

FME® Desktop Basic Training Course



Table of Contents

Introduction	1.1
About This Document	1.2
Course Overview	1.2.1
Course Resources	1.2.2
Data Translation Basics	1.3
What is FME?	1.3.1
How FME Works	1.3.2
FME Desktop Components	1.3.3
Introduction to FME Workbench	1.3.4
FME Workbench Components	1.3.5
Window Control in Workbench	1.3.6
Creating a Translation	1.3.7
The New Workspace	1.3.8
Running a Translation	1.3.9
Exercise: Basic Workspace Creation	1.3.10
Data Inspection	1.3.11
Introduction to FME Data Inspector	1.3.12
FME Data Inspector Components	1.3.13
Viewing Data in the FME Data Inspector	1.3.14
Querying Data in the FME Data Inspector	1.3.15
Exercise: Basic Data Inspection	1.3.16
More Data Inspector Functionality	1.3.17
More Data Inspector Functionality	1.3.18
Exercise: The FME Data Inspector	1.3.19
Data Translation Previews	1.3.20
Module Review	1.3.21
Q+A Answers	1.3.22
Exercise: Tourist Bureau Project	1.3.23
Data Transformation	1.4
What is Data Transformation	1.4.1
Structural Transformation	1.4.2
Schema Editing	1.4.3
Schema Mapping	1.4.4
Exercise: Grounds Maintenance Project - Schema Editing	1.4.5
Transformation with Transformers	1.4.6
Exercise: Grounds Maintenance Project - Structural Transformation	1.4.7
Content Transformation	1.4.8
Transformers in Series	1.4.9
Exercise: Grounds Maintenance Project - Calculating Statistics	1.4.10

Feature Count Display	1.4.11
Transformers in Parallel	1.4.12
Exercise: Grounds Maintenance Project - Labelling Features	1.4.13
Group-By Processing	1.4.14
Exercise: Grounds Maintenance Project - Neighborhood Averages	1.4.15
Data Inspection from Workbench	1.4.16
Coordinate System Transformation	1.4.17
Exercise: Grounds Maintenance Project - Data Reprojection	1.4.18
Module Review	1.4.19
Q+A Answers	1.4.20
Exercise: Voting Analysis Project	1.4.21
Translation Components	1.5
Key Components	1.5.1
Component Hierarchy	1.5.2
Workspaces	1.5.3
Exercise: Fundraising Walk - Creating a Workspace	1.5.4
Readers	1.5.5
Reader Parameters	1.5.6
Reader Dataset Parameter	1.5.7
Exercise: Fundraising Walk - Adding Readers	1.5.8
Reader Feature Types	1.5.9
Importing Reader Feature Types	1.5.10
Updating Reader Feature Types	1.5.11
Reader Feature Type Parameters	1.5.12
Exercise: Fundraising Walk - Reader Feature Types	1.5.13
Managing Reader Datasets	1.5.14
Exercise: Fundraising Walk - Unexpected Input	1.5.15
Unexpected Input	1.5.16
Dealing with Unexpected Input	1.5.17
Exercise: Fundraising Walk - Dealing with Unexpected Input	1.5.18
Writers	1.5.19
Controlling Writers	1.5.20
Exercise: Fundraising Walk - Adding Writers	1.5.21
Writer Feature Types	1.5.22
Add Writer Feature Types	1.5.23
Exercise: Fundraising Walk - Managing Writer Feature Types	1.5.24
Copy/Remove Writer Feature Types	1.5.25
Import Writer Feature Types	1.5.26
Writer Feature Type Parameters	1.5.27
Module Review	1.5.28
Q+A Answers	1.5.29
Exercise: Elevation Model Updates	1.5.30

Practical Transformer Use	1.6
The Transformer Gallery	1.6.1
Transformer Searching	1.6.2
Most Valuable Transformers	1.6.3
Managing Attributes	1.6.4
Creating and Setting Attributes	1.6.5
Constructing Attributes	1.6.6
Constructing Transformer Parameters	1.6.7
Renaming and Copying Attributes	1.6.8
Bulk Attribute Renaming	1.6.9
Removing Attributes	1.6.10
Exercise: Address Open Data Project	1.6.11
Conditional Filtering	1.6.12
Tester and TestFilter	1.6.13
Other Key Filter Transformers	1.6.14
Exercise: Noise Control Laws Project	1.6.15
Data Joins	1.6.16
Attribute Joins	1.6.17
Spatial Joins	1.6.18
Module Review	1.6.19
Q+A Answers	1.6.20
Exercise: Crime Mapping Data Request	1.6.21
Best Practice	1.7
Style	1.7.1
Annotating Workspaces	1.7.2
Bookmarks	1.7.3
Bookmarks for Sectioning	1.7.3.1
Bookmarks for Access	1.7.3.2
Bookmarks for Editing	1.7.3.3
Object Layout	1.7.4
Connection Styles	1.7.5
Exercise: The FME Style Guide	1.7.6
Methodology	1.7.7
Prototyping	1.7.8
Reusing Resources	1.7.9
Templates	1.7.10
Workspace Searching	1.7.11
Format Best Practices	1.7.12
Transformer Best Practices	1.7.13
Exercise: Design Patterns	1.7.14
Debugging	1.7.15
Logging	1.7.16

Interpreting the Log	1.7.17
Feature Counts	1.7.18
Inspecting Output	1.7.19
Testing a Workspace	1.7.20
Exercise: Debugging a Workspace	1.7.21
Feature Debugging	1.7.22
Module Review	1.7.23
Q+A Answers	1.7.24
Exercise: Shortest Route Hackathon	1.7.25
Course Wrap-Up	1.8
Product Information and Resources	1.8.1
Community Information and Resources	1.8.2
Feedback and Certificates	1.8.3
Thank You	1.8.4
Exercise: A Fun Challenge!	1.8.5

FME Desktop Basic Training Manual

This is the manual for the introductory-level training course for Safe Software's FME Desktop application.



The training will introduce basic concepts and terminology, help students become efficient users of FME, and direct you to resources to help apply the product to your own needs.

Course Structure

The full course is made up of five main sections. These sections are:

- Data Translation Basics
- Data Transformation
- Translation Components
- Practical Transformer Use
- Best Practice

Current Status

The current status of this manual is: **COMPLETE**: this manual **can** be used for training.

This manual applies to **FME2017.0** and **FME2017.1**

The status of each chapter is:

- Chapter 0: Complete content. No exercises
- Chapter 1: Complete content and exercises
- Chapter 2: Complete content and exercises
- Chapter 3: Complete content and exercises
- Chapter 4: Complete content and exercises
- Chapter 5: Complete content and exercises
- Chapter 6: Complete content. No exercises

- Slides: Complete
- FMEDData: Complete
- Course Outline: Updated

NB: Even for completed content, Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice. See the full licensing agreement for further details.

About This Document

This manual is the introductory-level training course for FME Desktop.



Look out for residents of the City of Interopolis, who will appear from time-to-time to give you advice and dispense FME-related wisdom. In fact, here comes someone now:

Mr. E.Dict (Attorney of FME Law) says...

On behalf of the City of Interopolis, welcome to this training course. Here is the standard legal information about this training document and the datasets used during the course.

Be sure to read it, particularly if you're thinking about re-using or modifying this content.

Licensing and Warranty

Permission is hereby granted to use, modify and distribute the FME Tutorials and related data and documentation (collectively, the “Tutorials”), subject to the following restrictions:

1. The origin of the Tutorials and any associated FME® software must not be misrepresented.
2. Redistributions in original or modified form must include Safe Software’s copyright notice and any applicable Data Source(s) notices.
3. You may not suggest that any modified version of the Tutorials is endorsed or approved by Safe Software Inc.

4. Redistributions in original or modified form must include a disclaimer similar to that below which: (a) states that the Tutorials are provided “as-is”; (b) disclaims any warranties; and (c) waives any liability claims.

Safe Software Inc. makes no warranty either expressed or implied, including, but not limited to, any implied warranties of merchantability, non-infringement, or fitness for a particular purpose regarding these Tutorials, and makes such Tutorials available solely on an “as-is” basis. In no event shall Safe Software Inc. be liable to anyone for direct, indirect, special, collateral, incidental, or consequential damages in connection with or arising out of the use, modification or distribution of these Tutorials.

This manual describes the functionality and use of the software at the time of publication. The software described herein, and the descriptions themselves, are subject to change without notice.

Data Sources

City of Vancouver

Unless otherwise stated, the data used here originates from open data made available by the [City of Vancouver](#), British Columbia. It contains information licensed under the Open Government License - Vancouver.

Others

Forward Sortation Areas: Statistics Canada, 2011 Census Digital Boundary Files, 2013. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada. © This data includes information copied with permission from Canada Post Corporation.

Digital Elevation Model: GeoBase®

Fire Hall Data: Some attribute data adapted from content © 2013 by [Wikipedia](#), used under a Creative Commons Attribution-ShareAlike license

Stanley Park GPS Trail: Used with kind permission of [VancouverTrails.com](#).

Copyright

© 2005–2017 Safe Software Inc. All rights are reserved.

Revisions

Every effort has been made to ensure the accuracy of this document. Safe Software Inc. regrets any errors and omissions that may occur and would appreciate being informed of any errors found. Safe Software Inc. will correct any such errors and omissions in a subsequent version, as feasible. Please contact us at:

Safe Software Inc.

Phone: 604-501-9985

Fax: 604-501-9965

Email: train@safe.com

Web: www.safe.com

Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice.

Trademarks

FME® is a registered trademark of Safe Software Inc. All brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Document Information

Document Name: FME Desktop Basic Training Manual 2017.1

Course Overview

This training course provides a framework for a basic understanding of FME. We find users come to master one function, but go home with many new FME uses.

The training will introduce basic concepts and terminology, help students become efficient users of FME, and direct you to resources to help apply the product to your own needs.

Course Structure

The full course is made up of five sections. These sections are:

- Data Translation Basics
- Data Transformation
- Translation Components
- Practical Transformer Use
- Best Practice

Course Length

The instructor may choose to cover as many of these sections as they feel are required, or possible, in the time permitted. They may also cover the course content in a different order and will skip or add new content to better customize the course to your needs.

Therefore the length and content of the course may vary, particularly when delivered online.

Safe Software offers training in various lengths, but usually either two days or five days. Both courses cover the same content. The two-day course is held over fewer, but longer, days and is more intense. The five-day course is held over more, but shorter, days and allows a little more time to cover the content from a beginner's perspective.

About the Manual

The FME Desktop training manual not only forms the basis for FME Desktop training – in-person or online – but is also useful reference material for future work you may undertake with FME. It is updated for each major release of FME.

All screenshots in these materials were taken using FME on Windows Server 2016. The fonts used (especially in screenshots of the log window) may be resized or otherwise changed for improved legibility.

.1 UPDATE

The FME development cycle includes a number of follow-up releases, numbered .1, .2, etc

A dialog box like this denotes features new or updated in FME2017.1, the first major update to the .0 release

Course Resources

A number of sample datasets and workspaces will be used in this course.

On Your Training Computer

The data used in this training course is based on open data from the City of Vancouver, Canada.

Most exercises ask you to assume the role of a city planner at the fictional city of Interopolis and to solve a particular problem using this data.

Whether it's a local computer or a virtual computer hosted in the cloud, you'll find resources for the examples and exercises in the manual at the following locations:

Location	Resource
C:\FMEData2017\Data	Datasets used by the City of Interopolis
C:\FMEData2017\Resources	Other resources used in the training
C:\FMEData2017\Workspaces	Workspaces used in the student exercises
C:\FMEData2017\Output	The location in which to write exercise output
< documents>\FME\Workspaces	The default location to save FME workspaces

You should also find FME pre-installed, plus a digital copy of this manual.

Please alert your instructor if any item is missing from your setup.

You can find the latest version of FME Desktop and FME Server for Windows, Mac, and Linux - together with the latest Beta versions - on the [Safe Software web site](#).

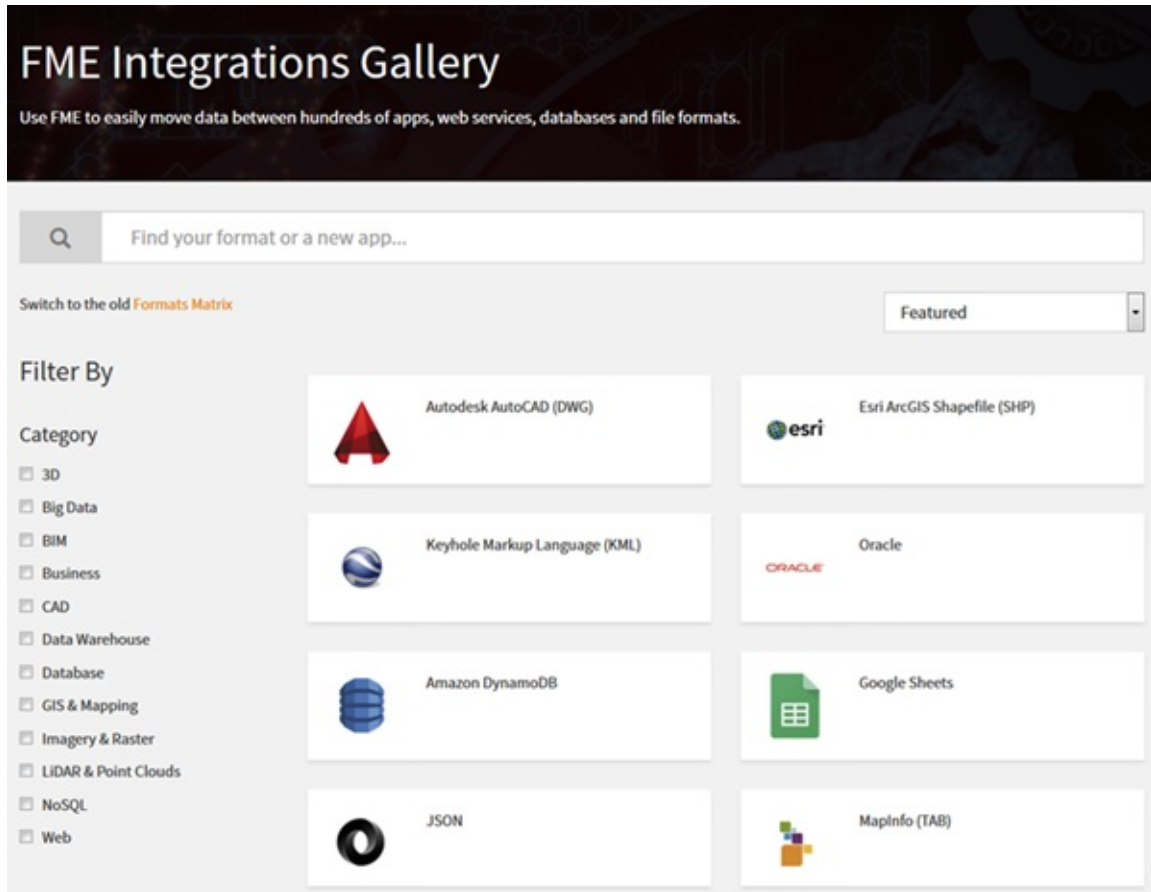
Course Etiquette

For online courses, please consider other students and test your virtual machine connection *before* the course starts. The instructor cannot help debug connection problems during the course!

For live courses, please respect other students' needs by keeping noise to a minimum when using a mobile phone or checking e-mail.

Data Translation Basics

At its heart, FME is a data translation tool, and this is usually the first aspect users wish to learn about.



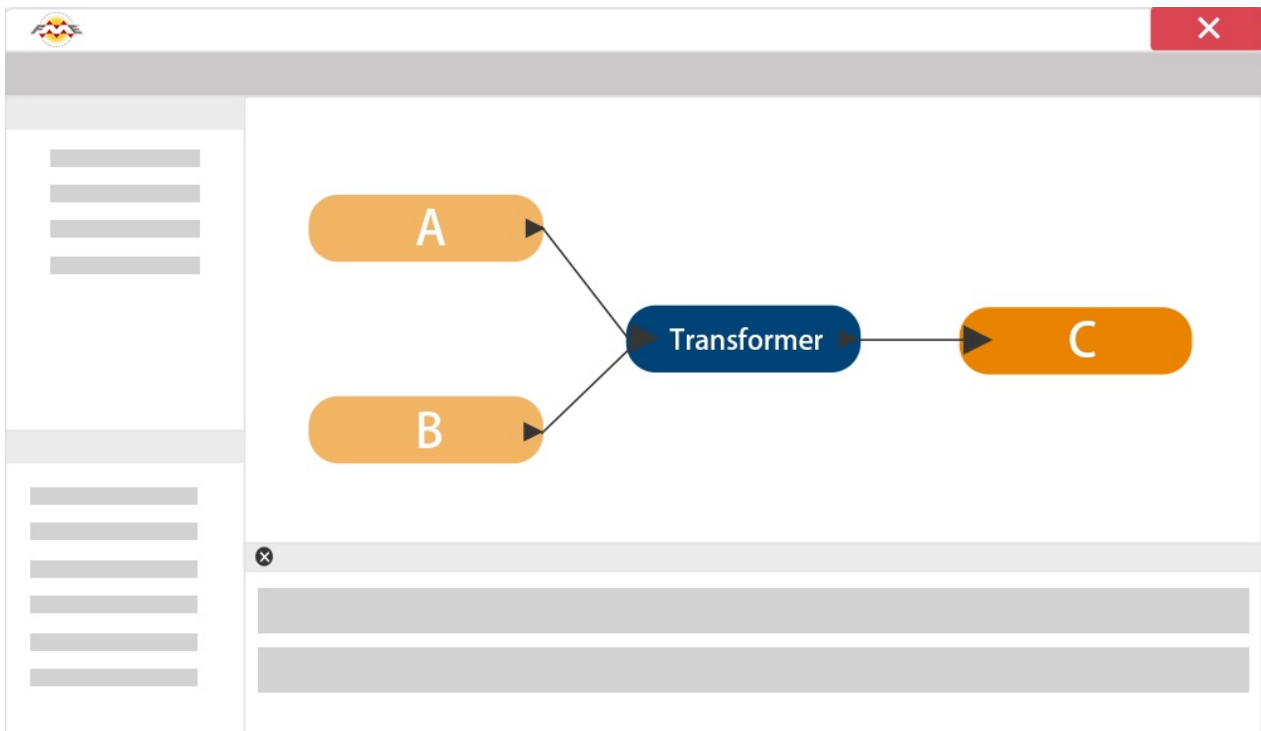
What is FME?

FME (the Feature Manipulation Engine) is a data translation and transformation tool for solving problems of data interoperability.

It is a way to integrate multiple data sources without the need for coding.

Extract, Transform, and Load

FME is sometimes classed as an **ETL** application. ETL stands for Extract, Transform and Load. It is a data warehousing tool that extracts data from multiple sources (here A and B), transforms it to fit the users' needs and loads it into a destination or data warehouse (C).



While most ETL tools process only tabular data, FME also has the geoprocessing capabilities required to handle spatial datasets, hence the term **Spatial ETL**.

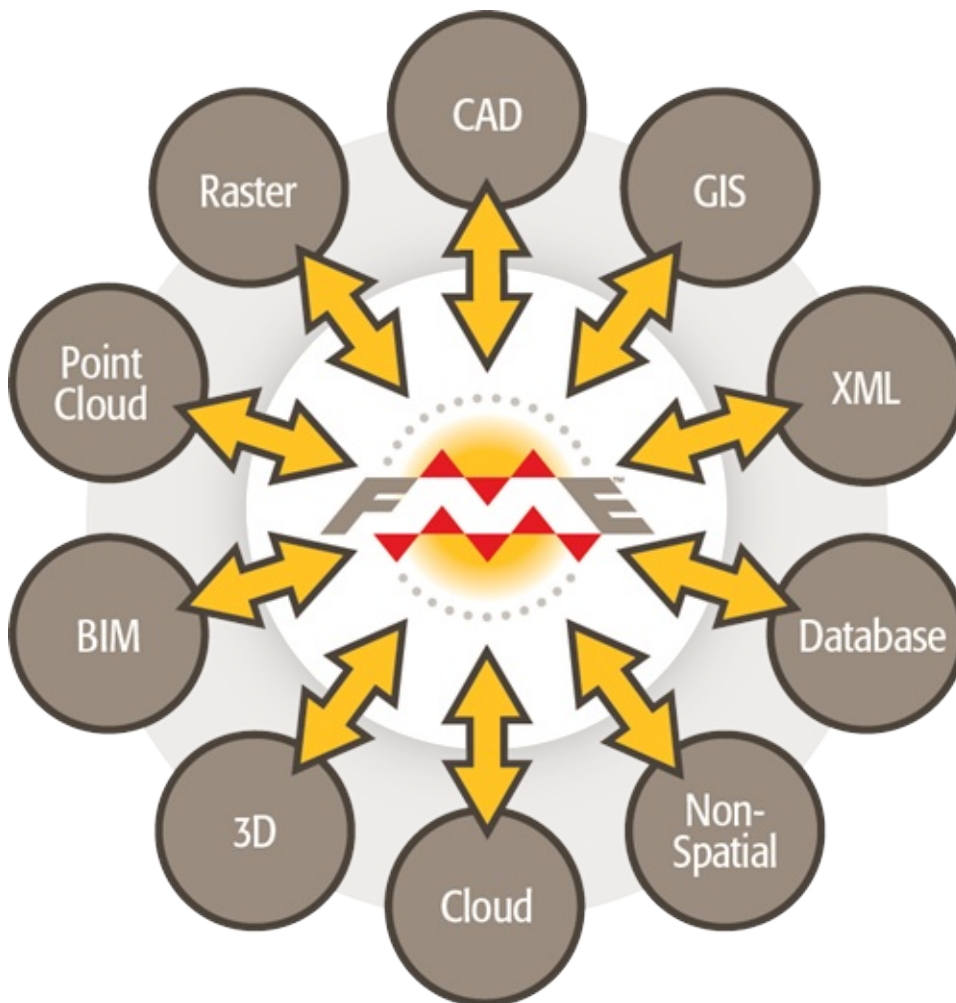
For more general information on FME and what it can do for you, see [our web site](#).

How FME Works

At the heart of FME is an engine that supports an array of data types and data formats; from GIS and CAD to BIM and Point Cloud, via XML, Raster, databases, and many more.

Rich Data Model

The capability to support so many data types is made possible by a rich data model that handles all possible geometry and attribute types.



Most importantly, the data translation process is seamless to the user; FME automatically converts between data types as required, and automatically substitutes one attribute or geometry type for another where the destination format does not support it.

Check out our web site for a [full list of data formats](#) supported in FME

Miss Vector says...

Attention students! I'm Miss Vector, FME schoolteacher. I'm here to give you tests on what you have learned. I really hope you don't get these questions wrong!

Q) ETL is an acronym for...?

- 1. Extra-Terrestrial Lifeform*
- 2. Extract, Transform, Load*
- 3. Express Toll Lane*
- 4. Eat, Transform, Love*

Q) FME can seamlessly translate between so many formats because it has...

- 1. A sentient data dictionary*
- 2. A retro-encabulator*
- 3. A rich data model*
- 4. A core of unicorn hairs*

FME Desktop

The FME Desktop product is for data translations and transformations at the desktop level (as opposed to FME Server, which is an enterprise-level, web-based product).

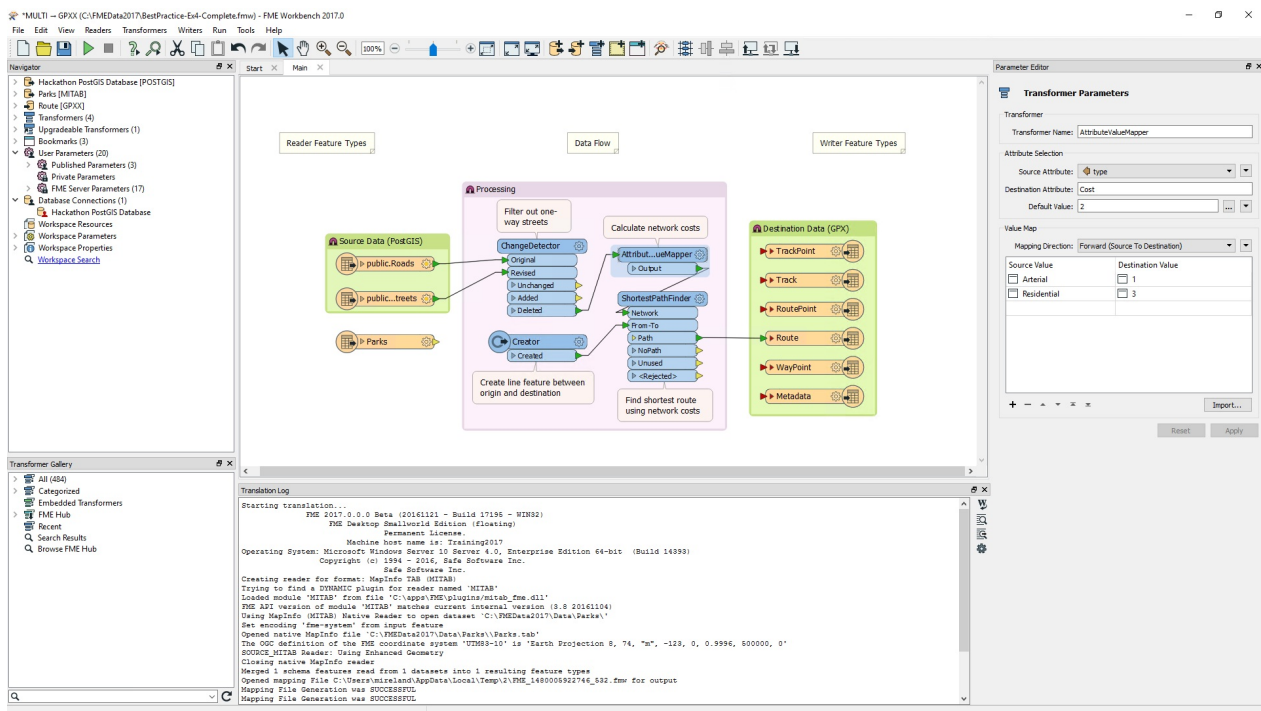
FME Desktop consists of a number of different tools and applications.

FME Desktop Applications

The two key applications within FME Desktop are **FME Workbench** and the **FME Data Inspector**.

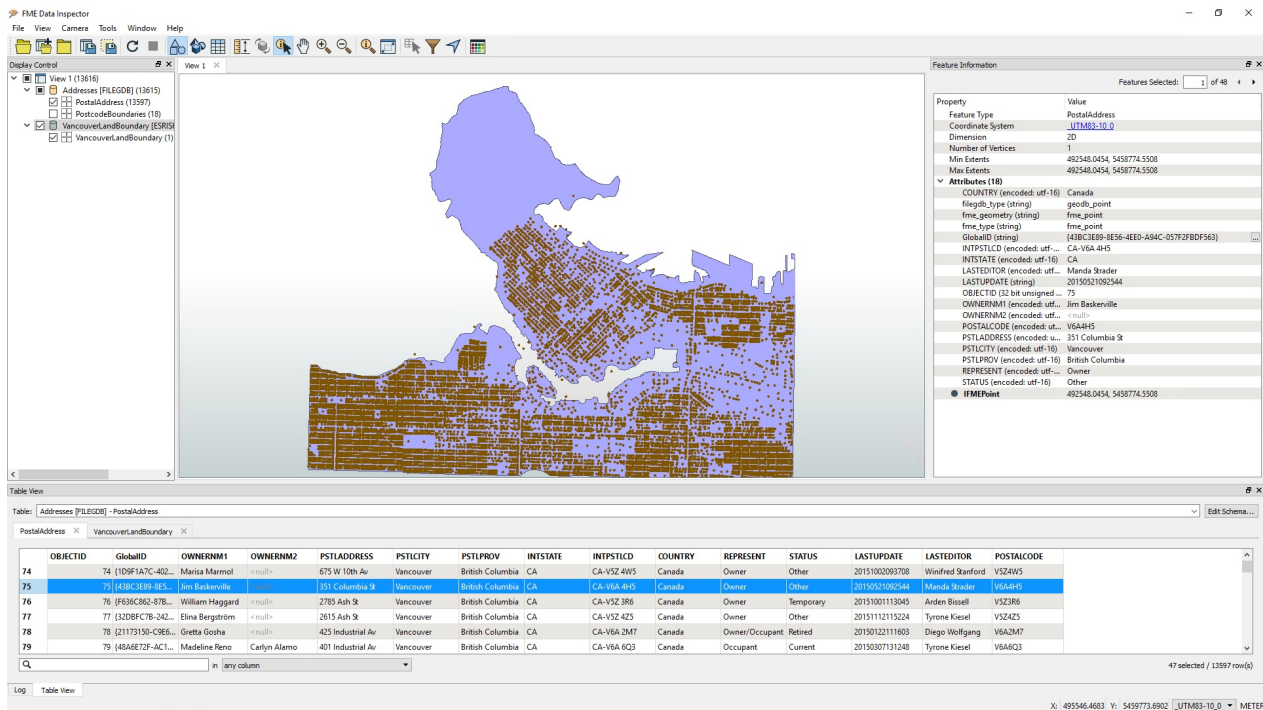
FME Workbench

FME Workbench is the primary tool for defining data translations and data transformations. It has an intuitive point-and-click graphic interface to enable translations to be graphically described as a flow of data.



FME Data Inspector

The FME Data Inspector is a tool for viewing data in any of the FME supported formats. It is used primarily for previewing data before translation or reviewing it after translation.



FME Utilities

Besides Workbench and the Data Inspector, there are several other FME utilities.

FME Help

A tool for browsing through the various help documents for FME.

FME Quick Translator

A precursor to FME Workbench that is used only for quick translations requiring no data transformation.

FME Integration Console

A tool for applying FME functionality to other GIS and CAD applications; commonly enabling use of datasets not normally supported by those applications.

FME Licensing Assistant

An application for managing FME licensing.

Other FME Desktop Components

Additional components are also included as part of FME Desktop (Professional Edition or higher).

FME Command Line Engine

The FME Command Line Engine enables translations to be initiated at the command line level.

FME Plug-In SDK

The FME Plug-In SDK allows developers to add formats and functionality to the FME core.

Introduction to FME Workbench

Let's take a look at what FME Workbench is and does.

What is FME Workbench?

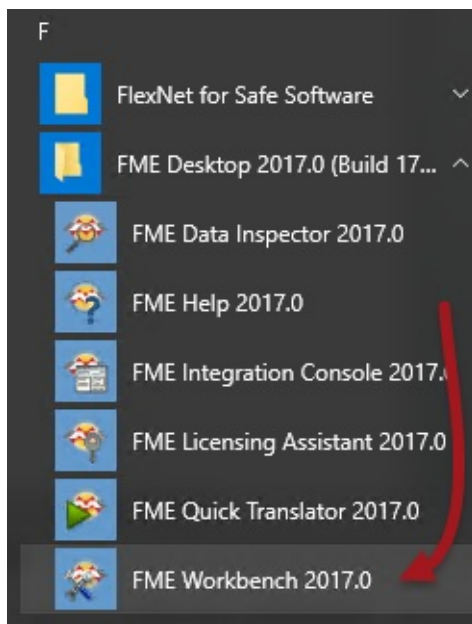
Workbench is FME's tool for authoring data translations and transformations.

It has an intuitive point-and-click interface to define flows of data graphically.

Workbench is fully integrated to interact with other FME Desktop applications such as the FME Data Inspector and other products such as *FME Server* and *FME Cloud*.

Starting FME Workbench

Find FME Workbench in the FME Desktop sub-menu in the Windows start menu. Click on the entry to start Workbench.



Miss Vector says...

As we work through the course the questions will get harder. Still, these are pretty easy:

Which of the following applications is NOT a part of FME Desktop?

- 1. FME Workbench*
- 2. FME Integration Console*
- 3. FME Server Console*
- 4. FME Data Inspector*

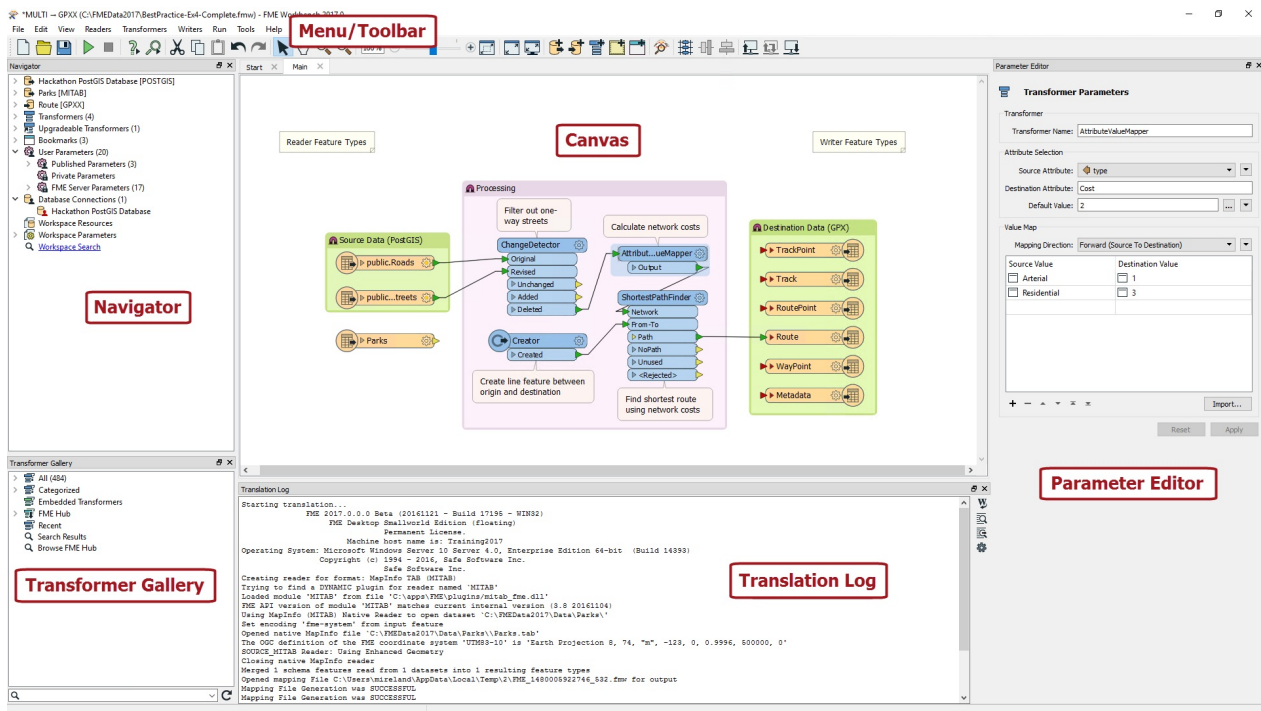
FME Workbench allows you to define flows of data in which way...

- 1. Graphically*
- 2. Telepathically*
- 3. Problematically*
- 4. By writing lots of code in C++ or Java*

Now try starting FME Workbench as described above.

Major Components of FME Workbench

The FME Workbench user interface has a number of major components:



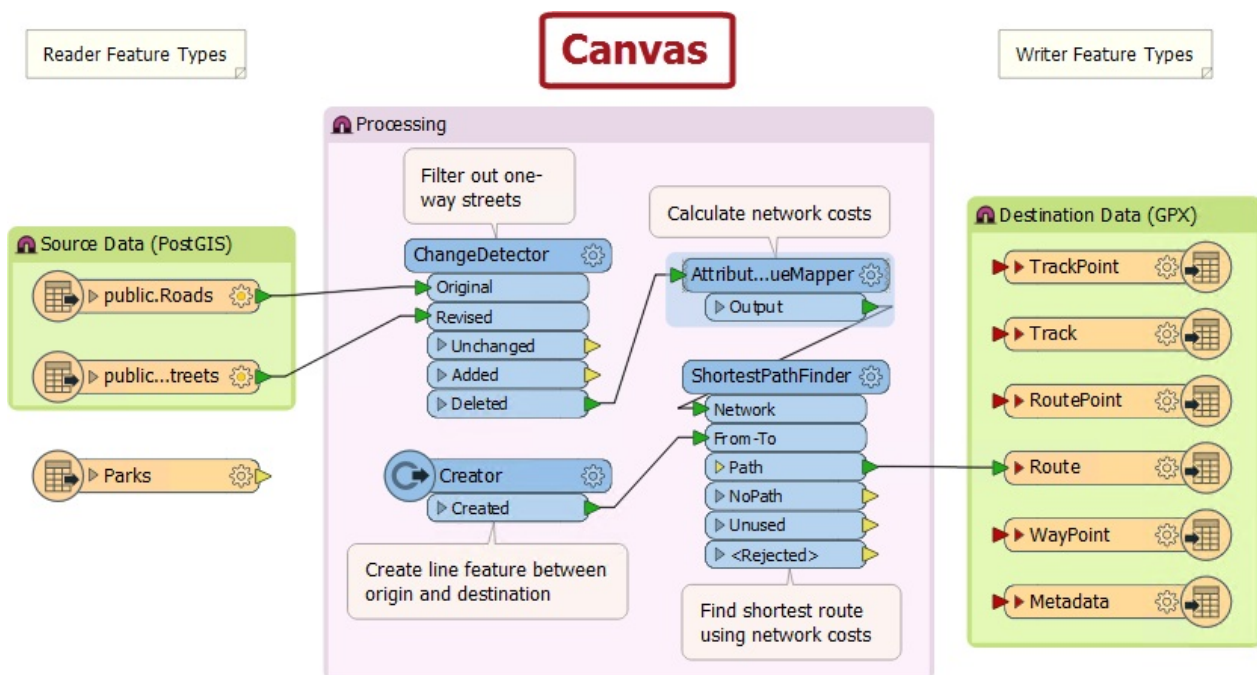
Menu/Toolbar

The menubar and toolbar contain a number of tools: for example, tools for navigating around the Workbench canvas, controlling administrative tasks, and adding or removing reader (source) datasets.



Canvas

The FME Workbench canvas is where users graphically define a translation. This definition is called a "workspace" and can be saved for re-use later.



By default the workspace reads from left to right; data source on the left, transformation tools in the center, and data destination on the right. Connections between each item represent the flow of data and may branch in different directions, merge together, or any combination of the two.

The canvas is the primary window within Workbench and the focus of all your work.

Police-Chief Webb-Mapp says...

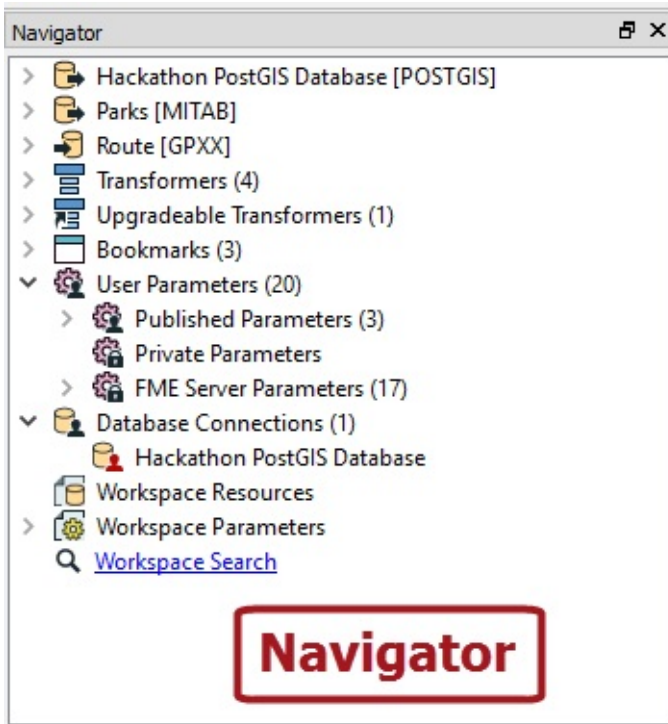
I'm the police chief, responsible for tracking down crimes against FME.

So, let's make sure you get the terminology right. The application itself is called FME "Workbench", but the process defined in the canvas window is called a "Workspace". The terms are so similar that they are easily confused, but please don't, otherwise I will have to send my grammar squad to arrest you!

Although mistreating FME terminology is a minor offence the ignominy of being caught is long lasting!

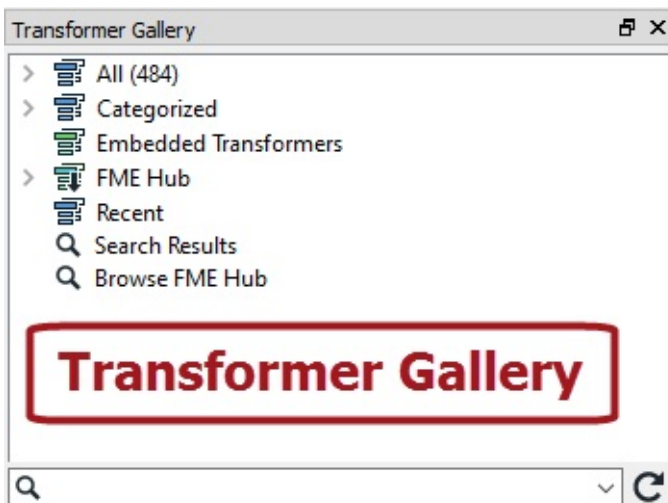
Navigator

The Navigator window is a structured list of parameters that represent and control all of the components on your workspace canvas.



Transformer Gallery

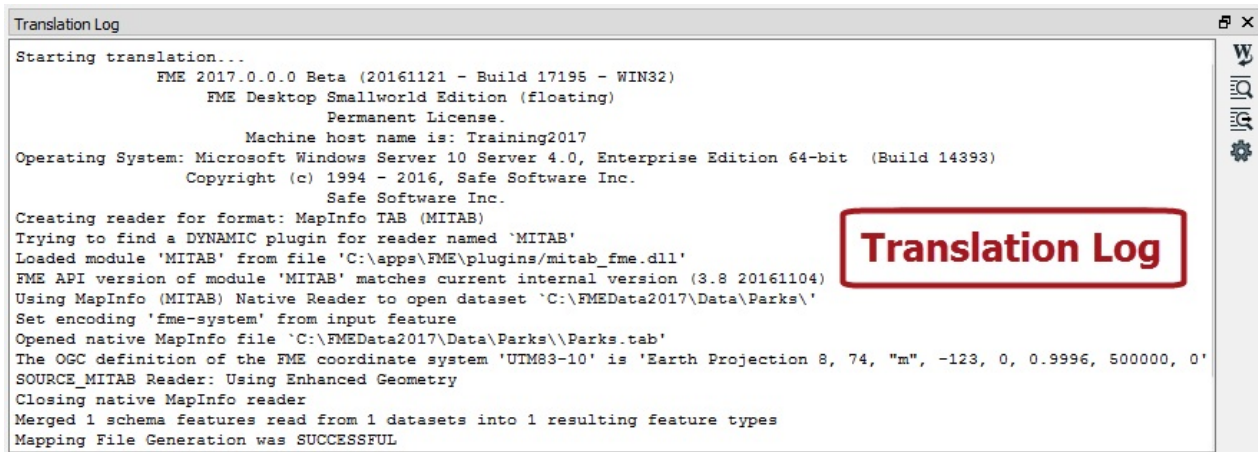
The transformer gallery is a tool for the location and selection of FME transformation tools.



The number of transformers (above, 484) will vary depending on the version of FME and any optional custom transformers installed.

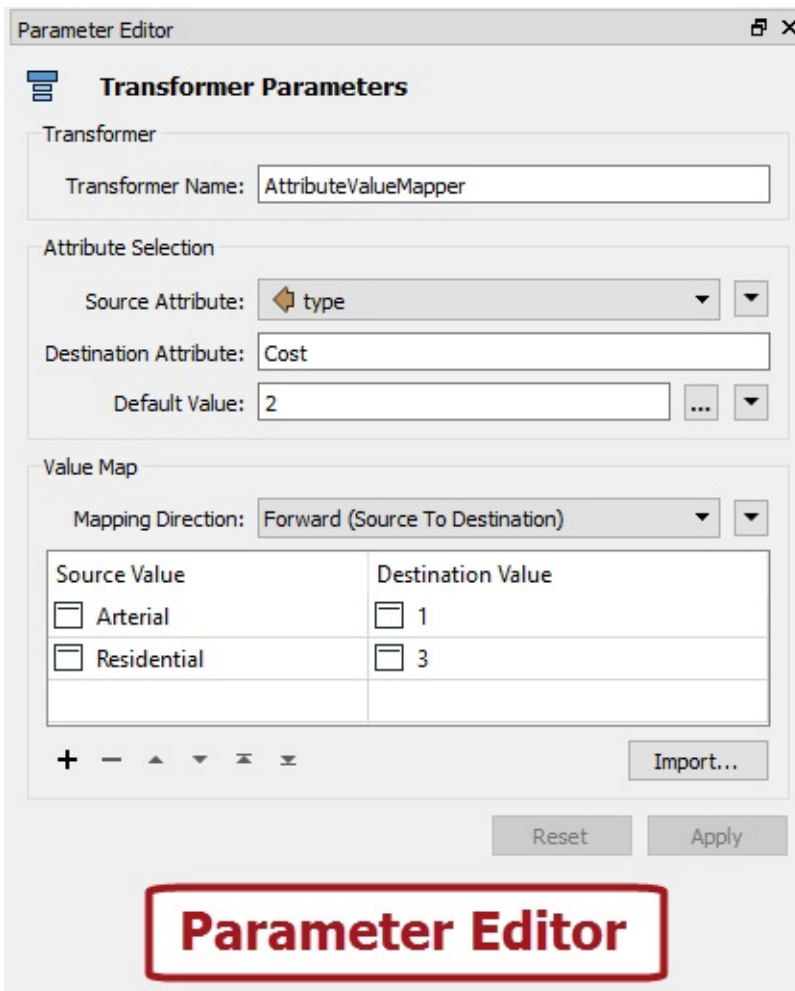
Translation Log

The translation log reports on translations and other actions. Information includes any warning or error messages, translation status, length of translation, and number of features processed.



Parameter Editor

The parameter editor window is a dialog in which transformation parameters can be entered and adjusted.



NEW

New for FME2017, the Parameter Editor is a single dialog is intended to replace having a dialog for each object on the canvas. It also replaces the Workspace Properties window.

Other Windows

Help

The Help window displays information from the FME documentation. It automatically refreshes the content depending on what aspect of a workspace is being authored.

History

The History window displays the workspace editing history. It is a form of tree-based undo/redo that allows a user to revisit previous revisions in a workspace.

Overview

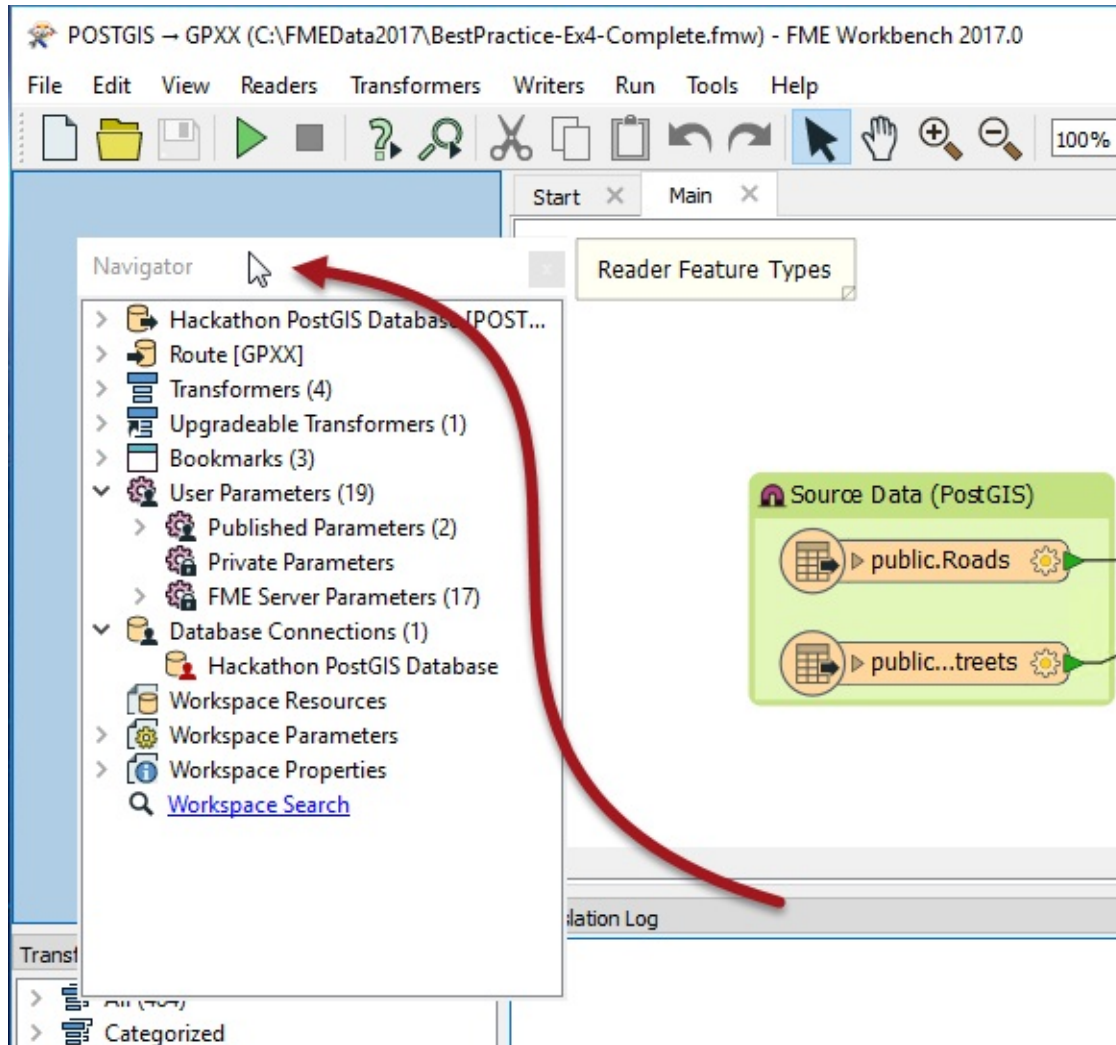
This window shows a small scale representation of the workspace canvas. It is useful as an aid to navigating through the canvas.

Feature Type and Attribute Connections

These windows help to connect objects on the Canvas. It's part of a process called Schema Mapping that you'll find out about later...

Window Control

All windows in Workbench can be detached from their default position and deposited in a custom location. To do this simply click on the frame of the window and drag it into a new position.



If a window is dropped on top of an existing window, then the two will become tabbed.

If a window is dropped beside an existing window (or between two existing windows), then they will become stacked.

In the above screenshot the user is dropping the Navigator into position on the left-hand side of Workbench, stacked on top of the Transformer Gallery.

FireFighter Mapp says...

Hi. I'm your friendly neighborhood firefighter, with a hot tip for you. Don't feel put out by a lack of canvas space. Press F11 to instantly dispatch lesser-used windows to one side and expand the canvas window to an alarming size!

NEW

New for FME2017, pressing F11 will NOT hide windows that are floating. That way you can keep open the Parameter Editor and Help Window (for example) while maximizing the canvas to its full size on a second monitor.

Miss Vector says...

Switch over to FME Workbench and experiment with the windows (look under the View menu) to answer these questions.

Which of these is a window in FME Workbench?

- 1. The Maths Window*
- 2. The Geography Window*
- 3. The English Literature Window*
- 4. The History Window*

Which of these is NOT an arrangement of Windows in FME Workbench?

- 1. Stacked*
- 2. Floating*
- 3. Double-Glazed*
- 4. Tabbed*

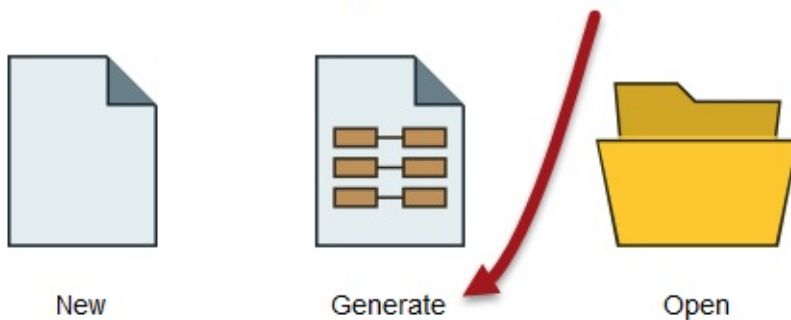
Creating a Translation

Workbench's intuitive interface makes it easy to set up and run a simple format-to-format ('quick') translation.

The Start Tab

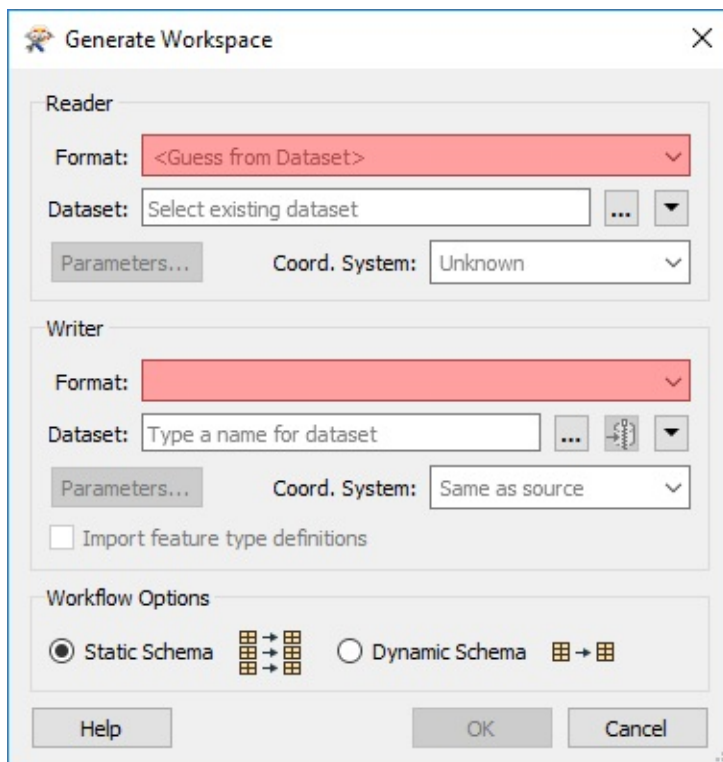
The Start Tab in FME Workbench includes a section called "Getting Started." This section has links to the different ways in which a workspace can be created. The simplest method is Generate Workspace:

Create Workspace



Generate Workspace Dialog

The Generate Workspace dialog condenses all the choices to be made into a single dialog box. It has fields for defining the format and location of both the data to be read, and the data to be written.

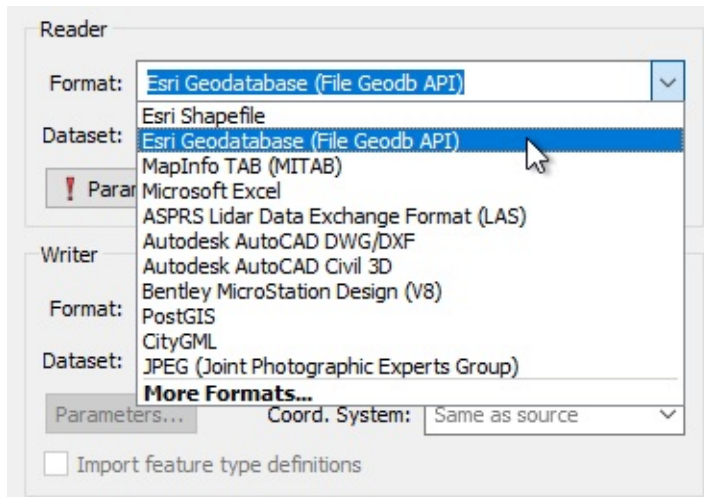


Format Selection

All format selection fields in FME are both a pull-down menu and a text entry field.

The text entry field allows you to type a format name directly. It has an 'intelli-complete' function that selects close matches as you type.

The drop-down list shows the last ten formats used, so favourite formats are instantly available:



Format selection can also be made from a table showing ALL of the formats supported by FME. To access this table, select 'More Formats...' from the foot of the drop-down formats list.

Dataset selection fields are a text entry field, but with a browse button to open an explorer-like dialog.

Red coloring in an FME dialog indicates mandatory fields. Users must enter data in these fields to continue. In most dialogs the OK button is de-activated until the mandatory fields are complete.

Miss Vector says...

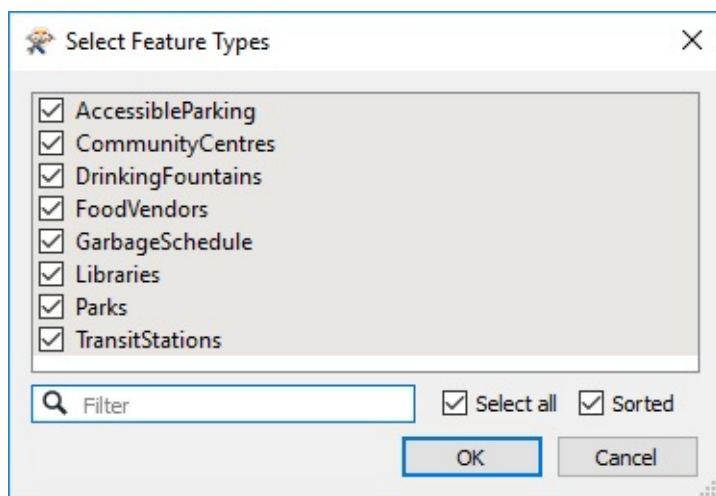
Here's a question you can't answer with 'a', 'b', 'c', or 'd'! In the Generate Workspace dialog, why might it be useful to set the data format before browsing for the source data?

Try browsing for a dataset before setting the format type and see if you can detect the difference. [Did you get it yet?](#)

Feature Types Dialog

Clicking OK on the Generate Workspace dialog causes FME to generate the defined workspace. However, whenever a source dataset contains multiple layers the user is first prompted to select which are to be translated.

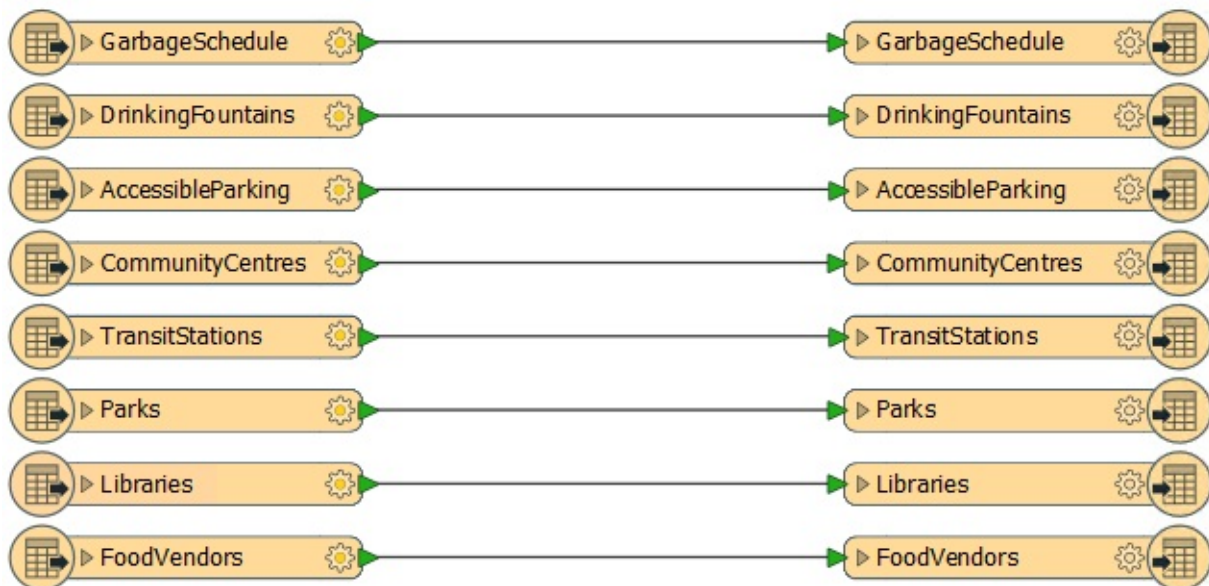
This is achieved through the Select Feature Types dialog. In FME *Feature Type* is another term for *layer*. Only selected layers show in the workspace.



Here, for example, is a Select Feature Types dialog where the user has chosen to include all available layers in the workspace.

The New Workspace

A new workspace reads from left to right, from the source (Reader) layers, through a dataflow, to the destination (Writer) layers. Arrows denote the direction of data flow, from source to destination.



In the above screenshot eight layers of data are being read from one format and written to another.

TIP

In most cases FME uses the terms 'Reader' and 'Writer' instead of 'Source' and 'Destination.' A later chapter explains why. For now, just be aware that a reader reads datasets and a writer writes datasets, and these terms are analogous to source/destination and input/output.

Saving the Workspace

Workspaces can be saved to a file so that they can be reused at a later date. The save button on the toolbar is one way to do this:



There are also menu options to do the same thing, in this case File > Save (shortcut = Ctrl+S) or File > Save As.

The default file extension is .fmw. Double-clicking a *.fmw file in Explorer starts FME Workbench and opens up the workspace.

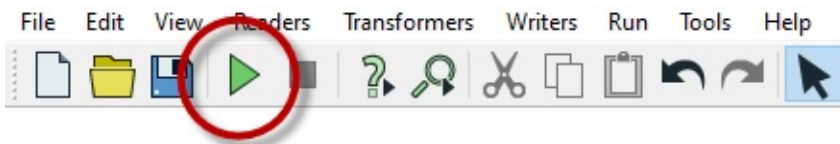
Sister Intuitive says...

Greetings. I am Sister Maria Intuitive of the Order of the Perpetual Translation. I'm here to demonstrate some of the intuitive, but less obvious, features of the FME Workbench interface.

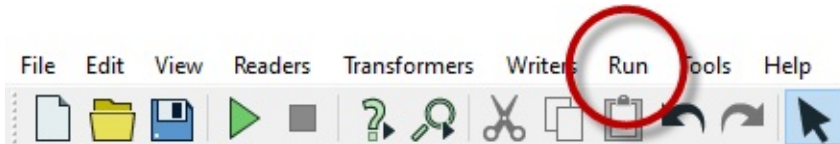
For example, select File > Open Recent on the menubar to reveal a list of previously used workspaces. This list can show up to a huge 15 entries.

Running a Workspace

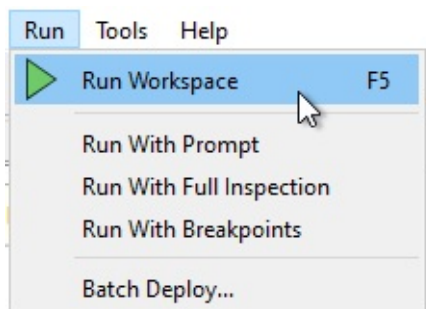
The green arrow (or 'play' button) on the Workbench toolbar starts a translation:



Alternatively, look under Run on the menubar:



The same toolbar options appear on both the menubar and toolbar. Notice the shortcut option F5 can be used instead:



TIP

The action of the Run button is modified by a series of options. These options include the ability to prompt the user for input (Run with Prompt), the ability to inspect the data (Run with Full Inspection) and the ability to run in debug mode (Run with Breakpoints). These are toggle options that can be turned on and off at will. By default Run with Breakpoints does not appear on the toolbar, only the menubar.

Workspace Results

After running a workspace, related information and statistics are found in the translation log, which is displayed in the Workbench log window.

The translation log reveals whether the translation succeeded or failed, how many features were read from the source and written to the destination, and how long it took to perform the translation.

```
=====
                        Features Read Summary
=====
AccessibleParking                      42
CommunityCentres                      10
DrinkingFountains                    113
FoodVendors                          91
GarbageSchedule                       6
Libraries                             8
Parks                                69
TransitStations                      11
=====
Total Features Read                    350
=====
                        Features Written Summary
=====
AccessibleParking                      42
CommunityCentres                      10
DrinkingFountains                    113
FoodVendors                          91
GarbageSchedule                       6
Libraries                             8
Parks                                69
TransitStations                      11
=====
Total Features Written                 350
=====
Translation was SUCCESSFUL with 0 warning(s) (8 feature(s) output)
FME Session Duration: 2.1 seconds. (CPU: 1.1s user, 0.5s system)
END - ProcessID: 4084, peak process memory usage: 131024 kB
Translation was SUCCESSFUL
```

In this example the log file reveals that 350 features were read (from an Esri Geodatabase) and written out (to a GML dataset).

The overall process was a success, with zero warnings. The elapsed time for the translation was 2.1 seconds.

Miss Vector says...

I bet you've got all of the questions correct so far! Well done. Now see if you can get these:

Which of these is NOT a way to set the format of a translation?

- 1. Typing the format name*
- 2. Selecting the format from a drop-down list*
- 3. Browsing for the format in the formats gallery*
- 4. By selecting a dataset with a known file extension*
- 5. None of the above (they are all valid ways to set the format)*

Which key is a shortcut to run a workspace?

- 1. F4*
- 2. F5*
- 3. F5.6*
- 4. F#*

Exercise 1 Basic Workspace Creation	
Data	Zoning Data (MapInfo TAB)
Overall Goal	Create a workspace to translate zoning data in MapInfo TAB format to AutoCAD DWG
Demonstrates	Basic workspace creation with FME Workbench
Start Workspace	None
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Basics-Ex1-Complete.fmw

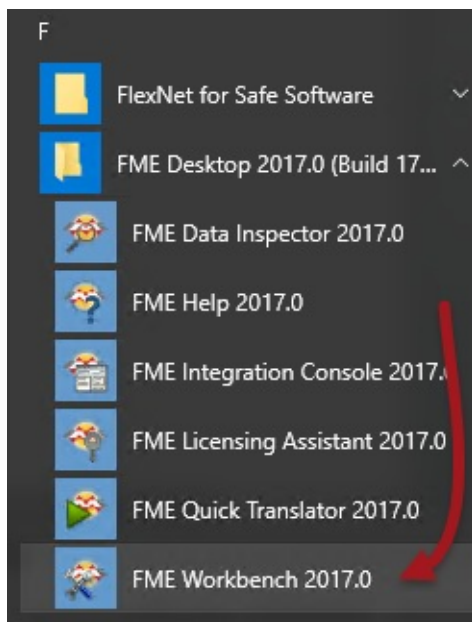
Congratulations! You have just landed a job as technical analyst in the GIS department of your local city. Your old schoolteacher, Miss Vector, gave you a reference, so don't let her down!

On your first day you've been asked to do a simple file format translation.

We've outlined all of the actions you need to take; though FME's interface is so intuitive you should be able to carry out the exercise without the need for these step-by-step instructions.

1) Start FME Workbench

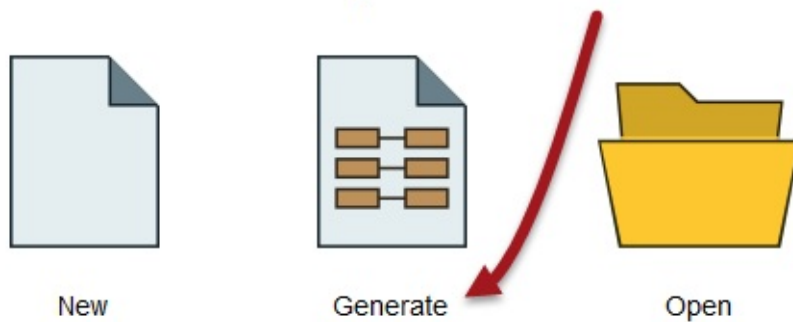
Start FME Workbench by selecting it from the Windows start menu. You'll find it under Start > All Programs > FME Desktop 2017.0 > FME Workbench 2017.0.



