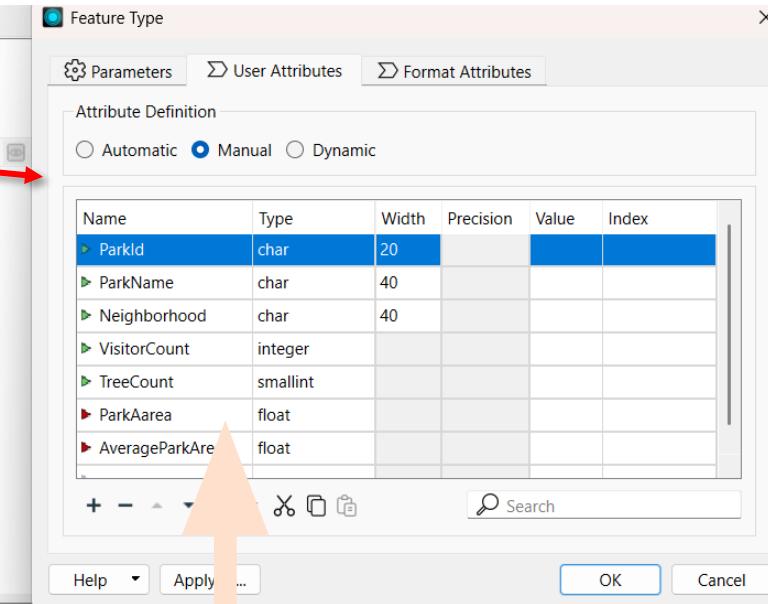
# Schema Mapping



*Reader Feature Type Attributes*  
– ‘What We Have’



*Writer Feature Type Attributes*  
– ‘What We Want’



We can manage attribute mapping  
using **Transformers**

- Rename attributes
- Create attributes
- Remove attributes
- Order attributes

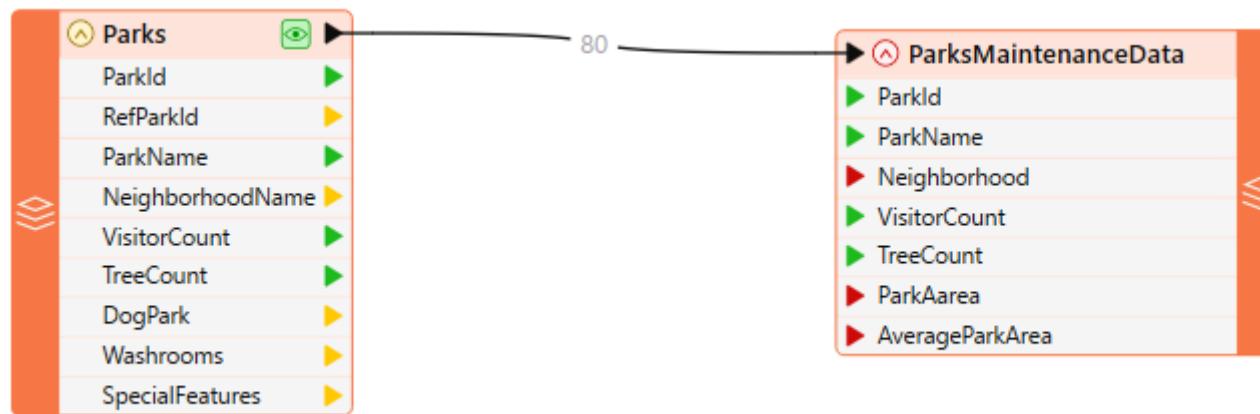
We can edit Writer **Schema**

- Attribute naming
- Attribute order
- Add/Remove attributes
- Attribute Data Types

# Attribute Routing



- Attributes are **name-driven** through the workspace
- Attribute names are **Case Sensitive**
- Attribute port colours - indicate if the attribute is being successful routed, or not:



Green – the attribute is being successful routed out/in

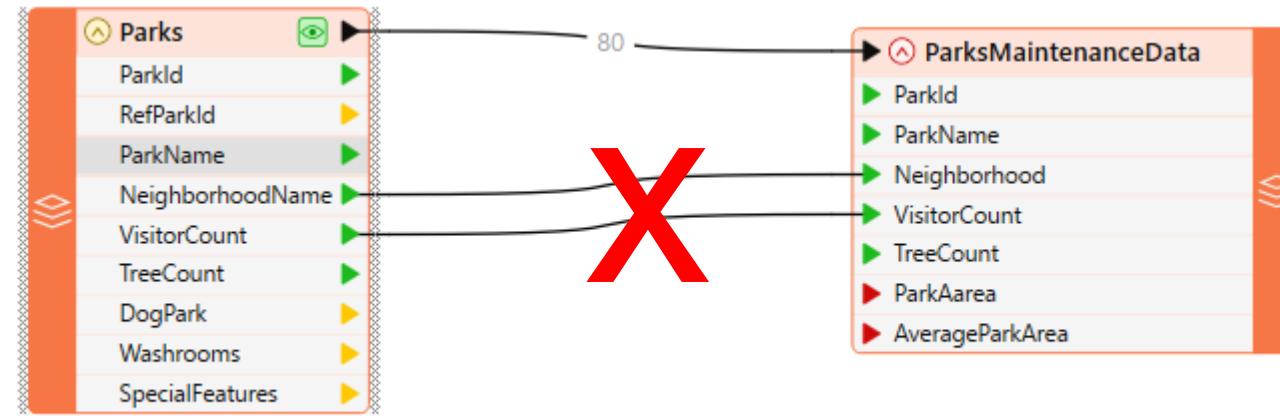
Yellow – the attribute is not being routed beyond this point of the workspace

Red – the attribute has no input. It will be an empty attribute on written data

# Attribute Routing



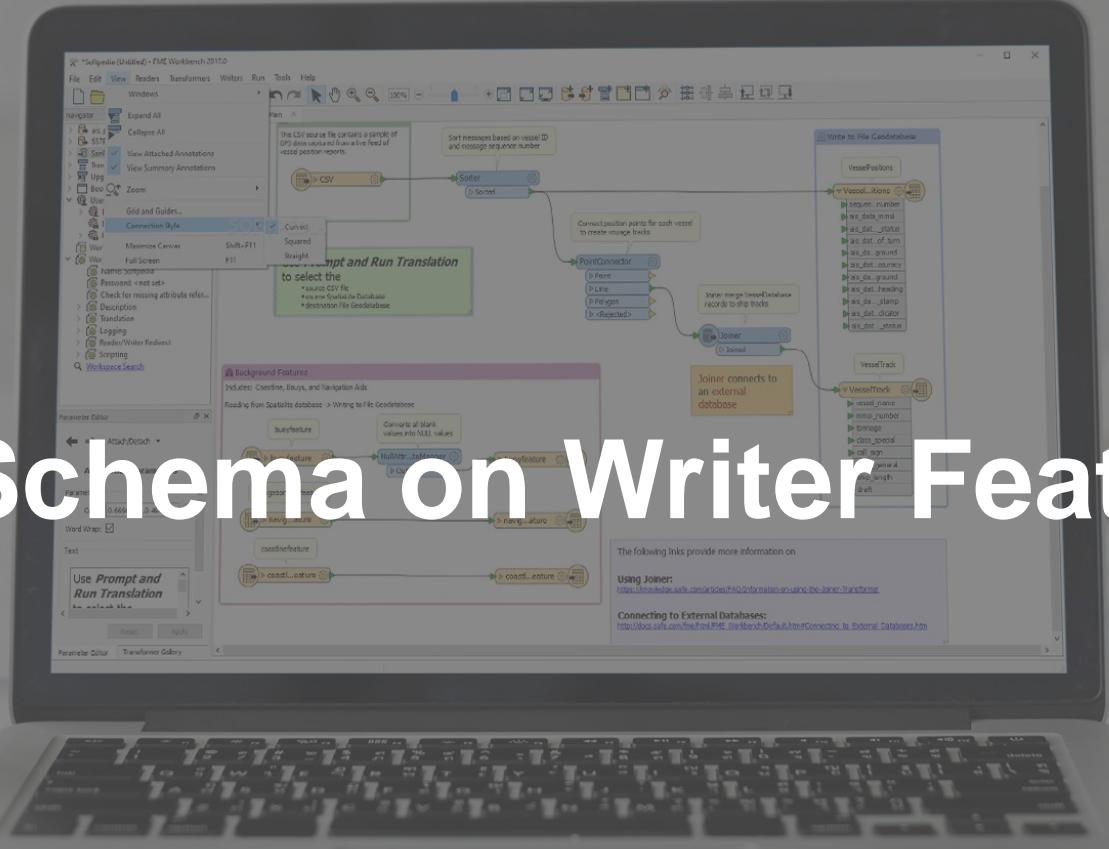
It is possible to manually connect attributes, but **don't** - it's bad practice and these connector lines can easily be corrupted if changes are made to the workflow or source data. Leading to attributes being incorrectly connected and routed!



Instead, **use Transformers** to manage your attribute routing

# Defining Schema on Writer Feature Type

Connect. Transform. **Automate**

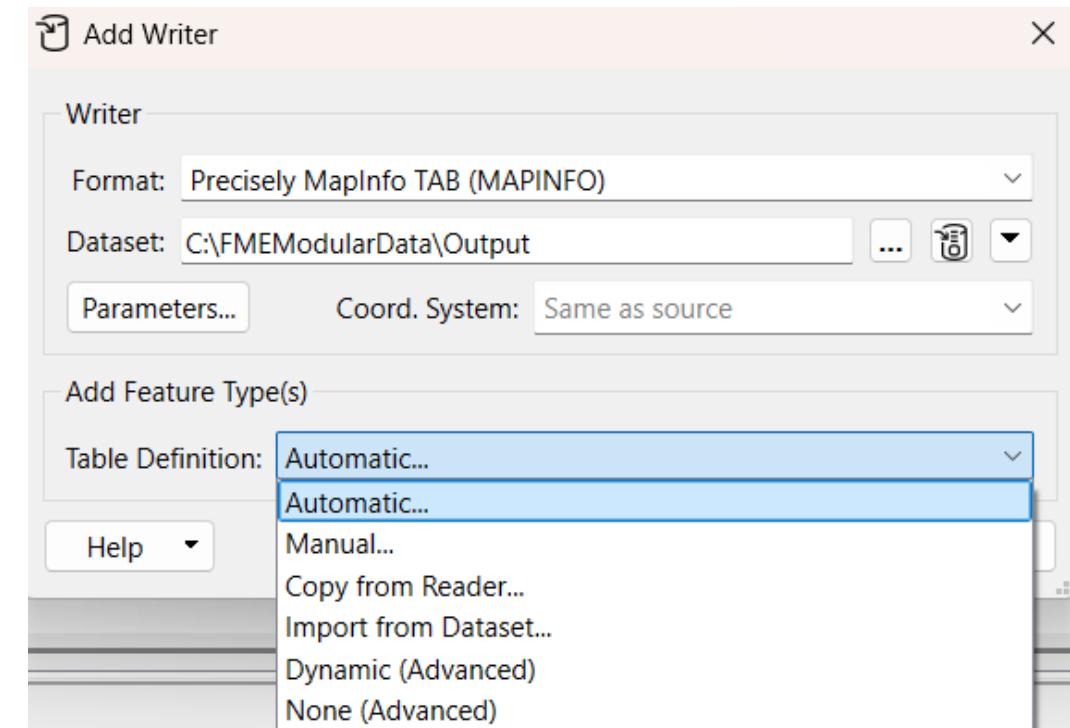


# Attribute Definition – When Adding Writer



The attributes on your Writer Feature Types can be defined in a number of ways:

- Copy from a Reader, another Feature Type or Transformer
- Manually
- Automatic
- Dynamic (advanced)



These options are available when **adding a Writer**

But we can also switch between Automatic/Manual later if we change our mind!  
- *in the Writer Feature Types parameters*

# Schema editing on the Writer Feature Type



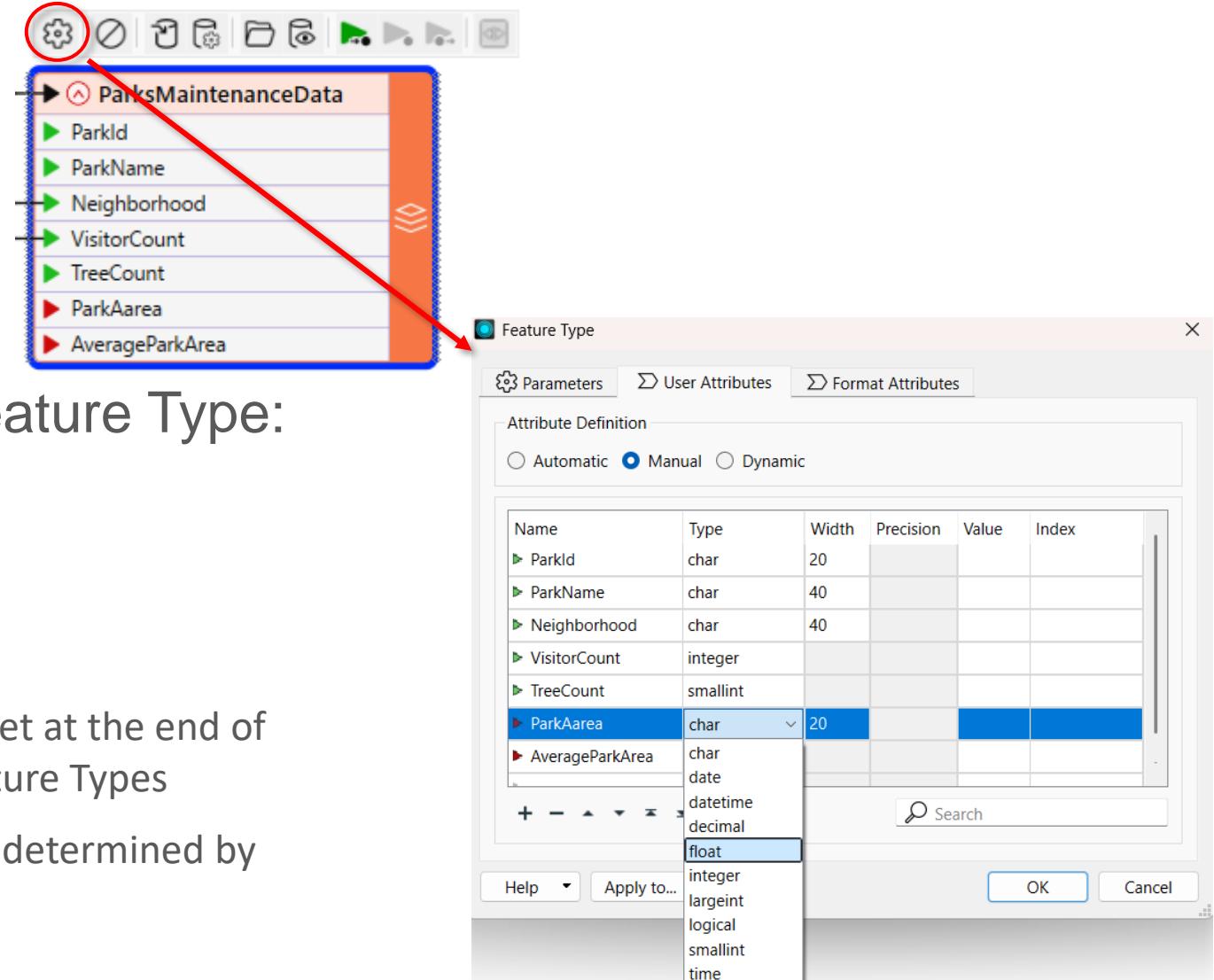
- Attribute definition:

- Automatic
  - Manual

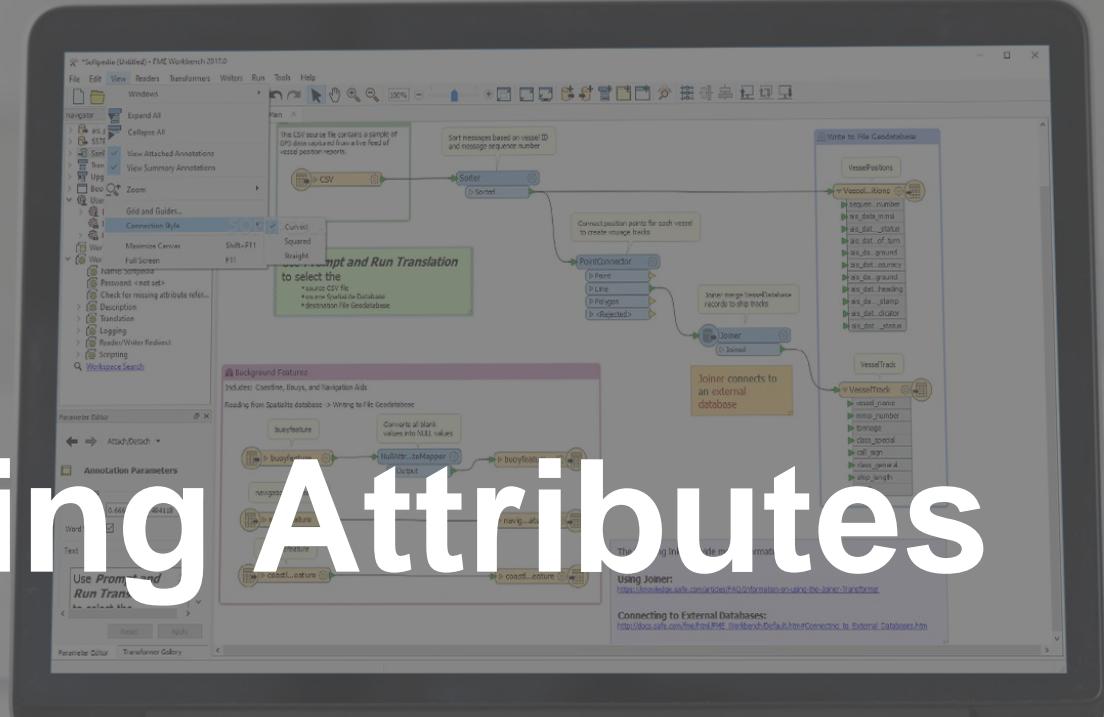
- Schema editing on writer Feature Type:

- Attribute renaming
  - Attribute order
  - Attribute Data Types:

- The data type of attributes are set at the end of the workflow on the Writer Feature Types
    - The Data Types available will be determined by the format being written to



# Managing Attributes



Connect. Transform. **Automate**

# Managing Attributes



There are numerous attribute management transformers. Here are the most popular:

Task	Transformers
Create Attributes	AttributeCreator, AttributeManager
Set Attribute Values	AttributeCreator, AttributeManager
Remove Attributes	AttributeKeeper, AttributeManager, AttributeRemover, BulkAttributeRemover
Rename Attributes	AttributeManager, AttributeRenamer, BulkAttributeRenamer
Copy Attributes	AttributeCopier, AttributeCreator, AttributeManager
Sort Attributes	AttributeManager
Change Attribute Case	BulkAttributeRenamer
Add Prefixes/Suffixes	BulkAttributeRenamer

- They **create** new attributes, **rename** them, **copy**, **sort** them, **set** values, and **delete** them
- Useful for the purposes of schema mapping
- Bulk - tasks allow the user to perform the same operation on a large number of attributes

# The Attribute Manager



- Create New attributes
- Reorder attributes

AttributeManager Parameters

Transformer Name: AttributeManager

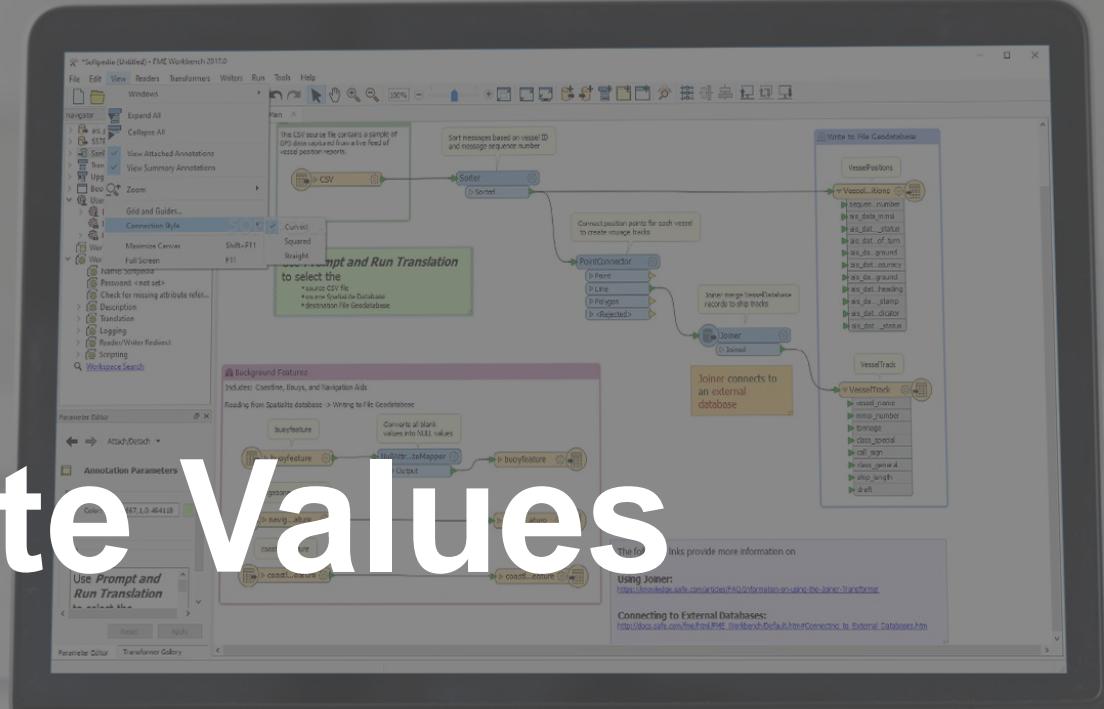
Advanced - Attribute Value Handling

Input Attribute	Output Attribute	Value	Action
ParkId	ParkId	<Enter value (op...)	Do Nothing
RefParkId	RefParkId		Remove
ParkName	ParkName	<Enter value (op...)	
NeighborhoodName	Neighborhood	<Enter new valu...	
VisitorCount	VisitorCount	<Enter value (op...)	
TreeCount	TreeCount	<Enter value (op...)	
DogPark	DogPark		Remove
Washrooms	Washrooms		Remove
SpecialFeatures	SpecialFeatures		Remove
<Expose existing attribute>		>Add new attribute>	

+ - ▲ ▼ × ✎ ↻ Import OK Cancel

- Rename attributes
- Remove attributes

# Attribute Values

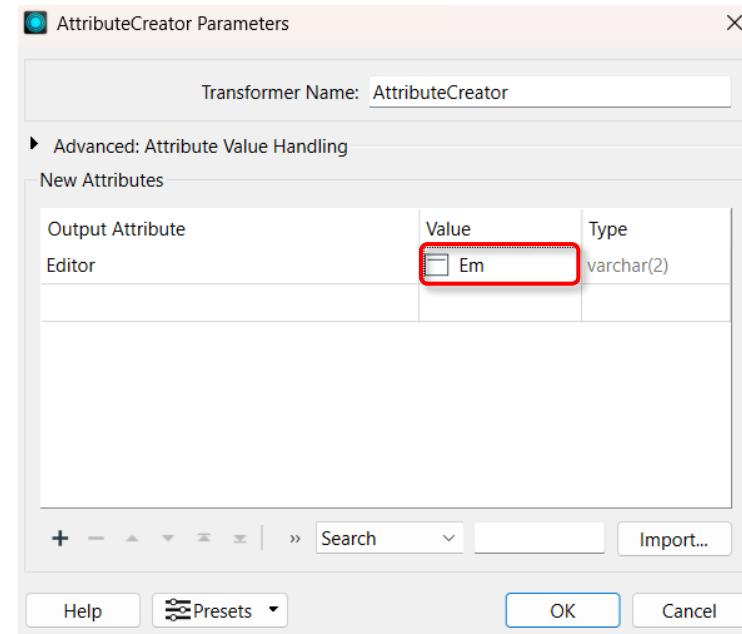
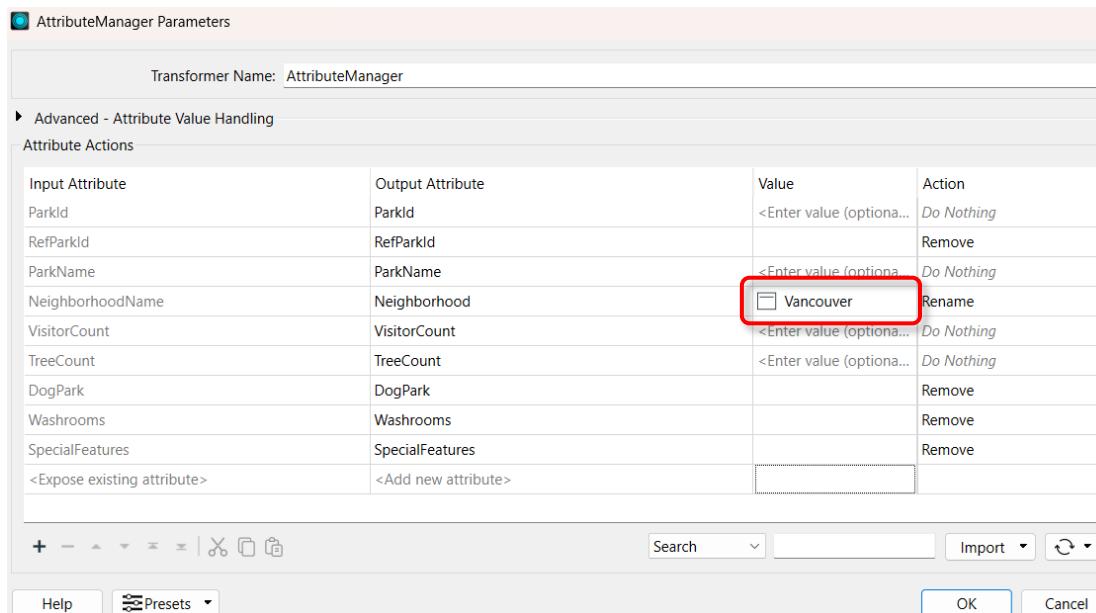


Connect. Transform. **Automate**

# Set a Fixed Attribute Value



- A fixed (or *constant*) value for an attribute can be created by simply entering the required value into the **Attribute Value field** of an attribute, within an attribute management transformer:



AttributeManager: Output					
	ParkId	ParkName	Neighborhood	VisitorCount	TreeCount
1	1	<missing>	Vancouver	9406	10
2	2	Rosemary Brow...	Vancouver	13100	8
3	3	Tea Swamp Park	Vancouver	11275	2
4	4	<missing>	Vancouver	9755	6

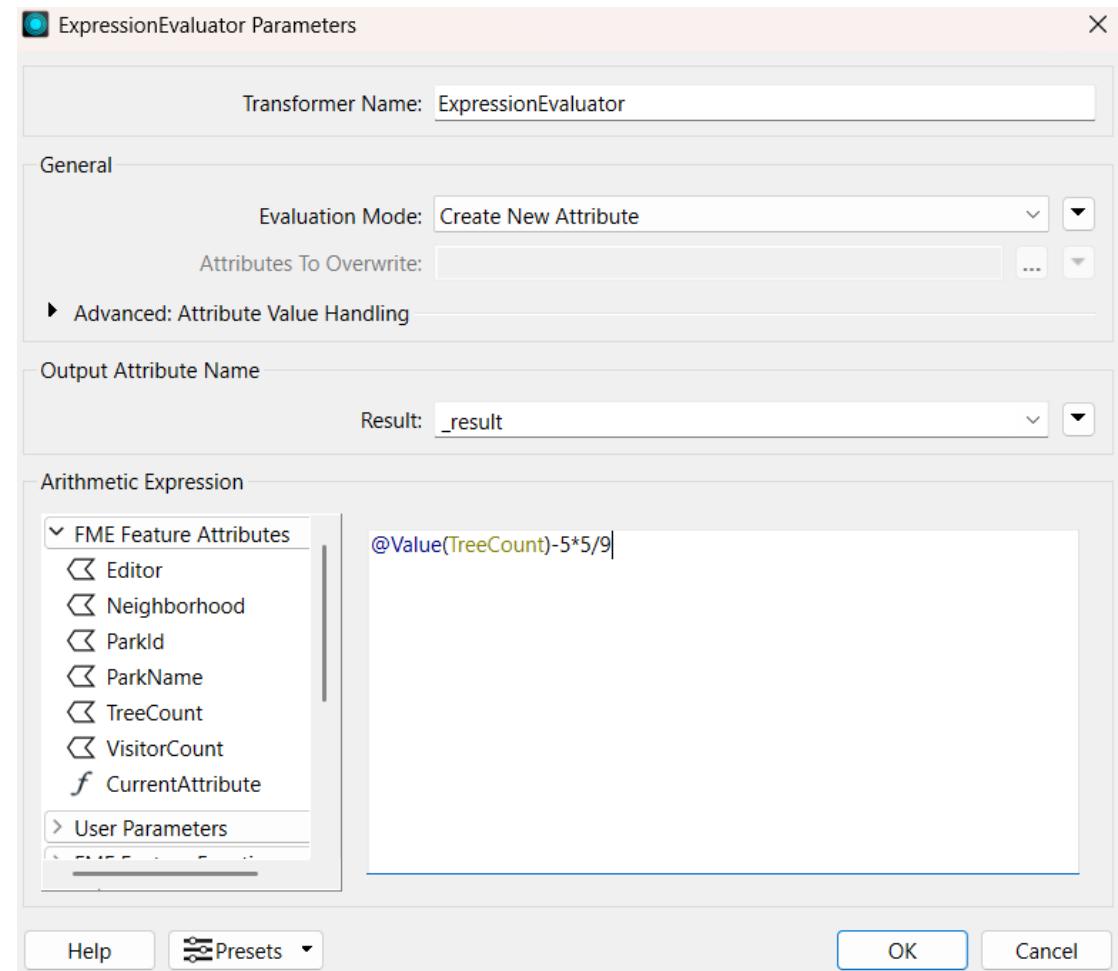
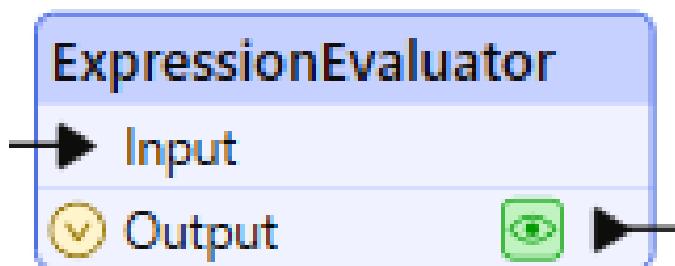
AttributeManager: Output						
	ParkId	ParkName	Neighborhood	VisitorCount	TreeCount	Editor
1	1	<missing>	Vancouver	9406	10	Em
2	2	Rosemary Brow...	Vancouver	13100	8	Em
3	3	Tea Swamp Park	Vancouver	11275	2	Em
4	4	<missing>	Vancouver	9755	6	Em

# Calculate Numeric attribute values



There are a couple of ways to perform mathematical calculations to create new attributes (or update existing attributes).

One method is to use the ExpressionEvaluator transformer:



# Exercise 2.1



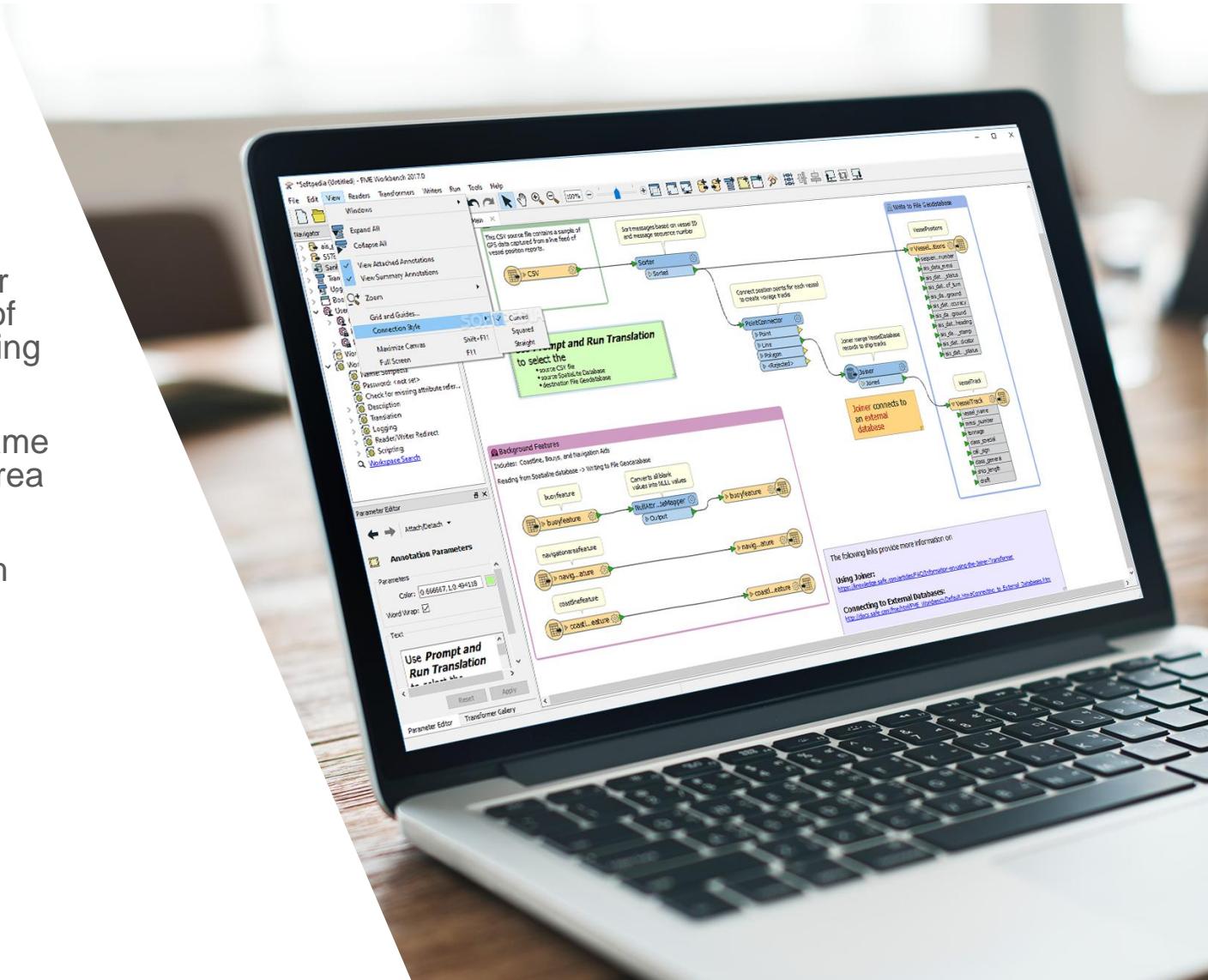
## Managing Attributes

- You are working as a technical analyst in the GIS department of your local city.
- The team responsible for maintaining parks and other grassed areas needs to know the area and facilities of each park in order to plan their budget for the upcoming year.
- You will need to remove unnecessary attributes, rename existing attributes and create new ones – including area calculations.
- The grounds maintenance team want the data in both spatial and CSV format.

Data - City Parks (MapInfo TAB)

### Goals:

- Update the schema for grounds maintenance data
- Calculate the average area of parks in the city
- Output results to multiple formats



# More Constructing Attribute options

---



On our ***Advanced Attribute Handling and Lists*** module we explore more options for constructing and manipulating attribute values:

- string manipulation - Text Editor
- arithmetic calculations - Arithmetic Editor
- Working with Date and Time:
  - Date/Time stamps
  - Date/Time formatting
  - Date/Time converting and calculations
- Functions - both text and arithmetic
- Conditional Values
- *...plus more...*

# Exercise 2.2



## Summary Statistics

- The parks team has decided that they want to know the average size of parks in each neighborhood. They also want to know the smallest and largest park size for each neighborhood. Let's generate these summary stats for them.

**Data - City Parks (MapInfo TAB)**

**Starter workspace –**

C:\FMEModularData\Workspaces\2.02-Attributes-SummaryStatistics-Begin.fmw

**Goal – Calculate summary statistics for the city parks at neighborhood level.**

# Useful Transformers for Attribute Value Manipulation

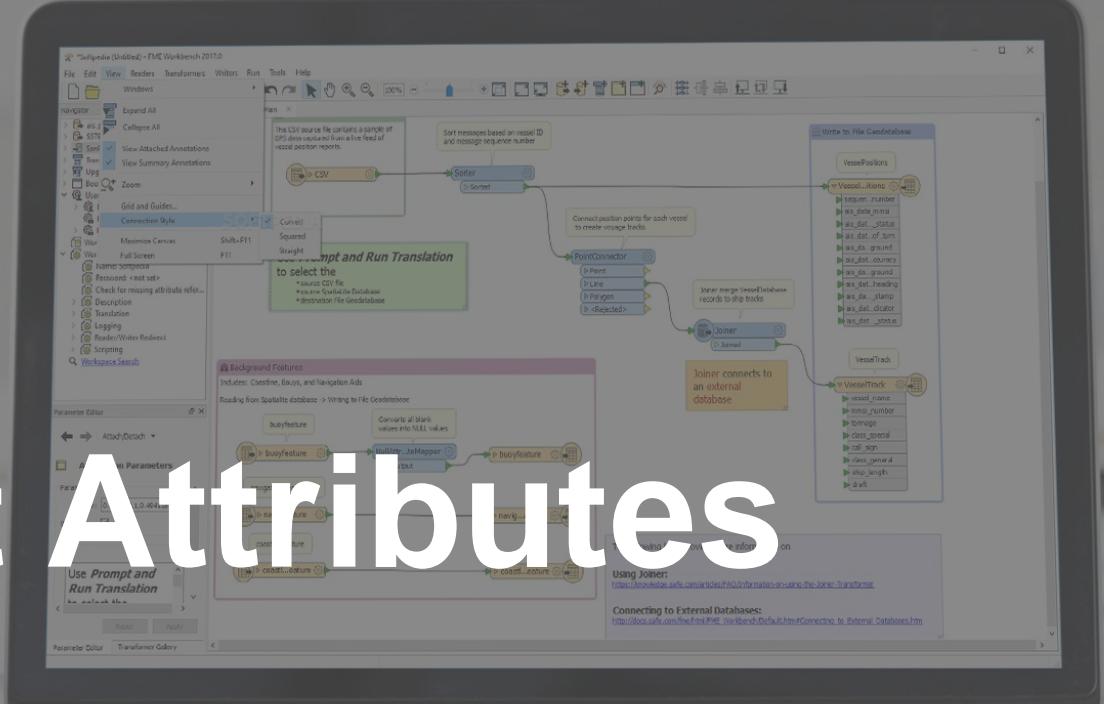
---



There are many transformers to construct, manipulate and format attribute values, but here are a few to get you started:

- **StringReplacer** - replaces substrings matching a string or regular expression in the string contained in the source attribute. (you can use it like Find & Replace)
- **StringCaseChanger** - changes the case of text attribute values to UPPERCASE, lowercase, Title case, or Full Title Case
- **StringConcatenator** - bring multiple attribute values together into one attribute
- **SubstringExtractor** - extracts a substring from the source attribute - using the range of characters specified.
- **AttributeSplitter** - splits attribute values into parts, based on a delimiter or fixed-width pattern
- **AttributeRounder** - Rounds numeric attribute values to the specified number of decimal places.

# Format Attributes



Connect. Transform. **Automate**

# Attributes in FME



There are two main types of attributes in FME:

- **User Attributes**

- These are the main attributes, and are always part of the feature.
- Typically these are what ‘end-users’ will see when working with the data in their systems.
- They are visible on the canvas (in attribute lists) and within the Table view of the Workbench Visual Preview and Data Inspector.

- **Format Attributes**

- FME reads the source data and stores information about its features and properties of the dataset as format attributes
- Format attributes are specific to your format's schema.
- A particular set of Format Attributes has the prefix **fme\_**. These attributes represent the data as it is perceived by FME and are sometimes also known as *FME attributes*

Exposed	Name	Type	Width	Precision	Index
✓	▶ ParkId	smallint			
✓	▶ RefParkId	smallint			
✓	▶ ParkName	char	40		
✓	▶ NeighborhoodName	char	40		
✓	▶ VisitorCount	smallint			
✓	▶ TreeCount	smallint			

Exposed	Name	Type
□	▶ fme_basename	char(50)
□	▶ fme_color	char(50)
□	▶ fme_dataset	char(50)
□	▶ fme_feature_type	char(50)
□	▶ fme_fill_color	char(50)
□	▶ fme_geometry	char(50)

# Format Attributes

---



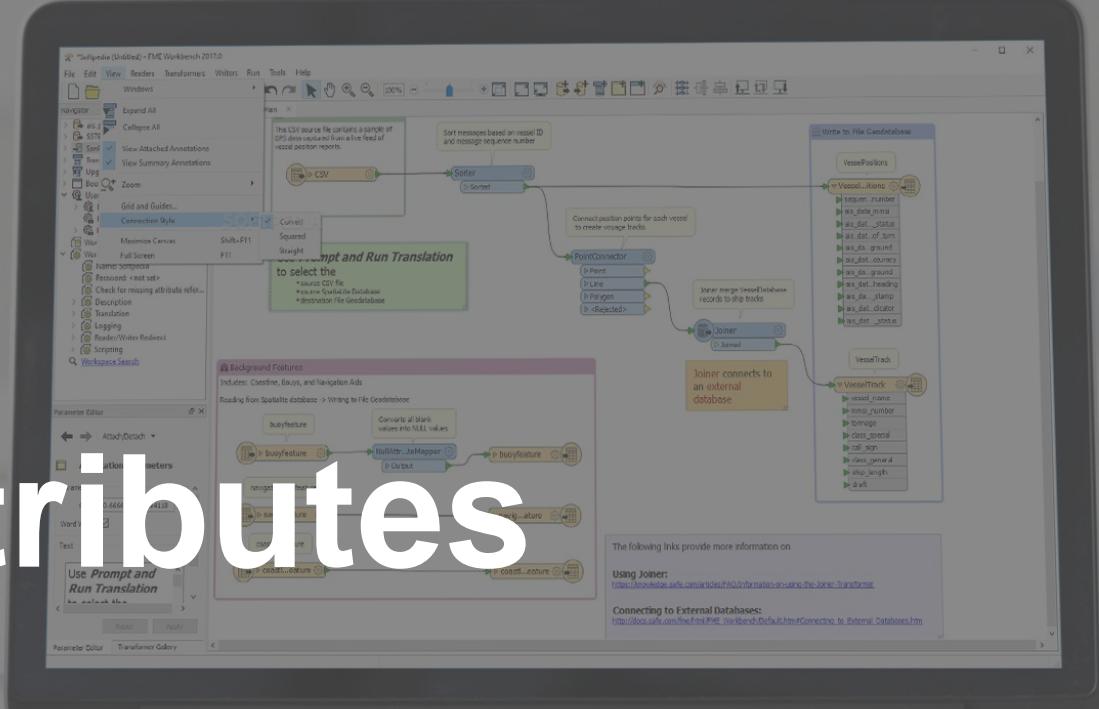
The most commonly used FME Format Attributes are:

- **fme\_feature\_type** - identifies a feature's original feature type (e.g. file, feature class, or table)
- **fme\_basename** - identifies the dataset's filename without the path or extension.
- **fme\_dataset** - contains path of the dataset, a URL, or the name of a database.
- **fme\_geometry** - indicates the geometry type of the actual coordinates (e.g. *fme\_aggregate*)
- **fme\_type** - specifies how that geometry is to be interpreted (e.g. *fme\_area*)

<b>fme_basename</b>	<b>fme_dataset</b>	<b>fme_feature_type</b>	<b>fme_type</b>
CompleteRoads	C:\FMEModularData\Data\Transportation\CompleteRoads.dwg	Secondary	fme_line
CompleteRoads	C:\FMEModularData\Data\Transportation\CompleteRoads.dwg	Secondary	fme_line
CompleteRoads	C:\FMEModularData\Data\Transportation\CompleteRoads.dwg	Other	fme_line
CompleteRoads	C:\FMEModularData\Data\Transportation\CompleteRoads.dwg	Arterial	fme_line

We'll see how these can be useful during our ***Workflow Design*** module!