2) Select Generate Workspace

FME Workbench will start up and begin with the Start tab active. In the Create Workspace part of the Start tab, select the option to Generate (Workspace). Alternatively you can use the shortcut Ctrl+G.

Create Workspace



3) Define Translation

The Generate Workspace tool opens up a dialog in which to define the translation to be carried out. Fill in the fields in this dialog as follows:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEDData2017\Data\Zoning\Zones.tab
Writer Format	Autodesk AutoCAD DWG/DXF
Writer Dataset	C:\FMEDData2017\Output\Training\Zones.dwg

The dialog will look like this:

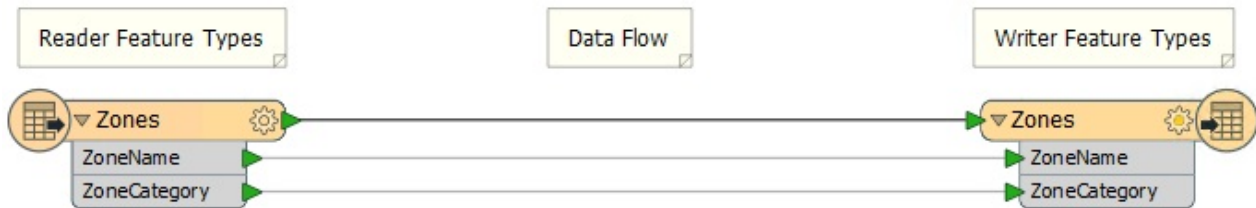
The screenshot shows the 'Generate Workspace' dialog box. It contains the following fields and options:

- Reader Section:**
 - Format: MapInfo TAB (MITAB)
 - Dataset: C:\FMEDData2017\Data\Zoning\Zones.tab
 - Coord. System: Read from source
- Writer Section:**
 - Format: Autodesk AutoCAD DWG/DXF
 - Dataset: FMEDData2017\Output\Training\Zones.dwg
 - Coord. System: Same as source
 - ☐ Import feature type definitions
- Workflow Options:**
 - ☒ Static Schema
 - ☐ Dynamic Schema
- Buttons:** Help, OK, Cancel

Remember, you can set a format by typing its name, by selecting it from the drop- down list, or by choosing "More Formats" and selecting the format from the full table of formats. For now, ignore the Workflow Options and leave the default of 'Static Schema.'

4) Generate and Examine Workspace

Click OK to close the Generate Workspace dialog. A new workspace will be generated into the FME Workbench canvas:



The list of attributes is exposed by clicking the arrow icon on each object.

5) Run Workspace

Run the workspace by clicking the run button on the toolbar, or by using Run > Run Translation on the menubar. The translation will run and the log file reports something like this:

```

=====
                        Features Read Summary
=====
Zones                                     416
=====
Total Features Read                       416
=====
                        Features Written Summary
=====
Zones                                     416
=====
Total Features Written                     416
=====
Closing native MapInfo reader
Translation was SUCCESSFUL with 0 warning(s) (416 feature(s) output)
FME Session Duration: 2.3 seconds. (CPU: 1.2s user, 0.2s system)

```

6) Locate Output

Locate the destination data in Windows Explorer to prove that it's been written as expected. In the next section we'll cover how to inspect the dataset to ensure that it is correct.

This PC > OS (C:) > FMEDData2017 > Output > Training				
Name	Date modified	Type	Size	
Zones.dwg	12/8/2016 12:24 PM	DWG File	186 KB	

TIP

When a translation is run immediately in Workbench or Quick Translator, without further adjustment, it's known as a 'Quick Translation.'

Because FME is a 'semantic' translator, with an enhanced data model, the output from a quick translation is as close to the source data in structure and meaning as possible, given the capabilities of the destination format.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Create an FME workspace*
- *Run an FME workspace*

Introduction to Data Inspection

To ensure that you're dealing with the right information you need a clear view of your data at every stage of the transformation process.

Data Inspection meets this need: it is the act of viewing data for verification and debugging purposes, before, during, or after a translation.

What Can Be Inspected?

A number of different aspects of data may be inspected, including the following:

- **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 correct?
 - **Symbology:** Is the color, 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?
-

Chef Bimm says...

Hi. I'm Chef Bimm and I'm here to help you cook up some tasty data translations.

I have a great recipe for loading CAD files into a Building Information Model. Inspecting the ingredients... I mean data... before I use them lets me detect problems before they affect the translation.

Features in the wrong source layer could need the whole process to be repeated. Data Inspection saves me that hassle. Now I know that if the "sauce" is fine, the final result will be too!

Introduction to the FME Data Inspector

The role of inspecting data in FME is not carried out in FME Workbench, but in a complementary application: the FME Data Inspector.

What is the FME Data Inspector?

The FME Data Inspector is a utility that allows viewing of data in any of the FME supported formats. It is used primarily to preview data before translation or to verify it after translation. The FME Data Inspector is closely tied to FME Workbench so that Workbench can send data directly to the Inspector.

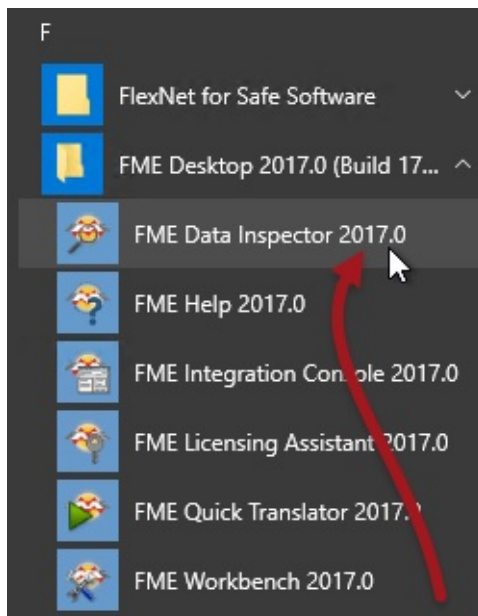
The FME Data Inspector can also be used to check data at any point *during* a translation; as you use FME you'll find this is useful for step-by-step examination of complex translations.

What the FME Data Inspector Is Not!

The FME Data Inspector isn't designed to be a form of GIS or mapping application. It has no all-around analysis functionality, and the tools for symbology modification and printing are rudimentary and intended for data validation rather than producing map output.

Starting the FME Data Inspector

Find FME Data Inspector in the FME Desktop sub-menu in the Windows start menu. Click on the entry to start the Data Inspector.



Miss Vector says...

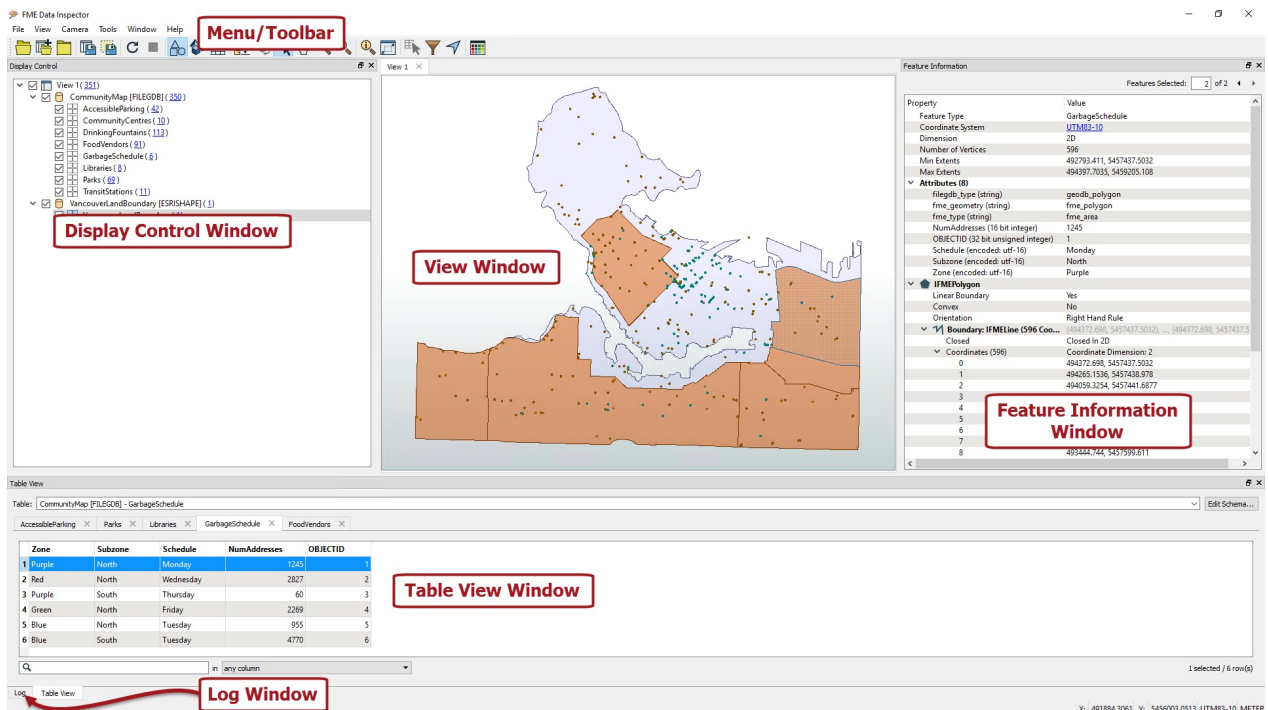
FME is so easy to use, it's hard to think up some difficult questions! But I'll try.

*When you are inspecting *schema*, what are you trying to verify?*

- 1. The color and linestyle of features*
- 2. The number of features*
- 3. The layers (classes, tables, types)*
- 4. Where the nearest coffee shop is*

Major Components of the FME Data Inspector

When the FME Data Inspector is started, and a dataset is opened, it looks something like this:



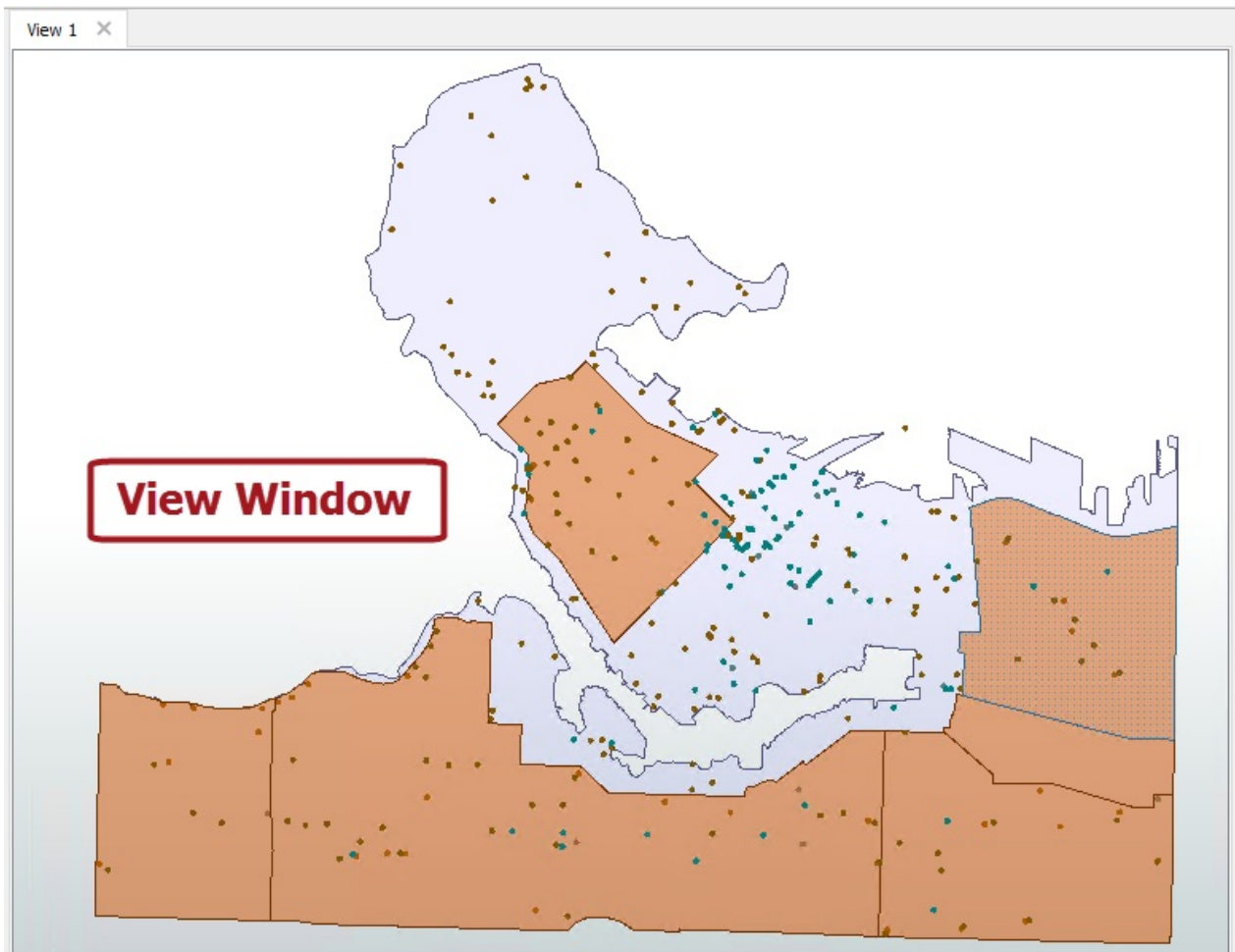
Menu bar and Toolbar

The menu bar and toolbar contain a number of tools. Some are for navigating around the View window, some control administrative tasks such as opening or saving a dataset, and others are for special functionality such as selective filtering of data or the creation of dynamic attributes.



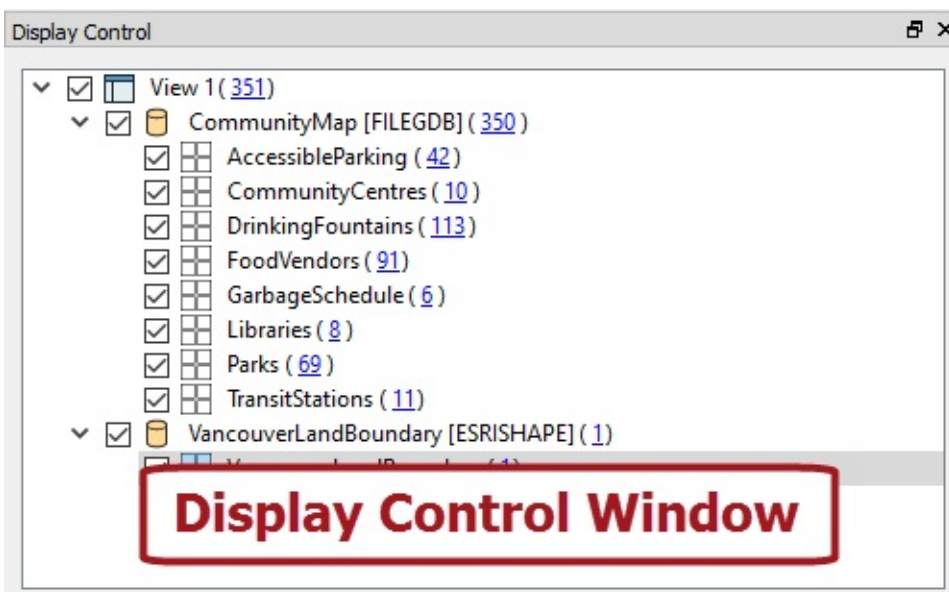
View Window

The View window is the spatial display area of the FME Data Inspector. Multiple views of different datasets may be opened at any one time.



Display Control Window

The Display Control window shows a list of the open datasets and their feature types. Tools here let users turn these on or off in the display, alter their symbology, and adjust the display order.



Feature Information Window

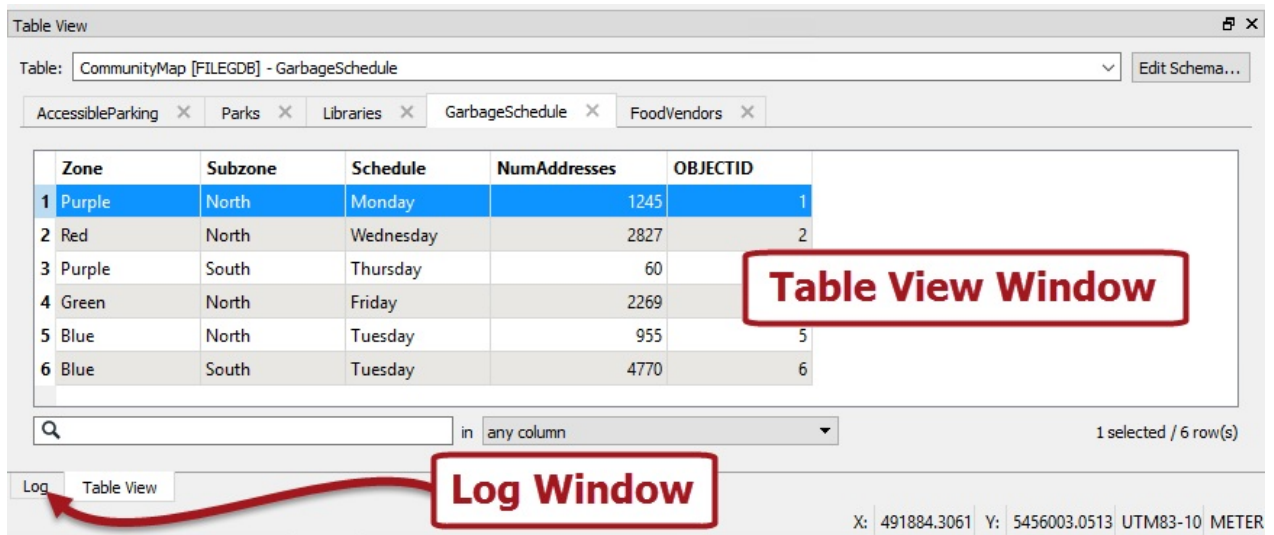
When users query a feature in the View window, information about that feature is shown in the Information window. This information includes the feature's feature type, attributes (both user and format attributes), coordinate system and details about its geometry.

Feature Information	
Features Selected: 2 of 2	
Property	Value
Feature Type	GarbageSchedule
Coordinate System	UTM83-10
Dimension	2D
Number of Vertices	596
Min Extents	492793.411, 5457437.5032
Max Extents	494397.7035, 5459205.108
▼ Attributes (8)	
fileddb_type (string)	geodb_polygon
fme_geometry (string)	fme_polygon
fme_type (string)	fme_area
NumAddresses (16 bit integer)	1245
OBJECTID (32 bit unsigned integer)	1
Schedule (encoded: utf-16)	Monday
Subzone (encoded: utf-16)	North
Zone (encoded: utf-16)	Purple
▼ IFMEPolygon	
Linear Boundary	Yes
Convex	No
Orientation	Right Hand Rule
▼ Boundary: IFMELine (596 Coor...	(494372.698, 5457437.5032), ..., (494372.698, 5457437.5
Closed	Closed In 2D
▼ Coordinates (596)	Coordinate Dimension: 2
0	494372.698, 5457437.5032
1	494265.1536, 5457438.978
2	494059.3254, 5457441.6877
3	
4	
5	
6	
7	
8	493444.744, 5457599.611

Feature Information Window

Table View Window

The Table View window is a spreadsheet-like view of a dataset and includes all of the features and all of the attributes, with a separate tab for each feature type (layer).



Other Windows

Log Window

The Log window (tabbed underneath the Table View in above screenshot) reports information relating to the reading and look of a dataset that can be used to confirm whether data has been read correctly. Some functions on the toolbar also generate messages in the Log window.

Using the FME Data Inspector

With the FME Data Inspector it's easy to open and view any number of datasets and to query features within them.

Viewing Data

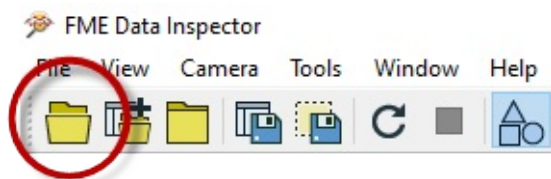
The FME Data Inspector provides two methods for viewing data: opening or adding.

Opening a dataset opens a new view window for it to be displayed in. **Adding** a dataset displays the data in the existing view window; this way multiple datasets can be viewed simultaneously.

Opening a Dataset

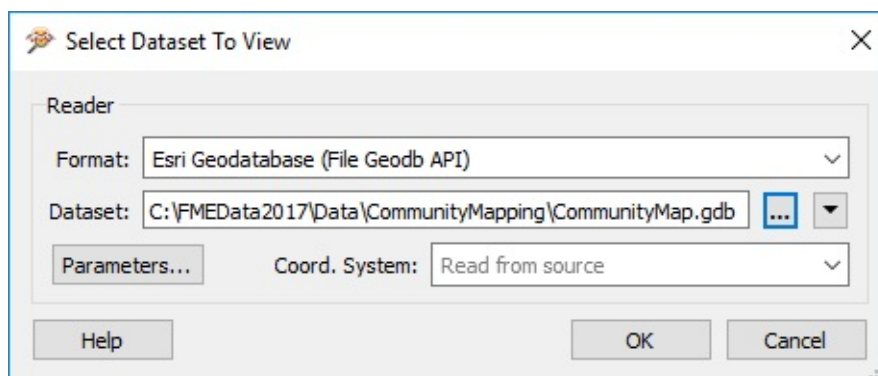
Datasets can be opened in the FME Data Inspector in a number of ways.

- Select File > Open Dataset from the menu bar
- Select the toolbar button Open Dataset.
- Drag and drop a file onto any window (except the View window)
- Open from within Workbench



Opening data from within FME Workbench is achieved by simply right-clicking on a canvas feature type (either source or destination) and choosing the option 'Inspect'.

All of these methods cause a dialog to open in the FME Data Inspector in which to define the dataset to view.

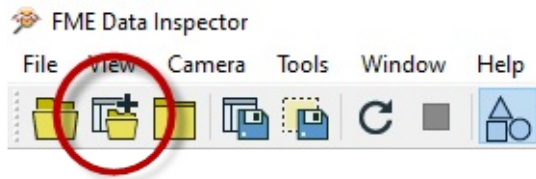


Adding a Dataset

Opening a dataset causes a new View tab to be created and the data displayed. To open a dataset within an existing view tab requires use of tools to add a dataset.

- Selecting File > Add Dataset from the menu bar
- Selecting the toolbar button Add Dataset

- Dragging and Dropping a file onto the view window



Windowing Tools

Once data has been opened in the FME Data Inspector, there are a number of tools available for altering the view.

- Pan
- Zoom In
- Zoom Out
- Zoom to a selected feature
- Zoom to the full extent of the data



TIP

Press the Shift key on the keyboard and it will activate the zoom-in tool in the Inspector. Press the Ctrl key and it will activate the zoom-out tool. Release the key to revert to the previous tool.

This functionality allows users to quickly move between query and navigation modes at the press of a key, so there's no need to click between query and navigation tools on the menubar or toolbar.

Inspecting Data

THE FME Data Inspector includes several querying tools, but in particular:

- Query individual feature(s)
- Measure a distance within a View window



The query tool button is like a toggle. By default it is active when you start the FME Data Inspector; if you click it again - or select a windowing tool - you turn the query tool off.

The results of a feature query are shown in the Feature Information window.

Feature Information Window

The upper part reports on general information about the feature; which feature type (layer) it belongs to, which coordinate system it is in, whether it is two- or three-dimensional, and how many vertices it possesses.

Feature Information	
Features Selected: 2 of 23	
Property	Value
Feature Type	Libraries
Coordinate System	UTM83-10
Dimension	2D
Number of Vertices	1
Min Extents	491595.3554, 5458555.6753
Max Extents	491595.3554, 5458555.6753
▼ Attributes (7)	
filegdb_type (string)	geodb_point
fme_geometry (string)	fme_point
fme_type (string)	fme_point
LibraryAddress (encoded: utf-16)	350 W Georgia St
LibraryName (encoded: utf-16)	Central Branch
LibraryURL (encoded: utf-16)	http://www.vpl.ca/branches/details/central_library ...
OBJECTID (32 bit unsigned integer)	3
● IFMEPoint	491595.3554, 5458555.6753

The middle part reports the attributes associated with the feature. This includes user attributes and format attributes (for example fme_type).

The lower part reports the geometry of the feature. It includes the geometry type and a list of the coordinates that go to make up the feature.

Table View Window

Also available is a window called the Table View.

Table View

Table: CommunityMap [FILEGDB] - GarbageSchedule Edit Schema...

AccessibleParking × Parks × Libraries × GarbageSchedule × FoodVendors ×

	Zone	Subzone	Schedule	NumAddresses	OBJECTID
1	Purple	North	Monday	1245	1
2	Red	North	Wednesday	2827	2
3	Purple	South	Thursday	60	3
4	Green	North	Friday	2269	4
5	Blue	North	Tuesday	955	5
6	Blue	South	Tuesday	4770	6

in any column 6 row(s)

The table view is a way to inspect data in a tabular, spreadsheet-like, layout. Although it does not have the same depth of information shown by the Information Window, the Table View is particularly useful for inspecting the attribute values of multiple features simultaneously.

NEW

In FME2017 the Table View Window changed slightly. In previous versions, all tables were displayed automatically. Now, to improve performance, tables are only displayed when selected from the drop-down list, or when queried in the current view window.

Miss Vector says...

Start the FME Data Inspector and open a dataset.

In the Table View window right-click on records and column-headers to view the context menus. Which of the following is NOT an available menu option(s):

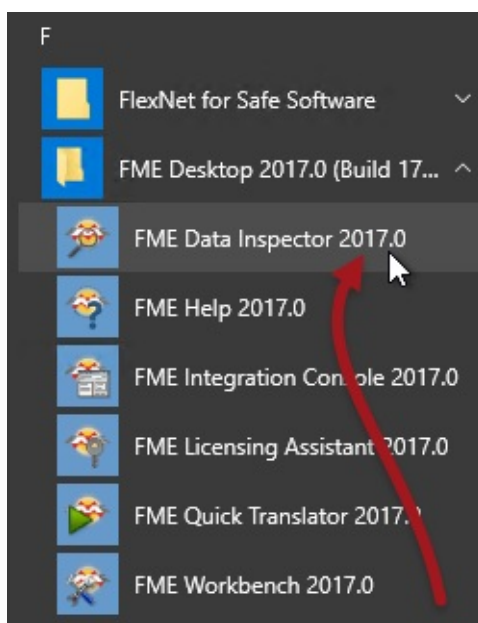
1. Sort (Alphabetical or Numeric)
2. Inspect Value
3. Cut/Copy/Paste
4. Save Selected Data As

Exercise 2 Basic Data Inspection	
Data	Zoning Data (MapInfo TAB)
Overall Goal	Inspect the output from a previous translation
Demonstrates	Basic data inspection with the FME Data Inspector
Start Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Basics-Ex1-Complete.fmw
End Workspace	None

In the previous exercise you were asked to convert some data between formats. Before you send the converted data out you should really inspect it to make sure it is correct. If you haven't been trying it already, let's see how the FME Data Inspector interface works by inspecting the output from that quick translation.

1) Start FME Data Inspector

Start the FME Data Inspector by selecting it from the Windows start menu. You'll find it under Start > All Programs > FME Desktop 2016.0 > FME Data Inspector 2016.0.



2) Select Dataset

The FME Data Inspector will start up and begin with an empty view display.

To open a dataset, select File > Open Dataset from the menubar. When prompted, fill in the fields in the Select Dataset dialog as follows:

Reader Format	Autodesk AutoCAD DWG/DXF
Reader Dataset	C:\FMEData2017\Output\Training\Zones.dwg

NB: If you can't find the dataset - maybe you didn't complete the first exercise, or wrote the data to a different location - then you can open the original zoning dataset as:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2017\Data\Zoning\Zones.tab

3) Open Dataset

The dataset should look something like this:



4) Browse Data

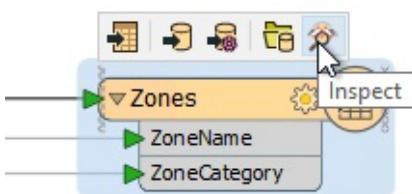
Use the windowing tools on the toolbar to browse through the dataset, inspecting it closely. Use the Query tool to query individual features and inspect the information in the Feature Information Window.

Try right-clicking in the different Data Inspector windows, to discover functionality that exists on context menus.

5) Open Data from Within Workbench

Back in FME Workbench, open the workspace from the previous exercise.

Click on any Reader or Writer feature type and a small pop-up menu has an option to 'Inspect'. This alternative method of opening a dataset in the Data Inspector is very convenient when you are already using FME Workbench.



TIP

Similar to the above, you can locate the output dataset by clicking the pop-up option to Open Containing Folder.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Open datasets in a new view in the FME Data Inspector*
- *Use the windowing and inspection tools in the FME Data Inspector*
- *Use FME Workbench functionality to open a dataset in the FME Data Inspector*

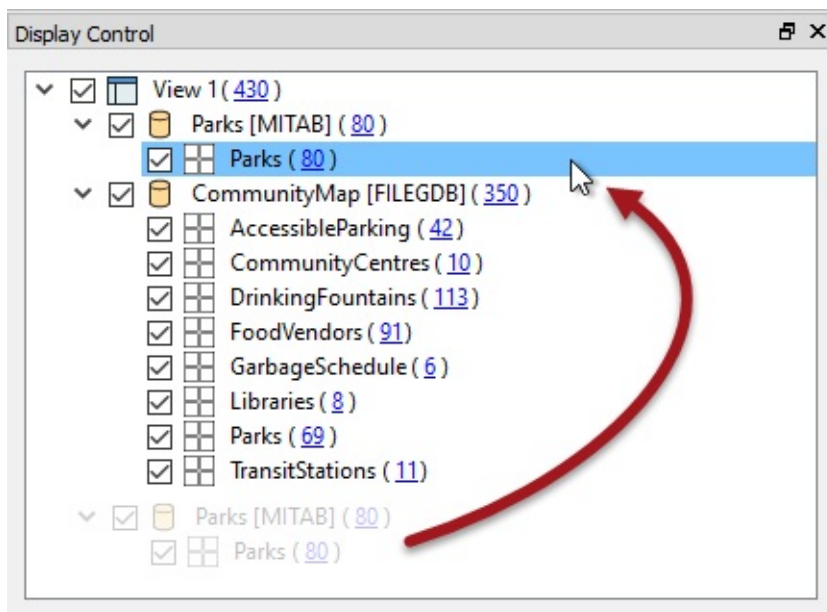
Data Inspector Display Control

The FME Data Inspector has a number of controls to assist in showing the data in an orderly manner.

Display Control Window

Management of feature display order is carried out in the Display Control window.

Each dataset and feature type can be dragged above any other to promote its display order in the View window.

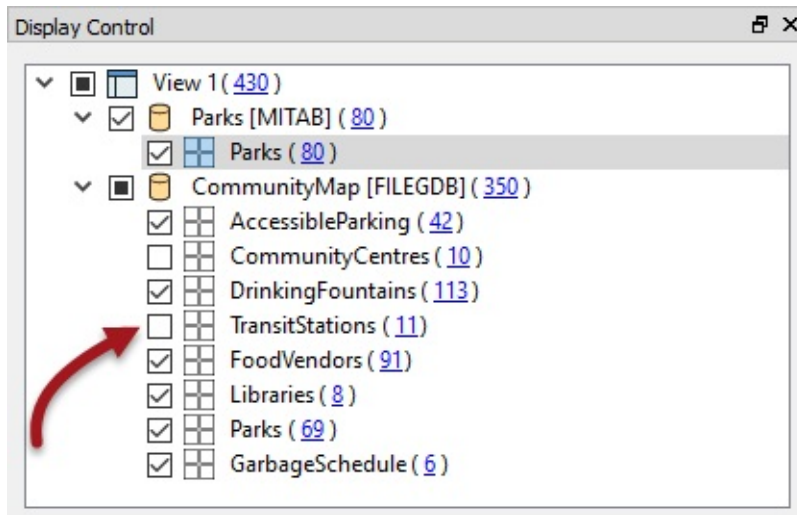


Feature Types can only be ordered within their container dataset.

For example, "Libraries" can be promoted above "GarbageSchedule" in the display, but they cannot by themselves be promoted above the MapInfo Parks dataset.

Display Status

Each level of the Display Control window has a checkbox to turn data on and off at that level.



You can choose to turn off individual layers or an entire dataset at once.

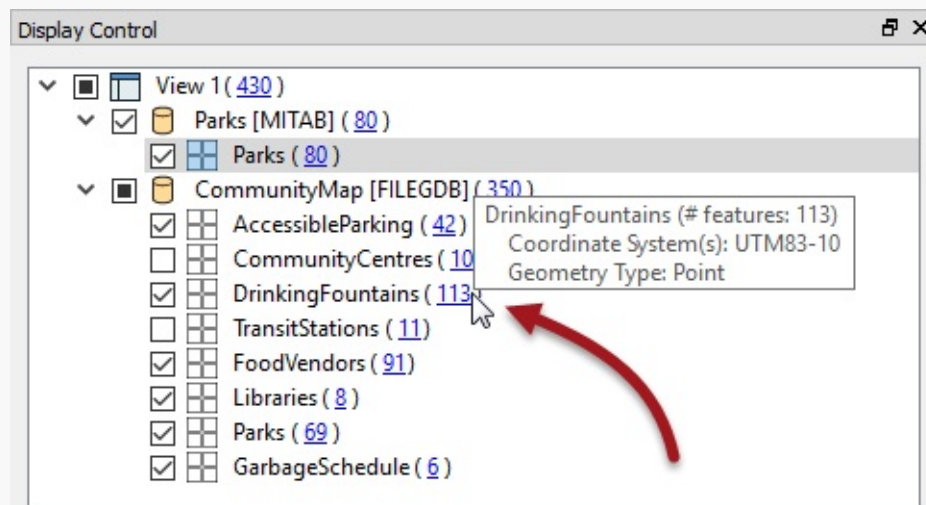
Feature Counts

Each Feature Type in the Display Control window is tagged with the number of features it contains, in parentheses after the feature type name. So in the previous image we can see that there are 113 drinking fountains (or 113 features called DrinkingFountains) in the city of Vancouver.

NEW

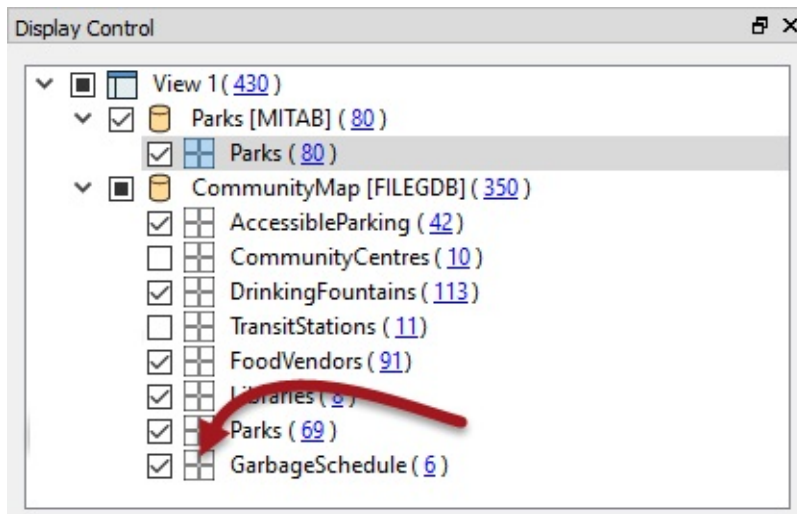
FME2017 introduces clickable feature counts. Clicking a feature count selects all of those features for display in the Feature Information and Table View windows, and also activates the tool File > Save Selected Data As.

Also notice that hovering over an entry in the list now causes an informational tooltip to appear:

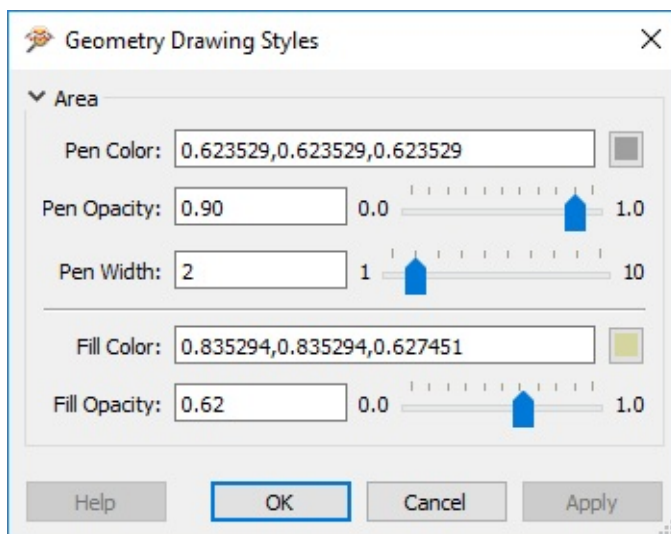


Style

Each feature type can be assigned a different color or style that applies to its geometries. To access this functionality click on the style icon for that feature type in the Display Control window:



The Drawing Style dialog allows you to set all manner of symbology for a feature. That includes feature color (and degree of transparency), point icon/symbol, point size, line thickness, etc:



Notice that the dialog only displays symbology options for the type of geometry available. Here it is for a polygon feature. Other geometry types display different options.

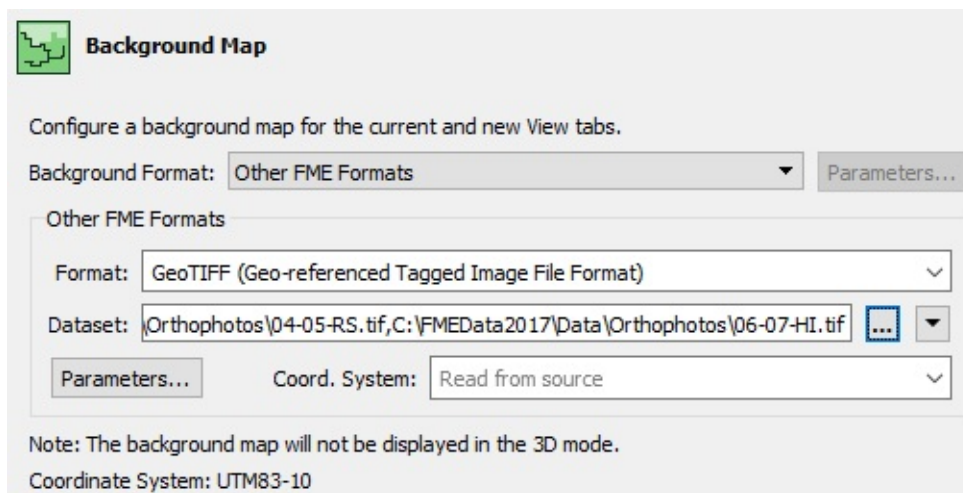
Miscellaneous Data Inspector Functionality

The FME Data Inspector has a number of miscellaneous functions that help users view background maps, filter data, and even translate data to a different format!

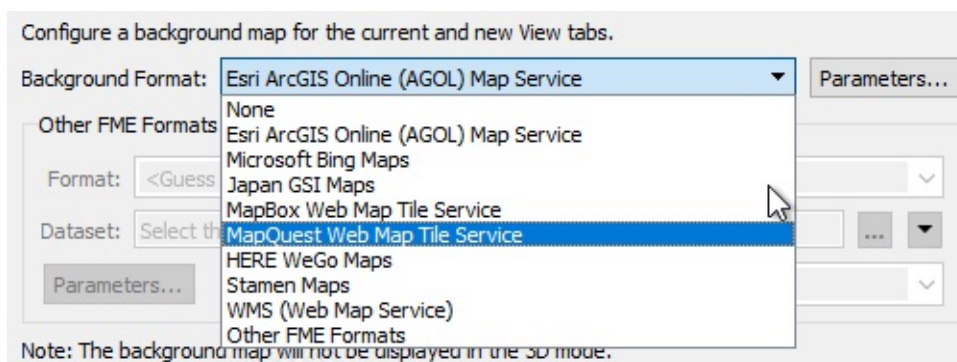
Background Maps

The ability to view maps (or other imagery) as a backdrop to your spatial data is activated by a tool under Tools > FME Options on the menubar.

The background map dialog lets the user select an existing dataset (of any FME-supported format) to use as a backdrop, like so:

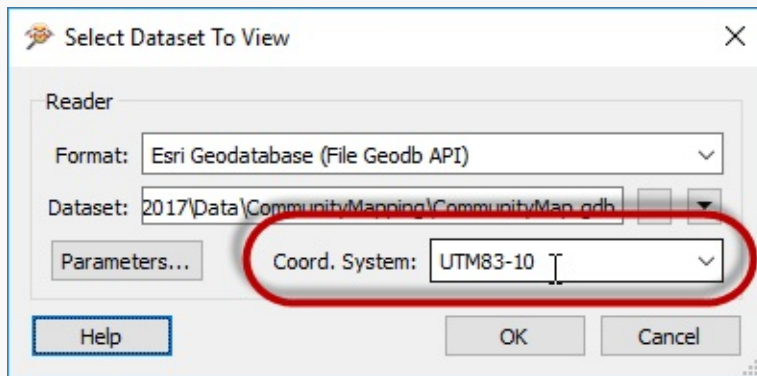


It's also possible to use a number of different web services that supply mapping on demand. Some of these - such as ArcGIS Online - do require an existing account:



Police Chief Webb-Mapp says...

When I'm investigating - sorry, inspecting - source data I know it must be referenced against a valid coordinate system for background data to be displayed. If the coordinate system is not recorded in the dataset itself, I can enter it into a field when opening the dataset



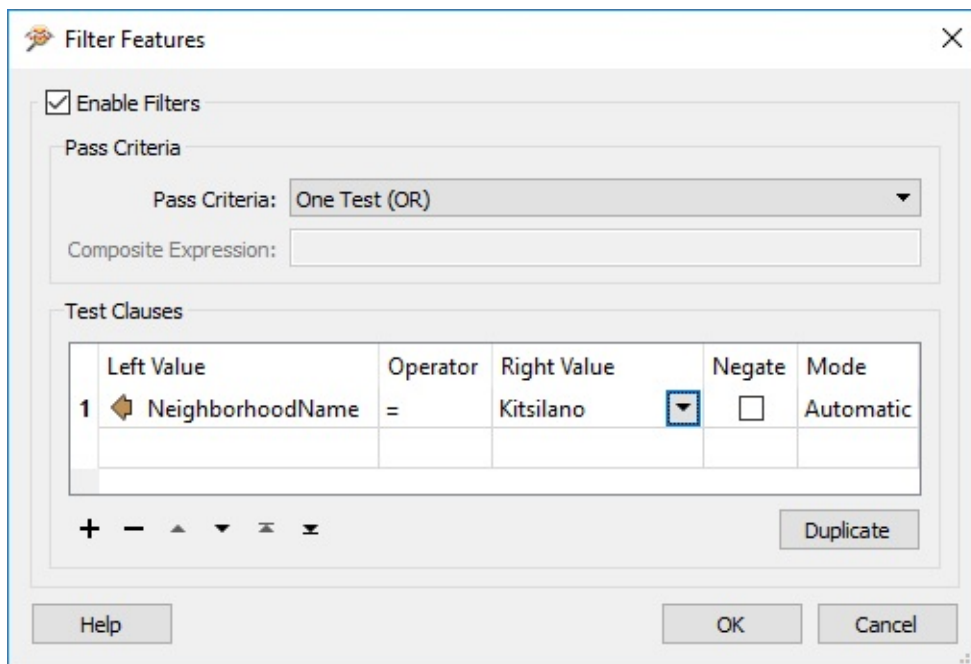
I've also deduced that it doesn't matter what coordinate system the data is referenced against; FME will automatically convert it to whatever system is being used by the background map.

Data Filtering

Data in the View window can be filtered by a set of user-defined criteria to show only the features that are required at the time. This functionality is activated by Tools > Filter Features on the menubar, or the filter icon on the toolbar:

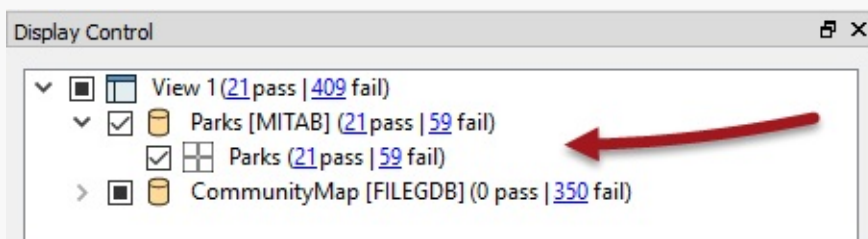


The Filter Features dialog allows a whole series of test clauses to be set up, using a number of operators to test the values of source data attributes. For example, here the user is filtering (keeping displayed) all features that are located in the neighborhood of Kitsilano.



NEW

In FME2017 the feature counts on the Display Control window now update to reflect the status of any filter that is applied, showing how many features pass the filter test and how many fail:



Save Tools

The FME Data Inspector save tool lets you save whatever data is currently being displayed in the view window. The data can be saved in any FME-supported format of your choice. Simply select File > Save Data As to open the prompt for saving data.

File > Save Selected Data As saves only the data that is currently selected in the view or table view windows.

Exercise 3 The FME Data Inspector

Data	Parks (MapInfo TAB) Fire Halls (GML - Geographic Markup Language) Neighborhoods (Google KML)
Overall Goal	Examine city data to select a suitable neighborhood to live in
Demonstrates	Viewing, symbolizing, and querying data with the FME Data Inspector
Start Workspace	None
End Workspace	None

The mayor of the city drops in to welcome you in your new job as technical analyst. He is fascinated by what FME can do and asks you to inspect some data, to help pick a neighborhood for him to buy a new house in! As a GIS professional we wouldn't normally do spatial analysis in this way, but the mayor is unlikely to know that and anyway it's good practice with the FME Data Inspector.

1) Start FME Data Inspector

Start the FME Data Inspector.

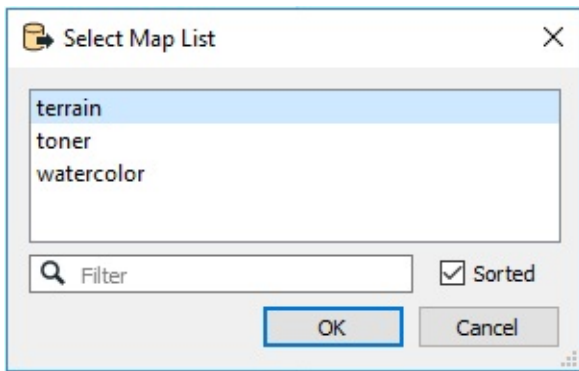
2) Set Background

When inspecting data it will help to have a background map to provide a sense of location. The FME Data Inspector is capable of displaying a backdrop from several different mapping services.

Select Tools > FME Options from the menubar. In the Background Map section, select a background map format. If you have an account with a specific provider, please feel free to use that. Otherwise select Stamen Maps:



Click the Parameters button. A map constraint (type) dialog will open. Select "terrain".

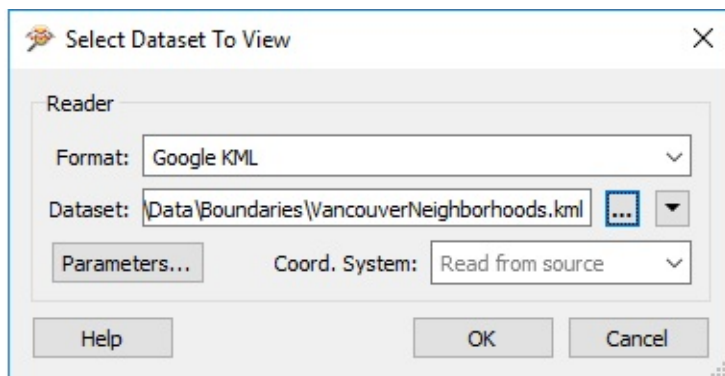


Click OK and OK again to close these dialogs.

3) Add Data

Now let's add some data. Select File > Open Dataset from the menubar. Set the reader parameters as follows:

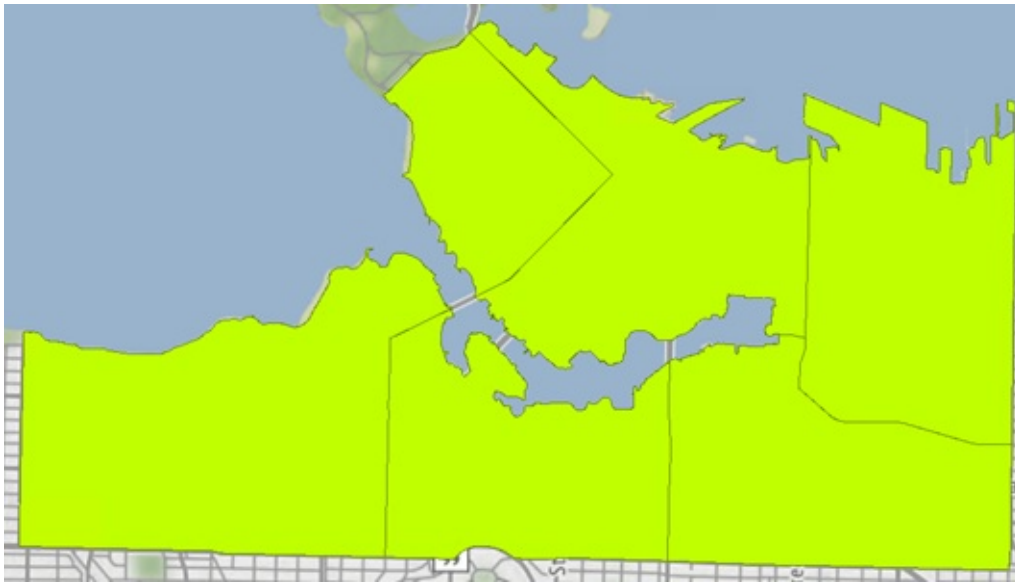
Reader Format	Google KML
Reader Dataset	C:\FMEData2017\Data\Boundaries\VancouverNeighborhoods.kml



Click OK to close the dialog and add the data.

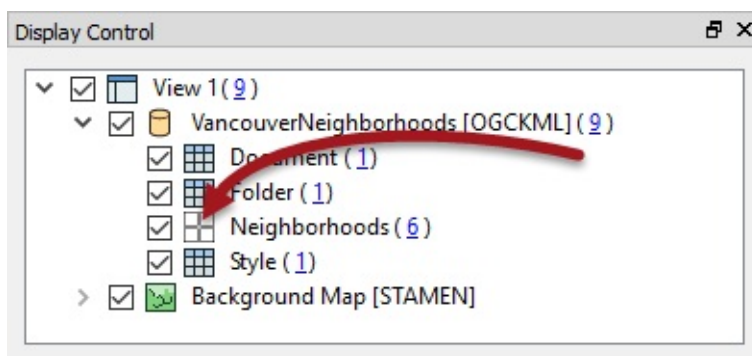
4) Set Data Symbology

The data is added to the Data Inspector, but it is a solid-filled color and therefore obscures the background data.

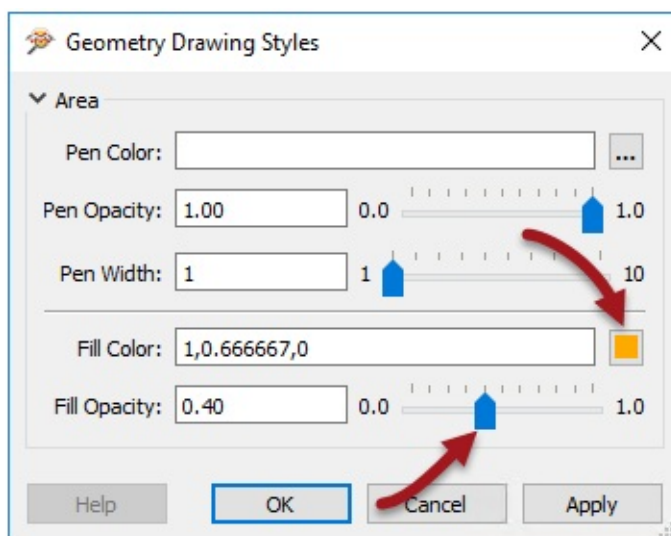


Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

We can fix this by setting the symbology. Click the symbology icon for the Neighborhoods data in the Display Control window:



In the style dialog that opens, select a suitable color for the neighborhoods (say, orange). Then set the Transparency/Opacity scale to be somewhat less than half (i.e. more transparent than opaque).



Click OK to accept the parameters and close the dialog. The background map will now be visible through the neighborhood data.

.1 UPDATE

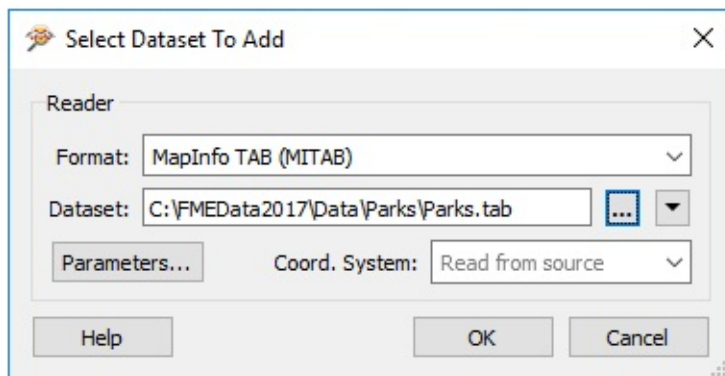
In FME2017.1, polygon features are now styled semi-transparent automatically, with a default Fill Opacity of 0.3

5) Add More Data

The mayor has a pet dog and therefore wishes to live in an area that has a dog park. We must therefore add some parks data to the view.

Select File > Add Dataset from the menubar. Set the reader parameters as follows:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2017\Data\Parks\Parks.tab



Click OK to close the dialog and add the data.

6) Filter Data

The parks data is opaque too (you may optionally change this to be more transparent) but more importantly we cannot tell which parks are dog parks.

Click on the tab marked Parks in the Table View window or, if the tab does not exist, select Parks from the drop-down list of tables.

Notice that there are many parks in the city, but that there is also a DogPark attribute to tell us which parks have a dog run area. Click on the DogPark name to sort the table data by that attribute.

Table View

Table: Parks [MITAB] - Parks Edit Schema...

Document x Parks x

	ParkId	RefParkId	ParkName	NeighborhoodName	EWStreet	NSStreet	DogPark	SpecialFeatures
1	29	181	Strathcona Park	Strathcona	Malkin Avenue	Hawks Avenue	Y	N
2	35	28	CRAB Park at Portside	Downtown	E Waterfront Road	Main Street	Y	N
3	50	-9999	Hadden Park	Kitsilano	<missing>	<missing>	Y	<missing>
4	53	43	Charleson Park	Fairview	Charleson Street	Laurel Street	Y	Y
5	56	14	Coopers' Park	Downtown	Nelson Street	Marinaside Crescent	Y	Y
6	61	-9999	Sunset Beach Park	West End	<missing>	<missing>	Y	<missing>
7	73	-9999	Nelson Park	West End	<missing>	<missing>	Y	<missing>
8	1	-9999	<missing>	Kitsilano	<missing>	<missing>	N	<missing>
9	2	208	Rosemary Brown Park	Kitsilano	W 11th Avenue	Vine Street	N	N

Q in any column 80 row(s)

Now we know which are dog parks, clicking on a feature will highlight it on the display window. However, it would be easier to find dog parks if we could filter the data. Therefore choose Tools > Filter Features from the menubar.

In the Filter Features dialog, double click in the Left Value field, click the drop down arrow, and select Attribute Value. Choose DogPark as the attribute to filter by and click OK.

For the Operator field select the = (equals) symbol, if it is not already selected.

For the Right Value field, click in the field and type the character Y (it should be upper case, not lower).

☒ Enable Filters

Pass Criteria

Pass Criteria: One Test (OR)

Composite Expression:

Test Clauses

	Left Value	Operator	Right Value	Negate	Mode
1	DogPark	=	<input type="checkbox"/> Y	<input type="checkbox"/>	Automatic

+ - < > < >

Duplicate

Filtering in the Data Inspector is applied to all visible data (if you clicked OK now the Neighborhood data would also be filtered out) therefore we must also add a clause to enable the neighborhood data to remain on screen.

Create a second test clause using the same techniques as before. This time test for where NeighborhoodID > (is greater than) 0 (zero)

	Left Value	Operator	Right Value	Negate	Mode
1	DogPark	=	<input type="checkbox"/> Y	<input type="checkbox"/>	Automatic
2	NeighborhoodId	>	<input type="checkbox"/> 0	<input type="checkbox"/>	Automatic

The Pass Criteria parameter should be set (or left as) "One Test (OR)."

☒ Enable Filters

Pass Criteria

Pass Criteria: One Test (OR) ▼

Composite Expression:

Now click OK to close the dialog. The display will be filtered to show only the neighborhood features plus parks with a dog run facility.



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

The Display Control Window will also show the effects of the filter:

- ✓ ☒ View 1 (13 pass | 84 fail)
 - ✓ ☒ Parks [MITAB] (7 pass | 73 fail)
 - ☒ Parks (7 pass | 73 fail)
 - ✓ ☒ VancouverNeighborhoods [OGCKML] (6 pass | 3 fail)
 - ☒ Document (0 pass | 1 fail)
 - ☒ Folder (0 pass | 1 fail)
 - ☒ Neighborhoods (6 pass | 0 fail)
 - ☒ Style (0 pass | 1 fail)
 - > ☒ Background Map [STAMEN]

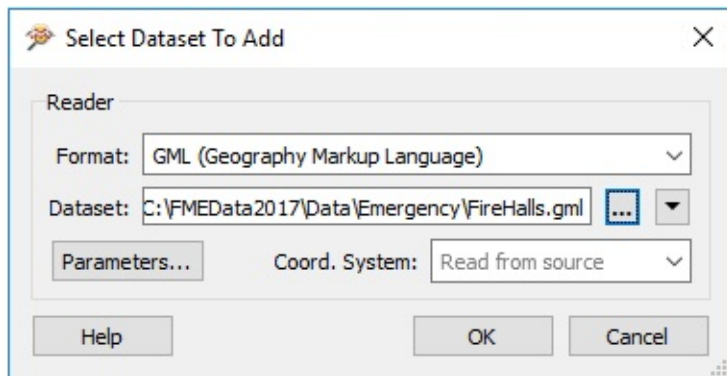
7) Add Data

The mayor's dog really is his best friend, and the mayor refuses to live in an area where

there are no rescue services, just in case he gets lost chasing a cat! So let's add some emergency facilities data.

Select File > Add Dataset from the menubar. Set the reader parameters as follows:

Reader Format	GML (Geography Markup Language)
Reader Dataset	C:\FMEData2017\Data\Emergency\FireHalls.gml

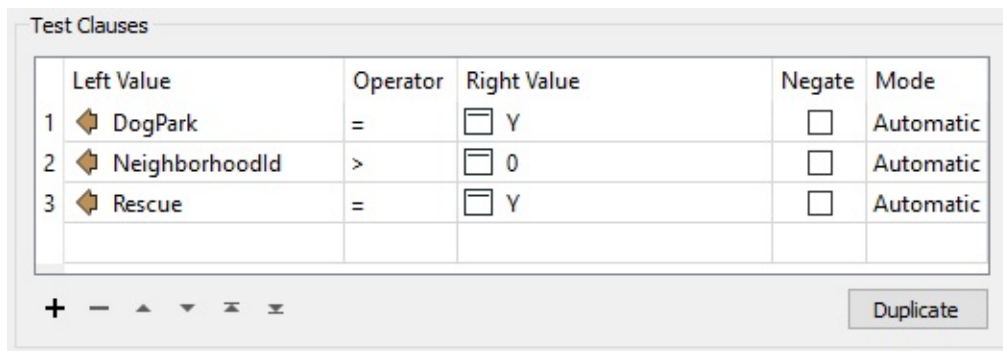


Click OK to close the dialog and add the data.

8) Filter Data

Initially no data will appear on screen because we already have a filter set that will exclude it.

So, again select Tools > Filter Features from the menubar. This time set up a test to filter where Rescue = Y (i.e. Fire Halls which are also a rescue facility).



At this point you should be able to suggest to the mayor a neighborhood that has both a dog park and an emergency rescue facility. Click on the neighborhood feature to find out which it is.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Set a background map for the FME Data Inspector*
- *Add datasets to an existing view in the FME Data Inspector*
- *Set symbology for features in the FME Data Inspector*
- *Query and sort data in the Data Inspector Table View window*
- *Filter data using test clauses in the FME Data Inspector*

Translation Previews

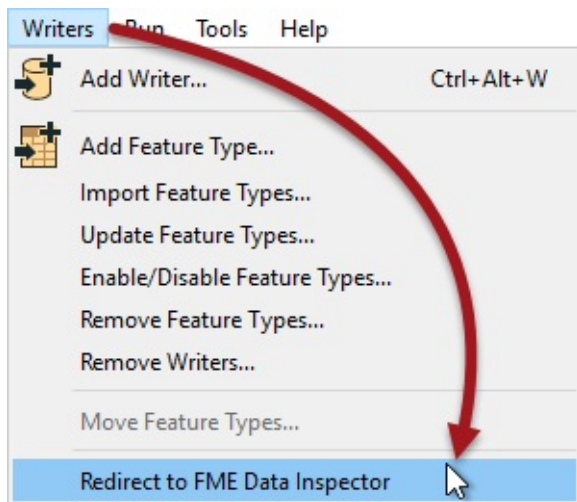
A key ability of FME is communication of data between Workbench and the FME Data Inspector.

Redirect to FME Data Inspector

In some cases it's desirable to inspect output data, but undesirable to actually have to write the data to do so. In other words, you want to preview what the output of a translation will be.

For example, it would be useful to preview the results of a workspace that updates a spatial database. That way mistakes can be detected before writing to the database.

The Workbench tool to do this is the menu option Writers > Redirect to FME Data Inspector:



When this setting is applied, the output from a translation is redirected away from the specified output and sent directly to FME Data Inspector instead.

This setting is a toggle, which means that each subsequent selection alternately turns the setting on and off.

```

=====
Features Written Summary
=====
Total Features Written                                0
=====
~
~ *** All writer output has been redirected to the Inspector application ***
~
~
=====
Translation was SUCCESSFUL with 5 warning(s) (0 feature(s) output)
FME Session Duration: 1.7 seconds. (CPU: 1.2s user, 0.2s system)

```

71

Module Review

This module introduced you to FME and investigated the basics of FME data translations.

What You Should Have Learned from this Module

The following are key points to be learned from this session:

Theory

- FME is a tool to translate and transform spatial data. Quick Translation is the technique of carrying out a translation with minimum user intervention. The semantic nature of FME is the means by which this is permitted.
- FME Workbench is an application to graphically define a translation and the flow of data within it. This definition is known as a workspace and can be saved to a file for later use.
- The Generate Workspace dialog is the main method by which a *quick translation* can be set up in FME Workbench.
- Data Inspection is a technique for checking and verifying data before, during, and after a translation. The FME Data Inspector is a tool for inspecting data. Multiple datasets – including data from different formats – can be opened within the same view window.

FME Skills

- The ability to start FME Workbench, set up a *quick translation*, and run it
- The ability to start the FME Data Inspector, open a dataset in a new view, and add a dataset to an existing view
- The ability to navigate a dataset and to query features within it
- The ability to control minor FME Data Inspector functionality for debugging data and translations
- The ability to inspect data by redirecting it from FME Workbench to the FME Data Inspector

Further Reading

For further reading why not browse [articles about data formats](#) or [articles tagged with Data Inspector](#) on our blog?

Questions

Here are the answers to the questions in this chapter.

Miss Vector says...

ETL is an acronym for...?

1. *Extra-Terrestrial Lifeform*
2. **Extract, Transform, Load**
3. *Express Toll Lane*
4. *Eat, Transform, Love*

Miss Vector says...

FME can seamlessly translate between so many formats because it has...

1. *A sentient data dictionary*
2. *A retro-encabulator*
3. **A rich data model**
4. *A core of unicorn hairs*

Miss Vector says...

Which of the following applications is NOT a part of FME Desktop?

1. *FME Workbench*
 2. *FME Integration Console*
 3. **FME Server Console**
 4. *FME Data Inspector*
-

Miss Vector says...

FME Workbench allows you to define flows of data in which way...

- 1. Graphically**
2. Telepathically
3. Problematically
4. By writing lots of code in C++ or Java

Miss Vector says...

Which of these is a window in FME Workbench?

1. The Maths Window
2. The Geography Window
3. The English Literature Window
- 4. The History Window**

Miss Vector says...

Which of these is NOT an arrangement of Windows in FME Workbench?

1. Stacked
2. Floating
- 3. Double-Glazed**
4. Tabbed

Miss Vector says...

Q) In the Generate Workspace dialog, why might it be useful to set the data format before browsing for the source data?

A) Because then the explorer dialog only displays files whose extension matches the format type

Miss Vector says...

Which of these is *NOT* a way to set the format of a translation?

1. Typing the format name
2. Selecting the format from a drop-down list
3. Browsing for the format in the formats gallery
4. By selecting a dataset with a known file extension
5. **None of the above (they are all valid ways to set the format)**

Miss Vector says...

Which key is a shortcut to run a workspace?

1. F4
2. **F5**
3. F5.6
4. F#

Miss Vector says...

When you are inspecting **schema**, what are you trying to verify?

1. The color and linestyle of features
2. The number of features
3. **The layers (classes, tables, types)**
4. Where the nearest coffee shop is

Miss Vector says...

Open a dataset and right-click on records and column-headers in the Table View window. Which of the following is *NOT* an available option(s):

1. Sort (Alphabetical or Numeric)
2. Inspect Value
3. **Cut/Copy/Paste**
4. Save Selected Data As

Exercise 4 Tourist Bureau Project	
Data	Community Mapping/Food Vendors (Esri Geodatabase)
Overall Goal	Create a GPS-compatible dataset of food vendors for the local tourist bureau
Demonstrates	Basic Data Translation and Data Inspection
Start Workspace	None
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Basics-Ex4-Complete.fmw

You've barely started in your new job, but requests are coming in fast!

The local tourist bureau is running a promotion where they provide tourists with a GPS device to help them visit street food vendors in the city. Your manager wonders whether you can use FME to produce the data to be used in this scheme.

Let's get onto that right away shall we?!

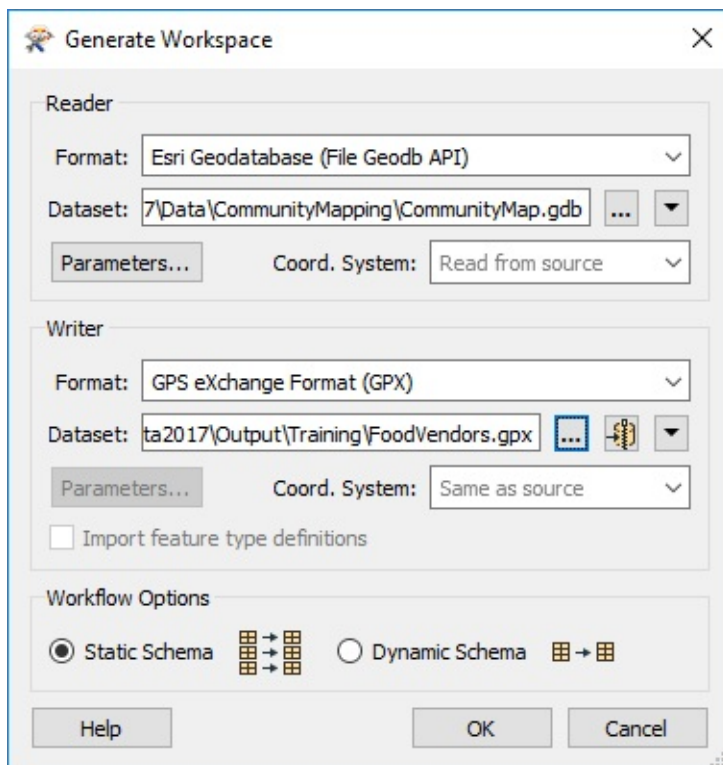
1) Start FME Workbench

Start FME Workbench. In the Create Workspace section of the Start window, choose the option to Generate (Workspace). When prompted generate a translation with the following parameters:

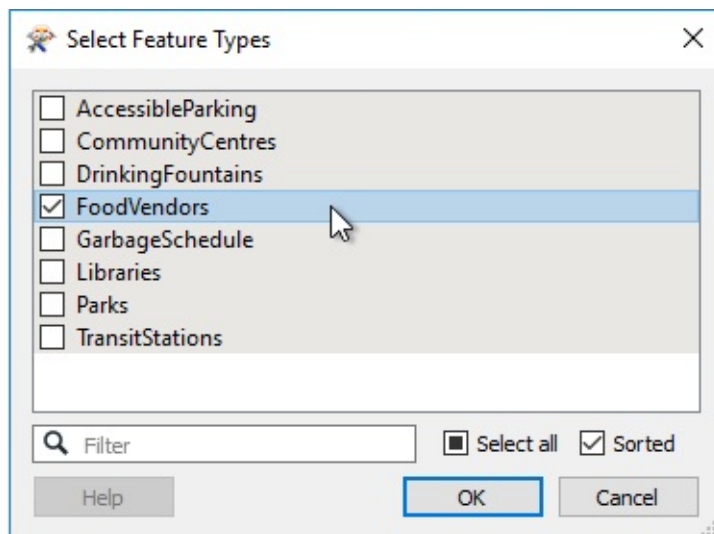
Reader Format	Esri Geodatabase (File Geodb API)
Reader Dataset	C:\FMEDData2017\Data\CommunityMapping\CommunityMap.gdb
Writer Format	GPS eXchange Format (GPX)
Writer Dataset	C:\FMEDData2017\Output\Training\FoodVendors.gpx

.1 UPDATE

In FME2017.1 the format is now called Esri Geodatabase (File Geodb Open API)



Click OK to accept the parameters. When prompted which tables to use from the source data (there are several) deselect all tables except for FoodVendors and click OK to create the workspace.

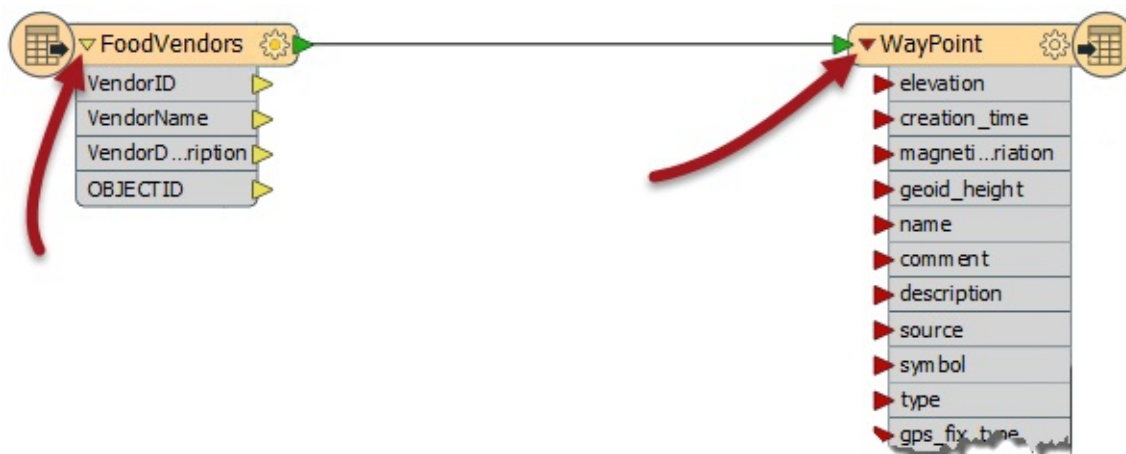


2) Connect Reader/Writer

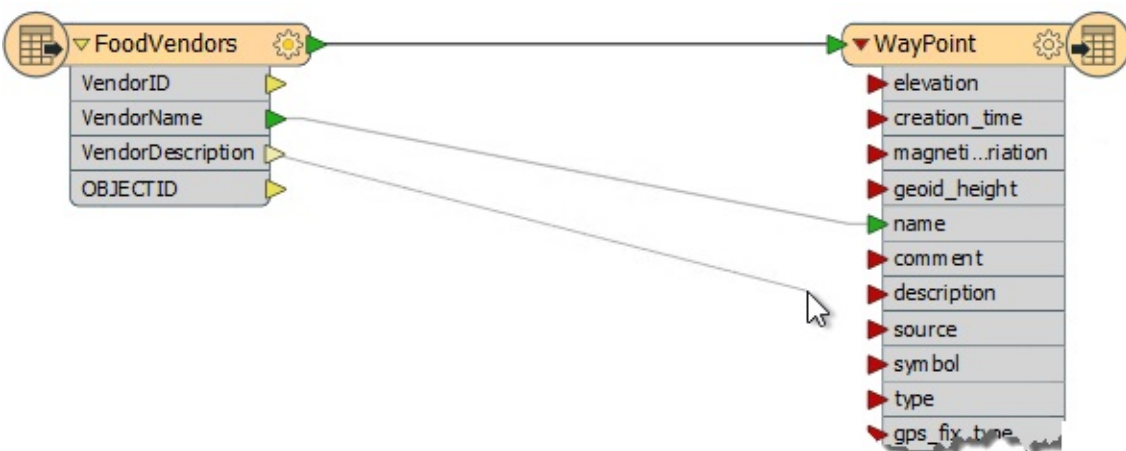
When first created, the Reader and Writer are not connected in this workspace. Connect them by dragging a connection from the output port of the Reader to the input port of the Writer object labelled WayPoint:



Click the expand buttons on the two objects to expose the list of attributes on each:



Now drag a connection between the Reader attribute VendorName and the Writer attribute name. Repeat the process for VendorDescription and description:



The technique of connecting objects like this is called Schema Mapping, and we shall learn more about it later.

3) Run Workspace

Save the workspace so you have a copy of it, then run the workspace by pressing the green play button. The workspace will run and the data written to a Garmin POI dataset.

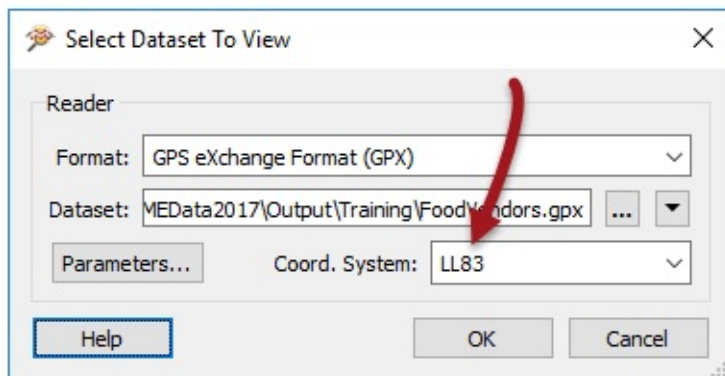
```
=====
                        Features Read Summary
=====
FoodVendors                                                    91
=====
Total Features Read                                           91
=====
                        Features Written Summary
=====
WayPoint                                                       91
=====
Total Features Written                                         91
=====
Translation was SUCCESSFUL with 0 warning(s) (1 feature(s) output)
FME Session Duration: 1.9 seconds. (CPU: 1.2s user, 0.2s system)
```

4) Inspect Data

Go to the FME Data Inspector. Select File > Open Dataset from the menubar. This opens

the dialog titled "Select Dataset to View".

Set the format type and select the GPX dataset. However, the GPX format does not record its coordinate system inside the dataset, so to include a background map you must also set the coordinate system (LL83) in this dialog.

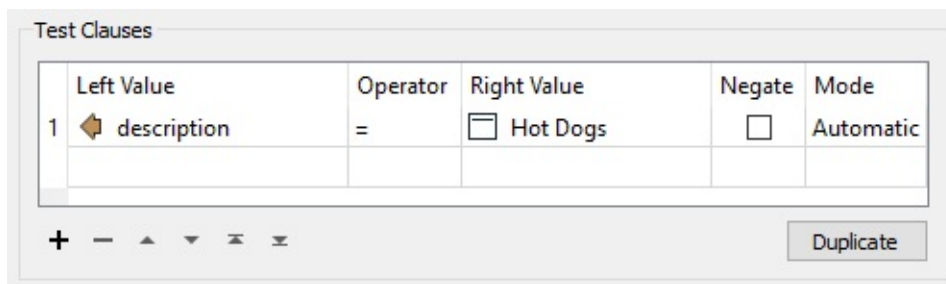


NB: Because of the coordinate system limitation for this format, you can't use the Inspect options inside Workbench. That's because the data will be passed directly to the Data Inspector without the option to set the coordinate system. You have to open it manually as above.

Click OK and the dataset will be opened for you to verify that it is correct.

5) Filter Data

All this talk of food is making you hungry. It must be lunchtime. To find somewhere to get a quick lunch filter the data to show hot dog vendors in the city.



CONGRATULATIONS

By completing this exercise you proved you know how to:

- Create and run a workspace in FME Workbench
- Carry out basic 'schema mapping' in FME Workbench
- Use FME Workbench functionality to open a dataset in the FME Data Inspector
- Filter data using test clauses in the FME Data Inspector

Data Transformation

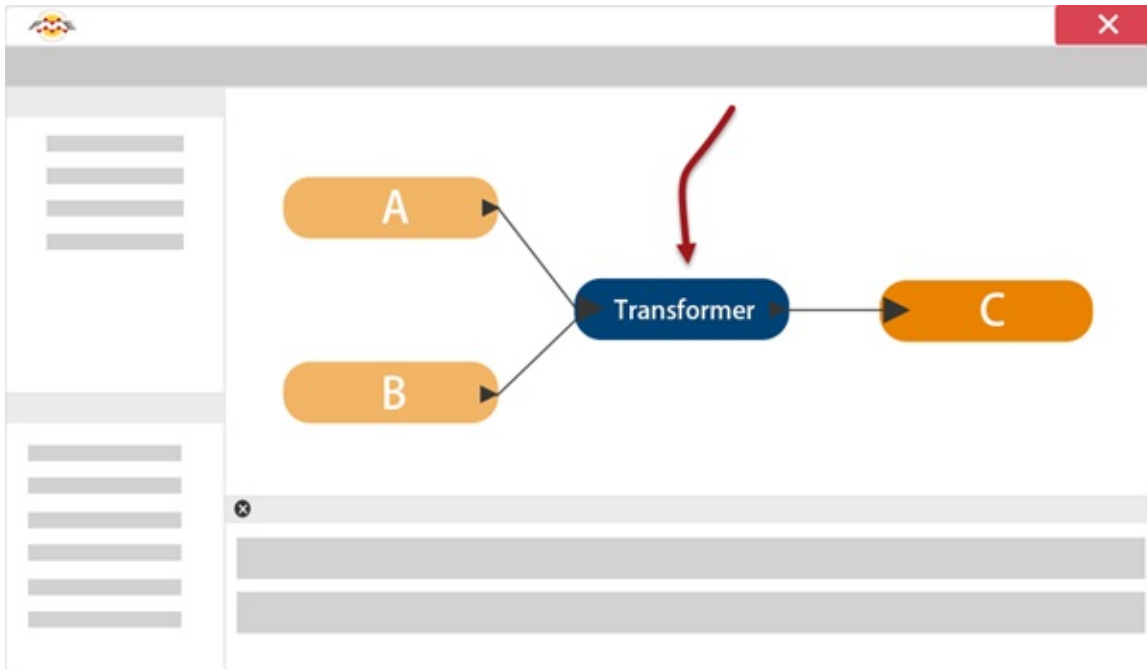
Data Transformation is the ability to manipulate data during Format Translation – even to the extent of having an output greater than the sum of the inputs!



What is Data Transformation?

Data Transformation is FME's ability to manipulate data.

The transformation step occurs during the process of format translation. Data is read, transformed, and then written to the new format.



Data Transformation Types

Data transformation can be subdivided into two distinct types: *Structural Transformation* and *Content Transformation*.

Structural Transformation

Structural transformation is perhaps better called 'reorganization'. It refers to FME's ability to channel data from source to destination in an almost infinite number of arrangements.

This includes the ability to merge data (as in the image above), divide data, re-order data, and define custom data structures.

Transforming the structure of a dataset is carried out by manipulating its schema.

Content Transformation

Content transformation is perhaps better called 'revision'. It refers to the ability to alter the substance of a dataset.

Manipulating a feature's geometry or attribute values is the best example of how FME can transform content.

Content transformation can take place independently or alongside structural transformation.

Mr Flibble says...

Mr. Flibble - certified FME jester - here to entertain you. Here's a riddle, but can you solve it?

The most common format translation defined with FME is from Esri Shapefile to which format?

- 1. Esri Geodatabase*
- 2. AutoCAD DXF/DWG*
- 3. Google KML*
- 4. Esri Shapefile*

If you're in a class, have a group vote!

Structural Transformation

Transforming a dataset's structure requires knowledge of *schemas* and how to use FME to manipulate them.

Schema Concepts

A **schema** is the structure of a dataset or - more accurately - a formal definition of a dataset's structure. FME uses the term 'schema', but you may know this as 'data model'.

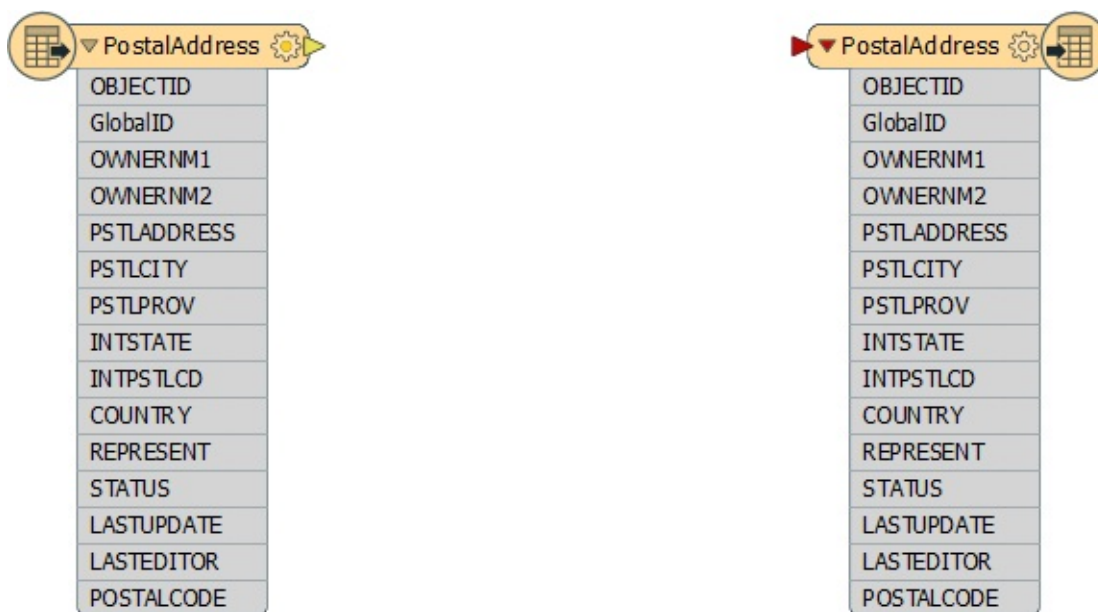
Each dataset has its own unique schema; it includes layers, geometries, attributes, and other rules that define or restrict its content.

Schema Representation

When a new workspace is created, FME scans the source datasets and creates a visual representation of its schema on the left side of the workspace canvas.

On the right side of the canvas, FME creates a representation of how this schema will be duplicated in the chosen destination format.

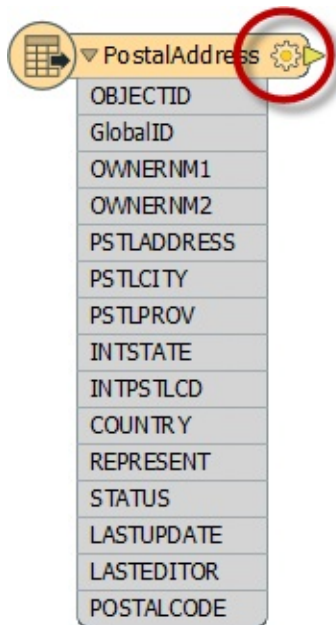
The workspace therefore reads from left to right.



Feature Type Schemas

A schema is not just represented as a set of layers on the Workbench canvas; for example, each different layer in a dataset can have a different structure.

This part of the schema is revealed by clicking the cog-wheel icon on the canvas object representing that layer.

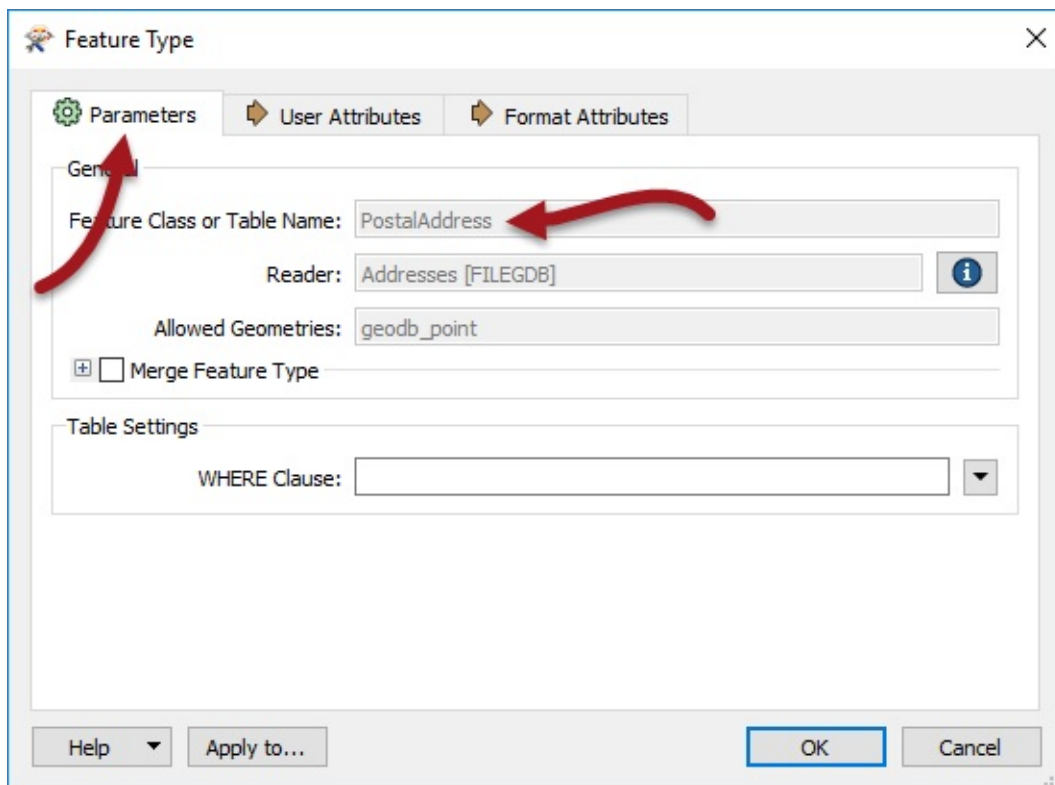


This opens the Feature Type Properties dialog. This dialog represents schema under a number of tabbed tables.

NEW

In FME2017 or newer feature type specific parameters can also be found in the Parameter Editor window.

Under the Parameters tab is a set of general parameters, such as the name of the feature type (in this case PostalAddress) the allowed geometry types, and the name of the parent dataset.



The 'Feature Type' dialog box is shown with the 'General' tab selected. A red arrow points to the 'Parameters' tab icon, and another red arrow points to the 'Feature Class or Table Name' field.

Feature Type

Parameters | User Attributes | Format Attributes

General

Feature Class or Table Name: PostalAddress

Reader: Addresses [FILEGDB] ⓘ

Allowed Geometries: geodb_point

☐ Merge Feature Type

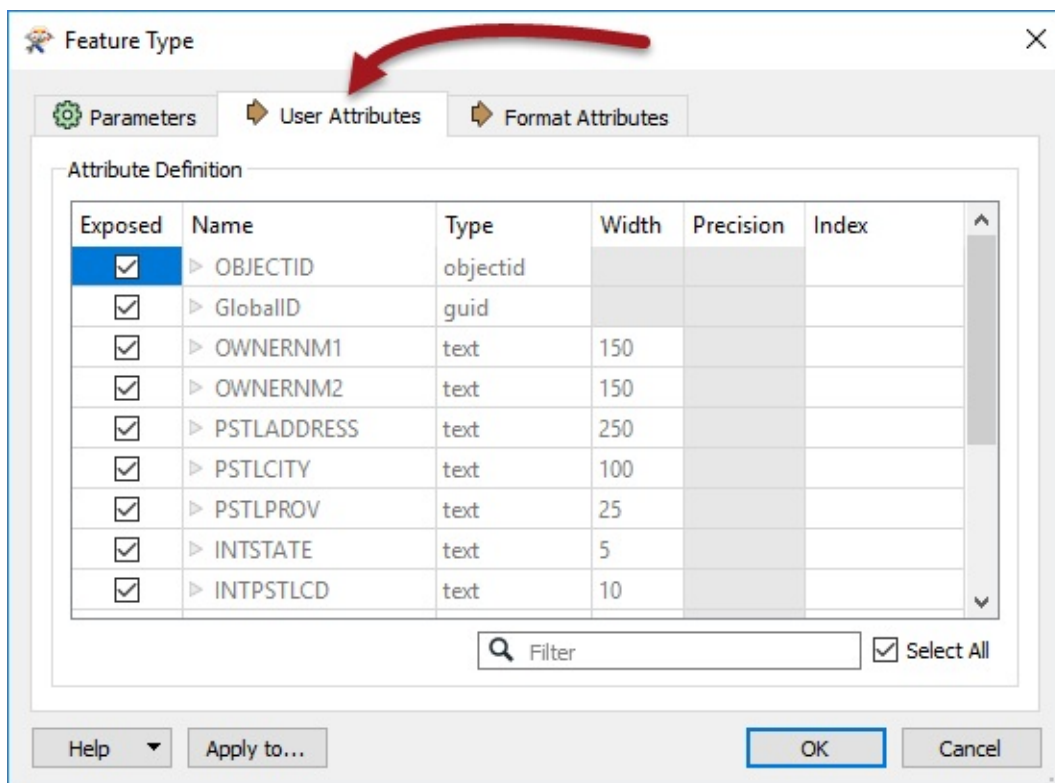
Table Settings

WHERE Clause:

Help Apply to... OK Cancel

General parameters are those that you might find for any feature type in any format. However, below this are a set of specific parameters for this particular feature type.

The User Attributes tab shows a list of attributes. Each attribute is defined by its name, data type, width, and number of decimal places.



The 'Feature Type' dialog box is shown with the 'User Attributes' tab selected. A red arrow points to the 'User Attributes' tab icon.

Feature Type

Parameters | User Attributes | Format Attributes

Attribute Definition

Exposed	Name	Type	Width	Precision	Index
<input checked="" type="checkbox"/>	OBJECTID	objectid			
<input checked="" type="checkbox"/>	GlobalID	guid			
<input checked="" type="checkbox"/>	OWNERNM1	text	150		
<input checked="" type="checkbox"/>	OWNERNM2	text	150		
<input checked="" type="checkbox"/>	PSTLADDRESS	text	250		
<input checked="" type="checkbox"/>	PSTLCITY	text	100		
<input checked="" type="checkbox"/>	PSTLPROV	text	25		
<input checked="" type="checkbox"/>	INTSTATE	text	5		
<input checked="" type="checkbox"/>	INTPSTLCD	text	10		

Filter Select All

Help Apply to... OK Cancel

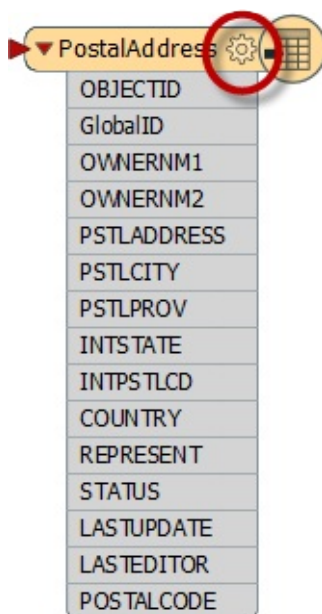
NEW

Prior to FME2017, feature type specific parameters were found in a separate tab. In FME2017 they have been merged under the General tab.

Writer Schema

All of the above information goes to make up the Reader schema. It is literally **"what we have"**.

There is also a schema for each Writer in FME. As with the reader, you can access a writer schema in the Parameter Editor window or by opening the properties dialog for a Writer feature type:



By default, the Writer schema (**"what we want"**) is a mirror image of the source, so the output from the translation will be a duplicate of the input. This allows users to translate from format to format without further edits (Quick Translation).

If 'what we want' is different to the default schema in FME, we simply have to change it using **Schema Editing**.

TIP

FME supports 300+ formats and there are almost as many terms for the way data is subdivided. The most common terms are layer, table, class, category, level, or object.

As noted, the general FME term for these subdivisions is Feature Type, and that is what is used throughout the manual.

Be aware though, that all dialogs in FME Workbench use format-specific terminology. So, screenshots in the manual may display different terms depending on what format of data is being used!

Schema Editing

Schema Editing is the process of altering the Writer schema to customize the structure of the output data. One good example is renaming an attribute field.

After editing, the source schema still represents *"what we have"*, but the destination schema now truly does represent *"what we want"*.

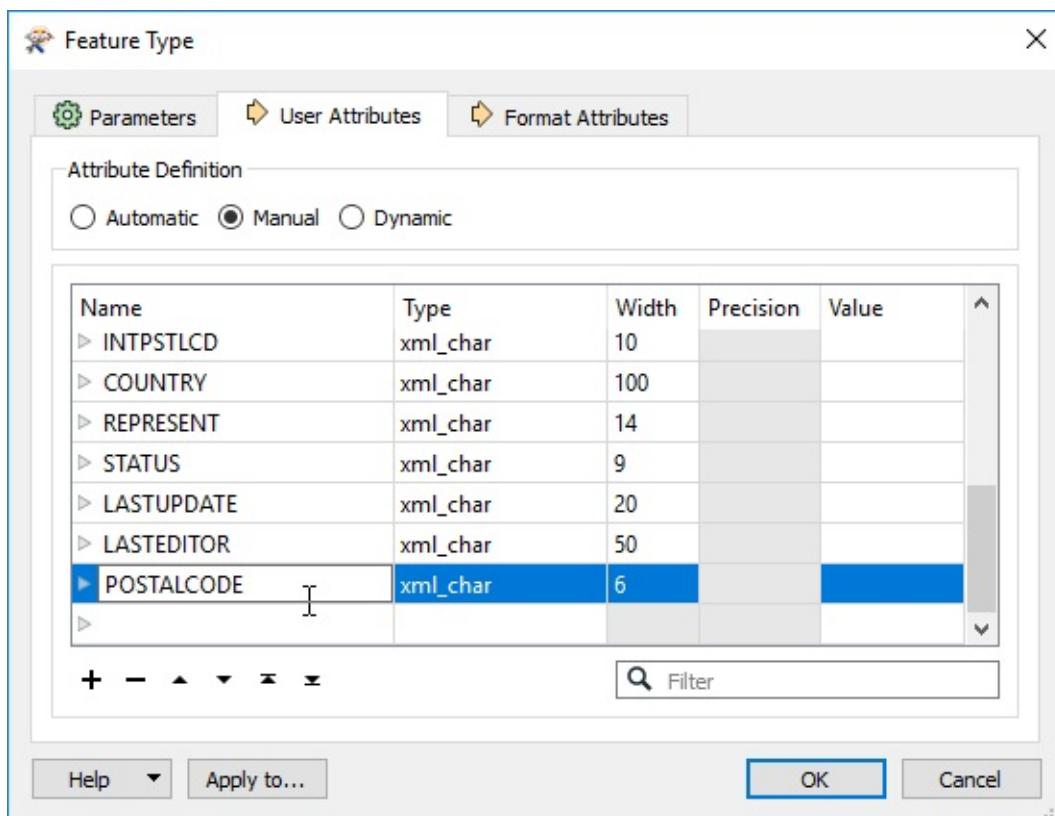
Editable Components

There are a number of edits that can be performed, including, but not limited to the following.

Attribute Renaming

Attributes on the destination schema can be renamed, such as renaming POSTALCODE to PSTLCODE.

To rename an attribute open the Feature Type Properties dialog and click the User Attributes tab. Click the attribute to be renamed and enter the new name.



Attribute Type Changes

Any attribute on the writer schema can have a change of type; for example, changing STATUS from a character field to a boolean field.

To change an attribute type open the Feature Type Properties dialog and click the User Attributes tab. Use the Data Type field to change the type of an attribute.

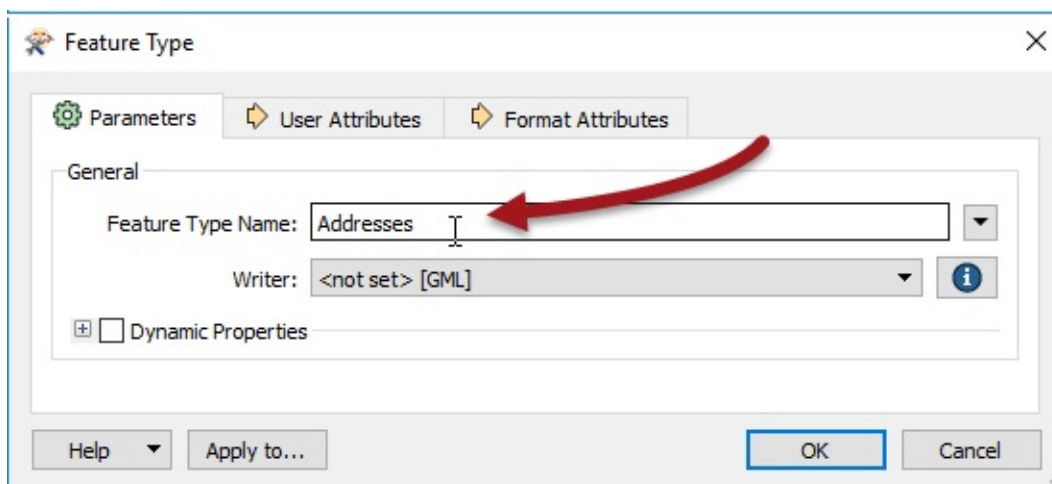
Name	Type	Width	Precision	Value
▶ INTPSTLCD	xml_char	10		
▶ COUNTRY	xml_char	100		
▶ REPRESENT	xml_char	14		
▶ STATUS	xml_char	9		
▶ LASTUPDATE	xml_boolean	20		
▶ LASTEDITOR	xml_buffer	50		
▶ POSTALCODE	xml_byte	6		
▶	xml_char			
▶	xml_date			
▶	xml_datetime			
▶	xml_decimal			
▶	xml_geometry			
▶	xml_int16			
▶	xml_int32			
▶	xml_int64			

TIP

The Data Type column for an attribute shows only values that match the permitted types for that data format. For example, an Oracle schema permits attribute types of varchar or clob. MapInfo does not support these data types so they would never appear in a MapInfo schema. The above screenshot shows data types for the GML format.

Feature Type Renaming

To rename a destination feature type (for example, rename PostalAddress to Addresses) open the Feature Type Properties dialog. Click the Parameters tab. Click in the Feature Type Name field and edit the name as required.



You can also rename a feature type by clicking on it in the Workbench canvas and pressing the F2 key.

Geometry Type Changes

To change the permitted geometry for a feature type, (for example, change the permitted geometries from lines to points) open the Feature Type Properties dialog. Click the General tab. Choose from the list of permitted geometries.

This field is only available where the format requires a decision on geometry type. Where formats allow any mix of geometry type in a single feature type, this setting is greyed out.

Once the user has made all the required changes to the Writer schema, the workspace reflects "*what we want*" - but it doesn't yet specify how the Reader and Writer schemas should be connected together. This is the next task and it is called **Schema Mapping**.

TIP

*You might be wondering "what would happen if I edited the **Reader** schema, instead of the Writer"?*

Well, by default, you can't! The schema fields for a Reader are greyed out to stop you, since they would no longer match the Reader dataset.

Attribute Definition

Exposed	Name	Type	Width	Precision	Index
<input checked="" type="checkbox"/>	▶ OBJECTID	objectid			
<input checked="" type="checkbox"/>	▶ GlobalID	guid			
<input checked="" type="checkbox"/>	▶ OWNERNM1	text	150		
<input checked="" type="checkbox"/>	▶ OWNERNM2	text	150		
<input checked="" type="checkbox"/>	▶ PSTLADDRESS	text	250		
<input checked="" type="checkbox"/>	▶ PSTLCITY	text	100		
<input checked="" type="checkbox"/>	▶ PSTLPROV	text	25		

☒ Select All

There are a few, rare, use cases - but they're outside the scope of this training course!

Schema Mapping

Schema Mapping is the second key part towards data restructuring in FME.

In FME Workbench, one side of the workspace shows the source schema ("*what we have*") and the other side shows the destination schema ("*what we want*"). Initially the two schemas are automatically joined when the workspace is created, but when edits occur then these connections are usually lost.

Schema Mapping is the process of connecting the source schema to the destination schema in a way that ensures the correct reader features are sent to the correct writer feature types and the correct reader attributes are sent to the correct writer attributes.

In FME, it is permitted to do this mapping in any way that is desired.

Ms. Analyst says...

Hi. I'm Ms. Analyst, one of your colleagues at the city. I think of Schema Editing and Mapping as a means of reorganizing data.

A good analogy is a wardrobe full of clothes. When the wardrobe is reorganized you throw out what you no longer need, reserve space for new stuff that you're planning to get, and move existing items into a more usable arrangement.

The same holds true for spatial data restructuring: it's the act of reorganizing data to make it more usable.

In Workbench's intuitive interface, the most common way to make feature type and attribute connections is by dragging connecting lines between these parts of the schema.

Feature Mapping

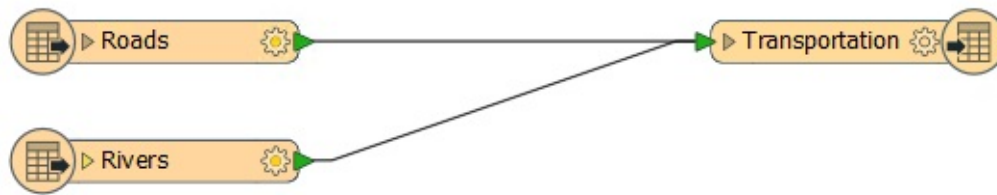
Feature mapping is the process of connecting source feature types to destination feature types.

Feature Mapping is carried out by clicking the output port of a source feature type, dragging the arrowhead across to the input port of a destination feature type, and releasing the mouse button.

Here, a connecting line from source to destination feature type is created by dragging the arrowhead from the source to the destination.



Merging and splitting of data is permitted. Here, a user wishes to create a single layer called Transportation and so is merging two input feature types (Roads and Rail) into one output feature type (Transportation).

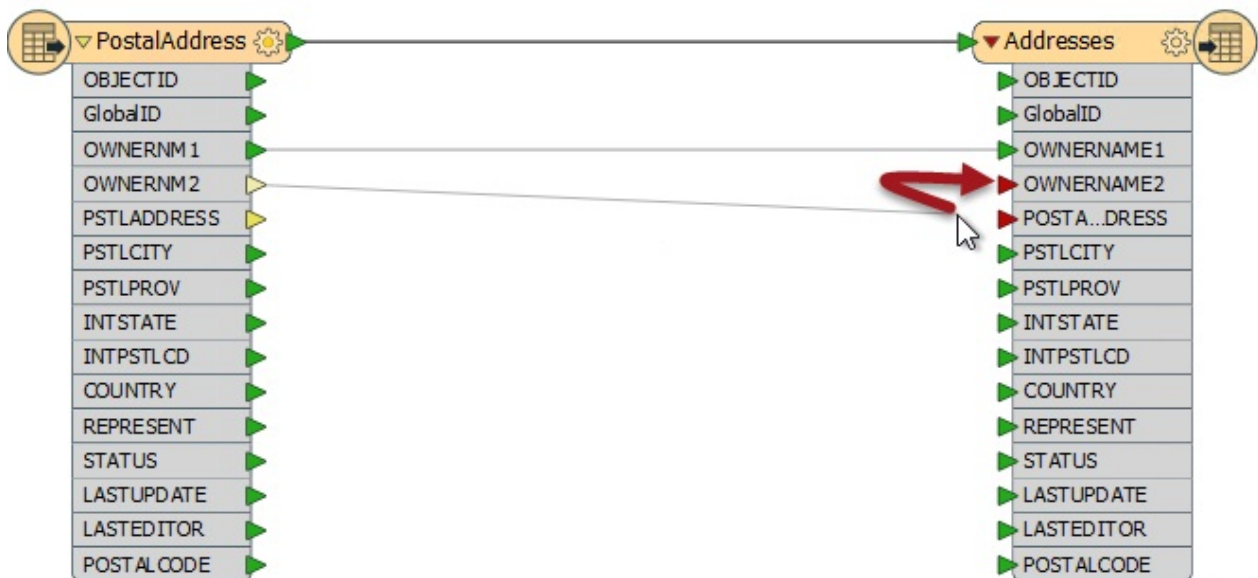


Attribute Mapping

Attribute Mapping is the process of connecting reader attributes to writer attributes.

Attribute Mapping is performed by clicking the output port of a reader attribute, dragging the arrowhead to the input port of a writer attribute, and releasing the mouse button.

Here feature mapping has been carried out already and attribute connections are being made.



Notice the green, yellow, and red color-coding that indicates which attributes are connected.

Green ports indicate a connected attribute. Yellow ports indicate a reader attribute that's unconnected to a writer. Red ports indicate a writer attribute that's unconnected to by a reader.

Feature mapping connections (or links) are shown with a thick, black arrow.

Attribute mapping connections are shown with a thinner, gray arrow.



Attributes with the same name in reader and writer are connected automatically, even though a connecting line might not be visible; the port color is the key.

WARNING

*Names are case-sensitive, therefore **ROADS** is not the same as **roads**, **Roads**, or **rOADS**.*

That's important to know if you are relying on automatic attribute connections!

Exercise 1 Grounds Maintenance Project - Schema Editing	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Structural Transformation, Schema Editing
Start Workspace	None
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex1-Complete.fmw

You have just landed a job as 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 have been assigned to this project and will use FME to provide a dataset of this information.

The first step in this example is to rename existing attributes and create new ones in preparation for the later area calculations.

1) Start Workbench

Use the Generate Workspace dialog to create a workspace using these parameters:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEDData2017\Data\Parks\Parks.tab
Writer Format	MapInfo TAB (MITAB)
Writer Dataset	C:\FMEDData2017\Output\Training

Yes! Here we write back to the same format of data we are reading from!

2) Update Attributes

FME creates a workspace where the destination schema matches the source. However, the end user of the data has requested the attributes get cleaned up so that unnecessary information is removed. Also others need to be renamed and some extra ones added to store the calculation results.

Click the writer feature type to have its parameters appear in the Parameter Editor window (or open the Feature Type Properties dialog by clicking the feature type's Properties button). Click the User Attributes tab to open a list of the destination attributes. The Parameter Editor dialog will look like this:

Parameter Editor

← → ↗ ?

Table Parameters

Parameters User Attributes Format Attributes

Attribute Definition

☐ Automatic ☒ Manual ☐ Dynamic

Name	Type	Width	Precision	Value	Index
▶ ParkId	smallint				
▶ RefParkId	smallint				
▶ ParkName	char	40			
▶ NeighborhoodName	char	40			
▶ EWStreet	char	30			
▶ NSStreet	char	30			
▶ DogPark	char	1			

+ - ▲ ▼ ↕

Filter

Reset Apply

In turn, carry out the following actions:

Delete Attribute	RefParkID	
Delete Attribute	EWStreet	
Delete Attribute	NSStreet	
Delete Attribute	DogPark	
Delete Attribute	Washrooms	
Delete Attribute	SpecialFeatures	
Rename Attribute	from: NeighborhoodName	to: Neighborhood
Add Attribute	ParkArea	type: Float
Add Attribute	AverageParkArea	type: Float

...and click the Parameter Editor "Apply" button. The attribute list should now look like this:

Table Parameters

Parameters User Attributes Format Attributes

Attribute Definition

☐ Automatic ☒ Manual ☐ Dynamic

Name	Type	Width	Precision	Value	Index
ParkId	smallint				
ParkName	char	40			
Neighborhood	char	40			
ParkArea	float				
AverageParkArea	float				

+ - ▲ ▼ ↕

Filter

Reset Apply

3) Rename Feature Type

Now click back on the Parameters tab.

Click in the field labelled Table Name (remember this label is format-specific and in MapInfo we deal with "tables") and change the name from Parks to ParksMaintenanceData.

Table Parameters

Parameters User Attributes Format Attributes

General

Table Name: ParksMaintenanceData

Writer: Training [MITAB]

☐ Dynamic Properties

Don't forget to click the Apply button to accept these changes.

Now when the workspace is run the output will be named ParksMaintenanceData.tab and will contain only the specified attributes.

4) Un-Expose Source Attributes

The workspace will now look like this:



TIP

Feature type objects can be resized (as in the above screenshot) if the feature type name or attribute names are too large to be displayed properly at the default size.

Notice there are several source attributes that are not going to be used in the workspace or sent to the output. We can tidy the workspace by hiding these.

Display the feature type properties for the reader (either the Properties dialog or in the Parameter Editor window) and click the User Attributes tab to open a list of the source attributes. It will look like this:

Exposed	Name	Type	Width	Precision	Index
<input checked="" type="checkbox"/>	ParkId	smallint			
<input checked="" type="checkbox"/>	RefParkId	smallint			
<input checked="" type="checkbox"/>	ParkName	char	40		
<input checked="" type="checkbox"/>	NeighborhoodName	char	40		
<input checked="" type="checkbox"/>	EWStreet	char	30		
<input checked="" type="checkbox"/>	NSStreet	char	30		
<input checked="" type="checkbox"/>	DogPark	char	1		
<input checked="" type="checkbox"/>	Washrooms	char	1		
<input checked="" type="checkbox"/>	SpecialFeatures	char	1		

Filter ☒ Select All

Uncheck the check box for the following attributes, which we do not need:

- RefParkID
- EWStreet
- NSStreet

- Washrooms
- SpecialFeatures

This is basically the list of attributes we deleted, except for DogParks, which we will make use of in the translation.

Click Apply/OK to confirm the changes.

5) Save the Workspace

Save the workspace – it will be completed in further examples. It should now look like this:



Police Chief Webb-Mapp says...

Some Writer attributes (ParkArea and AverageParkArea) have red connection arrows because there is nothing yet to map to them, while another (Neighborhood) is just unconnected.

That's OK. I'll let you off with a caution if you promise to connect them later. And you can still run this workspace just to see what the output looks like anyway.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Edit the attributes on a Writer schema*
- *Edit the output layer name on a Writer schema*
- *Hide attributes on a Reader schema*

Transformation with Transformers

Besides Schema Editing and Schema Mapping, transformation can be carried out using objects called **transformers**.

What is a Transformer?

As the name suggests, a transformer is an FME Workbench object that carries out transformation of features. There are lots of FME transformers, each of which carry out many different operations.

Transformers are connected somewhere between the Reader and Writer feature types, so that data flows from the Reader, through a transformation process, and on to the Writer.

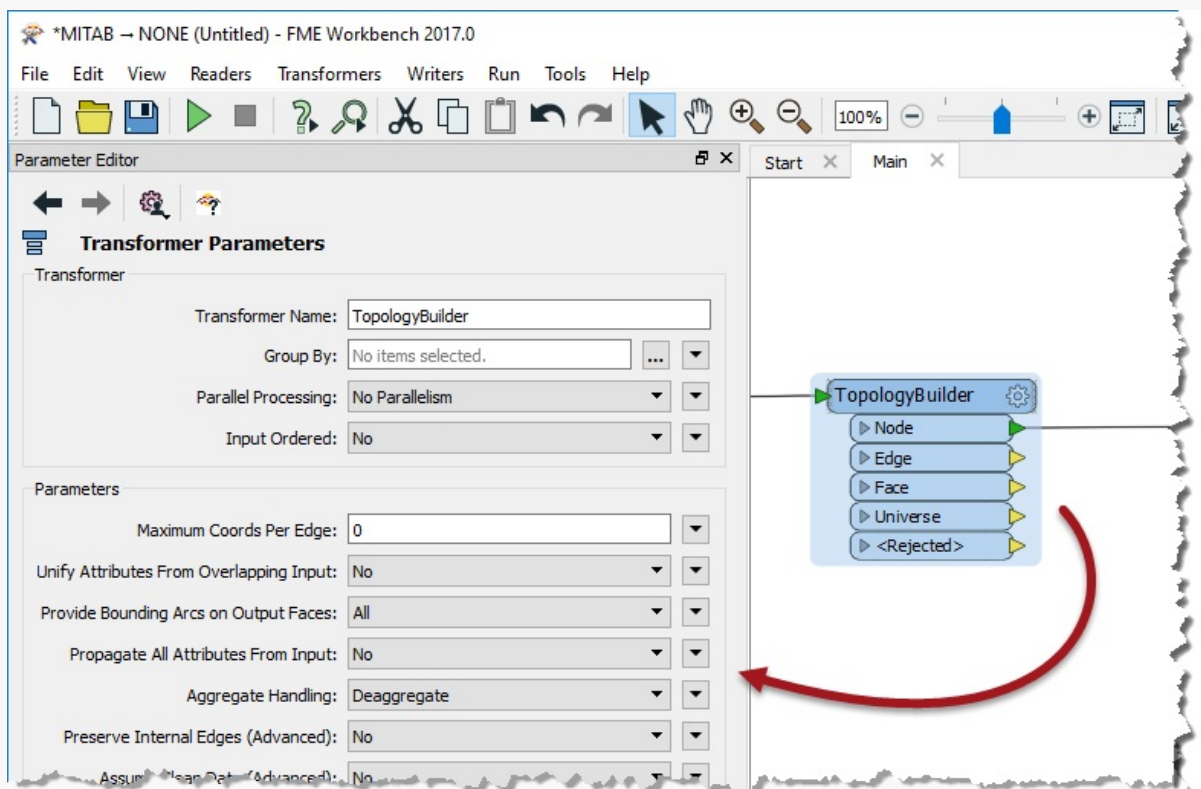
Transformers usually appear in the canvas window as rectangular, light-blue objects.

Transformer Parameters

Each transformer may have a number of parameters (settings). You can access the parameters in the Parameter Editor window by simply clicking on the transformer:

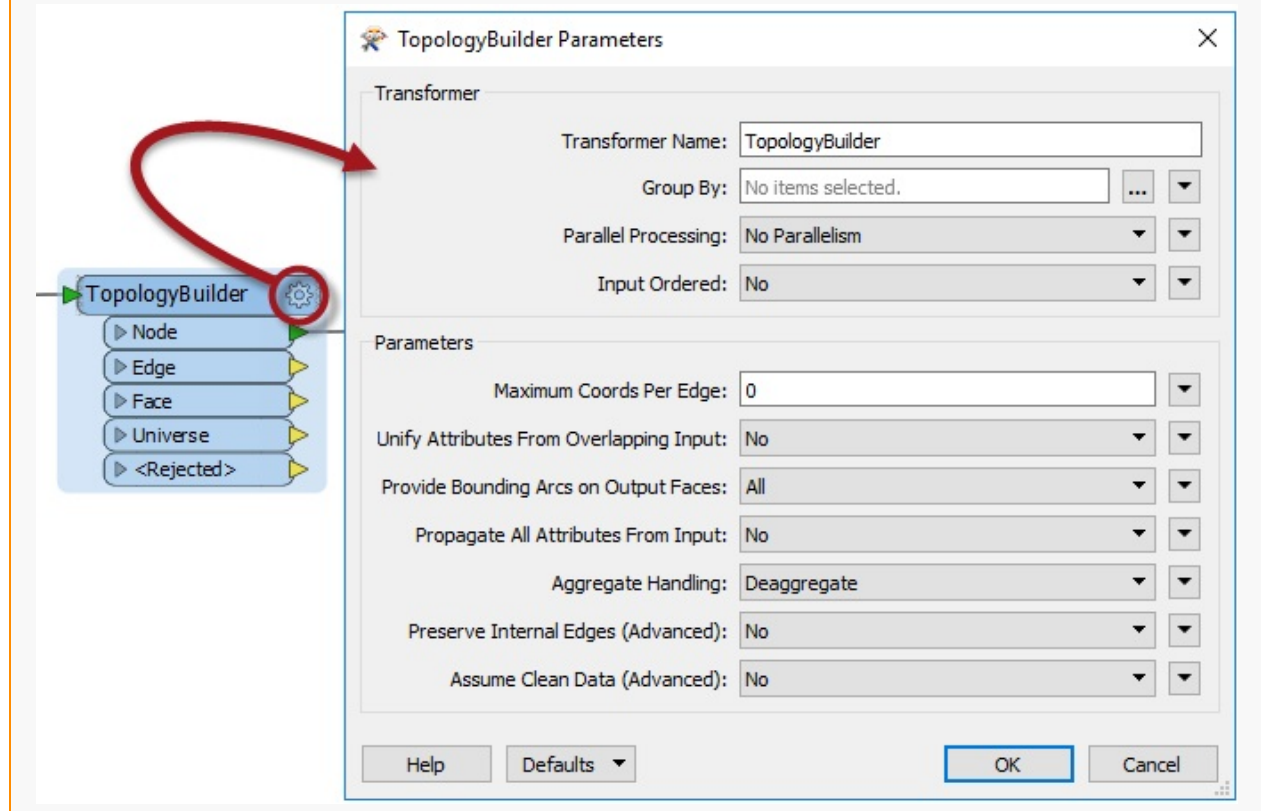
NEW

The Parameter Editor window is new for FME2017 and can be used to set parameters for any transformer.



Alternatively, you can open a separate dialog specifically for a transformer by

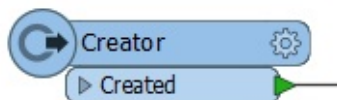
clicking the parameter button at the top right of it:



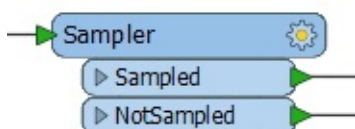
Color-Coded Parameter Buttons

The parameter button on a transformer is color-coded to reflect the status of the settings.

A blue parameter button indicates that the transformer parameters have been checked and amended as required, and that the transformer is ready to use.



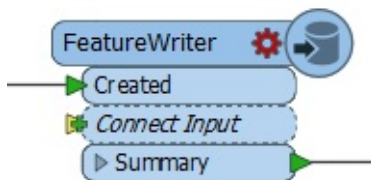
A yellow parameter button indicates that the default parameters have not yet been checked. The transformer can be used in this state, but the results may be unpredictable.



NEW

If the new Parameter Editor window is open - which it is by default - then you will rarely see a yellow icon. That's because if a transformer is touched or placed it's assumed you have seen and reviewed the parameters in the Parameter Editor window. If you are upgrading from an older version of FME, you will need to get into the habit of checking that window more often, to ensure you don't miss setting a parameter that you need.

A red parameter button indicates that there is at least one parameter for which FME cannot supply a default value. The parameter must be provided with a value before the transformer can be used.



First-Officer Transformer says...

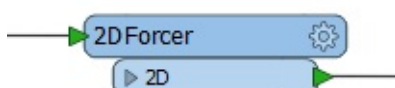
Good morning everyone, I'm First Officer Transformer and I'd like to welcome you aboard today's training.

Please be sure to check your parameters before your try to take off. Your workspaces just won't fly if there are any red-flagged transformers in them!

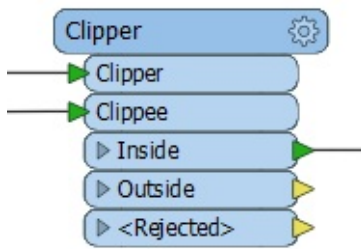
Transformer Ports

Far from having just a single input and/or output, a transformer can have multiple input ports, multiple output ports, or both.

This 2DForcer transformer has a single input and output port.



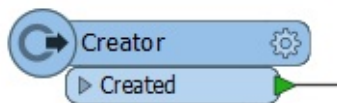
This Clipper has multiple input and output ports. Notice that not all of them are – or need to be – connected.



This Inspector has just a single input port...

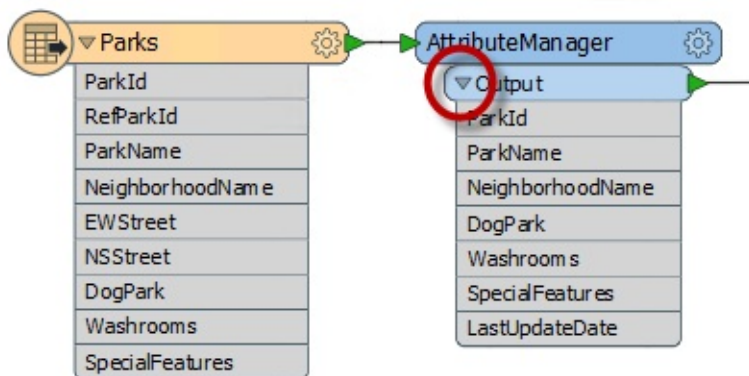


...whereas this Creator has only a single output port!



Transformer Attributes

Click on the drop-down arrow of a transformer output port to see all of the attributes that exit the transformer. This includes all changes applied within the transformer.



This is a good way to visualize which attributes have been created or lost within the transformer.

Exercise 2 Grounds Maintenance Project - Structural Transformation	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Structural Transformation with Transformers
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex2-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex2-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex2-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In case you forgot, 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.

In this part of the project we'll filter out dog parks from the source data (as these have a different scale of maintenance costs) and write them to the log window. We'll also handle the renamed attribute NeighborhoodName.

1) Start FME Workbench

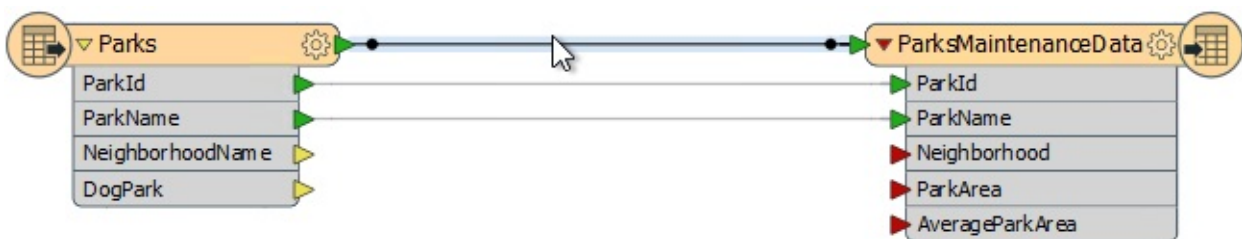
Start FME Workbench and open the workspace from the previous exercise. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex2-Begin.fmw.

2) Add Transformer

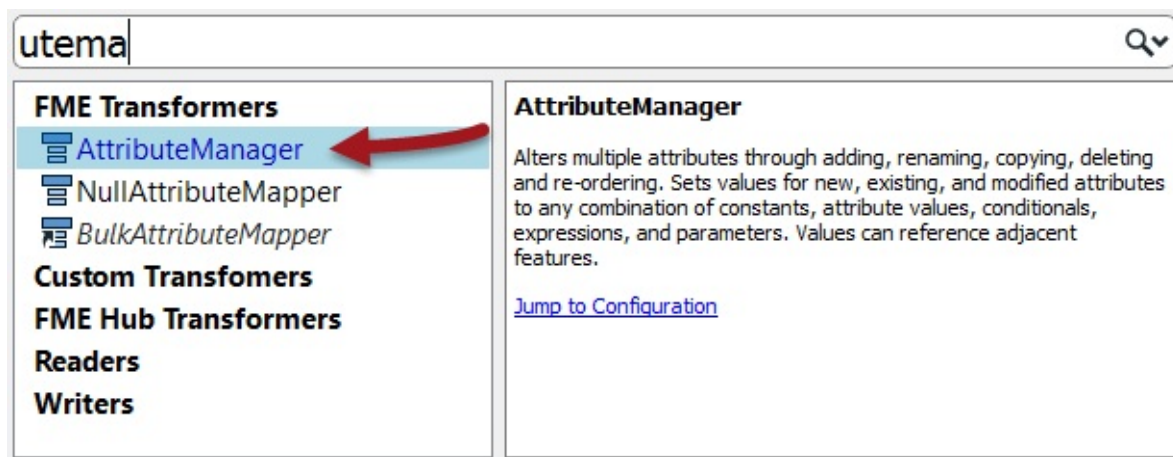
Let's first handle the source attribute NeighborhoodName, which was renamed Neighborhood on the writer but not yet connected.

We could do this by simply drawing a connection, but it's generally better to use a transformer. There are two transformers we could use. One is called the AttributeRenamer and the other - which we shall use here - is the AttributeManager.

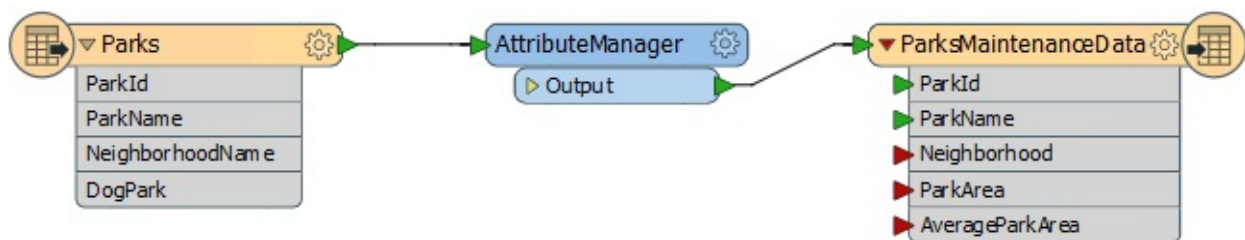
Therefore click on the feature connection from reader to writer feature type:



Start to type the phrase "AttributeManager". This is how we can place a transformer in the workspace. As you type, FME searches for a matching transformer. When the list is short enough for you to see the AttributeManager, select it from the dialog (double-click on it):



This will place an AttributeManager transformer like so:



.1 UPDATE

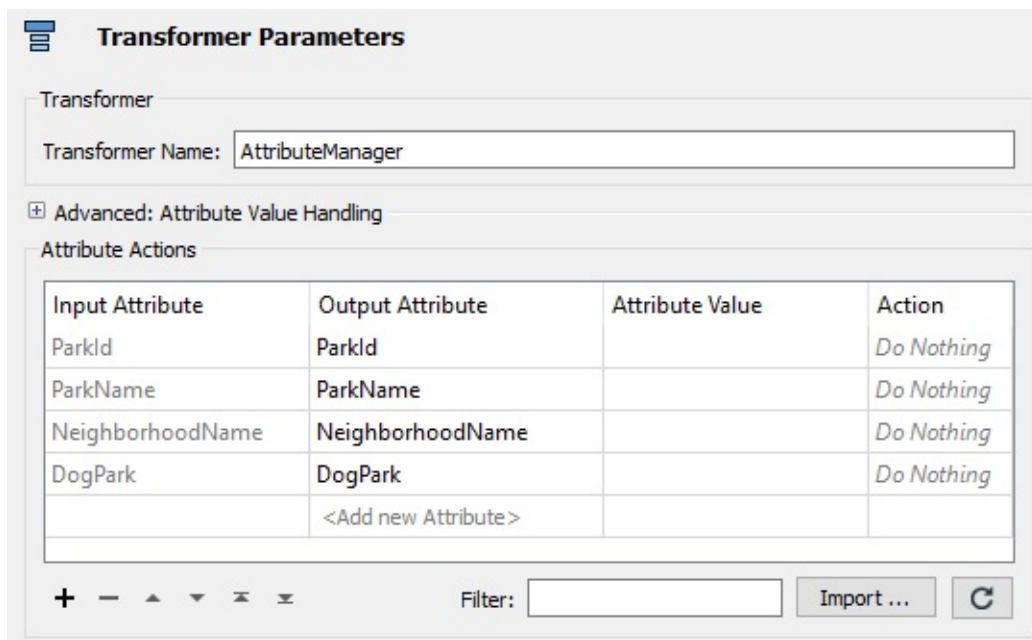
In 2017.1 the help contents of the Quick Add window are now simplified, but include a clickable link to the full documentation. It's also labelled 'Browse Additional Help' instead of 'Jump to Configuration'.

TIP

For a great tip on adding transformers to a workspace, see #5 in our list of [The Top Ten FME Tips of All Time!](#)

3) Set Parameters

View the AttributeManager parameters in the Parameter Editor window by clicking on it in the canvas (alternatively click its cogwheel icon to open the parameters dialog). It will look like this:



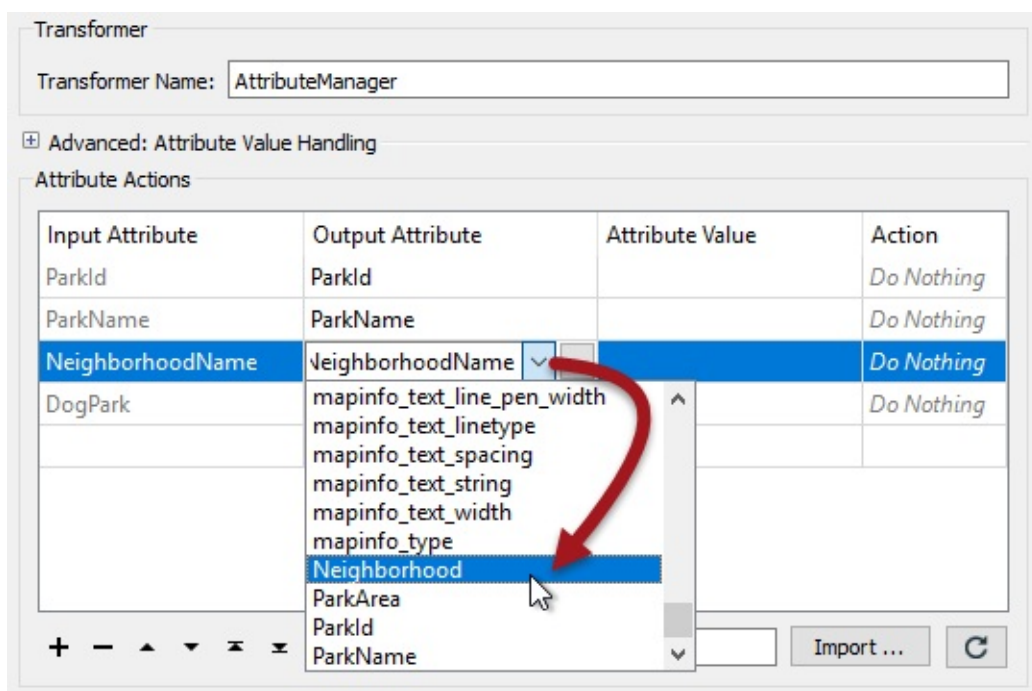
The image shows the 'Transformer Parameters' dialog box. The 'Transformer Name' field is set to 'AttributeManager'. The 'Advanced: Attribute Value Handling' section is expanded, showing the 'Attribute Actions' table. The table has four columns: 'Input Attribute', 'Output Attribute', 'Attribute Value', and 'Action'. The rows are:

Input Attribute	Output Attribute	Attribute Value	Action
ParkId	ParkId		Do Nothing
ParkName	ParkName		Do Nothing
NeighborhoodName	NeighborhoodName		Do Nothing
DogPark	DogPark		Do Nothing
	<Add new Attribute>		

At the bottom of the dialog, there are buttons for '+', '-', '▲', '▼', '↔', and '≡', a 'Filter' text box, an 'Import ...' button, and a circular refresh button.

Notice that all of the attributes on the stream in which it is connected will automatically appear in the dialog.

Where the Input Attribute field is set to NeighborhoodName, click in the Output Attribute field. Click on the button for the drop-down list and in there choose Neighborhood as the new attribute name to use.



The image shows the 'Transformer Parameters' dialog box with the 'Attribute Actions' table. The 'Input Attribute' field is set to 'NeighborhoodName'. The 'Output Attribute' field is set to 'NeighborhoodName'. A red arrow points to the dropdown button next to the 'Output Attribute' field. The dropdown menu is open, showing a list of attributes: 'mapinfo_text_line_pen_width', 'mapinfo_text_linetype', 'mapinfo_text_spacing', 'mapinfo_text_string', 'mapinfo_text_width', 'mapinfo_type', 'Neighborhood', 'ParkArea', 'ParkId', and 'ParkName'. The 'Neighborhood' attribute is highlighted.

Most importantly, check that the Action field is set to Rename and click the Apply (OK) button to confirm the action.

TIP

Neighborhood only appears in the list because it already exists on the writer schema. If we had done this step before editing the writer schema, we would have had to manually enter the new attribute name in this dialog.

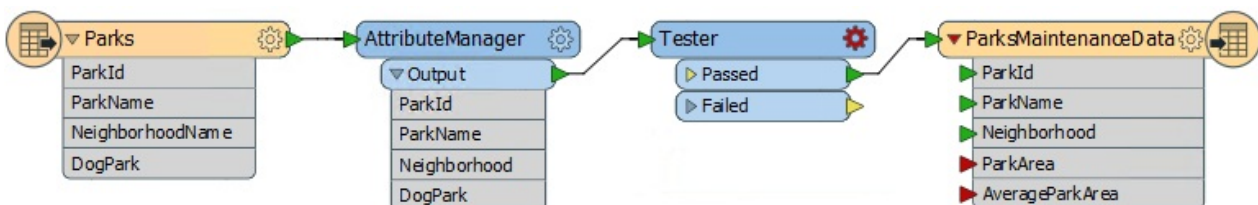
Click OK to close the dialog. Now in the Workbench canvas window you will see the Neighborhood attribute is flagged with a green arrow, to confirm that a value is being provided to that attribute.

**4) Add Transformer**

Now we should remove dog parks from the data, because these have their own set of costs.

This can be done with a Tester transformer. Click on the connection from the AttributeManager output port to the ParksMaintenanceData feature type on the Writer.