# List Attributes



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

PointOnAreaOverlayer	
→ Point	
→ Area	
▼ Point	12,623
▲ Area	92
<code>_trees{}.PSTLADDRESS</code>	
<code>_trees{...mmonName</code>	
<code>_trees{}.Cultivar</code>	
<code>_trees{}.Diameter</code>	
<code>_trees{}.Genus</code>	
<code>_trees{}.Height</code>	
<code>_trees{}.PlantingDate</code>	
<code>_trees{}.Species</code>	
<code>_trees{}.TreelD</code>	
ParkId	
RefParkId	
ParkName	
NeighborhoodName	
VisitorCount	

# 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

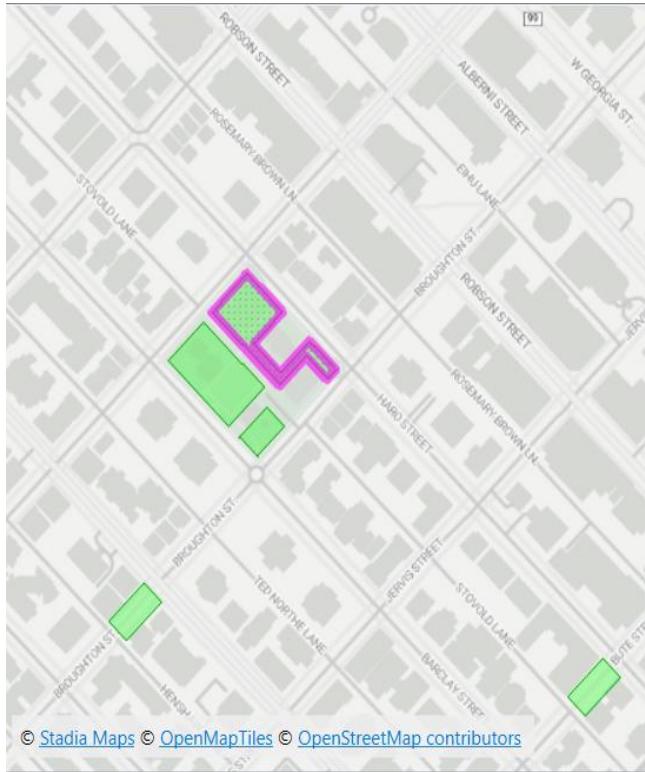
The screenshot shows a GIS map with several green polygon features labeled "David Lam Park" and "Hoppers' Park". Within these parks, there are numerous small green tree icons. Overlaid on the map is a configuration dialog for the "PointOnAreaOverlayer" transformer.

The dialog contains the following settings:

- Transformer Name:** PointOnAreaOverlayer
- General** section:
  - Group Processing
  - Areas First: No
  - Aggregate Handling: Deaggregate
- Attribute Accumulation** section:
  - Merge Attributes
  - Generate List On Output 'Point'
  - Generate List On Output 'Area'
    - 'Area' List Name: \_trees
    - Add To 'Area' List: All Attributes
    - Selected Attributes: (empty)
- Output Attribute Name** section:
  - Overlap Count: \_overlaps

A red box highlights the "Generate List On Output 'Area'" section, specifically the dropdown menu where "\_trees" is selected as the list name.

# List Attributes - example



Feature Information		
Features Selected: 1 of 1 In: PointOnAreaOverlayer: Area		
Property	Data Type	Value
Exposed Attr...	i	
ParkId	int16	11
RefParkId	int16	200
ParkName	varchar(40)	Barclay Heritage Square
NeighborhoodNa...	varchar(40)	West End
VisitorCount	int16	12918
TreeCount	int16	8
DogPark	varchar(1)	N
Washrooms	varchar(1)	Y
SpecialFeatures	varchar(1)	N
_trees	uint32	4
_trees{0} (4)		
_trees{0}		
PSTLADDR...	varchar(250)	1415 Barclay St
CommonN...	varchar(200)	Schwedler Norway Maple
Cultivar	varchar(200)	Schwedleri
Diameter	varchar(200)	26
Genus	varchar(200)	Acer
Height	varchar(200)	4
Species	varchar(200)	Platanoides
TreID	varchar(200)	58777

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

- CommonName
- Cultivar
- Diameter
- Genus
- Height
- PlantingDate
- PSTADRDRDRESS
- Species
- TreID

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)

The screenshot illustrates the difference in inspecting list attributes between a Table View and a Feature Information Window.

**Table View (Left):** Shows a grid of data for features. The columns include VisitorCount, TreeCount, DogPark, Washrooms, SpecialFeatures, and \_trees. The \_trees column contains numerical values (e.g., 35, 8, 7, 6, 6, 5, 4) which represent the count of trees. The Feature Information Window shows the details for these tree lists.

	VisitorCount	TreeCount	DogPark	Washrooms	SpecialFeatures	_trees
1	15042	6	N	<missing>	<missing>	35
2	12988	7	N	Y	N	8
3	12868	8	Y	<missing>	<missing>	7
4	11014	0	N	N	Y	6
5	15157	9	N	N	N	6
6	11410	2	N	N	N	5
7	18404	11	N	N	Y	4

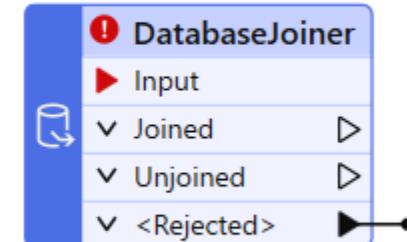
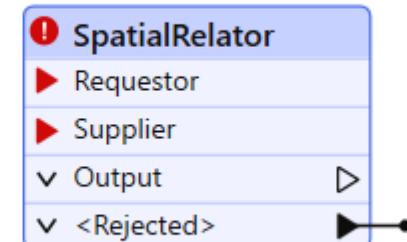
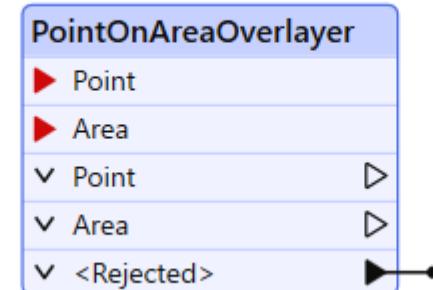
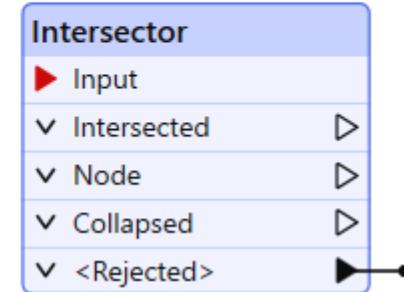
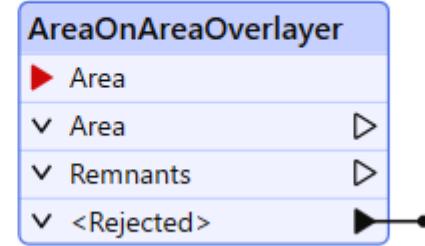
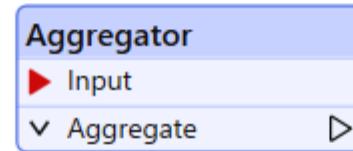
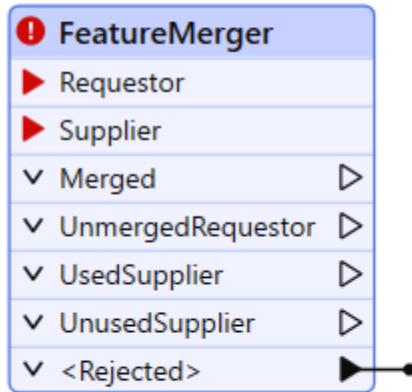
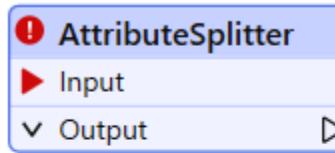
**Feature Information Window (Right):** Shows the properties for a selected feature. A red box highlights the list attribute '\_trees{0}' under the 'Property' column, which is expanded to show its elements: PSLADDRESS, CommonName, Cultivar, Diameter, Genus, Height, PlantingDate, Species, and TreeID.

Property	Data Type	Value
✓ _trees{0} (35)		
✓ _trees{0}		
PSTLADDRESS	varchar(250)	955 Evans Av
CommonName	varchar(200)	Capital Pear
Cultivar	varchar(200)	Capital
Diameter	varchar(200)	3
Genus	varchar(200)	Pyrus
Height	varchar(200)	1
PlantingDate	varchar(200)	20050321
Species	varchar(200)	Calleryana
TreeID	varchar(200)	76320

# Which transformers create lists?



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



# Exercise 2.3



## Create, View and Use Lists Attributes

- We have some address data for the city. We need to generate a new ID for each record. This new ID will be based on a component of an existing attribute called GlobalID.
- We will first split the GlobalID into its component parts, using a transformer. These component parts will be output as a list attribute. Which we will examine, then extract the required data to create our new ID attribute.

**Data - Addresses Full (GeoPackage)**

### Goal:

- use a transformer to generate list attributes
- inspect list attributes using the Feature Information window
- extract data from list attribute to create a new attribute

# Benefits of Lists

---



1. Generating lists for one-to-many relationships in database joins 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!

On our ***Advanced Attribute Handling and Lists*** module we explore working with Lists in more detail, including transformers to help you extract data from lists to use in your workspace

# Testing and Filtering data

Connect. Transform. **Automate**



# Testing and filtering data

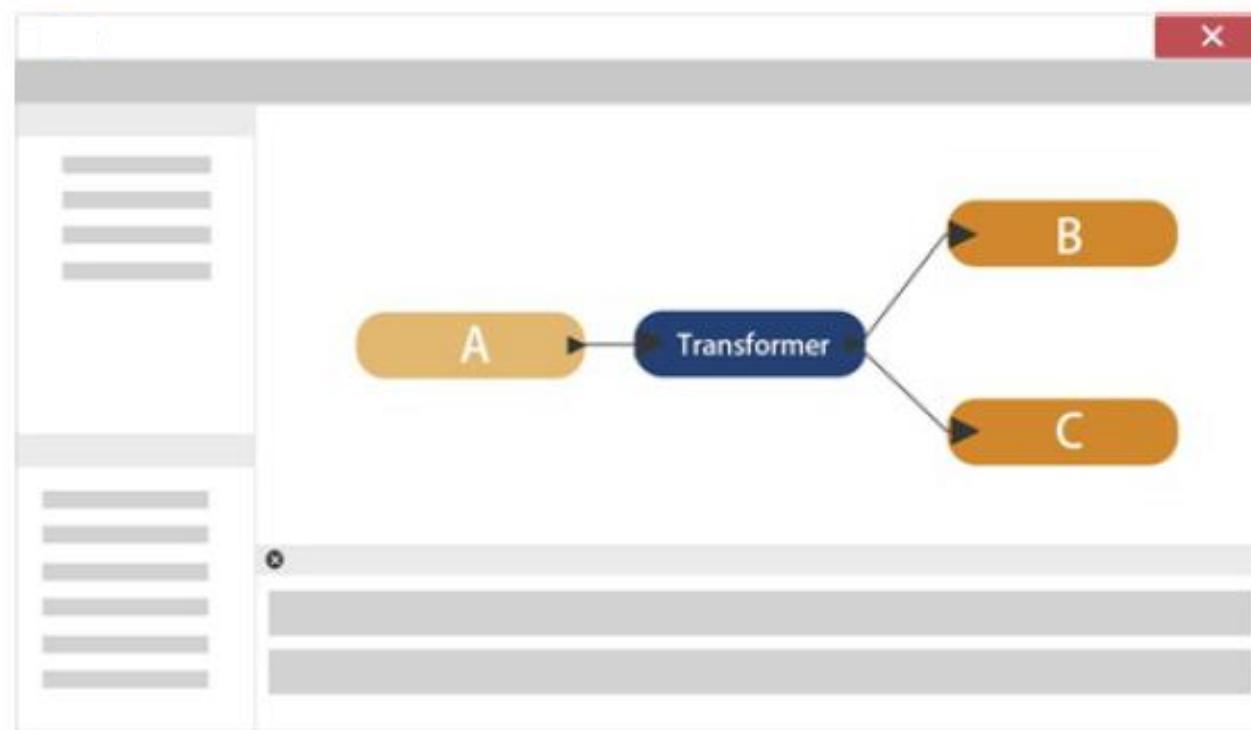


Transformers that test/filter don't transform data content, yet they are amongst the most commonly used Transformers

Some of the most popular Filter and Test Transformers are:

- **Tester**
- **TestFilter**
- **GeometryFilter**
- **AttributeFilter**
- **SpatialFilter**
- **Sampler**

A filtering transformer may be any transformer with multiple output ports:

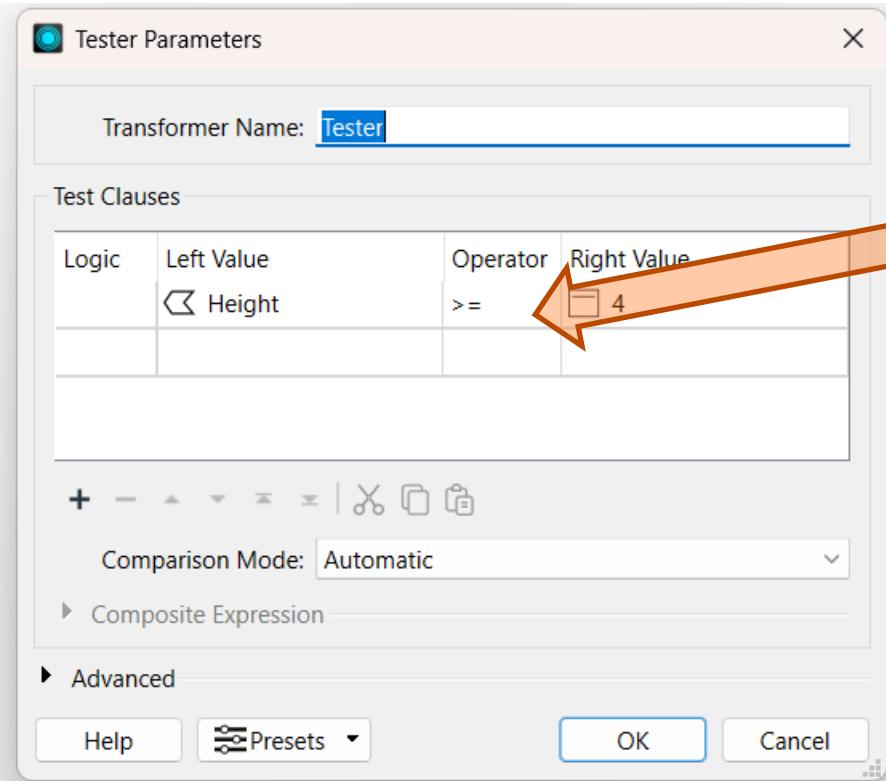
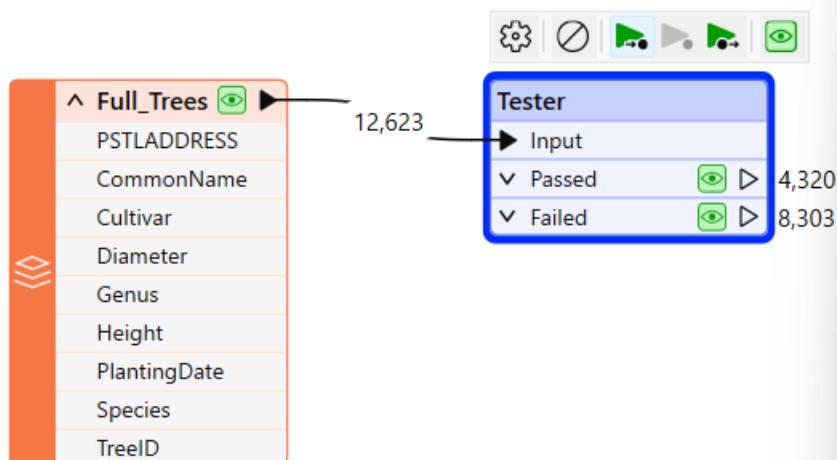


# The Tester transformer



## Single Clause

- Produces a Pass / Fail result
- *How many trees are greater than 4m tall?*



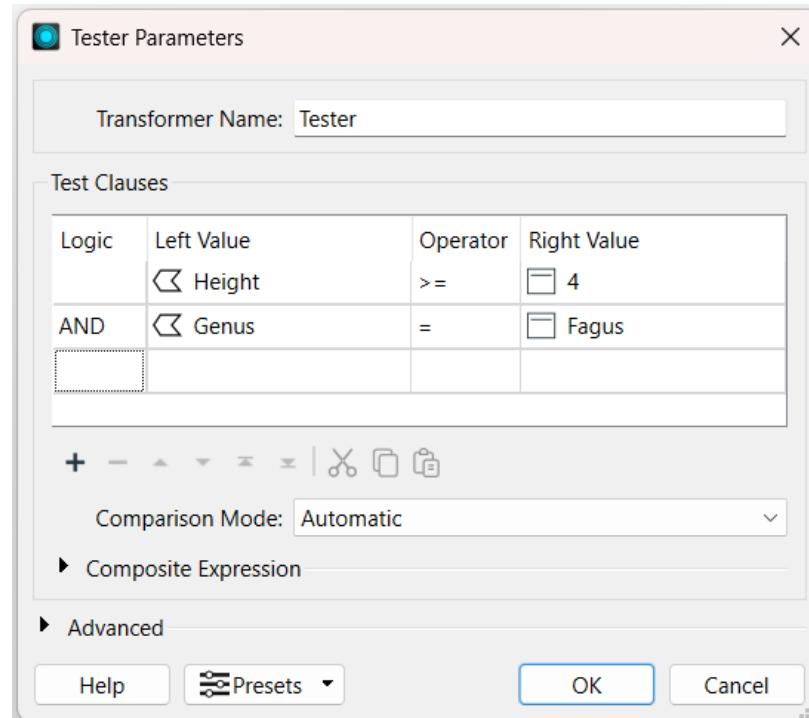
=	In Range
!=	In
<	Like
>	Contains
<=	Begins With
>=	Ends With
In Range	Contains Regex
In	Type Is
Like	Encodable In
Contains	Attribute Has a Value
Begins With	Attribute Is Null
Ends With	Attribute Is Empty String
Contains Regex	Attribute Is Missing

# The Tester transformer



## Multiple Clause

- Produces a Pass / Fail result
- Allows a combination of Multiple Tests – using AND / OR
- *How many trees are over 4m tall AND are of the Fagus genus?*



# The Tester transformer



## Composite Test

- Produces a Pass / Fail result
- Allows a blend of Multiple AND / OR tests
- *How many trees are equal to or greater than 4m tall and of the genus Fagus OR how many trees have a diameter over 4cm in diameter and of the genus Alnus?*

Tester Parameters

Transformer Name: Tester

Test Clauses

Logic	Left Value	Operator	Right Value
AND	Height	>=	4
OR	Genus	=	Fagus
AND	Diameter	>	4
	Genus	=	Alnus

+ - ▲ ▼ = × | ✖️ 🗑️ 🗑️

Comparison Mode: Automatic

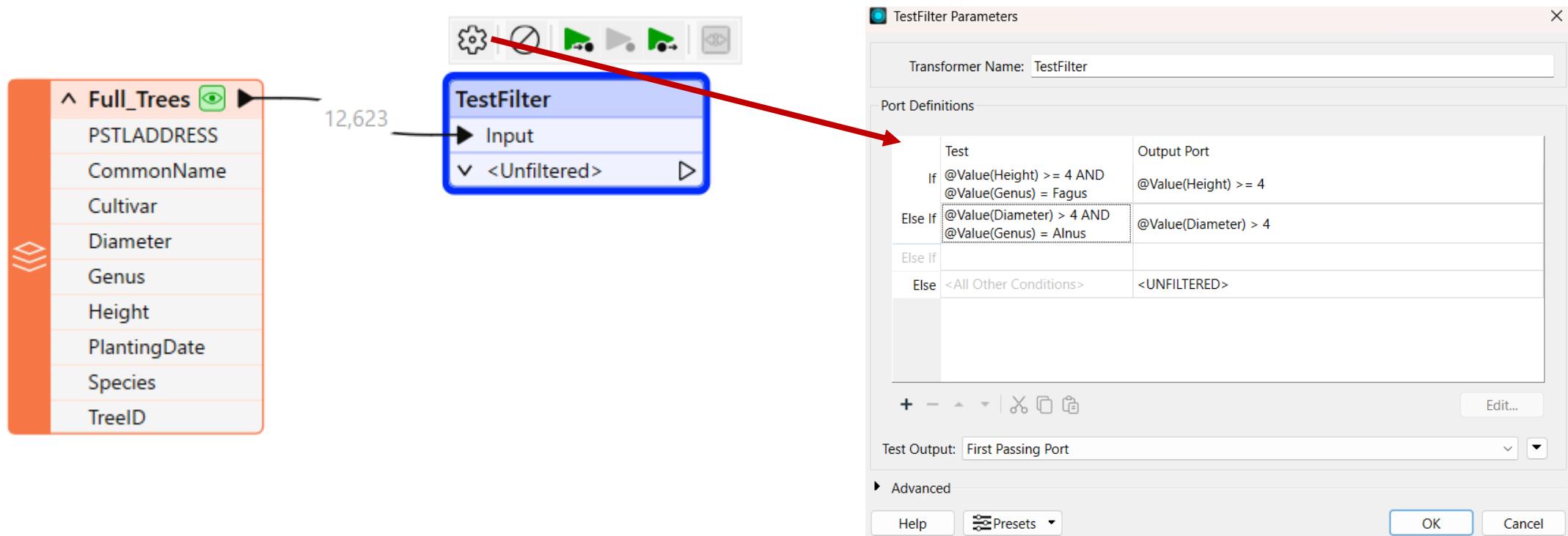
Composite Expression

Help Presets OK Cancel

# The TestFilter transformer



- Allows a number of conditions to be tested
- Each is given its own output port



# Exercise 2.4



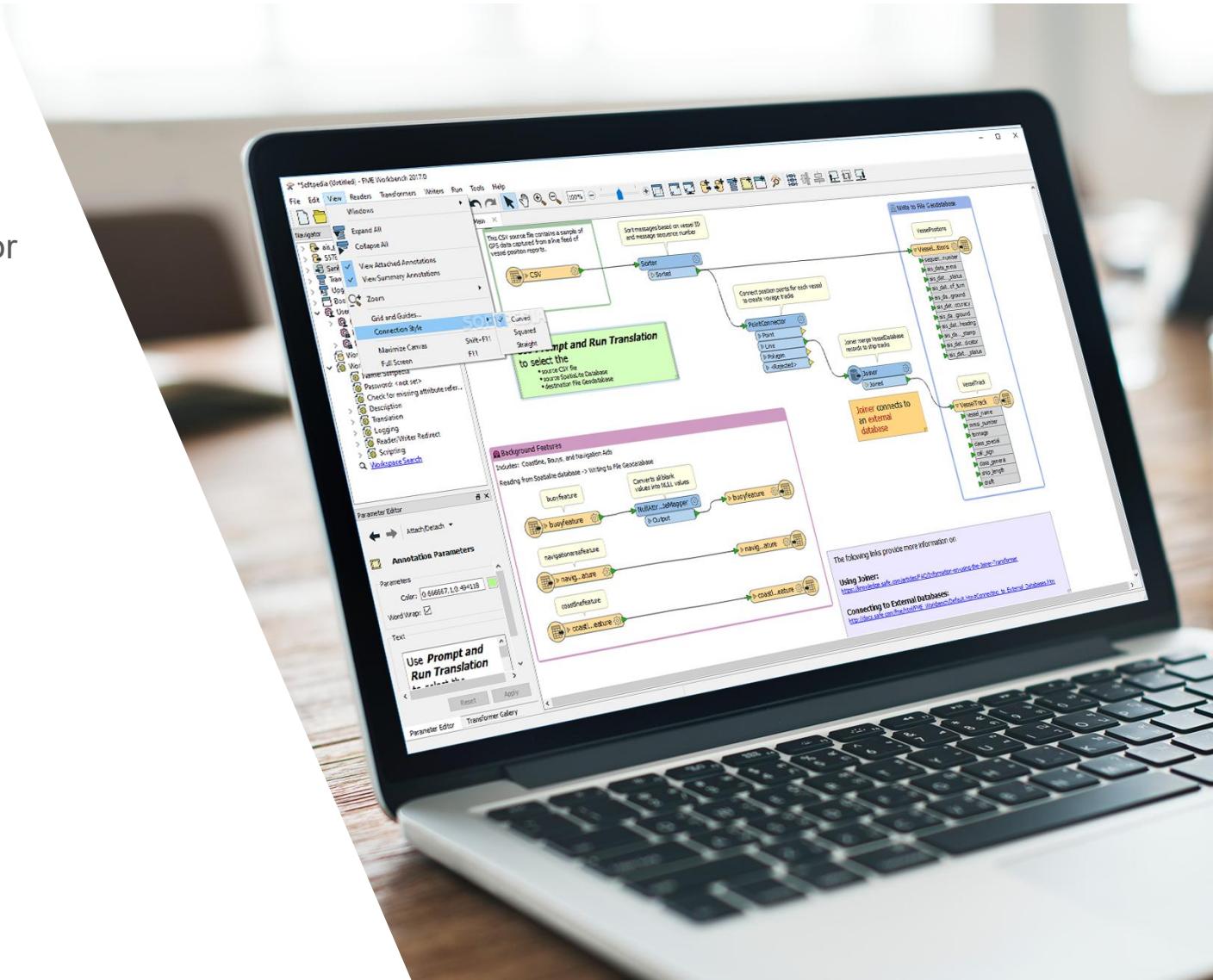
## Testing to Filter Data

- The Council's Environmental Directorate want to categorize the city Parks into 'premier', 'destination' or 'local' - based on their features.

Data - Parks (MapInfo TAB)

Start Workspace - none

Goal - use testing to filter the parks into groups, then assign the park category as a new attribute.



## Exercise 2.5



# Testing to QA Data

- We have a shapefile containing data about Sites of Special Scientific Interest, this needs validating and before it can be sent to a client.
  - The SSSI dataset already contains an attribute called Hectares. However this is a static value on the source data, and we don't know when this was last populated, or if it's still correct.

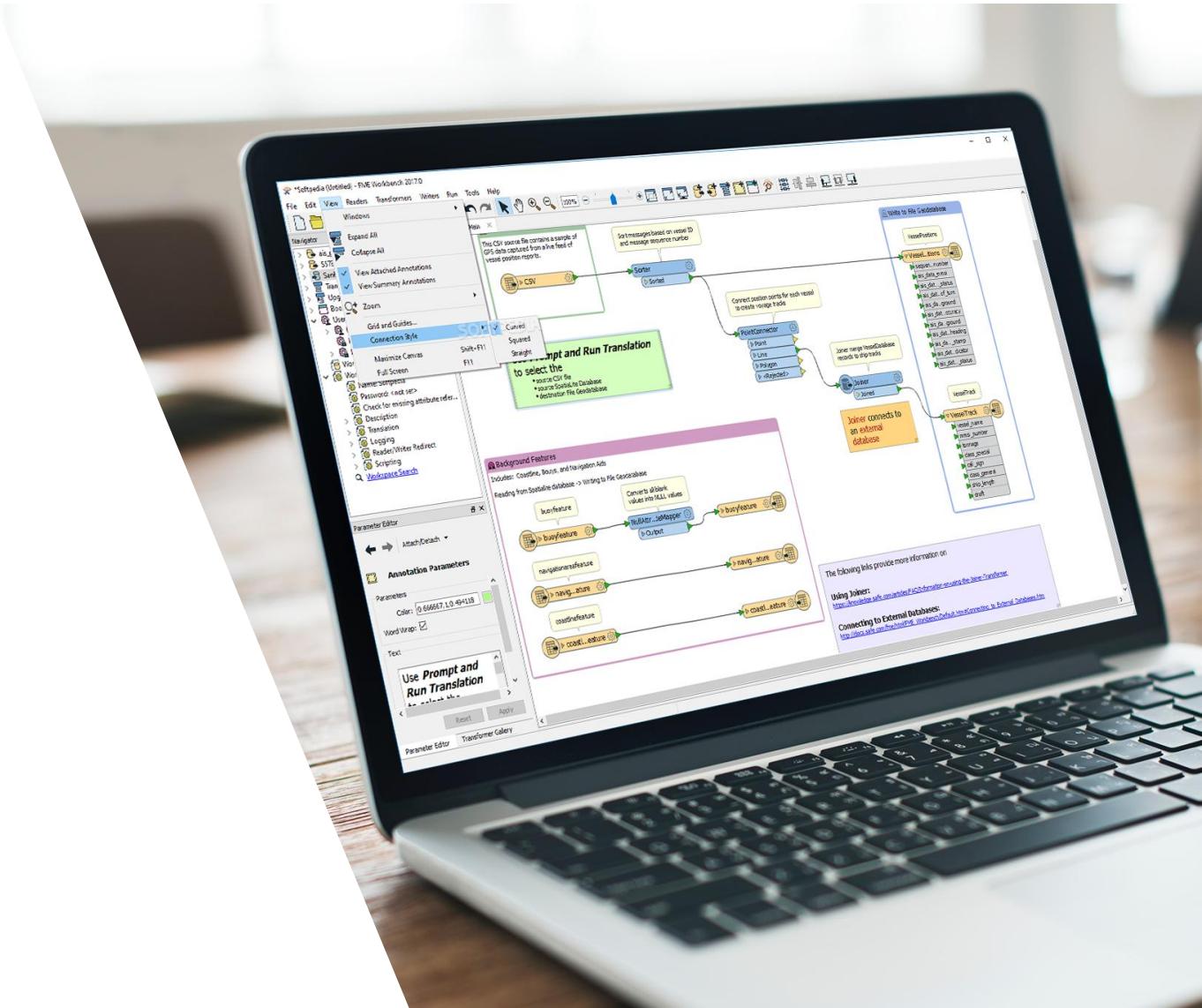
Data - SSSI (MapInfo TAB)

## Starter workspace –

C:\FMEModularData\Workspaces\2.05-Attributes-TestingQA-Begin.fmw

## Goals -

- use Testing to validate Hectares attribute
  - Manage, update and create new attributes



# The AttributeFilter transformer



- Directs features by the values in a chose attribute
- Each is given its own output port

The screenshot shows the AttributeFilter Parameters dialog box and a feature view in a GIS application.

**AttributeFilter Parameters Dialog:**

- Transformer Name: AttributeFilter
- Attribute Selection: Attribute to Filter by: Genus
- Possible Attribute Values:
  - <Empty>
  - <Missing>
  - <Null>
  - <Unfiltered>
  - Abies
  - Acer
  - Aesculus
  - Ailanthus
  - Albizia
  - Alnus
  - Amelanchier
  - Araucaria
  - Arbutus
  - Betula
  - Calocedrus
  - Carpinus
  - Castanea
  - Catalpa
  - Cedrus
  - Celtis

**Feature View:**

A feature view titled "Full\_Trees" is shown with 12,623 features. The tree species listed in the feature view correspond to the attribute values selected in the dialog box: Abies, Acer, Aesculus, Ailanthus, Albizia, Alnus, Amelanchier, Araucaria, Arbutus, Betula, Calocedrus, Carpinus, Castanea, Catalpa, Cedrus, and Celtis.

# The AttributeRangeFilter transformer



- The same as AttributeFilter – but numeric
- Each is given its own output port

The screenshot shows the configuration of an AttributeRangeFilter transformer. On the left, a data source named "Full\_Trees" is connected to the "Input" port of the transformer. The "Input" port has a value of 12,623 assigned to it. The transformer's interface includes a "General" section where the "Transformer Name" is set to "AttributeRangeFilter" and the "Source Attribute" is set to "Diameter". Below this is a "Range Lookup Table" which maps ranges of the "Diameter" attribute to output ports. The table is as follows:

From	To	Output Port
1	5.9	Tiny
5.9	10.8	Small
10.8	15.7	Small
15.7	20.6	Medium
20.6	25.5	Medium
25.5	30.4	Large
30.4	35.3	Large

At the bottom of the dialog are standard buttons: Help, Presets, OK, and Cancel.

# Testing and filtering data

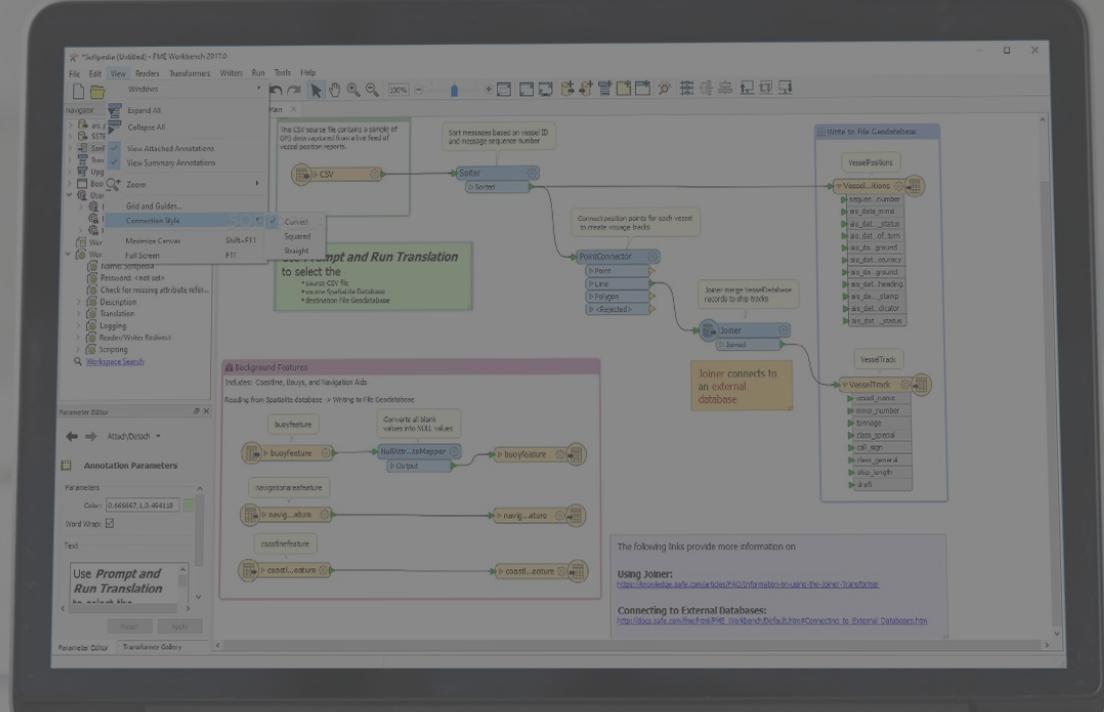


If the Tester, TestFilter, and AttributeFilter all filter features on the basis of an attribute condition, then what's the difference? When would I use each?

	Single Test		Multiple Tests		Test Type		Operators (Number of)	Attributes (Number of)
	Single Clause	Multi Clause	Single Clause	Multi Clause	String	Numeric		
Tester	Y	Y	-	-	Y	Y	16	Multiple
TestFilter	Y	Y	Y	Y	Y	Y	16	Multiple
AttributeFilter	Y	Y	-	-	Y	-	1	1
AttributeRangeFilter	Y	Y	-	-	-	Y	6	1



# Joins

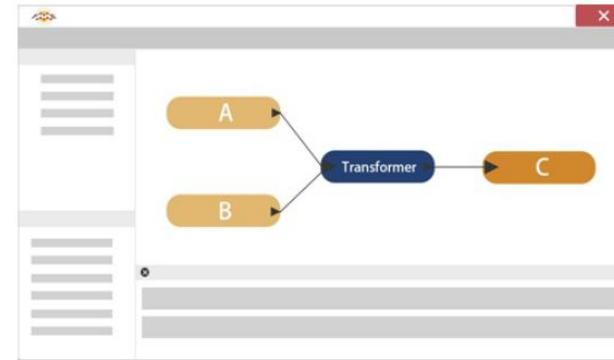


Connect. Transform. **Automate**

# Data Joins



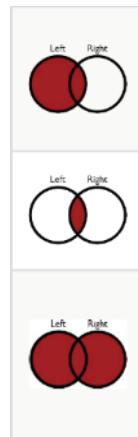
Data Join transformers bring data streams together, merging the data according to a set of user-defined conditions



## Key-Based Join Transformers

For joins based on matching attribute values:

- DatabaseJoiner
- FeatureMerger
- FeatureJoiner
- InlineQuerier



## Spatially Based Join Transformers

For joins based on a spatial relationship such as an overlap or proximity:

- NeighborFinder
- Overlayers
- SpatialRelator



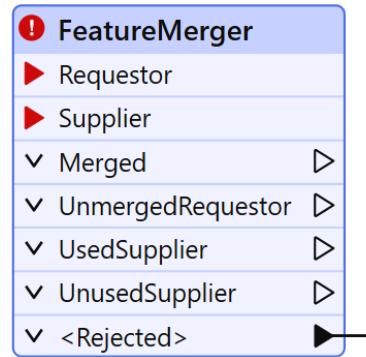
# Key based joins



## No SQL

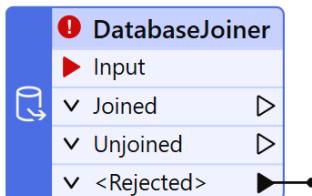


Joins features by combining the attributes/geometry of features based on common attribute values, like a SQL join operation



Copies and merges the attributes/geometry from one feature (or multiple features) onto another feature (or multiple features). When working with lists, try the ListBasedFeatureMerger

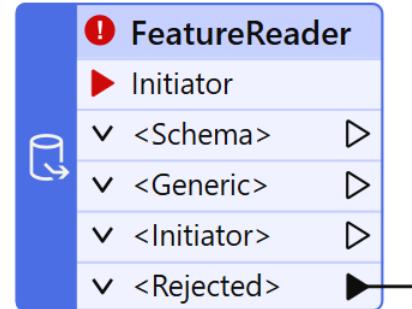
*Predecessor to the FeatureJoiner*



Joins attributes from an external table to incoming features as they are being processed through a translation

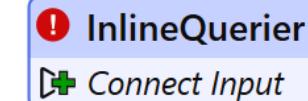
*Many small queries, 1:Many, uses look up tables*

## SQL



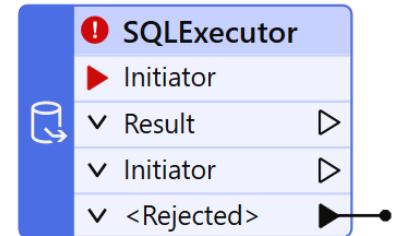
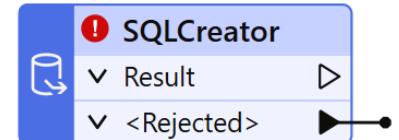
Reads features from any FME-supported format. The features read can be constrained by specifying WHERE clause or a spatial filter for formats that support them.

*Joins any format, bulk queries, spatial joins*



Executes SQL queries against a temporary database consisting of tables created from incoming features, returning results as new features.

*Robust functionality*



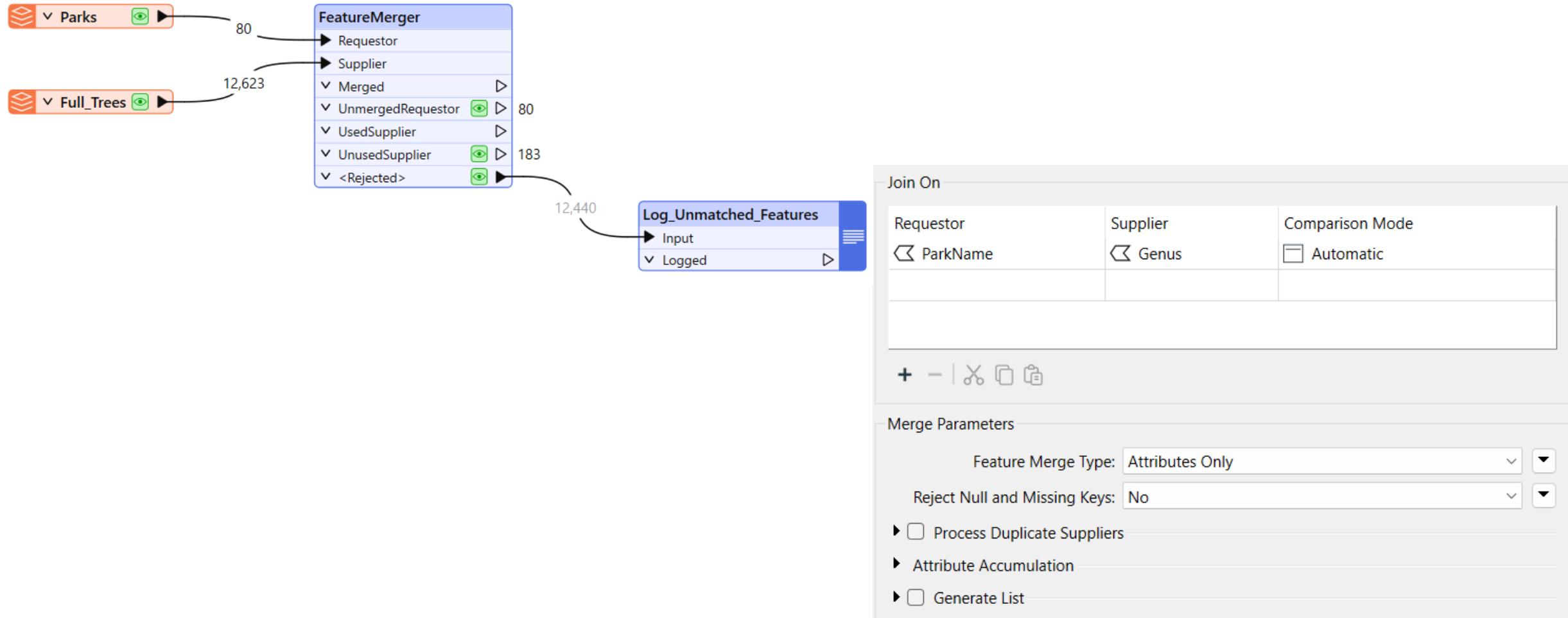
Performs SQL queries against a database. Either can be used interchangeably the only difference is the SQLCreator works at the beginning of the workflow.

*Reads in data, query can be made against other sources*

# The FeatureMerger Transformer



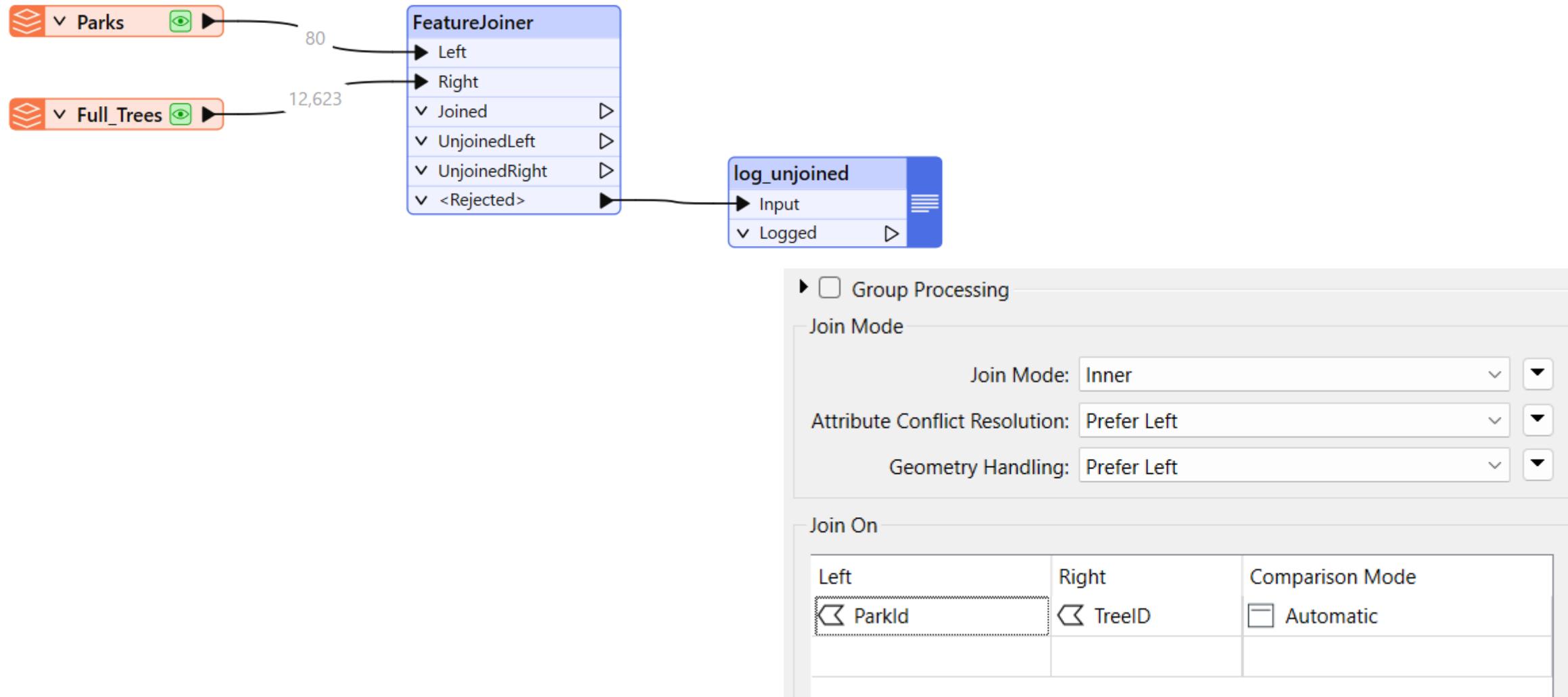
- Joins 2 or more streams of data – based on a key field match



# The FeatureJoiner Transformer

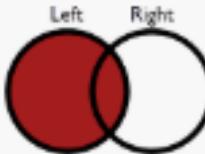
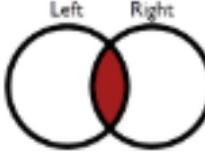
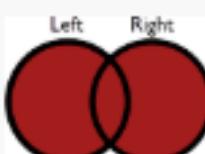


- Joins 2 or more streams of data – based on a key field match



# Data Joins – Key based



Mode	Description	Depiction	Joined Output	Unjoined Left	Unjoined Right
Left	Left features look for a match and are output whether they find a match or not		All matches plus unmatched Left features	None	Unused Right features
Inner	Left features look for a match and are output if they find one		All matches only	Unmatched Left features	Unused Right features
Full	Both Left and Right features output through the Joined output port, whether they find a join or not		All matches plus unmatched Left and Right features	None	None

# Exercise 2.6



## Joining Datasets - Voting Analysis

- The Electoral Services Officer has asked for help identifying voting divisions that had a low turnout at the last election, or divisions where voters had difficulties understanding the process.
- They ask for your help, and you suggest the results should be provided MapInfo TAB format to enable them to generate visualizations for reporting.

### Data –

- Election Results (Microsoft Excel)
- ElectionVoting (OGC GML)

### Starter Workspace –

C:\FMEModularData\Workspaces\2.06-Attributes-Joins-Begin.fmw

**Goal** – Generate statistics of voting patterns by performing key-based join and mathematical calculations.



Thanks for joining!

Any questions?



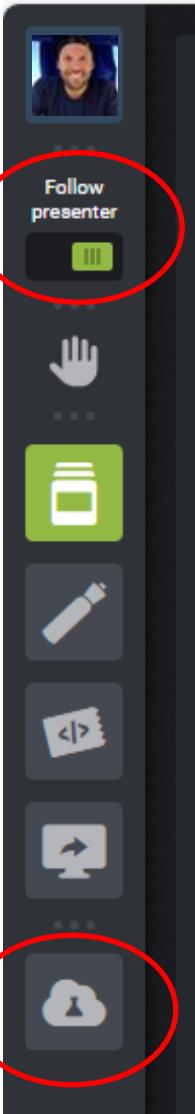
# FME Form Training

## - Module 3

Working with Spatial Data



# Training Environment



< keep 'Follow presenter' on

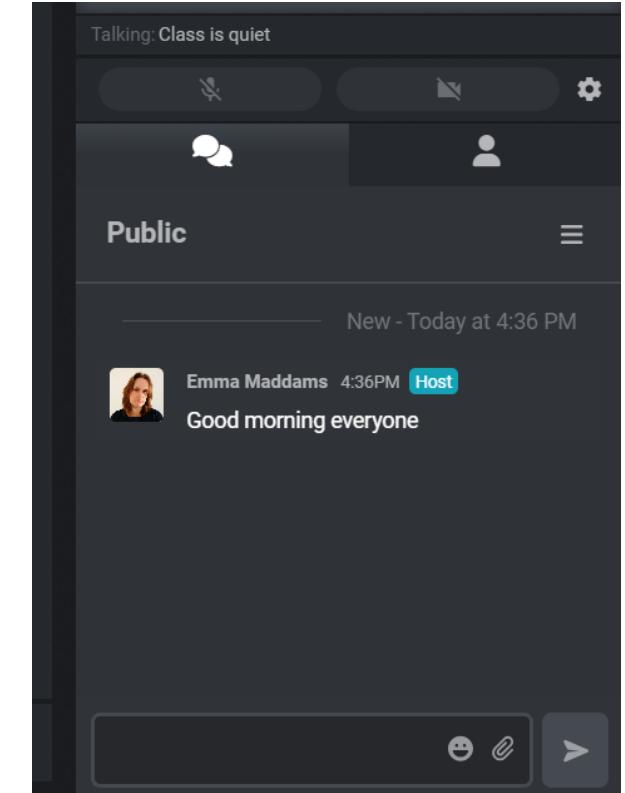
< Lab

need help when using your Lab  
Use either:

- 'Raise hand' or
- 'Lab Assistance'

## Chat

- to everyone
- to trainer



# Training Environment

---



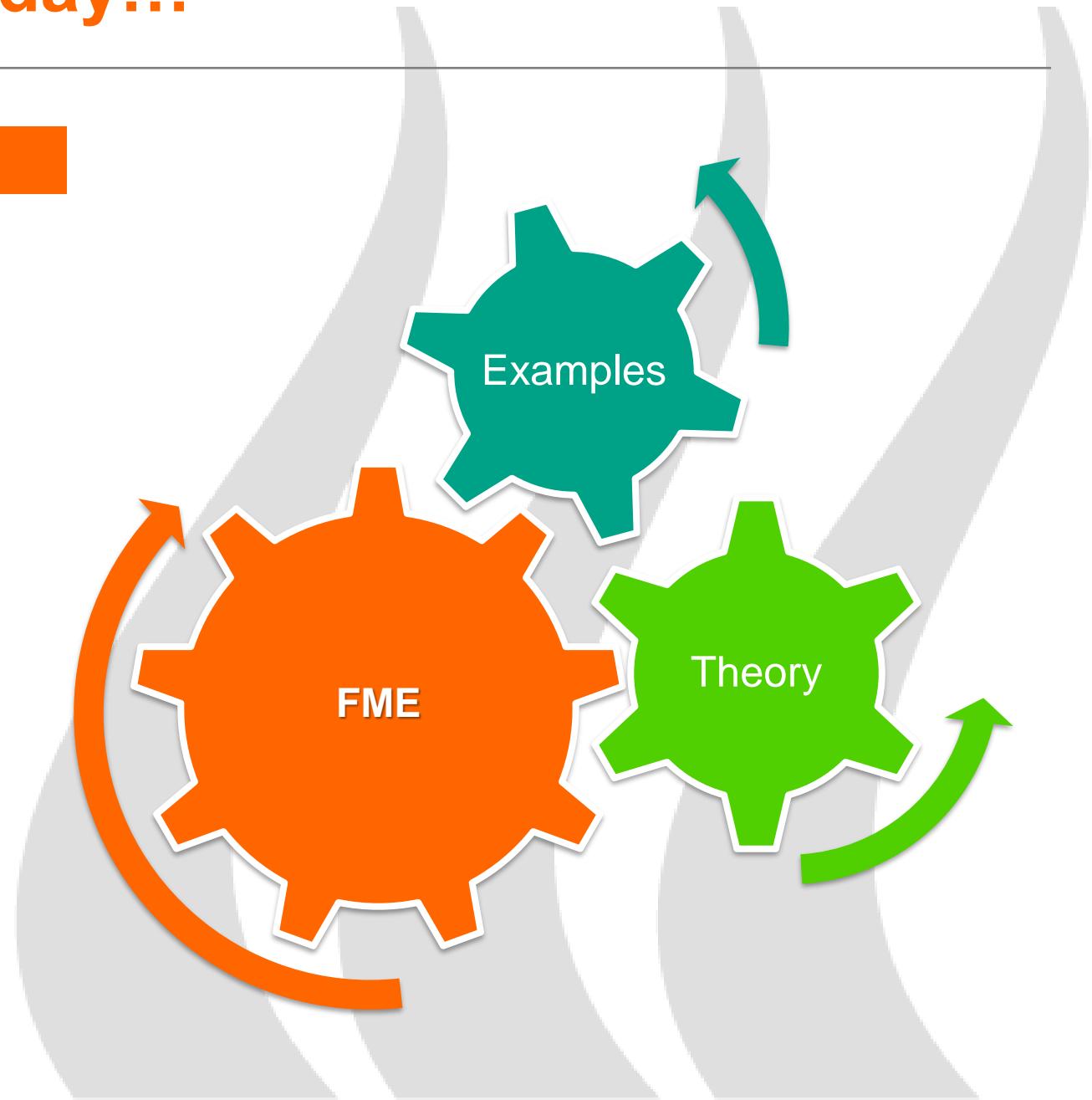
- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

# What we'll be covering today...

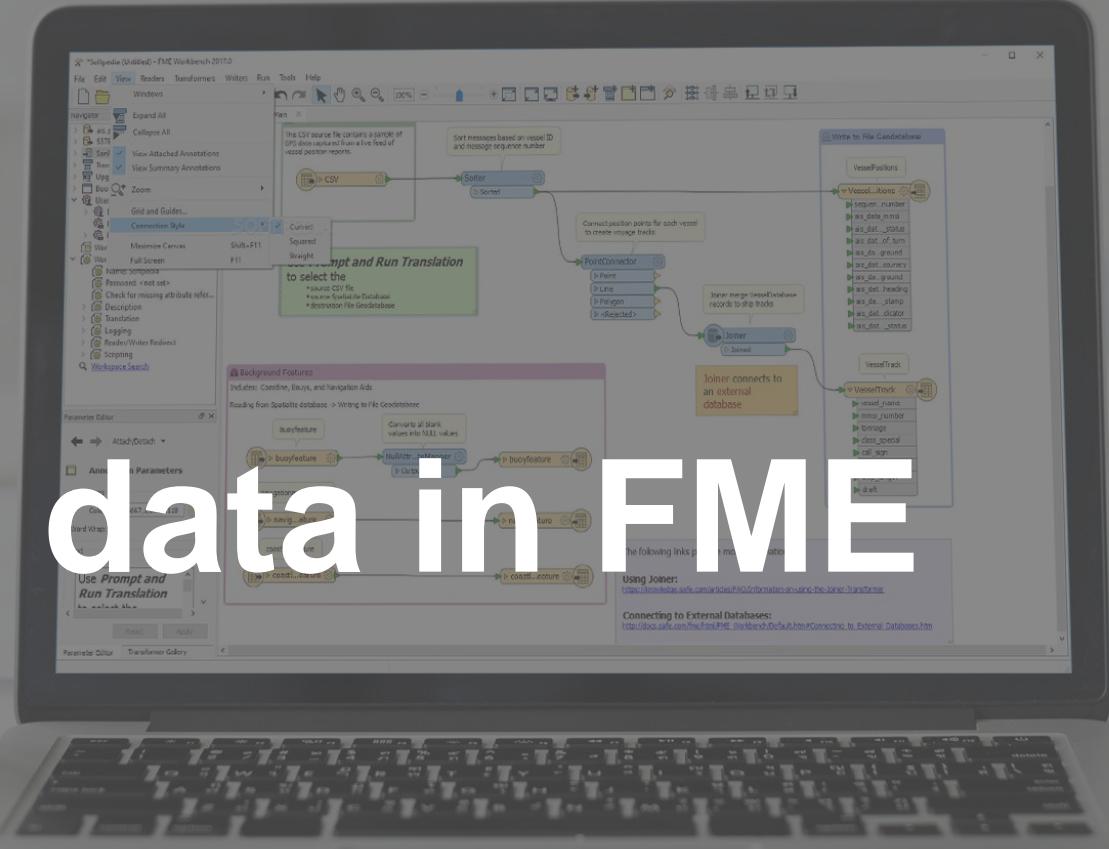
---

## Agenda

- Overview of geospatial data
- Coordinate systems
- Geometry types
- Symbology
- Spatial transformers
  - geometry
  - analysis
  - validation
- Spatial Joins



# Spatial data in FME



Connect. Transform. **Automate**

# Geospatial data basics



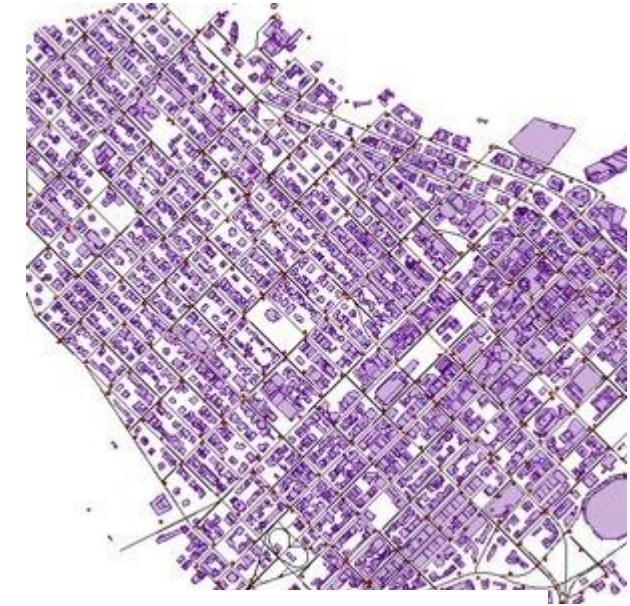
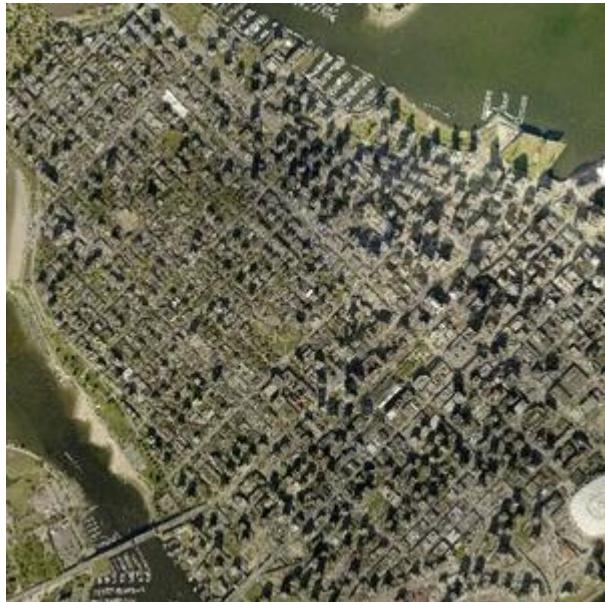
## Vector

Vector data is best described as graphical representations of the real world.

There are three main types of vector data:

- Points
- Lines
- Polygons

Connecting points create lines, and connecting lines that create an enclosed area create polygons.



## Raster

Raster data is data that is presented in a grid of pixels. Each pixel within a raster has a value, whether it be a colour or unit of measurement, to communicate information about the element in question. Rasters typically refer to imagery. In the spatial world, this may specifically refer to orthoimagery which are photos taken from satellites or other aerial devices.

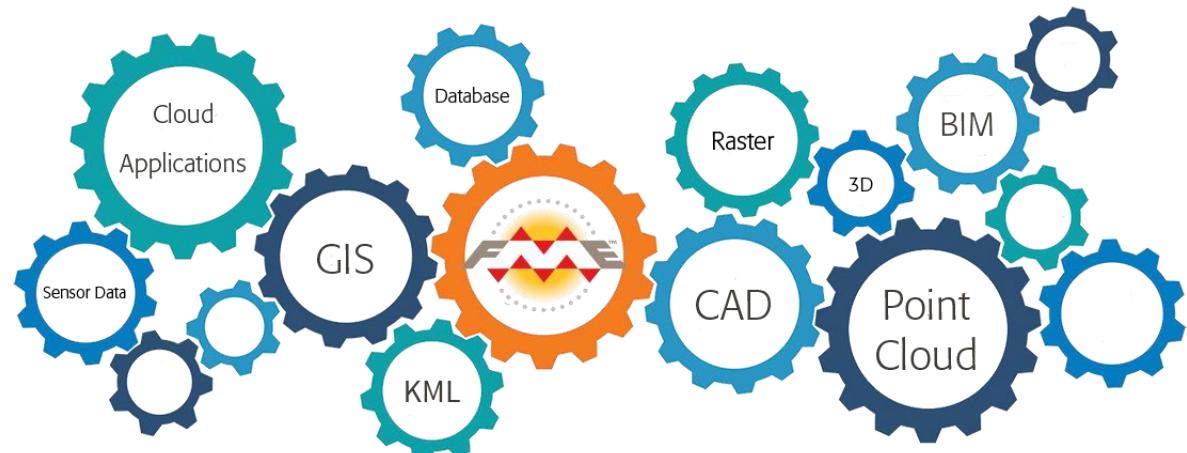
# Geospatial data

---



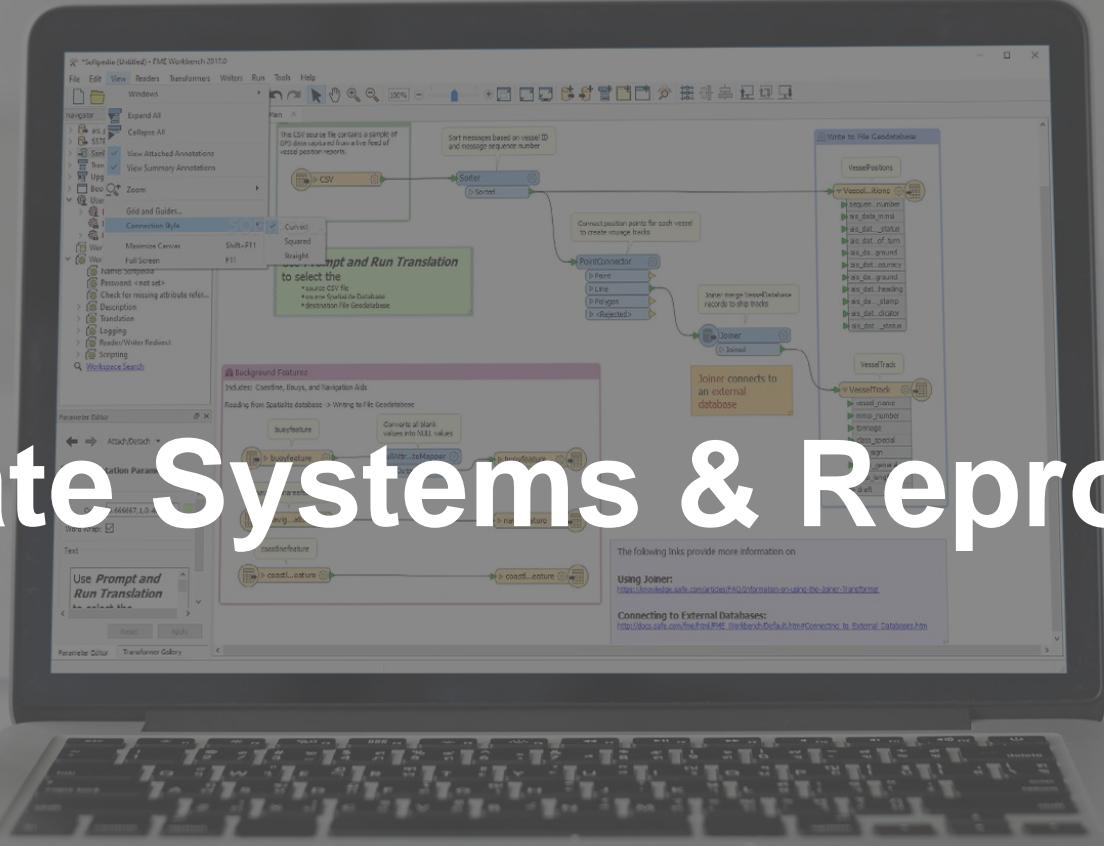
FME supports a wide range of commonly used spatial formats and applications including:

- GIS: ArcGIS, MapInfo, GeoMedia, Smallworld
- CAD: AutoCAD, MicroStation
- Spatial databases: PostGIS, Oracle Spatial, Geodatabase
- Rasters
- Point clouds
- XML-based spatial formats: KML, GML, CityGML
- BIM: Revit, IFC, SketchUp
- Cloud-based mapping platforms: ArcGIS Online, CARTO, Socrata
- and many more



# Coordinate Systems & Reprojecting

Connect. Transform. **Automate**



# Geospatial data basics - Coordinate Systems

---



- Coordinate systems
  - Coordinate systems enable geographic datasets to use common locations for integration.
  - Used to represent the locations of geographic features, imagery, and GPS locations, within a common geographic framework.
  - British National Grid
  - World Geodetic System 1984 (WGS84)
- FME interprets the coordinate system being read in
- **Data must be in the same coordinate system to calculate any spatial relationships**

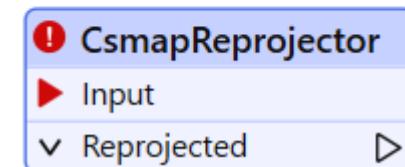
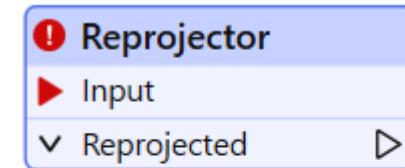


# Coordinate Systems

---



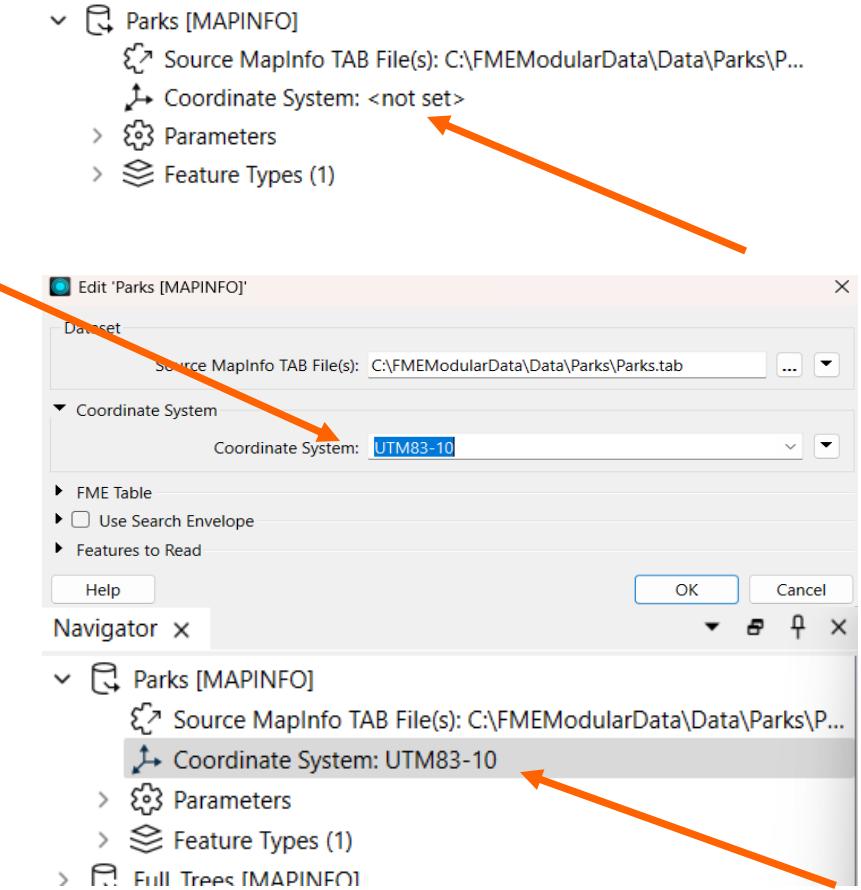
- FME interprets the coordinate system being read in and will carry that through the workspace and output it if you are writing to a spatial file
- You can set coordinate system on readers/writers or use the **CoordinateSystemSetter**
- You may also need to **reproject** your data, this may be to do a spatial join or to ready your file for another application
  - **Reprojector**: performs simple reprojections, without z value handling
  - **CsmapReprojector**: reprojections with finer control than the Reprojector, including z values (uses the CS-MAP library)



# Coordinate Systems Parameter on Reader/Writer



- when a reader's Coordinate System parameter is defined as **<not set>** FME will automatically try to determine the correct coordinate system from the dataset itself. If it does not do that, double click on the '**Coordinate System**' tab and enter the desired coordinate system. In this case, it is **UTM83-10**.
- When the source dataset is in a format that stores coordinate system information you can safely leave the parameter unset. - *However, it can be useful to users to see the coordinate information, especially in workspaces with multiple inputs with different coordinate systems.*
- You **must** set this parameter when you wish to reproject **source data that does not store coordinate system information**; otherwise, an error will occur in the translation.



# Demo – Coordinate Systems



Coordinate system Defined in data

C:\FMEModularData\Data\GB\local authority boundaries.gpkg

The screenshot shows the FME Data Inspector interface. In the top bar, the 'Format' is set to 'OGC GeoPackage' and the 'Dataset' is 'C:\FMEModularData\Data\GB\local authority boundaries.gpkg'. The 'Parameters...' button is highlighted with a red box. In the 'Coordinator System' dropdown, the option 'Read from source' is selected and highlighted with a red box.

**Navigator**

- local authority boundaries [OGCGEOPACKAGE]
  - GeoPackage File: C:\FMEModularData\Data\GB\local authority boundaries.gpk...
  - Coordinate System: <not set>
- Parameters
- Feature Types (1)

**Feature Information**

Features Selected: 1 of 1      In: local\_authority\_boundaries

Property	Data Type	Value
Exposed Attributes (7)		
Unexposed Attributes (8)		
Geometry	Coordinate System	OSGB-GPS-2015-OSTN15
	Dimension	3D
Number of Vertices		3634
Min Extents		397807.696, 275890.896
Max Extents		418493.301, 301227.804
MultiArea (1 Part)		

A map view shows the boundaries of local authorities in England. A callout bubble labeled 'set parameter' points to the 'Coordinate System' entry in the 'Geometry' row of the 'Feature Information' table.

Coordinate system Not defined in data

C:\FMEModularData\Data\ElevationModel\Bathymetry.dgn

The screenshot shows the FME Data Inspector interface. In the top bar, the 'Format' is 'Bentley MicroStation Design (V8)' and the 'Dataset' is 'C:\FMEModularData\Data\ElevationModel\Bathymetry.dgn'. The 'Parameters...' button is highlighted with a red box. In the 'Coordinator System' dropdown, the option 'Unknown' is selected and highlighted with a red box.

**Reader**

Format: Bentley MicroStation Design (V8)  
Dataset: C:\FMEModularData\Data\ElevationModel\Bathymetry.dgn

**Bathymetry [DGNV8]**

- Source Bentley MicroStation Design File(s): C:\FMEModularData\Data\Elev...
- Coordinate System: <not set>
- Parameters
- Feature Types (2)

**Feature Information**

Property	Value
Feature Type	Contours
Coordinate System	Unknown
Dimension	3D

A 3D point cloud visualization of bathymetry data is shown. A callout bubble labeled 'set parameter' points to the 'Coordinate System' entry in the 'Feature Information' table.

**Bathymetry [DGNV8]**

- Source Bentley MicroStation Design File(s): C:\FMEModularData\Data\Elev...
- Coordinate System: UTM83-10
- Parameters
- Feature Types (2)

**Geometry**

Coordinate System	UTM83-10
Dimension	3D
Number of Vertices	12
Min Extents	487302.54805583955, 5462202
Max Extents	488567.5954343319, 5462384.0
Line (12 ...)	(488323.69410257344, 5462384.0)

# Geometries in FME



Connect. Transform. **Automate**

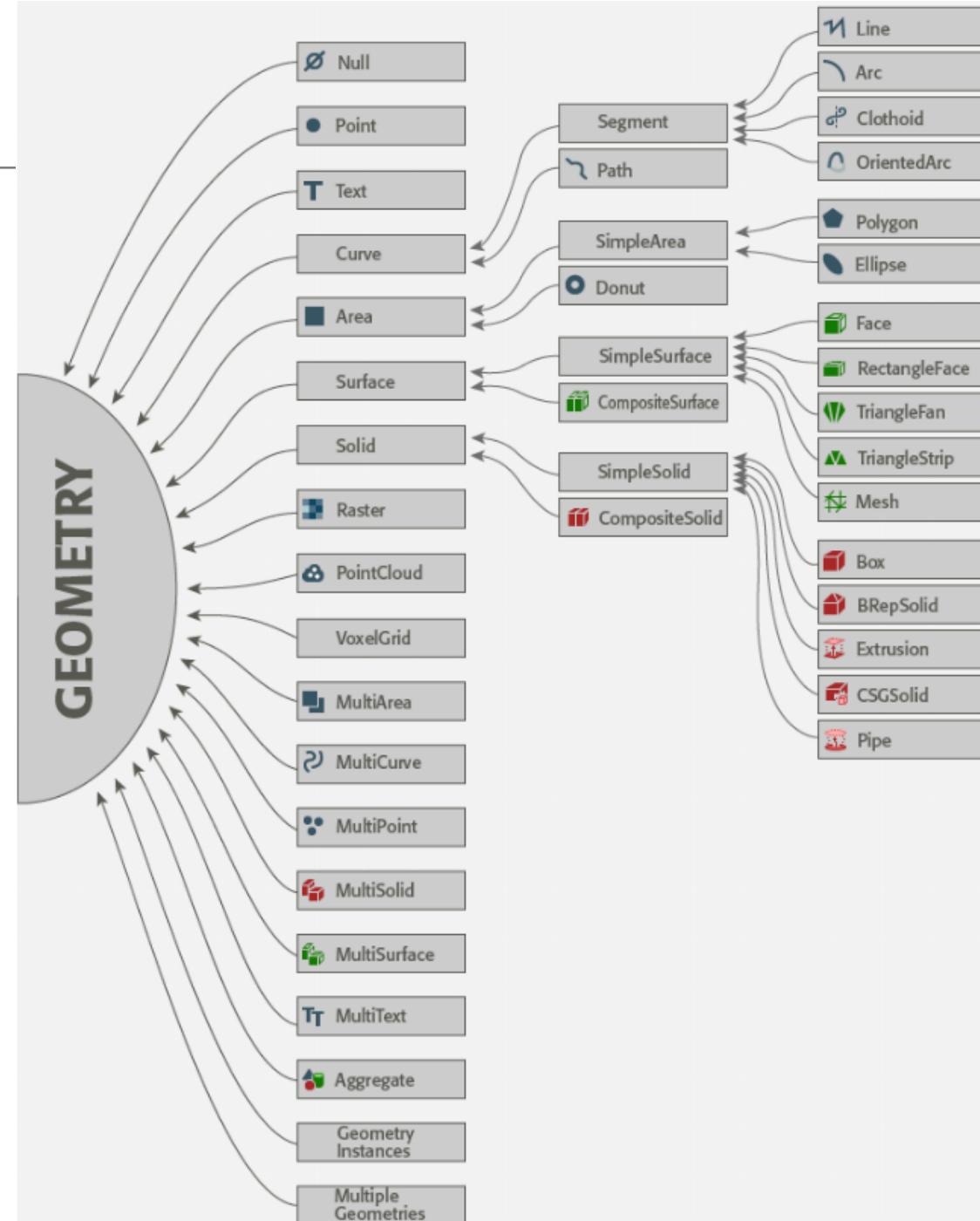
# Geometry and Type

FME provides a comprehensive geometry model that includes everything from the simplest geometry to the most complex

There are two attributes to be aware of when working with spatial data:

*fme\_type* and *fme\_geometry*

Both of these relate to the geometry of a feature.



# Geometry and Type



Examples:

Simple Polygon



fme\_geometry = *fme\_polygon*  
fme\_type = *fme\_area*

Donut Polygon



fme\_geometry = *fme\_donut*  
fme\_type = *fme\_area*

Point



fme\_geometry = *fme\_point*  
fme\_type = *fme\_point*

Line



fme\_geometry = *fme\_line*  
fme\_type = *fme\_line*

Text



fme\_geometry = *fme\_point*  
fme\_type = *fme\_text*

Arc



fme\_geometry = *fme\_point*  
fme\_type = *fme\_arc*

Ellipse



fme\_geometry = *fme\_point*  
fme\_type = *fme\_ellipse*

# Multis

- A Multi is a **collection** of geometries of a **single type** that is treated as a single unit.
- Multis **cannot** be nested or have a **hierarchy within them**.

Examples:

**MultiPoint** (*IFMEMultiPoint*)

a defined collection of Point geometries

fme\_geometry = *fme\_aggregate*

fme\_type = *fme\_point*

**MultiCurve** (*IFMEMultiCurve*)

a defined collection of Curve geometries

fme\_geometry = *fme\_aggregate*

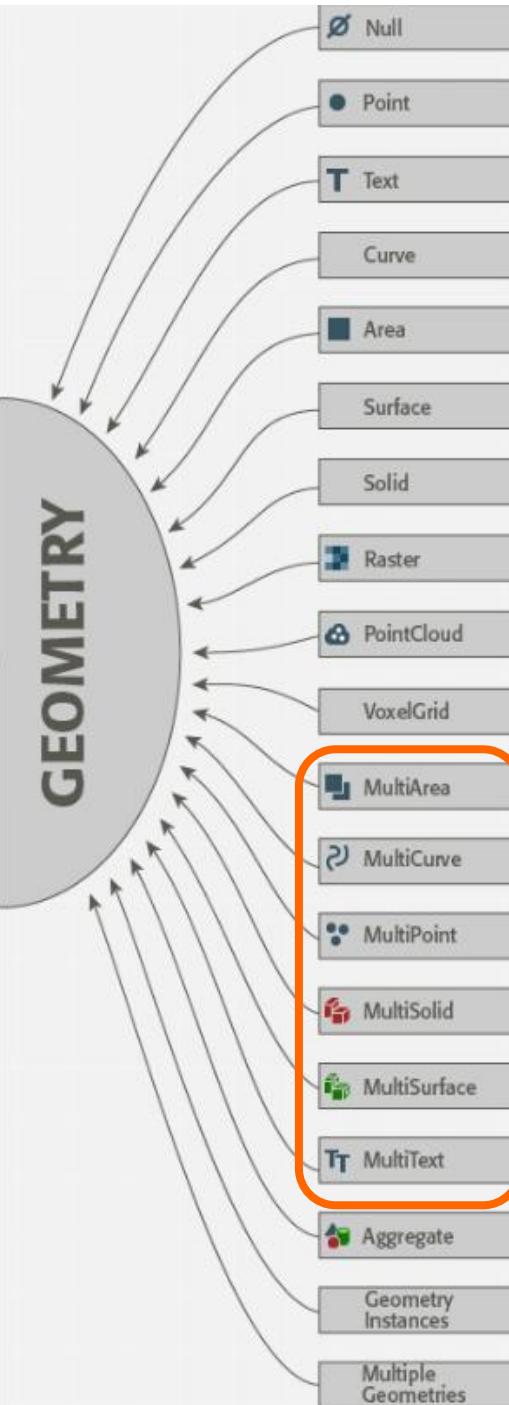
fme\_type = *fme\_line*

**MultiArea** (*IFMEMultiArea*)

a defined collection of Area geometries

fme\_geometry = *fme\_aggregate*

fme\_type = *fme\_area*



# Aggregates

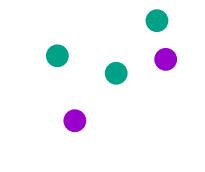
- An aggregate is a collection of geometries **of any type** that is treated as a single unit.
- Aggregates may or **may** not be homogenous and/or **hierarchical**:

- **Hierarchical Geometries** - a collection of geometries that may, in turn, contain other collections (an aggregate that contains aggregates)

- **Homogeneous Aggregates** - made up of features of the same fme\_type

Attributes:

- fme\_geometry = *fme\_aggregate*
    - fme\_type = *<type of contained geometries>* e.g. *fme\_point*



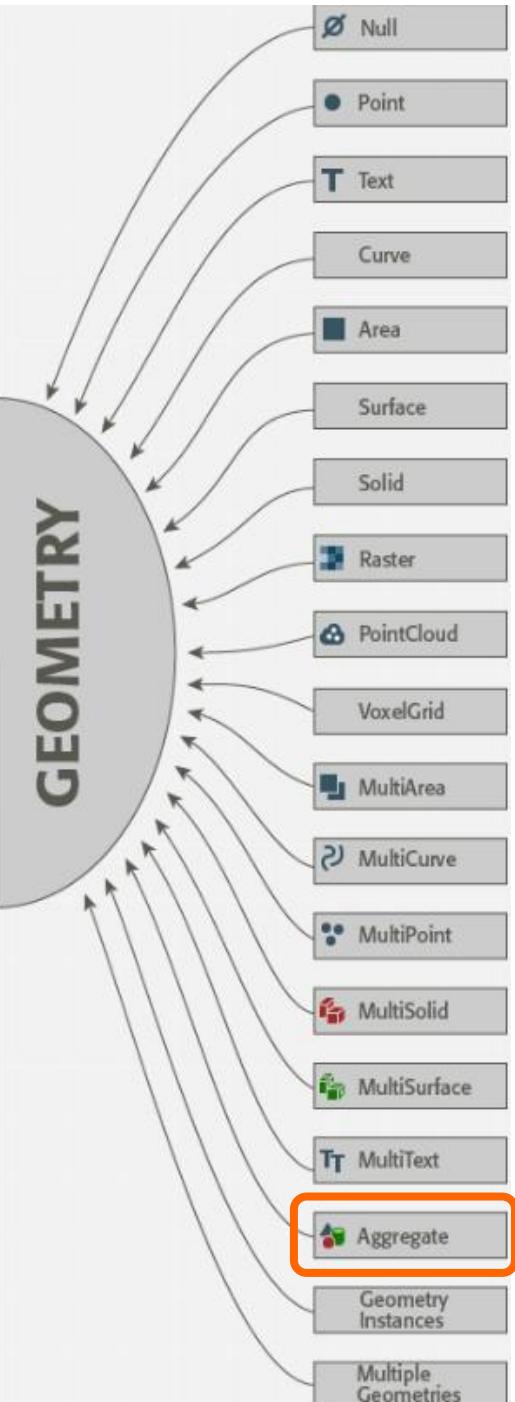
Note: some types of homogeneous aggregates may also be represented as multi geometries. For example, a collection of only Points could equally be represented as a MultiPoint Geometry.

- **Non-Homogeneous Aggregates** - made up of features of differing geometry types.

e.g. a collection of point, line, and polygon geometries within an aggregate

Attributes:

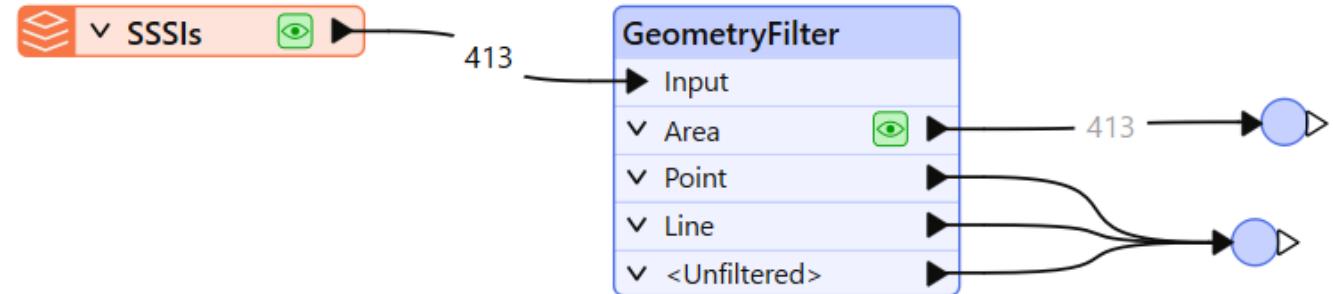
- fme\_geometry = *fme\_aggregate*
    - fme\_type = *fme\_collection*



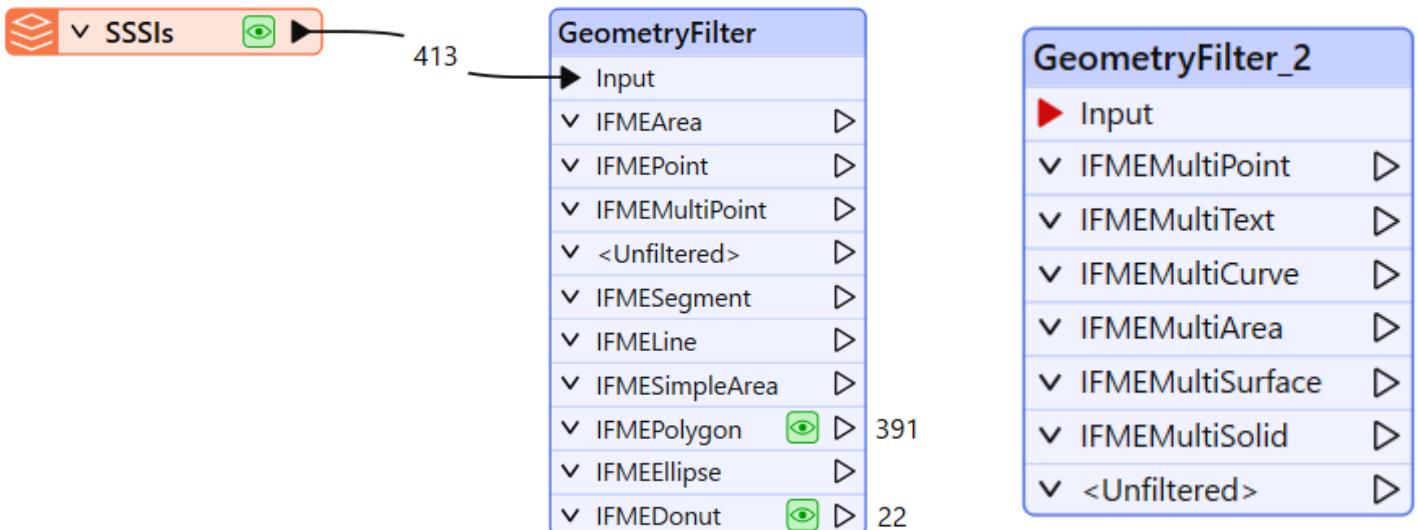
# The GeometryFilter transformer



- Filters features based on Geometry Type
- User can select which geometry types to filter by



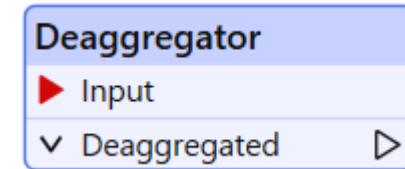
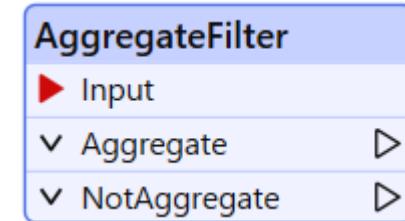
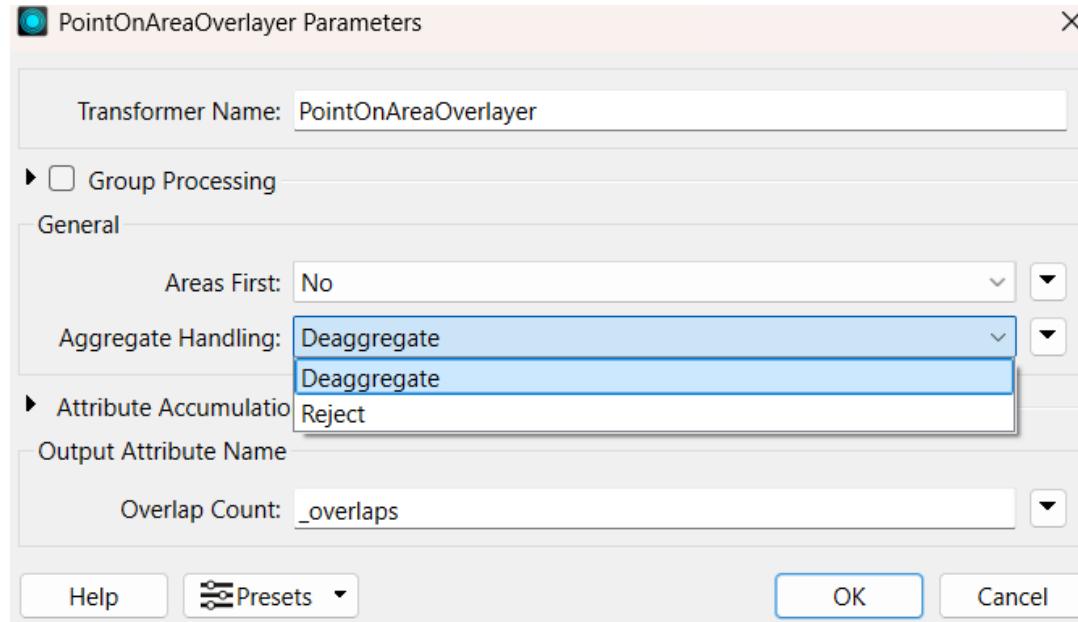
- switch to **detailed mode** to filter:
  - by *fme\_geometry*
  - for Multis\*/Aggregates



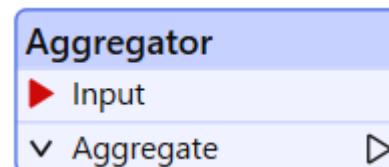
# Working with Aggregates



- Are you expecting aggregate features within the data?  
How would they behave/impact your workflow and results?  
Do you need to check for them or deaggregate them?
- Some transformers will need to deaggregate or reject aggregates:



- Do you want to create aggregate features?