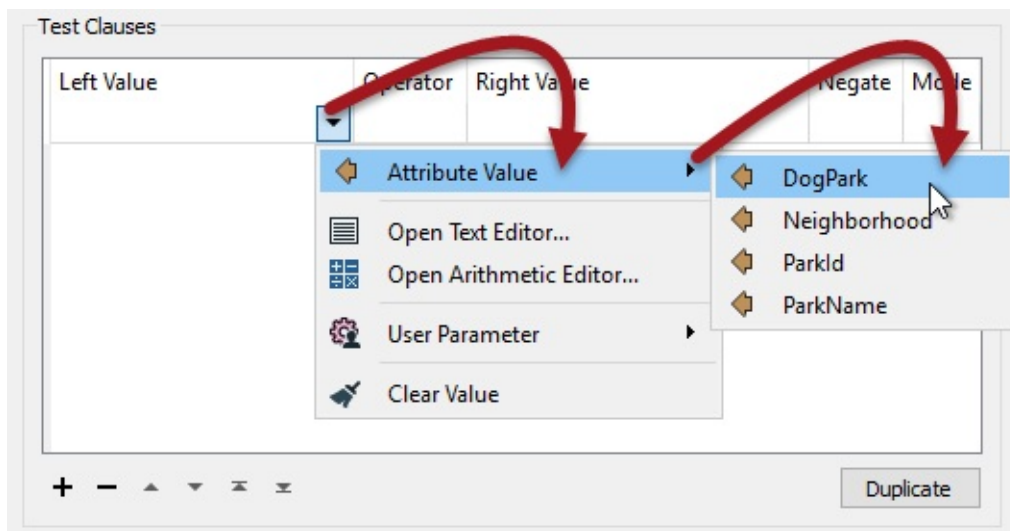
Start typing the word Tester. When you spot the Tester transformer double-click on it to add one to the workspace. After tidying up the layout of the canvas, the workspace will now look like this:



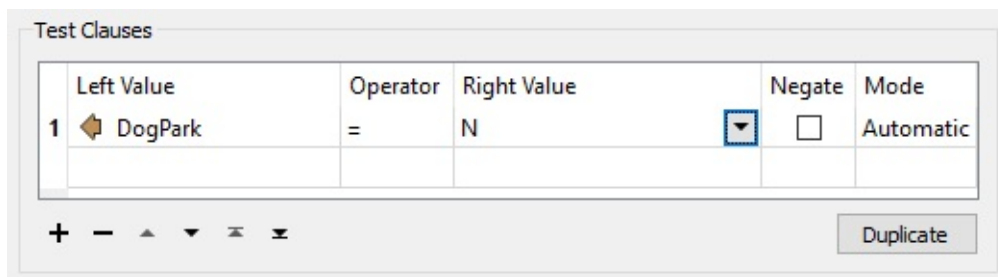
Notice that the Passed output port is the one connected by default.

5) Set Parameters

View the parameters on the Tester transformer by opening the parameters dialog (the Tester has a special dialog that can't be viewed in the Parameter Editor window). Double click in the Left Value field and from there click the down arrow and choose Attribute Value > DogPark.



For the Right Value click into the field and type the value N. The operator field should be filled in automatically as the equals sign (=), which is what we want in this case.



Click OK to accept the values and close the dialog.

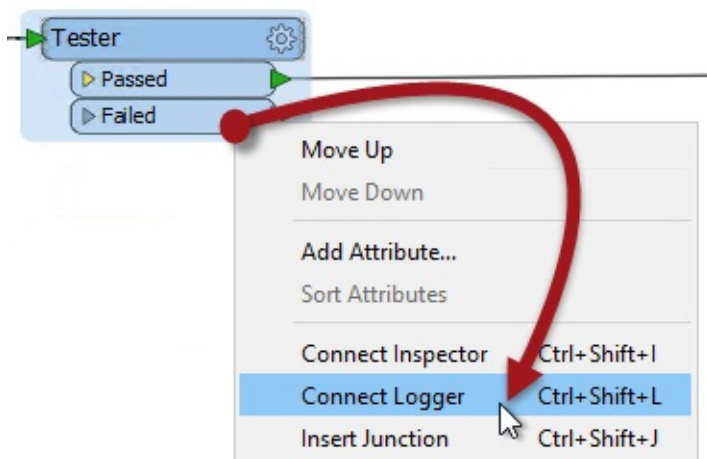
TIP

The test is for DogParks=N because we want to keep those features, and it is the Passed port that is connected. If the test was DogParks=Y then the Failed port would be the one to connect.

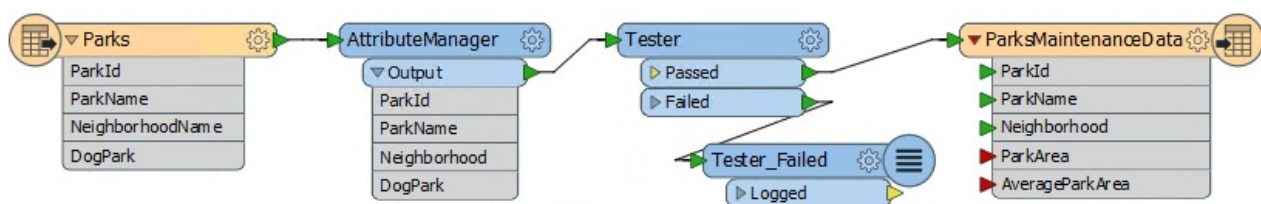
6) Add Transformer

We are now filtering out dog parks from our data, using a test on an attribute value. The question is, what should we do with this data we have filtered out. There are many things we could do, but for now we'll simply log the information to the FME log window.

To do this, right-click on the Tester Failed port and choose the option Connect Logger:



A Logger transformer will be added to the workspace. This will record all incoming features to the log window:



Loggers inserted by this method are named after – and reported in the log with – the output port they are connected to, here `Tester_Failed`.

7) Run Workspace

Save and run the workspace. It is not yet complete but running it will prove that everything is working correctly up to this point.

Advanced Exercise

As an advanced task, if you have time, filter the data further to remove parks that do not have a name; i.e. their name attribute is missing or empty. Would you need to place a second `Tester` transformer, or could you incorporate the test into the existing one?

CONGRATULATIONS

By completing this exercise you have learned how to:

- Add transformers to a workspace
- Carry out schema mapping with the `AttributeManager` transformer
- Filter data using the `Tester` transformer
- Record data using the `Logger` transformer

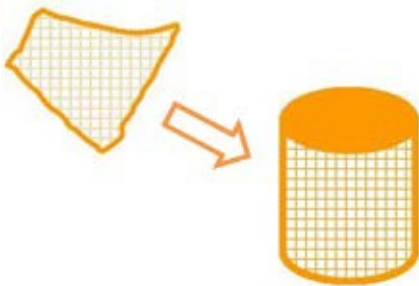
Content Transformation

Content transformations are those that operate on the geometry or attribute content of a dataset.

What is a Feature?

A feature in FME is an individual item within the translation. It is the fundamental (that is, smallest) unit of FME data.

Typically a GIS or cartographic feature consists of a geometric representation plus a set of related attributes. FME is capable of restructuring either of these components.



Sometimes content transformation operates on single features, sometimes on multiple features at once.

Features in FME have a flexible, generic representation that is unrelated to the format from which they originated. That means any transformer can operate on any FME feature, regardless of its source format.

Ms. Analyst says...

You can think of Content Transformation as altering or editing data.

The wardrobe analogy still works here. You might take your clothes from the wardrobe to clean them, or alter them, or repair them, or dye them a new color, or all sorts of other tasks, before returning them to their place.

The same holds true for spatial data transformation: it's the act of fixing up your data to be cleaner and in the style you really want

Geometric Transformation

Geometric Transformation is the act of restructuring the spatial component of an FME feature. In other words, the physical geometry of the feature undergoes some form of change to produce a different output.

Some examples of geometric transformation include the following:

- **Generalization** – a cartographic process that restructures data so it's easily visualized at a given map scale.
- **Warping** – adjustment of the size and shape of a set of features to more closely match a set of reference data.
- **Topology Computation** – conversion of a set of linear features into a node/line structure.



Line Intersection is another example of geometric transformation.

Here roads have been intersected with rivers to produce points that mark the location of bridges.

Attribute Transformation

Attribute Transformation is the act of restructuring the non-spatial component of an FME feature. In other words, the attributes relating to the physical geometry undergo some form of change to produce a different output.

Some examples of attribute transformation are:

- **Concatenation** – joining together of two or more attributes
- **Splitting** – splitting one attribute into many, which is the opposite of Concatenation
- **Measurement** – measuring a feature's length or area to create a new attribute
- **ID Creation** – creating a unique ID number for a particular feature

Attribute concatenation as an example of attribute transformation.

Each line of the address is concatenated to return a single line address.

```
Address1      Suite 2017,+
Address2      7445-132nd Street,+
City          Surrey,+
Province      British Columbia,+
PostalCode    V3W 1J8
```

```
= Address      Suite 2017, 7445-132nd Street,Surrey, British Columbia, V3W 1J8
```

Miss Vector says...

Did you miss me? You did? Well I'll cure that with some new questions for you!

Which three colours represent checked, need checking, and unset parameters on transformer objects?

- 1. blue, yellow, red*
- 2. green, yellow, red*
- 3. red, green, blue*
- 4. green, blue, yellow*

If I use a transformer to remove irregularities (like self-intersecting loops) in the boundary of a polygon, what type of transformation is it?

- 1. Structural Transformation of attributes*
 - 2. Structural Transformation of geometry*
 - 3. Content Transformation of attributes*
 - 4. Content Transformation of geometry*
-

Transformers used in Series

Much like a set of components in an electrical circuit, a series of Workbench transformers can be connected together to have a cumulative effect on a set of data.

Chaining Transformers

Even with the large number of transformers available in FME, users frequently need a combination or chain of transformers instead of a single one.

A string of transformers that graphically represent an overall workflow is a key concept of FME.

In this example, a DuplicateFilter transformer removes duplicate polygon features. A Dissolver transformer merges each remaining (unique) polygon with its neighbor where there is a common boundary. Finally each merged area gains an ID number from the Counter transformer.



Exercise 3 Grounds Maintenance Project - Calculating Statistics	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Content Transformation. Schema Mapping
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex3-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex3-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex3-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In case you forgot, 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.

In this part of the project we'll calculate the size and average size of each park, and ensure that information is correctly mapped to the destination schema.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex3-Begin.fmw

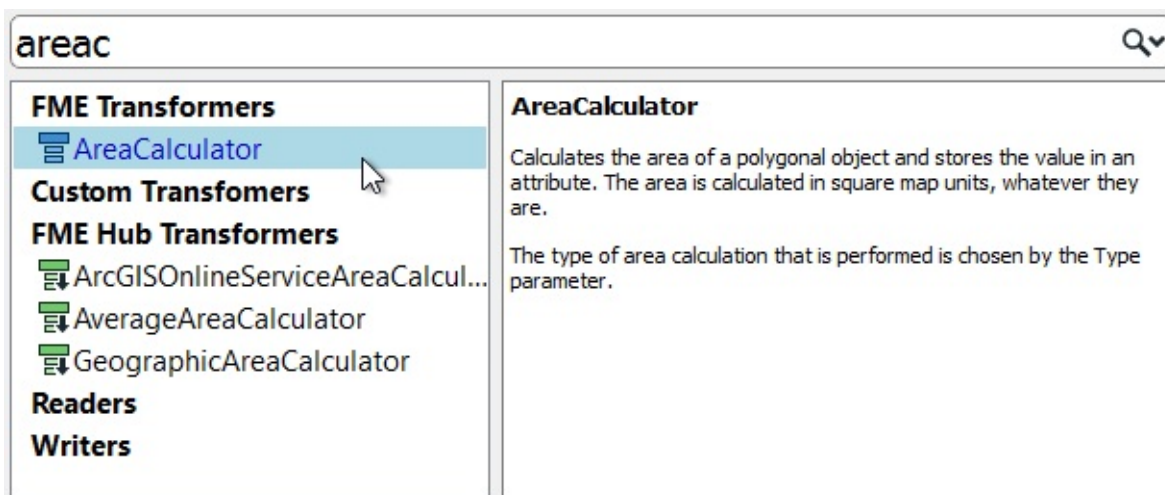
2) Add an AreaCalculator Transformer

To measure the area of each park feature, an AreaCalculator transformer must be used.

"Calculator" is the term for when FME computes new attribute values.

Click onto the connection between Tester:Passed port and the writer feature type *ParksMaintenanceData*. Start typing the letters "areac". You will see the Quick Add list of matching transformers appear beneath.

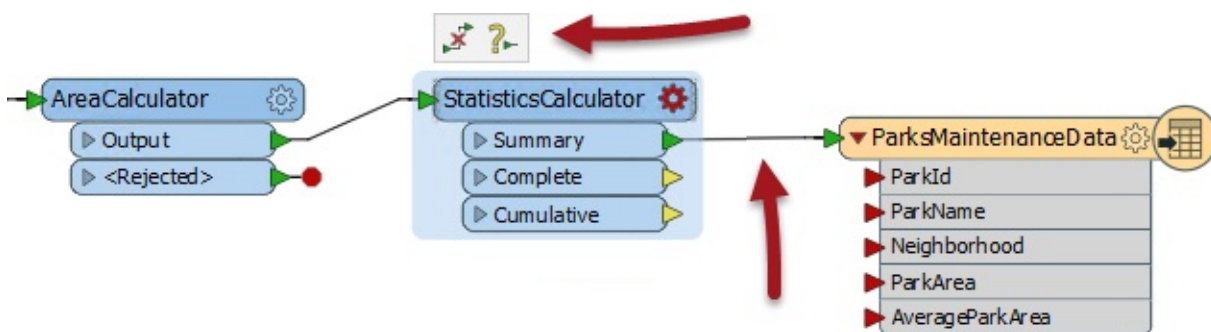
Select the transformer named AreaCalculator by double-clicking it:



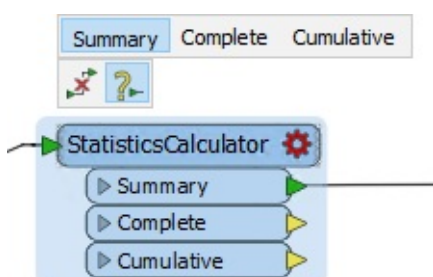
3) Add a StatisticsCalculator Transformer

Using the same method, place a StatisticsCalculator transformer between the AreaCalculator:Output port and the *ParksMaintenanceData* feature type.

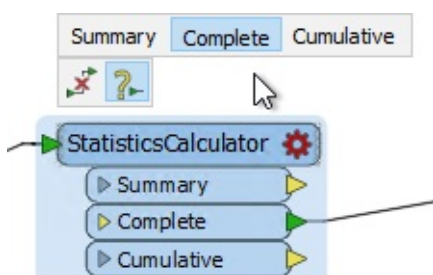
BUT! Do not click anything else yet! The transformer will now look like this:



By default the Summary port has been connected, and we need the Complete port connected instead. But notice the little pop-up icons over the top. Click the right-hand icon (the one with the ? character). This pops up a further list of ports:



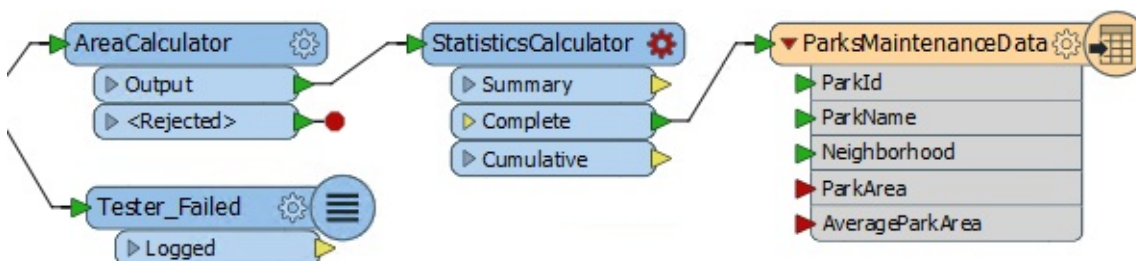
Click on the Summary port entry to disconnect that, and then on the Complete port entry to connect that:



TIP

These pop-up menus are a great help in schema mapping and other feature connections.

The latter part of the workspace now looks like this:



4) Check AreaCalculator Settings

View the AreaCalculator parameters in the Parameter Editor window by clicking on it in the canvas (alternatively click its cogwheel icon to open the parameters dialog). It will look like this:

Parameter Editor

Transformer Parameters

Transformer

Transformer Name: AreaCalculator

Parameters

Type: Plane Area

Area Attribute: _area

Multiplier: 1

The default settings cause the calculated value to be placed into an attribute called `_area`. However, the *ParksMaintenanceData* schema requires an attribute called `ParkArea`, so change this parameter to create the correct attribute.

Parameters

Type: Plane Area

Area Attribute: ParkArea

Multiplier: 1

Reset Apply

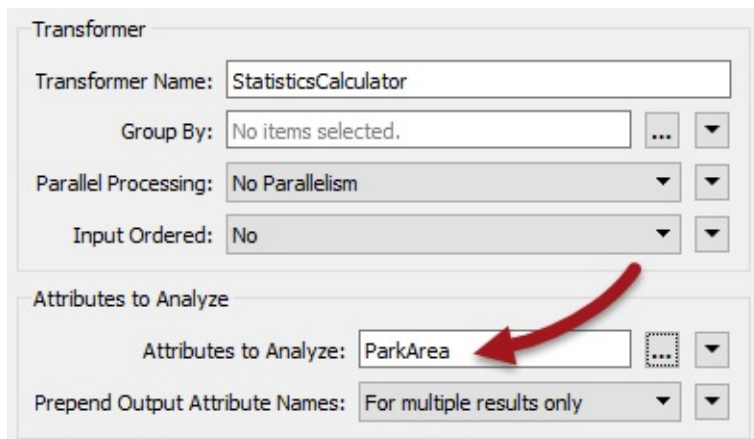
Once you click the Apply button, notice that the attribute on the writer feature type is now flagged as connected.

5) Check StatisticsCalculator Settings

A red icon indicates the StatisticsCalculator has parameters that need to be defined.

View the StatisticsCalculator transformer's parameters. Again you may use either the Parameter Editor window or the transformer's own Parameters dialog.

The attribute to analyze is the one containing the calculated area; so select ParkArea.



Transformer

Transformer Name: StatisticsCalculator

Group By: No items selected. ... ▼

Parallel Processing: No Parallelism ▼ ▼

Input Ordered: No ▼ ▼

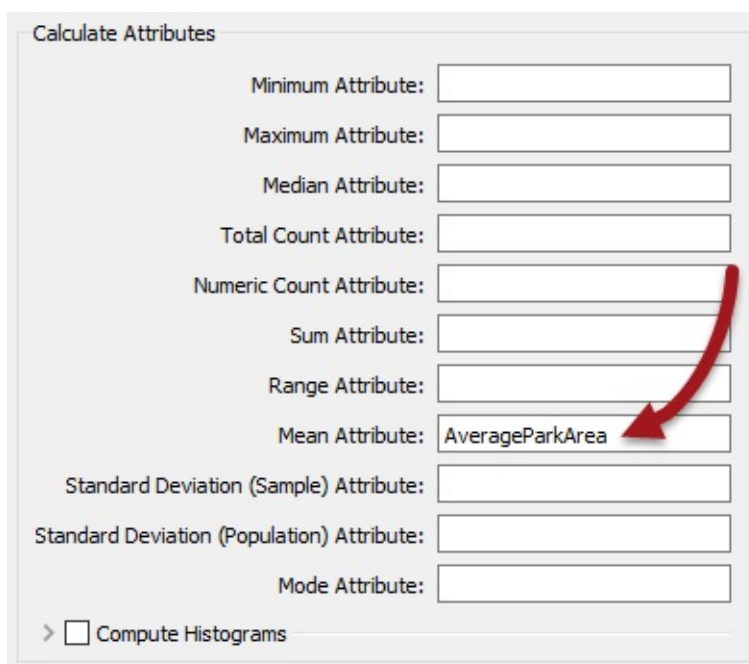
Attributes to Analyze

Attributes to Analyze: ParkArea ... ▼

Prepend Output Attribute Names: For multiple results only ▼ ▼

Examine what the default setting is for an attribute name for average (mean) park size. Currently it doesn't match the *ParksMaintenanceData* schema, which requires an attribute named AverageParkArea.

Change the attribute from `_mean` to AverageParkArea. For Best Practice reasons, delete/unset any StatisticsCalculator output attributes that aren't required (for example `_range` and `_stdev`).



Calculate Attributes

Minimum Attribute:

Maximum Attribute:

Median Attribute:

Total Count Attribute:

Numeric Count Attribute:

Sum Attribute:

Range Attribute:

Mean Attribute: AverageParkArea

Standard Deviation (Sample) Attribute:

Standard Deviation (Population) Attribute:

Mode Attribute:

> ☐ Compute Histograms

Finally, click Apply/OK to accept the changes.

6) Run the Workspace

Run the workspace.

Inspect the result of the translation using the FME Data Inspector.

Table View

Table: ParksMaintenanceData [MITAB] - ParksMaintenanceData Columns...

	ParkId	ParkName	Neighborhood	ParkArea	AverageParkArea
1	1	<missing>	Kitsilano	448.124680666006	70248.2733912836
2	2	Rosemary Brow...	Kitsilano	1035.07708082504	70248.2733912836
3	3	Tea Swamp Park	Mount Pleasant	2631.26398618223	70248.2733912836
4	4	<missing>	Strathcona	1984.83635717903	70248.2733912836
5	5	Morton Park	West End	2197.31819965096	70248.2733912836
6	6	Mcbride Park	Kitsilano	17125.7170839886	70248.2733912836
7	7	Granville Park	Fairview	19655.7053629716	70248.2733912836
8	8	<missing>	Mount Pleasant	3123.72307219952	70248.2733912836
9	9	Creekside Park	Mount Pleasant	23975.705264418	70248.2733912836

73 row(s)

Inspect the Table View window to discover the area of each park and the average area of all parks.

7) Save the Workspace

Save the workspace – it will be completed in further examples.

Advanced Exercise

Notice that the numbers in the Table View show the results have been calculated to 12 decimal places. This is in excess of the precision that you require. As an advanced task - if you have time - use the AttributeRounder transformer to reduce the values to just 2 decimal places.

If you wish, you can also calculate the smallest, largest, and total park areas; but don't forget to add them to the writer schema if you want them to appear in the output.

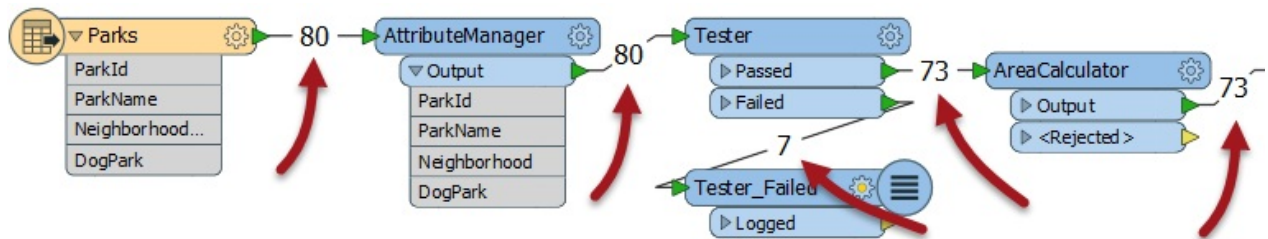
CONGRATULATIONS

By completing this exercise you have learned how to:

- *Carry out content transformation with transformers (AreaCalculator, StatisticsCalculator)*
- *Manage transformer connections using pop-up buttons*
- *Use transformer parameters to create attributes that match the writer schema*

Feature Count Display

Look back at the previous example's translation. Did you notice that while the workspace was running, each connection was updated with the number of features that had passed along it?



The final feature counts show that 80 features were read, 73 passed the Tester while 7 failed (for being dog parks) and went on to the Logger.

The Log window confirms the number of features written and lists the features that failed the Tester.

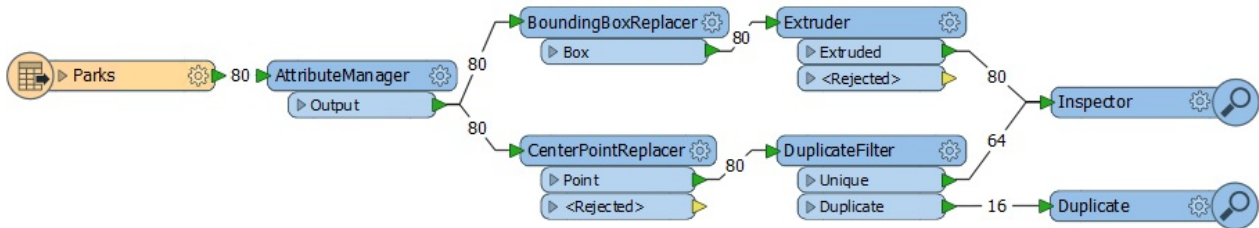
This number helps analyze the results of a workspace and provides a reference for debugging if the destination data differs from what was expected.

Transformers used in Parallel

FME Workbench permits multiple data streams, each of which passes through its own set of transformers.

Multiple Streams

A key concept in FME is the ability to have multiple processing streams within a workflow:



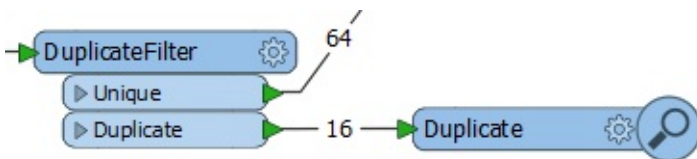
As above, a workspace author can split one stream into several, or combine several streams of data into one, as required.

TIP

Similar terms to 'stream' that people use are 'flows,' 'pipes,' or 'pipelines.'

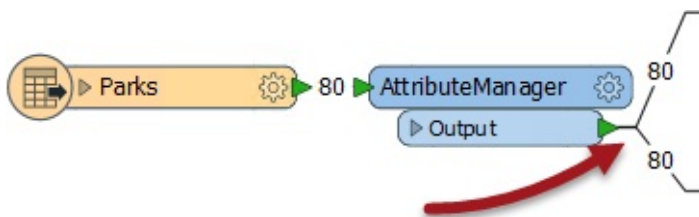
Creating Multiple Streams

Splitting data can occur in a number of ways. Sometimes a transformer with multiple output ports (a Tester transformer is a good illustration of this) will divide data into several output streams, like so:



Additionally, a full stream of data can be duplicated by simply making multiple connections out of a single output port. In effect it creates a set of data for each connection.

Here, 80 Parks features are being read but, because there are multiple connections from the feature type, multiple copies of the data are being created.

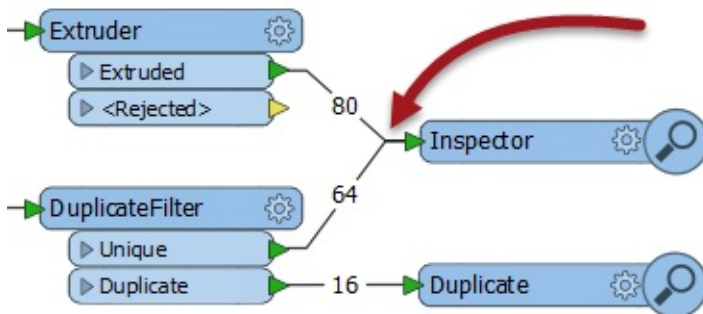


Multiple streams are useful when a user needs to process the same data, but in a number of different ways.

Bringing Together Multiple Streams

When multiple streams are brought into the same input port no merging takes place. The data is simply accumulated into a single stream. This is often called a "Union".

Here, two streams of data - 80 features in one, 64 features in the other - converge into a single Inspector:



No merging has taken place; the data simply accumulates into 144 distinct output features. To carry out actual merging of data requires a specific transformer such as the FeatureMerger.

Exercise 4 Grounds Maintenance Project - Labelling Features	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Content transformation with parallel transformers
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex4-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex4-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex4-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In this part of the project we'll create a label for each park and write it to a new output layer. This is best done using a parallel stream of data.

1) Start Workbench

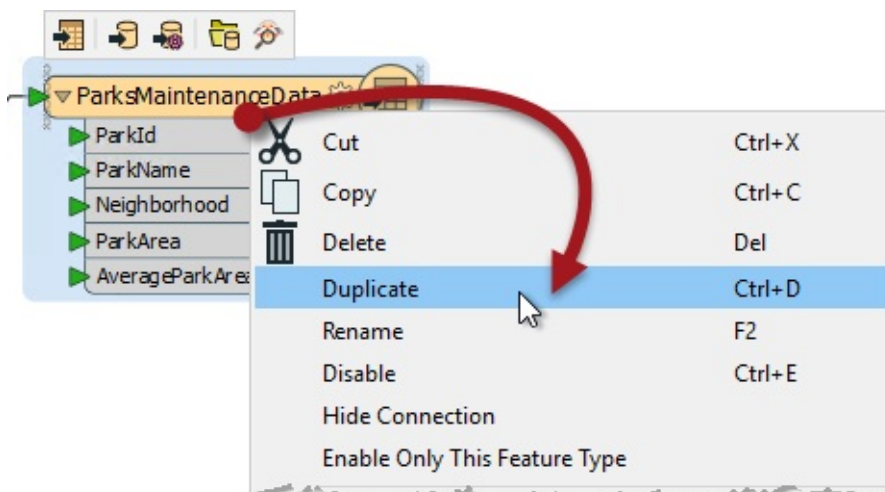
Start Workbench (if necessary) and open the workspace from Exercise 3. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex4-Begin.fmw

The previous exercise measured park areas with the AreaCalculator. Now we are asked to add this information as labels to the output dataset.

This can be achieved using the LabelPointReplacer transformer.

2) Create New Writer Feature Type

Because we want to write label features to a separate layer (table) in the output, we need to create another feature type object on the canvas. There is more about this in a later chapter, but for now right-click the writer feature type and choose the option Duplicate. This creates a new feature type (layer) in the output dataset.

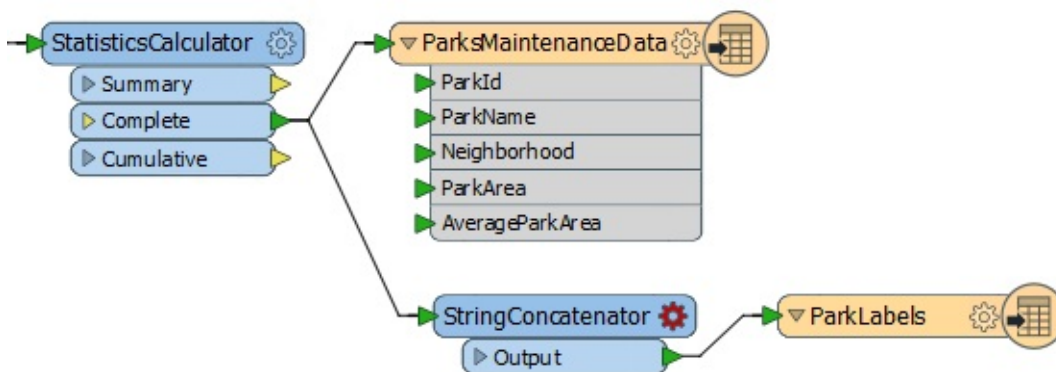


Now clean up this feature type's schema. View the feature type's properties and rename the new type to ParkLabels. In the User Attributes tab delete all of the existing user attributes.

3) Place a StringConcatenator Transformer

Click onto a blank area of canvas. Type "StringConcatenator" to add a transformer of this type.

Connect it to the Complete port of the StatisticsCalculator by dragging a second connection from there to the new transformer.



Make a new connection from the StringConcatenator to the new feature type.

4) Check Transformer Parameters

View the parameters for the StringConcatenator transformer. There are both basic and advanced dialogs, and the basic one looks like this:

Transformer

Transformer Name: StringConcatenator

Parameters

Expression Results: Create New Attribute

New Attribute: _result

Attributes To Overwrite: No items selected.

String Parts

String Type	String Value

+ - ▲ ▼ ↵ ✕

Concatenated Result

Switch To Advanced

Enter *LabelText* as the name for the new attribute to create.

In the String Parts section, set the following four parts:

String Type	String Value
Attribute Value	ParkName
New Line	
Attribute Value	ParkArea
Constant	sq metres

Be sure to include a space character in the constant before "sq metres".

The screenshot shows the 'Expression Builder' dialog in QGIS. The 'Parameters' section at the top has 'Expression Results' set to 'Create New Attribute', 'New Attribute' set to 'LabelText', and 'Attributes To Overwrite' set to 'No items selected.'. Below this is the 'String Parts' section, which contains a table with four rows: 'Attribute Value' pointing to 'ParkName', 'New Line', 'Attribute Value' pointing to 'ParkArea', and 'Constant' with the text 'sq metres'. At the bottom of the dialog, the 'Concatenated Result' preview shows the expression: `@Value(ParkName)`
`@Value(ParkArea) sq metres`. A 'Switch To Advanced' button is located at the bottom right of the dialog.

TIP

You may find it quicker to switch to the Advanced editor dialog and enter the content directly:

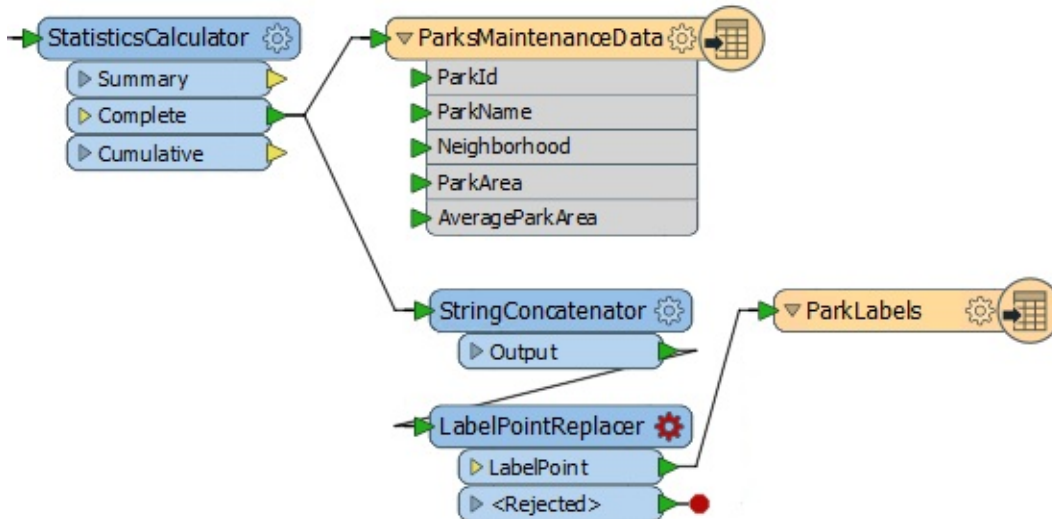
`@Value(ParkName)`

`@Value(ParkArea) sq metres`

5) Place a LabelPointReplacer Transformer

Click onto the connection between StringConcatenator:Output and the ParkLabels feature type. Type "LabelPointReplacer" to add a transformer of this type.

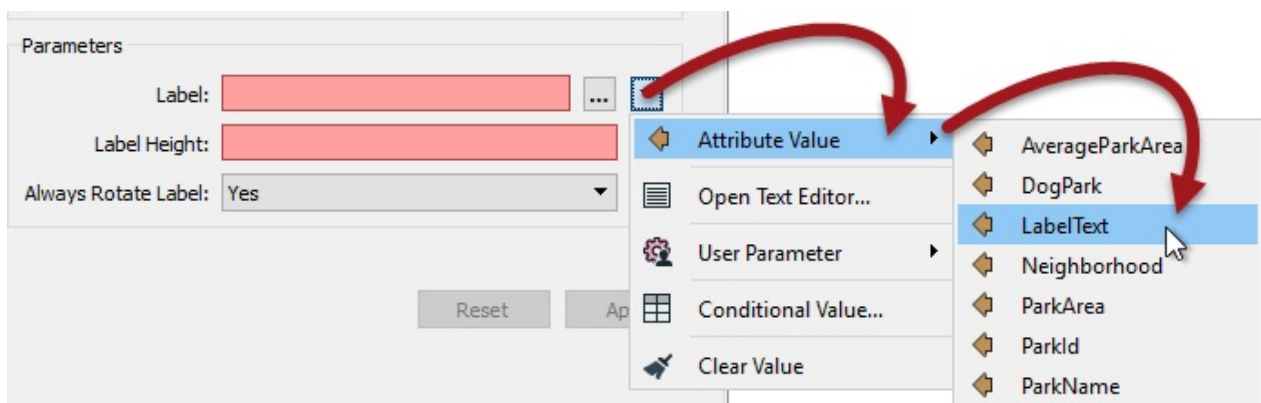
The new transformer will be added and automatically connected between those two objects.



6) Check LabelPointReplacer Parameters

View the LabelPointReplacer parameters.

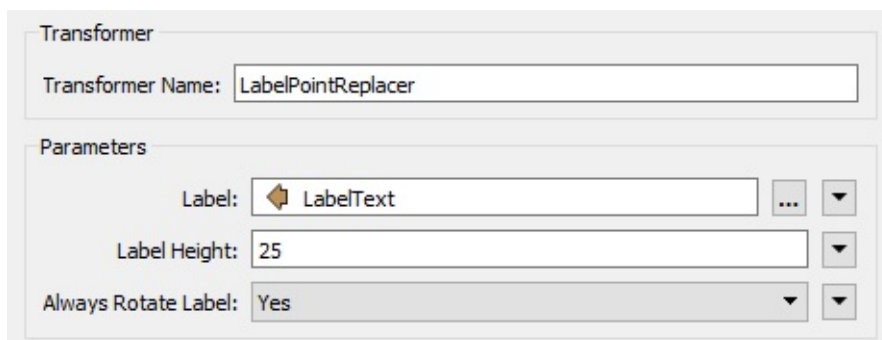
Firstly click the dropdown arrow to the right of the Label parameter:



Select Attribute Value > LabelText to select the label previously defined in the StringConcatenator.

Now click in the Label Height field and type 25 (that's 25 working units, which in this case is metres).

The “Always Rotate Label” parameter can be left to its default value.



TIP

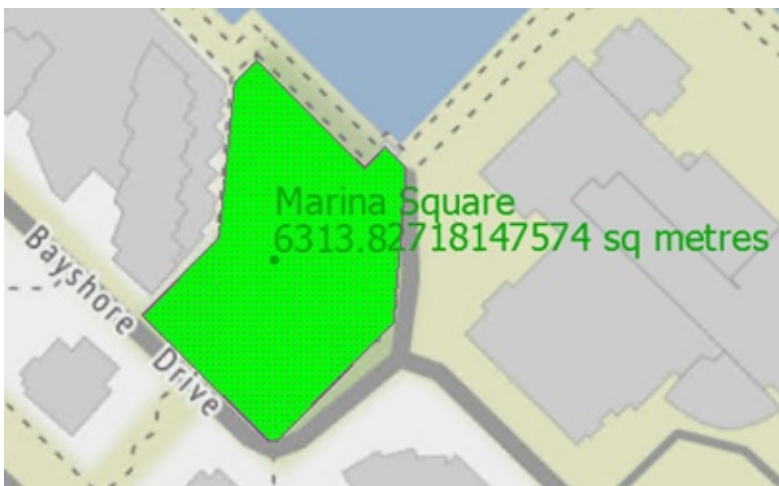
Many parameter fields (like Label Height) can be set either as a constant value (by typing it in) or set to an attribute by using “Set to Attribute Value.”

And - as you'll see shortly - it's also possible to construct a parameter value directly inside the transformer settings

7) Run the Translation

Run the translation and inspect the output.

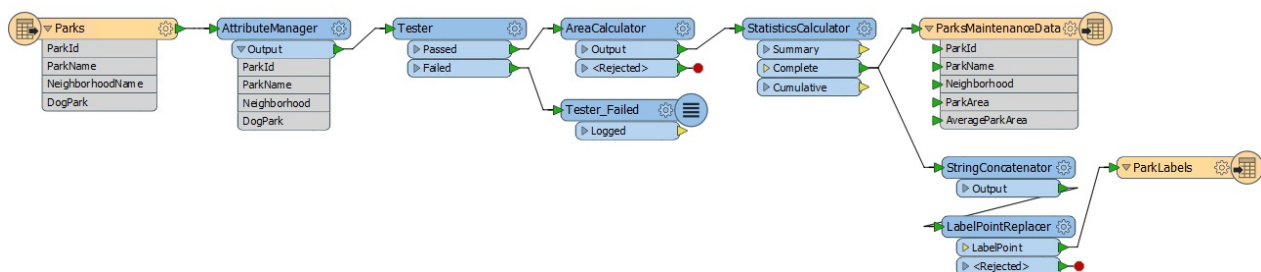
Notice that the output is in two layers in two files. Use the FME Data Inspector to open both output files in the same view.



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

6) Save the Workspace

Save the workspace – it will be completed in further examples.



Advanced Exercise

Now you know how to create a new feature type (layer) in the output, how to test data, and how to use parallel streams, why not try this task:

Identify which parks are smaller than average and which parks are larger than average, and write them out to different feature types.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Create a new writer feature type*
- *Use multiple streams of transformers in a single workspace*
- *Use the StringConcatenator to construct a string for use elsewhere*
- *Use an attribute as the value of a transformer's parameter*

Group-By Processing

Group-By parameters are an important tool for effective FME data transformations.

What is a Group?

FME transformers carry out transformations on either one feature at a time, or on a whole set of features at once.

For example, the *AreaCalculator* transformer operates on one feature at a time (to measure the area of that one polygon feature). We call it a **feature-based transformer**.

The *StatisticsCalculator* operates on multiple features at a time (to calculate an average value for them all). In FME we call this set of features a **group** and the transformer is a **group-based transformer**.

Creating Groups

By default a group-based transformer treats ALL the features that it is fed as a single group.

However, such transformers also have a **Group-By** parameter. This parameter lets the user define several groups based upon the value of an attribute.

Mr. Statistics-Calculator, CFO says...

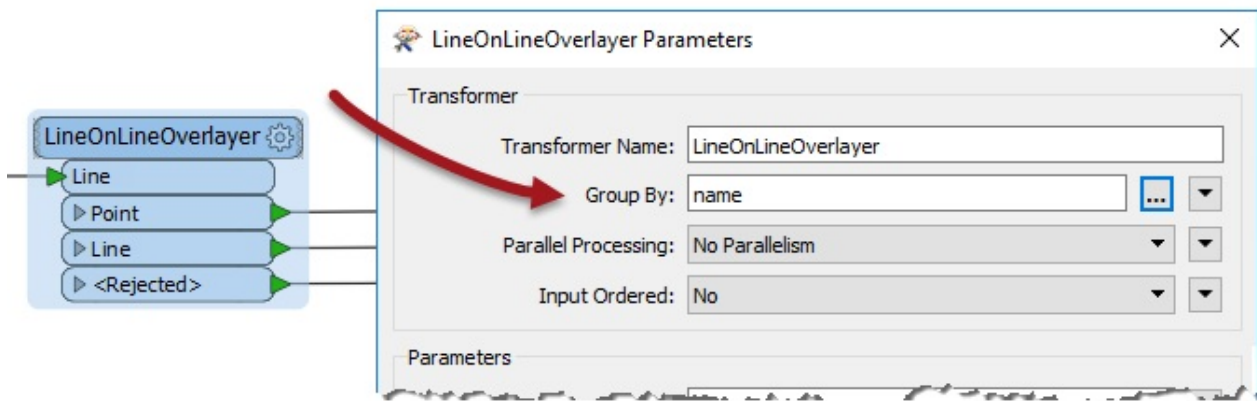
Hi. I don't think we've met yet. I'm Mr. Statistics-Calculator. I bet you can't guess my favourite transformer!

To illustrate groups let's consider calculating the mean age of FME users. Don't worry - I'll be discrete (ha ha)! The default group for the calculation includes ALL FME users.

But you could instead divide everyone up on into men and women, creating two groups, and calculate average age per gender. Or you could divide everyone into their nationality, and calculate average age per country.

This is the same as having a gender (or nationality) attribute in a dataset and selecting that in an FME group-by parameter.

Here, a *LineOnLineOverlayer* transformer is being used to intersect a number of line features. The selected Group-By attribute is name:



The result is a series of groups for overlaying where the features in each group share the same value for name. The practical outcome is that intersection will only take place on line features with the same name.

Miss Vector says...

Let's see if you've picked up the idea of what a group-based transformation is.

Which of the following transformers do you think is "group-based"? Feel free to use Workbench to help you answer this question.

1. *StringFormatter*
2. *Clipper*
3. *Rotator*
4. *AttributeRounder*

Exercise 5 Grounds Maintenance Project - Neighborhood Averages	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Group-By Processing
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex5-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex5-Complete.fmw

Let's continue your work on the grounds maintenance project.

The parks team has decided that they do not want the average area of park for the city as a whole. Instead they want the average size of park in each neighborhood; so let's do that for them.

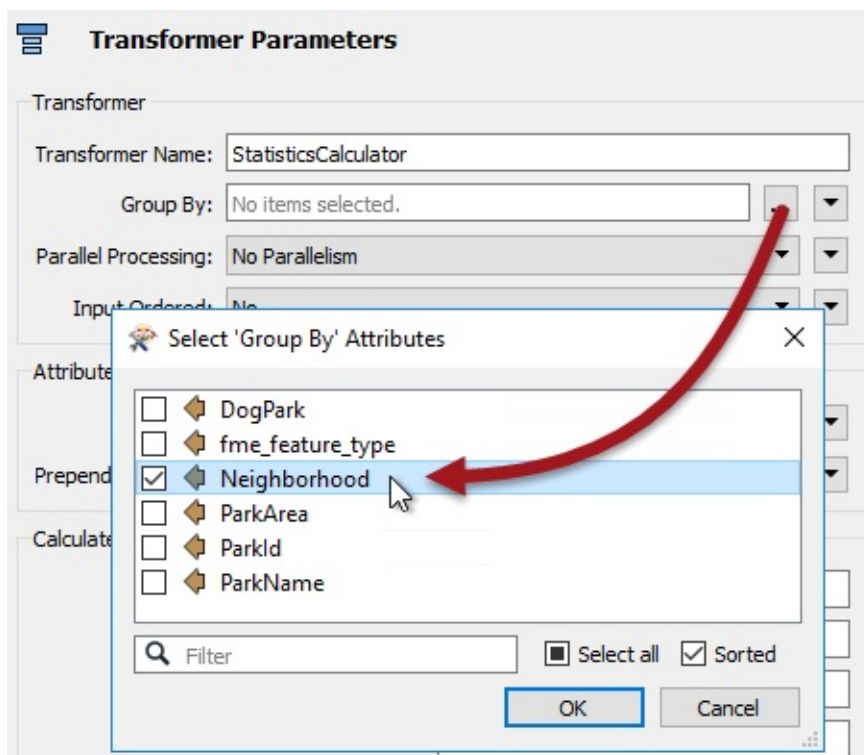
1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 4. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex5-Begin.fmw

2) Set Group-By in StatisticsCalculator

This is a really simple task to do. View the parameters for the StatisticsCalculator transformer and click the 'browse' button next to the Group By parameter.

Select the attribute called Neighborhood:



Click OK and apply the changes to the transformer.

3) Run the Workspace

Save and then run the workspace.

Inspect the output data in the Table View window of the FME Data Inspector.

You should see that each neighborhood now has its own value for AverageParkArea:

Table: ParksMaintenanceData [MITAB] - ParksMaintenanceData Columns...

	ParkId	ParkName	Neighborhood	ParkArea	AverageParkArea
1	1	<missing>	Kitsilano	448.124680666006	23986.3438250703
2	2	Rosemary Brow...	Kitsilano	1035.07708082504	23986.3438250703
3	3	Tea Swamp Park	Mount Pleasant	2631.26398618223	11660.2527620148
4	4	<missing>	Strathcona	1984.83635717903	10196.9647106941
5	5	Morton Park	West End	2197.31819965006	300747.867748344
6	6	Mcbride Park	Kitsilano	17125.7170839806	23986.3438250703
7	7	Granville Park	Fairview	19655.7053629706	14973.3037653848
8	8	<missing>	Mount Pleasant	3123.72307219902	11660.2527620148
9	9	Creekside Park	Mount Pleasant	23975.705264408	11660.2527620148
10	10	China Creek So...	Mount Pleasant	14750.3802947044	11660.2527620148
11	11	Barclay Heritag...	West End	5647.90222303407	300747.867748344

73 row(s)

CONGRATULATIONS

By completing this exercise you have learned how to:

- Use the group-by parameter in FME transformers

Data Inspection Using FME Workbench

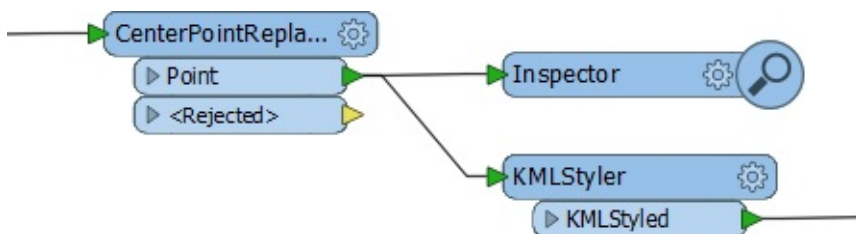
Besides Redirect to FME Data Inspector, Workbench can route data to the FME Data Inspector from individual transformers.

Using an Inspector Transformer

An **Inspector** is a Workbench transformer – with its own distinctive look and style – that causes data entering it to be directed to FME Data Inspector.

An Inspector transformer differs from the Redirect to FME Data Inspector setting because the transformer can be applied at any point in a translation (not just at the very end) and does not prevent the data being output to the writer. It also lets a user be more selective about which features should be inspected.

Here data is being directed to the Inspector after a CenterPointReplacer transformer, but before a KMLStyler.

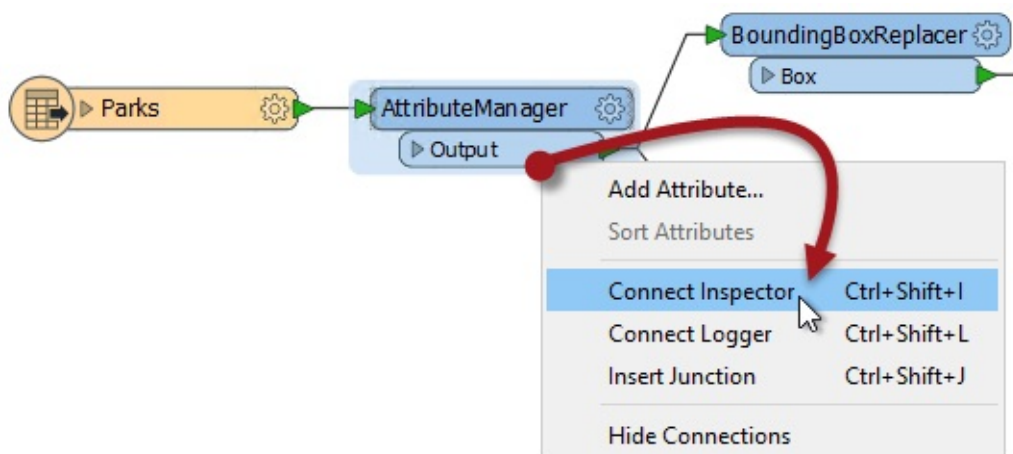


As you can see, it's also a form of parallel streams, since the data is duplicated to both Inspector and KMLStyler.

Placing an Inspector Transformer

The best, and simplest way to apply an Inspector is to right-click the output port of an object in a workspace and select the Connect Inspector option.

Here the user right-clicks the Output port of an AttributeManager and chooses the option Connect Inspector:



Notice that an Inspector is named automatically using the transformer and output port names. Here it will be "AttributeManager_Output". This helps to identify the data from this Inspector (as opposed to any others) in the FME Data Inspector.

Note that the Inspector transformer only opens the FME Data Inspector when there are features to view. If there are zero features, then the Data Inspector will not open!

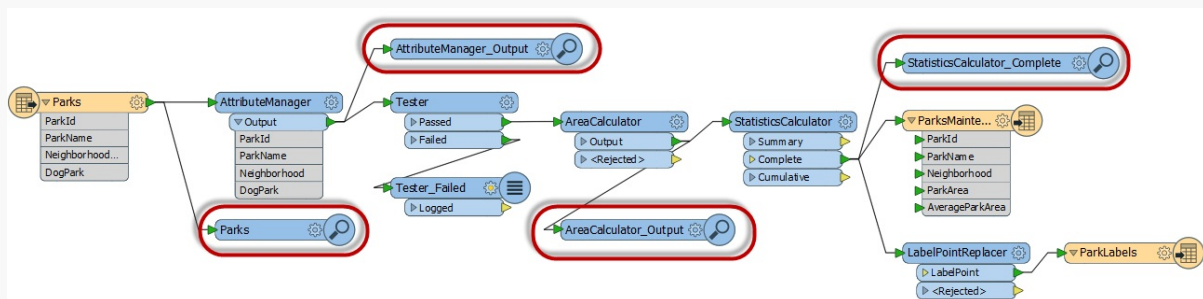
TIP

Inspectors are not useful for a workspace that is to be put into production, especially on FME Server. That's because an FME Server engine doesn't have the capability to open a Data Inspector.

For that reason, authors of FME Server workspaces tend to use a Logger transformer instead of an Inspector.

Mr. E. Dict, (Attorney of FME Law) says...

Pursuant to clause d30 (section 43) in your training course agreement, you are required to re-open the workspace from the previous example and add Inspector transformers to practice using this functionality.



Coordinate System Transformation

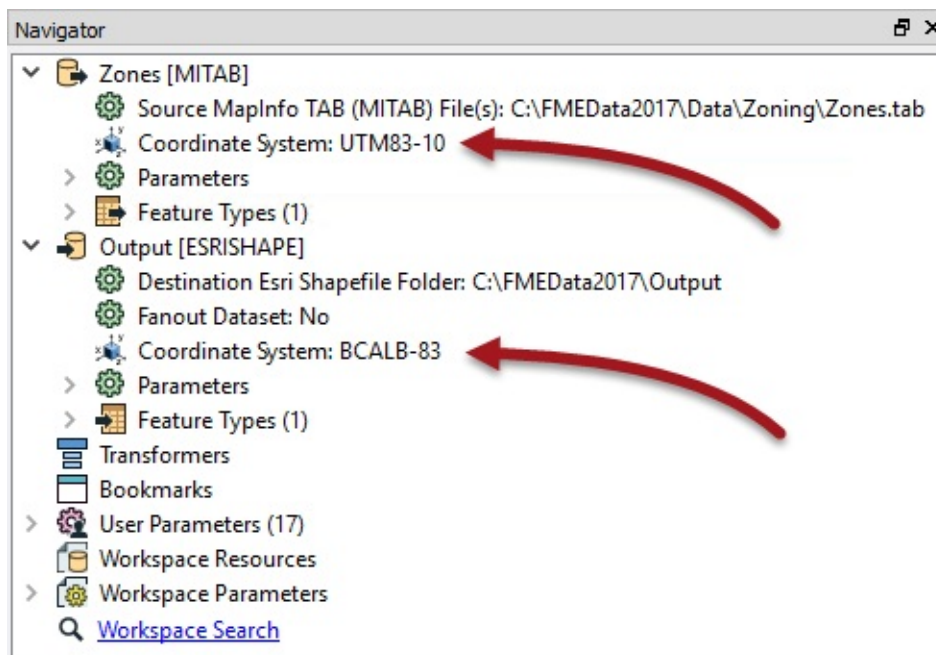
To be located in a particular space on the Earth's surface the majority of spatial data is related to a particular coordinate system.

Some users call this location of data a "projection," but projection is just one component of a definition within space. A true definition includes *projection*, *datum*, *ellipsoid*, *units*, and sometimes a *quadrant*, which together is called a **Coordinate System**.

Coordinate System Settings

Each reader and writer within FME can be assigned a coordinate system. That coordinate system is set in the Navigator window of Workbench, or in the Generate Workspace dialog.

Like the source schema, the reader coordinate system is **"what we have"** and the writer coordinate system is **"what we want"**. Here the source coordinate system has been defined as UTM83-10 and the destination as BCALB-83:



Each feature processed by the reader is tagged with the coordinate system defined in its parameter.

When a feature arrives at a writer, if it is tagged with a different coordinate system to what is defined for that writer, then FME automatically reprojects the data, so that the output is in the correct location.

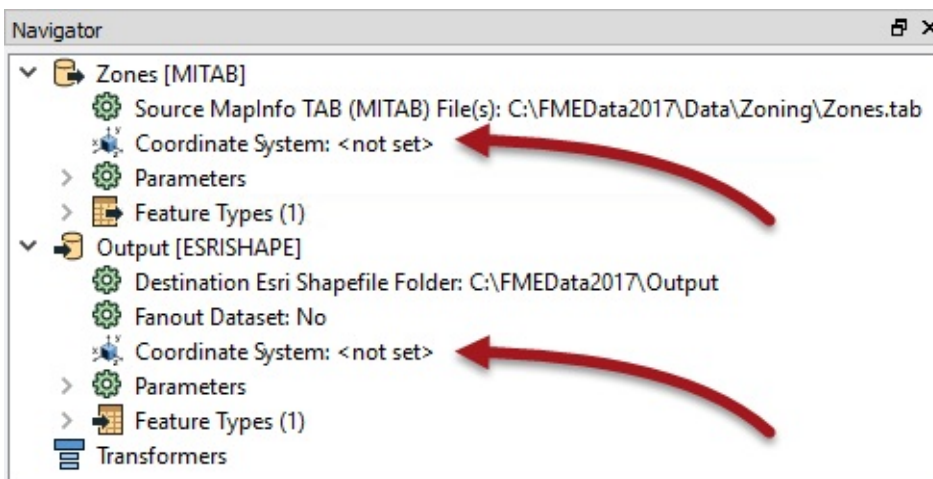
TIP

Once tagged with a coordinate system, each feature retains this throughout the translation; FME knows what coordinate system it belongs to at all times.

This is important when carrying out geometric transformations (like calculating area) or when reading multiple datasets that belong to different coordinate systems (yes, FME will handle that).

Automatic Detection of Coordinate Systems

It's not always necessary to set the coordinate system parameters manually. Some data formats (for example Esri Shapefile) are capable of storing information about the coordinate system in which they are held, and FME will retrieve this information where it can.



Here, because the reader coordinate system is marked <not set>, FME will try to determine the coordinate system from the source dataset. If it can't, then the feature will be tagged with a coordinate system of <unknown>.

Because the writer coordinate system is marked <not set>, FME will not reproject the data. Instead FME writes the data using the same coordinate system as the feature is currently tagged with.

WARNING

FME cannot reproject data if it does not know what coordinate system a feature belongs too.

Therefore problems can occur when a writer is set to create data in a specific coordinate system, but receives features tagged as <unknown>.

In that scenario either the writer will fail with an error message, or the data will be written, but with a series of warning messages in the FME log file.

This also occurs where the writer format has strict requirements on its coordinate system; for example KML datasets can only be written in a Latitude/Longitude series of coordinates.

Exercise 6 Grounds Maintenance Project - Data Reprojection	
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Data reprojection
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex6-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex6-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex6-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

The parks team has decided that the output data should be in an Albers Equal Area projection (coordinate system = BCALB-83). They think it will take ages to set this up! We'll show them differently...

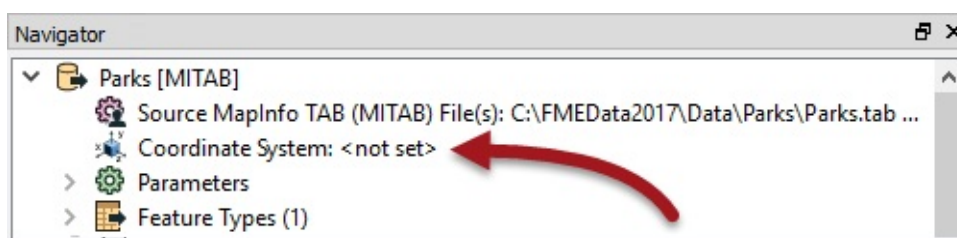
1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 5. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex6-Begin.fmw

2) Edit Reader Coordinate System

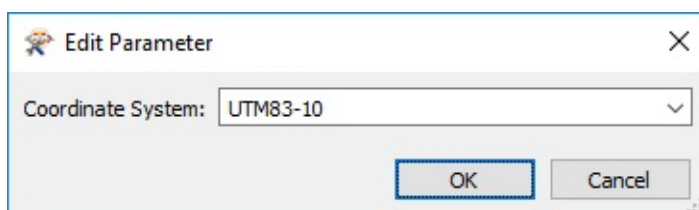
On the Navigator locate the Parks [MITAB] reader, and expand its list of settings.

Locate the setting labelled 'Coordinate System'. The original value should be <not set>:



Double-click the reader Coordinate System setting to open the Edit Parameter dialog.

Enter the coordinate system name UTM83-10 or select it from the Coordinate System Gallery by selecting "More Coordinate Systems..." from the bottom of the drop-down list.



TIP

Remember, 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.

When the source dataset is in a format that stores coordinate system information (as it does in this example) you can safely leave the parameter unset. So this step isn't really necessary.

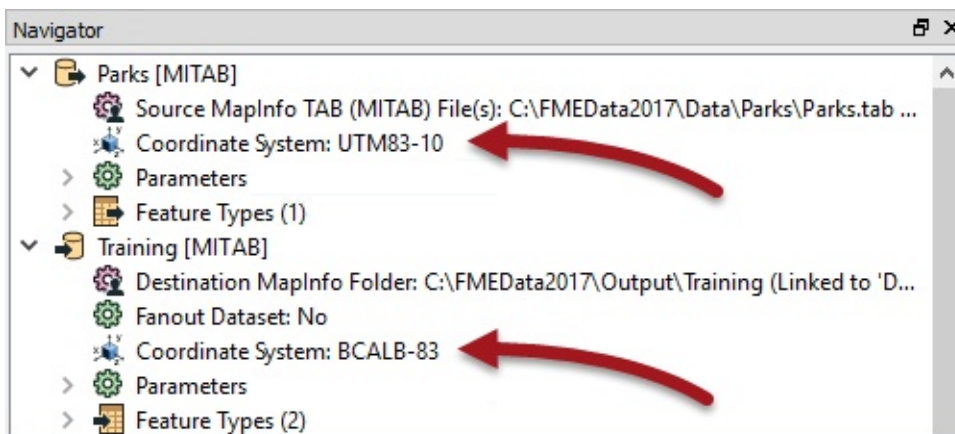
*However, you **MUST** set this parameter when you wish to reproject source data that does not store coordinate system information; otherwise an error will occur in the translation.*

3) Edit Destination Coordinate System

Now locate the coordinate system setting for the destination (writer) dataset.

Again the value should be the default value of <not set>.

Double-click the setting. Enter the coordinate system name BCALB-83 or select it from the Coordinate System Gallery by selecting "More Coordinate Systems..." from the bottom of the drop-down list.

**4) Run the Workspace**

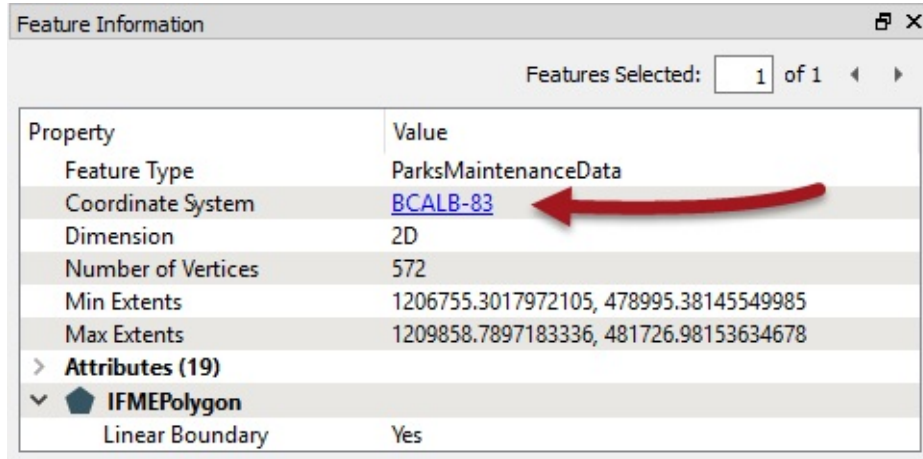
Save and then run the workspace.

In the log file you should be able to find:

```
FME Configuration: Source coordinate system for reader
MITAB_1[MITAB] set to `UTM83-10'
FME Configuration: Destination coordinate system set to `BCALB-
83'
```

5) Inspect the Output

Open the newly reprojected dataset and query a feature. The Feature Information window should report that the data is now in BCALB-83. Optionally, click on the coordinate system name in that window; a new dialog will open to display all of the coordinate system parameters.



TIP

*If the background map is activated when a dataset is opened then the contents of that dataset are automatically reprojected to spherical mercator to match the background map. If you wish to see the data as it appears in its own coordinate system, then use Tools > FME Options to turn off background maps **before** opening the source dataset.*

Advanced Exercise

Instead of using the reader/writer parameters in the Navigator window, why not try this exercise using the Reprojector (or CSMMapReprojector) transformer? Where should the transformer be placed in the workspace and why is this important?

CONGRATULATIONS

By completing this exercise you have learned how to:

- Use Coordinate System parameters to reproject spatial data
- Query features in the Data Inspector to inspect coordinate system information

Module Review

This module introduced you to the concept of Data Transformation and explained how to use FME Workbench for more than just quick translations.

What You Should Have Learned from this Module

The following are key points to be learned from this session:

Theory

- A Schema describes a dataset's structure, including its feature types, attributes, and geometries.
- Schema Editing is the act of editing the destination schema to better define what is required out of the translation.
- The act of joining the source schema to the destination is called Schema Mapping. Differences between the two schemas lead to Structural Transformation.
- Content Transformation is the modifying of data content during a translation. In FME Workbench data transformation is carried out using objects called Transformers, which can be found in the Transformer Gallery.
- Groups can be created using the Group-By option in a transformer's parameters dialog.
- Splitting data into multiple streams creates multiple copies and not a division of data. Bringing together multiple streams combines the data, rather than merges it.

FME Skills

- The ability to edit a Writer schema
- The ability to restructure data by mapping a reader to a writer schema, both manually and through transformers
- The ability to locate transformers in Workbench and use them to transform data content
- The ability to set transformer parameters in either the Parameter Editor or transformer dialogs
- The ability to define feature groups using the Group-By option
- The ability to reproject data using Workbench

Further Reading

For further reading why not browse [articles tagged with Transformation](#) on our blog?

Questions

Here are the answers to the questions in this chapter.

Mr Flibble says...

The most common format translation defined with FME is from Esri Shapefile to which format?

1. Esri Geodatabase
2. AutoCAD DXF/DWG
3. Google KML
- 4. Esri Shapefile**

Yes! The most common translation is from Esri Shapefile TO Esri Shapefile! It just proves how many people use FME for data transformation instead of format translation.

Miss Vector says...

Which three colours represent checked, need checking, and unset parameters on transformer objects?

- 1. blue, yellow, red**
 2. green, yellow, red
 3. red, green, blue
 4. green, blue, yellow
-

Miss Vector says...

If I use a transformer to remove irregularities (like self-intersecting loops) in the boundary of a polygon, what type of transformation is it?

1. Structural Transformation of attributes
 2. Structural Transformation of geometry
 3. Content Transformation of attributes
 - 4. Content Transformation of geometry**
-

Miss Vector says...

Which of the following transformers do you think is "group-based"?

- 1. StringFormatter*
- 2. Clipper**
- 3. Rotator*
- 4. AttributeRounder*

The StringFormatter, Rotator, and AttributeRounder can all operate on one feature at a time. For example, you can rotate one feature independently of any other. However, the Clipper can only operate on a group of features. The "Group-By" parameter is a huge clue to identifying group-based transformers!

Exercise 7 Voting Analysis Project	
Data	Election Mapping (GML) Election Statistics (Microsoft Excel)
Overall Goal	Map statistics of voting patterns
Demonstrates	Data Transformation
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex7-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex7-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Transformation-Ex7-Complete-Advanced.fmw

In a break from grounds maintenance projects, the municipal Elections Officer has heard about your skills and asked for help identifying voting divisions that had a low turnout at the last election, or divisions where there were difficulties understanding the voting process.

He asks for your help and you suggest the results should be presented in Google KML format, so staff can view them without having to use a full-blown GIS system.