# Example – Aggregates



C:\FMEModularData\Data\Zoning\Zones.tab

Visual Preview x

Table

Zones

ZoneName ZoneCategory

1 M-2	Industrial
2 M-1	Industrial
3 M-2	Industrial
4 M-1	Industrial

Search in any colu 1 selected / 416 row(s)

NORTH V

Lions Gate Lower Capilano

Inner Vancouver Harbour

English Bay

Vancouver

12TH AVE

## Aggregated by Zone Category

Aggregator\_Aggregate Columns...

ZoneCategory

- 1 Industrial
- 2 Historic Area
- 3 Commercial
- 4 Light Industrial
- 5 One Family Dwelling
- 6 Two Family Dwelling
- 7 Multiple Family Dwelling
- 8 Comprehensive Development

1 selected / 8 row(s)

Feature Information

Property

- Feature Type
- Coordinate System
- Dimension
- Number of Vertices
- Min Extents
- Max Extents

Attributes (3)

IFMEEAggregate (237 Parts)

## Dissolved & Aggregated by Zone Category

Aggregator\_Aggregate Columns...

ZoneCategory

- 1 Industrial
- 2 Historic Area
- 3 Commercial
- 4 Light Industrial
- 5 One Family Dwelling
- 6 Two Family Dwelling
- 7 Multiple Family Dwelling
- 8 Comprehensive Development

1 selected / 8 row(s)

Feature Information

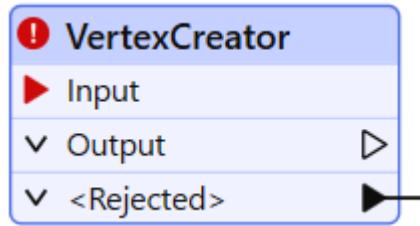
Property

- Feature Type
- Coordinate System
- Dimension
- Number of Vertices
- Min Extents
- Max Extents

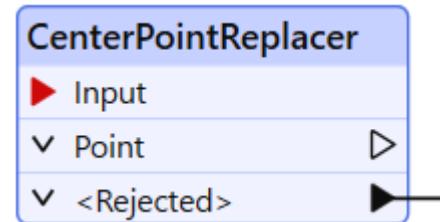
Attributes (3)

IFMEMultiArea (73 Parts)

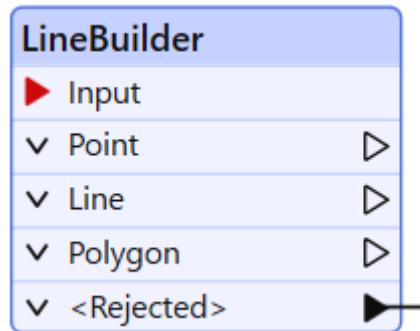
# Creating Geometry – getting started



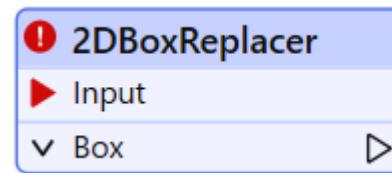
Creates a point from x,y coordinates



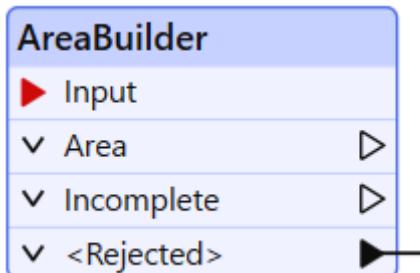
Replaces polygons with points



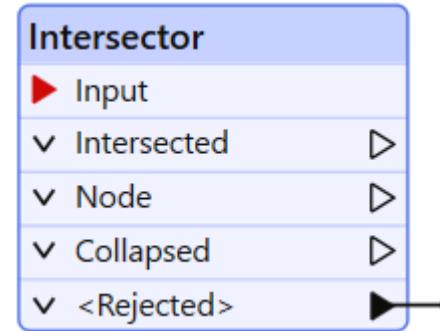
Creates lines from points  
- connects point features in the order they enter, forming lines or polygons



Creates a box from coordinates



Creates polygons from lines  
- uses topologically connected linework to create polygons

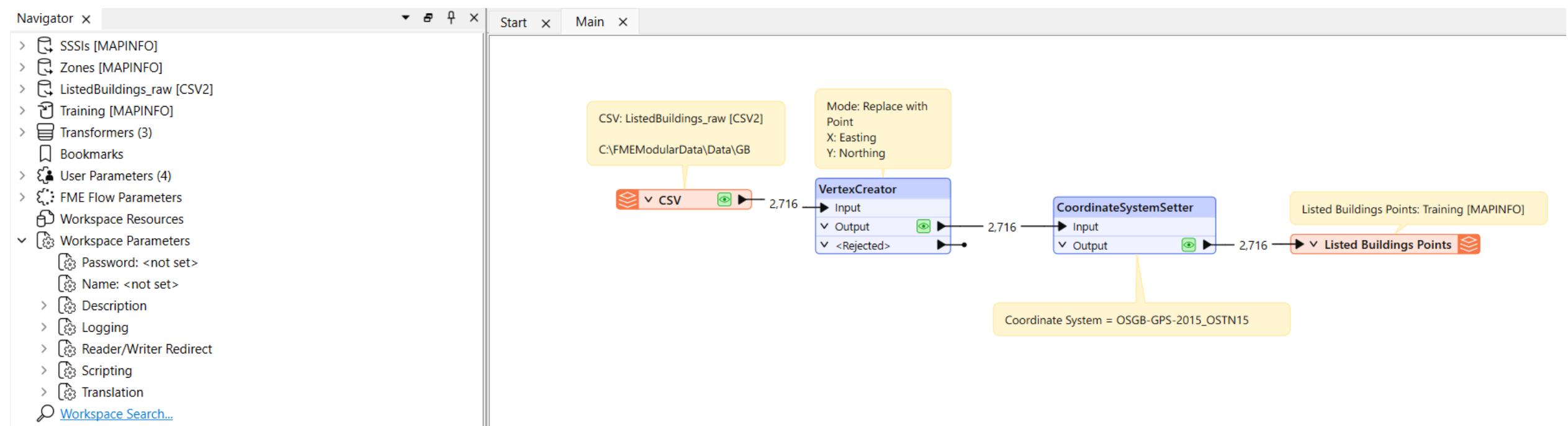


Break lines and polygons wherever an intersection occurs and creates nodes at those locations

# Demo – create points from XY coordinates



- Use Listed buildings csv



# Exercise 3.1



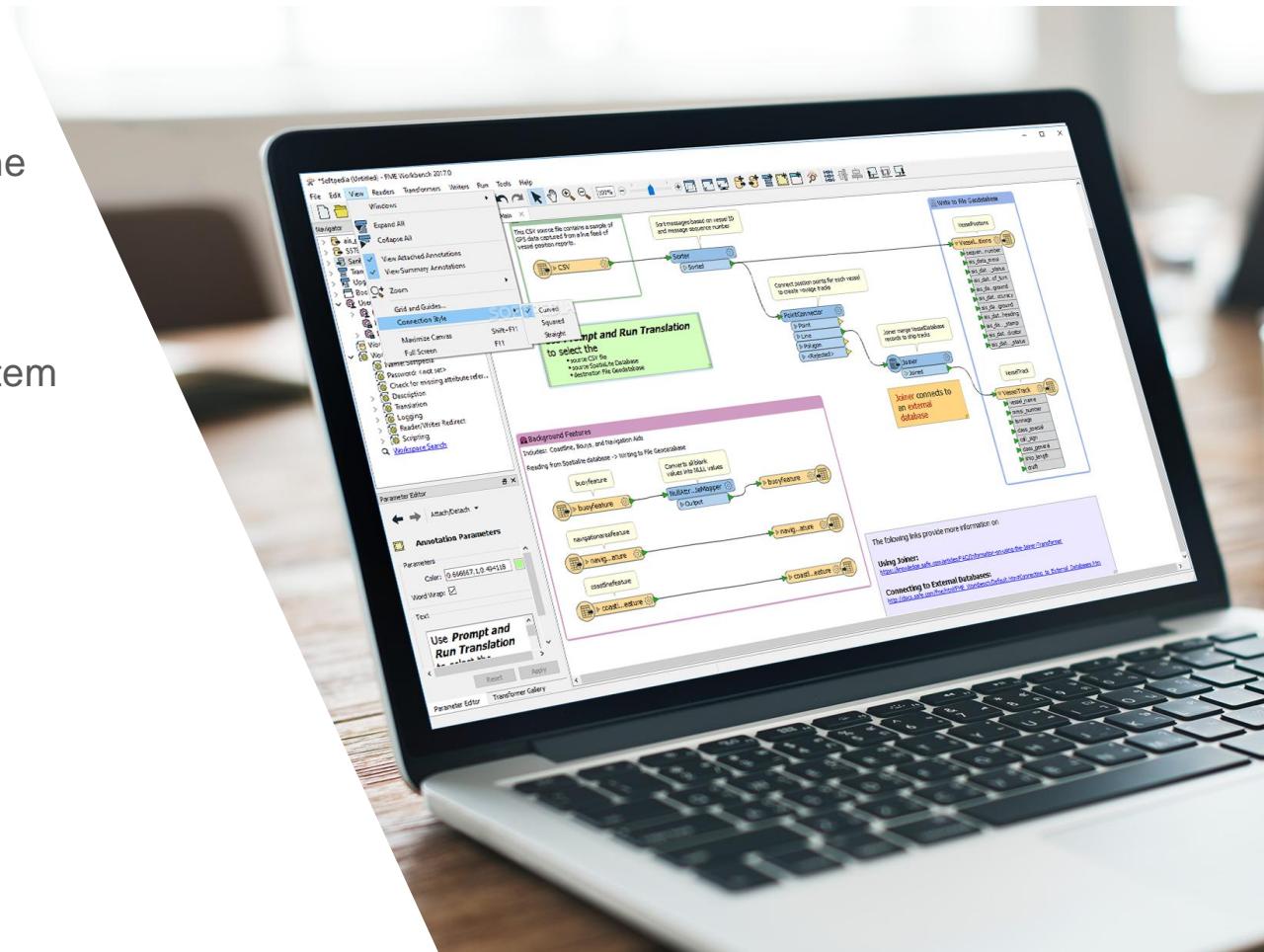
## Spatial Data Concepts

- The Community Engagement team has sourced a spatial dataset containing crime incidents and want to load it into one of their systems.
- However, the data needs to be structured in a certain way before it can be consumed by their system:
  - the data has to be in Lat/Long WGS84 coordinate system
  - contain only points
  - include the coordinates stored as attributes

Data – Crime Incidents (MapInfo TAB)

Starter workspace – none

**Goal** - Translate the Crime Incidents MapInfo TAB data to GeoPackage format, whilst also checking geometry type and reprojecting to the required coordinate system.



# Symbology and Styling

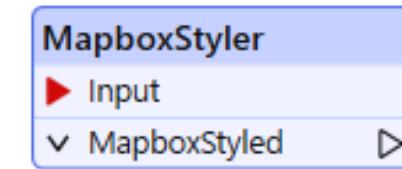
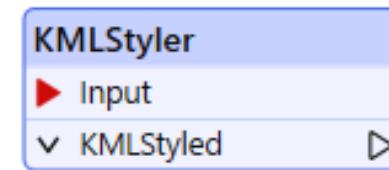
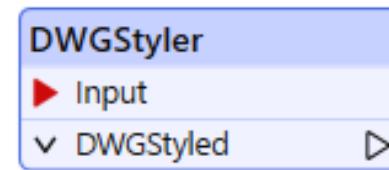


Connect. Transform. **Automate**

# Symbology and styling in FME



- Format attributes are useful for reading and applying styling to files
- Styler transformers are available for a number of formats:



# Spatial Joins and Filtering

# Connect. Transform. Automate

# Analyze Spatial Data

---



Conduct basic spatial analysis by comparing the location of features or filtering data based on location.

## What Is Spatial Joining and Filtering?

- **Spatial Join** - *combines* information between two datasets based on their spatial location.

For example, you might have a dataset of incident points and a polygon dataset of postcode boundaries. You could use how the incident points fall within the postcode polygons to obtain attributes from the postcode dataset and add them to the incident points dataset.

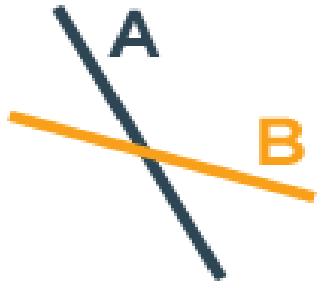
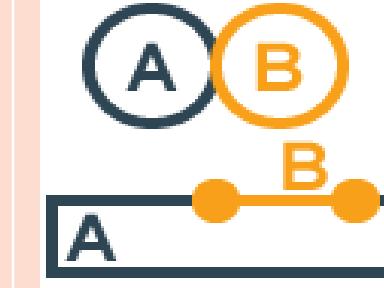
- **Spatial Filtering** - *separates* features based on their spatial location.

For example, you might have a dataset of incident points and a polygon representing an authority area. You could filter your incident data to only include incident points within the authority boundary.

# Spatial Tests



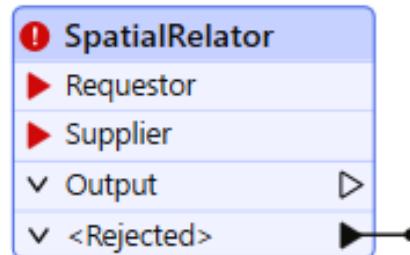
These tests can be performed in transformers using spatial filters and joins

Intersects	Contains	Equals	Disjoint From	Within	Touches
					
	*Inverse of Within <i>object A contains object B</i>			*Inverse of Contains <i>object A is within object B</i>	

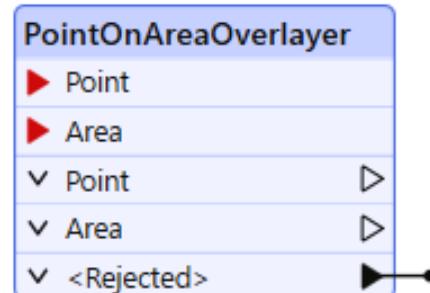
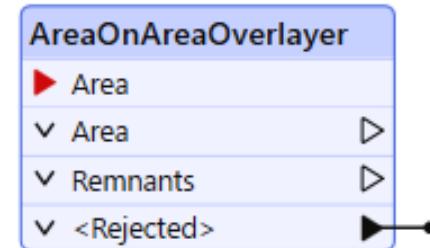
# Overview of spatial joiners and filtering transformers



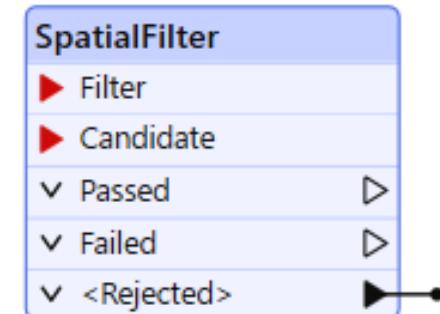
SpatialRelator determines the spatial relationship between geometry and can optionally join data



There are a number of overlayers which handle different forms of overlay



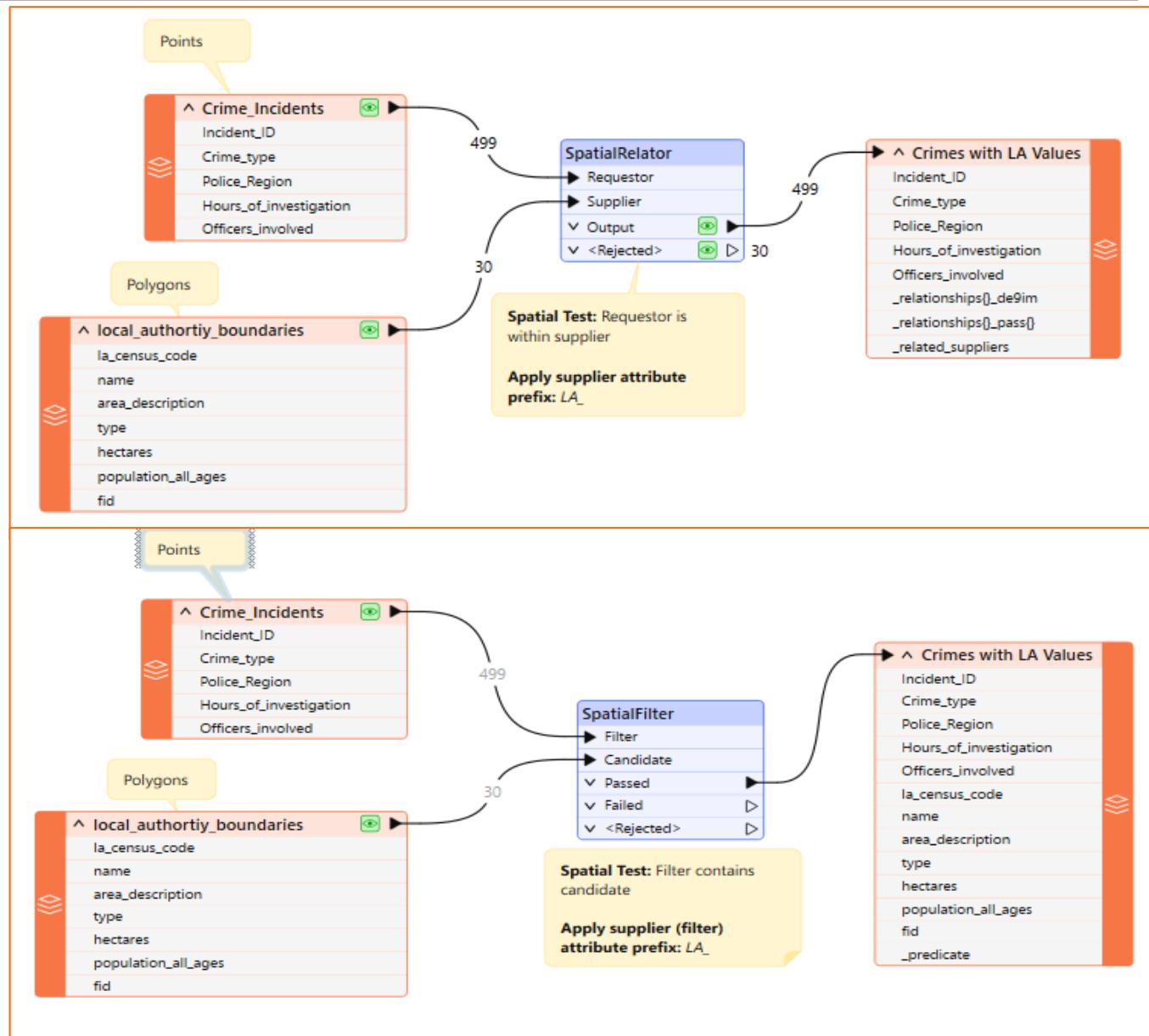
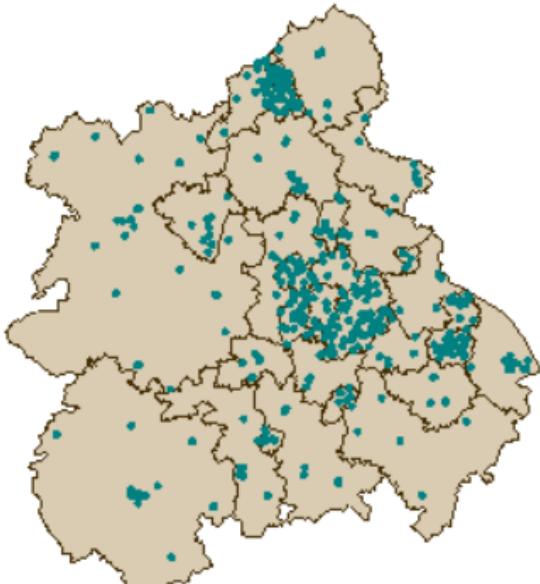
Filters according to spatial relationship but can also be used to merge attributes



# Example - Spatial Join



- To each Crime Incident add the attributes of the Local Authority it falls within.



# Exercise 3.2



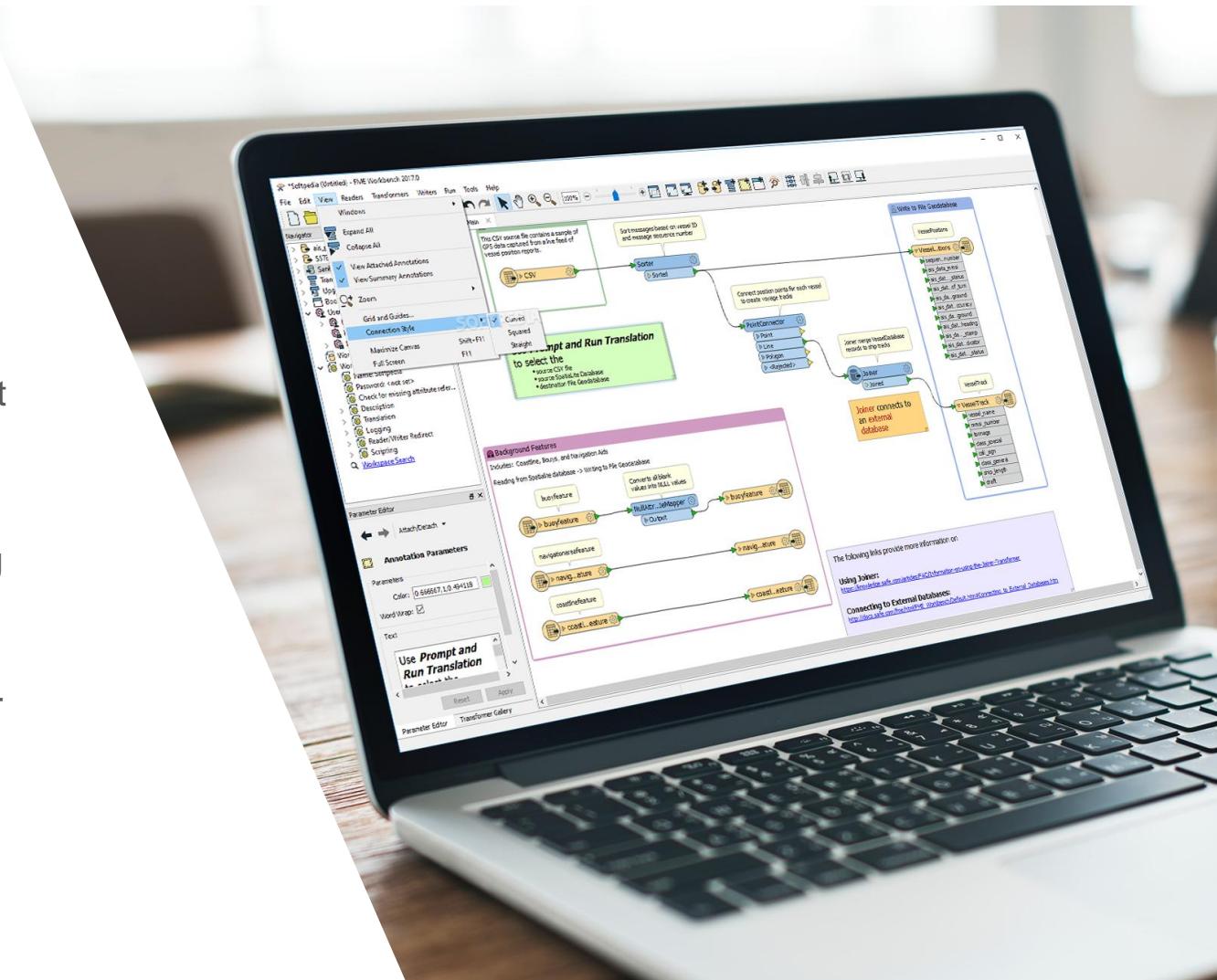
## Spatial Join

- You have been given the task of preparing a dataset for the Corporate Strategy and Environmental Services teams. They have requested a breakdown of regional fly-tipping incidents to local authority level.
- The task is to join two spatial datasets and generate a count of how many fly-tipping incidents occurred within each local authority area.
- Secondary to that we will output the results to KML format for use in GoogleEarth.

**Data – Local Authority Boundaries (Geopackage), FlyTipping (MapInfo TAB)**

**Starter workspace – C:\FMEModularData\Workspaces\3.02-Spatial-SpatialJoins-Begin.fmw**

**Goal - Create a new version of the local authority boundaries that includes the required fly-tipping incident counts.**

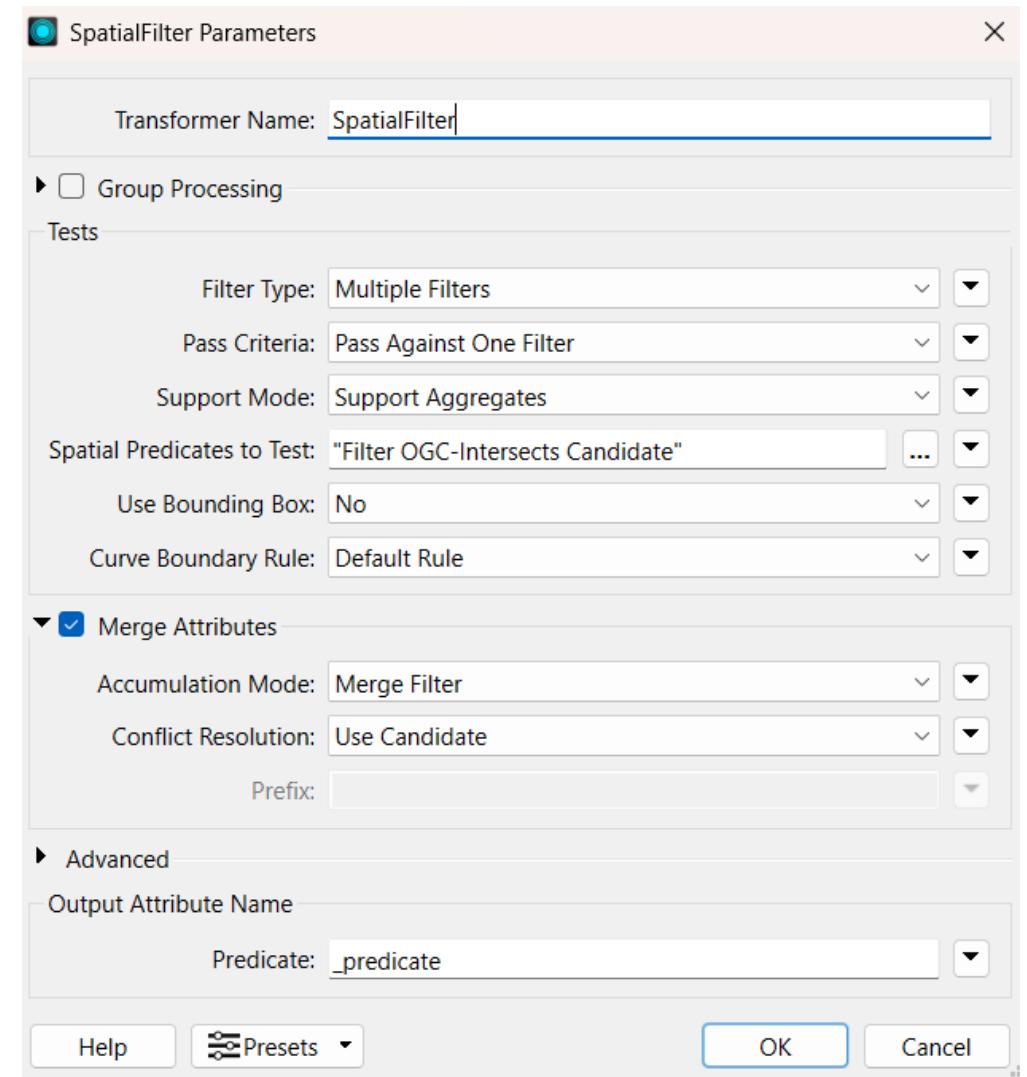
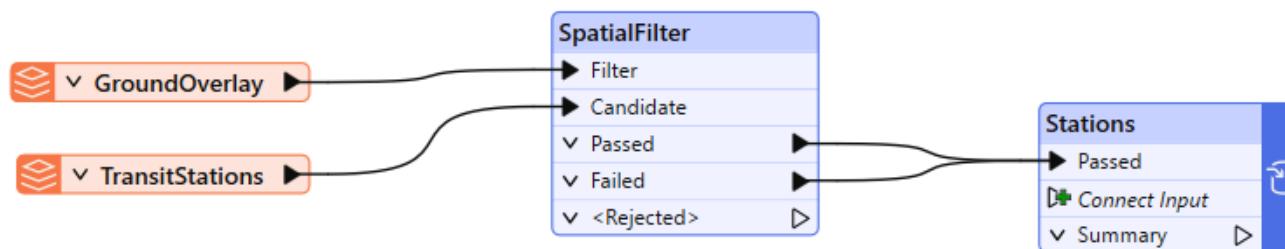


# SpatialFilter Transformer



## Typical Uses:

- **Routing features**, depending on whether a defined spatial relationship is true or false
- Performing **quality control** on a dataset by checking for expected spatial relationships with another dataset
- Performing a spatial join to **transfer attributes** from one feature to another based on their spatial relationship

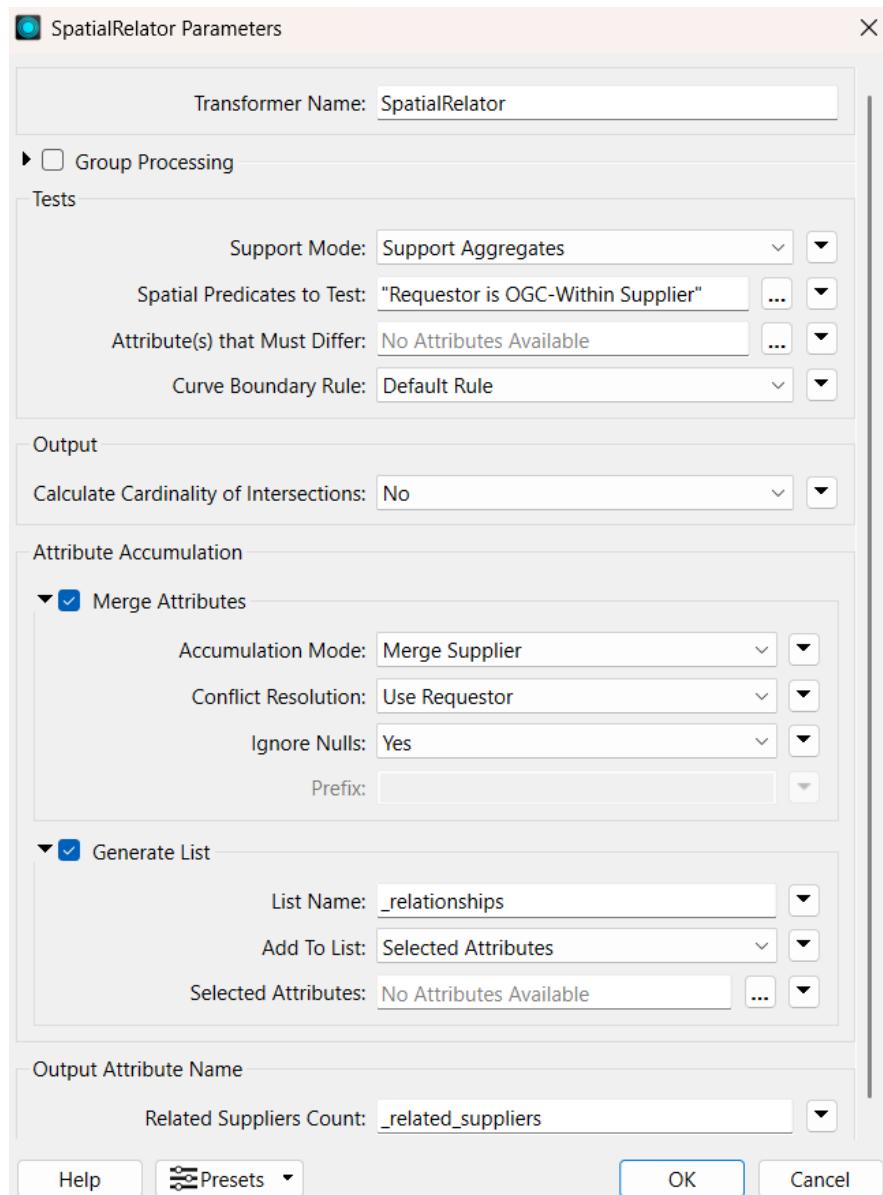
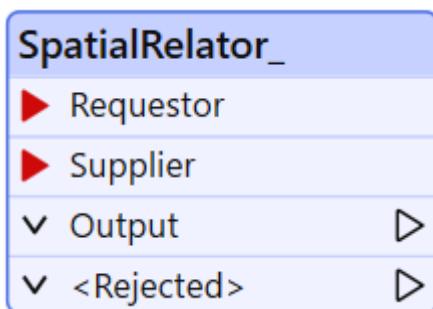


# SpatialRelator Transformer



## Typical Uses:

- Identifying what kind of **spatial relationship(s)** exist between features
- Performing a spatial join to **transfer attributes** from one feature to another based on their spatial relationship



## How does it compare to SpatialFilter?:

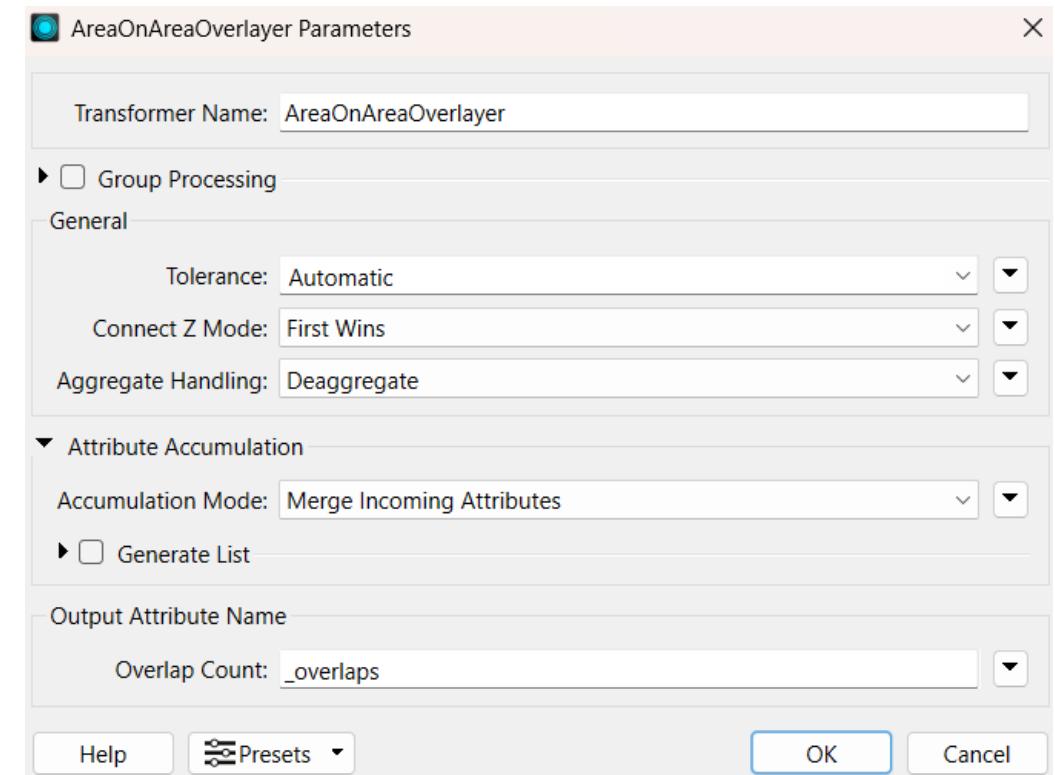
- Pros:
  - Includes more functionally/configuration options including *List* attribute generation and creation of 'Count' attribute
- Cons:
  - More resource intensive
  - More complex to use

# Overlayer transformers



- There are several different "overlayer" transformers, each handling a different form of overlaying.
- Overlayers look for the intersection of two streams of spatial features (**performing spatial tests**). They **return both streams** in full but add intersection information as **attributes**.
  - Merges attributes
  - Includes an '`_overlaps`' count attribute
  - List generation

The screenshot shows the FME Workbench interface. In the top left, there is a search bar with the text "overlayer". Below it, a navigation bar includes buttons for "Transformer" (highlighted in blue), "Reader", "Writer", "Official", "Verified", and "Community". A sidebar on the left lists various Overlayer transformer types: PointOnAreaOverlayer, AreaOnAreaOverlayer, PointOnLineOverlayer, LineOnAreaOverlayer, LineOnLineOverlayer, PointOnPointOverlayer, VectorOnRasterOverlayer, and SurfaceOnSurfaceOverlayer. The main panel displays the "PointOnAreaOverlayer" transformer details. It is marked as "Official" and published by "Safe Software". The "Category" is listed as "Filters and Joins, Spatial Analysis". The "Rank" is 63. A "Link" button is available for more information. At the bottom of the main panel, there is a "Description" section.



*The Overlayers are easier to use than the SpatialRelator and SpatialFilter. But do provided slightly different output.*

# Spatial filtering within transformers - Advanced



- Here the feature reader is being used to only read in data within a particular area
- Reads from external dataset and constrained by the defined spatial test
- Output from schema port is the queried feature

The screenshot shows the FME Workbench interface with the FeatureReader transformer selected. The workspace contains a single FeatureReader transformer, which has a 'Parks' source connected to its 'Schema' port.

**FeatureReader Parameters:**

- Transformer Name: FeatureReader
- Reading:**
  - Reader:
    - Format: Precisely MapInfo TAB (MAPINFO)
    - Dataset: C:\FMEModularData\Data\Parks\Full\_Trees.tab
    - Parameters...
    - Coord. System: Read from source
  - Constraints:
    - Feature Types to Read: <All Feature Types>
    - WHERE Clause: (empty)
    - Spatial Filter: Initiator OGC-Contains Result
    - Clip to Initiator Envelope:
    - Max Features to Read: (empty)

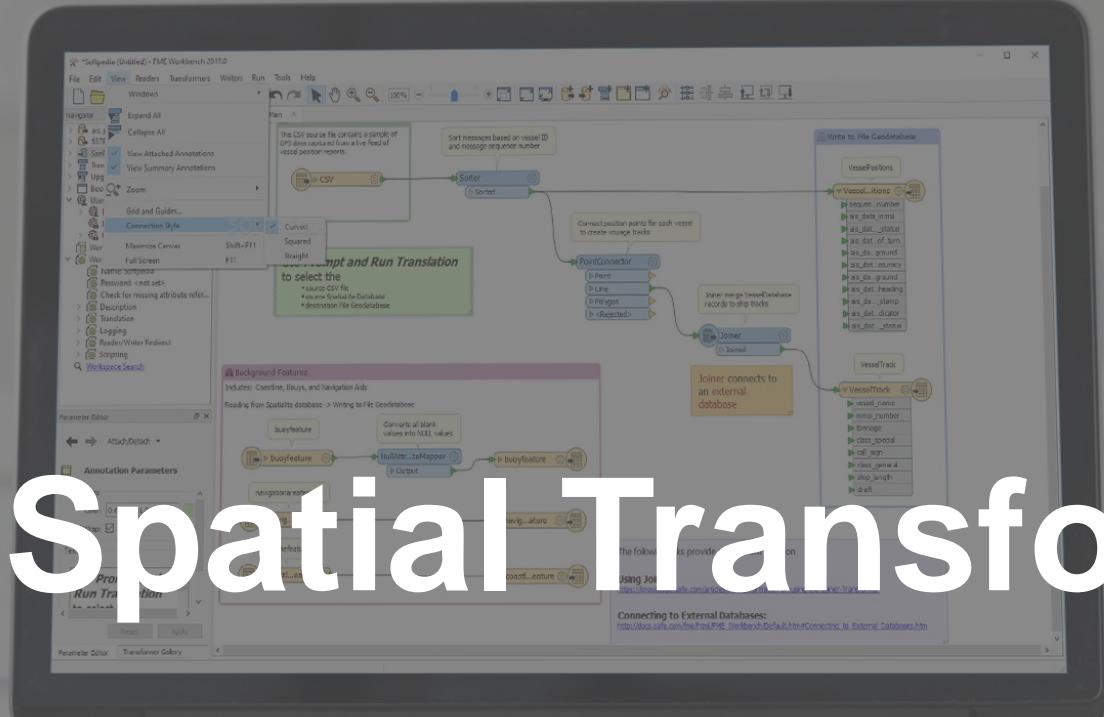
A red arrow points from the 'Spatial Filter' dropdown in the 'Constraints' section of the parameters to the 'OGC-Contains' dropdown in the 'Attribute and Geometry Handling' section of the output properties.