1) Inspect Data

Start the FME Data Inspector and open the two datasets we will be using:

Reader Format	GML (Geography Markup Language)
Reader Dataset	C:\FMEDData2017\Data\Elections\ElectionVoting.gml
Reader Format	Microsoft Excel
Reader Dataset	C:\FMEDData2017\Data\Elections\ElectionResults.xls

Notice that both datasets have a Division attribute by which to identify each voting division (area). The Excel data is non-spatial but has a set of other voting attributes:

- **Voters:** Number of registered voters
- **Votes:** Number of voters who voted
- **Blanks:** Number of voters who left a blank or spoiled vote
- **OverVotes:** Number of voters who voted for too many candidates
- **UnderVotes:** Number of votes not cast

The OverVotes and UnderVotes attributes are an indicator of how well the voting process was understood. Each voter gets to vote for up to 10 candidates (out of 30).

OverVotes are those voters who voted for more than ten candidates. UnderVotes are the number of votes that could have been cast, but were not (for example, the voter only voted for five candidates).

2) Start Workbench

Start Workbench and open the starting workspace. It already has Readers and Writers

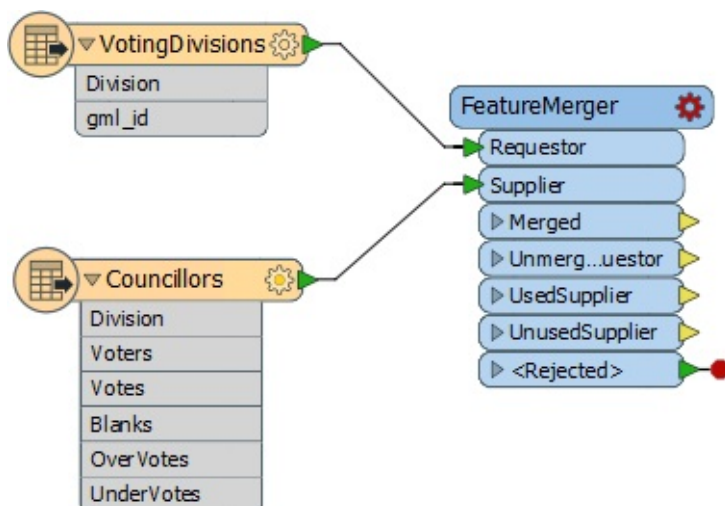
added to handle the data; all we need to do is carry out the transformation:



3) Add FeatureMerger

The first task is to merge the statistical election data onto the actual features. We'll use a FeatureMerger transformer to do this (more on this transformer appears later in this course).

Add a FeatureMerger transformer. Connect the VotingDivisions data to the Requestor port, and the Councillors (result) data to the Supplier.



The voting division features are literally requesting information from the results features.

4) Set Parameters

View the FeatureMerger parameters. For both the Requestor and Supplier join fields, click the drop-down arrow and Attribute Value > Division.

This is the common key by which our data is merged:

Transformer

Transformer Name:

Group By: ... ▼

Suppliers First: ▼

Join On

Requestor	Supplier	Comparison Mode
Division	Division	Automatic

+ -

Click Apply/OK to confirm the parameters.

5) Add Inspector Transformer

Add an Inspector transformer after the FeatureMerger:Merged output port. Run the workspace.

Ignore any warning or log message that reports Unexpected Input.

Ensure that the Feature Count on that connection is the same as the number of incoming features from the VotingDivisions (note, there will be extra features from the Councillors data that is not used, because that includes divisions outside our area of interest).

Also examine the data in the FME Data Inspector to ensure all division polygons now include a set of attribute data copied from the Excel spreadsheet:

View 1 X

Table View

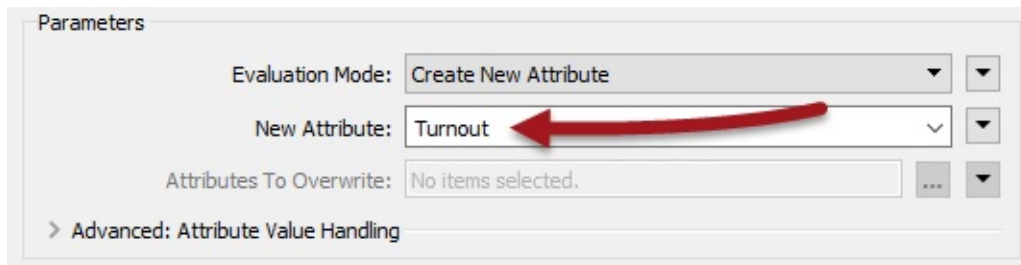
Table: inspector [FFS] - FeatureMerger_Merged Columns...

	Division	gml_id	Voters	Votes	Blanks	OverVotes	UnderVotes
1	35	id2a483603-57a...	3195	1129	32	0	1702
2	15	idd7ae6cc6-381...	3997	780	34	0	1160
3	19	idc1449819-747...	3062	522	35	0	852
4	18	id6d87e2c8-9e4...	3880	853	37	1	1463
5	38	id465b45b3-83f...	1521	559	12	0	710
6	37	id8a4509e0-9ac...	2703	833	25	0	1378

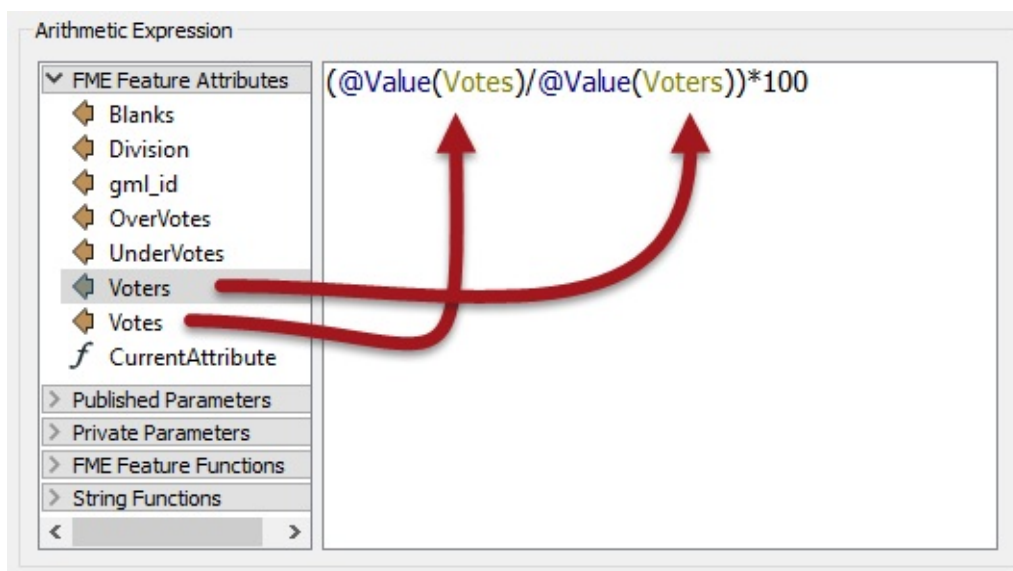
6) Add ExpressionEvaluator

Now that we have the numbers we need, we can start to calculate some statistics. To do this we'll use an ExpressionEvaluator transformer to first calculate voter turnout percentage for each division.

Place an ExpressionEvaluator transformer after the FeatureMerger - connect it to the FeatureMerger:Merged output port. View the transformer's parameters. Set the New Attribute to Turnout (to match what we have on the destination schema):



In the expression window set the expression to:



```
(@Value(Votes)/@Value(Voters))*100
```

You don't need to type this in - the @Value(Votes) and @Value(Voters) part can be obtained by double-clicking that attribute in the list to the left.

Click Apply/OK to confirm the parameters. If you wish you can reconnect an Inspector and re-run the translation, to see what the result is.

7) Add ExpressionEvaluator

Using a similar technique, use an ExpressionEvaluator to calculate the number of UnderVotes per voter and put it in an attribute that matches the output schema. The expression will be something like:

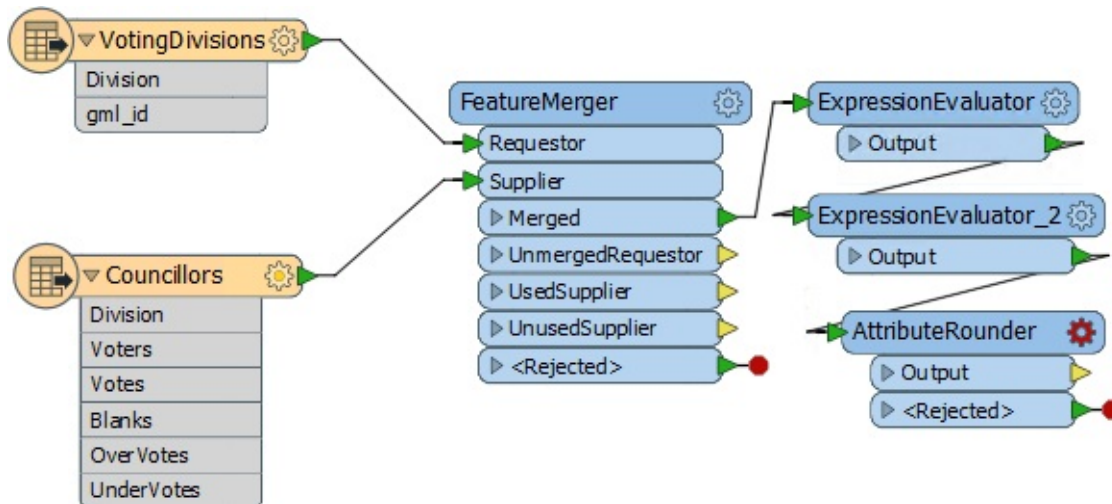
```
@Value(UnderVotes)/@Value(Voters)
```


***NB:** This isn't a percentage, like the previous calculation.*

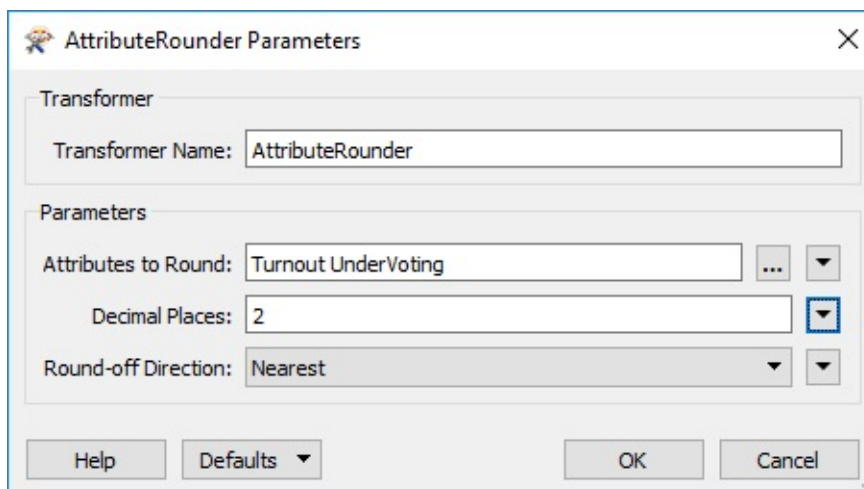
8) Add AttributeRounder

It's a bit excessive to calculate our statistics to 13 decimal places or more. We should truncate these numbers a bit.

Place an AttributeRounder transformer.



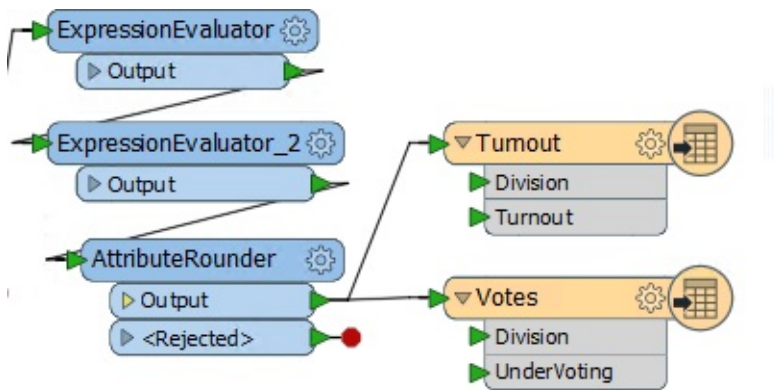
Open the parameters dialog. Under Attributes to Round select the newly created Turnout and UnderVoting attributes. Set the number of decimal places to 2:



Click OK to close the dialog and, again, run the workspace to check the results if you wish.

9) Connect Schema

For the final step let's connect the AttributeRounder to the output schema. Simply make connections from the AttributeRounder to both writer feature types:



Run the workspace and examine the output in Google Earth to prove it has the correct attributes and is in the correct location.

Advanced Exercise

The project is done, but the output is very plain. It would be much better to improve the look of the results and there are several ways to do this with KML.

We could simply color the voting divisions differently according to their turnout/overvotes, but a more impressive method is to use three-dimensional blocks.

Follow the steps below to create three-dimensional shapes in the output KML dataset...

10) Add ExpressionEvaluator

The height of each block should be proportional to the turnout for that division. However, for differences to be clearly visible, the vertical scale will need some exaggeration.

Place an ExpressionEvaluator between the AttributeRounder and the Turnout feature type. Set the parameters to multiply the Turnout attribute by a value of 50. Put it into a new attribute called TurnoutScaled.

The screenshot shows the 'Parameters' section of a dialog box. It includes three dropdown menus: 'Evaluation Mode' set to 'Create New Attribute', 'New Attribute' set to 'TurnoutScaled', and 'Attributes To Overwrite' set to 'No items selected.'. Below these is a link to 'Advanced: Attribute Value Handling'. The 'Arithmetic Expression' section shows a tree view on the left with categories like 'FME Feature Attributes', 'Published Parameters', 'Private Parameters', 'FME Feature Functions', 'String Functions', 'Math Functions', and 'Math Operators'. The main text area contains the expression '@Value(Turnout)*50'.

11) Add 3DForcer

Add a 3DForcer transformer after the new ExpressionEvaluator. This will elevate the feature to the required height. In the parameters dialog set the elevation to Attribute Value > TurnoutScaled.

The screenshot shows the 'Transformer' section with 'Transformer Name' set to '3DForcer'. The 'Parameters' section has 'Elevation' set to 'TurnoutScaled' (indicated by a small icon) and 'Preserve Existing Z Values' set to 'No'.

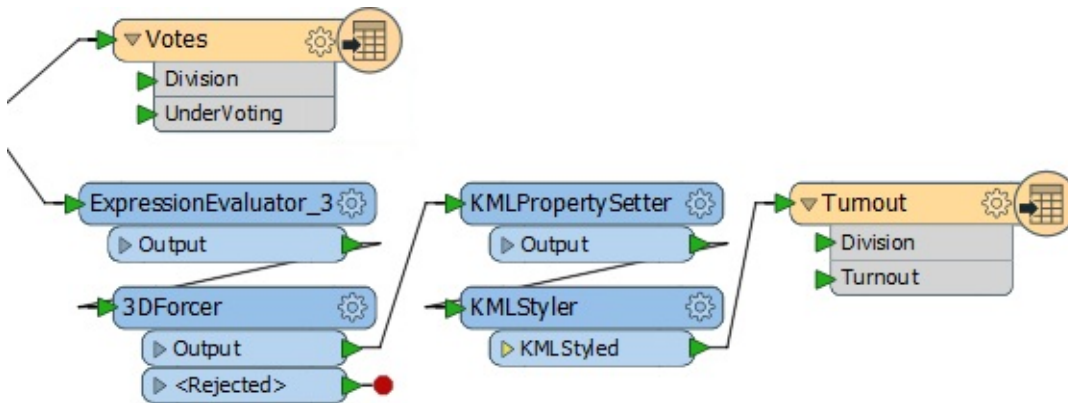
12) Add KMLPropertySetter

Add a KMLPropertySetter transformer after the 3DForcer. This will allow us to set up the 3D blocks in the output. Set the geometry parameters as follows:

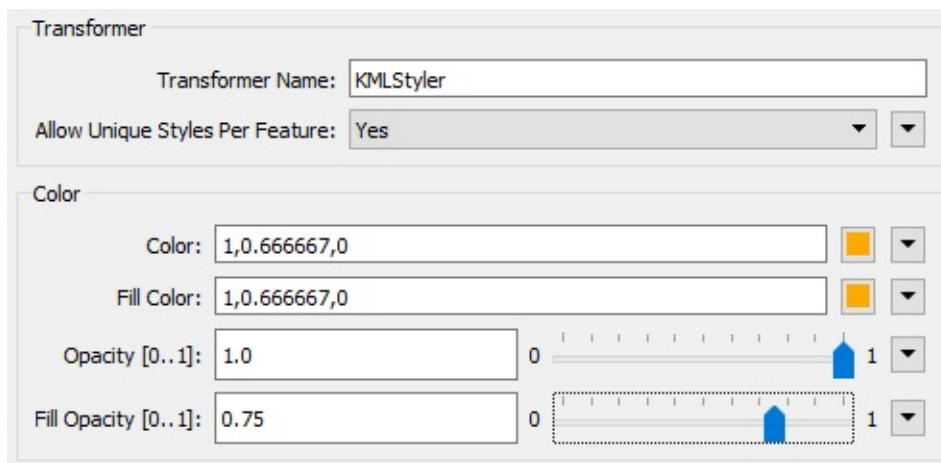
- Altitude Mode: Absolute
- Extrude: Yes

The screenshot shows the 'Geometry' section of a dialog box. It includes several settings: 'Geometry Type' set to 'Vector', 'Altitude Mode' set to 'Absolute', 'Draw Order' as an empty field, 'Raster Altitude' as an empty field, 'Raster Opacity [0..1]' with a slider set to 0, 'Extrude' set to 'Yes', and 'Follow Terrain' set to 'Yes'. Red arrows point to the 'Altitude Mode' and 'Extrude' settings.

13) Add KMLStyler Finally add a KMLStyler transformer. The workspace will now look like this:



Check the parameters. Select a color and fill color for the features. Increase the fill opacity to around 0.75.



Save and run the workspace. In Google Earth the output should now look like this:



These 3D blocks will show users where the voting turnout is high/low in the city.

If you wish, repeat these steps to give a 3D representation to the UnderVoting statistics.

CONGRATULATIONS

By completing this exercise you proved you know how to:

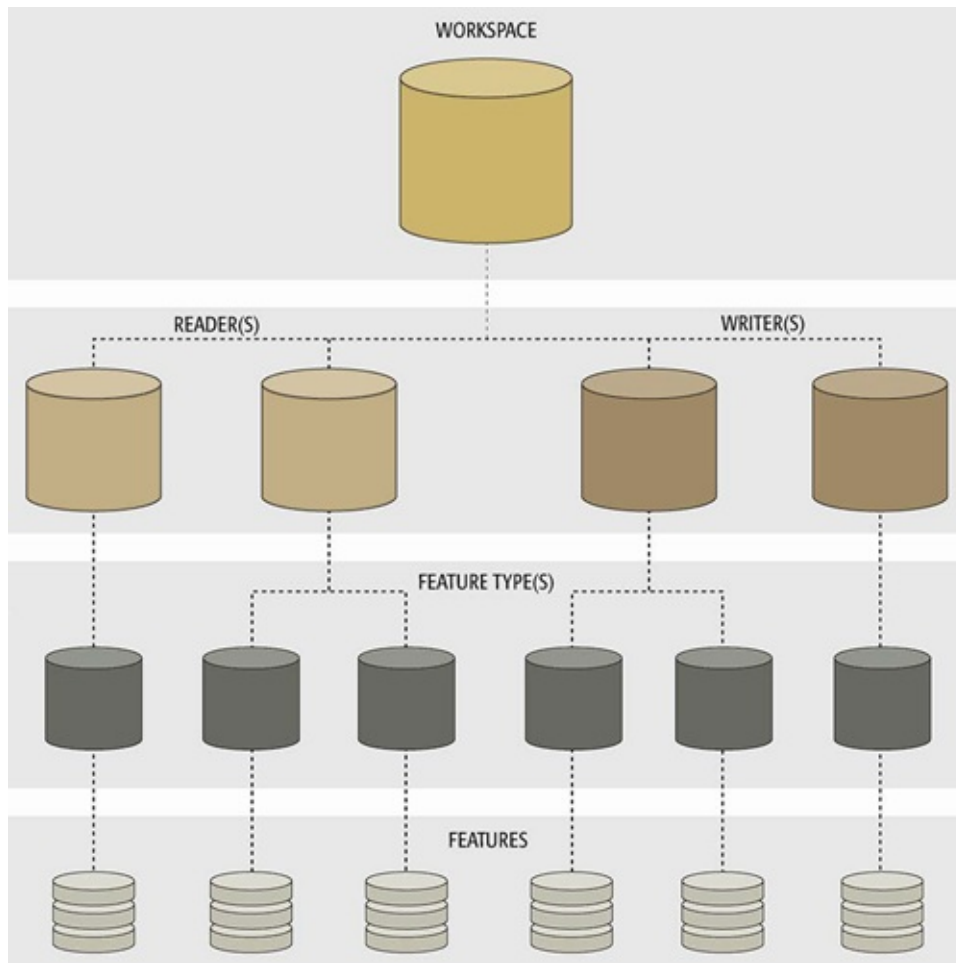
- *Carry out content transformation with transformers (`ExpressionEvaluator`, `AttributeRounder`, `3DForcer`)*
- *Use transformer parameters to create attributes that match the writer schema*
- *Use multiple streams of transformers in a single workspace*

You also learned how to:

- *Merge multiple streams of data using a common key (`FeatureMerger`)*
- *Use FME's built-in maths editor dialog*
- *Use transformers to set a symbology (style) for output features*

Translation Components

An FME translation is made up of a number of different components.



For each component there are:

- Tools to create, add, or remove them
- Parameters to control them

It's very useful to have an understanding of FME components, especially when you work with multiple datasets or multiple formats.

Key Components

The list of key components in an FME translation is as follows:

- Workspace
- Readers and Writers
- Feature Types
- Features

TIP

This section covers “official” FME components only. For example, it won’t cover any user-defined Python scripting that might be used to exert control over several workspaces.

However, once you understand the structure, it’s easy to imagine where such custom components might fit in.

Workspace

A workspace is the primary element in an FME translation and is responsible for storing a translation definition. A workspace is held as a file with an .fmw file extension. It can be run in either the Quick Translator or FME Workbench, but can only be opened for editing in Workbench.

Think of a workspace as the container for all the functionality of a translation.

Readers and Writers

A **reader** is the FME term for the component in a translation that reads a source dataset. Likewise, a **writer** is the component that writes to a destination dataset.

We'll see that readers and writers don't appear as objects on the Workbench canvas, but instead are represented by entries in the Navigator window.

Feature Types

Feature Type is the FME term that describes a subset of records. Common alternatives for this term are 'layer,' 'table,' 'sheet,' 'feature class,' and 'object class.' For example, each layer in a DWG file is defined by a feature type in FME.

Feature types are the brown-colored objects that appear on the Workbench canvas.

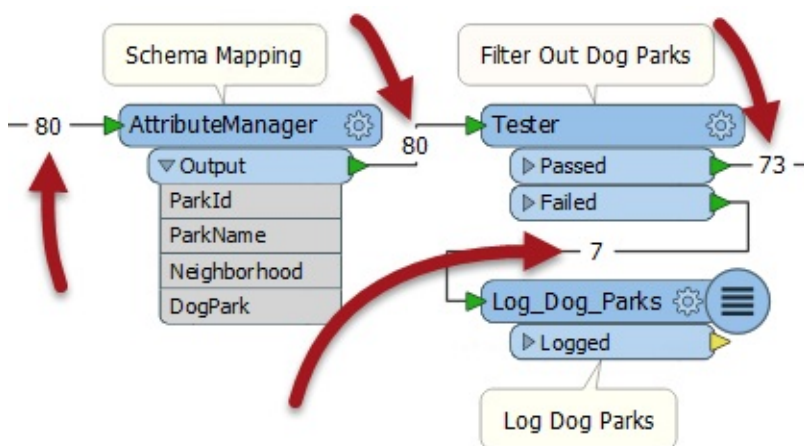
TIP

*Don't confuse the term Feature Type with Geometry Type.
Feature Type means "layer;" Geometry Type means "lines," "points," "polygons."*

Features

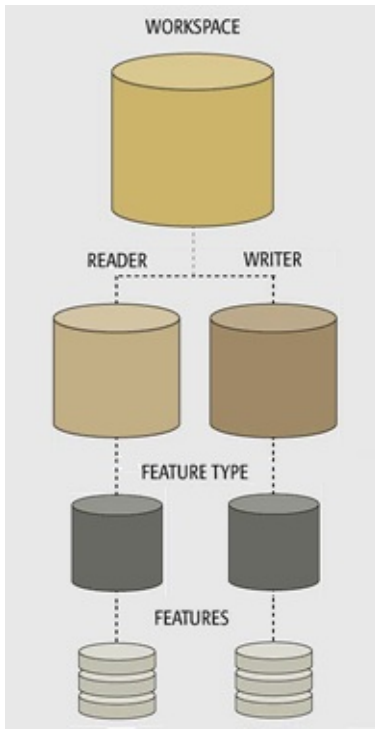
Features are the smallest single components of an FME translation.

They aren't individually represented within a workspace, except by the feature counts on a completed translation.



Component Hierarchy

It's important to notice that all these components exist in a related hierarchy:

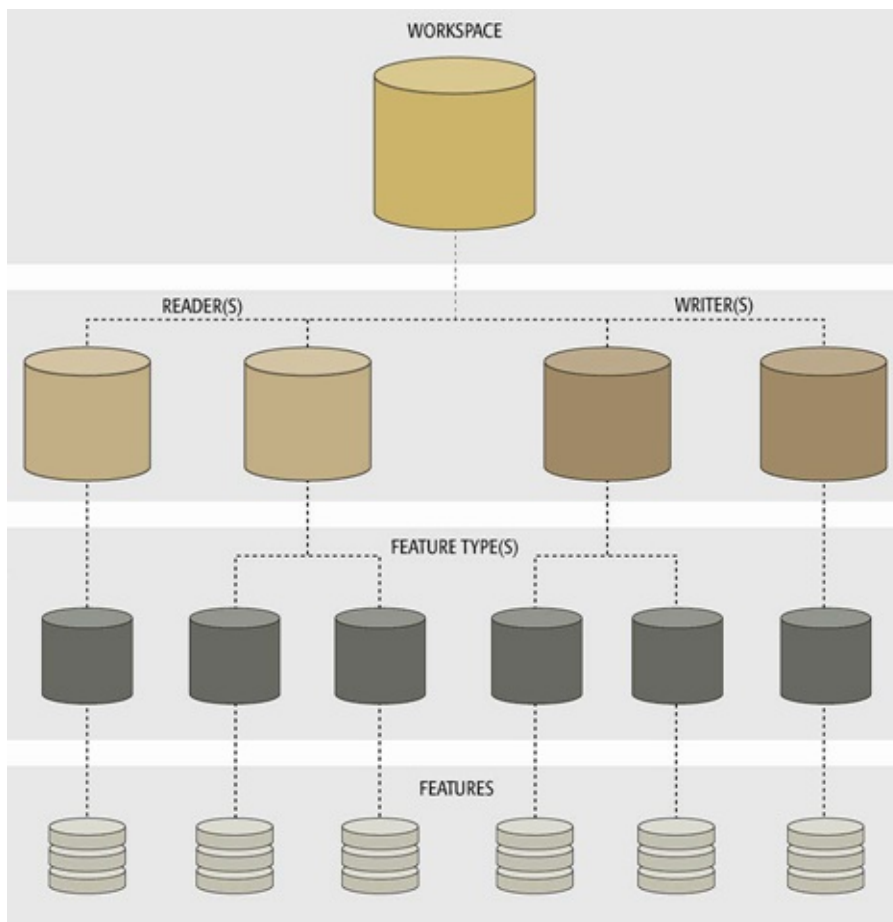


The workspace is at the top level, followed by readers and writers, feature types, then features.

Hierarchy is an important concept because it affects how components are added to a translation, and - more importantly - how they are controlled.

One-To-Many Relationships

The hierarchical relationship between workspace, readers, writers, feature types, and features isn't always one-to-one; often it is a one-to-many (1:M) relationship with the level beneath:



A single workspace can contain any number of readers and writers, each reader can contain a number of feature types, and each feature type can contain any number of features within it.

This means that a single workspace can read and write multiple datasets, in multiple formats, with multiple layers, and multiple features per layer.

Now we know what these components are, let's look at each of them individually, in detail, to see how they can be managed and how they can be controlled.

Miss Vector says...

Here's a fun question to test your knowledge of formats. Here are four different formats and four different terms for "feature type". Connect the format to the correct terminology.

Format	Feature Type Terminology
Oracle Spatial	Level
MicroStation Design	Layer
Esri Geodatabase	Table
Adobe PDF	Feature Class

Workspaces

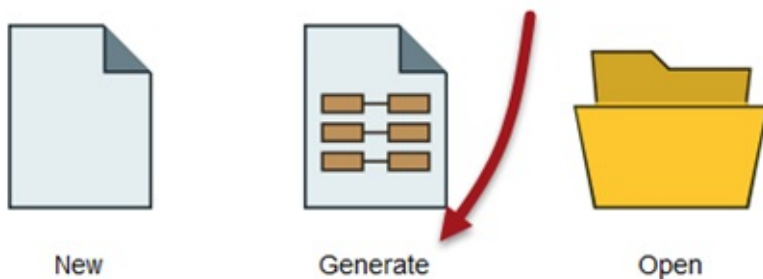
A **workspace** is a file responsible for storing a translation definition.

Workspaces are the primary containers of translation components. At the top of the hierarchy they can contain any number of readers, writers, and feature types; or sometimes none at all!

Creating a Workspace

Workspaces can be created using the commands on the File menu, or through shortcuts in the Start tab.

Create Workspace



Creating a workspace through the Generate option is a simple way to define a translation because it includes reader, writer and feature type components in the setup process.

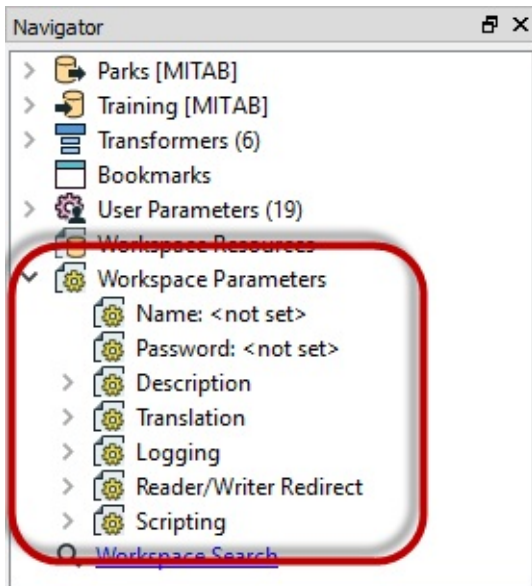
However, a workspace can also be created empty - i.e. the canvas is blank and each new component is added from scratch. The "New" option on the Start tab will allow you to do that.

Using File > New on the menubar opens up a dialog in which a new workspace can be generated or an empty one created. However, it also has the option to create a workspace from a template. That template can come from either local, user-defined templates, or templates available on the FME Hub.

Controlling a Workspace

Workspace parameters are those that relate to a workspace as a whole, and which have an effect on how the translation is performed. It is very important to remember that workspace parameters apply only to the current workspace. They are not global FME options and so may differ between workspaces.

Workspace parameters are shown and set in the Navigator Window.



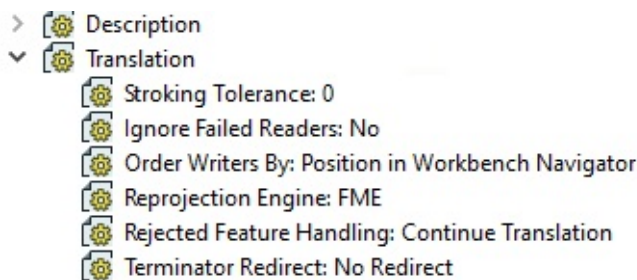
For ease-of-use, workspace parameters are divided into several sections. The most important ones are Translation, Logging, and Scripting.

NEW

This arrangement of parameters in FME 2017 is changed from 2016, which was only divided into basic and advanced, and had the Description fields under "Workspace Properties".

Translation Parameters

Translation parameters provide the ability to set values that are applied during a translation.

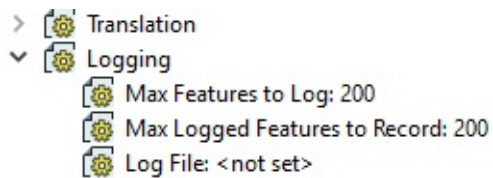


For example, Reprojection Engine allows the author to choose a different engine for reprojecting data between coordinate systems. This can help ensure the data FME writes matches exactly existing data from another application.

Rejected Feature Handling defines what happens when a feature is output through a <Rejected> port on a transformer. Either the translation can continue, or the translation can be halted.

Logging Parameters

There are three logging parameters:

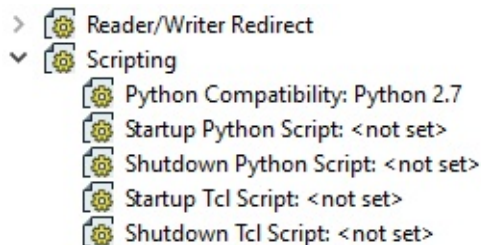


The *Max Features to Log* and *Max Logged Features to Record* parameters control how many features that are incorrect will be recorded in the log (or spatial log file). Often, just knowing that one feature was logged as a problem is enough to revisit the source data; you don't need to pad the log file by logging every single problem feature.

The *Log File* parameter specifies where the log file is written to. This is important when you want to keep that file for future reference. For example, you might (as an advanced task) create a user parameter to set the Log File location to C:\LogFileStorage\<date/time> so that each log file is recorded with a different file name matching the translation date/time.

Scripting Parameters

Scripting parameters deliver the ability to run a Tcl or Python script. There are parameters in which to enter a script and also a parameter that defines script compatibility:



A startup script is executed when a workspace is executed, before any of the workspace canvas instructions are carried out. A shutdown script is executed directly after the workspace has finished, usually once it is done writing data.

Potential uses of such scripts include:

- To check a database connection before running the translation
- To move data prior to or after the translation
- To write the translation results to a custom log or send them as e-mail to an administrator
- To run scripts from other applications; for example Esri ArcObjects Python scripts

The Python compatibility parameter allows the user to choose which version of Python will be used to run any scripts. A specific version can be chosen but if you choose the option "2.7 or 3.4+" then you are instructing FME that your script is compatible with both versions of Python and that it doesn't matter which one FME uses.

Exercise 1 Creating a Workspace

Data	City Parks (MapInfo TAB)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Workspace component and parameters
Start Workspace	None
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex1-Complete.fmw

The city is hosting a fundraising walk for a major charity. There will be many thousands of people taking part and good organization of the event is vital. One of the requirements is a map for participants and you have been tasked with using FME to put together the data that will form the map.

The data required for this project (and the format it is held in) includes:

Dataset	Format	Location
Park outline	MapInfo TAB	C:\FMEDData2017\Data\Parks\Parks.tab
Trail Route	Comma Separated (CSV)	C:\FMEDData2017\Data\GPS\WalkingTrail.csv
Food vendors	Esri File Geodatabase	C:\FMEDData2017\Data\CommunityMapping\CommunityMa
Parking facilities	OpenStreetMap	C:\FMEDData2017\Data\OpenStreetMap\amenity.osm
Roads into/out of park	OpenStreetMap	C:\FMEDData2017\Data\OpenStreetMap\highway.osm

The outputs required are Google KML and GPX. So with this and the amount of input data, you will need to add lots of Readers (and Writers) to your workspace.

1) Inspect Data

As usual, the first thing to do is inspect the source data. There is a lot of it and - in many cases - the required data is just one feature or one layer within a larger dataset. The final dataset we want will look something like this:



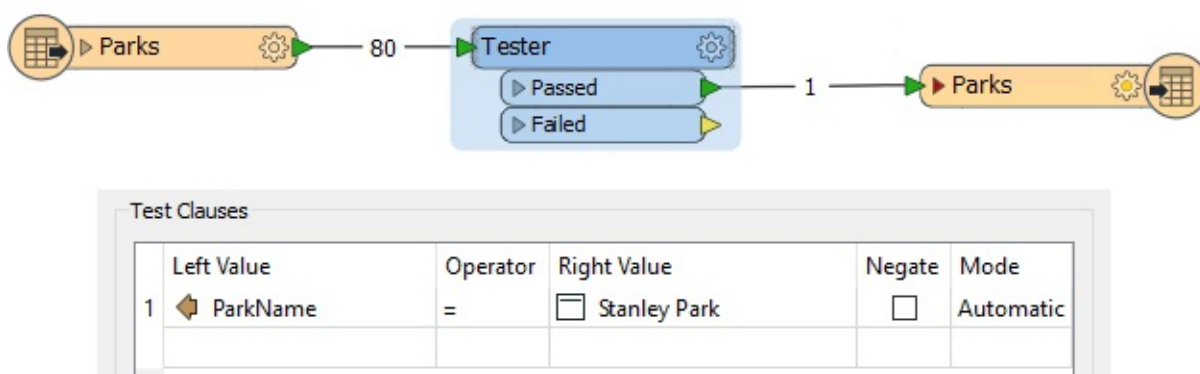
2) Create Workspace

OK. Let's get going. Start FME Workbench and choose the option to generate a workspace. We'll start with the parks data so, when prompted, enter the following choices:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2017\Data\Parks\Parks.tab
Writer Format	Google KML
Writer Dataset	C:\FMEData2017\Output\Training\FundraisingWalk.kml

3) Add Tester

The park that the walk is taking place in is called Stanley Park. Add a Tester transformer and use it to filter out any park features that are not part of Stanley Park:



Run the workspace (as you can see from above, only 1 feature passes the test) and view the output in either the FME Data Inspector or Google Earth.

4) Add Shutdown Script

The organizing team has created a share folder for materials. We could write our data to it directly, but instead let's write the file to our system, then copy it to the centrally shared folder.

In the Navigator window locate the parameter Workspace Parameters > Scripting > Shutdown Python Script and double-click it to open an editor window. In the window enter the following code:

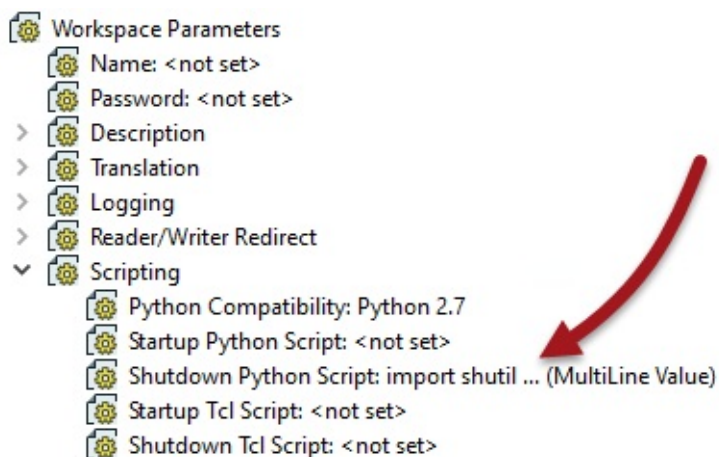
***NB:** The code can be copied from the file:

C:\FMEDData2017\Resources\DesktopBasic\ComponentsExerciseScript.py*

```
import shutil
import os

if not os.path.exists('C:/FundraisingWalk'):
    os.makedirs('C:/FundraisingWalk')
shutil.copy2('c:/FMEDData2017/Output/Training/FundraisingWalk.kml', 'C:/FundraisingWalk/FundraisingWalk.kml')
```

You'll notice that we're not really writing to a shared folder, just a different folder on our own system. That's OK. It's not a real project! Anyway, the workspace parameters should look like this:



5) Run Workspace

Run the workspace. You should see that the 'shared' folder is created (assuming it doesn't already exist) and the data is copied to it. This is an excellent example of the sort of thing you might do with a shutdown script.

Aunt Interop says...

If the workspace fails because of a Python error (it shouldn't, but just in case), don't worry about it. Everyone has a failure sometime. We aren't here to debug Python so just erase the script from the parameter and carry on. The important part is that you can see the type of thing a script can do and know where the parameters are to use them.

But if there was an error, it might be because you didn't write the output file with the same name or in the same directory as specified in the steps above!

CONGRATULATIONS

By completing this exercise you have learned how to:

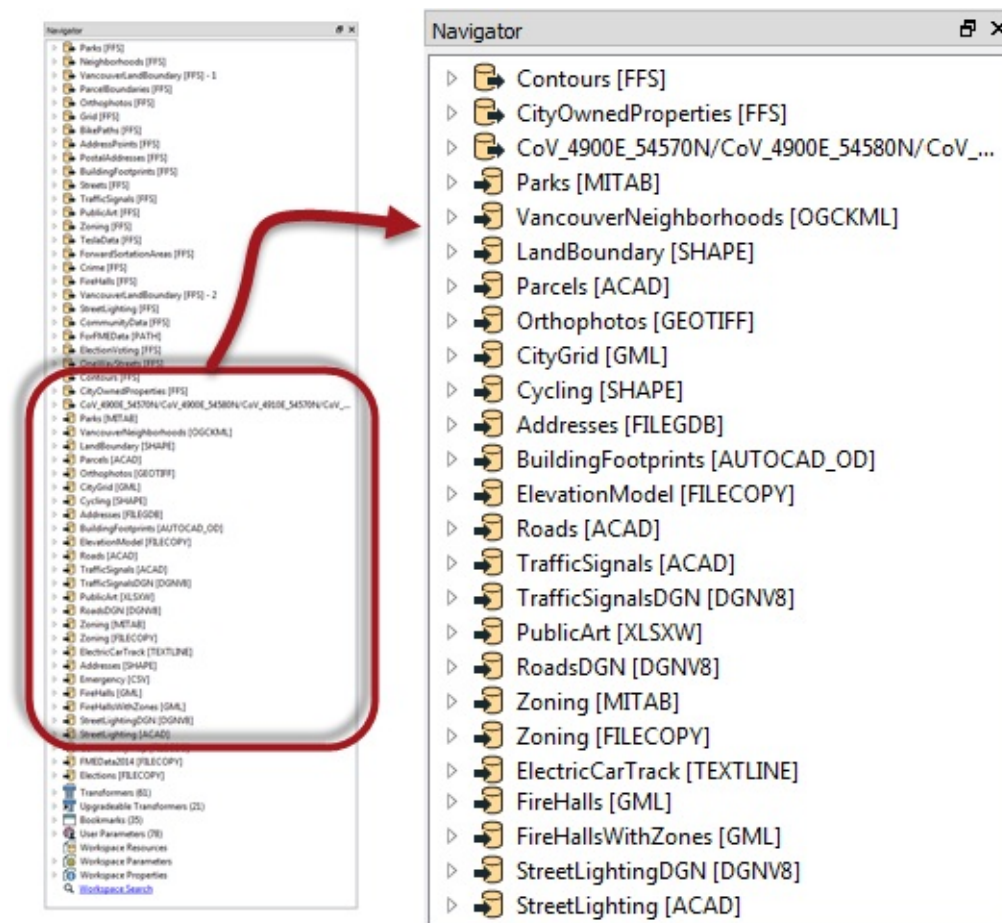
- *Create a new workspace with reader and writer as the first step in a larger project*
- *Locate and use a Python shutdown script*

Readers

A **reader** is the FME term for the component in a translation that reads a source dataset. A reader reads a single format of data, so to read multiple formats requires multiple readers. However, each reader can read any number of datasets in its format.

By default, the Generate Workspace dialog creates workspace with a single reader (and a single writer). However, this does not mean the workspace is forever limited to this. Additional readers can be added to a workspace at any time, any number of formats can be used, and there does not need to be an equal number of readers and writers.

For example, the Navigator window shows this workspace contains 50+ readers (and writers) of all data types and formats!



Ms. Analyst says...

It's important to note that readers and writers don't appear as objects on the Workbench canvas. Their feature types (layers) do, but readers and writers don't.

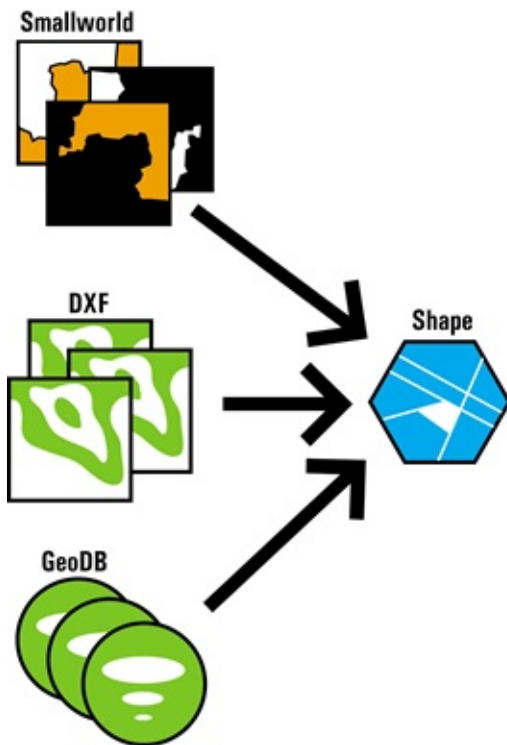
Instead they are represented by entries in the Navigator window, as in the above screenshot.

Adding a Reader

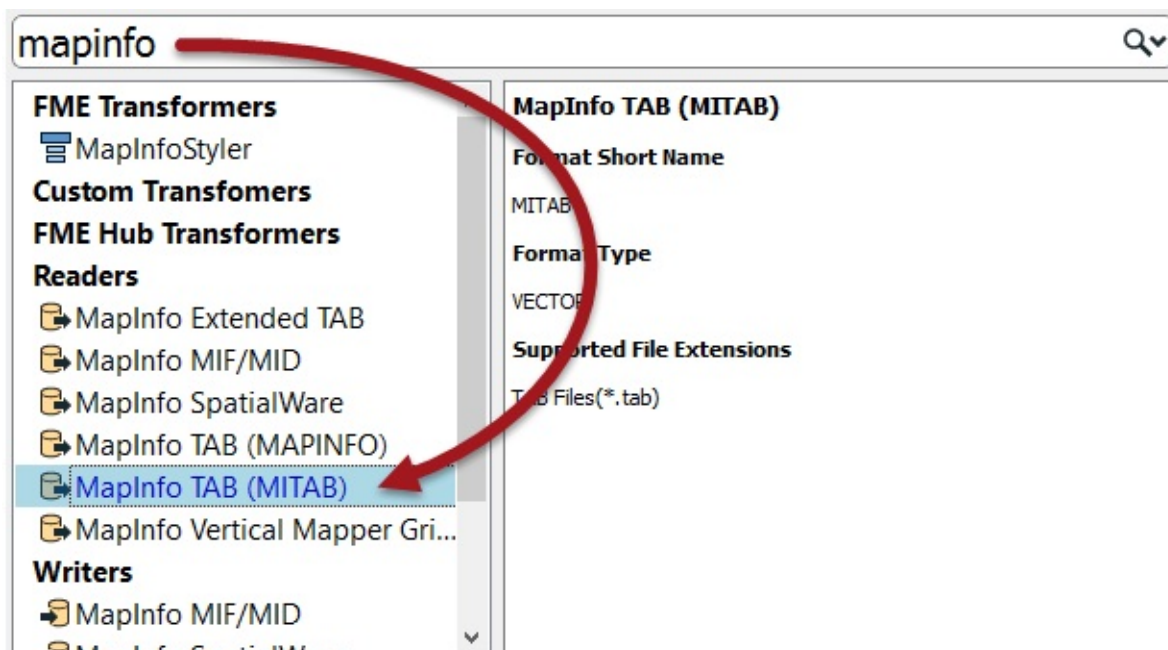
Adding a reader to a workspace is a common requirement. There are several reasons:

- The Generate Workspace dialog only adds a single reader
- Each reader handles only one format of data
- Different datasets (of the same format) may require reading with different parameters

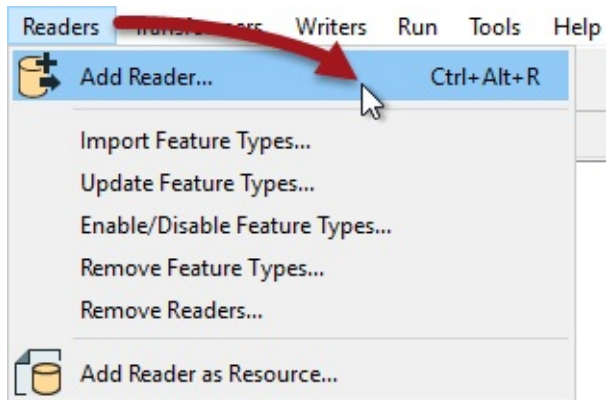
Therefore the need to read multiple formats of data – such as Smallworld, DXF, and Geodatabase – requires multiple readers.



Additional readers are added to a translation using the Quick Add menu:



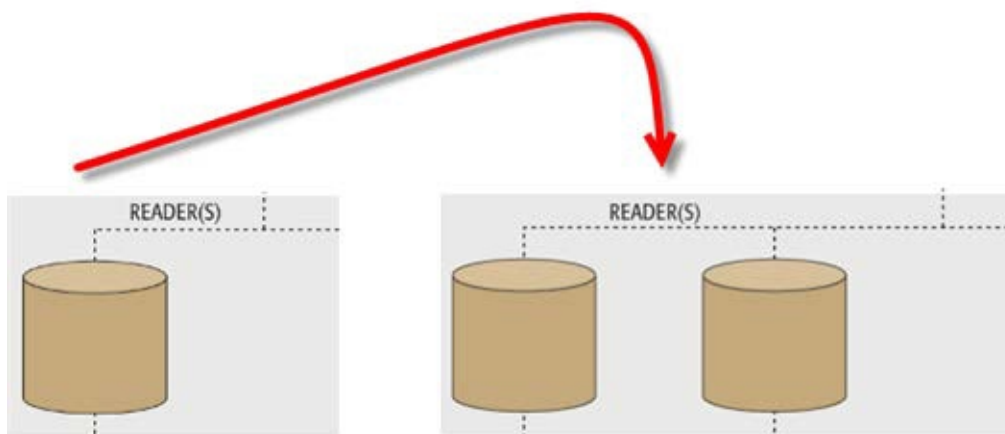
...Or by selecting Readers>Add Reader from the menubar.



NEW

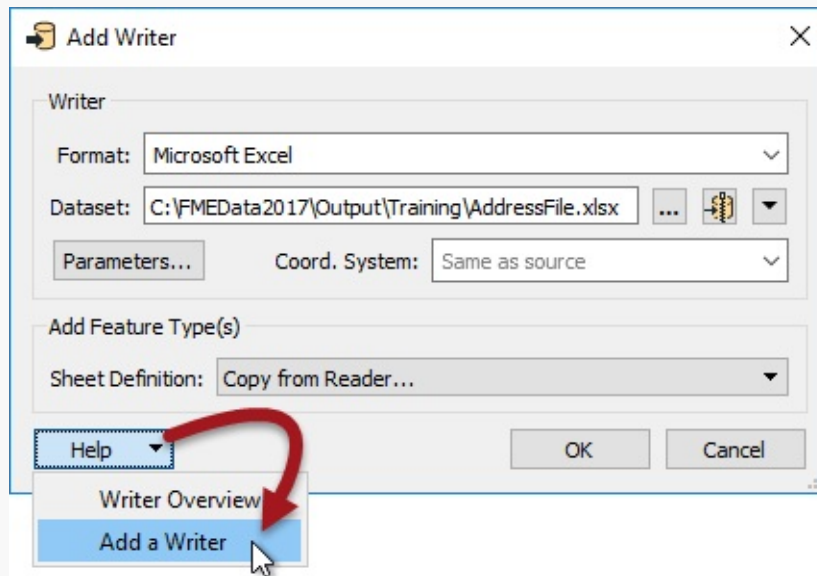
The ability to add a reader using Quick Add is new for FME 2017

Adding a reader has this effect on the hierarchy diagram:



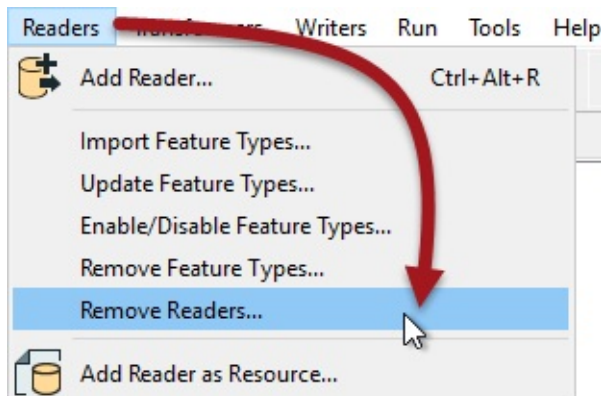
.1 UPDATE

In FME2017.1 the Add reader/writer dialogs have Help options for both the format and the act of adding the reader/writer:



Removing a Reader

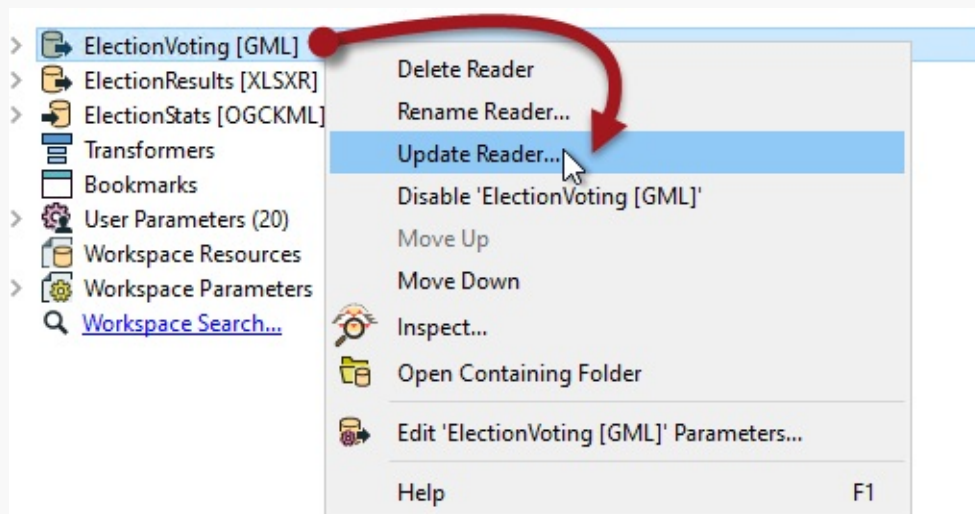
Not only can you add a new reader, you can remove an existing one; for example when you have an old reader whose input you no longer need. Tools exist to remove a reader from a workspace, both on the menubar and in context menus in the Navigator window.



Removing a reader obviously has the reverse effect on the hierarchy diagram!

.1 UPDATE

*FME2017.1 introduces a tool to **update** a reader in Workbench:*



Updating is a way to bring a reader's behaviour up to date with the current FME version. Obviously it only applies when the workspace was previously created/edited in an older version of FME. Such an update can improve performance or allow the use of new control parameters.

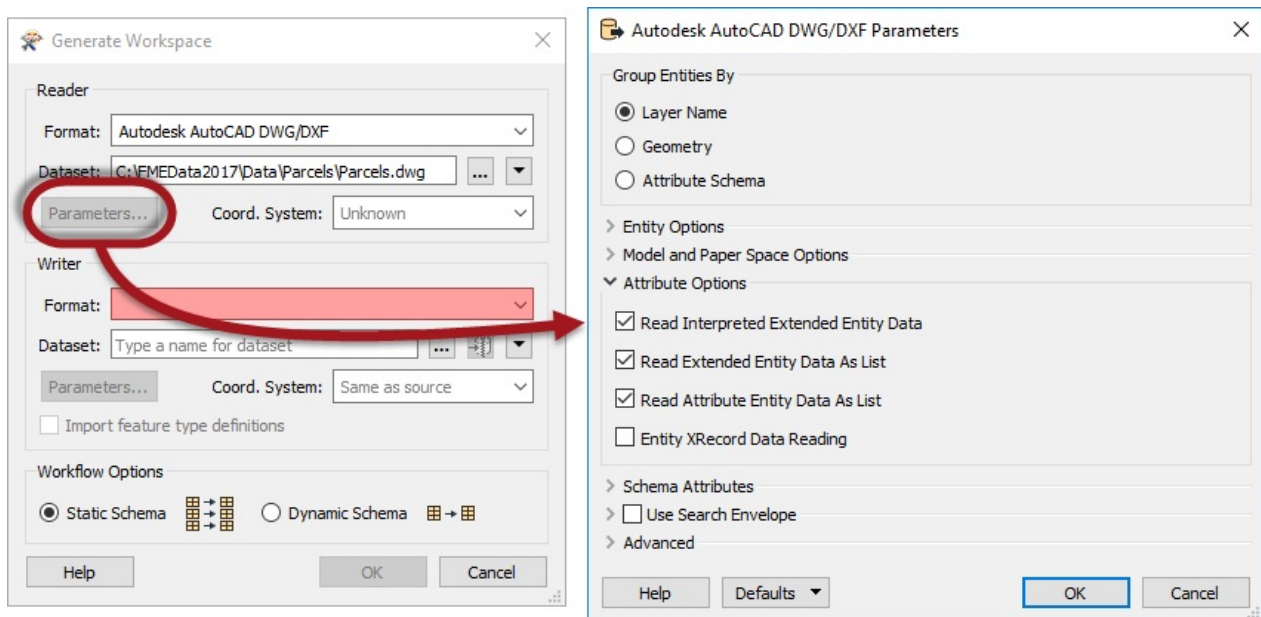
Controlling Readers

Readers are controlled with **reader parameters**.

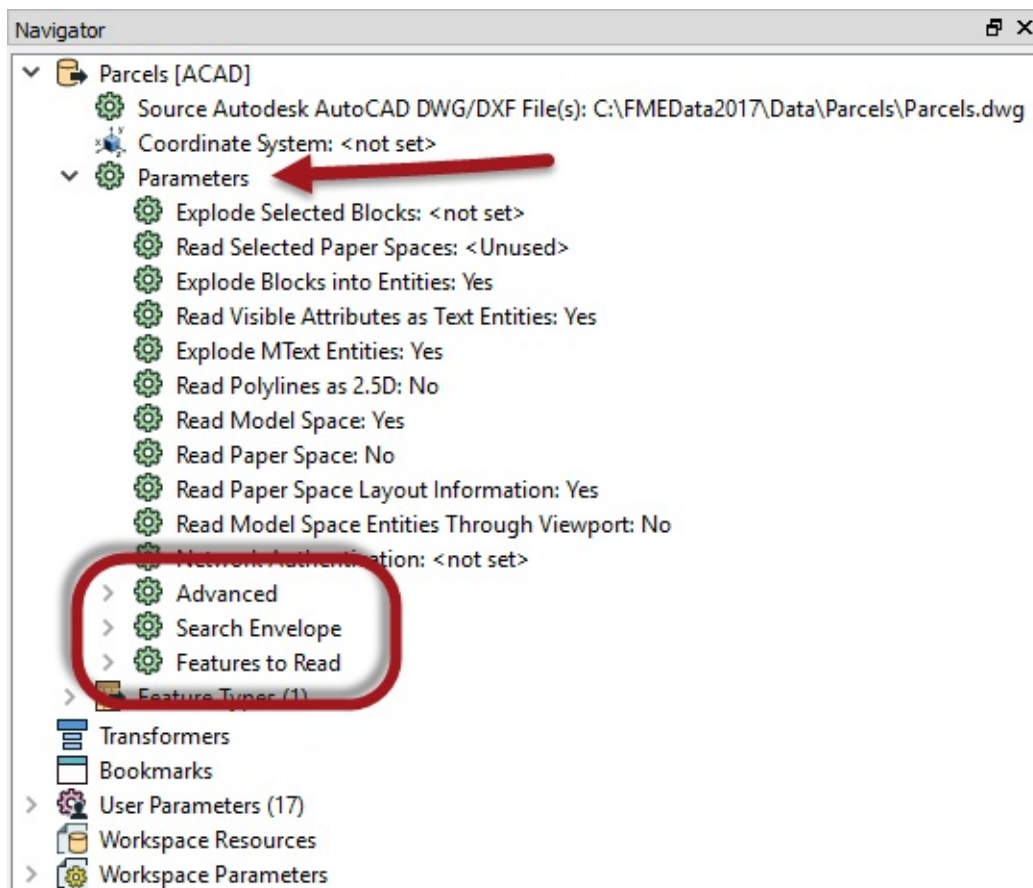
Because parameters refer to specific components and characteristics of the related format, readers of different formats have a different set of control parameters.

Reader Parameters

Reader parameters can be located - and set - in one of two locations. Firstly, these parameters can be found in a dialog when a new workspace is being generated, or a new reader added:



Secondly, after the workspace is generated/Reader is added, parameters are shown and set in the Navigator Window.



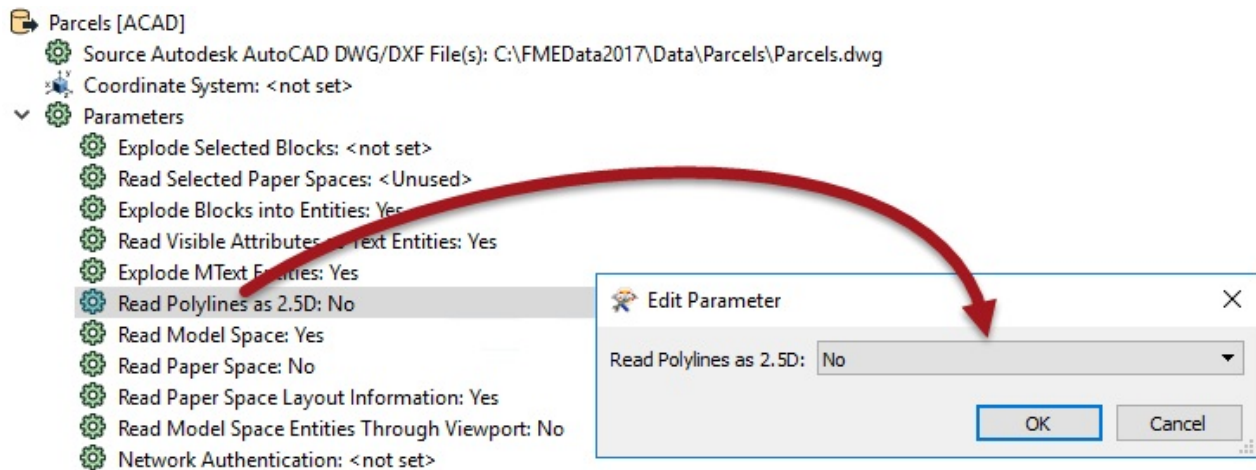
For ease-of-use, basic parameters are listed first, then the remainder are divided into a number of sections: Advanced, Search Envelope, and Features to Read.

TIP

Reader parameters only exist in the Navigator window. They don't appear in the Parameter Editor dialog because there are no reader objects on the canvas to click on.

In general, basic parameters are the ones used most often; advanced parameters a little less so. Search Envelope parameters relate to the geographic area of data to be read, and Features to Read parameters give control over how many features will be read and from which layers (feature types).

To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.



Dr. Workbench says...

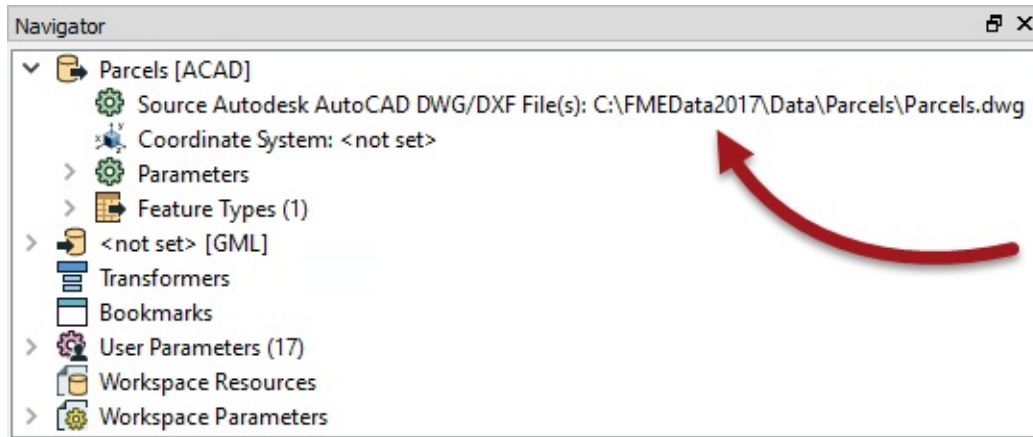
Some reader parameters are ONLY accessible through the Parameters button when you initially create a workspace or add the reader to an existing workspace. That's because they affect how the schema is read and therefore how the workspace is constructed.

It's like preparing a patient for surgery. Once the workspace (patient) is created (prepped) those parameters aren't available because you're past the point where they would have any effect.

Of course, sometimes you get such a parameter wrong, in which case you simply recreate the workspace. Or find yourself a new patient!

Reader Dataset Parameter

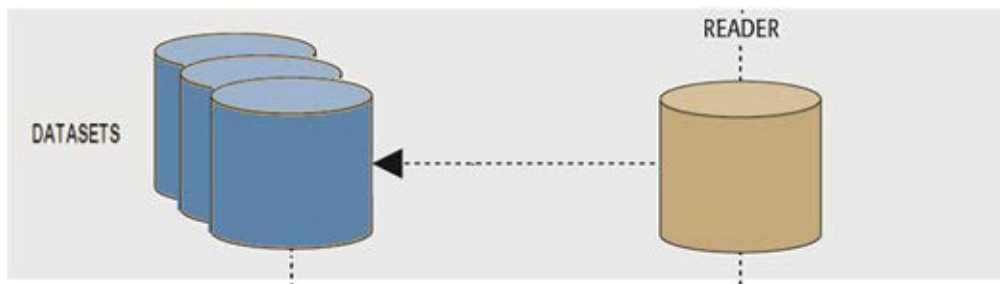
Of course, perhaps the most important parameter is that which defines the dataset to be read. This too is found in the Navigator window:



As usual, double-clicking the parameter opens a dialog for it to be changed. You can change the dataset being read by selecting any other dataset - of the same format - within that dialog.

Dataset Selection: Multiple Files

As long as the format is file-based (i.e. not a database) a reader is capable of reading multiple source datasets:



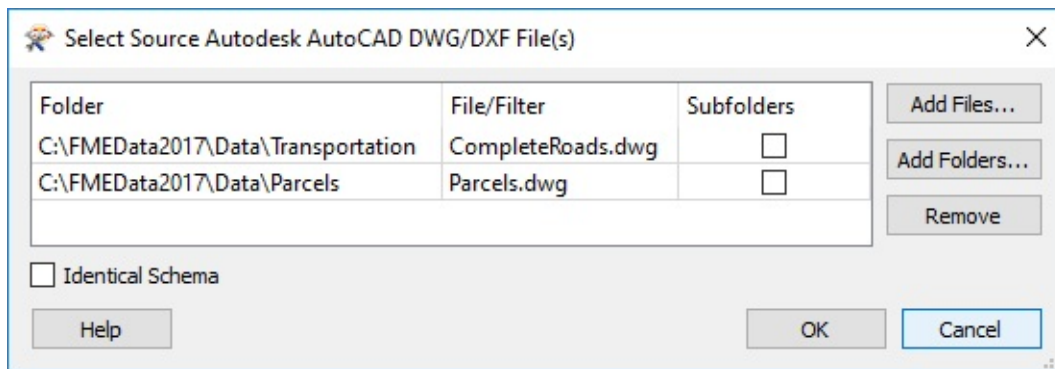
To do this, when selecting the dataset to read simply select multiple files from the source folder, instead of a single file.

Dataset Selection: Multiple Folders

In a similar way, multiple files can be selected from multiple folders, but this requires a slightly different dialog:



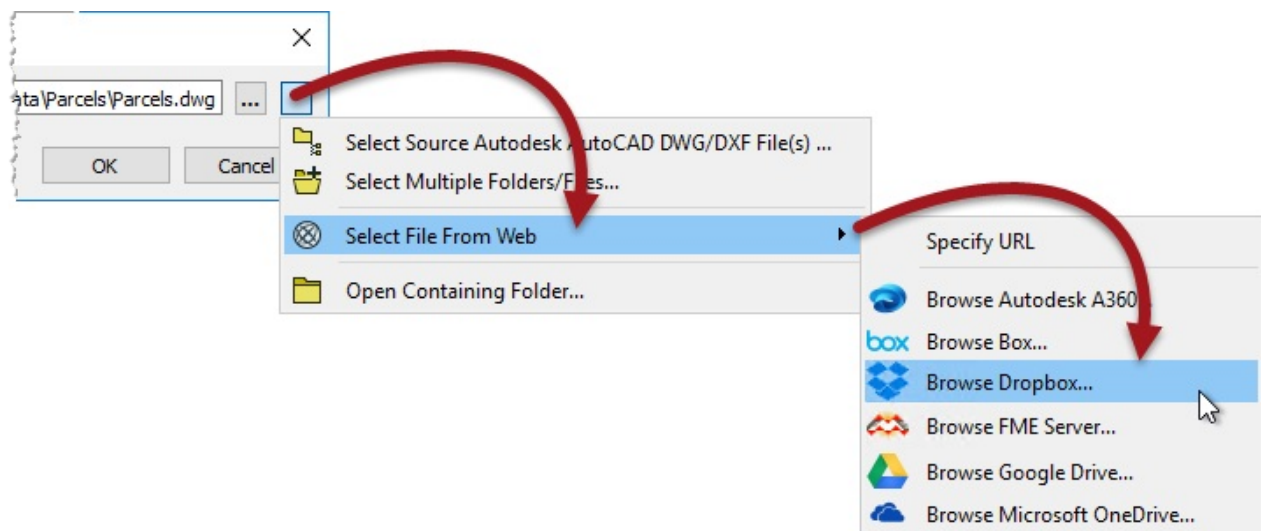
This tool opens an advanced file browser to allow you to select source files that reside in different folders; for example:



You can even select a single folder and have the reader read all files of the correct type from it. This is advantageous because the workspace can continue to function correctly even if the source files change (for example, some are deleted and others are added).

Dataset Selection: Web Sources

Besides local filesystems, it's also possible to select source data from web-based filesystems such as Dropbox or Google Drive:



To take advantage of this requires a pre-defined web connection to the chosen filesystem. If you haven't previously defined such a connection then you will be prompted to add one when choosing this method of selecting source data.

You can also define a connection at any time using Tools > FME Options > Web Connections on the FME Workbench menubar

NEW

Although the ability to enter a URL for a source dataset has existed in FME for some time, the ability to browse web-based filesystems such as Dropbox and Google Drive is new for FME2017

File Dataset Types

When dealing with files it's useful to consider datasets as existing in two different forms: File-based and Folder-based

File-Based Datasets

A file-based dataset is just that: a single file that contains all layers of data. A format of this type has an internal structure that assigns data to different layers within the file.

Good examples of this are:

- **AutoCAD DWG:** Each DWG file is a dataset containing its own set of layers
- **Microsoft Excel:** Each Excel file is a dataset containing a set of sheets

In this scenario the dataset parameter is simply a pointer to the location of the file(s), here a single AutoCAD dataset:



Notice that the dataset is comprised of a number of feature types (layers): Arterial, Collector, NonCity, etc.

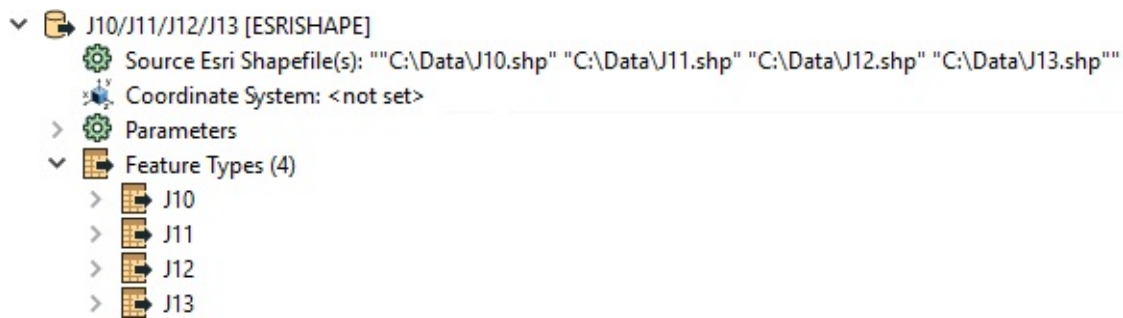
Folder-Based Datasets

A folder-based dataset is still made up of files (of course). However, a format of this type DOES NOT have an internal way to define layers. Therefore each layer becomes a separate file within a folder.

Good examples of this are:

- **Esri Shapefile:** To store different layers requires multiple Shapefiles: e.g. roads.shp and railways.shp
- **Comma-Separated (CSV):** To store separate tables requires two separate CSV files

In this scenario the dataset parameter is a pointer to the individual files, here a set of layers in a shapefile dataset:



Again notice there are a number of feature types; but there is a separate file for each of these.

Sister Intuitive says...

Both File or Folder dataset parameters can be pointers to a Zip file. You simply select the zip file in the source parameter and FME will extract the data when it is being read.

Similarly, a File or Folder dataset can be read directly from a URL. Simply enter the URL into the source parameter. For Folder datasets the URL must point to a zip file containing all of the relevant files.

Exercise 2 Adding Readers	
Data	City Parks (MapInfo TAB) Walking Trail (CSV)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Readers and Reader Parameters
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex2-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex2-Complete.fmw

Let's continue your work on the fundraising walk project.

In case you forgot, the city is hosting a fundraising walk for a major charity and you have been tasked with using FME to put together the data that will form the event map.

In this part of the project we'll add another of the source datasets to the workspace as a reader. This format will need close inspection of the reader parameters to make sure we're reading the data correctly.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 1. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex2-Begin.fmw

2) Add Reader

The existing workspace already has a Reader for MapInfo (parks) data. Now let's add one for the most important dataset - the walk trail. This was walked by a colleague and captured on his GPS device.

Select Readers > Add Reader from the menubar in Workbench. When prompted fill in the following details (but don't press OK):

Reader Format	CSV (Comma Separated Value)
Reader Dataset	C:\FMEDData2017\Data\GPS\WalkingTrail.csv
Coord. System	LL84

The coordinate system setting is below the Dataset parameter, on the right hand side of the dialog. It's important to specify the coordinate system here because we're going to be dealing with lots of datasets with different coordinate systems, and data unflagged with coordinate system information is going to be problematic.

Now press the Parameters button. This brings up a parameters dialog that is rather large and looks like this:

CSV (Comma Separated Value) Parameters

Dataset Parameters

Feature Type Name(s): From Format Name

Fields

Delimiter Character: ,

Field Names Line: 1

Data Start Line: 2

Advanced

Preview

	PointID	Latitude	Longitude	Elevation	Time
2	0	49.2989276	-123.1358735	11.1925050	20080302172152
3	1	49.2989527	-123.1365148	6.8664550	20080302172234
4	2	49.2989245	-123.1367537	6.3858640	20080302172245
5	3	49.2989030	-123.1368376	4.9438480	20080302172249
6	4	49.2988872	-123.1368965	6.8664550	20080302172253
7	5	49.2988165	-123.1370340	6.3858640	20080302172300

Attributes

Attribute Definition

☒ Automatic ☐ Manual

<input checked="" type="checkbox"/> Read	Name	Type
<input checked="" type="checkbox"/>	PointID	string
<input checked="" type="checkbox"/>	Latitude	string
<input checked="" type="checkbox"/>	Longitude	string
<input checked="" type="checkbox"/>	Elevation	string
<input checked="" type="checkbox"/>	Time	string

Schema Attributes

Help Defaults OK Cancel

The dialog includes various parameters for this reader and a preview of what the source dataset looks like. By default the Delimiter Character should be a comma (change it to a comma if it is not) and the Field Names Line parameter should be set to 1.

In the lower part of the dialog - under Attribute Definition - click the button marked Manual. Now change the type of the Latitude attribute to a y_coordinate and the Longitude attribute to an x_coordinate:

Attributes

Attribute Definition

☐ Automatic ☒ Manual

<input checked="" type="checkbox"/> Read	Name	Type
<input checked="" type="checkbox"/>	PointID	string
<input checked="" type="checkbox"/>	Latitude	y_coordinate
<input checked="" type="checkbox"/>	Longitude	x_coordinate
<input checked="" type="checkbox"/>	Elevation	string
<input checked="" type="checkbox"/>	Time	string

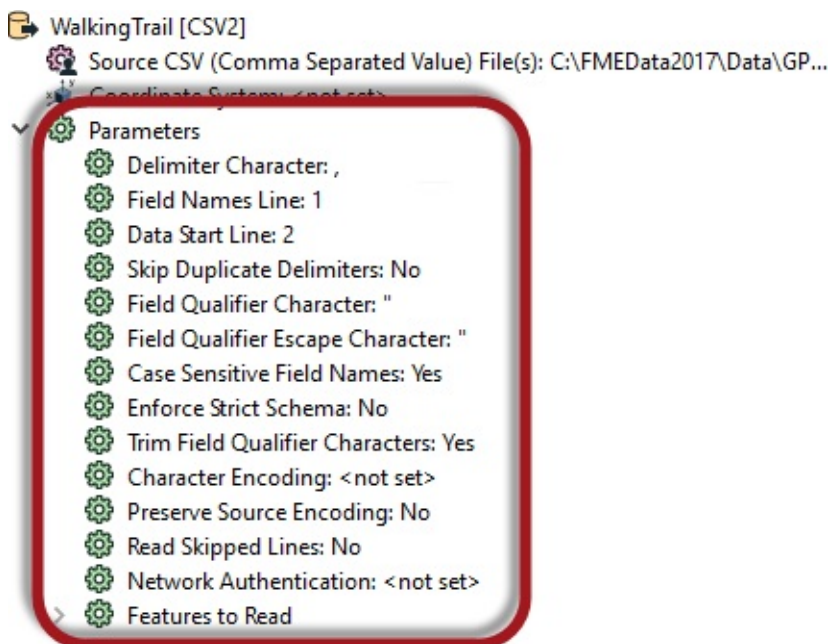
By setting these parameters FME will convert the plain-text CSV data into true spatial features. Click OK to close this dialog and OK again to close the Add Reader dialog.

Mr. Statistics-Calculator says...

Some things just won't wait. It's important to set these parameters now as they affect how the schema is depicted on the canvas. If I mess this up I'd have to delete the reader and re-add it. That's not an efficient use of time and resources.

3) Check Reader Parameters

Locate the CSV reader in the Navigator window and expand the list of parameters to see what there is:



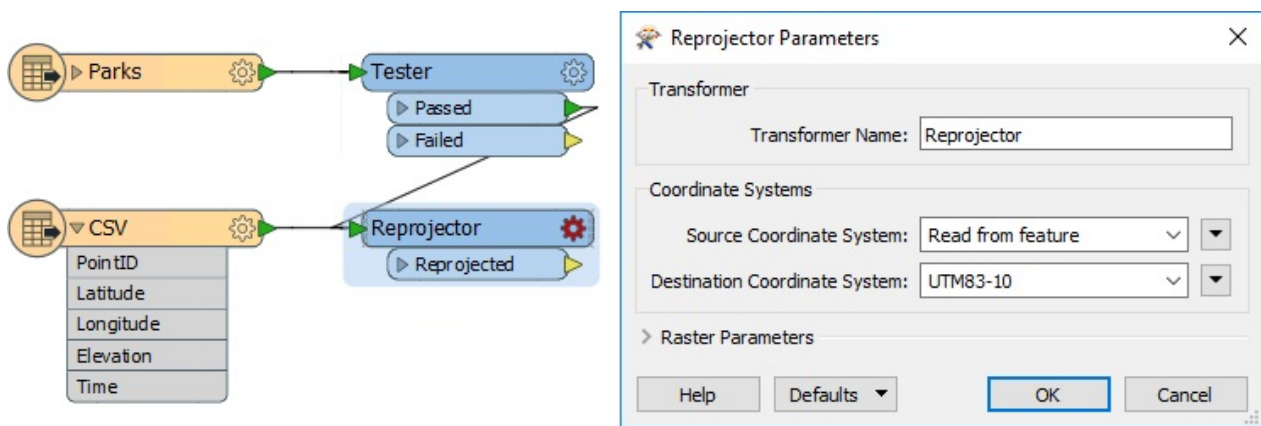
Notice that some of the parameters we set earlier are no longer available. That's because they specified how to define the feature type schema, and it's too late to set them now.

There are other parameters, but none we need to set for this exercise.

4) Add Reprojector

The one issue with the CSV data is that it is in a Latitude/Longitude coordinate system, and not the same UTM system as used by the parks data. Let's fix that by adding a Reprojector transformer to the workspace.

Connect it to the CSV reader feature type and set its parameters to reproject from *Read from feature* to *UTM83-10*:



Also, as in the above image, connect the Tester:Passed output port to the Reprojector.

First Officer Transformer says...

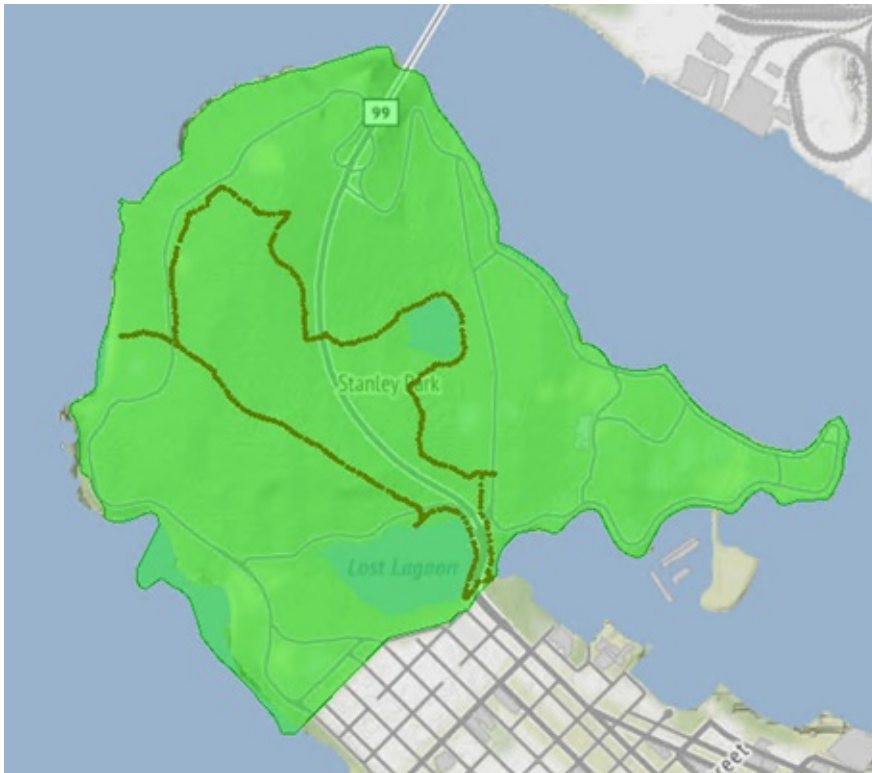
This might seem odd. The Parks data is already in the correct coordinate system, so why reproject it?

The answer is that we're trying to make a tidy workspace. If all features go through the Reprojector there will be fewer connections passing across the screen. We can separate the data out again at a later stage.

We don't need to worry that the Parks data will be wrongly reprojected. It is already tagged with UTM83-10 and FME will recognize this and carry out no action on the data.

5) Run Workspace

Feel free to add Inspector transformers and run the workspace to inspect what we have created so far.



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

CONGRATULATIONS

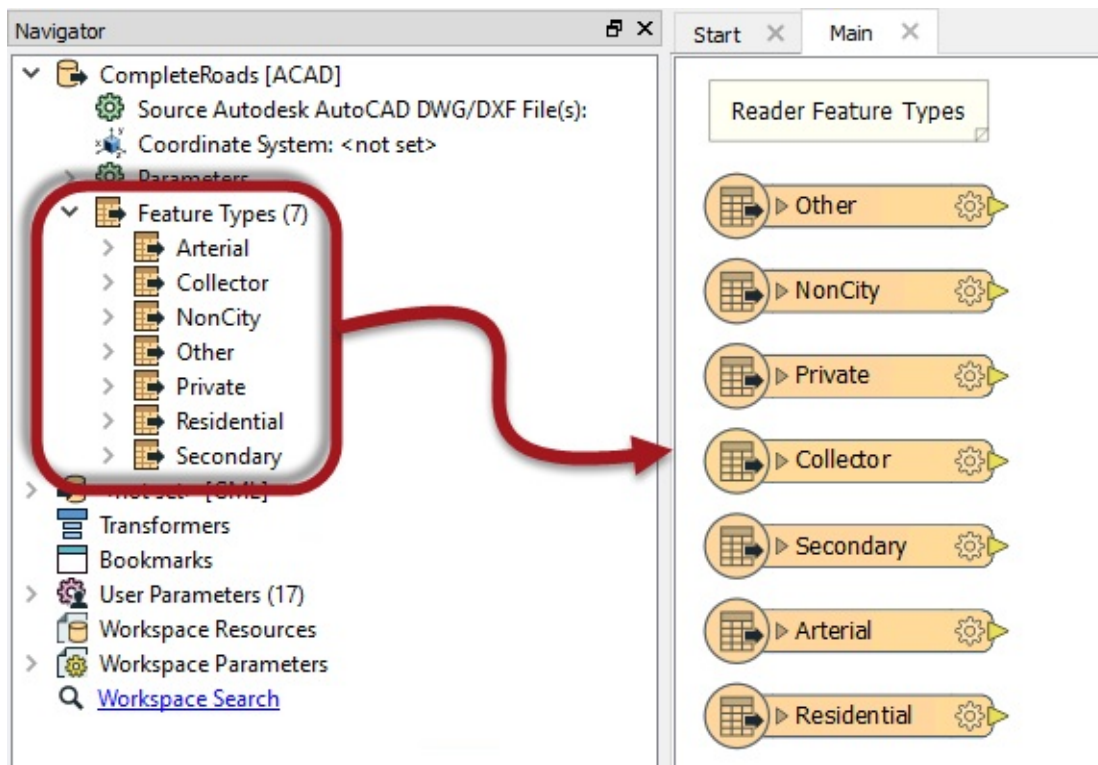
By completing this exercise you have learned how to:

- *Add readers to a workspace*
- *Set parameters when adding a new reader*
- *Set parameters after adding a new reader*
- *Convert CSV attributes into spatial features*

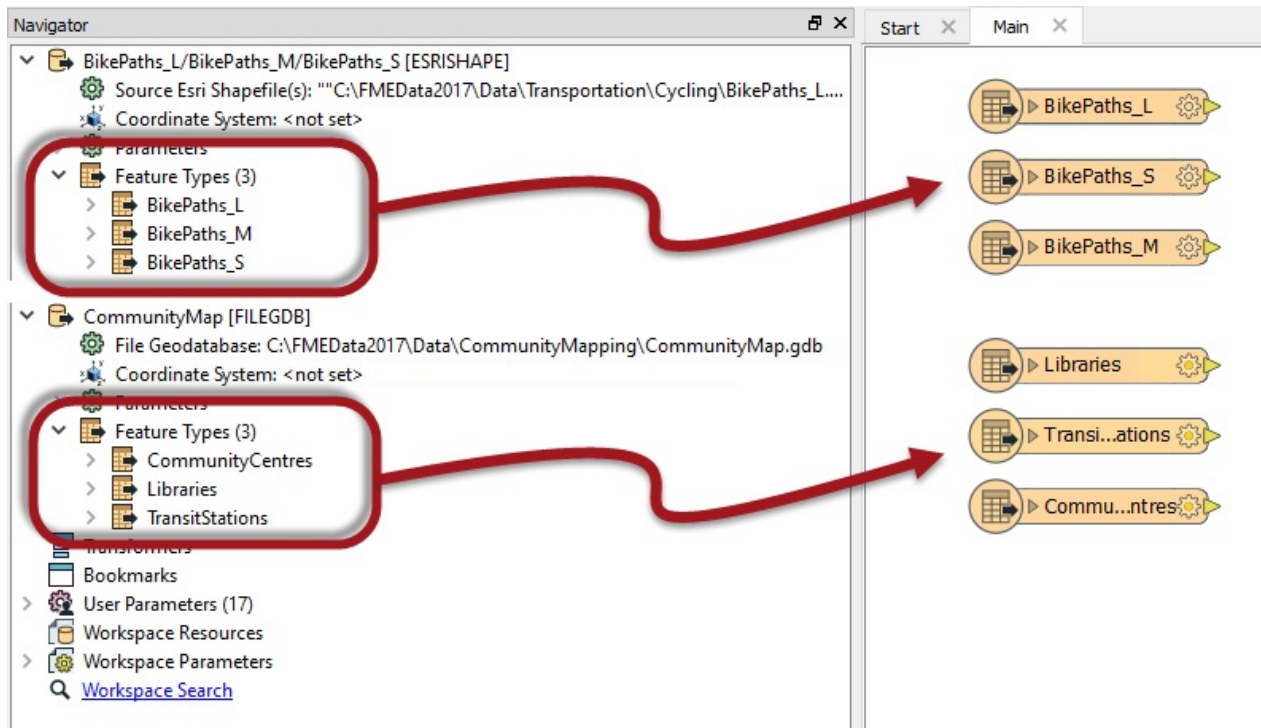
Reader Feature Types

As per the hierarchy diagram, each reader in FME can have a number of feature types. Feature types define the layers being read from a source dataset and the attributes that those layers possess. They are represented as objects on the workspace canvas and also listed in the Navigator window.

In this workspace, a reader is reading a source dataset of roads, with a different layer for each road type. Notice the reader in the Navigator window and how it has a list of feature types; then notice that each of these feature types is also depicted on the canvas:



In this workspace there are two readers (one Esri Shapefile, one File Geodatabase). Each reader has three different feature types:



Notice - and this is where it can be confusing - there is no visible connection between the Reader and its feature types. The connection is there, just not in a visible form.

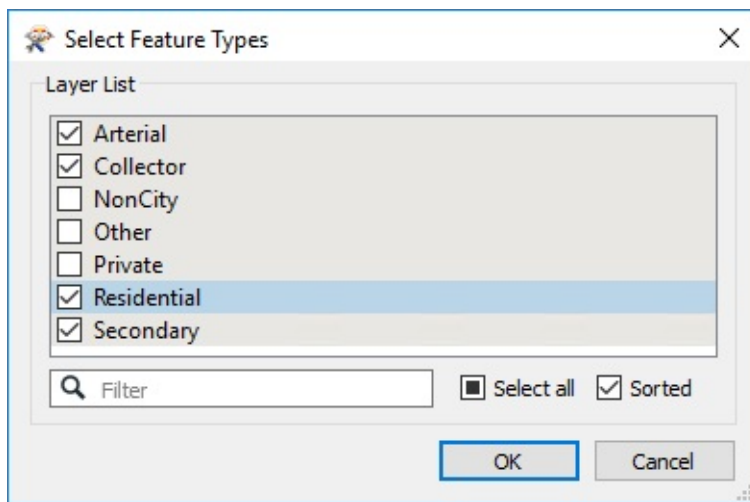
TIP

Clicking on a feature type in the Navigator will highlight it on the canvas - and vice versa

Adding Reader Feature Types

Adding a new reader is the point at which most feature types are added to the translation.

For a file dataset (like AutoCAD DWG) FME scans the source dataset(s) and prompts you which feature types to add to the canvas:

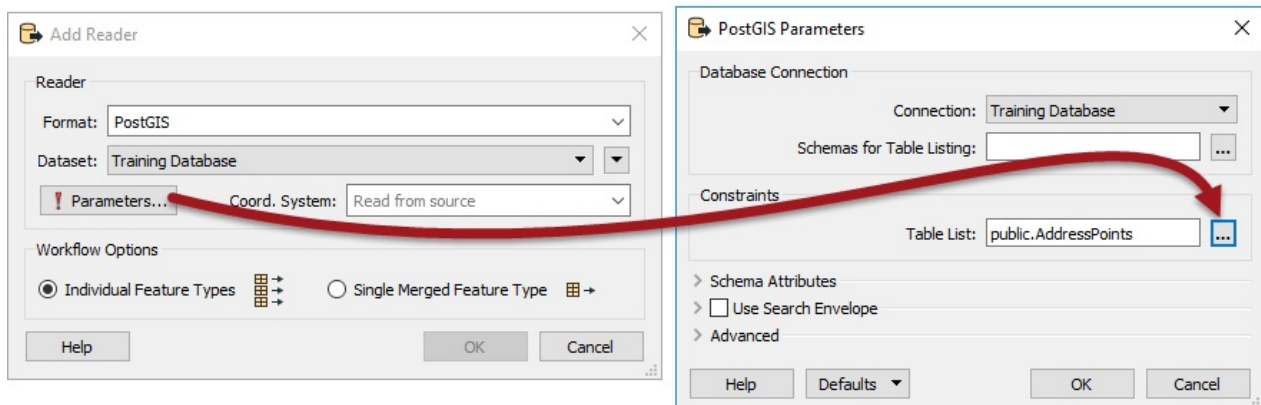


If a feature type is selected, then it will appear in the workspace and data on that layer will be read. If it is not selected then the feature type will not appear in the workspace and the data is discarded at run-time.

For a folder dataset (like Esri Shapefile) the user should just select the files that represent the feature types:

Name	Date modified	Size
BikePaths_L.shp	07/02/2014 11:12 AM	63 KB
BikePaths_M.shp	07/02/2014 11:12 AM	53 KB
BikePaths_S.shp	07/02/2014 11:12 AM	15 KB

For a database (like PostGIS) the user should click the parameters button and use the Tables to Read parameter:



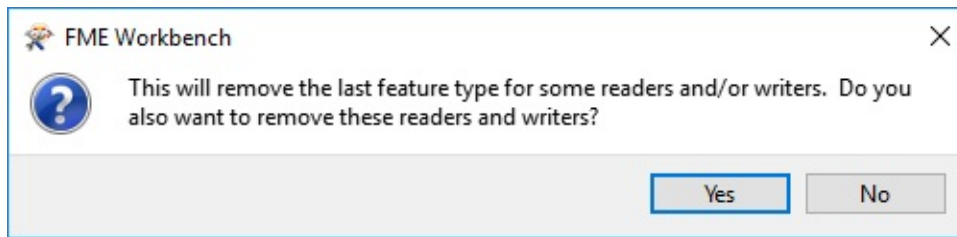
Note that it's not necessary to select all feature types that exist in a dataset. If you don't want to read a particular layer of data, don't select it. This way the feature types that you need to be read are all represented on the canvas, and feature types you don't need are not.

Removing Feature Types

A user might remove a feature type if there is a source dataset layer that they no longer wish to read.

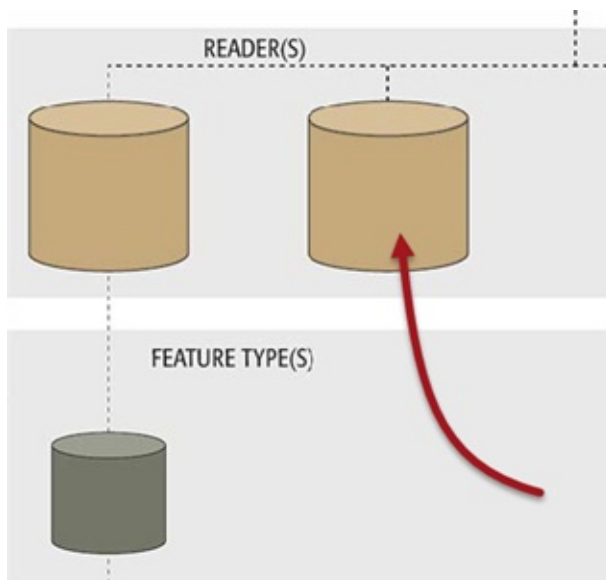
Feature types can be deleted by selecting them on the canvas and pressing the delete key. They can also be deleted using the *Remove Feature Types* tool on the menubar:

Whenever all feature types are deleted from a reader then FME will prompt the user to decide whether to remove the reader component as well.



It makes sense to remove the reader because if there are no feature types you wish to keep, why would you still wish to read the dataset at all? So Yes is the most common choice here.

If you answer No, then the feature types are all removed, but the reader is left in the translation. We call this a "dangling" reader because it has no children in the hierarchy.



Chef Bimm says...

A dangling reader isn't that useful, and the workspace should definitely not be run in this condition!

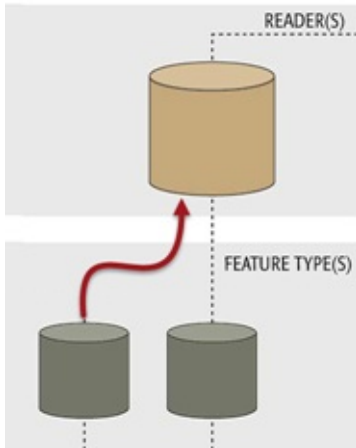
Performance suffers because - in some cases - all the source data is still being read, yet discarded immediately.

It's like ordering food in a restaurant for a friend who you know is not coming: a waste of time and resources!

Importing Reader Feature Types

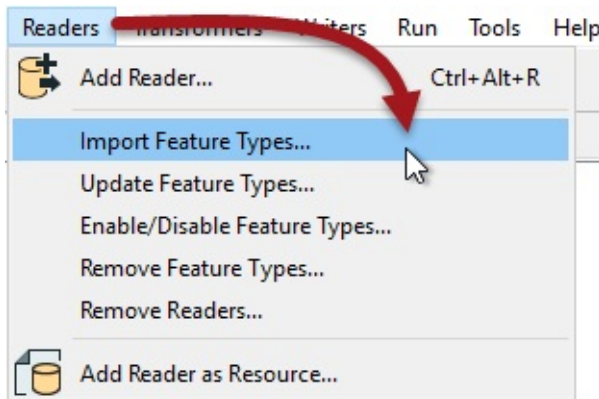
A further tool under the Reader menu is labelled Import Feature Types.

Importing feature types is like when you add a new reader; you pick the format and source dataset and FME gives you a list of feature types to add. However... instead of creating a new reader, the feature types are added to a reader that already exists.



One reason to import feature types is to read extra feature types from a dataset. Maybe the original author chose not to read those feature types when they added the reader, or maybe the dataset was edited to give it extra layers.

The way to do this is to use the menu tool Readers > Import Feature Types:



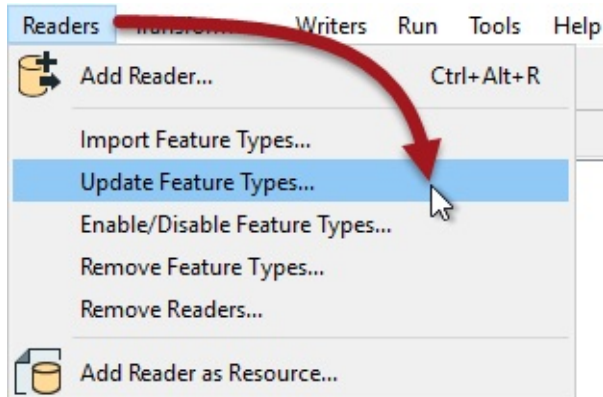
For example, the user has created a workspace with a reader that reads a dataset called 'Transportation'. The only feature type chosen at the time of creation was a layer called 'Roads'. If the workspace now needs to read a second layer from the dataset (say 'Railway') the import tool can be used to add it to the existing reader.

Firefighter Mapp says...

There's no tool to add a reader feature type in a manual way. That's because reader feature types are supposed to represent what already exists in a source dataset - and it's easier to just import a definition from one of them.

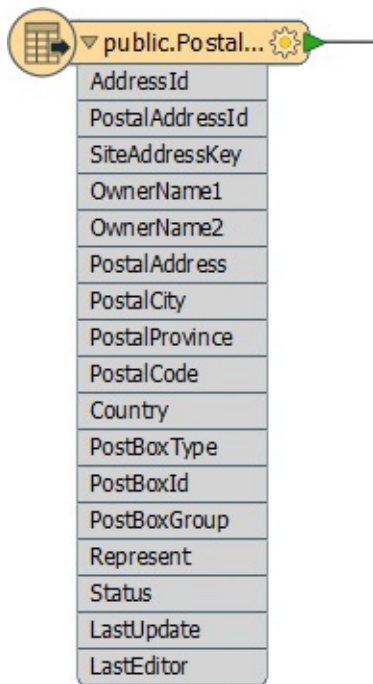
Updating Reader Feature Types

A final tool for reader feature types allows you to update them:



When you have created a workspace and more tables are added, then the Import tool is the one to use. However, when the schema of an existing table has changed, then the update tool is the one to use.

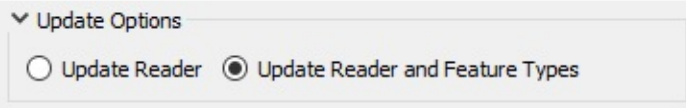
For example, I have a source feature type with this attribute schema:



The database administrator decides that the three PostBox attributes are no longer required and removes them from the database table. To fix my FME workspace to match, I would use the Update Feature Types tool.

.1 UPDATE

The new Update Reader functionality in FME2017.1 also has the option to update the reader feature types too:



▼ Update Options

☐ Update Reader ☒ Update Reader and Feature Types

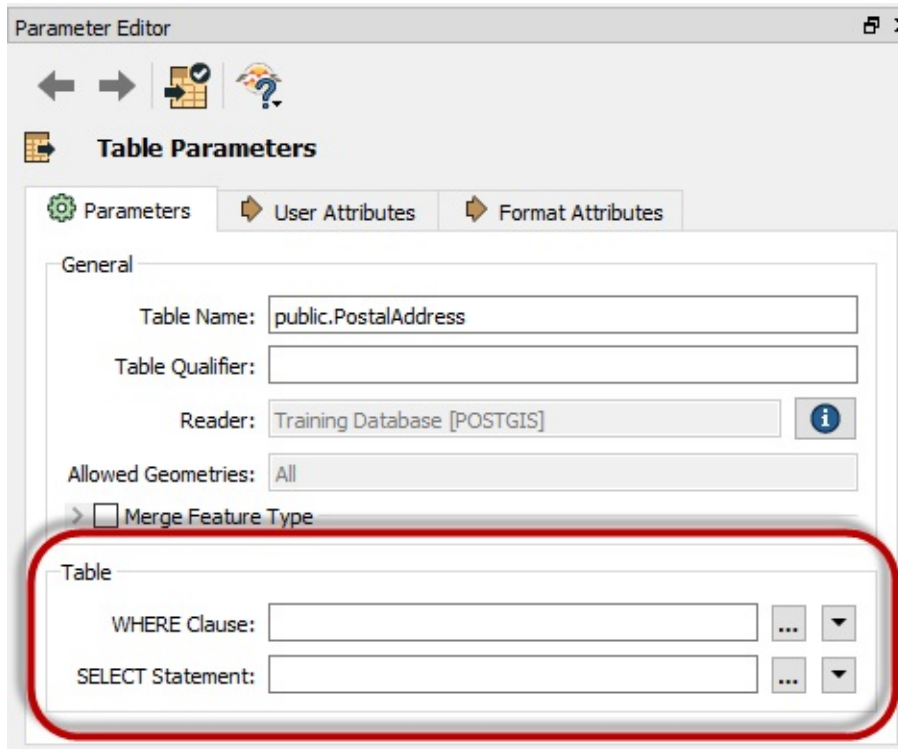
Miss Vector says...

Let's see if you can figure out which tool should be used in which scenario? Here are four scenarios and four tools or settings. Connect the scenario to the correct tool/setting.

Scenario	Tool
Source filename changes	Add Reader
Attribute type changes in database	Import Feature Type
Additional database table needs reading	Update Feature Type
New file dataset needs reading	Source Dataset Parameter

Reader Feature Type Parameters

Just like readers, feature types have their own set of parameters that control how that feature type (layer/table) is being read. These parameters appear both in the Parameter Editor window and in the Feature Type Properties dialog:



There are general parameters that appear for every feature type, but some feature types have additional parameters below that.

The important thing is that these parameters only apply to a single feature type, whereas a reader parameter would apply to all feature types.

For example, the above screenshot is from a PostGIS database feature type with a parameter to define a WHERE clause. This is a Feature Type parameter because each table might require a different WHERE clause.

Conversely, there is no password parameter for this database table, because authentication applies to the entire database, not the individual tables, so authentication parameters are at the reader level (or in a database connection).

TIP

As with readers, feature type parameters vary by format, and some formats have no extra parameters at all at this level.

Exercise 3 Reader Feature Types	
Data	City Parks (MapInfo TAB) Walking Trail (CSV) Food Vendors/Water Fountains (File Geodatabase)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Handling and Controlling Reader Feature Types
Start Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex3-Begin.fmw
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex3-Complete.fmw

Let's continue your work on the fundraising walk project.

In case you forgot, the city is hosting a fundraising walk for a major charity and you have been tasked with using FME to put together the data that will form the event map.

In this part of the project we'll add another of the source datasets to the workspace.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2. Alternatively you can open C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex3-Begin.fmw

2) Add Reader

The existing workspace already has various Readers; now it needs one for loading a layer of food vendor data. This is stored in the Community Mapping Geodatabase

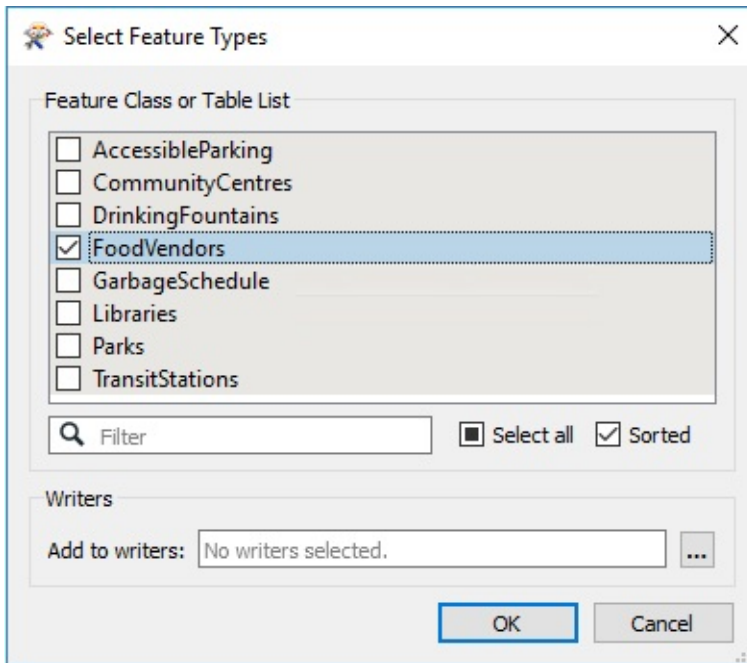
Select Readers > Add Reader from the menubar in Workbench. When prompted fill in the following details:

Reader Format	Esri Geodatabase (File Geodb API)
Reader Dataset	C:\FMEData2017\Data\CommunityMapping\CommunityMap.gdb

.1 UPDATE

In FME2017.1 the format is now called Esri Geodatabase (File Geodb Open API)

This dataset contains several tables, but we only need one of them. So, when prompted, deselect all feature types and leave only FoodVendors selected:

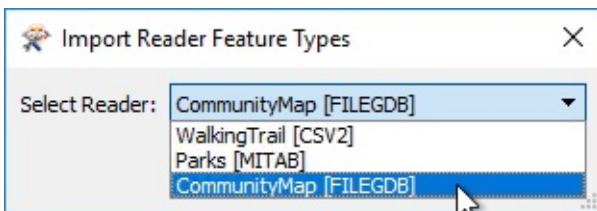


3) Import DrinkingFountains Feature Type

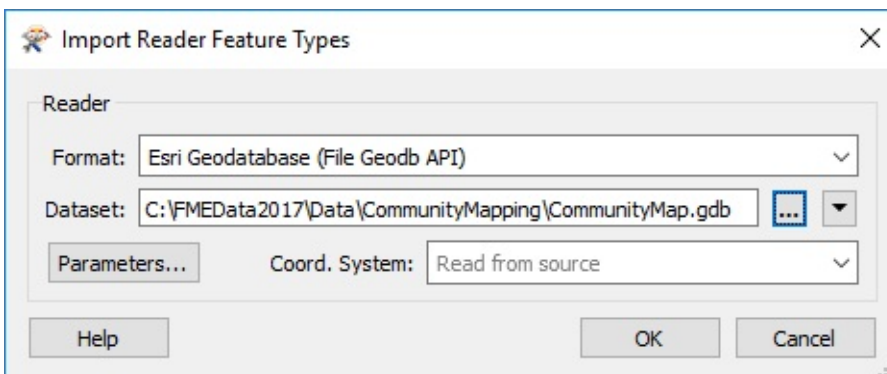
No sooner have you added the FoodVendors layer to the workspace than the telephone rings. It is the event organizers. As well as showing food vendors on the map, they now want to also show the location of drinking fountains.

Muttering under your breath, Click Readers > Import Feature Types on the menubar.

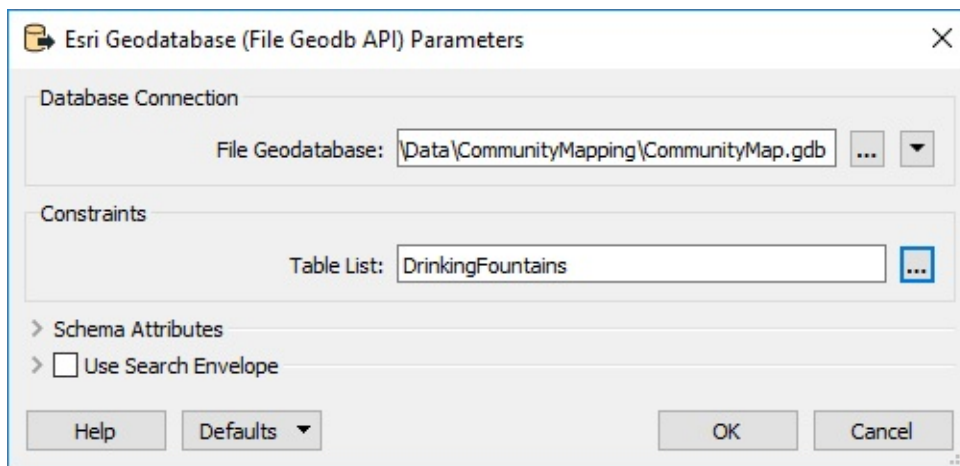
The first dialog that pops up will ask you which reader you want to import the feature type to. Select the Community Mapping Geodatabase:



Now you are prompted which dataset you wish to import the feature type from. Enter the details (as above) for the Community Mapping Geodatabase (but don't click OK yet):



Before clicking OK, click the Parameters button. In the parameters dialog select the DrinkingFountains table to read:



Click OK and OK again. This will add the DrinkingFountains feature type to the workspace.

4) Remove FoodVendors Feature Type

Again the phone rings. This time the organizers say they have changed their mind again and no longer need the FoodVendors table at all.

Well, this is a very simple fix. Put down the phone, click on the FoodVendors feature type, and press the delete key. The feature type is now removed from the workspace.

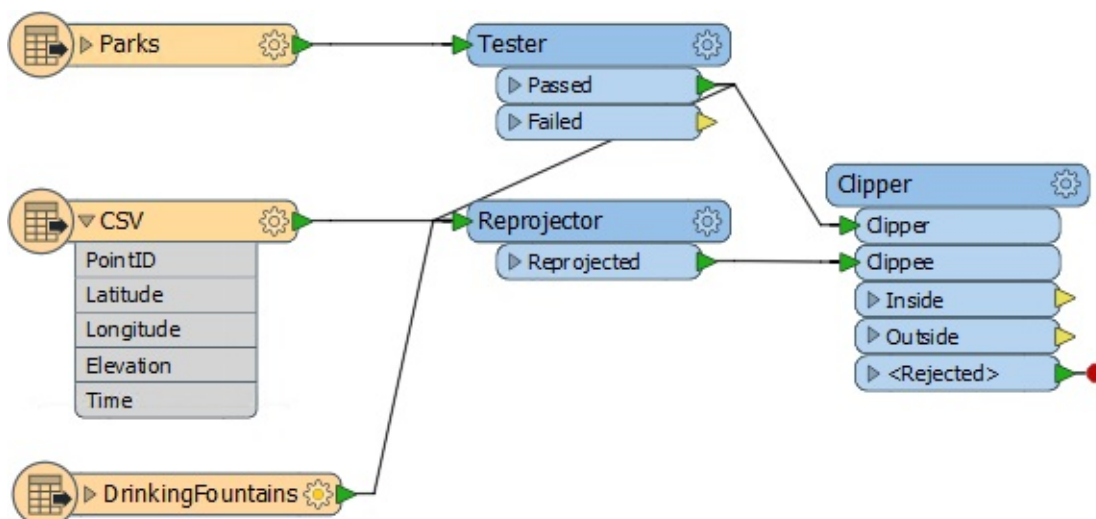
5) Connect to Reprojector

Like the parks data, we don't really need to reproject the Geodatabase data - it is already in UTM83-10 - but it makes for a tidier workspace, so connect the DrinkingFountains feature type to the Reprojector transformer.

6) Add Clipper

If you recall the Community Maps dataset covers the entire city, and yet we're restricting the map to the park boundaries. To cut the excess data away we'll use a Clipper transformer, so add one to the canvas. Then...

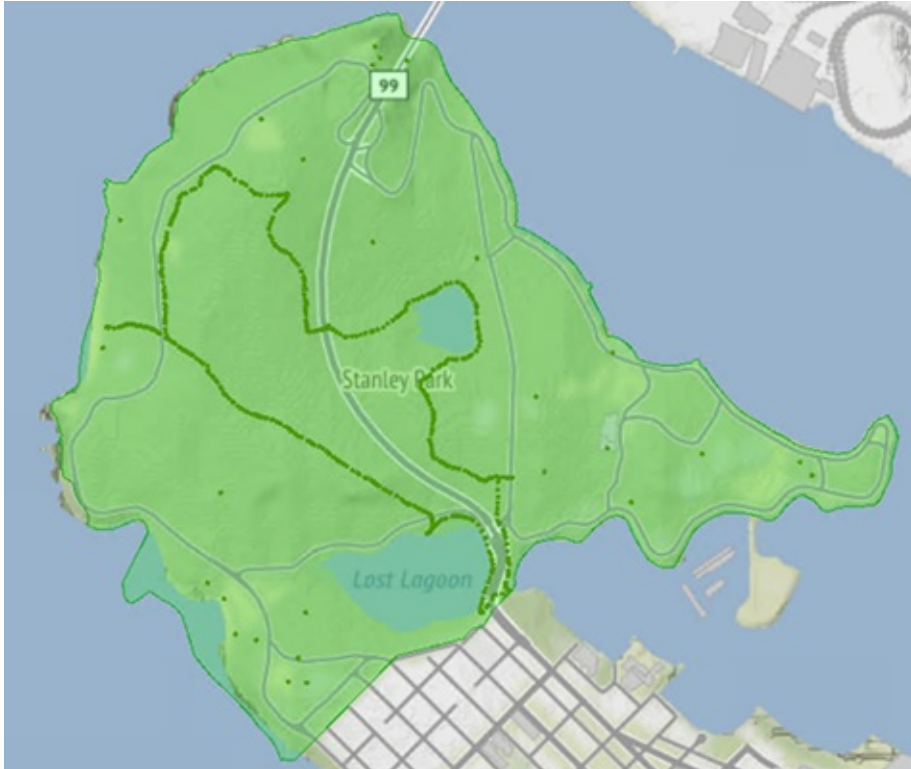
- Connect the Tester:Passed port to the Clipper:Clipper port.
- Connect the Reprojector:Reprojected port to the Clipper:Clippee port:



You can check the transformer parameters, but the defaults should be fine for this project. Again, all data is being routed through the Clipper, even if it doesn't need to be so, in order to make a tidier workspace.

7) Run Workspace

Again, add Inspectors and run the workspace to ensure the result is what we are expecting.



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Import a feature type from a dataset into a Reader in the workspace*
- *Delete feature types from a workspace*
- *Use the Clipper transformer to clip data*

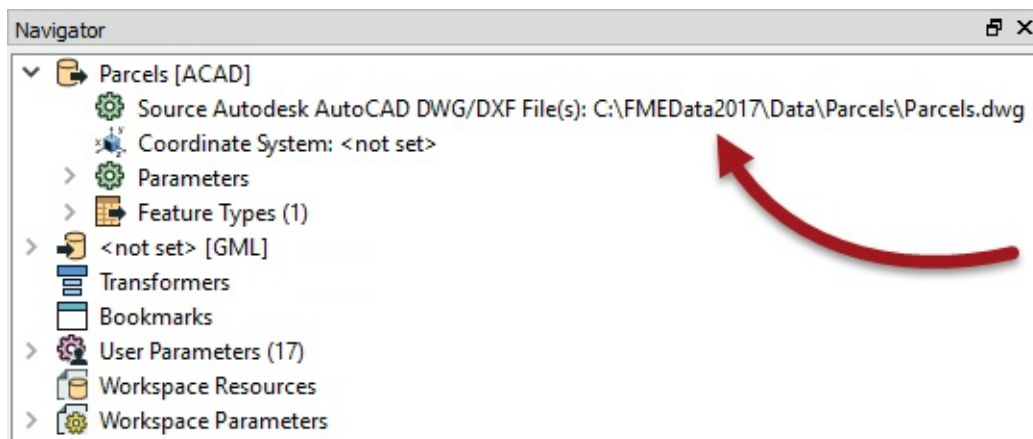
Managing Datasets and Feature Types

We've already seen that when you add a reader to a workspace its feature type schemas are greyed out. That's to stop edits being made and ensure that the schema correctly matches the source data.

However, there is another way in which the reader schema can become out of sync with the source datasets it claims to represent.

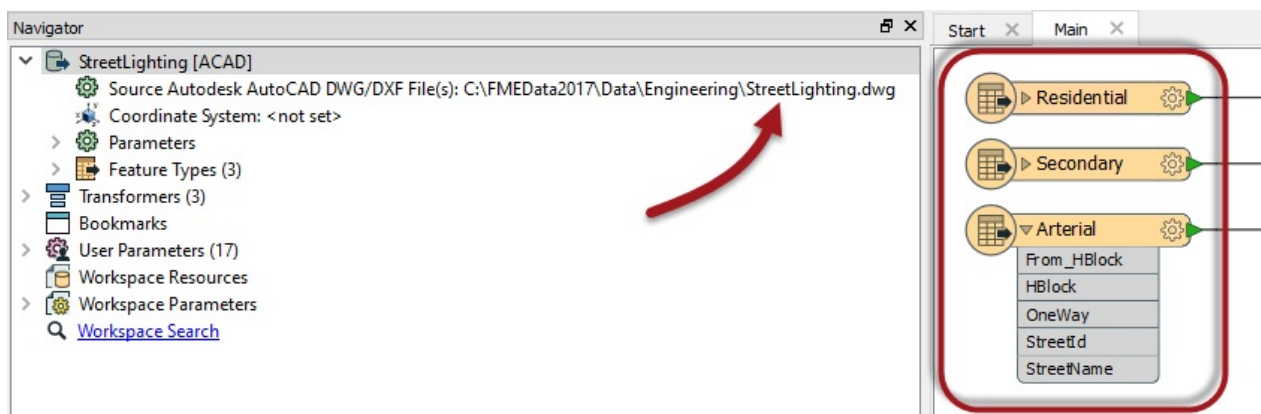
Changing Source Dataset Parameter

As we have seen, each reader in a workspace contains a source dataset parameter:



Using this parameter to select a different source dataset is another way for workspace schemas to become out of sync with the data.

For example, here an AutoCAD DWG reader was added for a roads dataset, containing layers and attributes for Residential, Secondary, and Arterial roads:



However, notice that the user has changed the source dataset parameter to read a dataset of street lighting. That dataset has different feature types and attributes to the roads data. Hence the workspace schema is not correct for the data being read.

This is a bigger issue than just leaving out a required feature type, and has severe consequences for the translation.

Consequences

It's important to remember that the original dataset establishes the basis for the reader schema definition, and that problems can arise if subsequent datasets do not conform to this original schema.

- If the feature types being read do not match one in the workspace, they are automatically discarded; i.e. reader feature types defined in the workspace act as a type of filter through which incoming data must pass.
- If the attributes being read do not match those defined in the workspace, they can be dropped or - at best - will be invisible to the workspace.

When incoming feature types do not find a match they are called "unexpected", and so the filter is called the ***Unexpected Input Remover***.

Exercise 4 Unexpected Input	
Data	City Parks (MapInfo TAB) Walking Trail (CSV) Water Fountains (File Geodatabase) Car Parking (OpenStreetMap) Roads (OpenStreetMap)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Handling Unexpected Input
Start Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex4-Begin.fmw
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex4-Complete.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex4-Complete.1.fmw

Let's continue your work on the fundraising walk project.

In this part of the project we'll add some OpenStreetMap format data to the workspace.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 3. Alternatively you can open C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex4-Begin.fmw

2) Add Reader

Let's add some more data. This time we'll add the car parking.

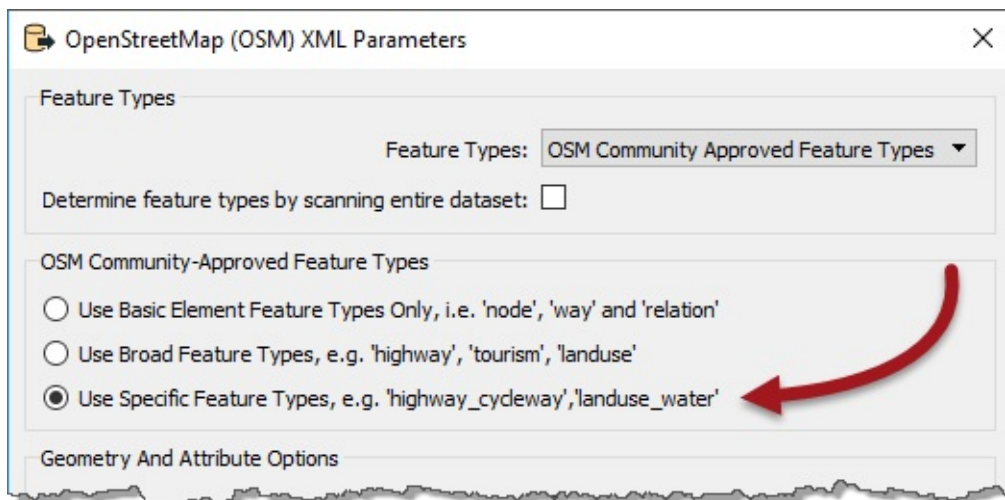
.1 UPDATE

The OpenStreetMap reader has changed significantly in FME2017.1. Therefore the next step has two sets of instructions; one for 2017.0 (using OpenStreetMap) and the other for 2017.1 (using GML).

In **FME2017.0**, Select Readers > Add Reader and choose the following data:

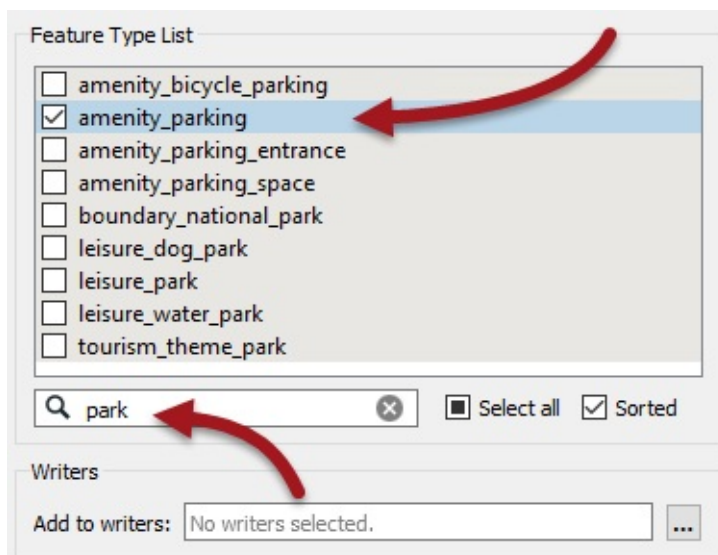
Reader Format	OpenStreetMap (OSM) XML
Reader Dataset	C:\FMEData2017\Data\OpenStreetMap\amenity.osm
Reader Parameters	Feature Types: Use Specific Features

The parameter is important because it specifies what feature types we will be presented with:



If we go with the Broad feature types, we will be presented with a broad set of layers and car parks will be hidden inside Amenities. If we go with Specific feature types we can select only car parks by themselves.

So, click OK to close this dialog and OK to add the reader. When prompted, deselect all feature types and leave only amenity_parking selected:



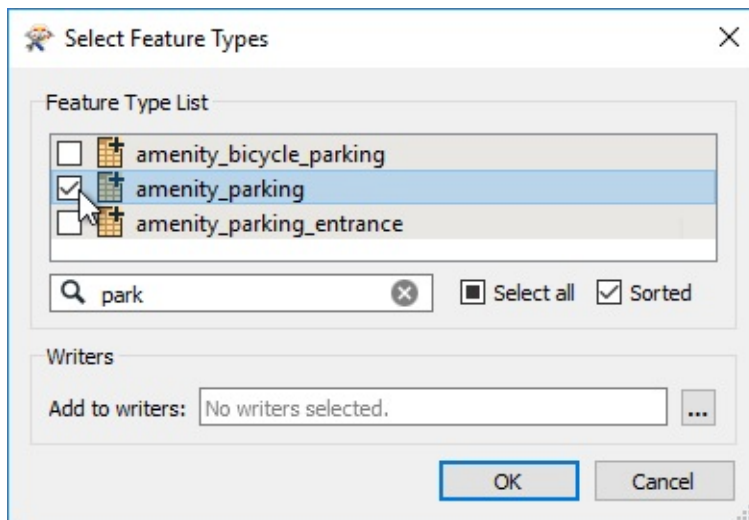
TIP

There are a lot of different amenity feature types in this dialog so using the Filter setting (as above) is useful.

In **FME2017.1**, Select Readers > Add Reader and choose the following data:

Reader Format	GML (Geography Markup Language)
Reader Dataset	C:\FMEData2017\Data\OpenStreetMap\amenity.gml

Click OK to close the dialog and when prompted browse through the features list and pick out amenity_parking:



Click OK to close this dialog and add the reader.

TIP

Why GML? It's not that the OSM reader no longer works in 2017.1, it just doesn't work in the best way needed to demonstrate unexpected input.

3) Connect to Reprojector

As with all the other data, connect the new amenity_parking feature type to the Reprojector transformer. If you inspect this source data you'll see that it is automatically tagged as LL84, so we don't need to define that manually.

4) Update Reader










We also need some data from a different dataset (again this differs for 2017.0 and 2017.1)

In **2017.0** we need to also read an OpenStreetMap dataset called highway.osm

We could add another reader, but we already have an OSM reader so why not use that? Locate the OSM reader in the Navigator window and double-click on the Source OpenStreetMap File(s) parameter:



In the dialog that opens, click the browse button. Now select both the amenity and highway osm files; we're resetting this parameter so if you don't select the amenity file as well it will be dropped.


Name	Date modified	Type	Size
 aeroway.osm	11/23/2016 12:47 ...	OSM File	11 KB
 amenity.osm	11/23/2016 12:47 ...	OSM File	526 KB
 barrier.osm	11/23/2016 12:47 ...	OSM File	66 KB
 building.osm	11/23/2016 12:47 ...	OSM File	4,696 KB
 craft.osm	11/23/2016 12:47 ...	OSM File	2 KB
 highway.osm	11/23/2016 12:47 ...	OSM File	2,012 KB
 historic.osm	11/23/2016 12:47 ...	OSM File	2 KB
 junction.osm	11/23/2016 12:47 ...	OSM File	1 KB
 landuse.osm	11/23/2016 12:47 ...	OSM File	374 KB

In **2017.1** we will read a GML dataset called highway.gml

As with OSM, we could add another reader, but we already have a GML reader so why not use that? Locate the GML reader in the Navigator window and double-click on the Source Geography Markup Language (GML) File(s) parameter:

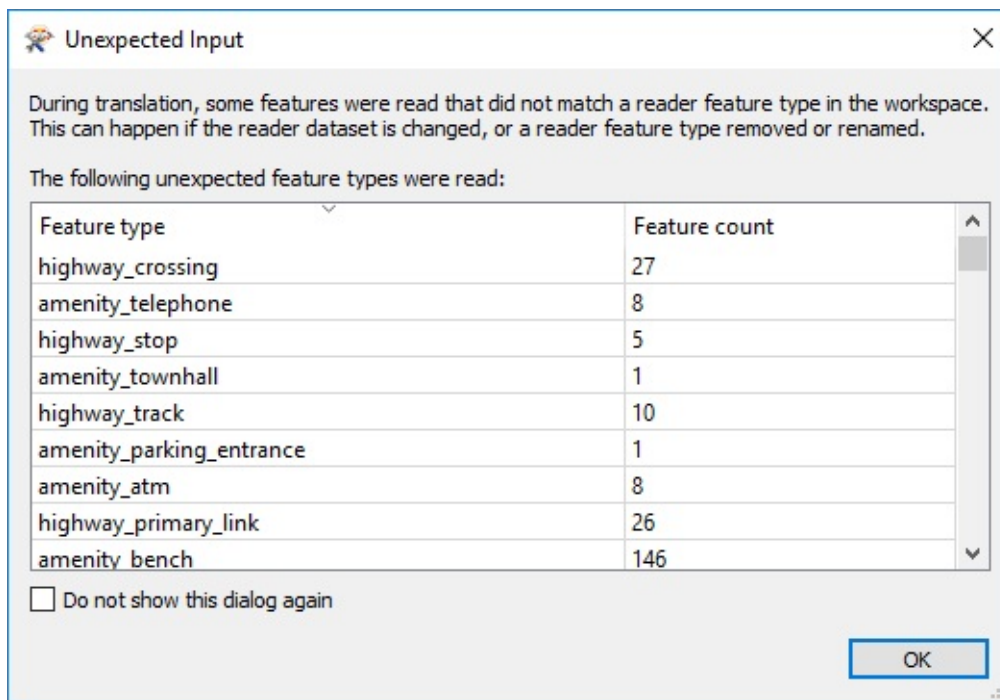


In the dialog that opens, click the browse button. Now select both the amenity and highway GML files; we're resetting this parameter so if you don't select the amenity file as well it will be dropped.

Name	Date modified	Type	Size
 amenity.gml	8/4/2017 12:04 PM	GML File	2,434 KB
 highway.gml	8/4/2017 12:16 PM	GML File	10,243 KB

5) Run Workspace

Re-run the workspace. The data appears to be read correctly, but we get a pop-up dialog like so:



In fact, the log window also reports this information:

During translation, some features were read that did not match a reader feature type in the workspace. This can happen if the reader dataset is changed, or a reader feature type removed or renamed.

This is worrying. A lot of these aren't layers we wanted to read, but we did want to read some highways data and this is also appearing in the list of "unexpected input".

What on earth can be going on? We'll find out shortly...

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Update a reader's source dataset parameter*
- *Select multiple files in a reader's source dataset parameter*
- *Detect what feature types are Unexpected Input*

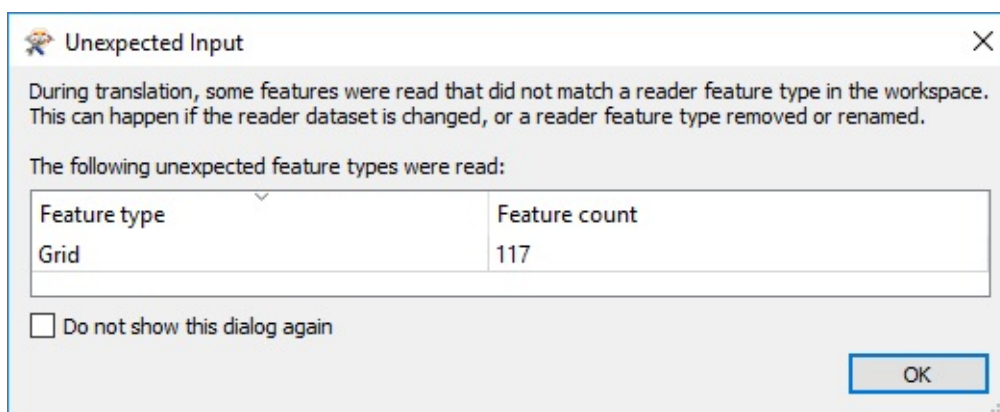
The Unexpected Input Remover

Every time FME reads a dataset, it checks the feature types inside that dataset to ensure that they are all defined within the workspace schema. If there are feature types that exist in the dataset, but do not exist in the workspace, then features are classed as "unexpected" and filtered out by a function called the Unexpected Input Remover.

The actions of the Unexpected Input Remover are reported in the Log file:

```
STATS |Router and Unexpected Input Remover(RoutingFactory): Tested 3995 input feature(s), wrote 3995 output feature(s): 957 matched merge filters, 957 were routed to output, 3038 could not be routed
```

...and also through a dialog that opens at the end of a translation:



Notice how each feature type that fails to match is listed along with a count of the affected features. Also notice the checkbox that allows this report to be turned off in future translations.

Dr Workbench says...

*It's important to consider that the feature types in a workspace might be **deliberately** different to those in the source dataset.*

For example, a user may have purposely left a feature type out of a workspace when it was generated, or may have decided a feature type was no longer required and so deleted it. In that scenario, the Unexpected Input dialog may still pop up, but can be safely ignored as the user deliberately requires this behaviour.

So this dialog is considered a reminder rather than an error, and not always an indication something is wrong.

Unexpected Input and Dataset Type

Remember that there are two different types of file dataset: *file-based* and *folder-based*.

For **file-based datasets**, if there is a layer **in the file** that is not represented on the canvas then FME treats the layer as "unexpected" and drops it from the translation.

This most commonly happens when the layers in the source data change, or a completely different file is selected.

For **folder-based datasets** - where layers are stored as separate files in a folder - if there is a **selected file** that is not represented on the canvas, then FME treats the file as unexpected and drops it from the translation.

This is most often a problem when a set of tiled datasets are being read, as each tile is a separate file (feature type) and needs a separate definition if it is to be allowed into the translation.

WARNING

Unexpected input is not reported in two cases: databases and AutoCAD files.

For databases it's because a user is unlikely to want to read every single table in the database; so it's not unexpected for them to miss some out.