**Output:**

- Output Ports:** (No ports defined)
- Attribute and Geometry Handling:**
  - Accumulation Mode: Merge Initiator and Result
  - Conflict Resolution: Use Result
  - Ignore Nulls: No
  - Prefix: (empty)
  - Geometry: Use Result
- <Generic> Port:** (No generic port defined)

# Useful Spatial Transformers

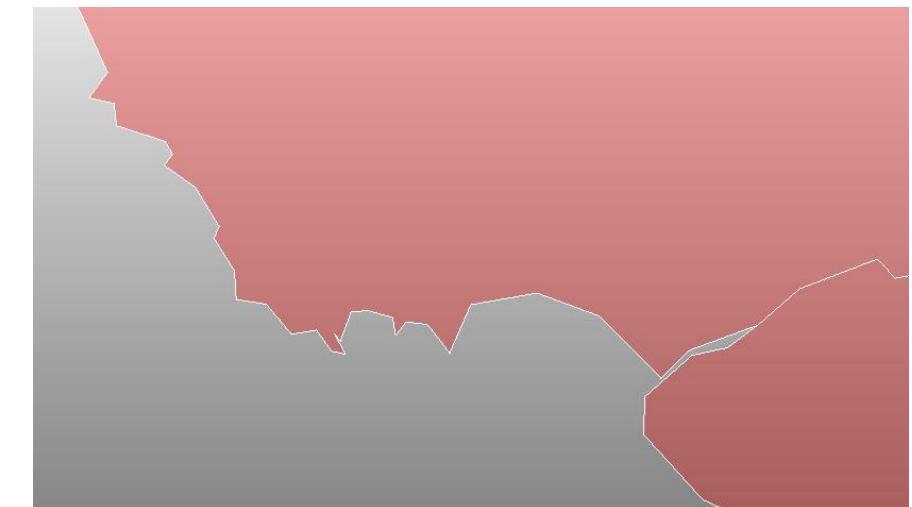
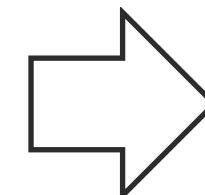
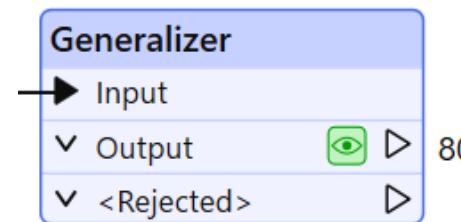
Connect. Transform. **Automate**



# Generalising



- Sometimes spatial files contain excess spatial information for the scale they are needed at
- Spatial files can be generalised to reduce the detail held in the data
- The Generalizer transformer has 4 types of algorithm:
  - **Generalising algorithms:** Reduce the density of coordinates by removing vertices
  - **Smoothing algorithms:** Determine a new location for each vertex
  - **Measuring algorithms:** Calculate the location of points, and return a list of these points
  - **Fitting algorithms:** Replace the original geometry completely, with a new feature fitted to a specified line



# NeighborFinder transformer



- Carries out analysis based on proximity relationship

In this example the NeighborFinder is being used to identify the closest fire hall to each transit station:

The screenshot shows the 'General' settings for the NeighborFinder transformer. A red arrow points from the 'Number of Neighbors to Find:' field, which is set to 1. Another red arrow points from the 'Merge Attributes' checkbox under the 'Attribute Accumulation' section, which is checked.

**General**

Number of Neighbors to Find: 1

Maximum Distance:

Insert Vertex On Base Feature: No

Treat Measures as: Continuous

Treat Polygons As: Lines

Replace Measures/Z with Candidate: Yes

**Attribute Accumulation**

Merge Attributes

Accumulation Mode: Merge Candidate

Conflict Resolution: Use Base

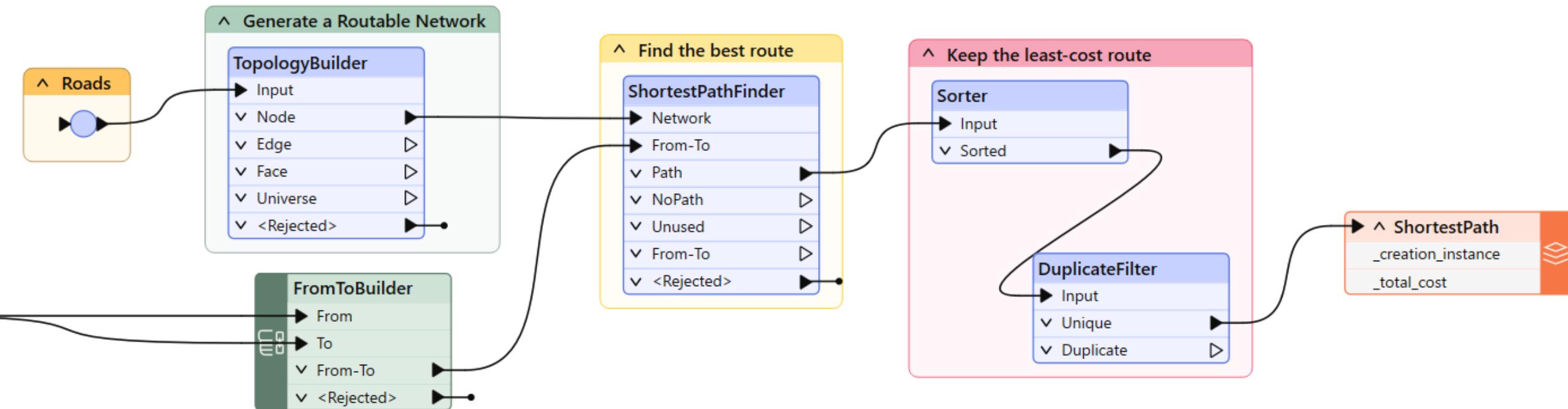
Prefix:

Generate List

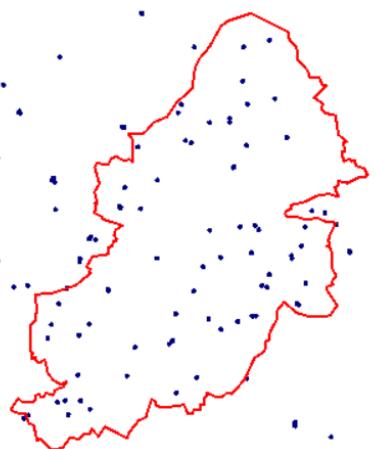
# Route Analysis



- Carry out routing analysis by creating topological networks, source & destination nodes, then identifying the shortest paths:

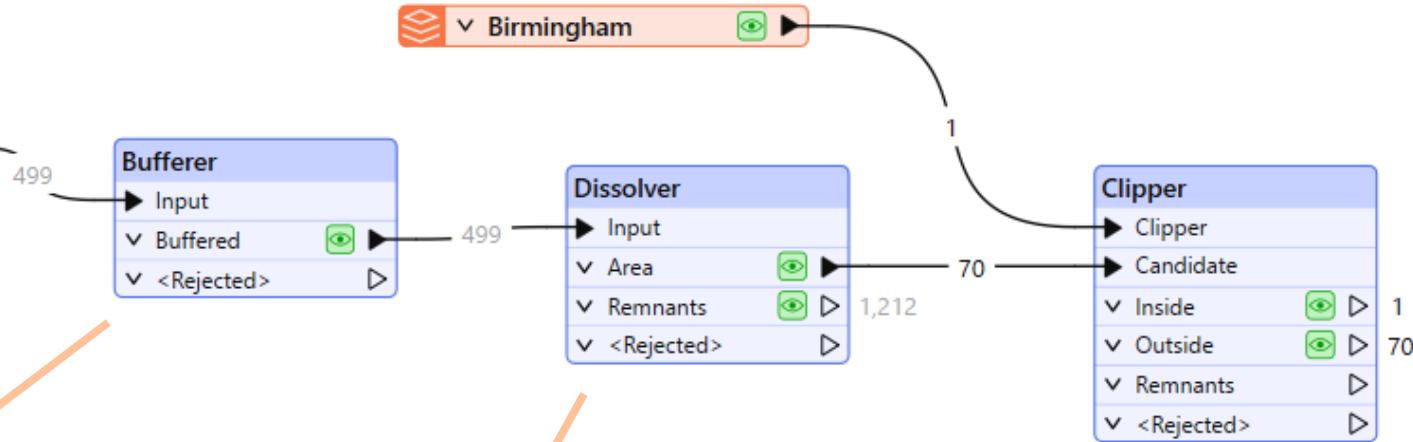


# Buffer, Dissolve and Clipping

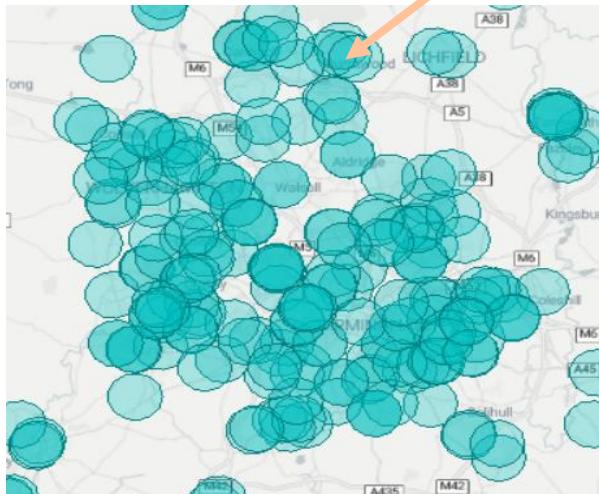


Crime Points and Birmingham Polygon

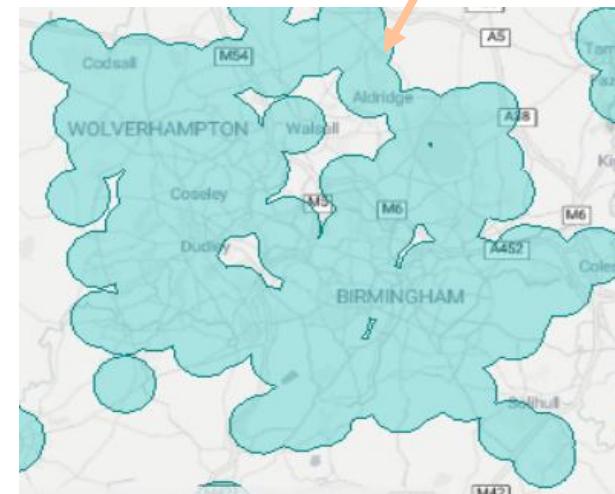
Crime_Incidents	
Incident_ID	499
Crime_type	
Police_Region	
Hours_of_investigation	
Officers_involved	



Buffer crime points creating buffer polygons



Dissolve buffer polygons into single crime zone polygon



Clip crime zone polygon to Birmingham boundary



# Useful Transformers for Validating Spatial Data



!	<b>GeometryFilter</b>
▶	Input
▼	<Unfiltered>

	<b>Deaggregator</b>
▶	Input
▼	Deaggregated

!	<b>GeometryValidator</b>
▶	Input
▼	Passed
▼	Failed
▼	Repaired
▼	IssueLocations
▼	InvalidParts
▼	<Rejected>

!	<b>AreaGapAndOverlapCleaner</b>
▶	Input
▼	Repaired
▼	Remnants
▼	<Rejected>

!	<b>SpikeRemover</b>
▶	Input
▼	Unchanged
▼	Changed
▼	Removed
▼	<Rejected>

!	<b>ChangeDetector</b>
▶	Original
▶	Revised
▼	Updated
▼	Inserted
▼	Deleted
▼	Unchanged

!	<b>DuplicateFilter</b>
▶	Input
▼	Unique
▼	Duplicate

# Geometry Transformers



- [2DArcReplacer](#)
- [2DBoxReplacer](#)
- [2DEllipseReplacer](#)
- [2DForcer](#)
- [2DGridAccumulator](#)
- [2DGridCreator](#)
- [3DArcReplacer](#)
- [3DForcer](#)
- [3DInterpolator](#)
- [3DRotator](#)
- [AffineWarper](#)
- [AnchoredSnapper](#)
- [AngleConverter](#)
- [ArcEstimator](#)
- [ArcGISGridSnapper](#)
- [ArcPropertySetter](#)
- [ArcStroker](#)
- [AreaAmalgamator](#)
- [AreaBuilder](#)
- [AreaGapAndOverlapCleaner](#)
- [BoundingBoxReplacer](#)
- [CenterlineReplacer](#)
- [CenterPointReplacer](#)
- [Chopper](#)
- [CoordinateRounder](#)
- [CoordinateSwapper](#)
- [CSGBuilder](#)
- [Curvefitter](#)
- [Densifier](#)
- [Displacer](#)
- [Dissolver](#)
- [DonutBridgeBuilder](#)
- [DonutBuilder](#)
- [DonutHoleExtractor](#)
- [EllipsePropertySetter](#)
- [Extruder](#)
- [FaceReplacer](#)
- [Generalizer](#)
- [GeometryCoercer](#)
- [GeometryExtractor](#)
- [GeometryRefiner](#)
- [GeometryRemover](#)
- [GeometryReplacer](#)
- [H3HexagonalIndexer](#)
- [HullReplacer](#)
- [Intersector](#)
- [LabelPointReplacer](#)
- [LineBuilder](#)
- [LineCloser](#)
- [LineCombiner](#)
- [LineExtender](#)
- [MeasureRemover](#)
- [MeasureSetter](#)
- [MeshMerger](#)
- [MeshSimplifier](#)
- [MinimumAreaForcer](#)
- [MinimumSpanningCircleReplacer](#)
- [MultipleGeometrySetter](#)
- [OffsetCurveGenerator](#)
- [Orientor](#)
- [PathBuilder](#)
- [PathSplitter](#)
- [PipeEvaluator](#)
- [PipeReplacer](#)
- [PointCloudSurfaceBuilder](#)
- [PointPropertySetter](#)
- [Rotator](#)
- [RubberSheeter](#)
- [Scaler](#)
- [SherbendGeneralizer](#)
- [Snapper](#)
- [Sniper](#)
- [SolidBuilder](#)
- [SolidDissolver](#)
- [SpikeRemover](#)
- [SurfaceDissolver](#)
- [SurfaceFootprintReplacer](#)
- [SurfaceSplitter](#)
- [TextAdder](#)
- [Tiler](#)
- [TINGenerator](#)
- [Triangulator](#)
- [VertexCreator](#)
- [VertexNormalGenerator](#)
- [VertexNormalRemover](#)
- [VertexRemover](#)

# Spatial Analysis Transformers



- 
- |  |   |
|--|---|
|  <a href="#">AffineWarper</a>             |  <a href="#">NetworkCostCalculator</a>       |
|  <a href="#">AnchoredSnapper</a>          |  <a href="#">NetworkFlowOrientor</a>         |
|  <a href="#">ArcGISGridSnapper</a>        |  <a href="#">NetworkTopologyCalculator</a>   |
|  <a href="#">AreaAmalgamator</a>          |  <a href="#">Offsetter</a>                   |
|  <a href="#">AreaBuilder</a>              |  <a href="#">PointOnAreaOverlayer</a>        |
|  <a href="#">AreaOnAreaOverlayer</a>      |  <a href="#">PointOnLineOverlayer</a>        |
|  <a href="#">BoundingBoxAccumulator</a>   |  <a href="#">PointOnPointOverlayer</a>       |
|  <a href="#">Bufferer</a>                 |  <a href="#">PointOnRasterValueExtractor</a> |
|  <a href="#">CenterlineReplacer</a>       |  <a href="#">SectorGenerator</a>             |
|  <a href="#">CenterPointReplacer</a>      |  <a href="#">ShortestPathFinder</a>          |
|  <a href="#">Clipper</a>                  |  <a href="#">Snapper</a>                     |
|  <a href="#">ContourGenerator</a>         |  <a href="#">SolidDissolver</a>              |
|  <a href="#">DEMDistanceCalculator</a>    |  <a href="#">SpatialFilter</a>               |
|  <a href="#">Displacer</a>                |  <a href="#">SpatialRelator</a>              |
|  <a href="#">Dissolver</a>                |  <a href="#">SpatialSorter</a>               |
|  <a href="#">DonutBridgeBuilder</a>       |  <a href="#">SurfaceBuilder</a>              |
|  <a href="#">FeatureReader</a>          |  <a href="#">SurfaceDissolver</a>          |
|  <a href="#">HullAccumulator</a>        |  <a href="#">SurfaceDraper</a>             |
|  <a href="#">Intersector</a>            |  <a href="#">SurfaceOnSurfaceOverlayer</a> |
|  <a href="#">LineOnAreaOverlayer</a>    |  <a href="#">TopferIndexCalculator</a>     |
|  <a href="#">LineOnLineOverlayer</a>    |  <a href="#">TopologyBuilder</a>           |
|  <a href="#">NeighborFinder</a>         |  <a href="#">VectorOnRasterOverlayer</a>   |
|  <a href="#">NeighborhoodAggregator</a> |  <a href="#">VoronoiCellGenerator</a>      |
|  <a href="#">NeighborPairFinder</a>     |  <a href="#">VoronoiDiagrammer</a>         |

# Exercise 3.3



## Analyze Spatial Data

- The council has asked you to identify addresses within the city that are at risk of flooding.
- You have elevation data (raster DEM), address points, land boundary polygon, and the coastline. You need to use spatial analysis techniques to model risk as a combination of closeness to the shoreline and elevation above sea level

**Data** - Addresses (ESRI Geodatabase), Land Boundary (ESRI Shapefile), Coastline Vancouver (ESRI Shapefile) Canadian Digital Elevation Data (CDED)

**Starting Workspace** - C:\FMEModularData\Workspaces\3.03-Spatial-AnalyzeSpatialData-Begin.fmw

**Goal** - Use spatial analysis techniques of buffering, clipping and spatial joins to identify addresses within the city that are at risk of flooding.



Thanks for joining!

Any questions?



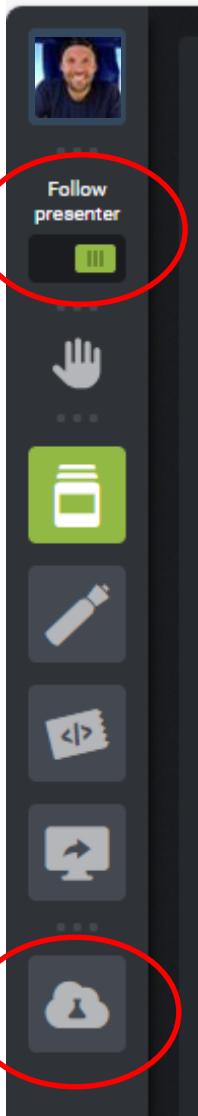
# FME Form Training

## - Module 4

### Workflow Design



# Training Environment



< keep 'Follow presenter' on

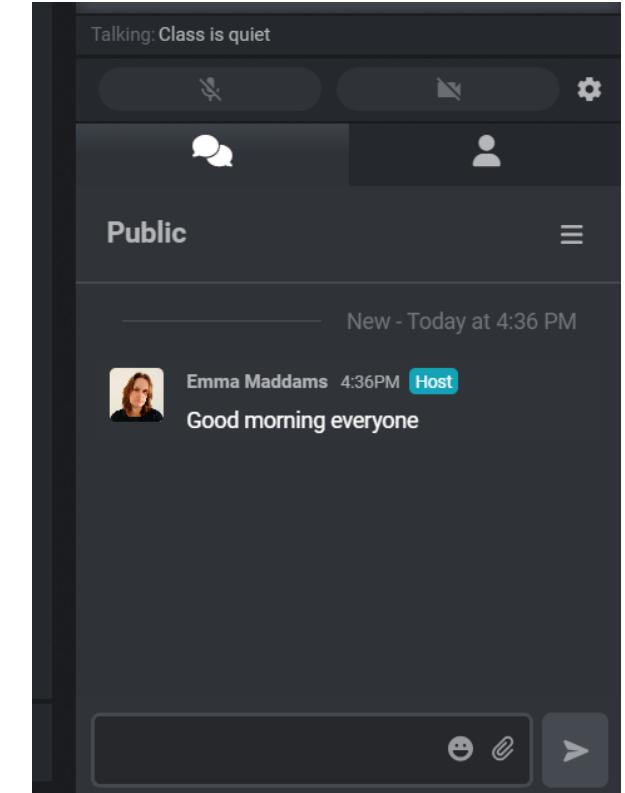
< Lab

need help when using your Lab  
Use either:

- 'Raise hand' or
- 'Lab Assistance'

## Chat

- to everyone
- to trainer



# Training Environment

---



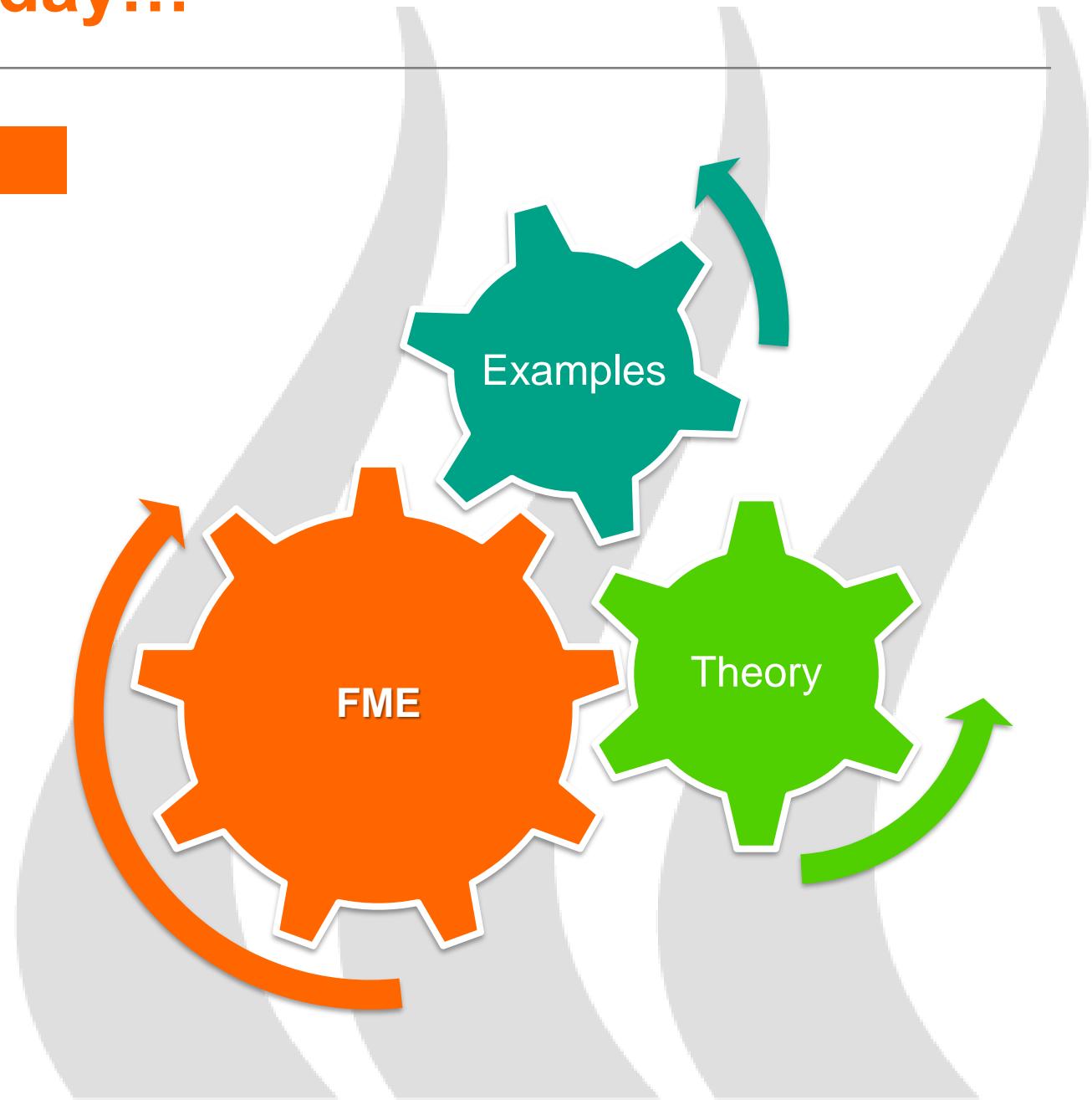
- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

# What we'll be covering today...

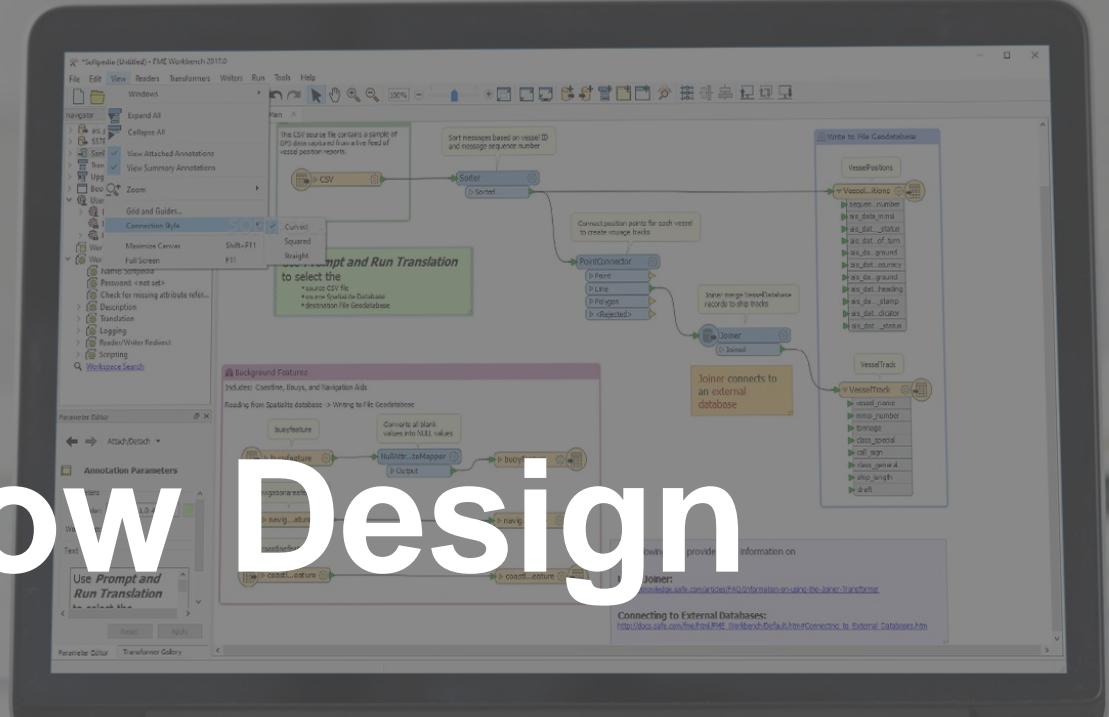
---

## Agenda

- Reading multiple input files
- Single Merged Feature Type
- Fanout
- Working with Zip files
- Error Trapping
  - Rejected Features
  - Logging
- Workspace Runner
- Emailer
- Scheduling Workspace runs



# Workflow Design



Connect. Transform. **Automate**

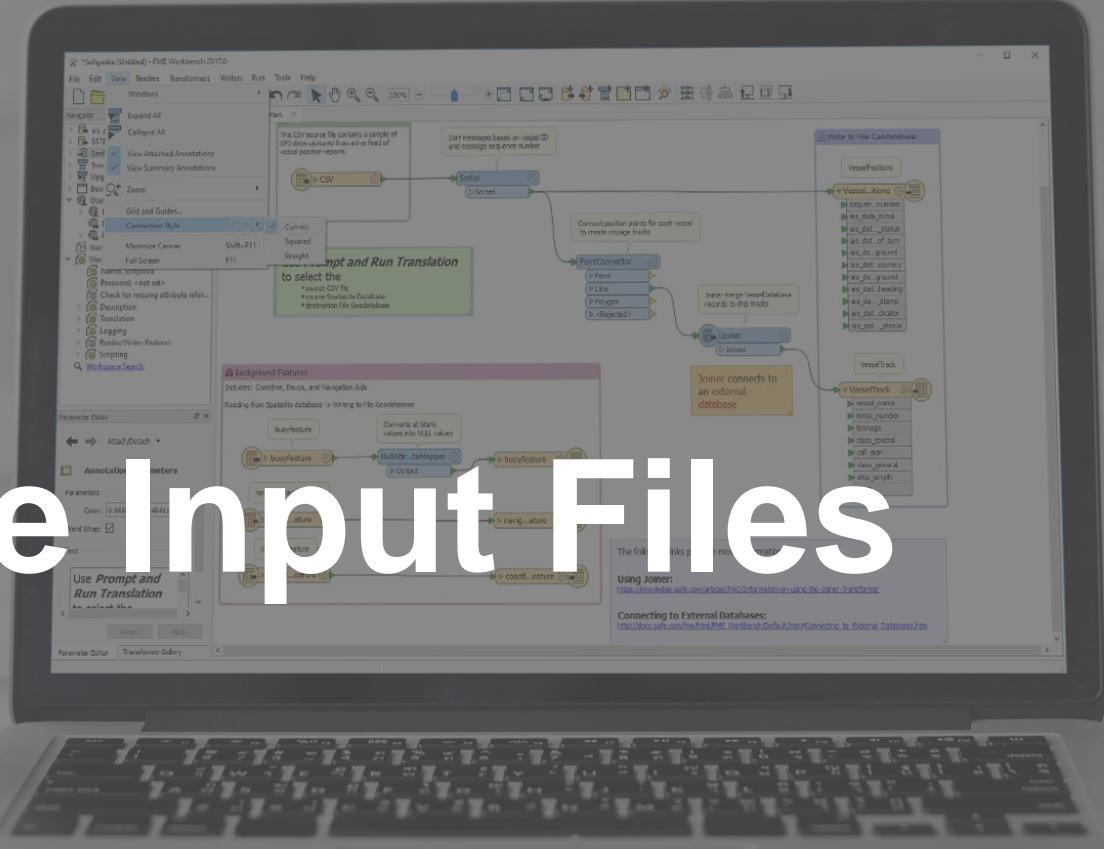


# Workflow Design

---

- Build flexible workflows using techniques that can save you time, make your workspaces more flexible and reusable.
- Learn useful functionality for deploying workspaces to run in production environment.

# Multiple Input Files



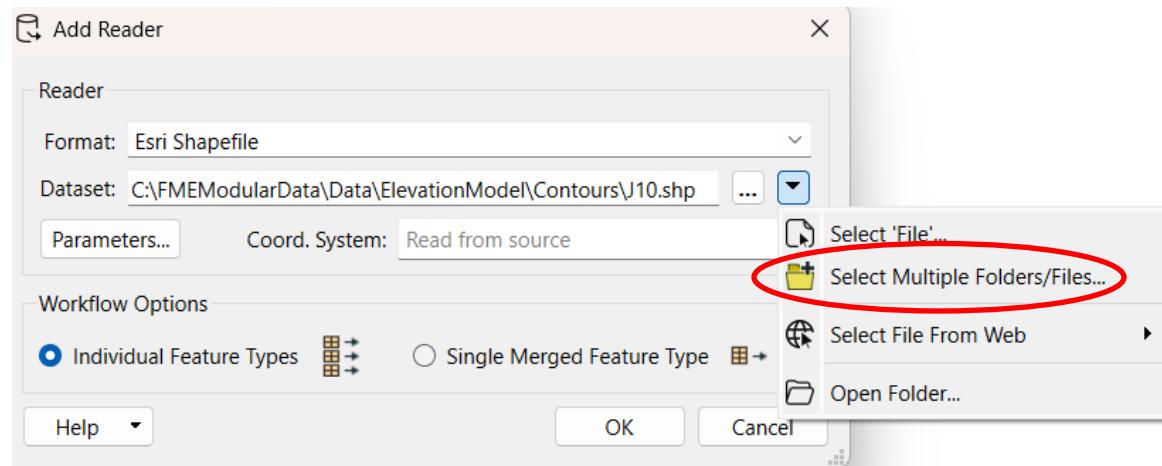
Connect. Transform. **Automate**

# Reading from Multiple Files



It's possible to read multiple datasets or layers using a single reader. There are a few ways to accomplish this goal:

- **Manually select** all individual files – *not very efficient!*
- **Select Multiple Folders/Files** option:



- **Directory and File Pathnames Reader** - an advanced method used in conjunction with a FeatureReader. Great for batch processing data! - we look at this method on our 'Advanced Workflow Design' training module

# Reading from Multiple Files

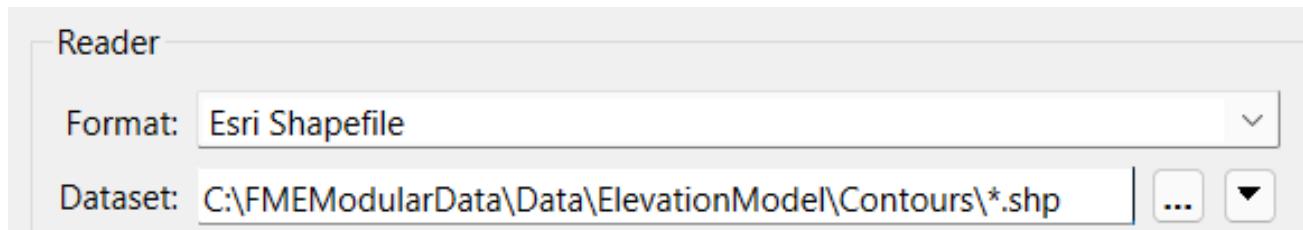


- **Wildcards \***

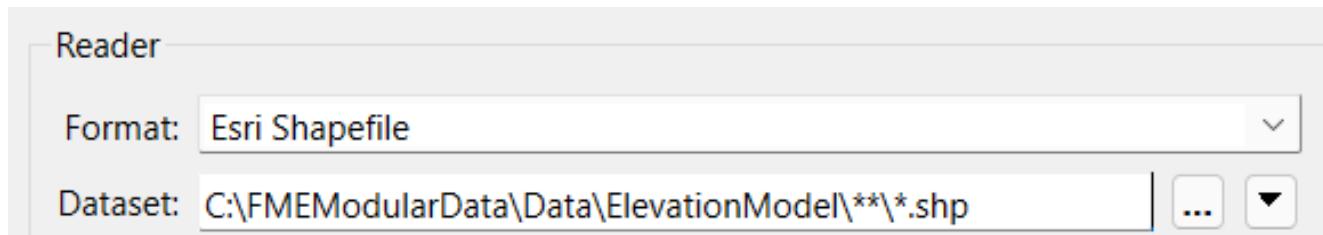
If you need to read multiple files, but you don't know their file name or path in advance. Or you want an easy way to "greedily" accept input datasets.

You can accomplish this by using wildcards in the Source Dataset parameter

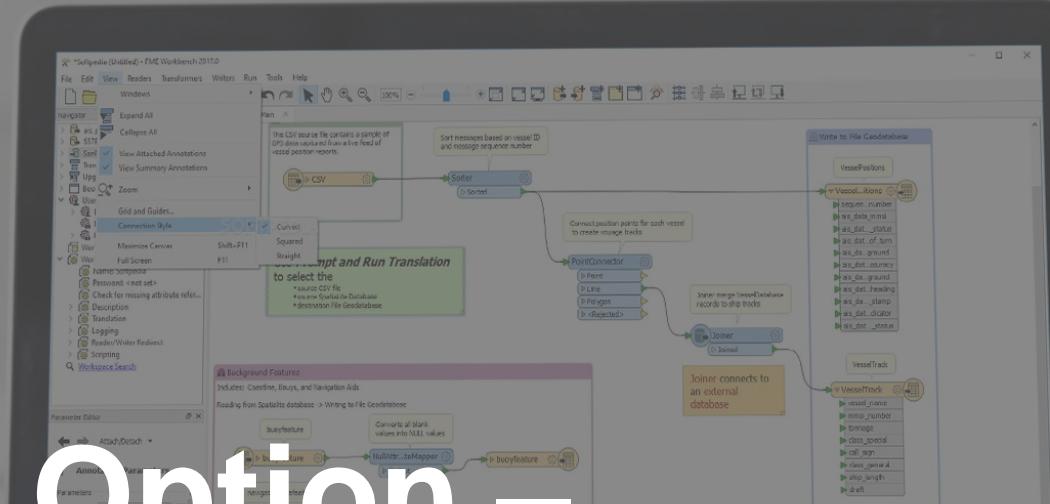
- This example will read in **all the files with the extension .shp** within the *Contours* folder:



- This example will read in **all the files with the extension .shp** within **all the subfolders** of the *ElevationModel* folder



# Workflow Option Single Merged Feature Type



Connect. Transform. **Automate**

# Single Merged Feature Type



- This is a Workflow option that is available on Readers
- Merges all the Reader's feature types into a single 'input' for entering the workflow

Add Reader

## Standard Individual Feature Types

Workflow Options

Individual Feature Types  Single Merged Feature Type

Rail 4

Rivers 117

Roads 2,186

Add Reader

## Single Merged Feature Type

Workflow Options

Individual Feature Types  Single Merged Feature Type

<All> 2,307

# Single Merged Feature Type

---



## Advantages

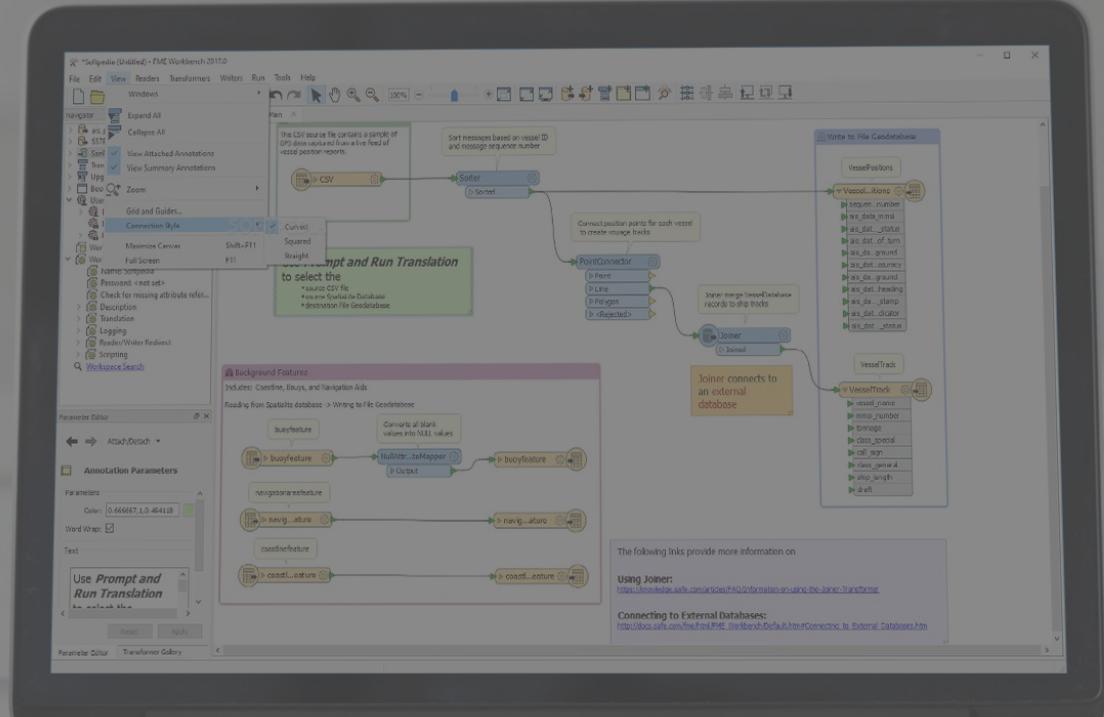
- Makes it easier to manage & connect a **large number of incoming feature types**
- Accepts incoming feature types of **any name** – great if the feature type names are not yet known
- Makes workspace more **reusable, flexible** and better suited for production environments

## Limitations

- **Schema/Attributes** – be mindful of data schema. Your incoming features must still be of the expected schema; attributes and geometry types.

For a truly dynamic workflow that is ‘schema-less’, the Dynamic Translation approach is required  
– we cover *Dynamic Translations* on our *Advanced Workflow Design* training module.

# Fanout



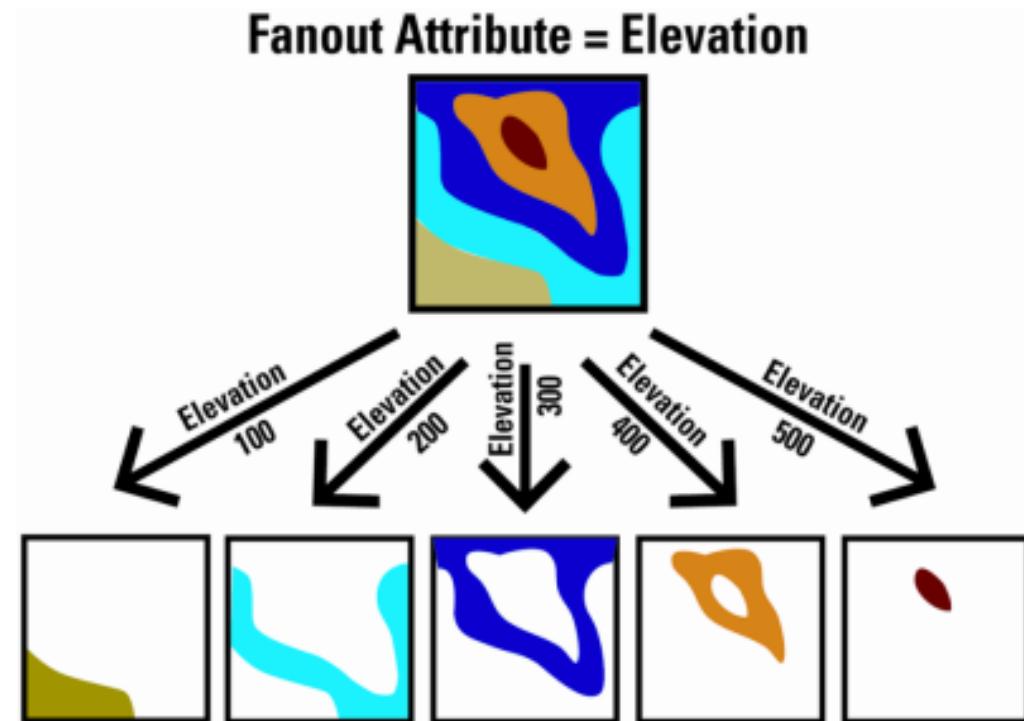
Connect. Transform. **Automate**



# Writing Multiple Files - Fanout

A **fanout** is a tool applied to a writer or writer feature type in FME. They are a way to write data divided into groups of features.

- Feature Type Fanout
- Dataset Fanout





# Fanout on Feature Type

- A Feature Type Fanout **splits data to multiple feature types within a single dataset**
- This is set within the Writer Feature Type.

The screenshot shows the 'Feature Type' dialog in FME Workbench. In the 'General' tab, the 'Table Name' is set to 'Parks' and the 'Writer' is set to 'Training [MAPINFO]'. The 'Attribute Value' dropdown menu is open, showing various attributes: DogPark, fme\_feature\_type, NeighborhoodName, ParkId, ParkName, RefParkId, SpecialFeatures, TreeCount, VisitorCount, and Washrooms. The 'NeighborhoodName' option is selected. To the right, a preview window displays the resulting schema for the 'Parks' table:

^ <@Value(NeighborhoodName)>
ParkId
RefParkId
ParkName
NeighborhoodName
VisitorCount
TreeCount
DogPark
Washrooms
SpecialFeatures

- The effect on the output will depend on the File Format being used:
  - AutoCAD – creates separate Layers/ Levels
  - Excel – creates separate Sheets
  - MapInfo TAB – creates separate .tab files

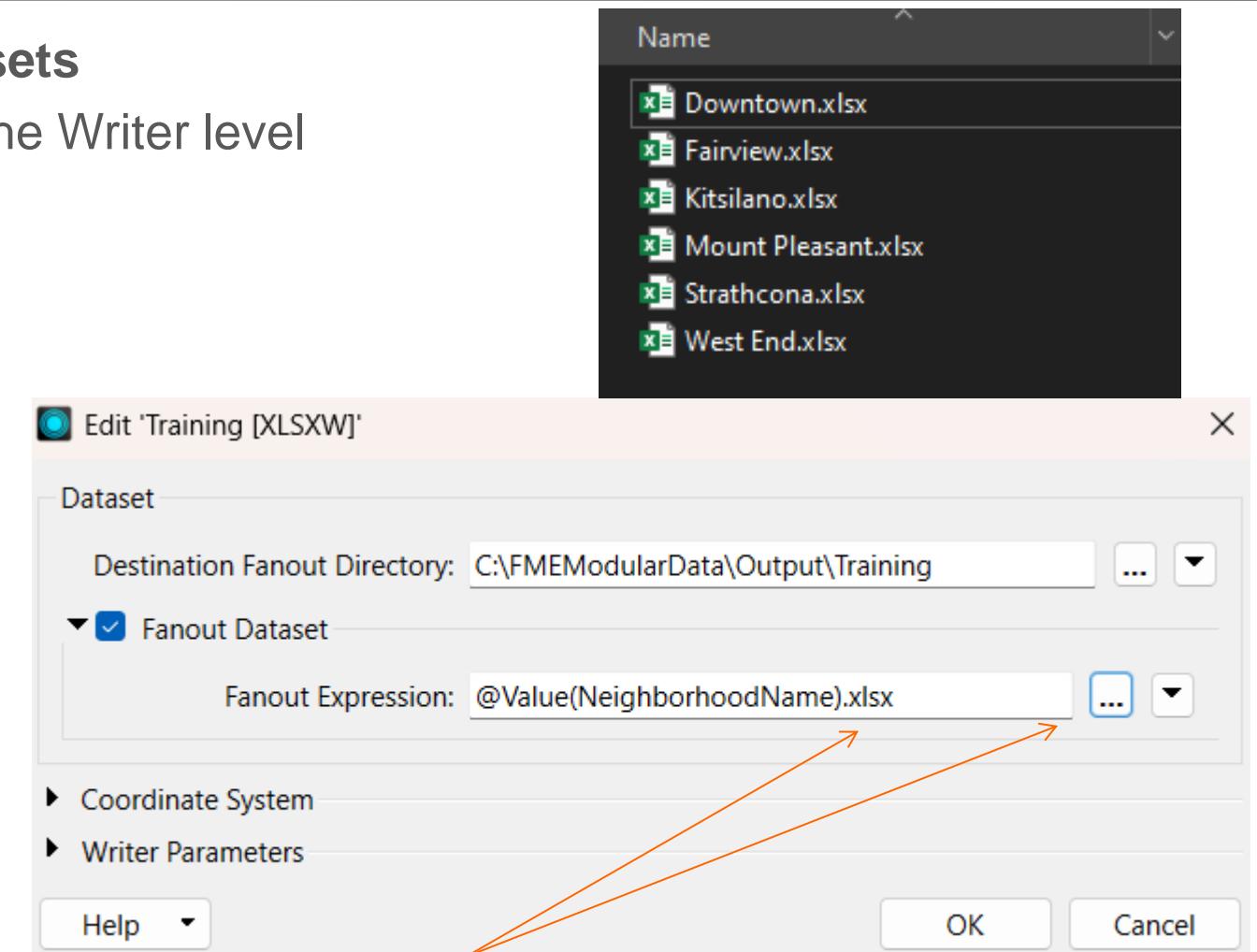


# Fanout on Dataset

- A Dataset Fanout delivers **multiple datasets**
- This is set within the Navigator Panel at the Writer level

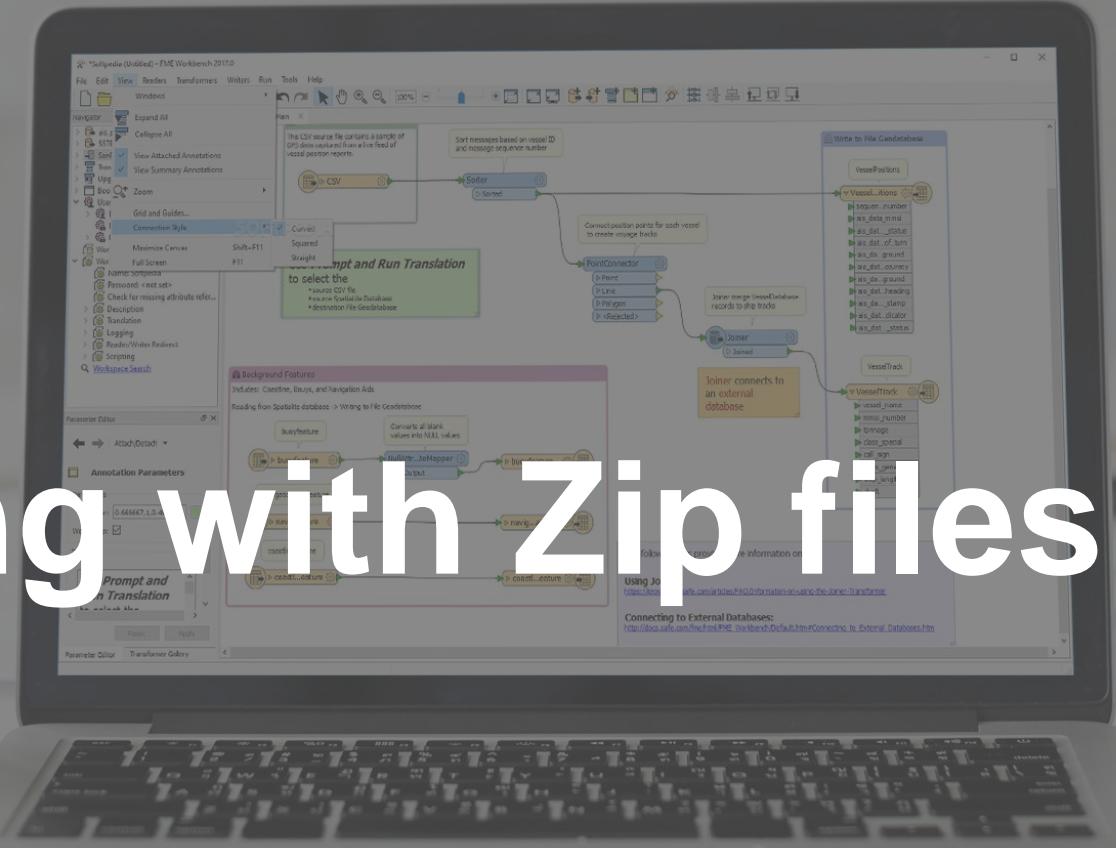
▼ Training [MAPINFO]

- ↳ Destination MapInfo Folder: C:\FMEModularData\Output\Trai...
- ↳ **Fanout Expression: @Value(NeighborhoodName)**
- ↳ Coordinate System: <not set>
- > Parameters



- For File-based formats you will also need to specify the file extension

# Working with Zip files



Connect. Transform. **Automate**

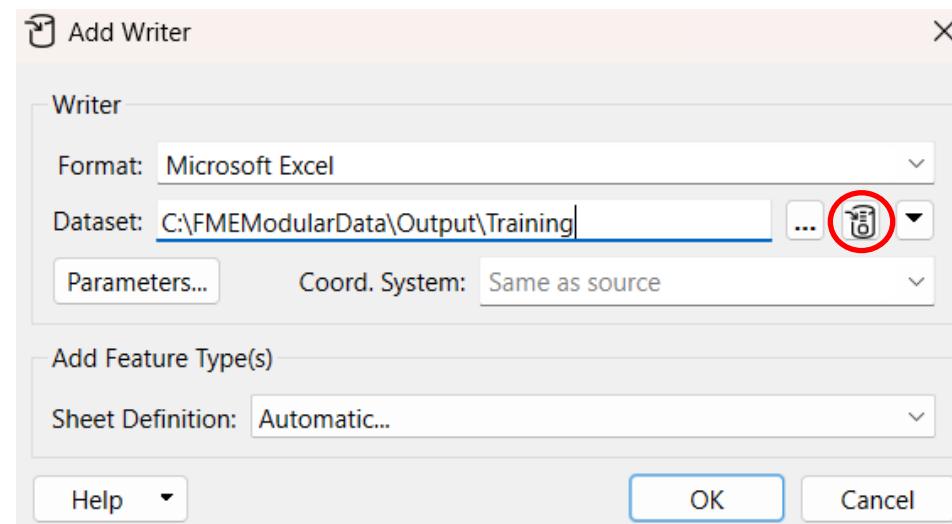


# Reading from and Writing to Archive Files

- Both FME readers and writers are capable of working with compressed (zip) files.
- Zip files are a convenient way to store datasets that need to be handled as a single unit; for example, a set of multiple files can be contained within a single zip file.
- The simplest way is to simply **change the file extension to .ZIP** or use the zip file icon on the Writer.

▼ **Training [XLSXW] - 2**

- ⚙ Destination Microsoft Excel File: C:\FMEModularData\Output\Training.zip
- ⚙ Fanout Expression: @Value(NeighborhoodName)
- ↗ Coordinate System: <not set>
- > ⚙ Parameters
- > ⚙ Feature Types (1)



# Exercise 4.1



## Reading Multiple Input Files and splitting output into Multiple Datasets

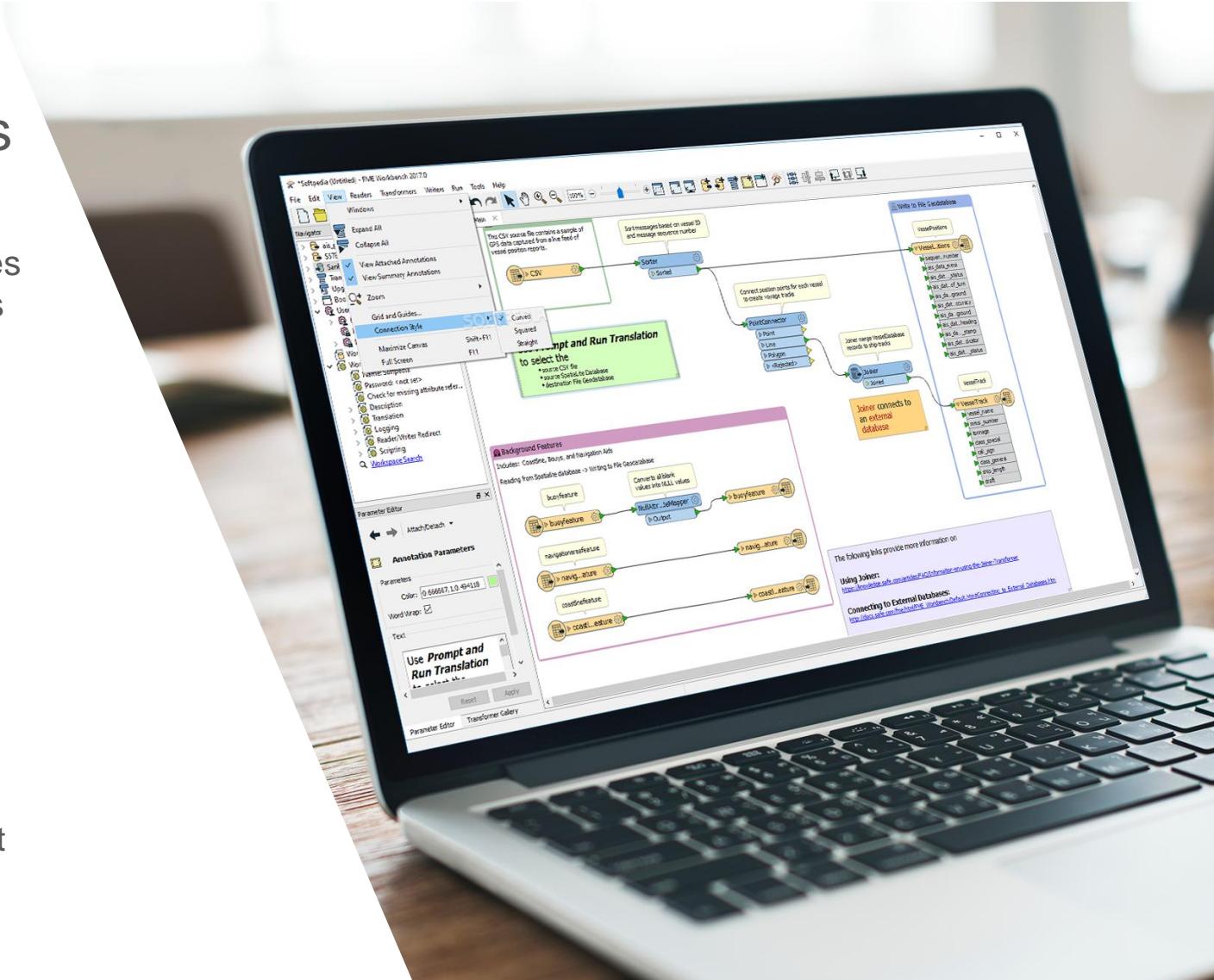
- You have been given the task of combining multiple tiles of contour data into a single coverage. This then needs to be divided into subsets for each elevation ‘band’
- The multiple output files need to be Zipped.

**Data – Contours (ESRI SHP)**

**Starter workspace –**

C:\FMEModularData\Workspaces\4.01-Workflow-MultipleFiles-Begin.fmw

**Goal -** Read in multiple files using wildcard and single merged feature type, creating a single holding. Then output into subsets using fanout.



## Exercise 4.2



# Building Flexible Workflows

- We have an ongoing need to convert community plans from DGN format to ESRI Shapefiles every year. It would be good if we could handle updated schemas without having to rebuild our workspace

**Data** - CommunityPlan2019 (Bentley MicroStation Design V8)  
CommunityPlan2020 (Bentley MicroStation Design V8)

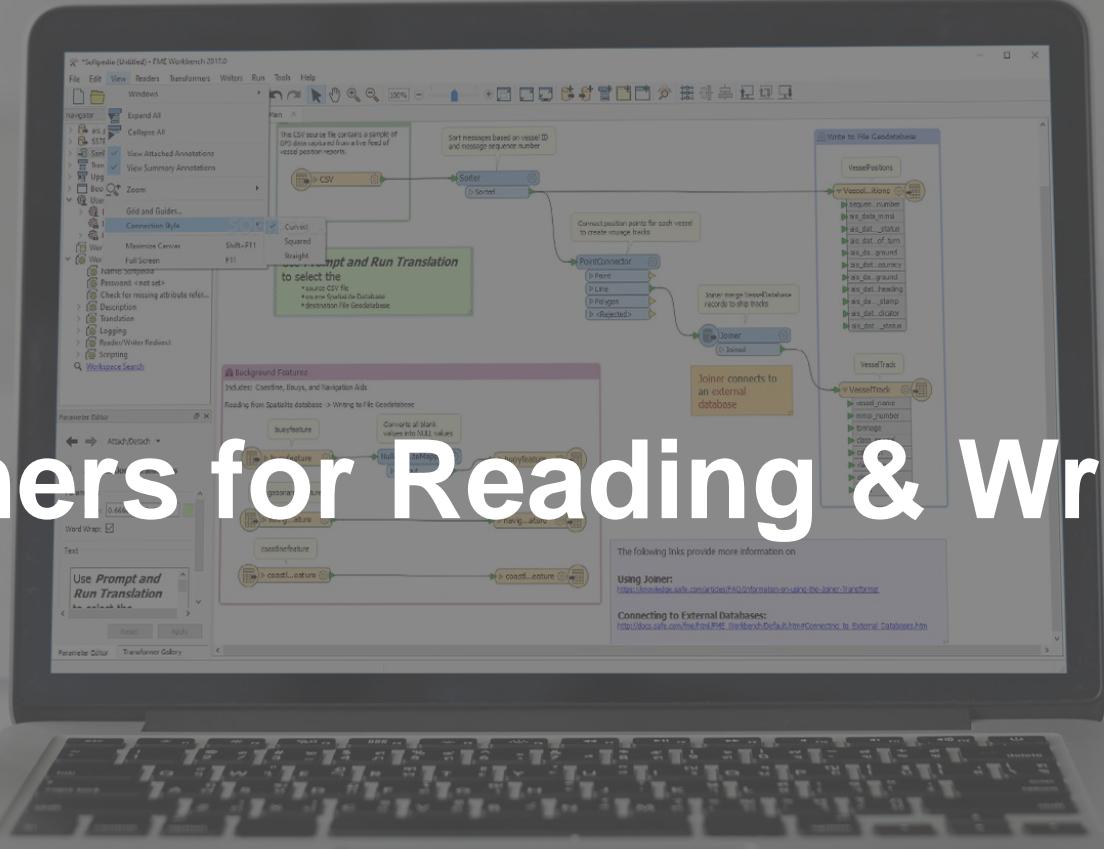
**Starter workspace** –

C:\FMEModularData\Workspaces\4.02-Workflow-Flexible-Begin.fmw

**Goal** – Create a flexible workspace to handle changing feature type names in source data

# Transformers for Reading & Writing

Connect. Transform. **Automate**

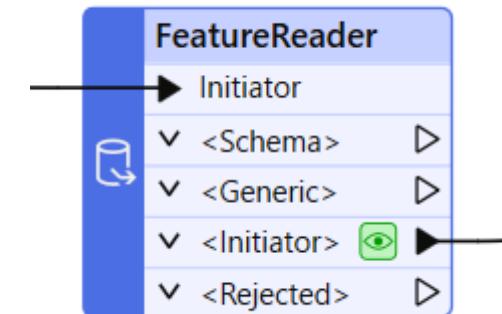


# FeatureReader and FeatureWriter transformers



- When designing some workflows the FeatureReader and FeatureWriter transformers can have advantages over standard Readers and Writers
- Amongst other things, they enable us to read and write data within the workflow, not just at the start and end of a workspace

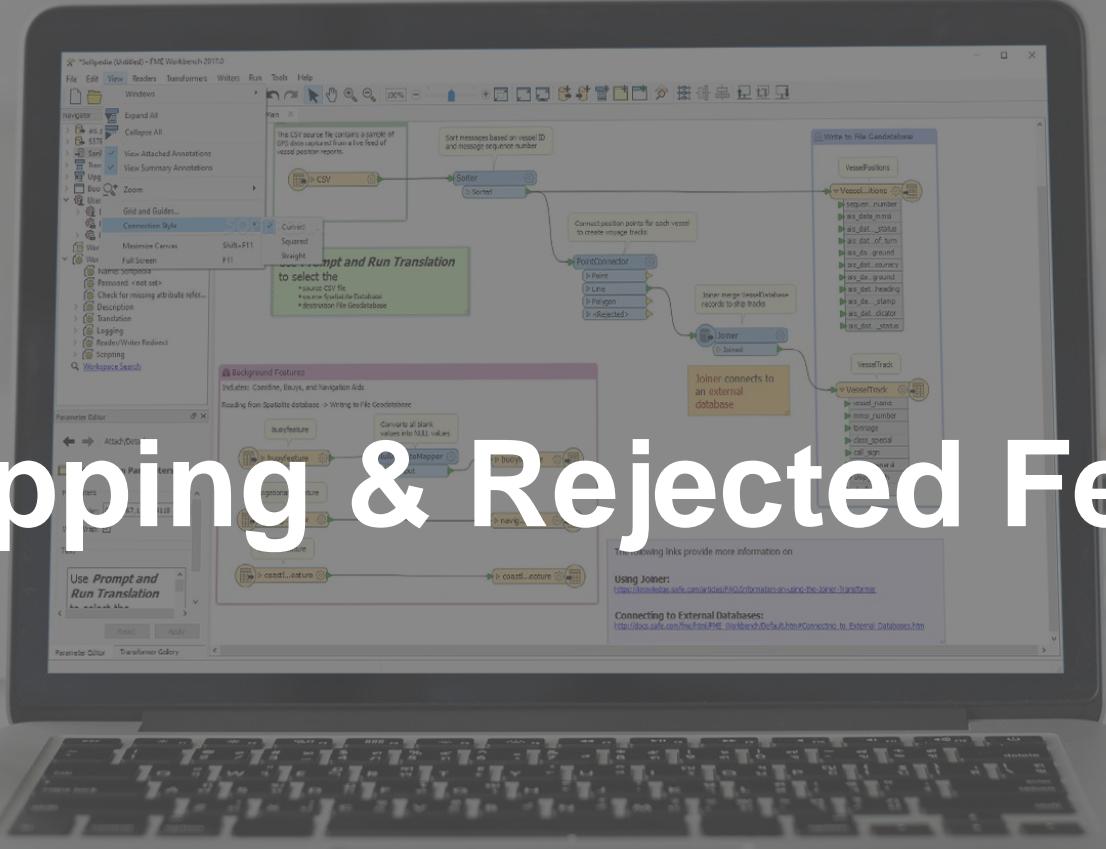
- the FeatureReader performs a complete read for each feature that enters the *Initiator* port.
- You can specify a WHERE clause or use a spatial filter
- the FeatureWriter allows post-processing of writer results
- After writing the output data it passes a single feature out of the Summary port



We'll see examples of these in action later.....

# Error Trapping & Rejected Features

Connect. Transform. **Automate**



# Error Trapping

---



- Error trapping is a way to design a workspace to trap unexpected data - so it does not have the chance to cause problems further downstream
- Error trapping can be as simple as adding a test or filter transformer to weed out bad or unwanted features, or it can be more complex.
- This is especially important in a production environment and where the workspace is reused repeatedly with resupplies of input data *where data changes may inadvertently be introduced.*

*As a workspace author you should attempt to foresee data problems that might arise, and build in methods to handle them.*

# Error Trapping - Logging

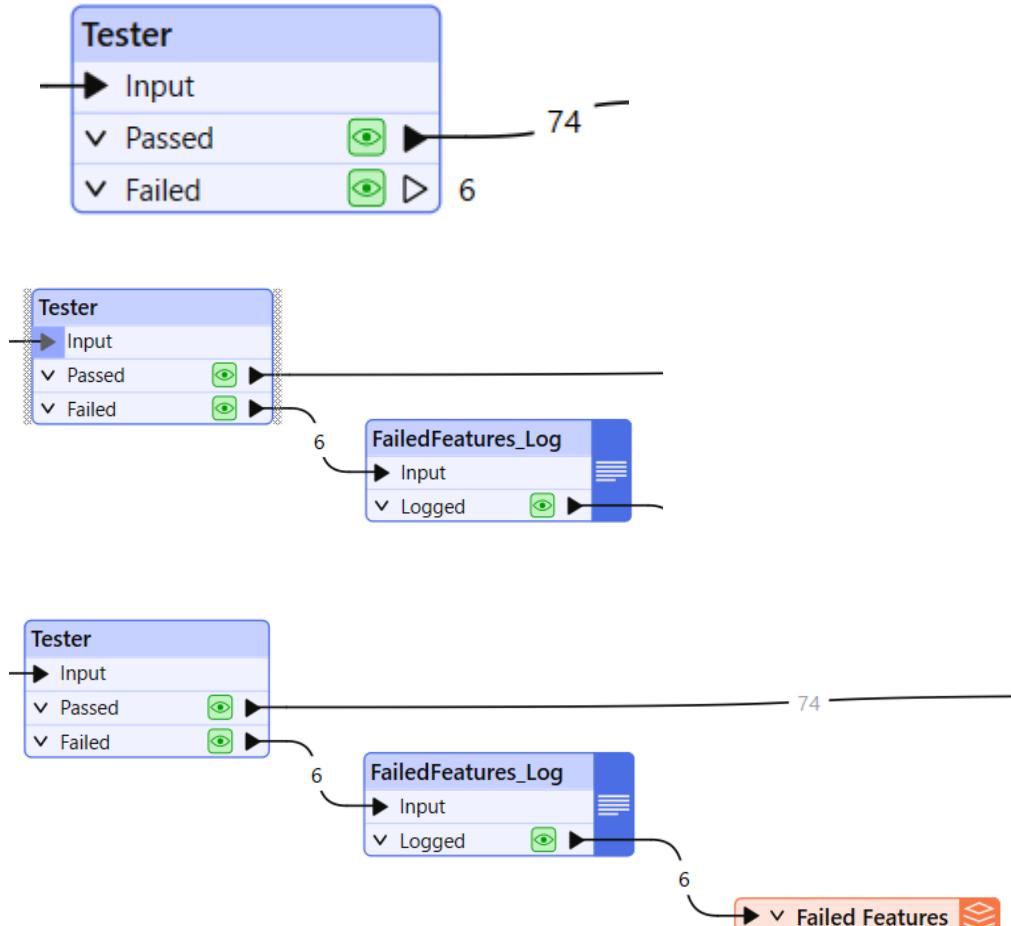


Here we are filtering our data, using a test on an attribute value to identify unexpected data.

The question is, what should we do with the features we have filtered out?

There are a number of things we could do:

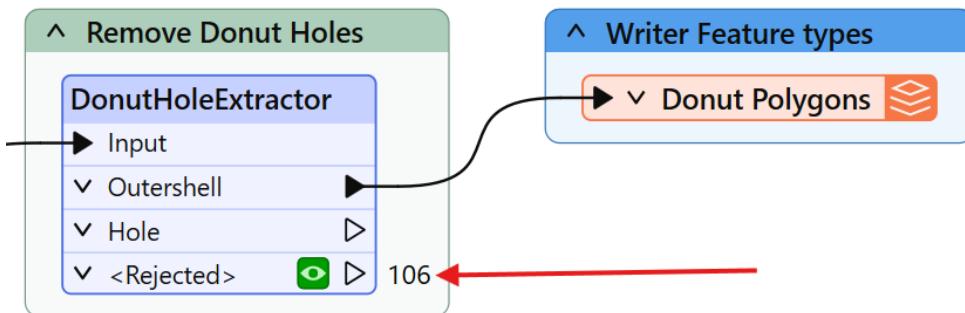
- Leave the features to drop out of the workflow at this point
- Record the data using a Logger transformer
  - this transformer will record all incoming features to the log
- Write the features out to a CSV file (or other format)



# Rejected Features



- Being able to identify when features are rejected (and excluding from the workflow) is important.
- Many transformers include a <Rejected> port to output these invalid features:



- features are automatically counted and stored on a <Rejected> port, even if feature caching is off.
- As an additional benefit, the rejected features will often include a rejection code attribute explaining the problem:

Table	
DonutHoleExtractor: <Rejected>	
1	ParkName
1	Tea Swamp Park
2	Mcbride Park
3	Creekside Park
4	Arbutus Greenway Park
5	<missing>
	fme_rejection_code
	INVALID_GEOMETRY_TYPE

# Rejected Features – Workspace Parameter

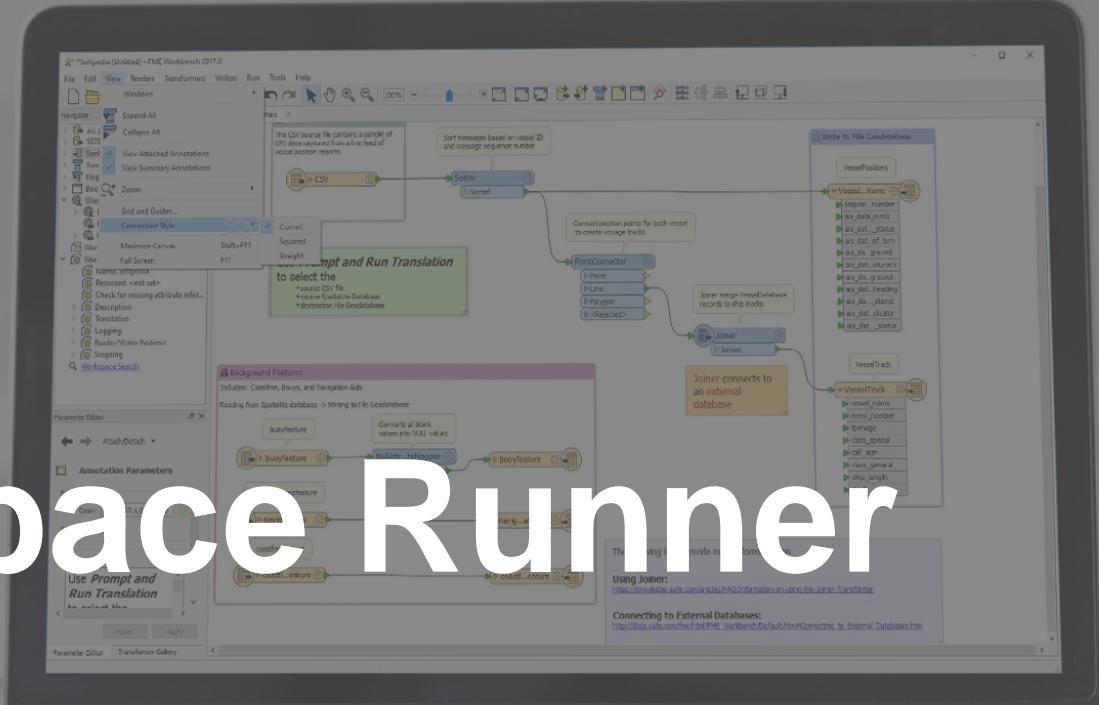


- Note: There is a Workspace Parameter which by default is set to **terminate translation** if features are rejected
- This can catch-out users, as its hidden in the Navigator Panel!
- This parameter can be changed to *Continue Translation*
  - This is particularly useful if the workspace is running as a scheduled task or you need the workspace to run to completion even if some features are rejected. In these scenarios capture the details of any rejected features by using a Logger or by writing them out to a file (e.g. CSV)

The screenshot shows the Navigator panel in FME Workbench. The 'Workspace Parameters' section is expanded, revealing various configuration options. The 'Rejected Feature Handling' parameter is highlighted with a blue selection bar, indicating it is the focus of the discussion.

Parameter	Description
Password: <not set>	Not applicable for this feature.
Name: <not set>	Not applicable for this feature.
Description	Not applicable for this feature.
Logging	Not applicable for this feature.
Reader/Writer Redirect	Not applicable for this feature.
Scripting	Not applicable for this feature.
Translation	Contains the 'Rejected Feature Handling' parameter.
Terminator Redirect: No Redirect	Not applicable for this feature.
<b>Rejected Feature Handling: Terminate Translation</b>	This parameter is highlighted. It controls whether the workspace terminates translation if any features are rejected.
Stroking Tolerance: 0	Not applicable for this feature.
Ignore Failed Readers: No	Not applicable for this feature.
Prompt And Run - Save Parameter Values: Yes	Not applicable for this feature.
Order Writers By: Position in Workbench Navigator	Not applicable for this feature.
Reprojection Engine: FME	Not applicable for this feature.
Esri ArcGIS Compatibility: Auto	Not applicable for this feature.

# Workspace Runner

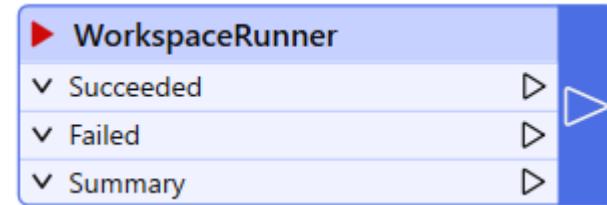


Connect. Transform. **Automate**

# Introduction to the WorkspaceRunner



- The WorkspaceRunner transformer runs an additional FME Workbench workspace on the local computer by spawning a new FME process

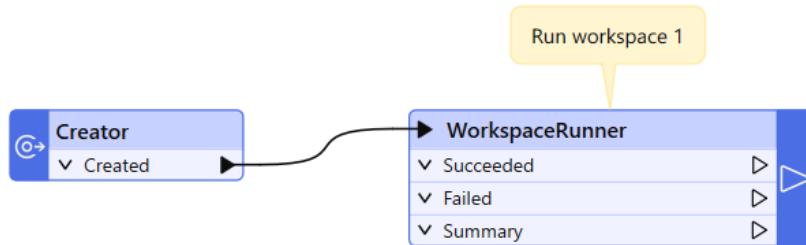


- This transformer can be useful in a number of scenarios:
  - chaining** multiple workspaces to run in series
  - workflows where different workspaces need to be triggered to run based on criteria (useful for **self-serve or production environments**)
  - in conjunction with Emailers for translation **status notifications**
  - Batch processing bulk data** – for performance
- Many of the above workflow design techniques require intermediate-advanced FME skills to deploy - *and some are covered in our 'Advanced Workflow Design' training module*  
However we will now take a look at the fundamentals of using this transformer

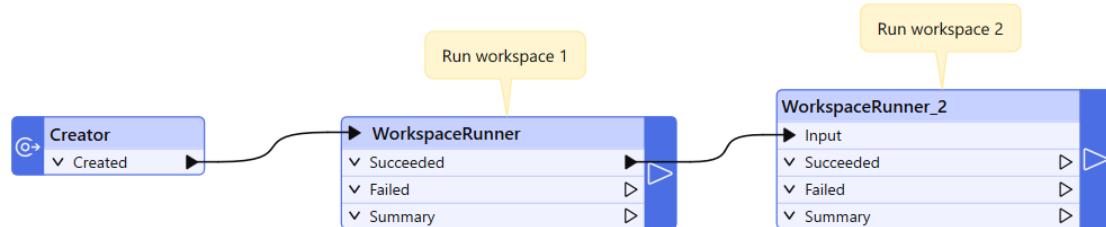
# Introduction to the WorkspaceRunner



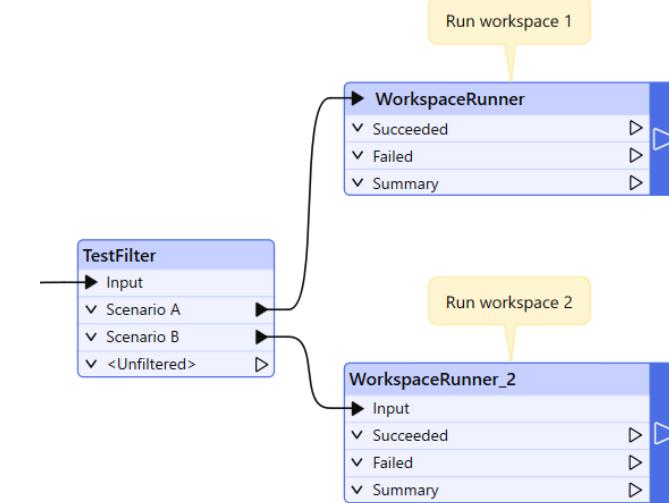
Single workspace triggered



Two workspaces chained to run in series



Criteria determining which workspace is required

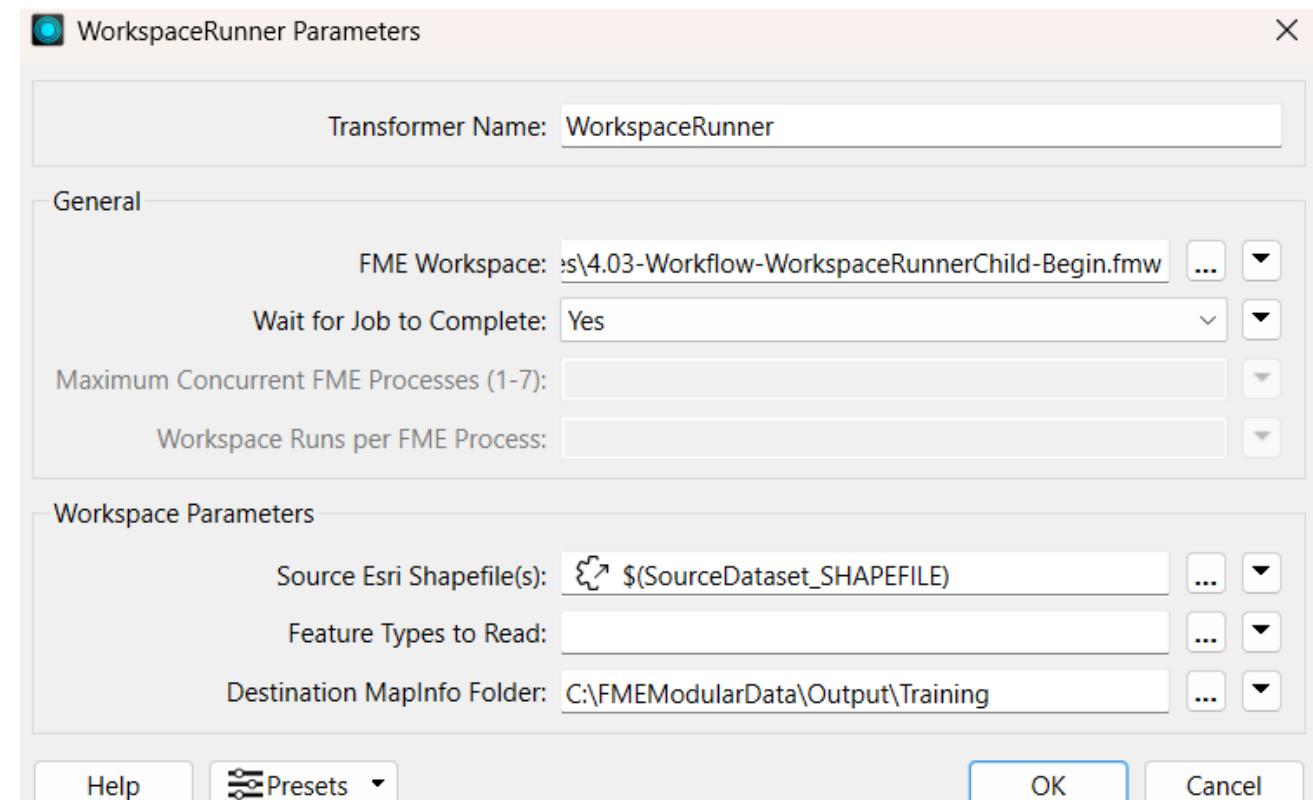


# Using the WorkspaceRunner

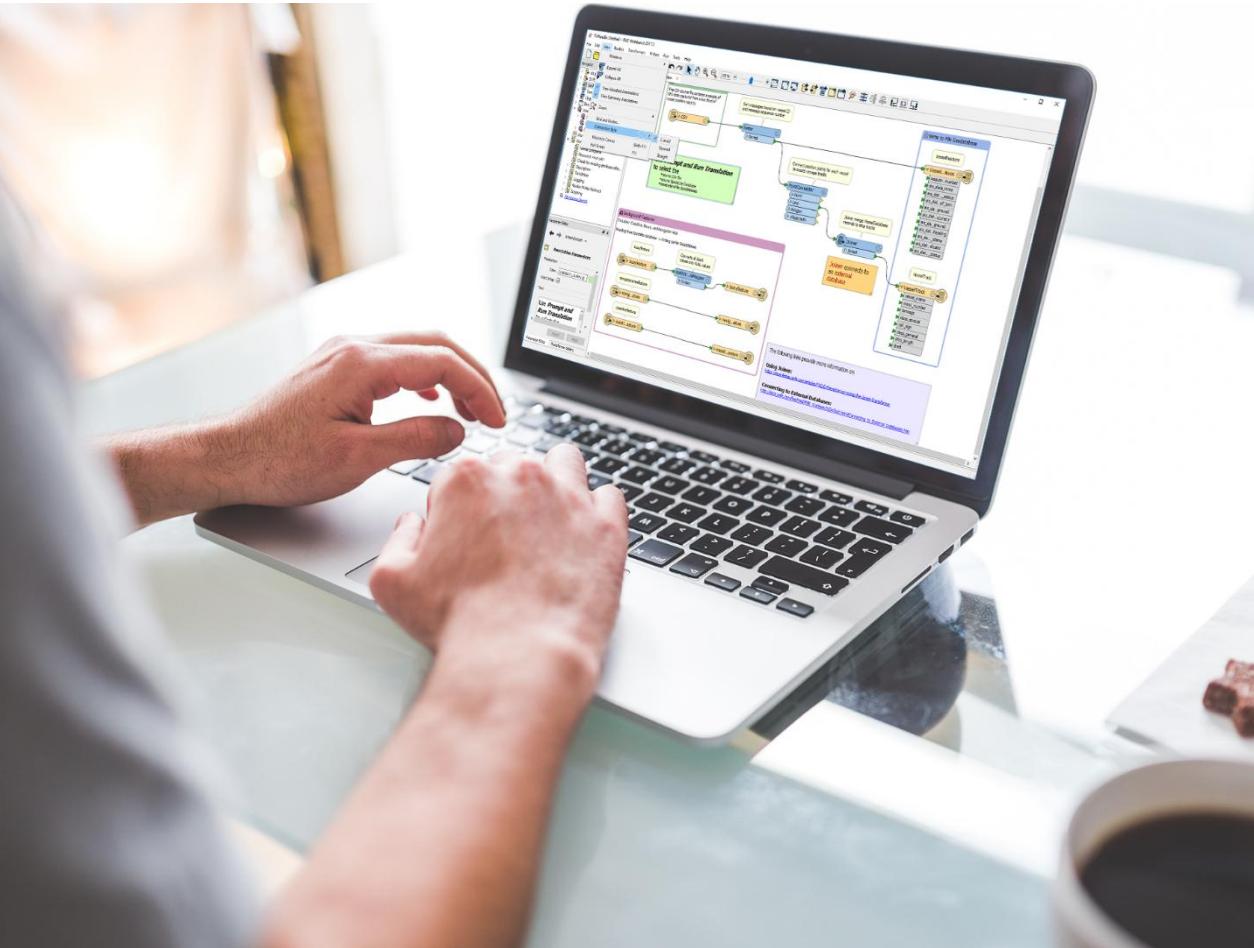


- Setup the '**child**' workspace first – as this will determine what parameters need to be passed to it by the WorkspaceRunner from the '**parent**' workspace e.g. *source data and output destination*
- Within the **parent** workspace add the WorkspaceRunner

- Within the WorkspaceRunner transformer specify:
  - The child workspace that it needs to run
  - Wait for Job to Complete: Yes/No
  - Parameters that are required by child workspace.



# Exercise 4.3



## Using the WorkspaceRunner

- We have a future need to enhance our production environment deployment of FME Form; batch processing of bulk data, generating email success/failure notifications for scheduled workspaces, and more.
- But before we can employ these techniques, we first need to understand the fundamentals of using the WorkspaceRunner transformer.