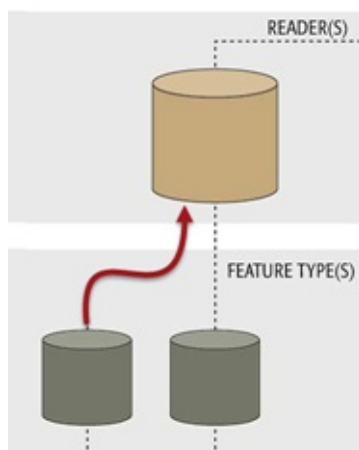
AutoCAD won't report mismatched features types as "unexpected", but it will report them in the log window:

AutoCAD Reader: Features for feature type 'Arterial' will be skipped based on the feature types requested

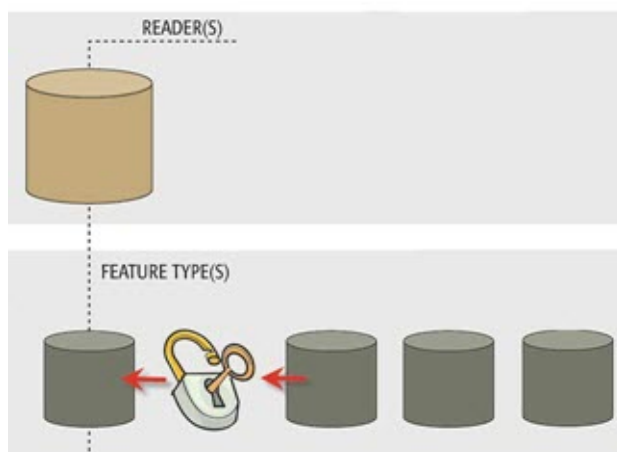
Dealing with Source Feature Types

Assuming the Unexpected Input dialog does indicate a problem, there are two different methods within Workbench by which to resolve the issue:

- Add the missing feature types
In other words, feature types are missing; so let's add them. The **Import Feature Type** tool can be used to do this.



- Relax the filtering process
In other words, allow unexpected feature types to pass through an existing one. **Merge Parameters** can be used to deliberately permit undefined feature types to pass.



Import Feature Type

The Import Feature Type tool will take the schema definition of a selected dataset and add it to the workspace. By adding in the missing feature types to the schema, features of that type will be allowed to pass into the workspace.

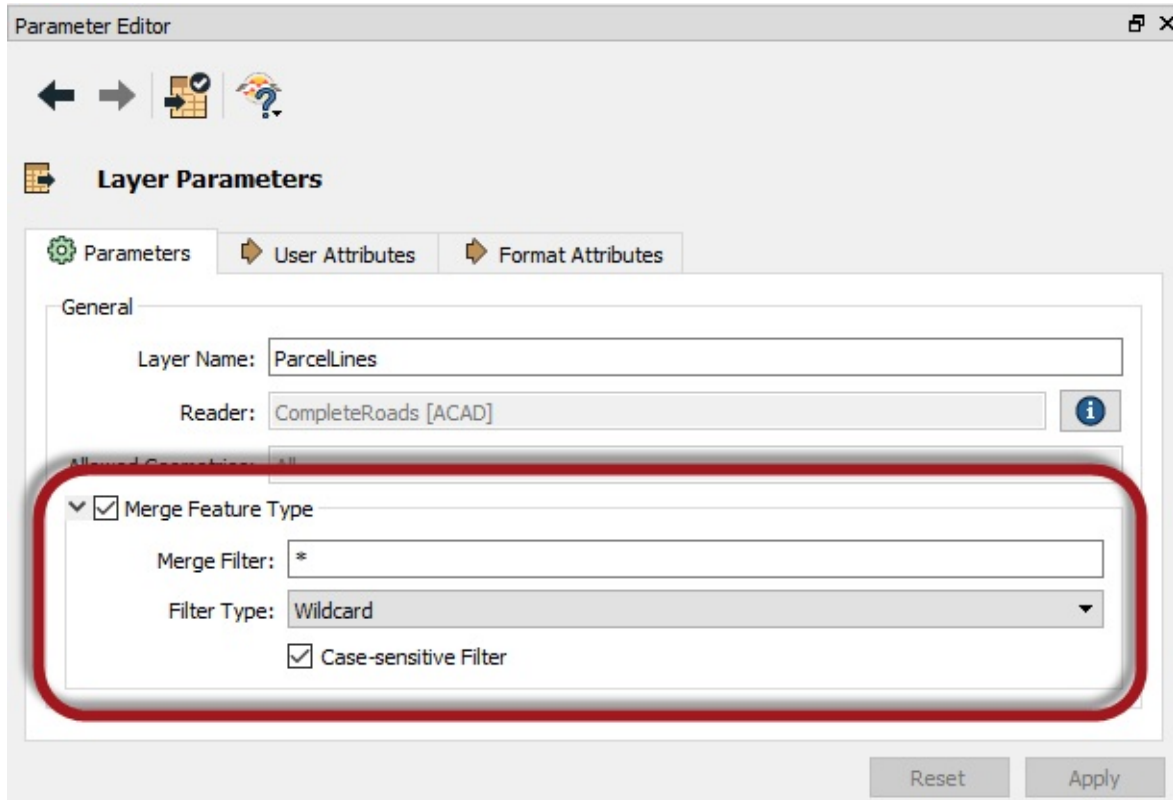
Once in the workspace the data will now be read and accepted as having a matching feature type.

Merge Parameters

Rather than adding in missing feature types to a workspace, the second option is to relax the restrictions on the feature type filtering process, purposely letting undefined feature types to pass.

The functionality is described as a merge, because features with an unknown feature type are literally merged into the input from an existing feature type.

Merge Parameters are available in each reader feature type for this purpose:



Once the Merge Feature Type option is checked a Merge Filter can be set. This is used to define exactly what incoming feature types are allowed to merge.

The filter type can be a wildcard or a regular expression.

In this example an asterisk (*) means FME should accept any unknown data into the workspace through this feature type. This filter is the default value for all Merge Feature Types.

In the workspace itself, the title of the feature type object is updated to reflect the filter being applied.



Once set, the data will now be read and - any feature types previously undefined - will be read through this feature type.

Exercise 5 Dealing with Unexpected Input	
Data	City Parks (MapInfo TAB) Walking Trail (CSV) Water Fountains (File Geodatabase) Car Parking (OpenStreetMap) Roads (OpenStreetMap)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Handling Unexpected Input
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Begin.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Begin.1.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Complete.fmw C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Complete.1.fmw

Let's continue your work on the fundraising walk project.

In this part of the project we'll look at whether the Unexpected Input warnings we received are something we need to be concerned about.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 4. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Begin.fmw (for 2017.0) or C:\FMEDData2017\Workspaces\DesktopBasic\Components-Ex5-Begin.1.fmw (for 2017.1).

Run the workspace (if you haven't before) to remind yourself of the warning dialog that pops up.

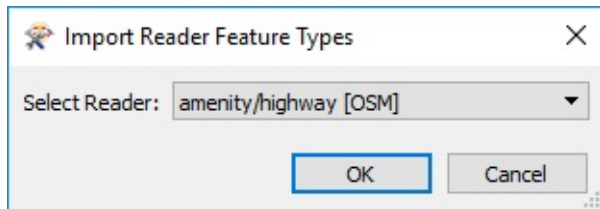
2) Import Feature Types

For the amenities dataset, car parks are the only feature type we require. We do not need - for example - either hospitals or schools to appear in our data. Therefore the reports of amenities data as unexpected input can be safely ignored.

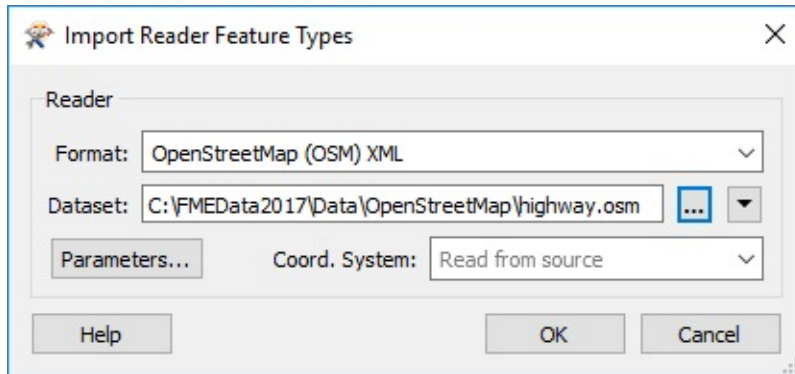
The problem with highways is that we do need them. But there are no feature types on the canvas for highway features. That's why highways are unexpected input and that is why they are getting dropped.

The solution is to add a feature type for highways.

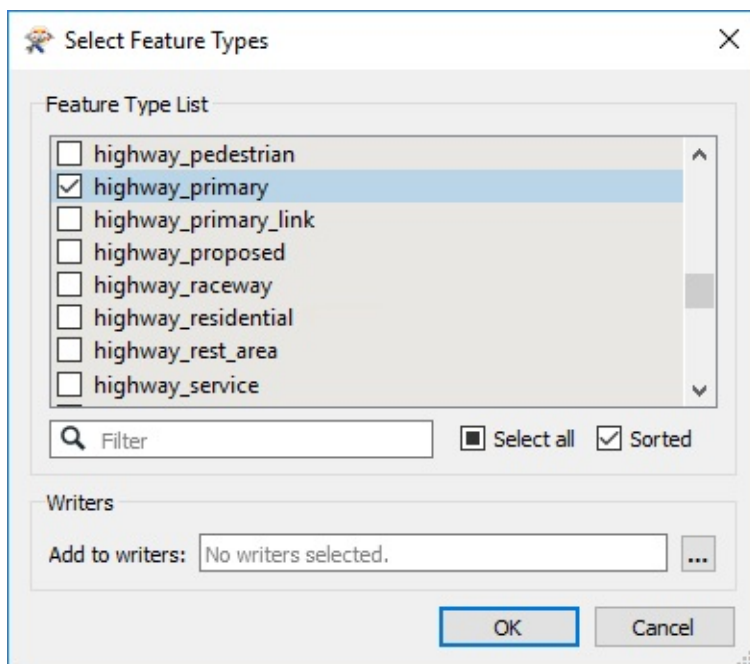
In **FME2017.0** choose Readers > Import Feature Types from the menubar. When prompted, choose to import them to the amenity/highway [OSM] reader:



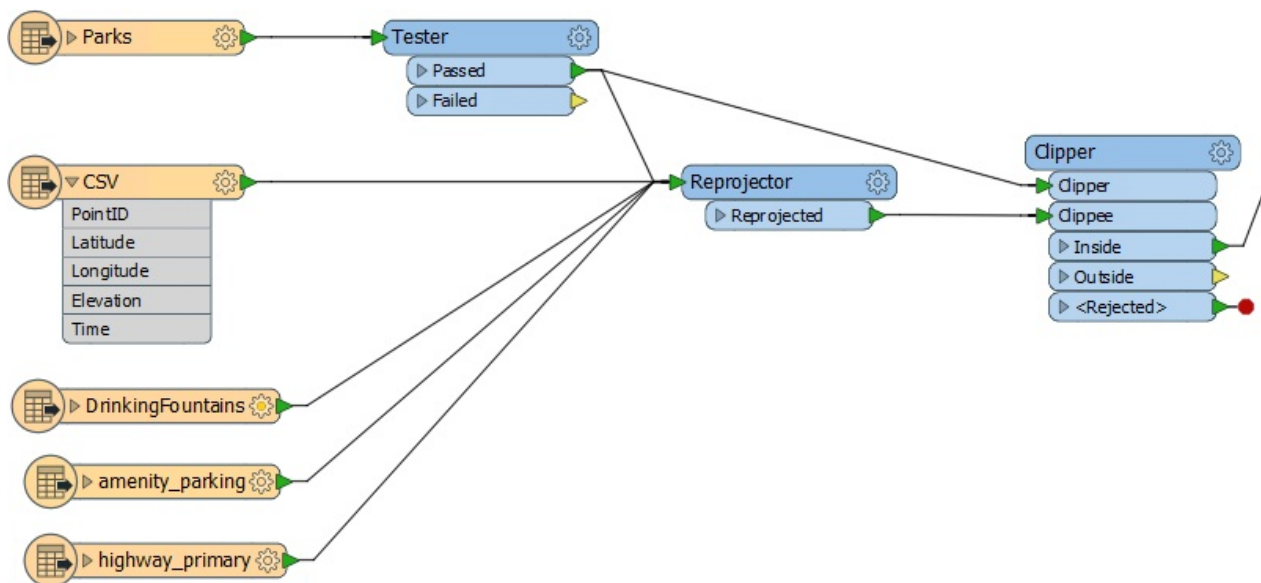
Click OK and you will be prompted for the dataset to import these feature types from. We have all the amenity types we need, so you don't need to select that; just select the highway.osm file:



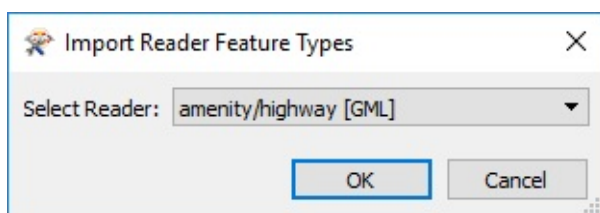
Our initial specification said only major roads are required so, when prompted, deselect all feature types and leave only highway_primary selected:



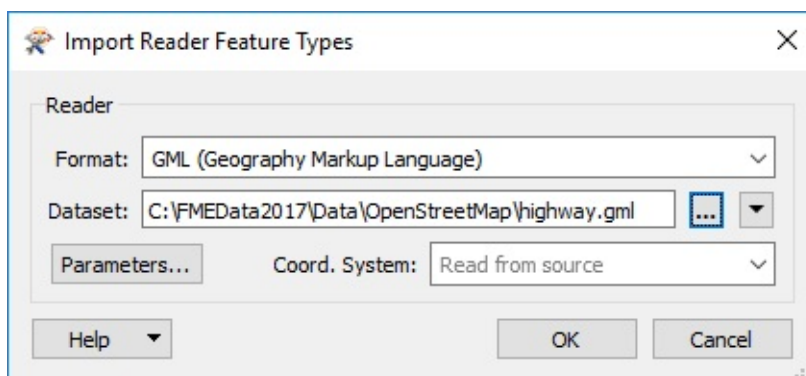
As with all other datasets, connect this feature type to the Reprojector transformer:



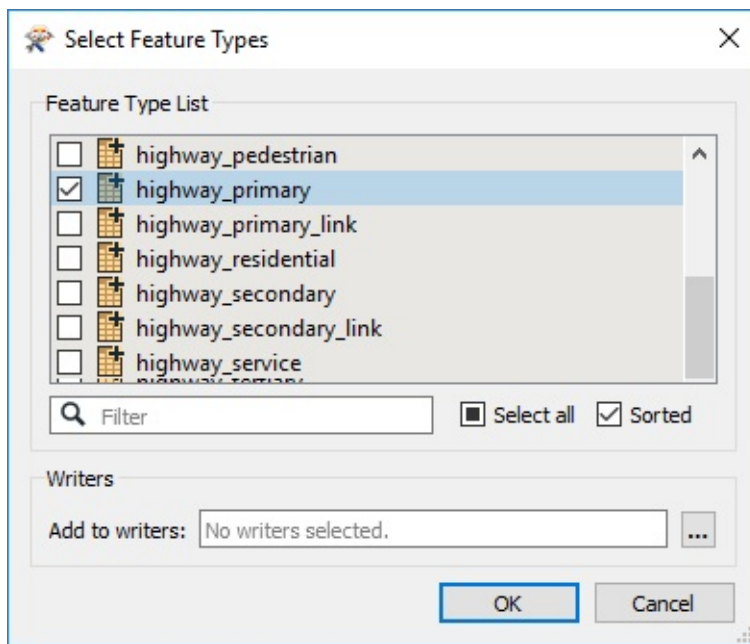
In **FME2017.1** choose Readers > Import Feature Types from the menubar. When prompted, choose to import them to the amenity/highway [GML] reader:



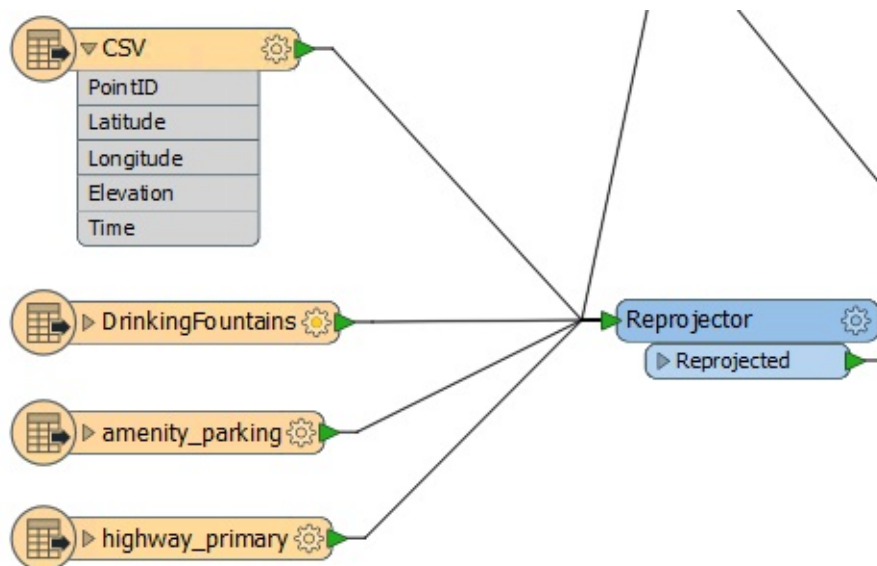
Click OK and you will be prompted for the dataset to import these feature types from. We have all the amenity types we need, so you don't need to select that; just select the highway.gml file:



Our initial specification said only major roads are required so, when prompted, deselect all feature types and leave only highway_primary selected:



As with all other datasets, connect this feature type to the Reprojector transformer:



3) Run Workspace

Run the workspace once more. This time there will still be unexpected input, but it will not include highways_primary; these will make it to the output dataset:



This is all good, except... I can only see a single road feature on the map. Other roads must be in a different feature type. In fact, if I inspect the source data (you can too if you like) I see that the two feature types I need are called *highway_primary_link* and *highway_unclassified* - both of which are reported as unexpected input.

So that is something that we need to fix.

4) Fix Unclassified Roads

Let's fix the unclassified roads first. Again, if a required source feature type is missing from the workspace, the simplest method is to add it.

So once more (as in step 2) select Readers > Import Feature Type from the menubar, and go through the import process, this time importing the *highway_unclassified* layer from either *highways.osm* (2017.0) or *highways.gml* (2017.1).

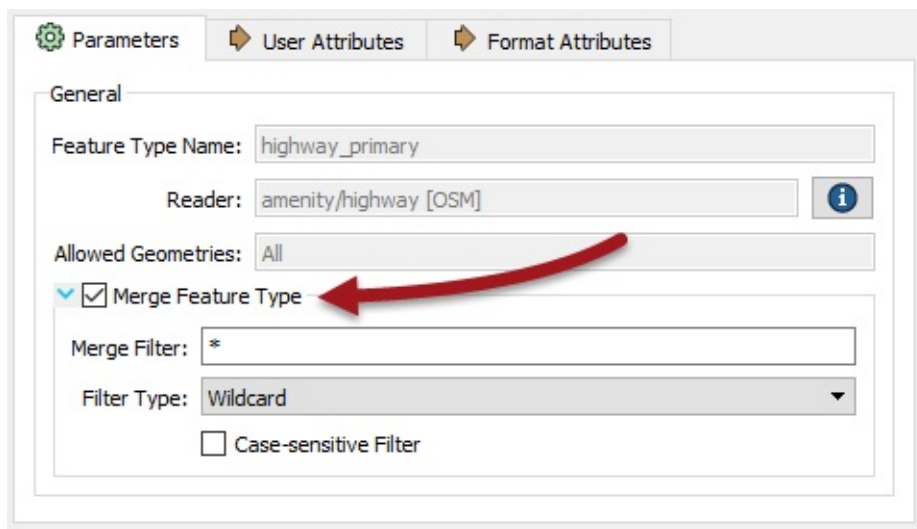
Again connect it to the Reprojector transformer.

5) Fix Primary Links

Now let's fix the primary links (these are short roads that connect the primary roads to the unclassified roads). We could use the same import tool to add a feature type, but let's try a different way.

NB: The images in this step are for OSM data in 2017.0, but the same method applies to GML dataset in 2017.1


View the properties for the *highway_primary* feature type. You will see a section of the General tab is labelled Merge Feature Types. Click in the Merge Feature Type toggle setting to turn it on:



Parameters User Attributes Format Attributes

General

Feature Type Name:

Reader: 

Allowed Geometries:

☒ Merge Feature Type

Merge Filter:

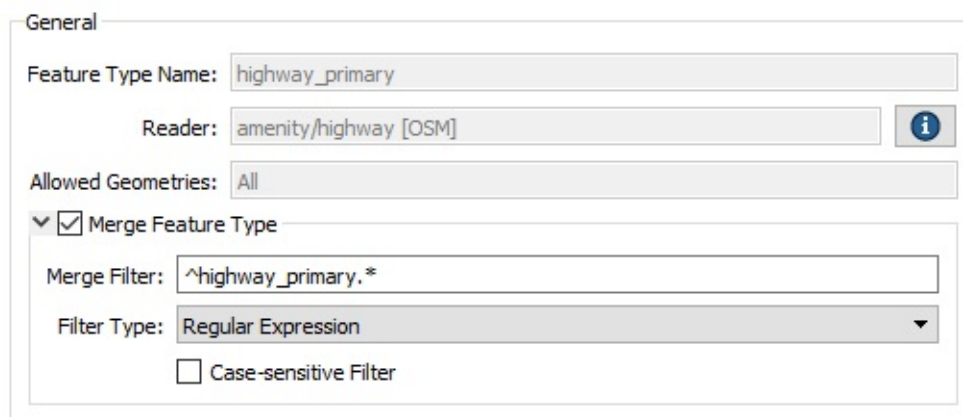
Filter Type:

☐ Case-sensitive Filter

By default the merge filter is set to allow everything in. This is a great way to read all of the data from a single dataset into a single feature type. But it's certainly not suitable here because it will allow FME to read ALL road features (even ones we don't want) plus it will allow in all of the amenity features that we've already decided aren't required.


So we have to be more subtle. In the Merge Filter settings, change the filter type to be a Regular Expression. Change the merge filter to be:

```
^highway_primary.*
```



General

Feature Type Name:

Reader: 

Allowed Geometries:

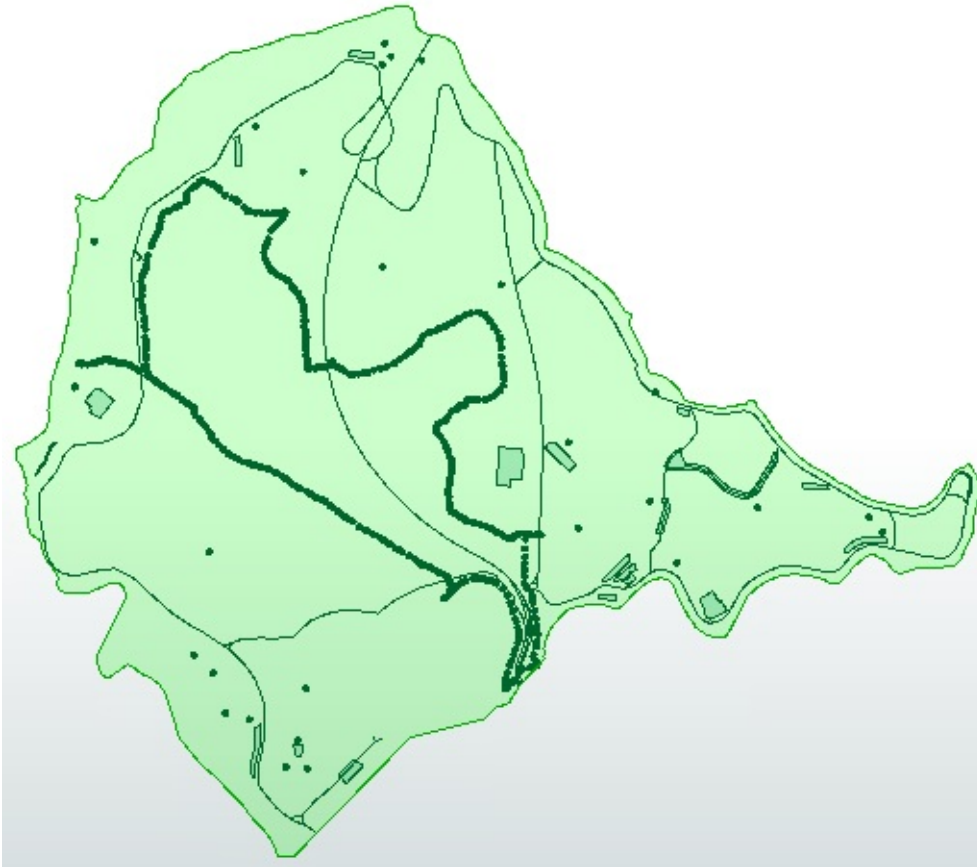
☒ Merge Feature Type

Merge Filter:

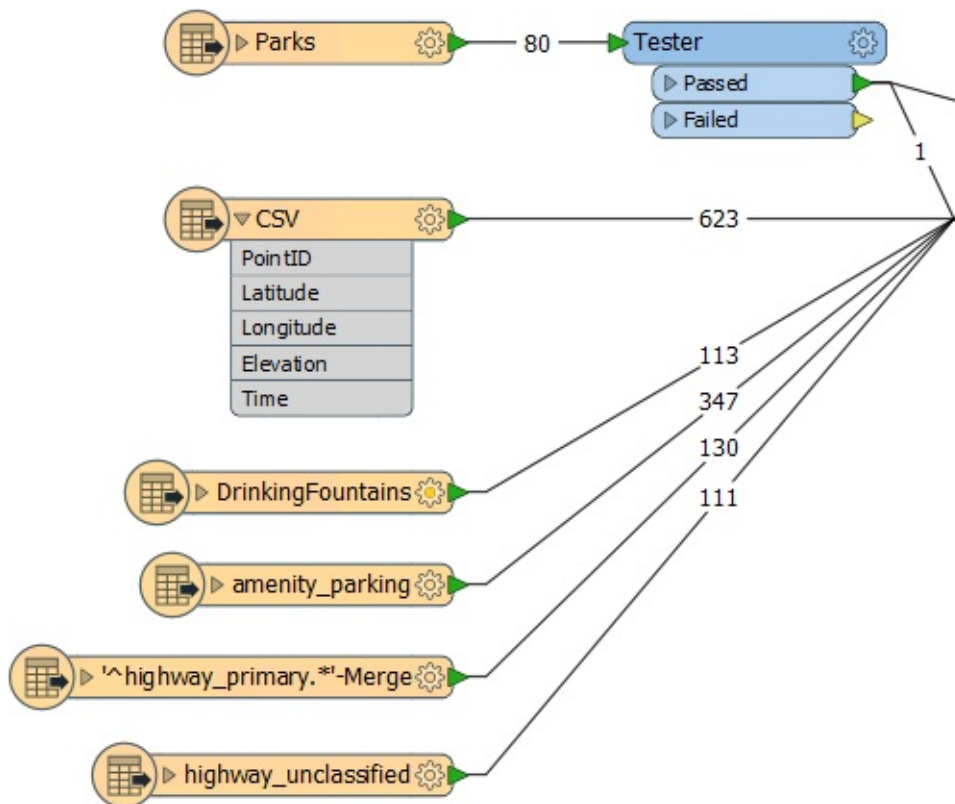
Filter Type:

☐ Case-sensitive Filter

Basically, this will allow feature types to pass only if they begin with "highway_primary". The result should be as follows:



Hurrah! We now have the data we want entering the workspace. Any other reports of Unexpected Input can now be ignored. We have completed the reading part of this project and the feature counts should now look like this:



CONGRATULATIONS

By completing this exercise you have learned how to:

- *Identify when unexpected input is (and isn't) a problem*
- *Handle unexpected input by using the Import Feature Type tool*
- *Handle unexpected input by using the Merge Feature Type tool*
- *Use a regular expression in the Merge Feature Type tool*

Writers

A **writer** is the FME term for the component in a translation that writes a destination dataset. A writer writes a single format of data. In general it also writes just a single dataset (i.e. a reader can read any number of datasets but, if sent to the same writer, they will be combined into a single output dataset).

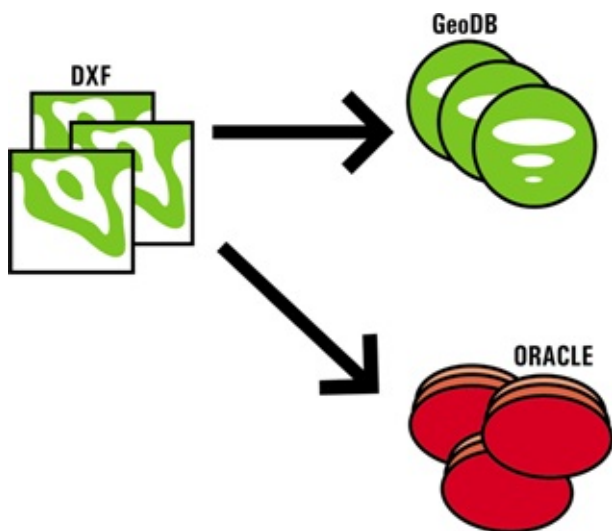
By default, the Generate Workspace dialog creates workspace with a single writer. However, this does not mean the workspace is forever limited to this. Additional writers can be added to a workspace at any time, any number of formats can be used, and there does not need to be an equal number of readers and writers.

Adding a Writer

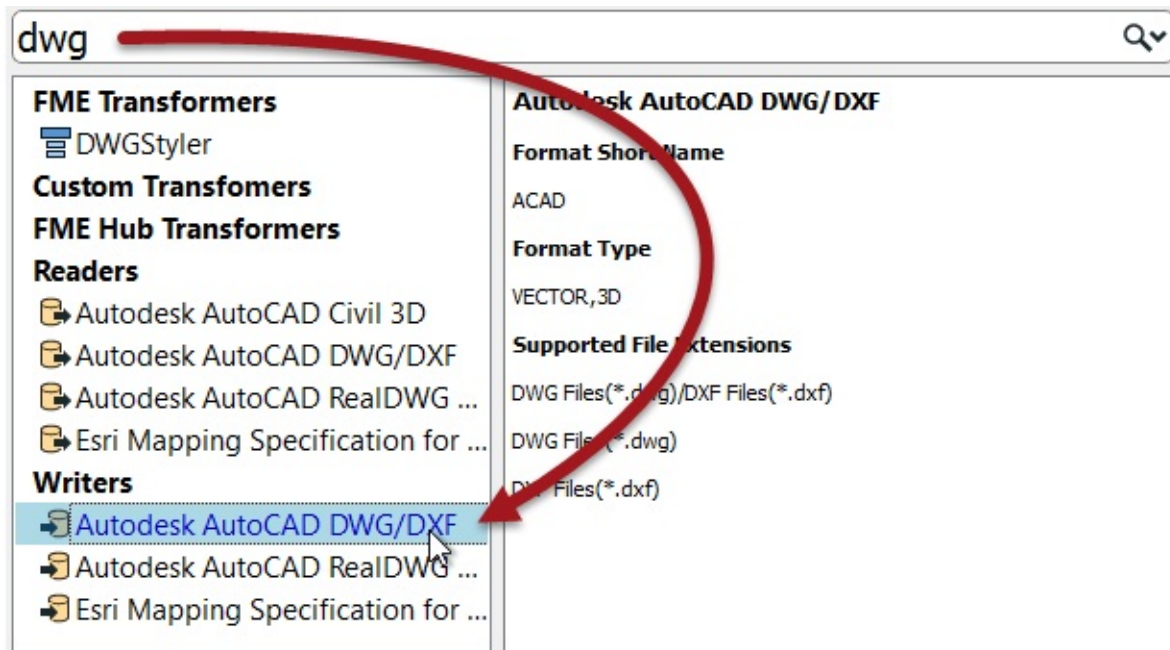
Adding a writer to a workspace is a common requirement. There are several reasons:

- The Generate Workspace dialog only adds a single reader and writer
- Each reader and writer handles only one format of data.
- Different datasets (of the same format) may require reading handling with different parameters

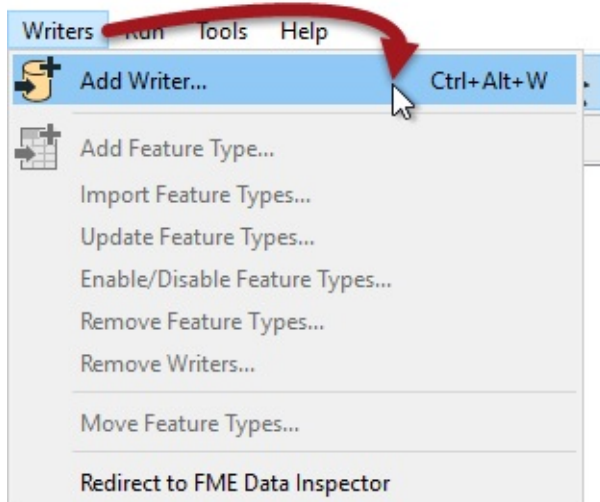
Therefore the need to write multiple formats of data – such as Geodatabase and Oracle – requires multiple writers.



Additional writers are added to a translation using the Quick Add menu:



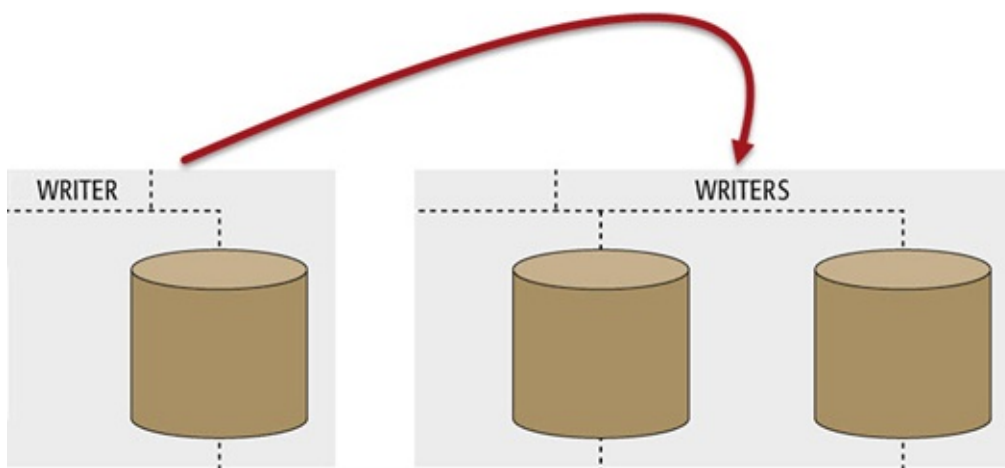
...Or by selecting Writers>Add Writer from the menubar.



NEW

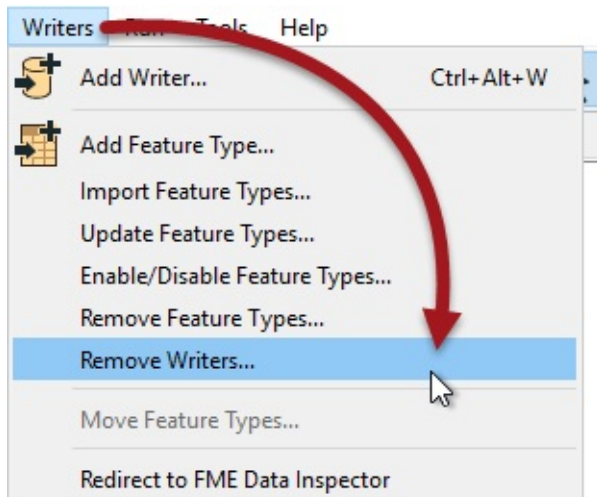
The ability to add a Writer using Quick Add is new for FME 2017

Adding a writer has this effect on the hierarchy diagram:



Removing a Writer

Not only can you add a new writer, you can remove an existing one; for example when you have an old writer whose output you no longer need. Tools exist to remove a writer from a workspace, both on the menubar and in context menus in the Navigator window.



Removing a writer obviously has the reverse effect on the hierarchy diagram!

.1 UPDATE

*As with readers, FME2017.1 introduces a tool to **update** a writer in Workbench, to bring its behaviour up to date with the current FME version.*

Controlling Writers

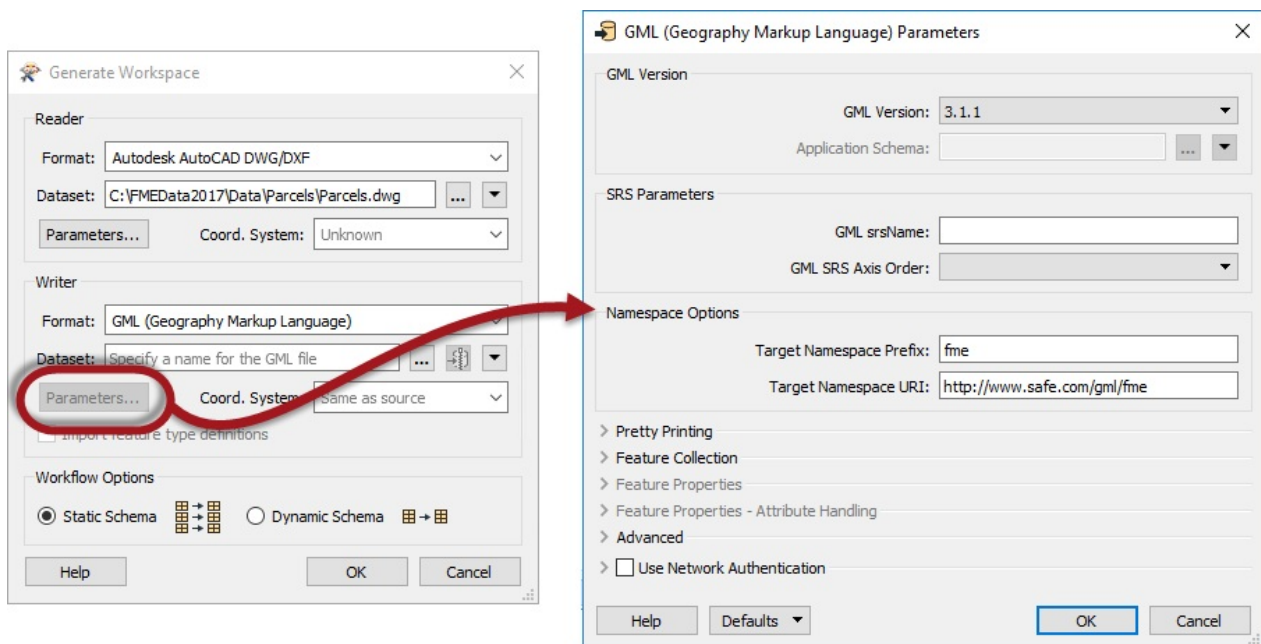
Writers are controlled with **writer parameters**.

Because parameters refer to specific components and characteristics of the related format, writers of different formats have a different set of control parameters.

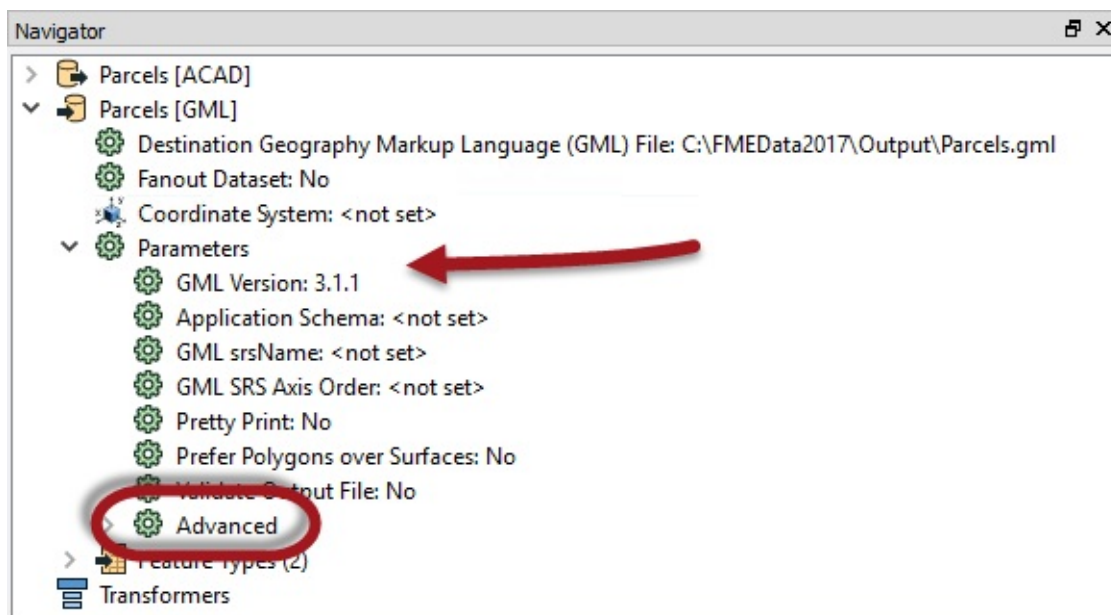
Writer Parameters

Writer parameters can be located - and set - in one of two locations.

Firstly, these parameters can be found in a dialog when a new workspace is being generated, or a new writer added:



Secondly, after the workspace is generated/writer is added, parameters are shown and set in the Navigator Window.

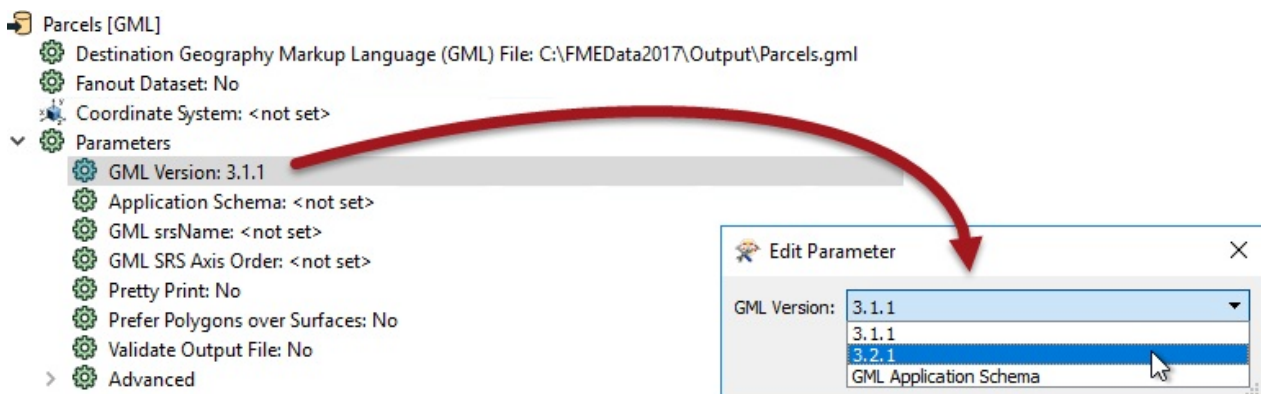


For ease-of-use, basic parameters are listed first, followed by advanced.

TIP

Writer parameters only exist in the Navigator window. They don't appear in the Parameter Editor dialog because there are no writer objects on the canvas to click on.

To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.



Exercise 6 Adding Writers	
Data	City Parks (MapInfo TAB) Walking Trail (CSV) Water Fountains (File Geodatabase) Car Parking (OpenStreetMap) Roads (OpenStreetMap)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Adding Writers
Start Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Begin.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Begin.1.fmw
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Complete.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Complete.1.fmw

Let's continue your work on the fundraising walk project. In this part of the project we'll start concentrating on the output requirements.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 5. Alternatively you can open C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Begin.fmw (for 2017.0) or C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex6-Begin.1.fmw (for 2017.1).

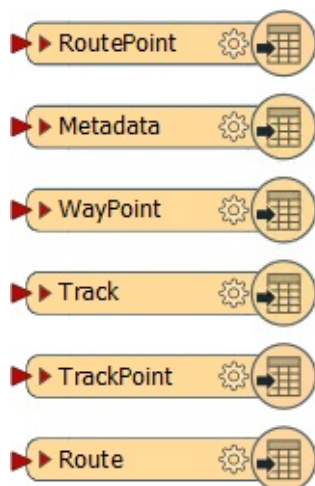
2) Add Writer

One of the output requirements is a GPX dataset of the walk route. So select Writers > Add Writer from the menubar and then enter these parameters:

Writer Format	GPS eXchange Format (GPX)
Writer Dataset	C:\FMEData2017\Output\Training\TrailRoute.gpx

Click OK to add the Writer.

This format is what we call Fixed Schema. It means the feature types are predefined by the format specification and we shouldn't try to change them. In this case we get six output feature types:



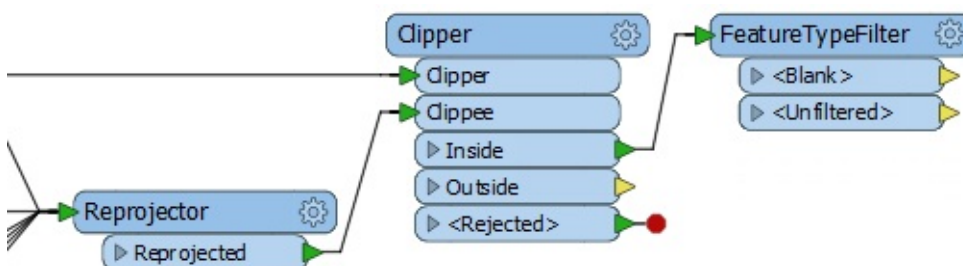
A route is a route to be followed, a track is a route that has been taken.

3) Add FeatureTypeFilter

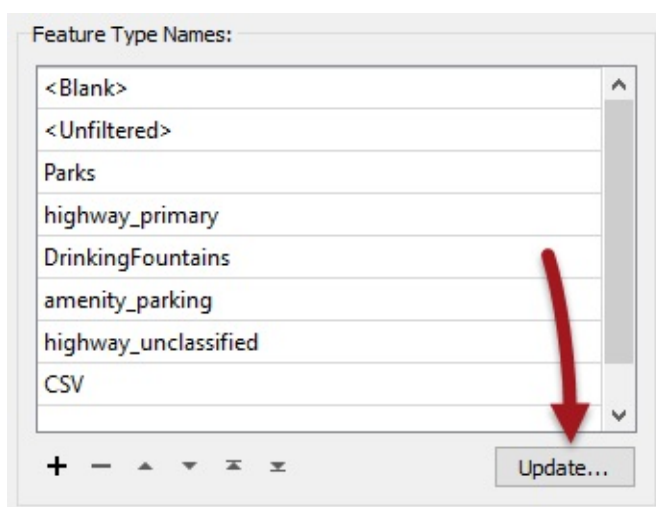
One issue about our workspace is that we merged all our data together through the Reprojector and Clipper transformers. It helped keep the workspace tidier, but now it's necessary to divide the data again if we want to write it onto separate layers.

We can do this with a FeatureTypeFilter transformer.

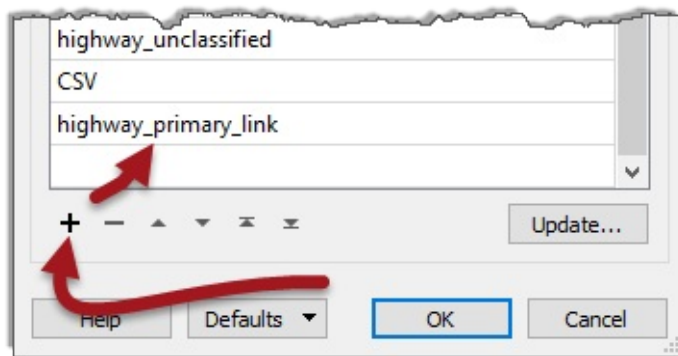
So, add a FeatureTypeFilter transformer and connect it to the Clipper:Inside output port:



Now view the transformer's parameters and click the Update button:



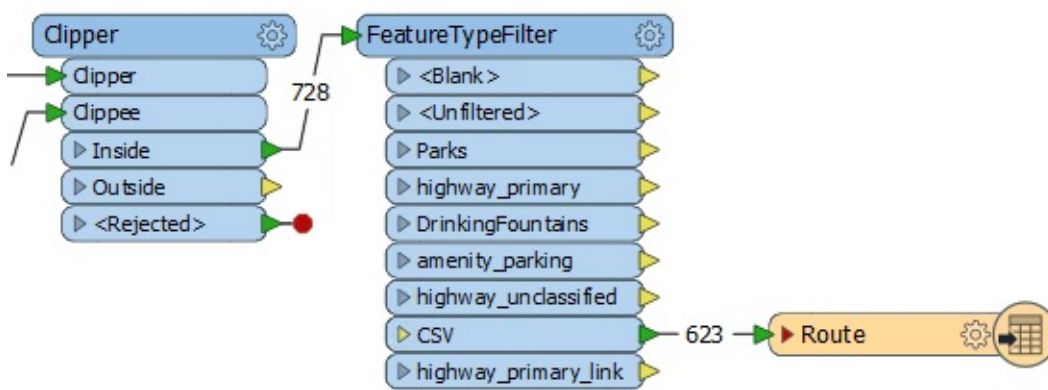
This fetches all the source feature type names in order to filter the data. However, it cannot identify merged feature types, so click the + button and manually add *highway_primary_link*:



Now we can map the trail data to the correct feature type.

4) Connect Schema

Simply connect the FeatureTypeFilter:CSV output port to the Route feature type:

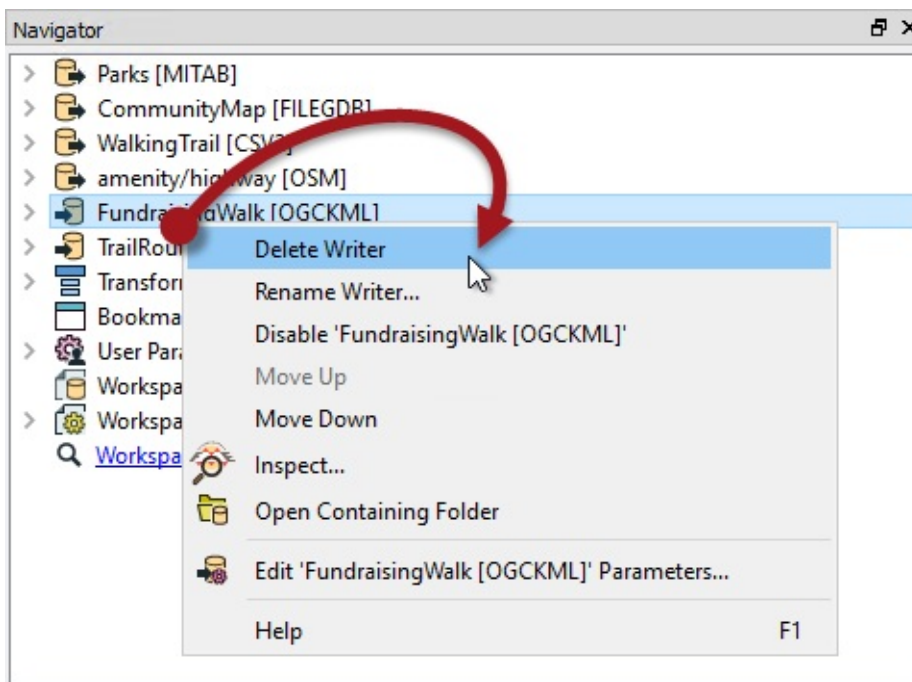


5) Run Workspace

Now run the workspace and examine the GPX output in the FME Data Inspector to ensure it is correct.

6) Delete Writer

Another phone call leads to another change. The organizers have decided not to write the main dataset to KML. They will let you know later which format to use. For now simply locate the Google KML Writer in the Navigator window, right-click on it, and choose the option to Delete Writer:



CONGRATULATIONS

By completing this exercise you have learned how to:

- *Add writers to a workspace*
- *Handle a fixed-schema writer*
- *Use the FeatureTypeFilter transformer to split data by reader feature type*
- *Delete a writer from a workspace*

Writer Feature Types

As per the hierarchy diagram, each writer in FME can have a number of feature types. Feature types define the layers being written to a destination dataset and the attributes for those layers to possess. They are represented as objects on the workspace canvas and also listed in the Navigator window.

In this workspace, a writer is writing a dataset of roads, with a different layer for each road type. Notice the writer in the Navigator window and how it has a list of feature types; then notice that each of these feature types is also depicted on the canvas:

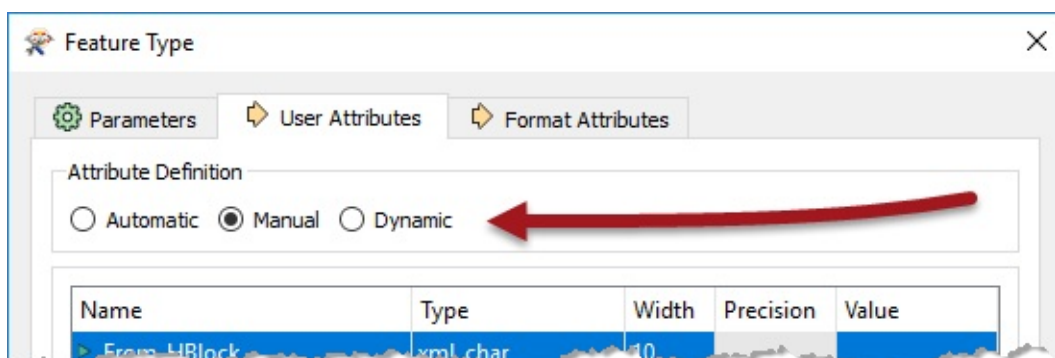


A workspace can contain any number of writers, each with any number of feature types.

Writer Feature Type Attributes

The attributes on a reader feature type represent '*what we have*' - literally what already exists in a source dataset. On the writer feature type attributes represent '*what we want*'.

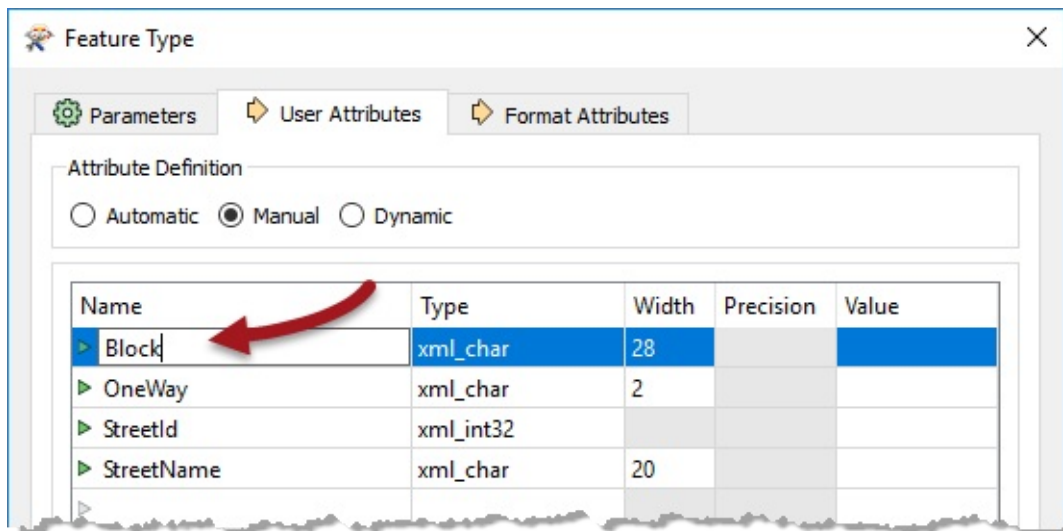
Each writer feature type has a User Attributes tab to define the attributes to be written. Unlike a reader feature type, there is a setting with three options to control how attributes can be defined:



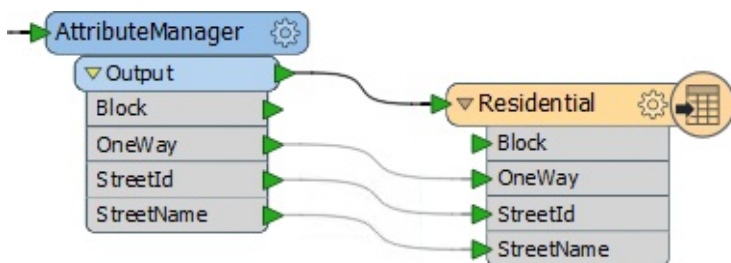
These options are Automatic, Manual, or Dynamic. Being aware of what these options do is important for managing writer feature types. So we'll look into them with Manual definitions first, since they are the easiest to understand.

Manual Attribute Definitions

A manual attribute definition is simply when the workspace author manually enters attribute names, and manually selects the attribute types, like so:



In this example, we are starting to define a table of road features. We have definitions for four attributes, one of which is just being edited. In the Workbench canvas, the feature type would look like this:

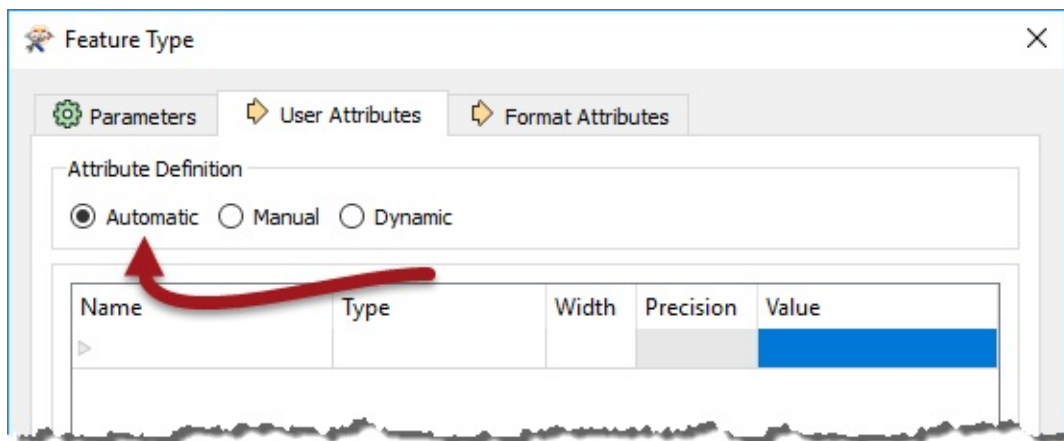


By default, when you create a new workspace the writer feature types are set in Manual attribute mode.

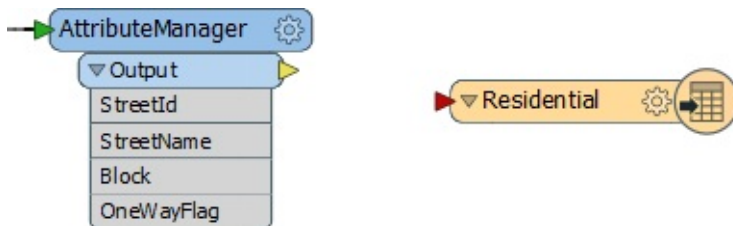
Automatic Attribute Definitions

An automatic attribute definition is when Workbench itself automatically defines the list of attributes, depending upon what reader feature types and transformers are connected.

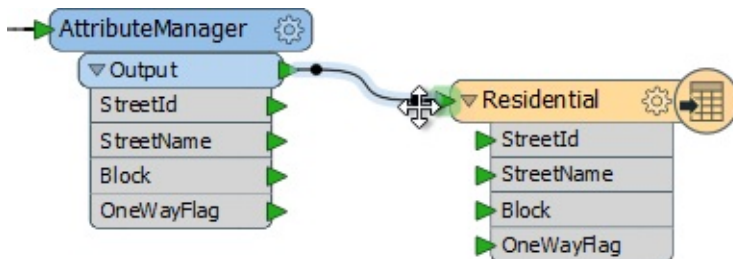
Here – when the user chooses the Automatic option - the attribute definition fields are left empty and uneditable, like so:



Similarly, on the canvas the feature type is also empty initially:

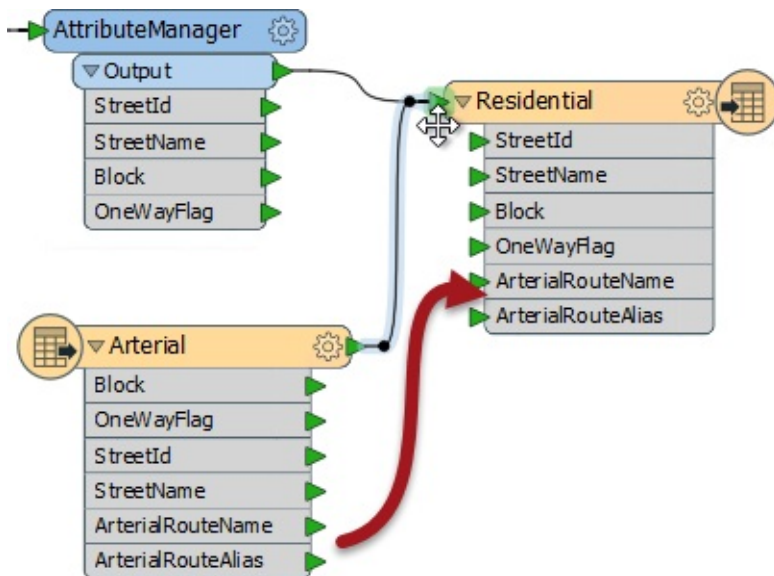


What makes the definitions automatic is the action that occurs when it is connected to the rest of the workspace:



Notice that all of the attributes defined - either on a reader feature type or a transformer - are now copied over to the writer feature type automatically.

Because this operates automatically, the list of attributes on the writer feature type will be immediately updated to match whenever attributes are changed in the workspace. For example, if a reader feature type is removed and another one connected, or if a second reader feature type is connected:

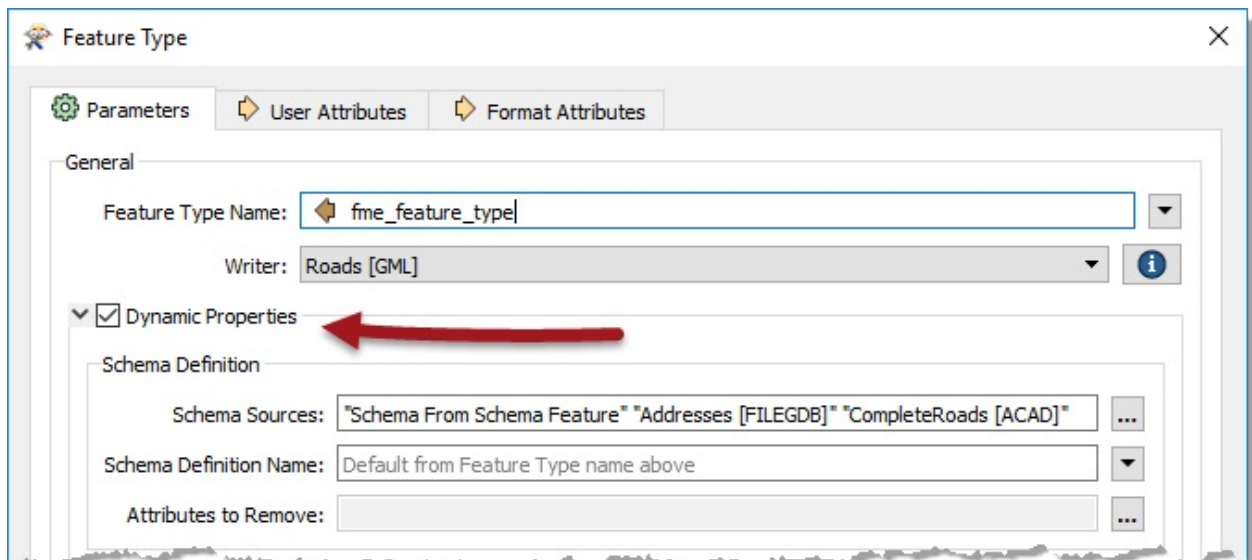


This is really useful because it means that multiple streams of data can be merged into a single feature type, and all attribute definitions are taken care of.

However, what this won't do is handle the scenario where you change a source dataset parameter to read a dataset with a different set of attributes. In other words, the writer attributes will only be defined by what is connected, not what actually exists in the source data. For that scenario you need dynamic attributes.

Dynamic Attribute Definitions

When the attribute definition is set to dynamic it causes the Dynamic Properties option (in the General tab) to be selected:



A dynamic feature type truly has no definition in advance. Instead, it dynamically takes its attribute definitions from either whatever data is fed to it when the workspace is run, or another set of schema definitions selected by the user.

So, like an automatic definition, you don't see attributes on the initial feature type. However, neither do you see attributes when reader feature types are connected!

Police Chief Webb-Mapp says...

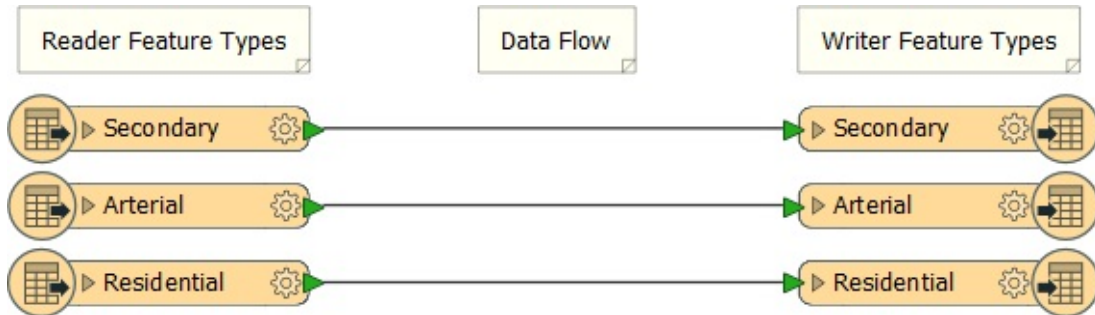
OK let me spell out these important differences.

Automatic attributes takes their definition from whatever is connected to them. If the Source Dataset parameter is changed, it will have no effect.

Dynamic attributes are different. If the Source Dataset parameter is changed, the attribute definition comes from whatever source data gets read.

Adding Writer Feature Types

Generating a workspace is the point at which most feature types are added to the translation. Whatever feature types are added by the reader are automatically duplicated on the writer:

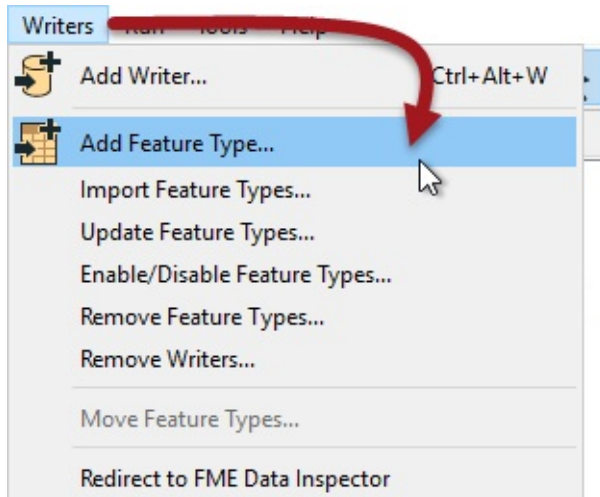


The writer feature types represent "what you want", but that is often not the same as "what you have". Therefore you need to edit the writer feature types to get the structure that you desire.

Also, depending on format, you may need to add further writers. How you add new feature types depends on whether it is a new writer or an existing one.

Adding Feature Types to an Existing Writer

Feature Types can be added manually to a writer using Writers > Add Feature Type on the menubar.