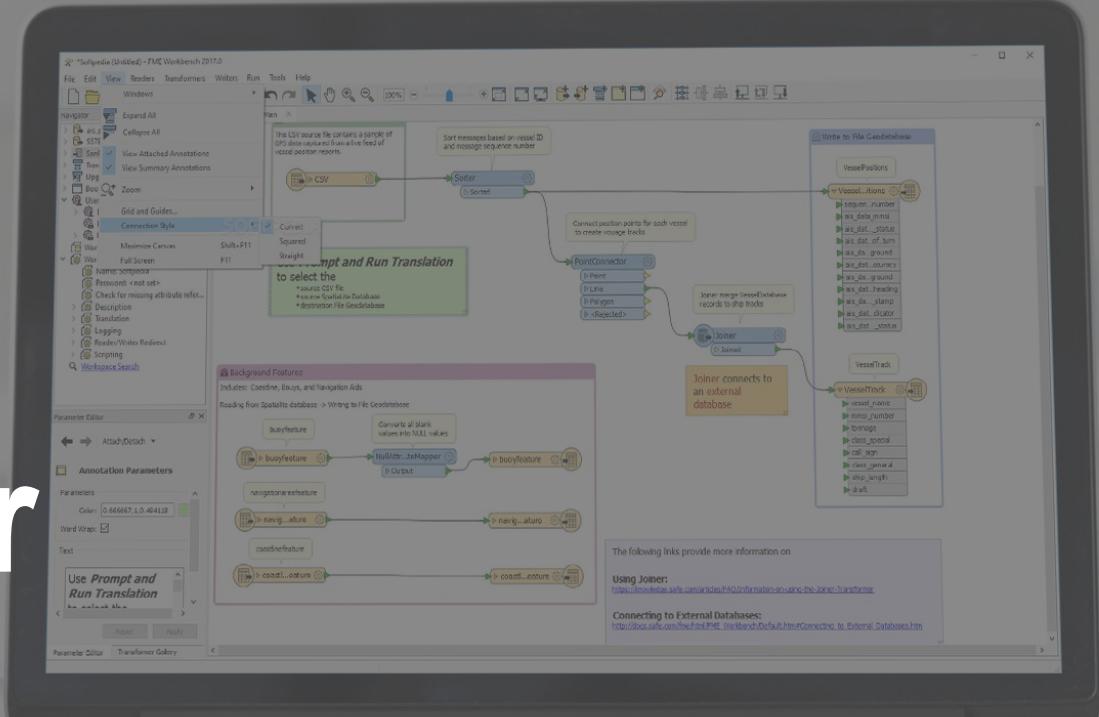
**Data - Contours (ESRI SHP)**

**Starter Workspace –**

C:\FMEModularData\Workspaces\4.03-Workflow-  
WorkspaceRunnerChild-Begin.fmw

**Goal –** Chain two workspaces using the WorkspaceRunner

# Emailer



Connect. Transform. **Automate**

# Emailer transformer

---

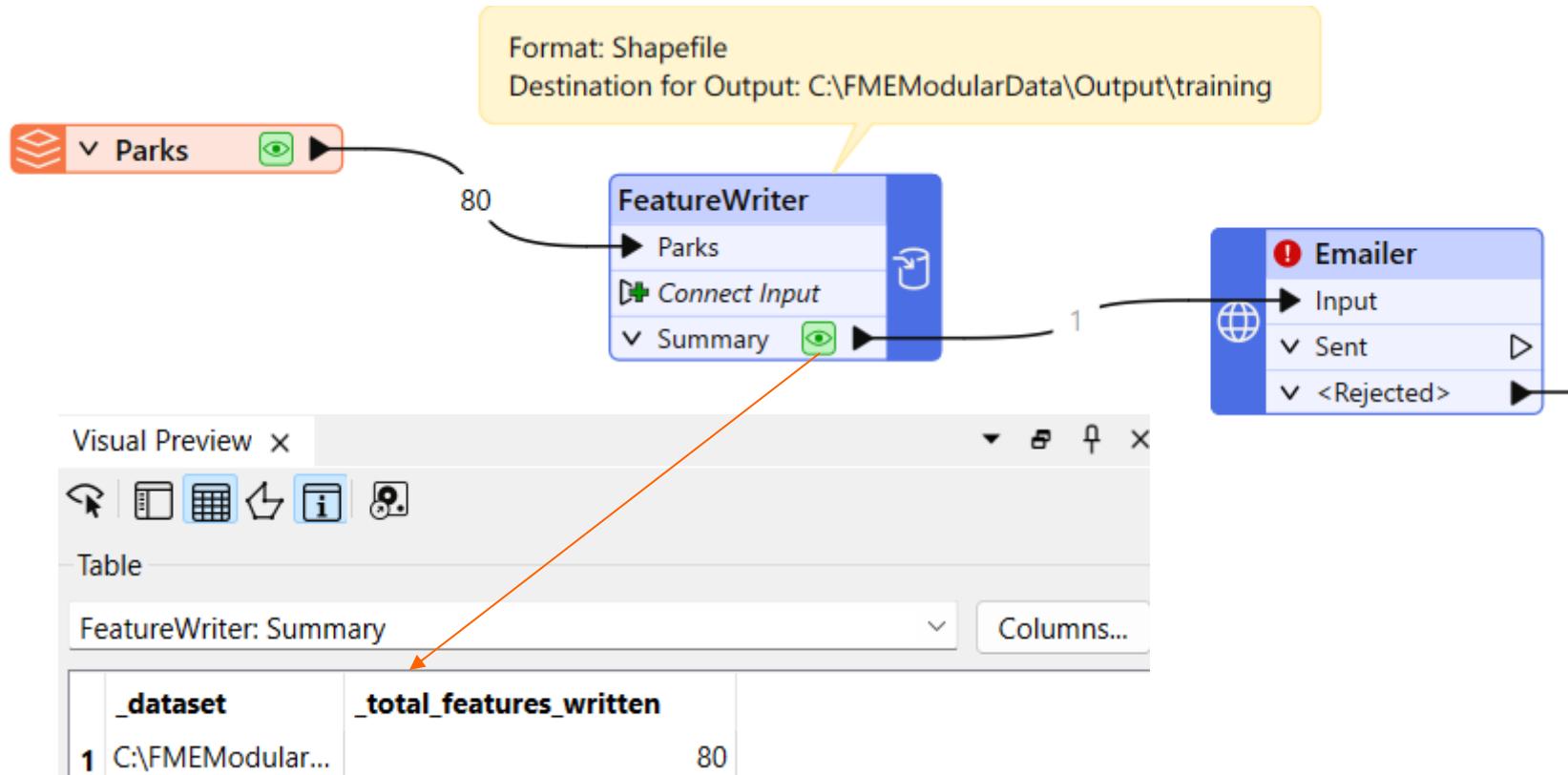


- The Emailer transformer uses Simple Mail Transfer Protocol (SMTP) to send email notifications as part of a workspace
- Both HTML and Plain Text emails may be sent
- Considerations:
  - The Emailer transformer will send one email per input feature – *so if multiple features are received, multiple emails will be sent*
  - If a workspace run **fails** and no features are passed to the Emailer, it will not be triggered - *so no email*

# Use Emailer to Notify Data Written



- Instead of a standard Writer, use a **FeatureWriter** transformer in conjunction with the Emailer
- the FeatureWriter allows post-processing of writer results
- After writing the output data it passes a single feature out of the Summary port

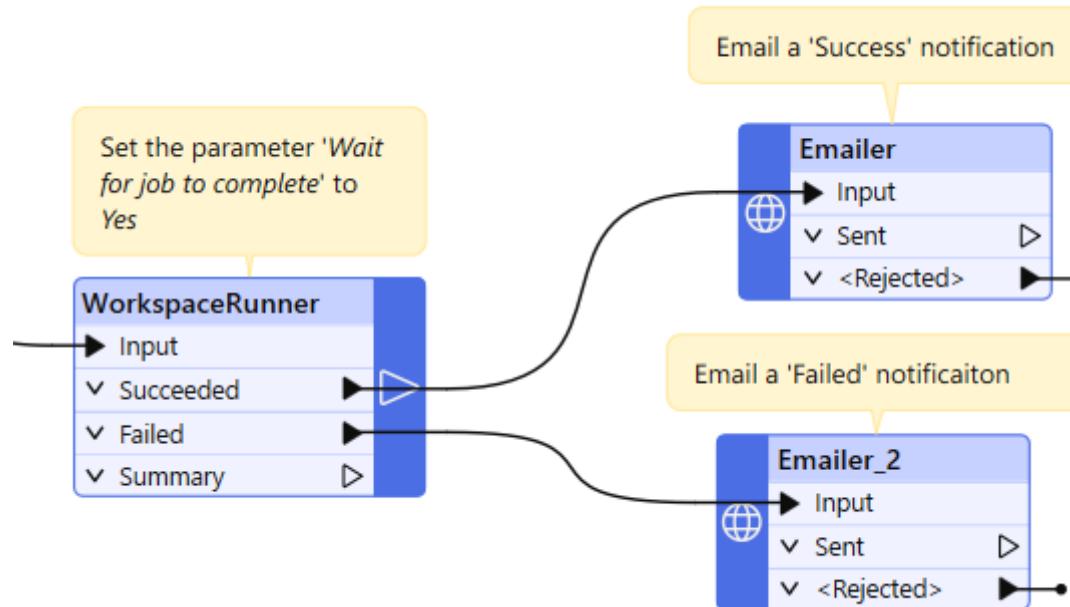


# Use Emailer to Notify Success or Failure of Workspace



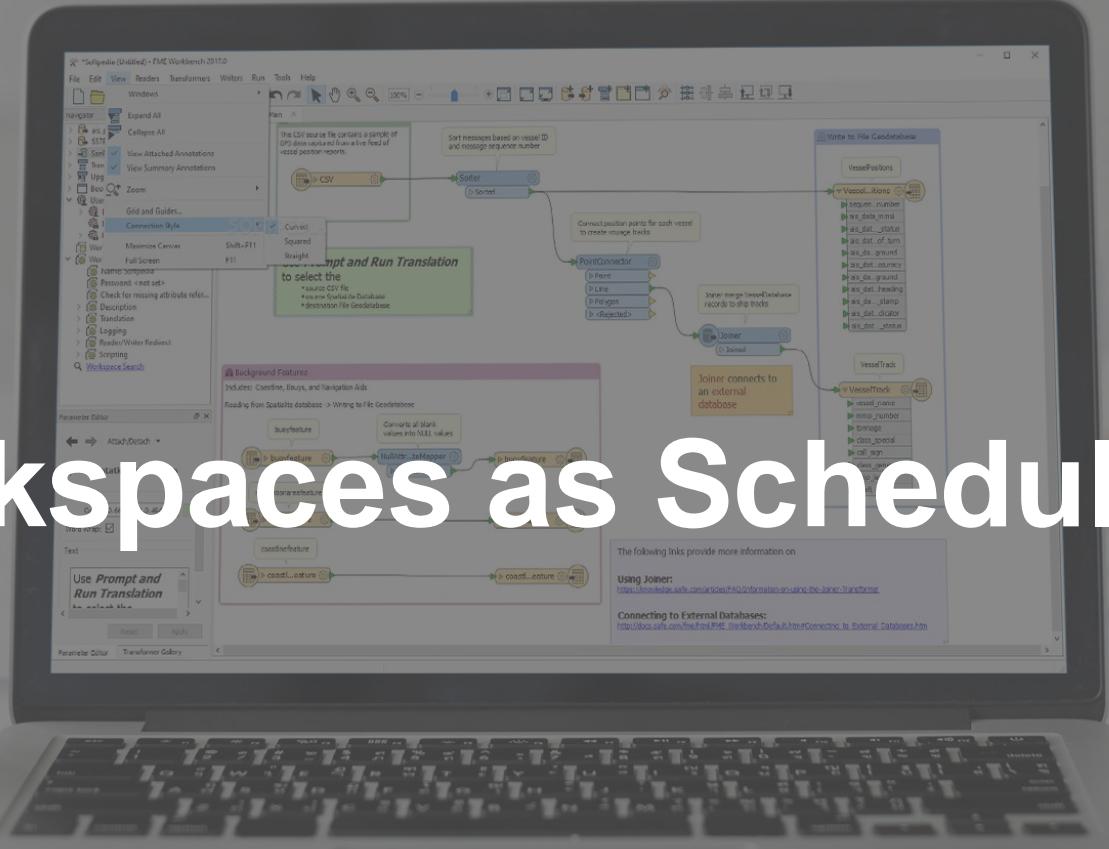
Triggering the Emailer when data is written is easy. But if a workspace run fails and no features are passed to the Emailer, how do you trigger a failure notification email?

- Use the Emailer transformer in conjunction with a **WorkspaceRunner** to notify of translation status (success/failure)



# Run Workspaces as Scheduled Tasks

Connect. Transform. **Automate**



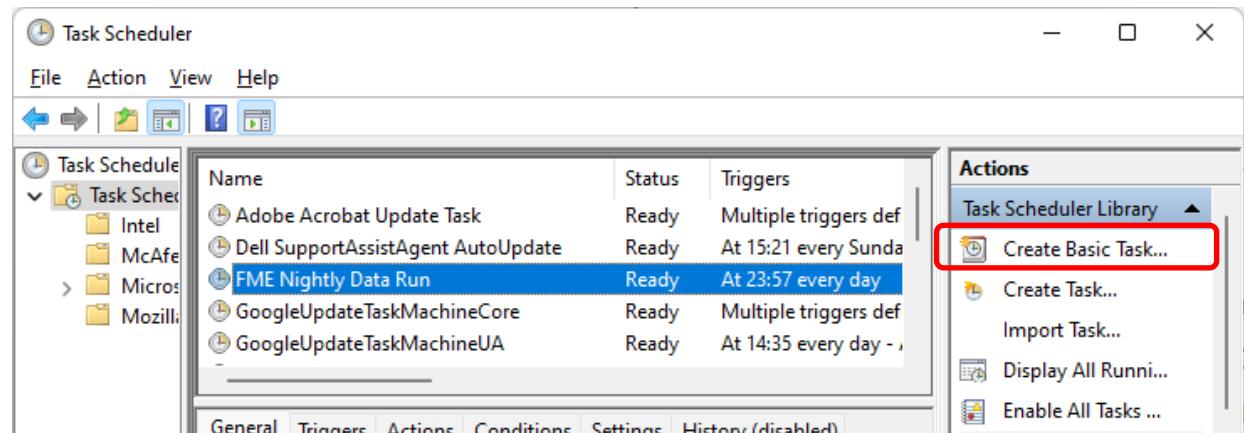
# Scheduling Workspaces to Run



- **FME Flow** is the ultimate method to schedule workspaces to automatically run on a regular basis
- But if you only have **FME Form**, then you can use **Task Scheduler** – a tool included with Microsoft Windows:
  - create a .txt file detailing the location of your fme.exe and the workspace that needs to be run. Then change the file extension to **.bat**

```
"C:\Program Files\FME\fme.exe" C:\FMEModularData\Output\Workspaces\data_transfer_between_systems.fmw
```

- Within the Task Schedule Windows tool:
  - Create a Basic Task...
  - Give the task a Name and Description
  - Choose when and how often it needs to run
  - Action is to *Start a program*
  - Browse and select the .bat file



Thanks for joining!

Any questions?



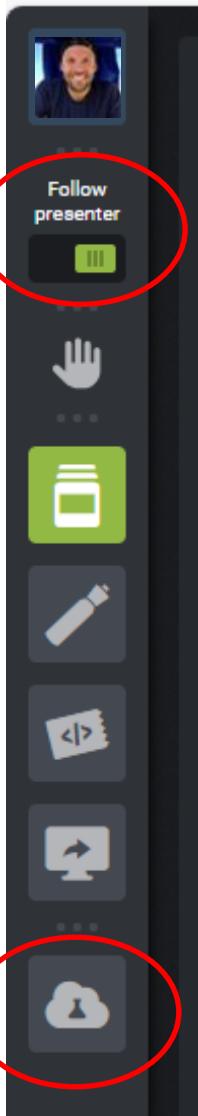
# FME Form Training

## - Module 5

Best Practice and Performance



# Training Environment



< keep 'Follow presenter' on

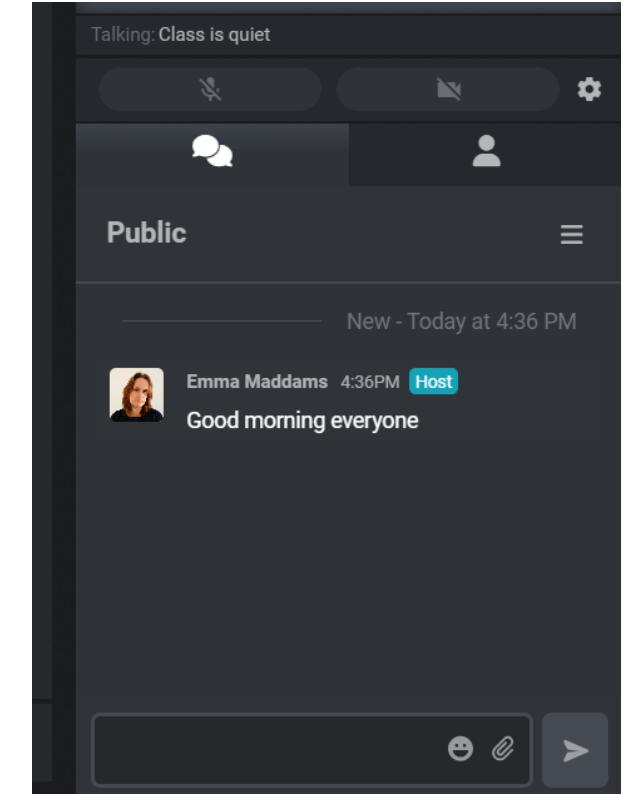
< Lab

need help when using your Lab  
Use either:

- o 'Raise hand' or
- o 'Lab Assistance'

## Chat

- o to everyone
- o to trainer



# Training Environment

---



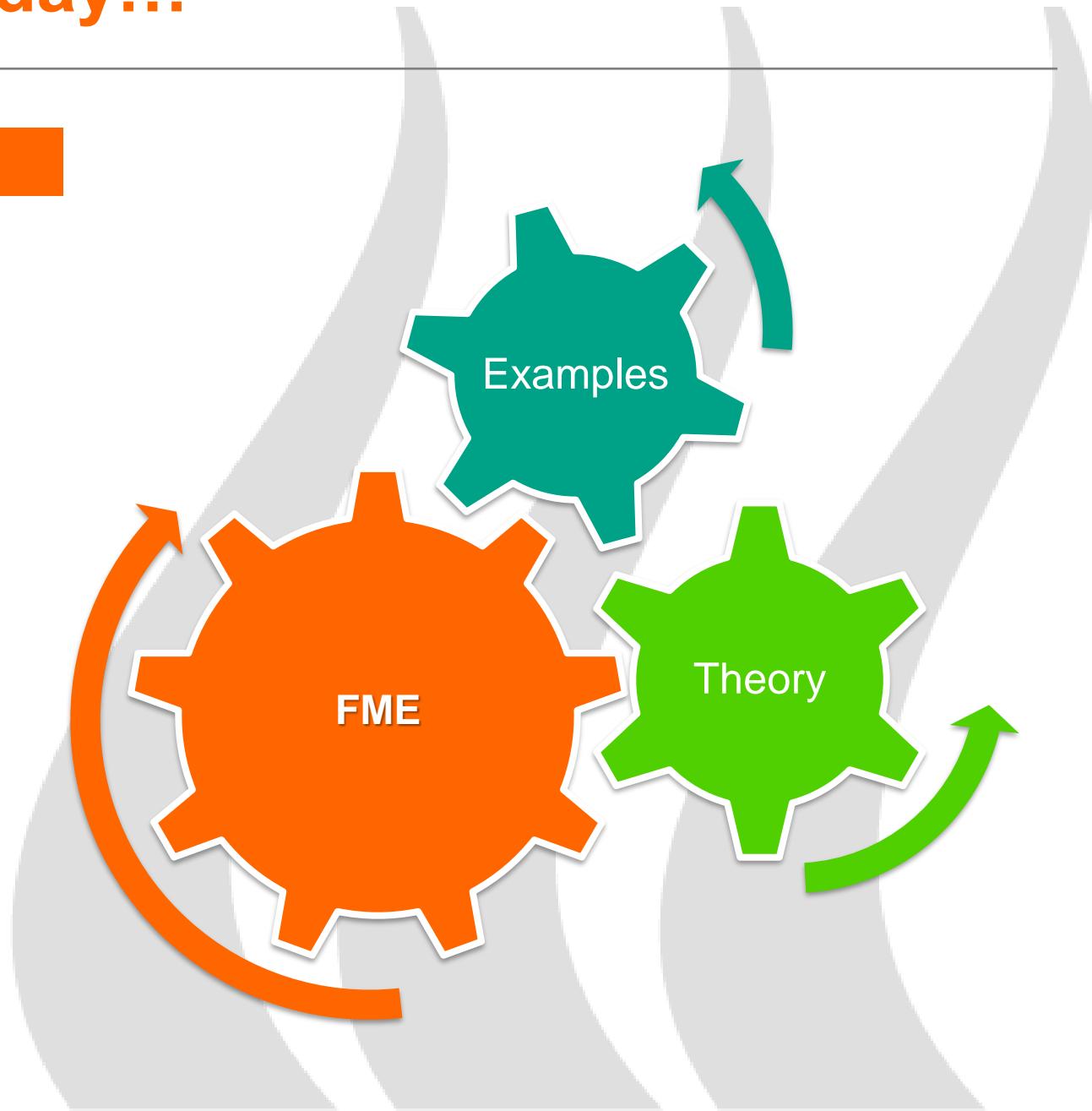
- Training Data folders **C:\FMEModularData**
  - Data
  - Output
  - Resources
  - Workspaces
- Slides
- Workbook – you need to download using link sent by the trainer

# What we'll be covering today...

---

## Agenda

- Best Practice
- Maintenance/Upscaling
- Error Trapping
- Debugging Workspaces
- Performance
- Documenting & Styling Workspaces



# Best Practice



Connect. Transform. **Automate**

# Best Practice

---



How to carry out a task in the most effective and efficient manner in FME?

Just remember some Best Practice rules!

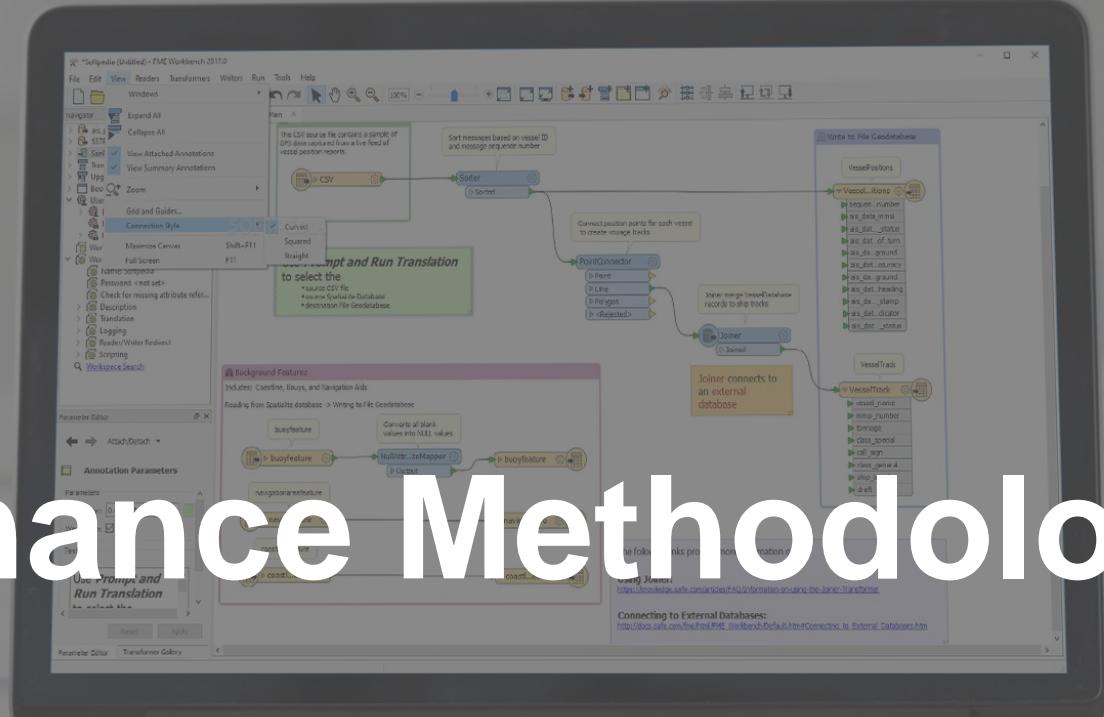
Best Practice can help us to...

- Use FME in a project-based environment
- Create workspaces that use the correct functionality in the correct place
- Create workspaces that can be easily maintained and are scalable
- Create high-performance workspaces
- Create well-styled workspaces that are self-documenting
- Debug a workspace when it doesn't work the way intended
- Use FME Workbench in the most efficient way



# Maintenance Methodology

Connect. Transform. **Automate**

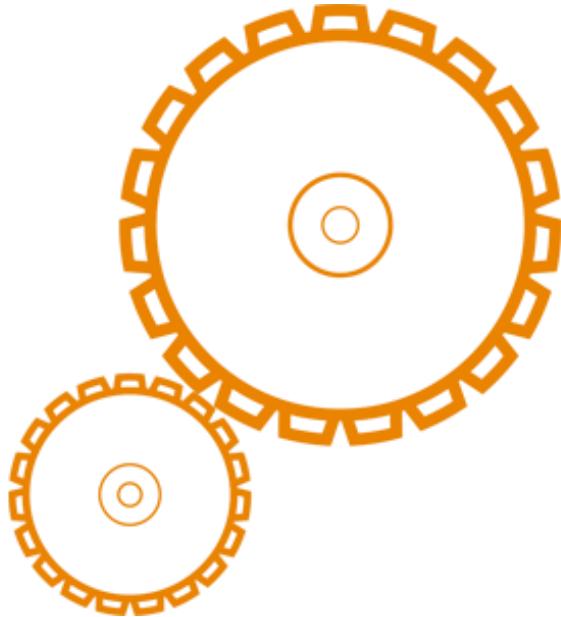




# Maintenance Methodology

---

The ability to update the workspace, and scale it up to a larger solution.



**Number of Transformers  
Complexity  
Connections**

*How easily will you and other users be able to maintain and expand your workspace?*

# Maintenance Methodology

---



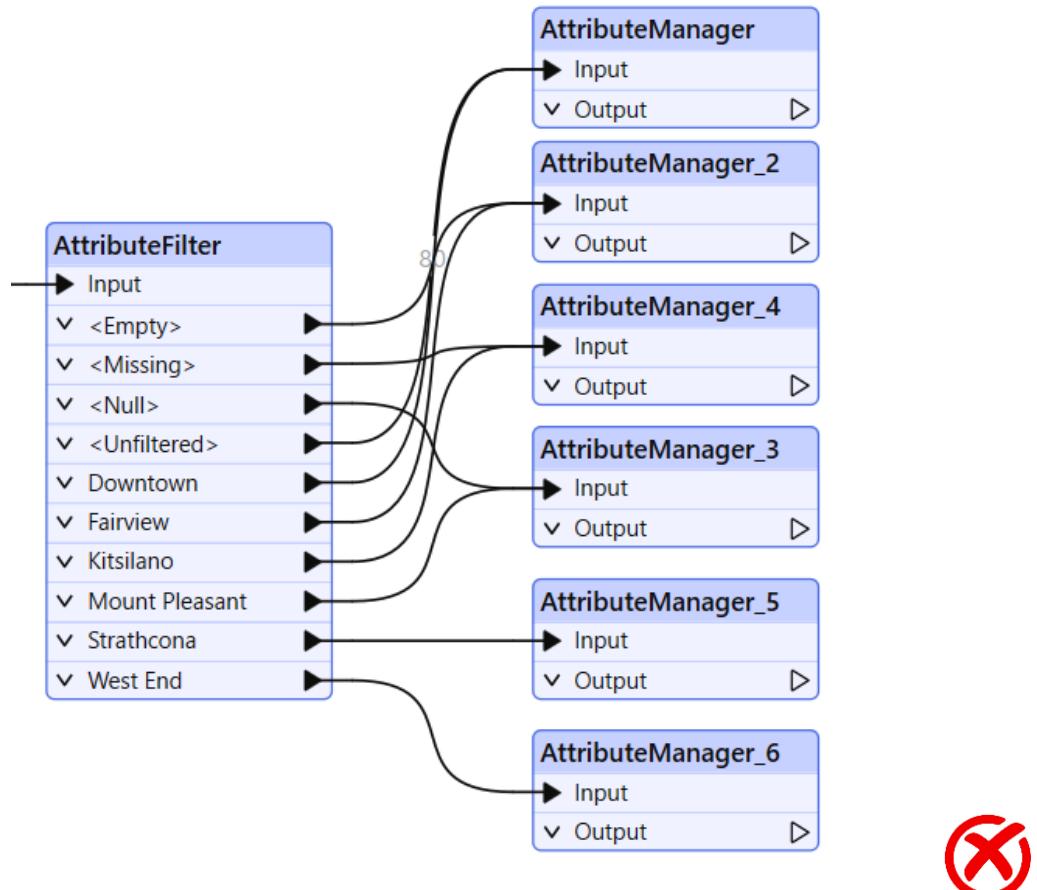
A weak workspace will be hard to update and expand. Things to watch out for:

- **Duplicated Transformers** - these can be cumbersome to edit.
- **Complexity**
  - **Excess Scripting** – Why put in Python Scripting when a Transformer will do?
  - **Under the hood functions** – using hidden/advanced functions, when a simple transformer does the same job. (e.g. use an ExpressionEvaluator transformer instead of Arithmetic Editor within an AttributeManager)
  - **Multiple Workspaces** – chaining workspaces unnecessarily
  - **Busy Workspace Style and Layout** – convoluted styling is hard to understand. Also, keep an eye on your connector lines; crisscrossing or poorly annotated tunnels

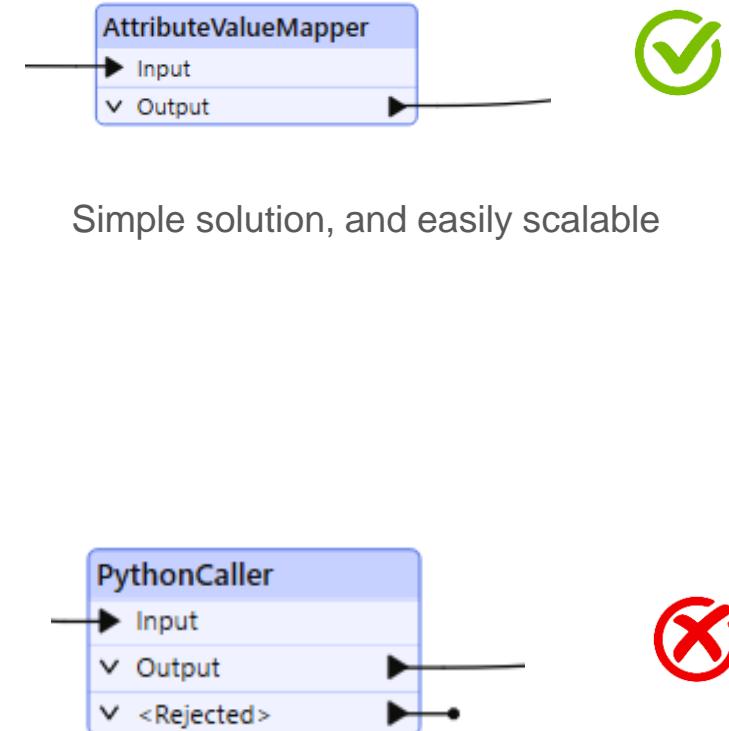
# Maintenance Methodology - example



All three methods produce the same result... but .....



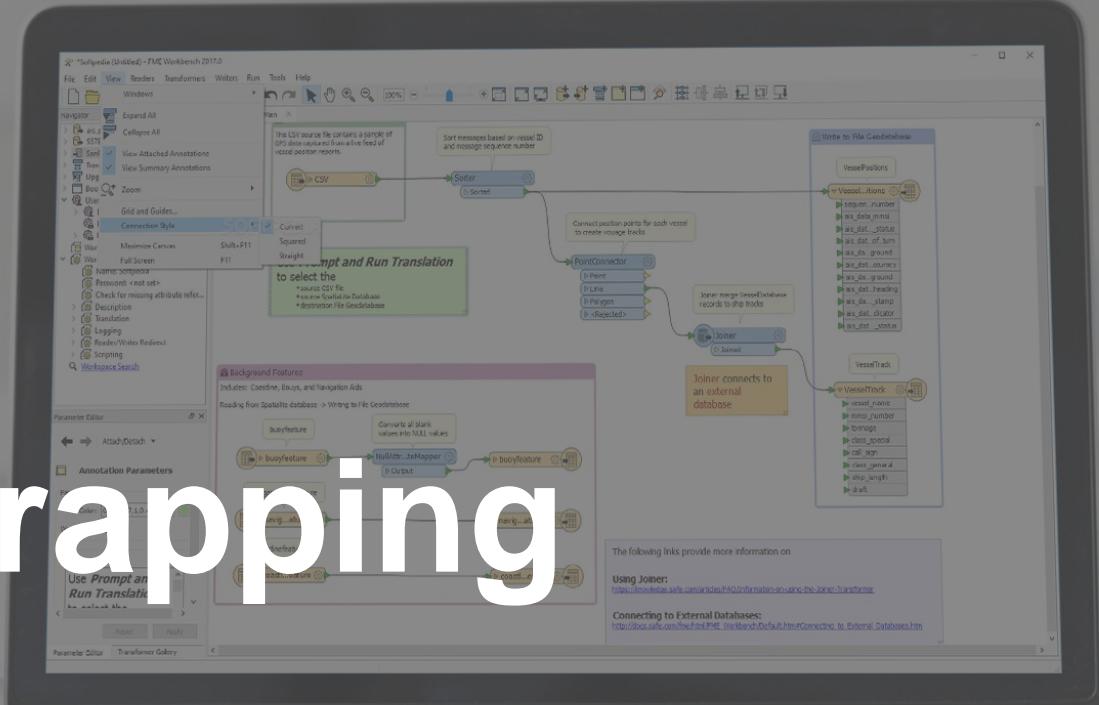
This method uses duplicate transformers and would be time consuming to scale-up in the event of extra years being added



Simple solution, and easily scalable

Over Complex - Python script is not easy to maintain

# Error Trapping



Connect. Transform. **Automate**

# Error Trapping & Rejected Features



Another workspace best practice technique is Error Trapping:

- Error trapping is a way to design a workspace to trap unexpected data
  - Sometimes scaling up means using more datasets of varying types and quality. If data quality is not considered, future performance can be compromised.
  - Use error trapping and logging to prevent unexpected data to cause workspace failure.
  - Error trapping can be as simple as adding a test or filter transformer to weed out bad features. This could include:
    - Attribute validation or filtering
    - Geometry validation or filtering
  - Error trapping and Rejected Feature attributes can be used to **debug a workspace**

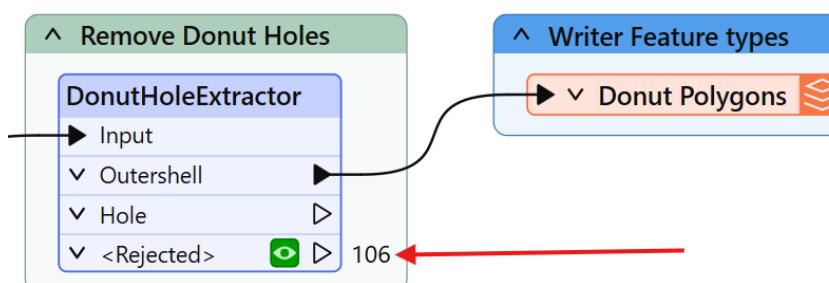
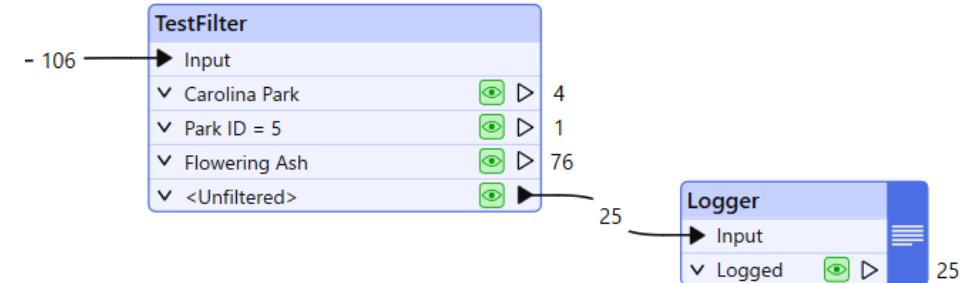
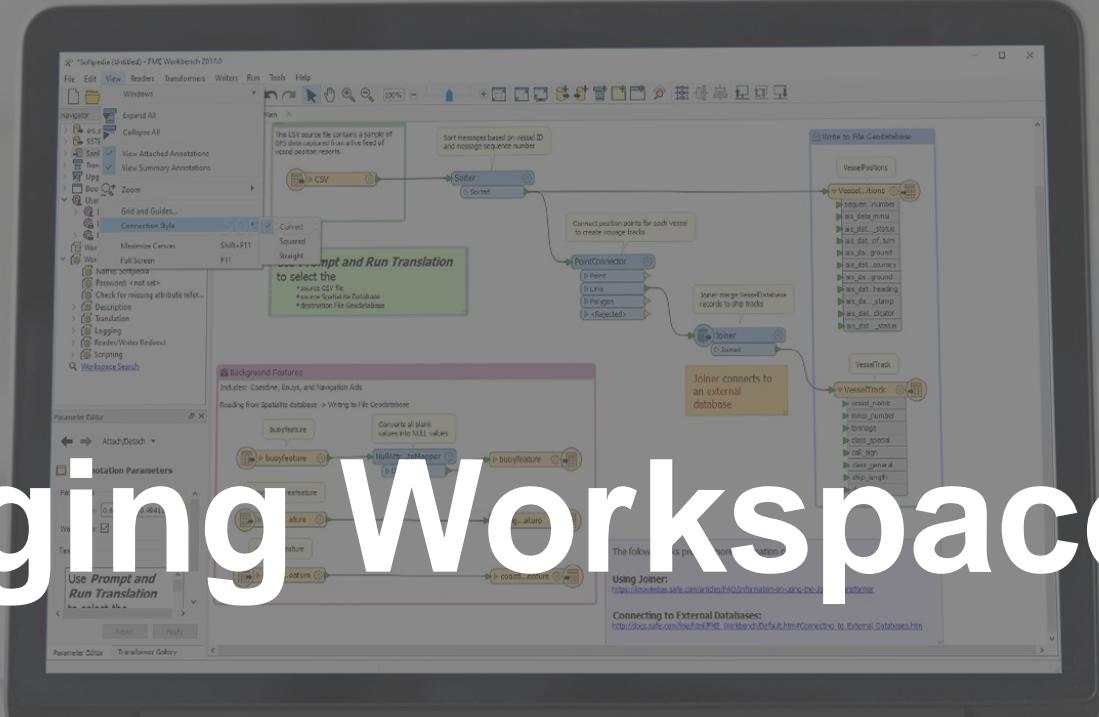


Table  
DonutHoleExtractor: <Rejected>

	ParkName	fme_rejection_code
1	Tea Swamp Park	INVALID_GEOMETRY_TYPE
2	Mcbride Park	INVALID_GEOMETRY_TYPE
3	Creekside Park	INVALID_GEOMETRY_TYPE
4	Arbutus Greenway Park	INVALID_GEOMETRY_TYPE
5	<missing>	INVALID_GEOMETRY_TYPE
6	Moira Matthews Park	INVALID_GEOMETRY_TYPE

# Debugging Workspaces



Connect. Transform. **Automate**

# Debugging



Don't worry, this has nothing to do with coding!

- Debugging in FME consists of first determining whether a problem has occurred and then tracking down the problem's source
- Even skilled FME users seldom produce new workspaces with zero defects, so debugging is an important skill



The logical order of debugging is:

- Check for any problem
  - Interpret the **log** for warnings and errors
  - Inspect the **output datasets**
- Locate the problem
  - Review connection **feature counts**
  - Inspect the data at key stages of the translation
- Determine the problem
  - Check reader, writer, or transformer parameters at the point of failure

# Debugging – Log Interpretation



FME logs contain a record of all stages and processes within a translation. The contents are vital for debugging purposes

## Log Message Types:

- **Error:** denoted in the log by red text and the term ERROR, indicates that a problem has caused FME to cease processing. For example, FME is unable to write the output dataset because of incorrect user permissions.
- **Warning:** denoted by blue text and the term WARN, indicates a processing problem. The problem is sufficiently minor to allow FME to complete the translation, but you may want to check if the output was adversely affected. For example, FME is unable to write features because its geometry is incompatible with the writer format. FME Workbench will drop the features from the translation and issue a warning in the log.
- **Information:** denoted by the term INFORM, indicate a piece of information that may help users determine whether their translation functioned correctly. For example, FME sometimes logs confirmation of a particular dataset parameter, such as the coordinate system.
- **Statistics:** denoted by the term STATS, provide information on various numbers relating to the translation; for example, the number of features FME read from a source dataset and the time it took to do so.

Translation Log		
	Transformer	Message
92	TestFilter	TestFilter_Condition_2 (TestFactory): Tested 383 input feature(s) -- 51 feature(s) passed
93	TestFilter	TestFilter_<lt>UNFILTERED<gt> (TeeFactory): Cloned 332 input feature(s) into 332 output
94	TestFilter	TestFilter_<lt>UNFILTERED<gt> Transformer Output Nuker (TeeFactory): Cloned 332 input fe
95		TestFilter_Industrial Feature Counter -1 8 (TeeFactory): Cloned 33 input feature(s) into
96		TestFilter_Commercial Feature Counter -1 9 (TeeFactory): Cloned 51 input feature(s) into
97	AttributeSplitter	AttributeSplitter (TeeFactory): Cloned 84 input feature(s) into 84 output feature(s)
98		AttributeSplitter_OUTPUT Feature Counter -1 10 (TeeFactory): Cloned 84 input feature(s)
99		Destination Feature Type Routing Correlator (RoutingFactory): Tested 84 input feature(s)
100		Final Output Nuker (TeeFactory): Cloned 0 input feature(s) into 0 output feature(s)
101		=====
102		Feature output statistics for 'CSV2' writer using keyword 'CSV2_1':
103		=====
104		Features Written
105		=====
106		zoning cats 84
107		=====
108		Total Features Written 84
109		=====
110		Features Read Summary
111		=====
112		Zones 416
113		=====
114		Total Features Read 416
115		=====
116		Features Written Summary
117		=====
118		zoning cats 84
119		=====
120		Total Features Written 84
121		=====

# Debugging – Output and Feature Counts



## Inspect Output

Even if a workspace ran to completion without warnings or errors, it does not follow that the output matches what is expected or required.

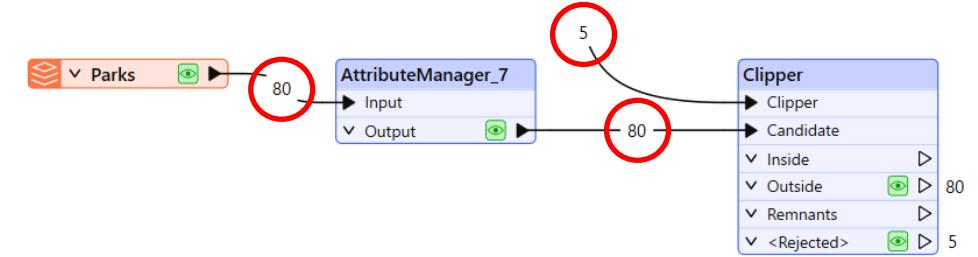
For whatever reason, the workspace may be producing data in the wrong manner

You should inspect several aspects of your output data:

- **Format:** Is the data in the expected format?
- **Schema:** Is the data subdivided into the correct layers, categories, or classes?
- **Geometry:** Is the geometry in the correct spatial location? Are the geometry types suitable?
- **Symbology:** Is the colour, size, and style of each feature correct?
- **Attributes:** Are all the required attributes present? Are all integrity rules being followed?
- **Quantity:** Does the data contain the correct number of features?
- **Output:** Has the translation process restructured the data as expected?

## Feature Counts

Once you find an error or problem, feature counts help us identify where that problem occurred



Turning on feature caching helps to confirm where features have gone



## Exercise 5.1



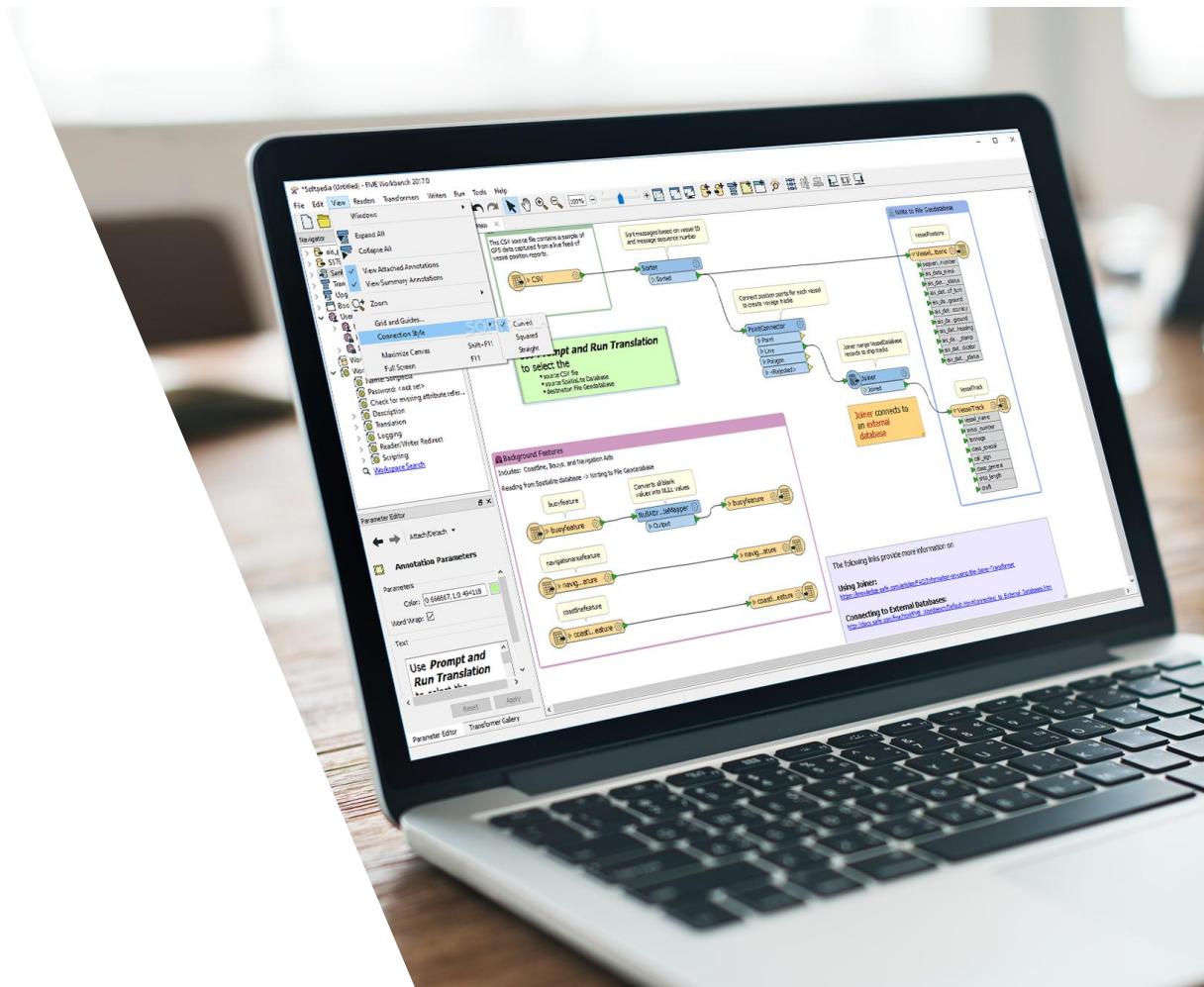
# Debugging a Workspace

- You have just taken over a project from your colleague and they've passed their workspace on to you.
  - This project is to calculate the "walkability" of each address in the city of Vancouver. Walkability is a measure of how easy it is to access local facilities on foot.
  - Currently the workspace calculates a number of metrics but not walkability – lets add that in

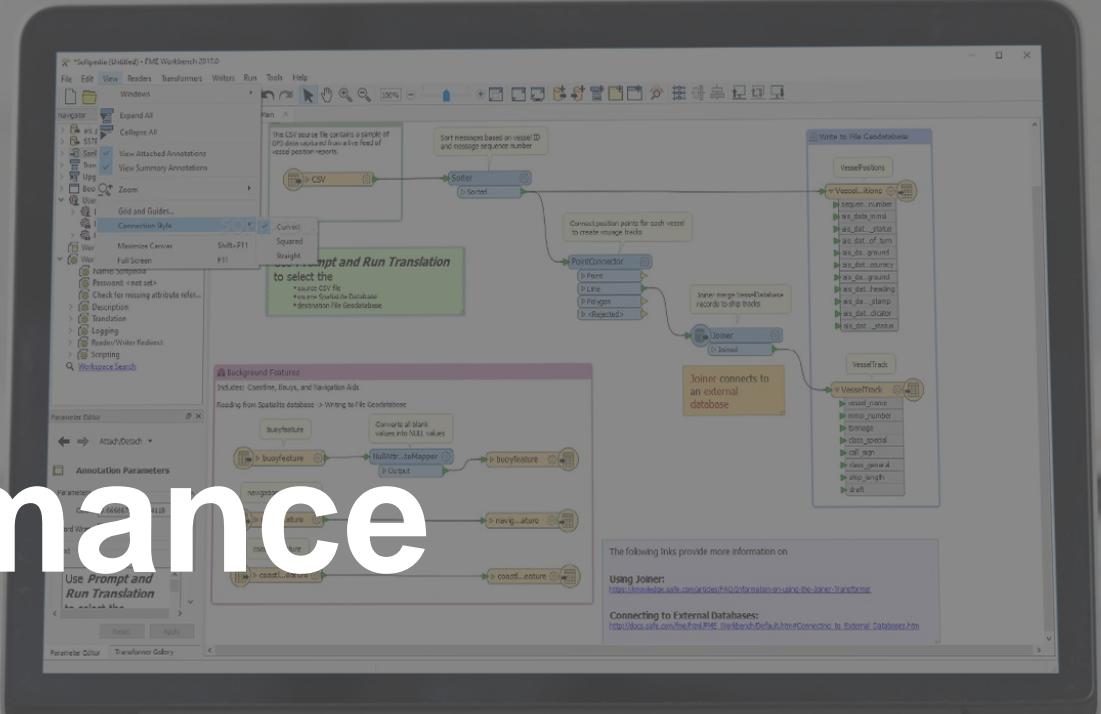
**Data - Addresses (ESRI Geodatabase), Crime Data (CSV), Parks (MapInfo TAB), Swimming Pools (OSM)**

**Starter Workspace - C:\FMEModularData\Workspaces\5.01-BestPractice-Debugging-Begin.fmw**

## Goal – Add to the workspace and debug any issues



# Performance

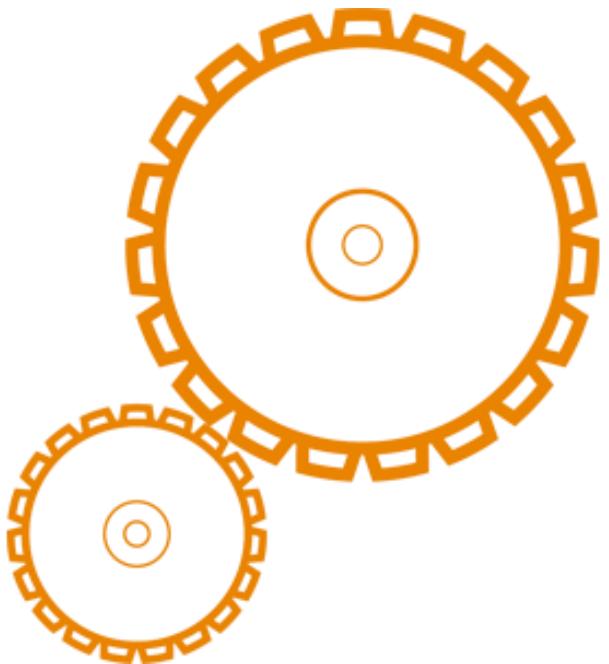


Connect. Transform. **Automate**

# Performance Methodology



A weak workspace design causes the workspace to use more system resources (memory and CPU) than necessary.



**Filtering Input**

**Excess Feature Types**

**Excess Attributes and Lists**

**Number of Transformers**

**Transformer Choice**

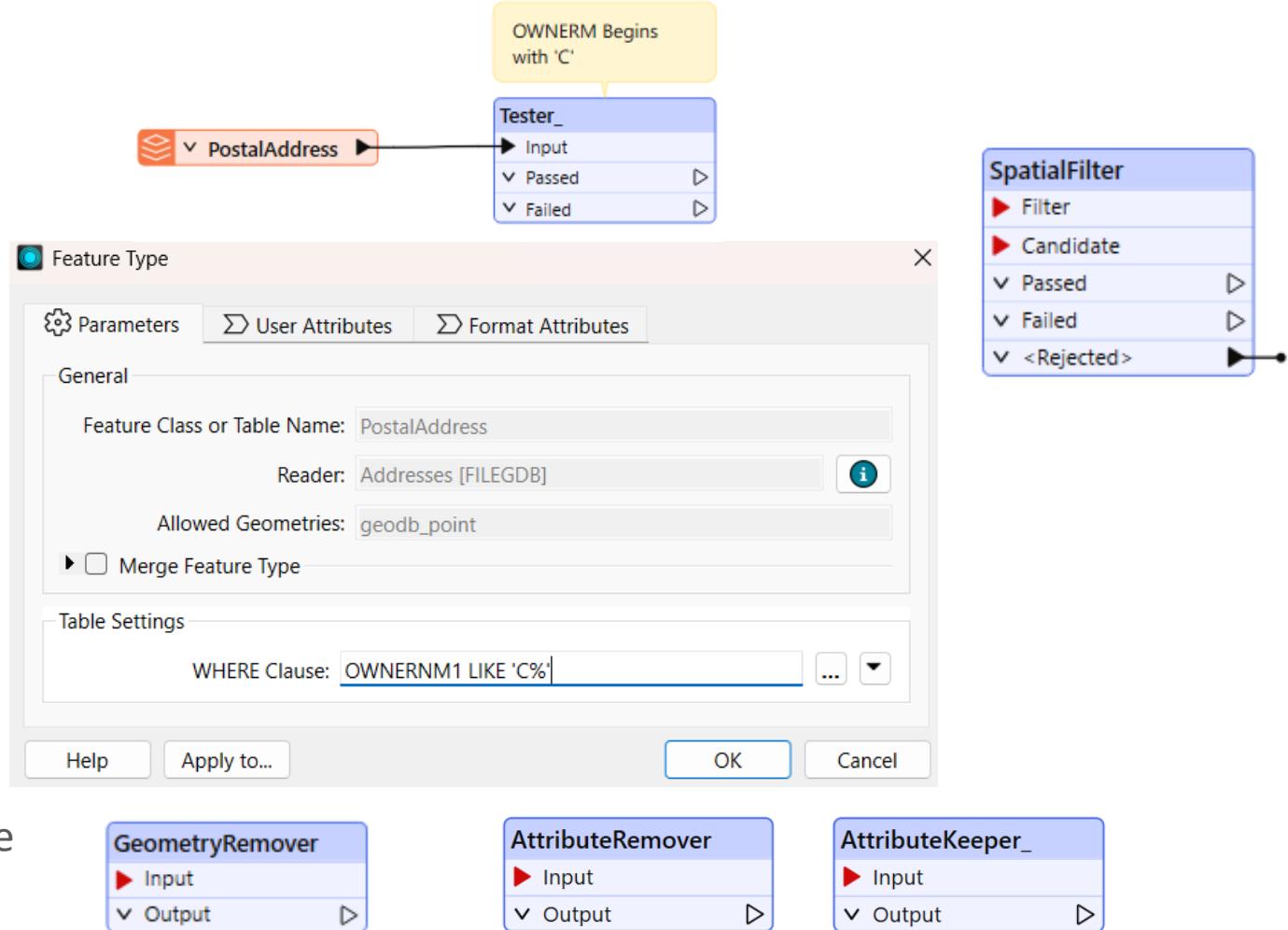
# Performance - Excess Data



Remove unnecessary data – this has the biggest impact on performance

Things to think about:

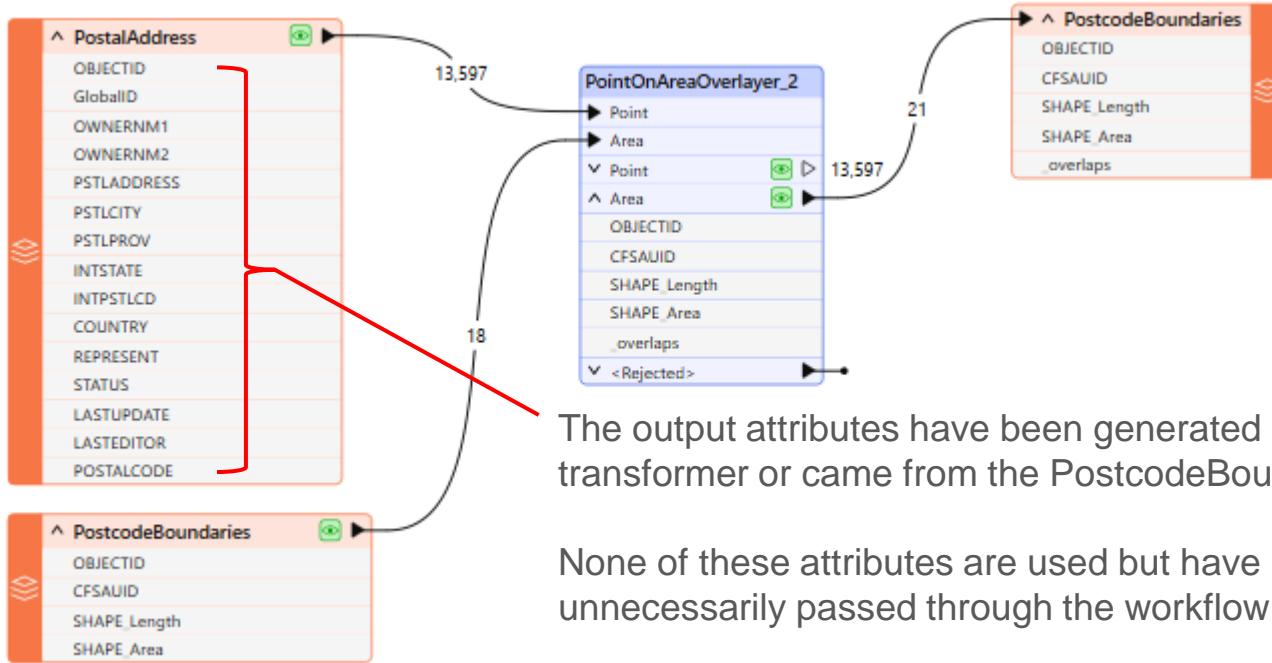
- **Filtering Input**
  - Filters / Testers
  - WHERE Clauses on Reader
- **Excess Feature Types**
- **Excess Attributes**
- **Lists**
- **Geometry**
  - geometry can be removed from a feature
  - spatial attributes can be very large



# Performance - Excess Data



## Excess Attributes



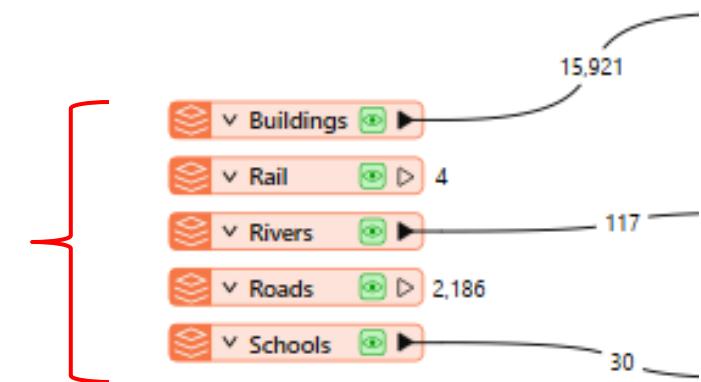
The output attributes have been generated by the transformer or came from the PostcodeBoundaries.

None of these attributes are used but have been unnecessarily passed through the workflow!

Only three of these five feature types are being routed into the workflow.

However, features for all the feature types are being unnecessarily read in!

## Excess Feature Types



# Group-based v Feature-based Transformers

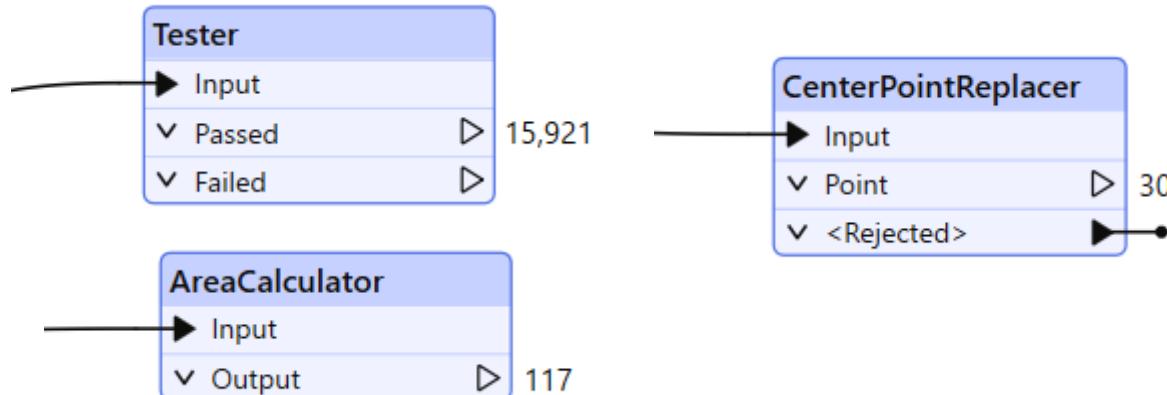


Before we talk about Transformers and performance, first we need to understand the difference between Group-based and Feature-based transformers:

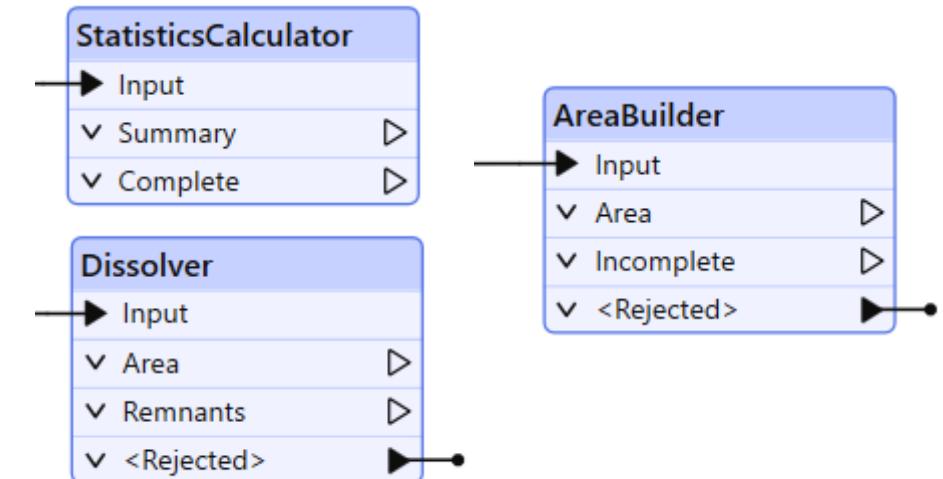
Transformers carry out transformations on either:

- One feature at a time – **Feature based**
- The whole set of features at once – **Group based** – *these are more resource intensive*

Feature based examples



Group based examples



# Performance - Transformers

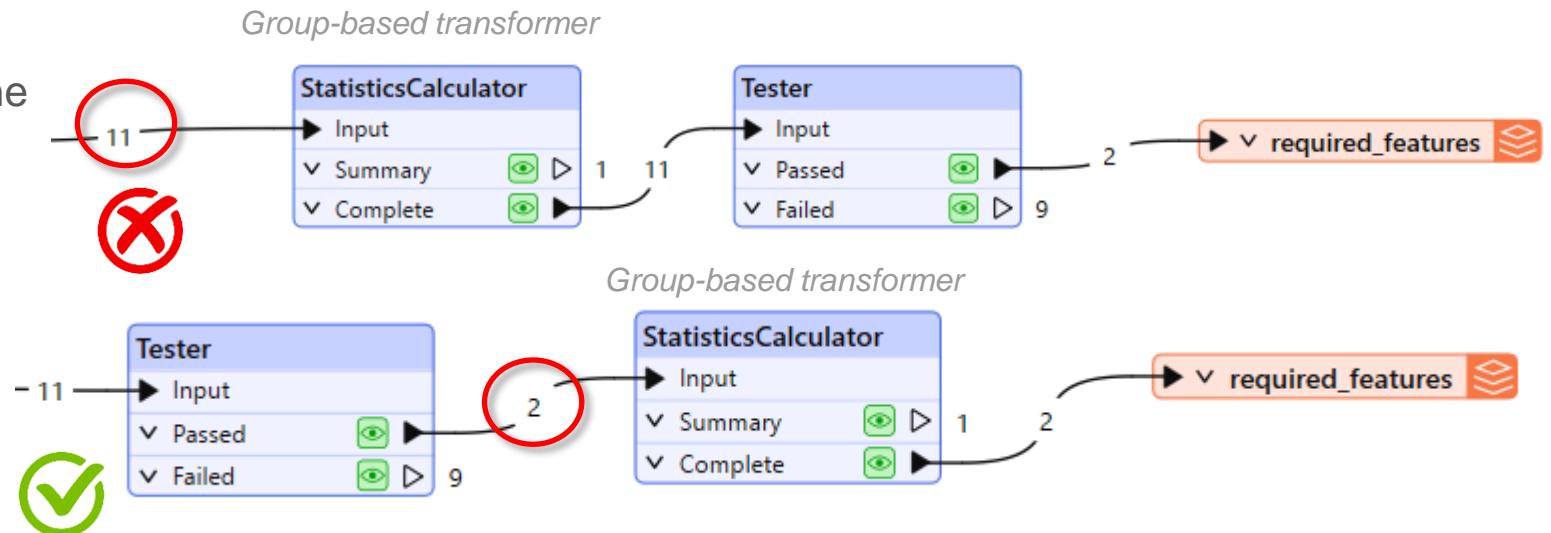


The number of transformers, transformer order and transformer type affect workspace performance.

- The key to improving transformer performance is to reduce the amount of memory used
- As a general rule, the greater the **number of transformers** in a workspace, the more system resource (memory and CPU) and time is required to run the workspace
- Although the **order of transformers** can sometimes vary without affecting the result, at other times it is important to get the correct order for performance reasons.

e.g. where possible put feature-based filter transformers *before* the group-based process, not after

*The best way to remember this is: Filter, Remove, Action!*

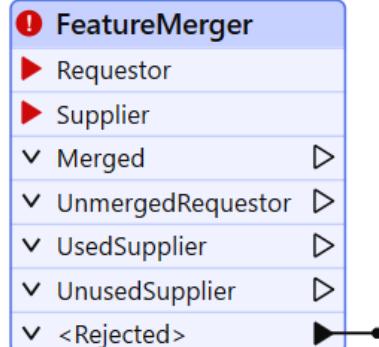


# Transformer Use when thinking about performance



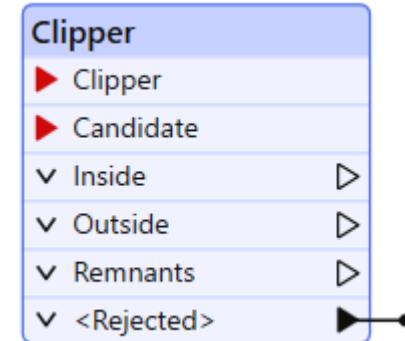
## Joiner vs Merger

FeatureJoiner is faster than the Merger  
– use the Joiner it if you can



## Clipper

Clip spatial data to remove unnecessary features, will reduce memory usage



## Group based transformers

Utilize the 'Group By' mode where you can – less data is stored in memory and processing is more efficient.

The condition for applying this parameter is that the incoming features are pre-sorted into their groups.

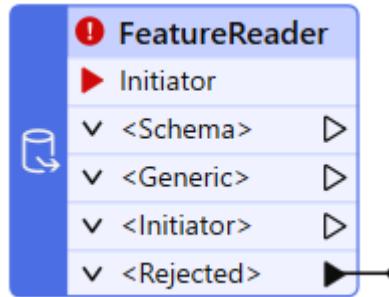
## SpatialFilter vs SpatialRelator

SpatialFilter is more efficient than the Relator – however, the Clipper is even better!

# Transformer Use when thinking about performance



FeatureReader is useful for not reading entire datasets when you don't need all the data



▼ Constraints

Feature Types to Read: [redacted]

WHERE Clause: [redacted]

Spatial Filter: <No Spatial Filter>

Clip to Initiator Envelope:

Max Features to Read: [redacted]

## 'Features First'

Some transformers have their own, unique, parameters for performance improvements.  
Many of these specify one type of feature to arrive "first."

e.g. the FeatureMerger

Transformer Name: FeatureMerger

>  Group Processing

Transformer Mode

Suppliers First:  No  
Yes  
No

Join On

or the PointOnAreaOverlayer

General

Areas First:  No  
Yes  
No

Aggregate Handling:  Yes  
No

# Exercise 5.2



## Methodology – Performance

- You have inherited a workspace from a colleague and already made some changes to calculate a ‘walkability’ score.
- Next you now need to improve the performance of the workspace

**Data** - Addresses (ESRI Geodatabase), Crime Data (CSV), Swimming Pools (OSM)

### Starter Workspace –

- C:\FMEModularData\Workspaces\5.02-BestPractice-Performance-Begin.fmw
- *or carry on from previous exercise*

**Goal** – Improve the performance of the workspace

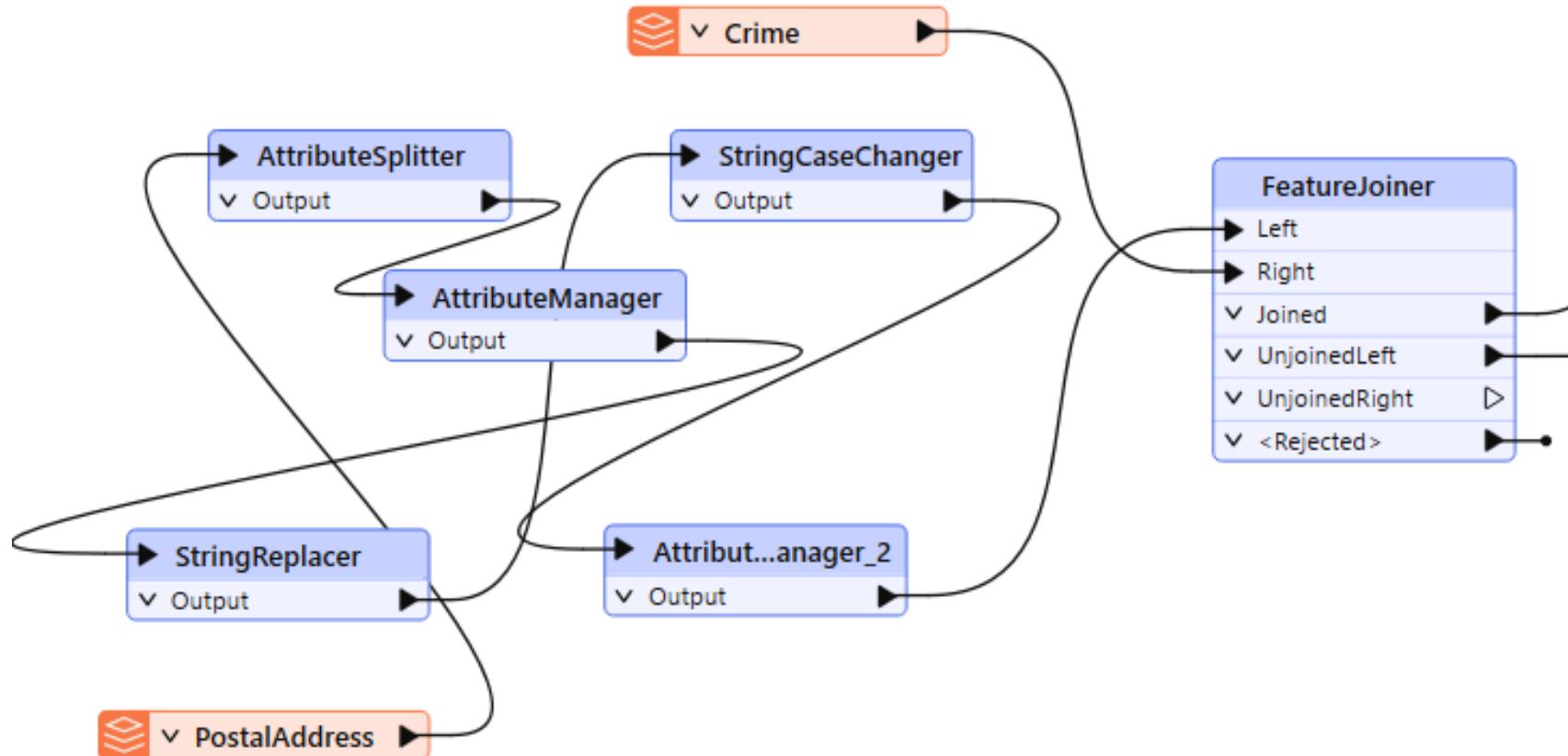
# Workspace Layout & Styling

Connect. Transform. **Automate**

# Style and Layout



Can you tell what this does and why?....

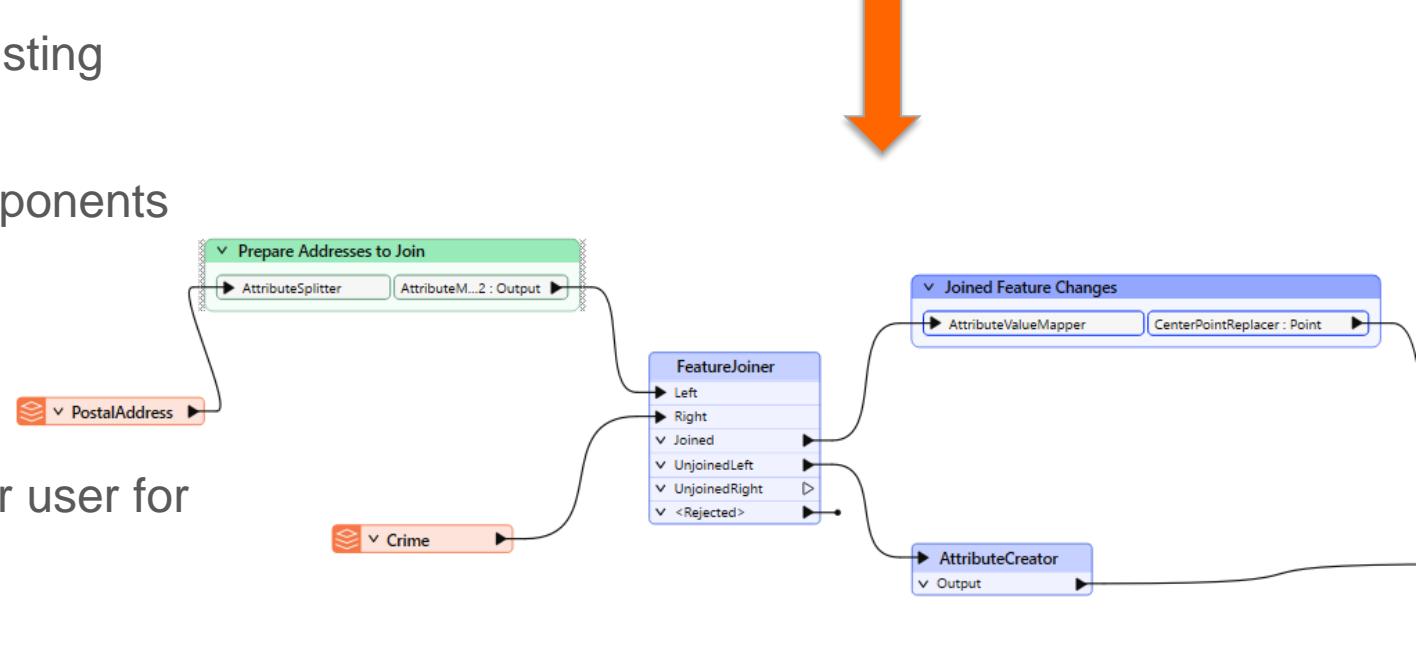
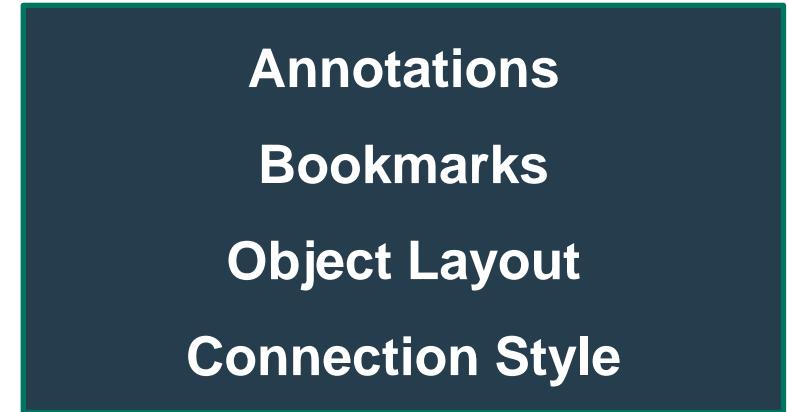


..... NO!

# Style and Layout



- Most obvious component of FME Best Practice!
- Well-designed Workspace demonstrates competence
- Benefits:
  - easier to navigate and understand an existing workspace
  - distinctly define different sections or components of a workspace
  - quickly navigate to a specified section or particular transformer
  - easier to pass a workspace on to another user for editing
  - easier to develop existing FME projects



# Style and Layout

---



## Annotations

- User annotation – created by the user and can be connected to an object or float freely
- Summary annotation – FME generated comment, automatically updates when parameters change

## Bookmarks

- Logically group together parts of a process
- Now Collapsible, feature caching only caches the output

## Object Layout

- Choose a sensible order – left to right – Avoid Overlaps
- Grids & Guides - Helps structure and line up workspace objects. Can snap to grid or guides
- Attribute Order – up/down to avoid connector line crossover
- AutoLayout

## Connection Lines

- Straight / Curved / Squared
- Hiding Connections – Tunnels
- Junctions / Vertices

# Exercise 5.3



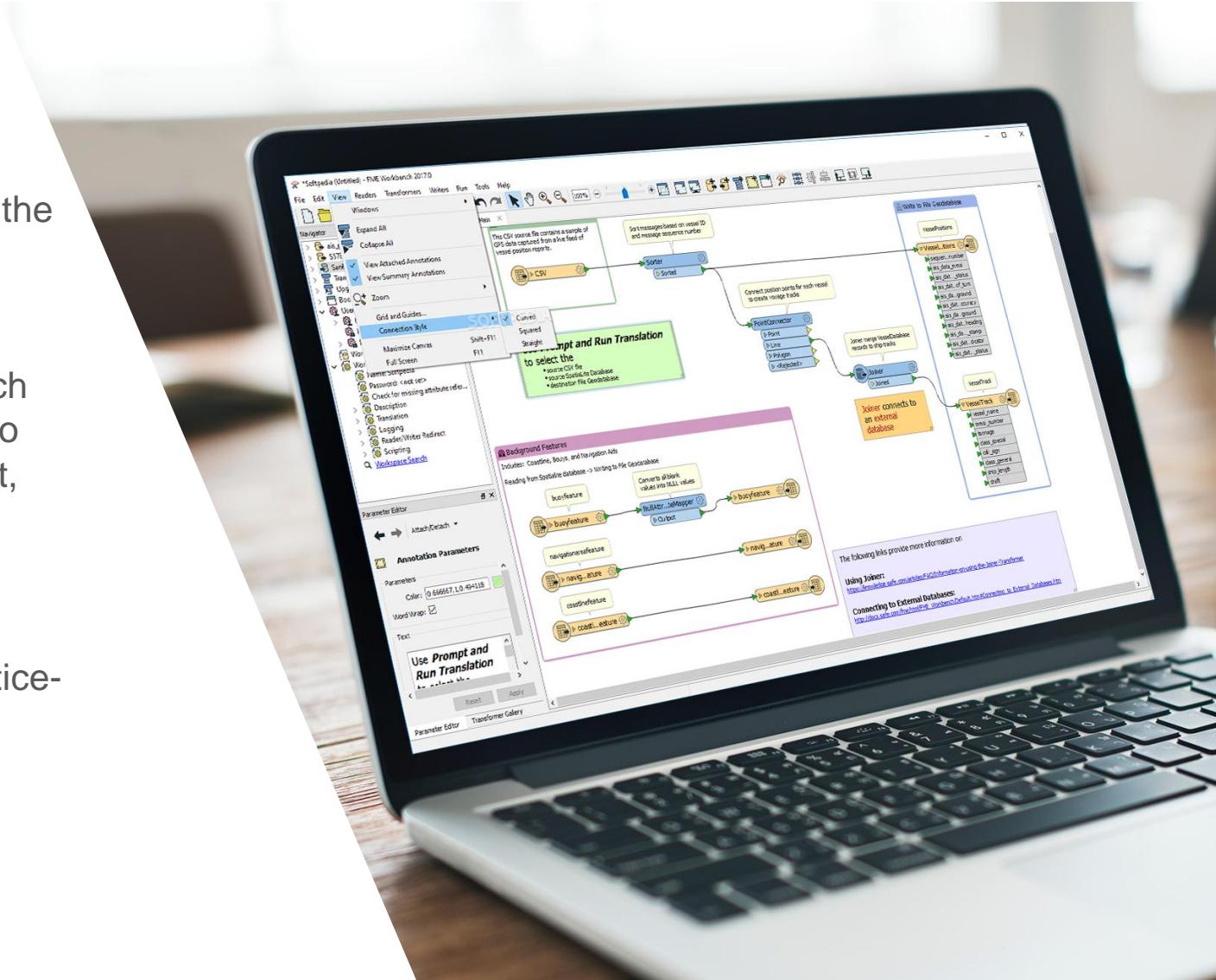
## Workspace Layout and Styling

- You have been assigned to a project to calculate the ‘walkability’ of each address in the city.
- Your colleague wasn't aware of FME style best practices when they gave us the workspace, which made working with it a bit challenging. We need to present our workspace, so we want it to look neat, organized, and well-documented

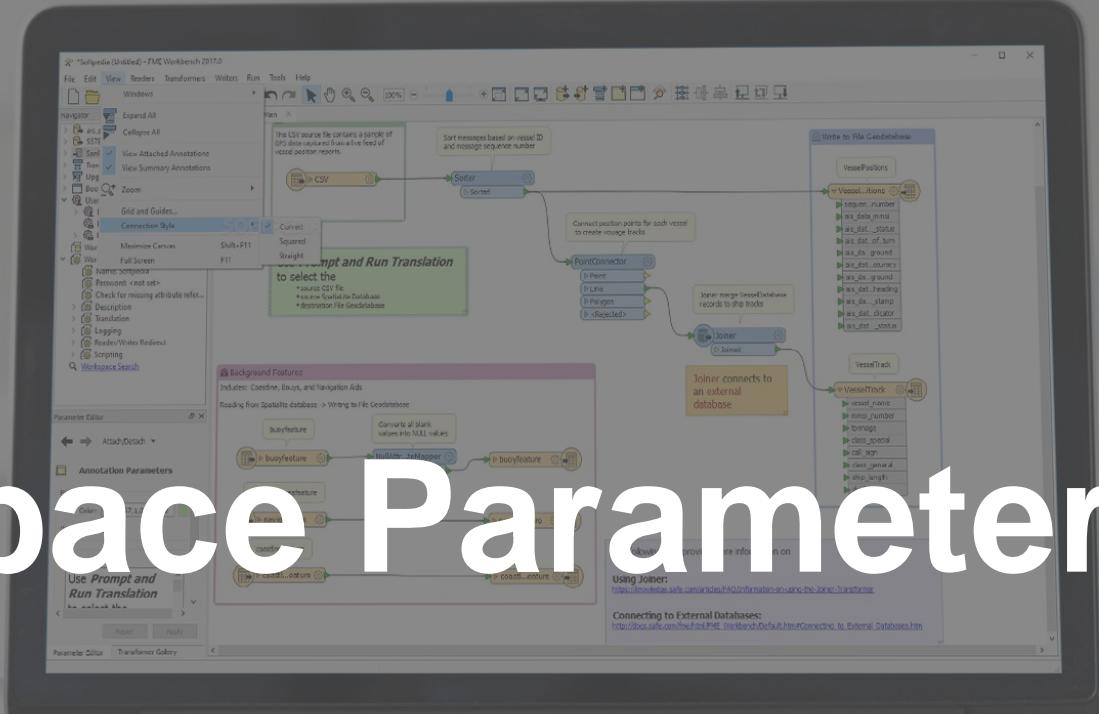
### Starter Workspace –

- C:\FMEModularData\Workspaces\5.03-BestPractice-WorkspaceStyling-Begin.fmw
- or carry on from previous exercise

### Goal – Implement best practice styling



# Workspace Parameters



Connect. Transform. **Automate**

# Workspace Description



The **Workspace Parameters** are located in the Navigator Panel

This includes a **Description** section, which should be populated with details of the workspace.

*- This is especially important and good practice if you are sharing or publishing the workspace*

Navigator x

- > Addresses [FILEGDB]
- > Crime [CSV2]
- > leisure [OSM]
- > Transformers (20)
- > Upgradeable Transformers (13)
- > Bookmarks (2)
- > User Parameters (2)
- > FME Flow Parameters
- < Workspace Resources
- < Workspace Parameters
  - >Password: <not set>
  - Name: BestPractice-Performance-Complete
  - < Description
    - Category: <not set>
    - Overview: <not set>
    - Help: <not set>
    - Use Markdown: No
    - History: <not set>
    - Last Save Date: 05/08/2022 13:39
    - Last Save Build: FME(R) 2022.0.0.1 (20220505 - Build 22339 - WIN64)
  - < Logging
  - < Reader/Writer Redirect
  - < Scripting
  - < Translation
- < Workspace Search...

# Workspace Description

---

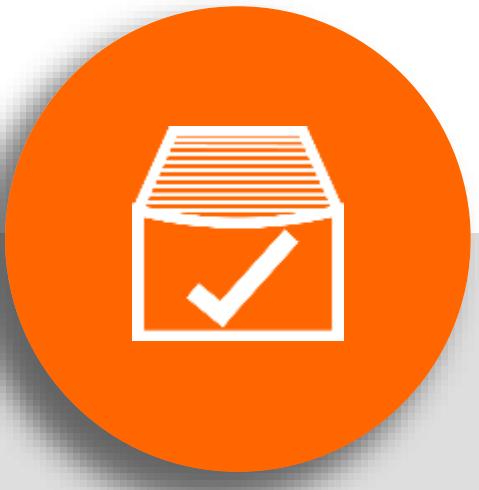


- **Name** - Update the default name, if desired. The name is displayed on the title bar of the Workbench window.
- **Overview** - Enter a description. This is particularly important. You can also use this area or the **Help** area to:
  - List any requirements that must be met before using the workspace or template, such as where the source data resides or minimum FME version required. You should also list any dependencies (for example, if you need a third-party application or an extra-cost transformer).
  - Record any legal information (for example, copyrights or data restrictions).

➤ When you are using the workspace via an FME Server the information in this Overview section is seen on the web page to describe what the process is intended to carry out.
- **History** - use this section to record changes made to the workspace.
- **Last Save Date, Last Save Build** - the date and FME build on which the workspace was last saved – these will automatically populate/update upon saving the workspace

# Resources

---



Transformer Gallery



Community pages and forums



Knowledge Base



Each Other

A blurred background image of a person speaking at a podium with a microphone, gesturing with their hand.

**Thanks for watching!**

**Good luck getting started with FME.**