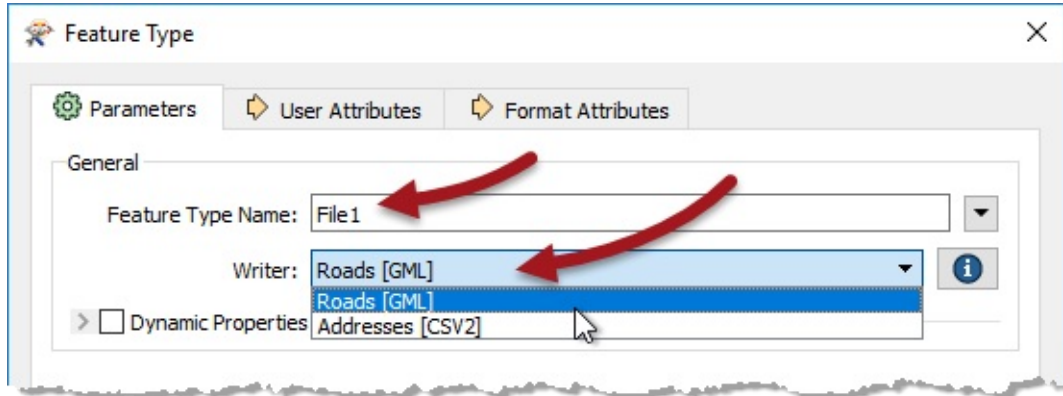


WARNING

At least one writer must exist in the translation hierarchy; else this option will be greyed out.

Adding a feature type opens the Feature Type Properties dialog in order to edit the new feature type's properties.

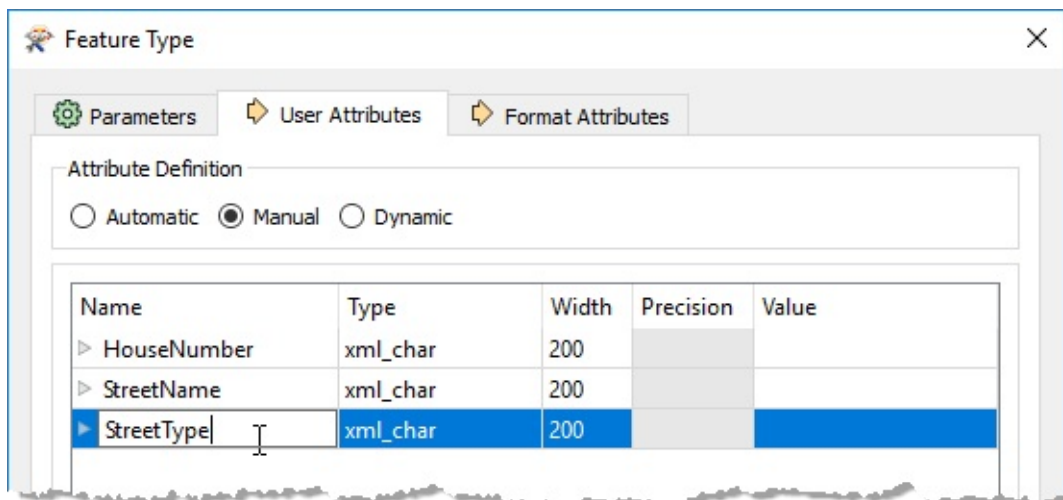
As when editing the schema, the General tab can be used to define the new feature type's name. Also, if there is more than one writer in the workspace, there is a parameter to specify which writer this new feature type belongs to:



TIP

It's worth remembering that, with format-specific terminology, instead of "Feature Type" this dialog's labels may state "Feature Class," "Layer," "Sheet," or whatever terminology is specific to the format of data you are writing.

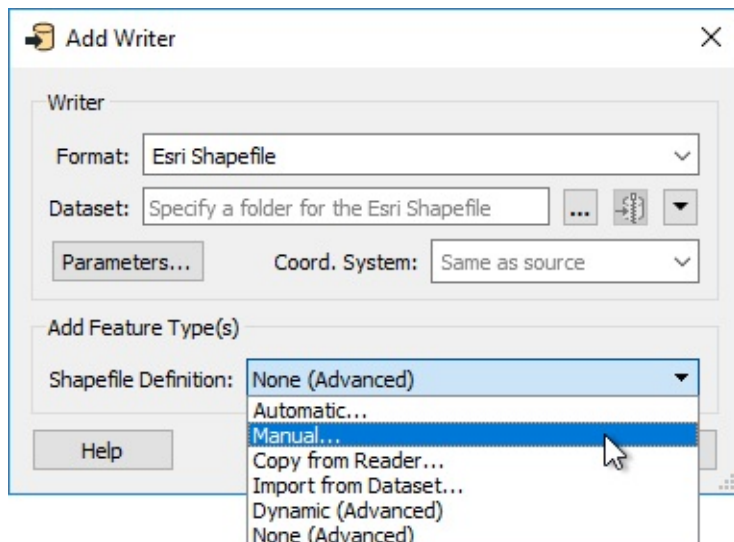
As usual, the User Attributes tab can be used to define the new feature type's attribute schema, in either of the three different ways:



Here the user has added a new feature type and is manually defining the attributes that belong to it.

Adding Feature Types to a New Writer

When a new writer is being created the option is provided to add new feature types:



As you can see, addition of feature types can be completed in a number of ways. The three methods we have already looked at are:

- **Automatic:** Adds a new feature type with the attribute definition parameter set to automatic
- **Manual:** Adds a new feature type with the attribute definition parameter set to manual
- **Dynamic (Advanced):** Adds a new feature type with the attribute definition parameter set to dynamic

The others are:

Copy from Reader: Adds a new feature type with manual attribute definition; however, the attributes are pre-defined to an existing reader schema. This is useful when you have manually added a reader and want to duplicate its schema on the writer.

Import from Dataset: Adds a new feature type with manual attribute definition parameter; however, the attributes are pre-defined to an external dataset schema. This is useful where - for example - you are adding a writer to write to a database; the database tables already exist and you use this tool to copy their schema definition into the workspace.

None (Advanced): Does not add any new feature type. This is a rare scenario, mostly used when you are adding a writer but are not yet ready to define its schema.

Exercise 7 Managing Writer Feature Types	
Data	City Parks (MapInfo TAB) Walking Trail (CSV) Water Fountains (File Geodatabase) Car Parking (OpenStreetMap) Roads (OpenStreetMap)
Overall Goal	Create a set of data for mapping a recreational event
Demonstrates	Adding Writer Feature Types
Start Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Begin.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Begin.1.fmw
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Complete.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Complete.1.fmw

Let's finish up your work on the fundraising walk project.

In this part of the project we'll finalize the output requirements.

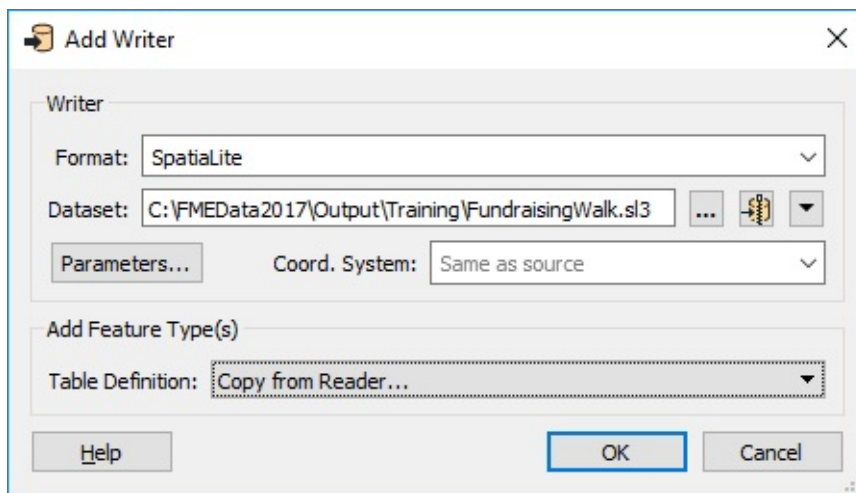
1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 6. Alternatively you can open C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Begin.fmw (for 2017.0) or C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex7-Begin.1.fmw (for 2017.1).

2) Add Writer

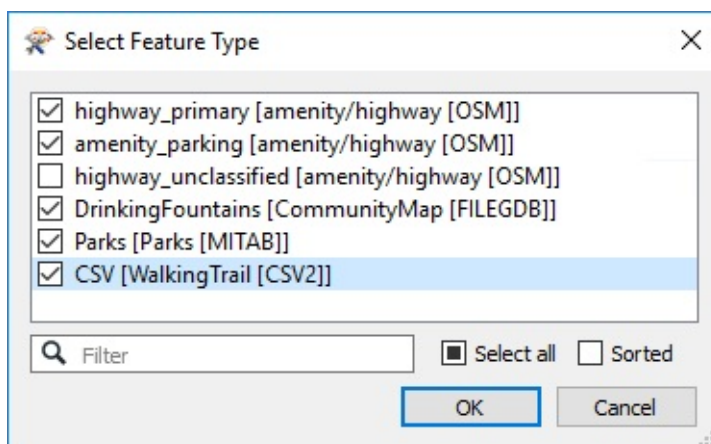
If you recall, one late change was to change the writer format. The organizers have decided to go with Spatialite instead of KML. So add a Spatialite Writer with the following parameters:

Writer Format	Spatialite
Writer Dataset	C:\FMEData2017\Output\Training\FundraisingWalk.sl3
Add Feature Type(s)	Table Definition: Copy from Reader



Click OK. When prompted select all the following reader feature types to copy onto the writer:

- highway_primary
- amenity_parking
- DrinkingFountains
- Parks
- CSV



.1 UPDATE

If you're using FME2017.1 the dialog will show GML files for amenities and highways, rather than OSM.

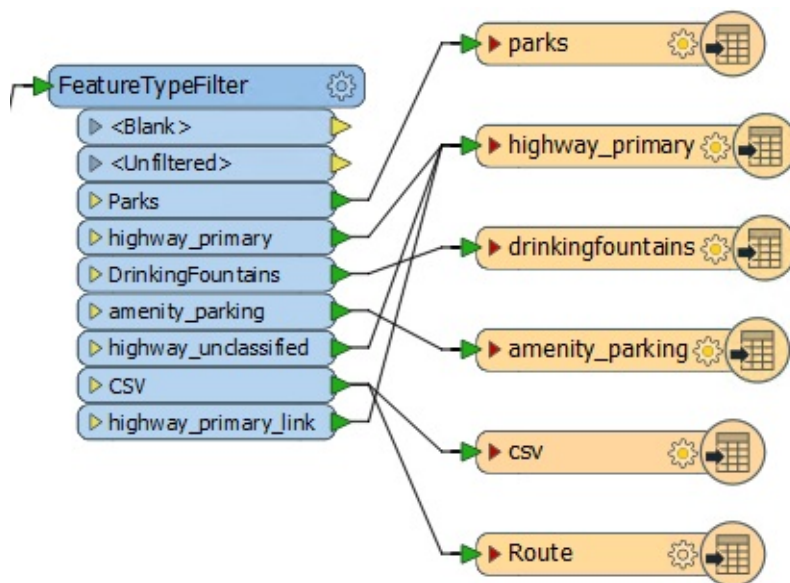
This will give a series of writer feature types. The highway_unclassified feature type isn't needed because we will write all road features to the same table.

Now all that is to do is do some schema mapping.

3) Map Schema

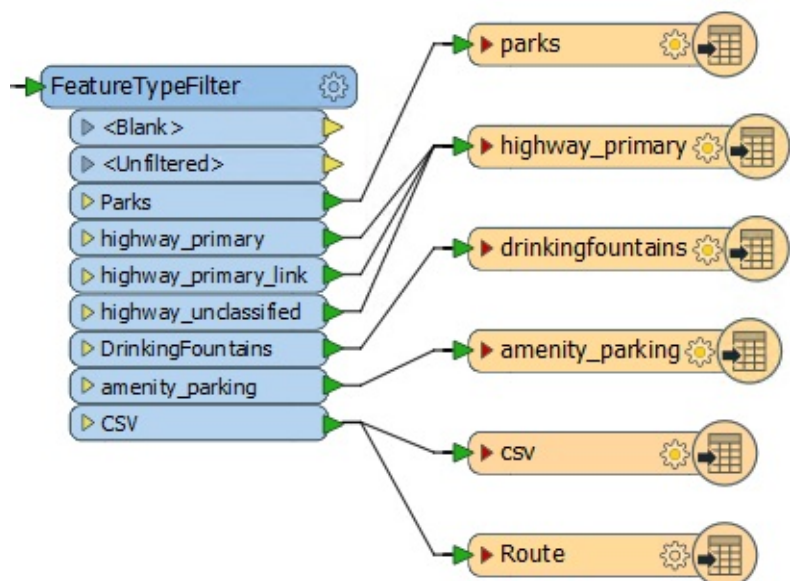
Make the following connections:

FeatureTypeFilter Port	Writer Feature Type
Parks	parks
highway_primary	highway_primary
highway_primary_link	highway_primary
highway_unclassified	highway_primary
DrinkingFountains	drinkingfountains
amenity_parking	amenity_parking
CSV	csv



4) Tidy Connections

Right now some connections overlap. This isn't a good idea as it makes the workspace harder to interpret. So check the parameters for the FeatureTypeFilter. Use the arrow buttons to move feature types up and down in the list until all connections are separated out:



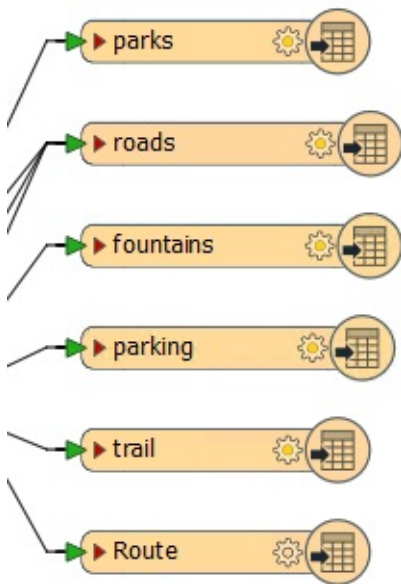
5) Tidy Feature Types

One last task is tidy up some of the writer feature types.

Firstly delete all writer feature types that aren't being used (RoutePoint, TrackPoint, Metadata, WayPoint, Track). You can do this using Tools > Remove > Unattached on the menubar.

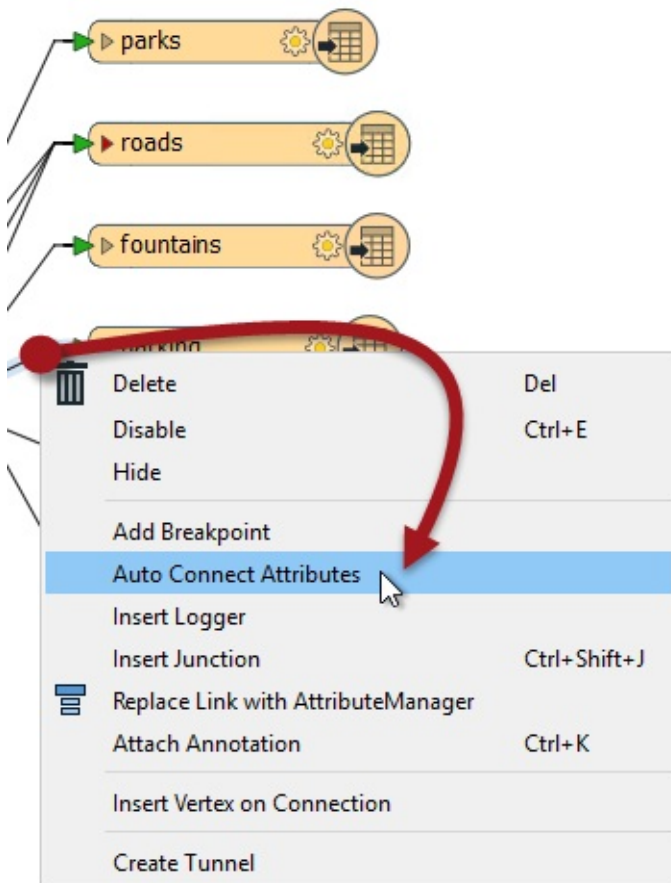
Secondly rename all Writer feature types to something more user friendly, for example:

- Rename highway_primary to roads
- Rename amenity_parking to parking
- Rename CSV to trail
- Rename drinkingfountains to fountains



NB: All feature type and attribute names in SpatiaLite are lower case.

Finally let's clean up and connect some of the writer feature type attributes. The simplest way to connect most of them is to right-click a connection and choose Auto Connect Attributes:



This works because most of the attributes have the same name, just a different case, and FME can figure that out automatically. If the attributes have a completely different name, then you would have to connect them manually; or you can just delete from the writer schema any attributes we don't really need.

Re-run the workspace and examine the Spatialite output in the FME Data Inspector. It should look like this:



That is the end of this project. The Spatialite database can now be passed on to the organizers to produce the actual event map.

Advanced Exercise

Oh! Now that we dropped the KML output dataset, the Python script we wrote probably won't work!

If you have time, and have experience of Python, why not edit the script to support copying the GPX and Spatialite output datasets to the "shared" folder?

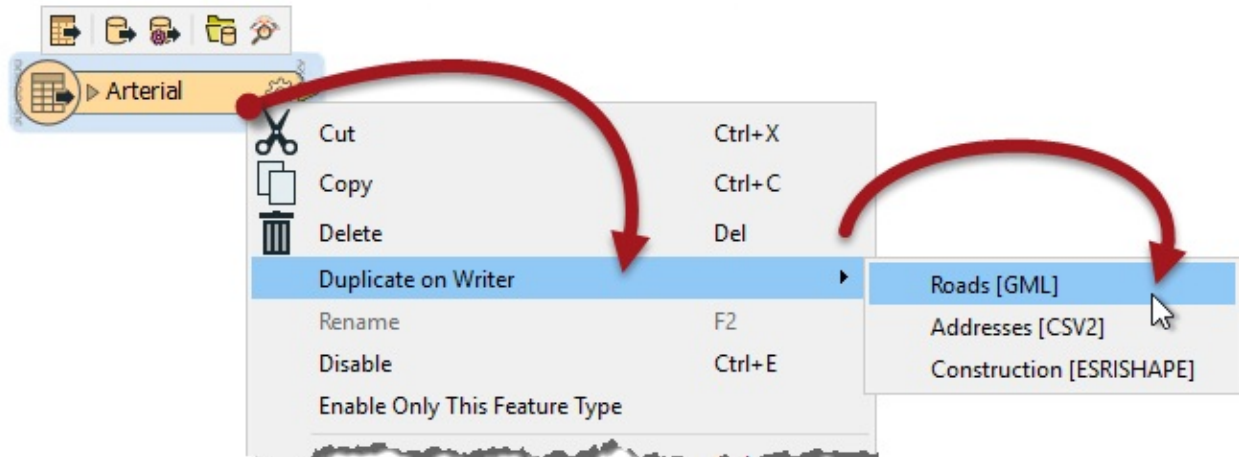
CONGRATULATIONS

By completing this exercise you have learned how to:

- *Copying reader feature types while adding a new Writer*
- *Untangling overlapping connections by moving transformer ports*
- *Deleting unattached writer feature types*
- *Cleaning up a writer schema by making automatic attribute connections*

Copying Feature Types to a Writer

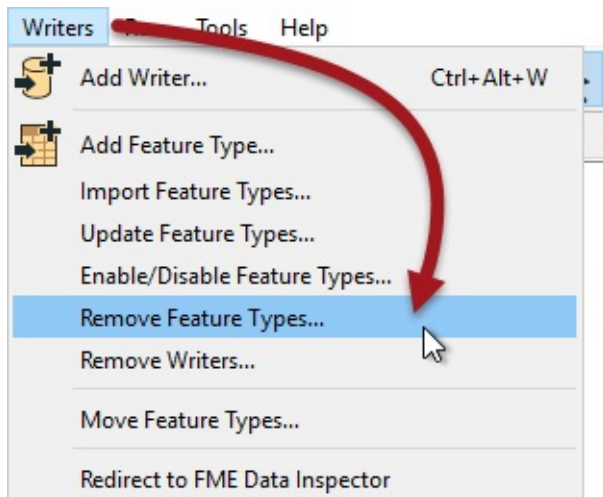
If a workspace already contains a writer, it's still possible to copy a reader schema into it. This is simply done by selecting the required reader feature types, right-clicking them, and using the option Duplicate (on Writer).



The command causes duplicates of the reader feature types to be added to the writer and automatically connected. Again, at least one writer must exist in the translation hierarchy; else this option will be greyed out.

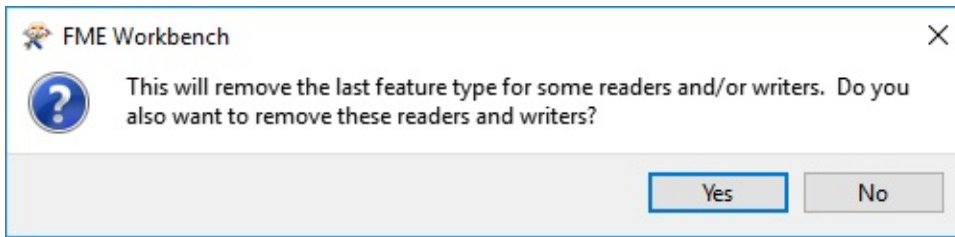
Removing Feature Types

Feature types can be deleted by selecting them on the canvas and pressing the delete key. They can also be deleted using the remove feature types tool on the menubar:



A user might delete a feature type if there is a destination dataset layer that they no longer wish to write.

Whenever all feature types are deleted from a writer then FME will prompt the user to decide whether to remove the writer component as well.



This makes sense because if there are no feature types you wish to write, why would you still wish to write the dataset at all?

If you answer No, then the feature types are all removed, but the writer is left in the translation. We call this a "dangling" writer, because it has no children in the hierarchy.

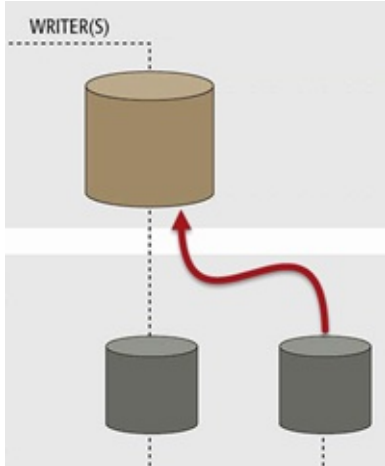
TIP

A dangling writer isn't as much of a problem as a reader, since there is no data to be written; although it can cause a loss of performance depending on other (advanced) factors such as the order of writers.

Import Writer Feature Types

A further tool under the Writer menu is labelled Import Feature Types.

Importing feature types is like when you add a new writer and choose to import the feature types, the difference being the writer here already exists.

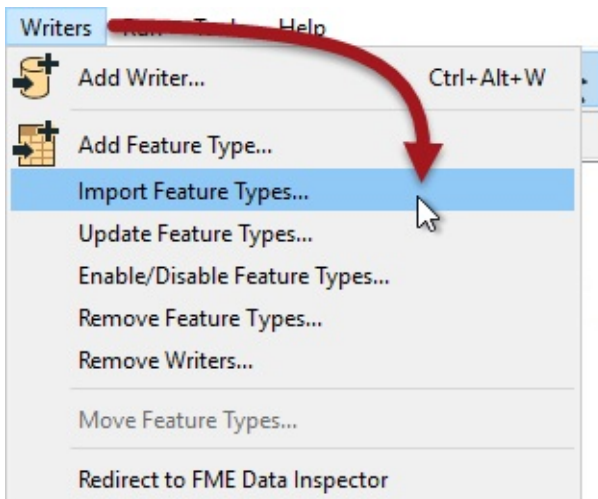


One reason to import feature types is to write to extra tables in a database.

For example, a user has a workspace that writes to a spatial database. It only needs to write to a single table (Roads), so there is a single writer representing the database, and a single feature type representing the Roads table.

However, at some future point the user decides the workspace also needs to write to a second existing table (Rail) and so imports the schema for the Rail table.

The way to do this is to use the menu tool Writer > Import Feature Types:



The user selects the writer to add the feature type to (if there is more than one writer in the workspace) and then is prompted for the format and dataset to import the schema from.

The import tool will take the chosen schema definition and add it to the chosen Writer in the workspace.

Mr E. Dict (attorney of FME law) says...

Whenever I create a new legal contract for a client, I copy previous contracts. I don't put the same clauses (data) in there, but they do have the same structure (schema). It's like using a template, and that's exactly what the Import Feature Types tool does.

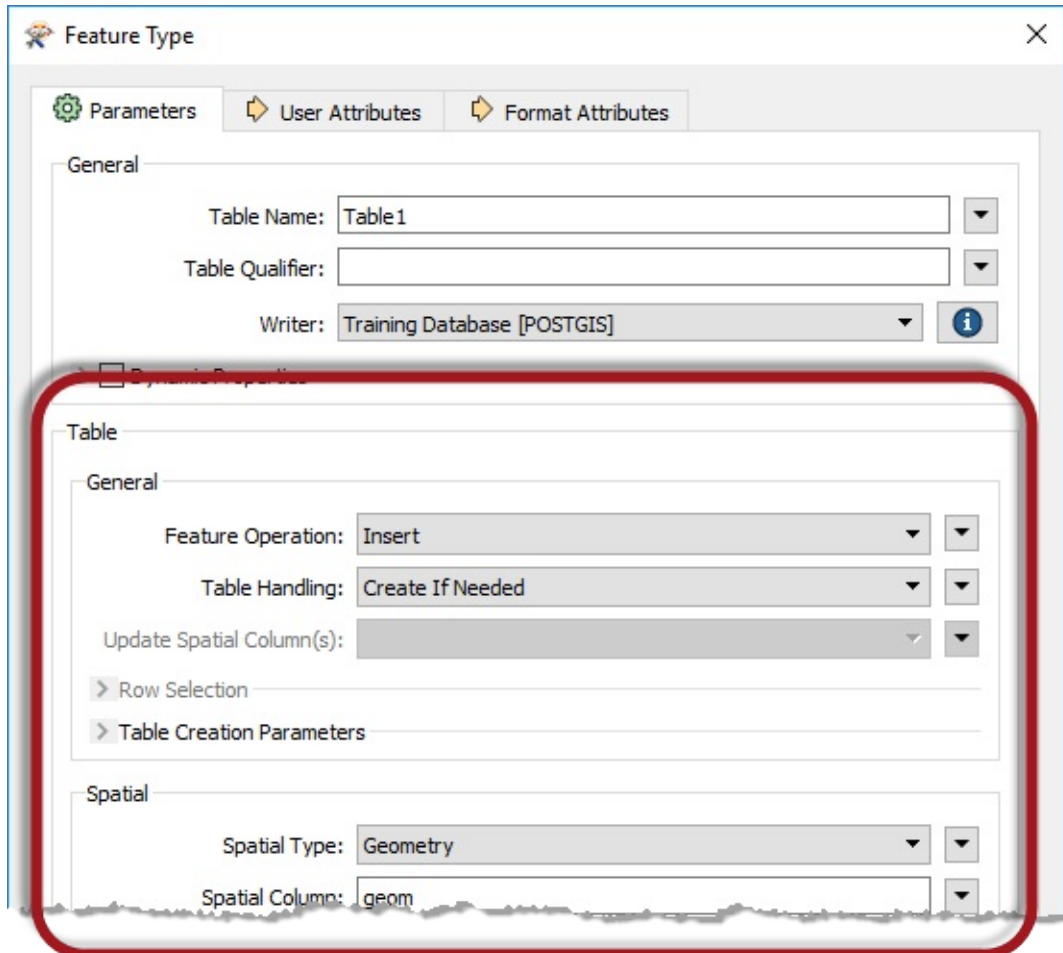
Miss Vector says...

Here's an interesting question for you to consider. I want to convert some data from Esri Shapefile to GML format, with the GML having the same schema as an existing PostGIS database. How can I do this?

- 1. Create a workspace to convert the Shapefile to PostGIS. Create a second workspace to convert the PostGIS to GML.*
- 2. Create a workspace to convert the PostGIS to GML. Create a second workspace to convert the Shapefile to GML, overwriting the previously created GML file.*
- 3. Create a workspace with PostGIS and Shapefile Readers, and a GML Writer. Copy the PostGIS Reader feature types to the GML Writer and connect the Shapefile Reader feature types to them. Delete the PostGIS Reader and run the workspace.*
- 4. Create a workspace with a Shapefile Reader and a GML Writer. Import feature type schemas from the PostGIS database into the GML Writer.*

Writer Feature Type Parameters

Just like writers, feature types have their own set of parameters that control how that feature type (layer/table) is being written. These parameters appear both in the Parameter Editor window and in the Feature Type Properties dialog:



There are general parameters that appear for every feature type, but some feature types have additional parameters below that.

The important thing is that these parameters only apply to a single feature type, whereas a writer parameter would apply to all feature types.

For example, the above screenshot is from a PostGIS database feature type with a parameter to define the Operation Type. This operation can be Insert, Update, or Delete. This is a Feature Type parameter because each table might require a different type of operation; for example I want to insert records into table A, but delete records from table B.

Conversely, there is no password parameter for this database table, because authentication applies to the entire database, not the individual tables, so authentication parameters are at the writer level (or in a database connection).

TIP

As with writers, writer feature type parameters vary by format, and some formats have no extra parameters at all at this level.

Module Review

This session was designed to help you understand the different components of a translation, how they can be controlled, and how they interact.

What You Should Have Learned from this Module

The following are key points to be learned from this session:

Theory

- A translation is made up of a Workspace, Readers and Writers, and Feature Types. Tools exist to manage all of these components
- Readers and Writers are defined by entries in the Navigator window, Feature Types by objects in the workspace canvas
- Translations are controlled by Reader and Writer Parameters, and Feature Type Parameters
- Datasets are selected by Reader and Writer parameters in the Navigator window, not in the canvas
- The Unexpected Input Remover filters incoming data if it doesn't match a known feature type defined in the workspace

FME Skills

- The ability to set up and manage the components of a translation
- The ability to control a translation with read and write parameters
- The ability to handle source datasets regardless of feature type or attributes

Further Reading

For further reading why not browse [articles tagged with Feature Type](#) on our blog?

Questions

Here are the answers to the questions in this chapter.

Miss Vector says...

Here are four different formats and four different terms for "feature type". Connect the format to the correct terminology.

Format	Feature Type Terminology
Oracle Spatial	Table
MicroStation Design	Level
Esri Geodatabase	Feature Class
Adobe PDF	Layer

Technically, Esri feature classes are stored in tables, but feature class is the best match in this list.

Miss Vector says...

Here are four scenarios and four tools or settings. Connect the scenario to the correct tool/setting.

Scenario	Tool
Source filename changes	Source Dataset Parameter
Attribute type changes in database	Update Feature Type
Additional database table needs reading	Import Feature Type
New file dataset needs reading	Add Reader

Technically, if a new file dataset needs reading, and you already have a Reader of the same format, you could add the dataset to that Reader using the Source Dataset parameter. However, usually you would create a new Reader entirely.

Miss Vector says...

I want to convert some data from Esri Shapefile to GML format, with the GML having the same schema as an existing PostGIS database. How can I do this?

- 1. Create a workspace to convert the Shapefile to PostGIS. Create a second workspace to convert the PostGIS to GML.*
- 2. Create a workspace to convert the PostGIS to GML. Create a second workspace to convert the Shapefile to GML, overwriting the previously created GML file.*
- 3. Create a workspace with PostGIS and Shapefile Readers, and a GML Writer. Copy the PostGIS Reader feature types to the GML Writer and connect the Shapefile Reader feature types to them. Delete the PostGIS Reader and run the workspace.*
- 4. Create a workspace with a Shapefile Reader and a GML Writer. Import feature type schemas from the PostGIS database into the GML Writer.**

While it's true that #3 might do the job (#1 and #2 are just plain wrong) #4 is the best way. It works because you don't have to import the schema from the same dataset you are writing to - or even a dataset of the same format!

Exercise 8 Elevation Model Updates	
Data	Contours (Esri Shapefile) Raster Orthophoto (JPEG)
Overall Goal	Generate a 3D surface model using updated information
Demonstrates	Workspace components and parameters
Start Workspace	None
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex8-Complete.fmw C:\FMEData2017\Workspaces\DesktopBasic\Components-Ex8-Complete-Advanced.fmw

A new property development is being planned within the city. The planning department has asked for a 3D visualization of the city elevation model, taking this new development into account.

The task is to read the current elevation model, apply changes provided by the property developer, and create a three-dimensional output in Adobe PDF format.

Your manager has handed you this task, thinking that it will be very difficult. He doesn't know FME as well as you do!

1) Inspect Data

The first task is to read the existing contour files into a workspace. Examine the folder and you see that the contours are held in a set of Shapefile files:

Name	Date modified	Size
J10.dbf	07/02/2014 11:13 AM	2 KB
J10.prj	07/02/2014 11:13 AM	1 KB
J10.shp	07/02/2014 11:13 AM	35 KB
J10.shx	07/02/2014 11:13 AM	1 KB
J11.dbf	07/02/2014 11:13 AM	3 KB
J11.prj	07/02/2014 11:13 AM	1 KB
J11.shp	07/02/2014 11:13 AM	157 KB
J11.shx	07/02/2014 11:13 AM	1 KB
J12.dbf	07/02/2014 11:13 AM	2 KB
J12.prj	07/02/2014 11:13 AM	1 KB
J12.shp	07/02/2014 11:13 AM	66 KB
J12.shx	07/02/2014 11:13 AM	1 KB

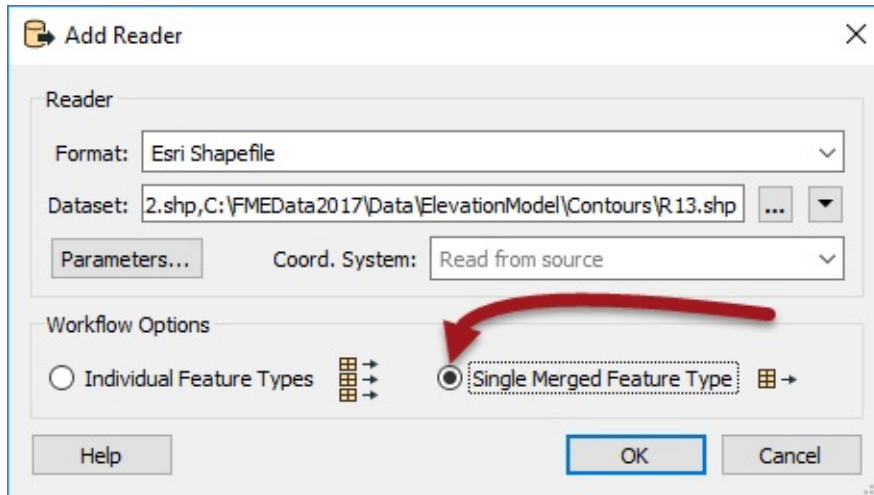
Feel free to use the FME Data Inspector to inspect these source files.

2) Add Reader

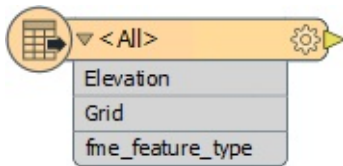
Start FME Workbench and begin with a blank workspace. Select Readers > Add Reader to read the contour files:

Reader Format	Esri Shapefile
Reader Dataset	C:\FMEDData2017\Data\ElevationModel\Contours*.shp

When it comes to selecting the data, select all of the Shape files in that folder. But don't click OK yet. First of all, select the workflow option for a "Single Merged Feature Type":



What this will do is add a single feature type with a Merge Filter automatically set:



Sister Intuitive says...

This is a HUGE shortcut here. It has added a single feature type and set a merge filter to read ALL data. In this scenario it makes the workspace way tidier and easy to use than if each file had its own feature type.

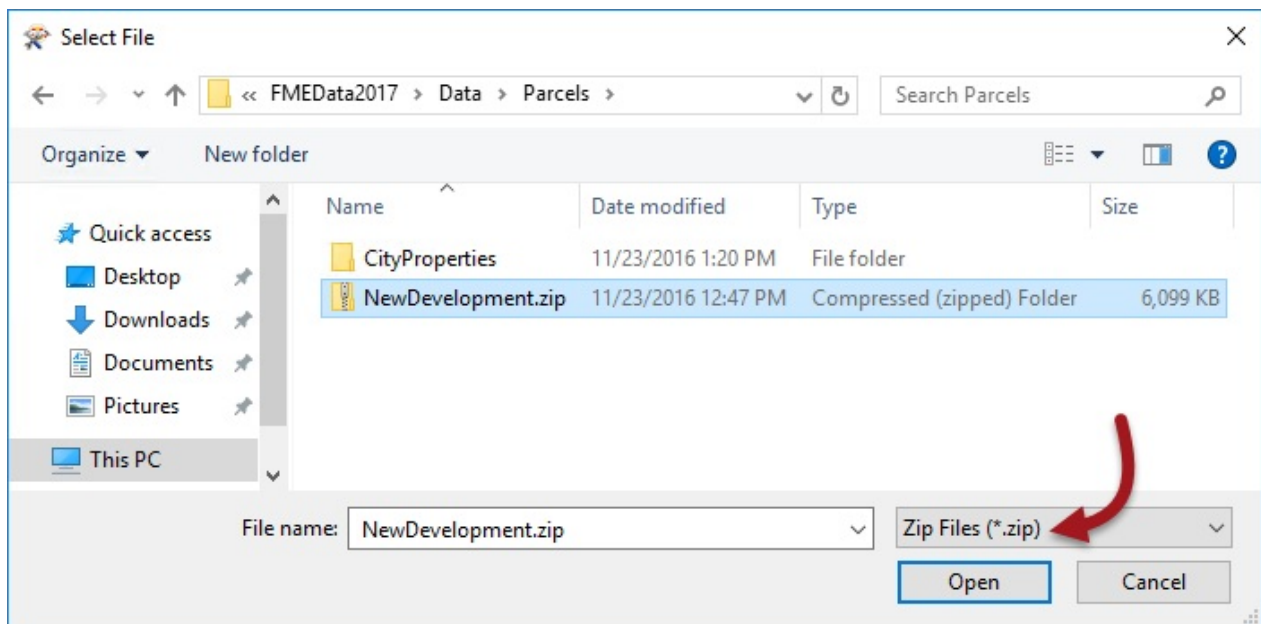
Of course, this only works where all of the source files have the same attribute schema (like here) and we are sure that we want to read all of the source layers (we do).

3) Add Reader

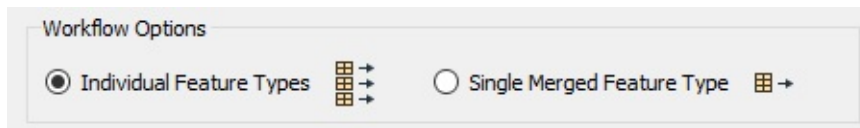
Now let's add the developer's data. Again select Readers > Add Reader from the menubar.

Reader Format	Esri Shapefile
Reader Dataset	C:\FMEDData2017\Data\Parcels\NewDevelopment.zip

Because the data has been sent in a zip file, we can still read it, but when we browse for the data we need to change the filter to look for zip files:



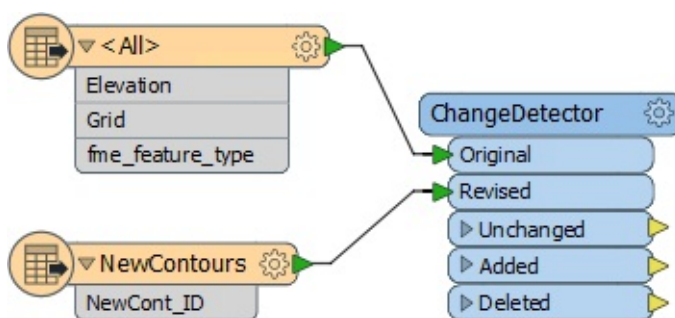
The other thing we should do is set the workflow option back to Individual Feature Types, in case there is data other than contours in there.



Now click OK. When prompted for which feature types you want from the zip file, select NewContours only.

4) Add ChangeDetector

The ChangeDetector transformer will tell us which contours are new in this dataset, which are replacements, and which are the same as before. So, place a ChangeDetector transformer and connect it in. The original contours should be connected to the Original port(!) and the revised contours to the Revised port.



You can check the parameters, but none should need changing. We can check in 2D because the contours are 2D with an elevation attribute.

5) Add Inspectors

Add Inspector transformers to the output ports of the ChangeDetector and run the workspace to see what we have got so far.

TIP

The Unexpected Input Remover might appear, but you know this isn't a problem, right? We deliberately chose not to read the BuildingFootprint or Labels from the source data.

You should find 3,967 contours were unchanged; i.e. they were in the original data and the new version. Seven (7) contours have been added; i.e. they appear in the new data but not the original version. Three (3) contours have been deleted; i.e. they appear in the original data but not in the new data.

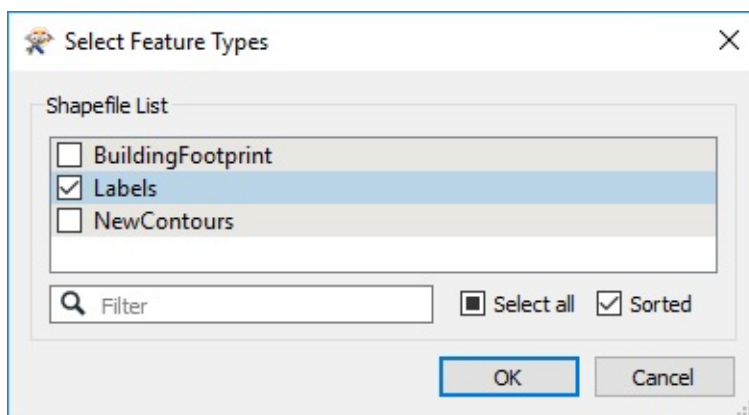
We'll want to generate the new elevation model from the Unchanged and the Added features; deleted ones we can drop. But first, there is a more pressing problem...

6) Import Feature Type Definition

Did you notice that the Added contours do not have an elevation value set? This is a problem because we can't generate a proper 3D model without elevation! It must be that the labels in the source data are meant to denote the contour elevation (you can use the Data Inspector to open the data and confirm this is so).

Therefore we need to add that data to the workspace. We already have a reader so you just need to add the feature type.

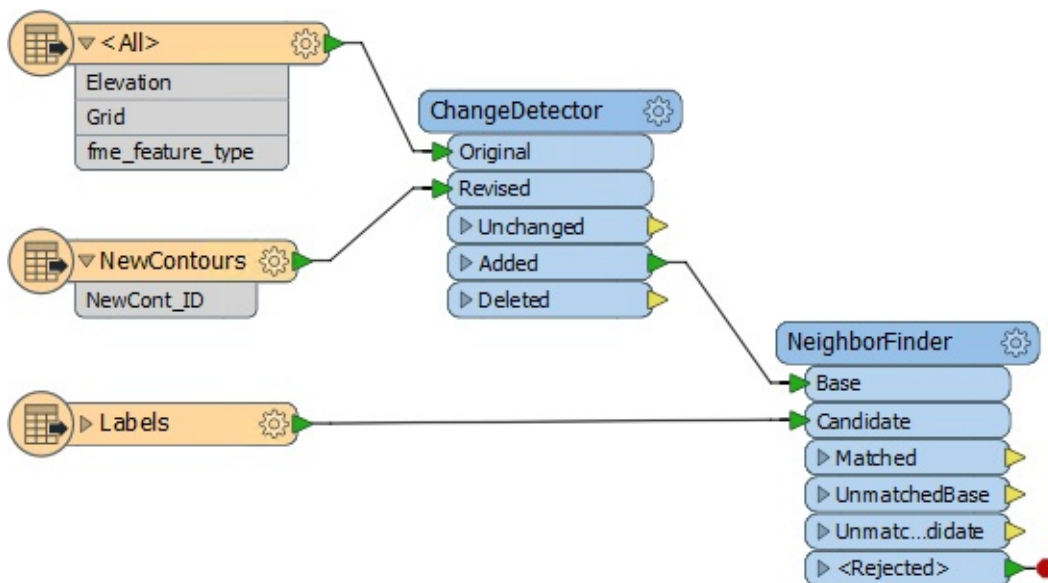
So choose Readers > Import Feature Type and select the NewDevelopment reader to import to. This time, when the list appears, be sure to select the Labels feature type:



Click OK and now you should get a new feature type. Because we're reading from a zip file we don't even need to change the dataset parameter; FME will pick up the data anyway.

7) Add NeighborFinder

A NeighborFinder transformer will transfer the label values onto the nearest contour. So, add a NeighborFinder transformer and connect it in. The ChangeDetector:Added features are the Base and the labels are the Candidate:

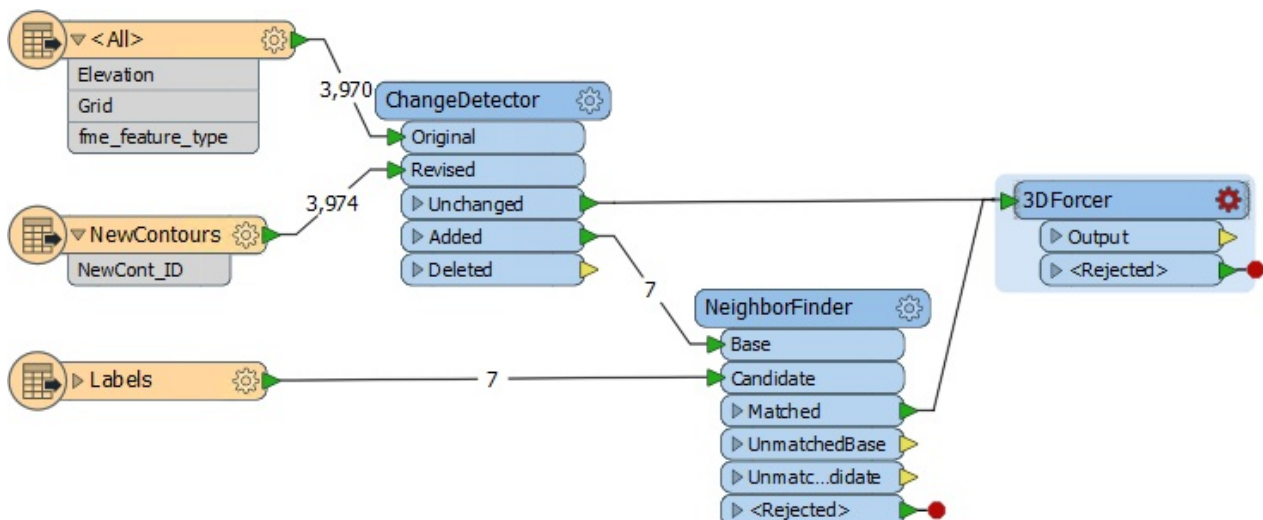


Check the parameters and set the Maximum Distance. This defines the maximum distance from the contour to the label. Because the labels appear to be almost on top of the contour you can set this to a low number, for example 5.

Now add some more Inspectors and rerun the workspace to ensure all seven contour features have an elevation value now.

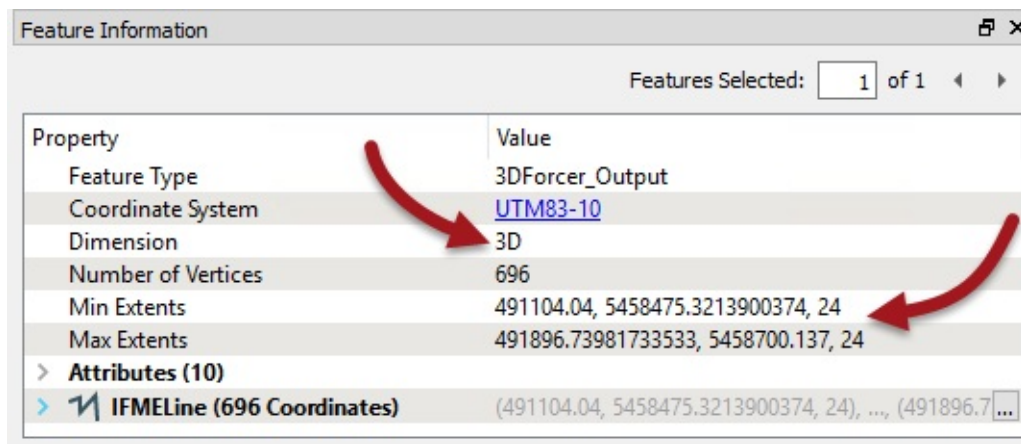
8) Add 3DForcer

Having an elevation attribute is fine, but to actually create a 3D model we must convert this into a true Z value. This is done with a 3DForcer transformer, so place one of these. The ChangeDetector:Unchanged and NeighborFinder:Matched are the features we need to process.



Now check the parameters. For Elevation, click the drop-down arrow and choose Attribute Value > Elevation.

Run the workspace again and you will see the contours now have a Z value and the Data Inspector reports they are true 3D features:



9) Add SurfaceModeller

Now we have 3D contours we can create a surface model. Add a SurfaceModeller transformer connected to the 3DForcer output. The input port to connect is Points/Lines.

Check the parameters. Set a Surface Tolerance of 1, and turn off the checkbox for Output Contours (it will run faster without). If you now connect an Inspector to the SurfaceModeller:TINSurface output port and run the workspace, you will be able to inspect the data in 3D.

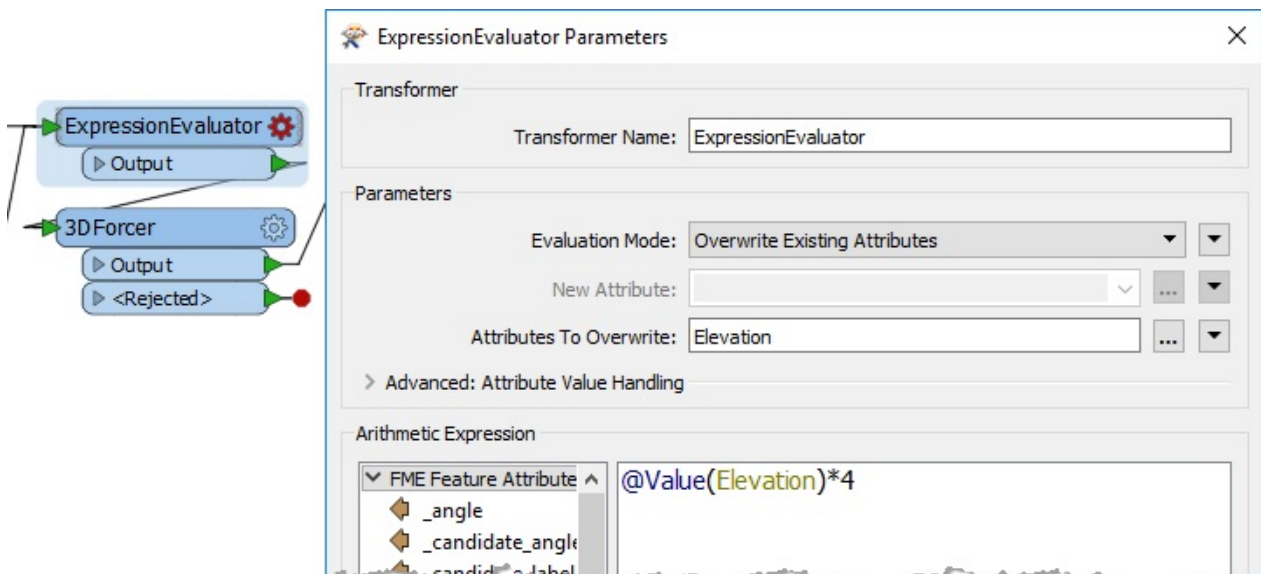
TIP

To rotate the data in 3D in the FME Data Inspector, click the orbit icon on the toolbar, or press down on the centre mouse button if you have one.

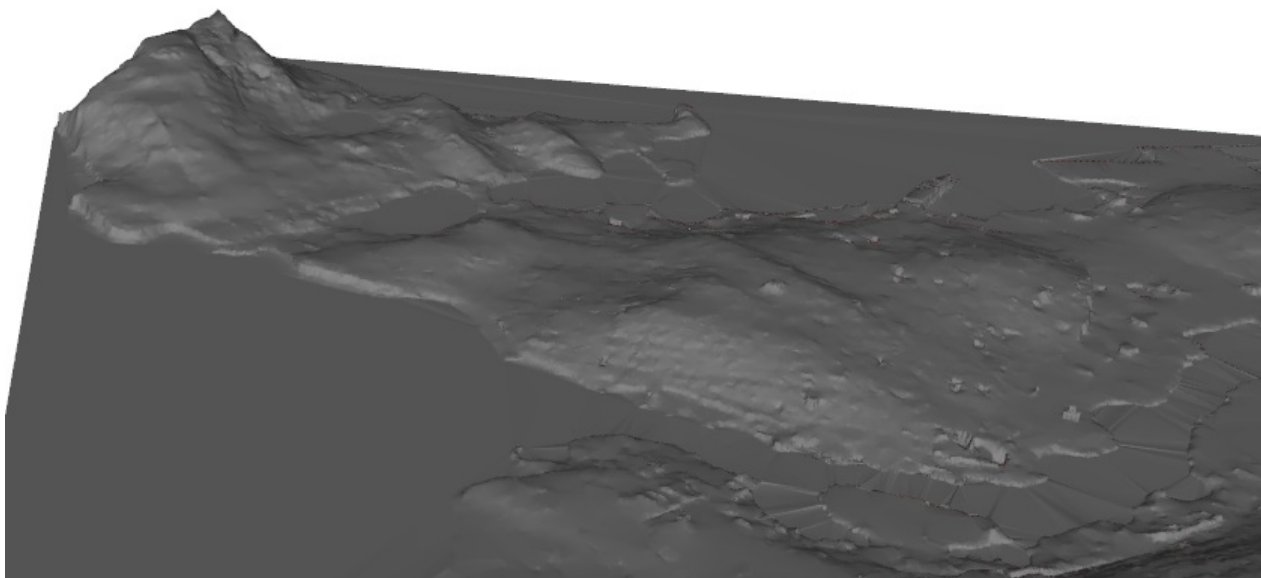
10) Exaggerate Scale (Optional)

If Vancouver seems a little flat, then why not exaggerate the Z scale a little to compensate!

Add an ExpressionEvaluator before the 3DForcer and use it to multiply the value of Elevation by a factor of 4. You'll want to use the option to apply the result to an existing attribute:



Now Vancouver will have a more interesting profile:



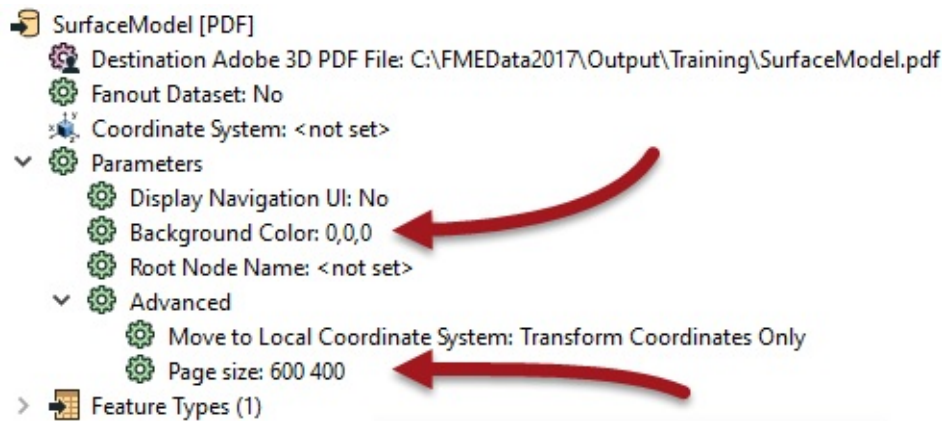
11) Add Writer

Now let's write this data. Select Writers > Add Writer and add a writer with the following specifications:

Writer Format	Adobe 3D PDF
Writer Dataset	C:\FMEData2017\Output\Training\SurfaceModel.pdf

Make sure you choose the 3D (not 2D) version of the PDF writer. You can choose to add a new feature type for the output in whatever way you want. The best method will be to select either Manual or Automatic from the Add Writer dialog. Give the feature type a name such as Surface and connect it up to the SurfaceModeller:TINSurface port.

In the writer parameters (Navigator window) locate the background color parameter and change it to black (0,0,0). Locate the Page Size parameter and change it to 600 400:



Now run the translation. Locate the output data - it will take a minute or so to write - and open it in Adobe Reader.

Advanced Exercise

You've now used readers and writers (and transformers) to update an elevation model and create a 3D rendering of it. But there is so much more we can do with this.

As an advanced task let's drape a raster image onto the surface model to make it look prettier.

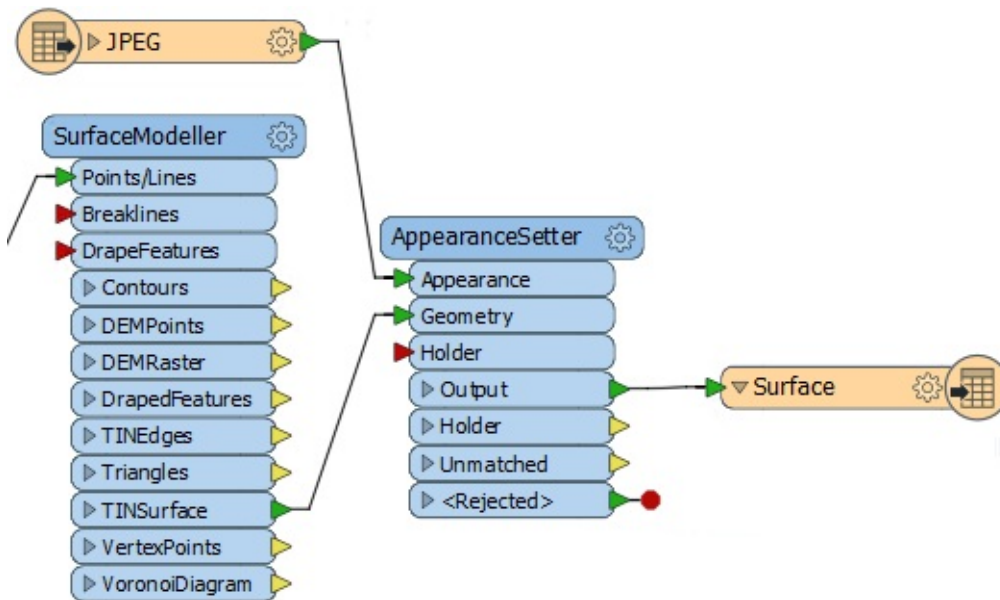
12) Add Reader

Add a reader to read the raster data. The details are:

Reader Format	JPEG (Joint Photographic Experts Group)
Reader Dataset	C:\FMEDData2017\Resources\DesktopBasic\OverlayImage.jpg

13) Add AppearanceSetter

Now add an AppearanceSetter transformer. This is used to apply the raster as an "appearance" or texture onto the DEM surface. The Raster data is the Appearance and the TINSurface port is the Geometry:



Check the parameters. Change "Set Appearance On" to be Front Side.

Expand the section Texture Coordinate Generation Parameters and set "Texture Mapping Type" to be From Top View:

Geometry Part Selection

Geometry XQuery: ... ▼

Set Appearance On: ▼ ▼

Appearance Join Attribute: ▼ ▼

Geometry Join Trait: ▼

Appearance Storage: ▼ ▼

> Color Parameters

> Texture Parameters

▼ Texture Coordinate Generation Parameters

Use Existing Texture Coordinates: ▼ ▼

Texture Mapping Type: ▼ ▼

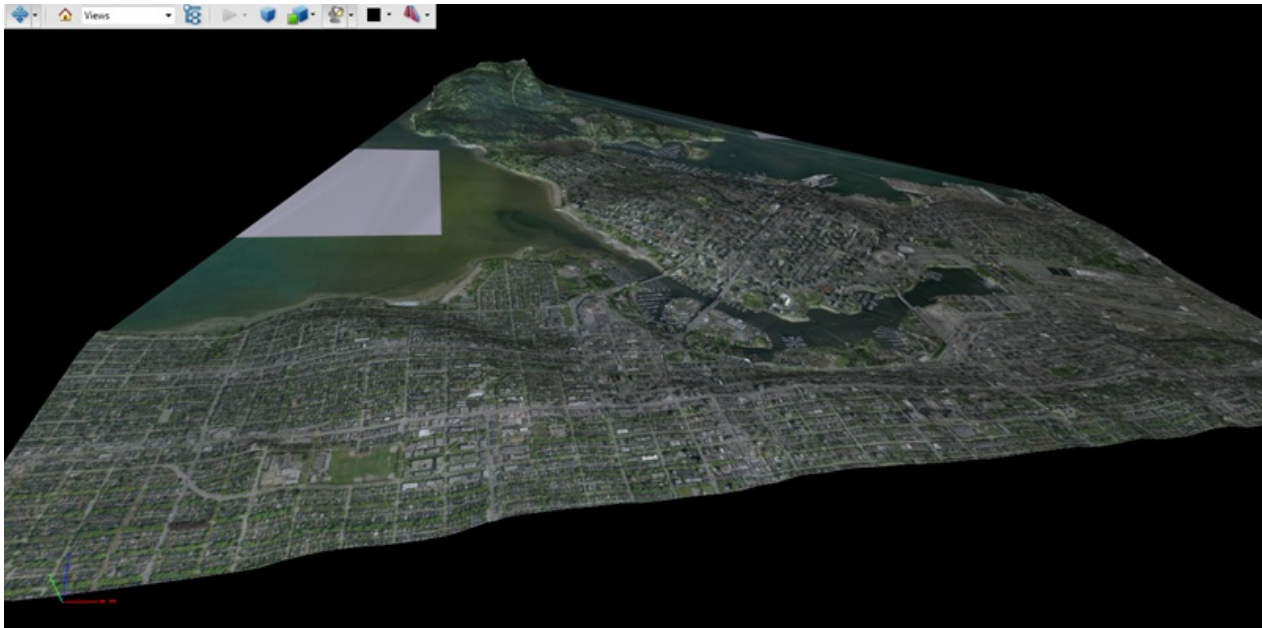
u Origin Offset: ▼

v Origin Offset: ▼

Texture u Repeat Factor: ▼

Texture v Repeat Factor: ▼

Run the workspace (be sure to close the PDF first if it is open in Adobe Reader). This time the PDF output should have a raster image draped on it.



WARNING

By default the PDF output will now be close to 100mb in size. If this is too large for your computer to handle, add a `RasterResampler` transformer before the `JPEG` feature type and set the parameters to resample to 25% of the original.

The PDF will now only be 8mb in size, but the quality will, of course, be much less.

14) Overlay Building Footprint

In the previous output, did you spot the lump in the surface where the new property is going to be developed? The one last thing we could do is add the building outline so that we can see where it will go.

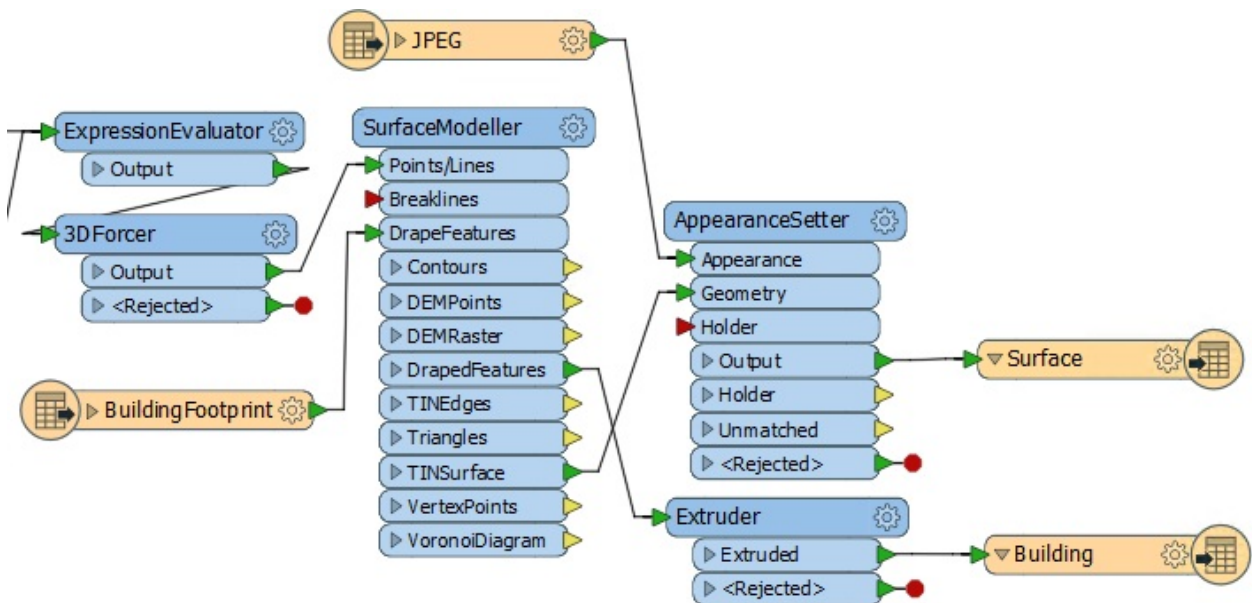
Because we don't have the building data in the workspace the first task is to use Import Feature Types to add that back to the NewDevelopment Shapefile reader.

Connect that up to the `SurfaceModeller:DrapeFeatures` input port (i.e. we want to drape the building onto the 3D surface that is being generated).

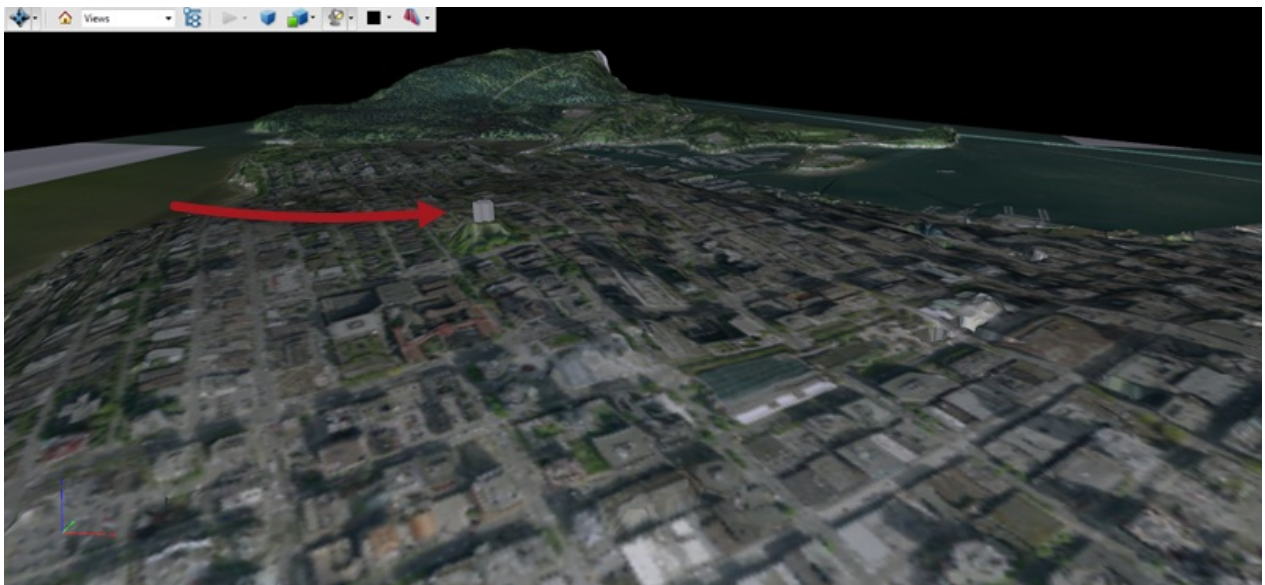
Add an `Extruder` transformer, connected to the `SurfaceModeller:DrapedFeatures` output port.

Set the `Extruder` parameters to extrude the data by a height of 25. This will create a 25m high building.

Now add a new writer feature type to create a new output layer for the buildings. Connect it to the `Extruder` transformer and you will have a workspace that looks like this:



Run the workspace and the output will now look like this:



Wow! Even allowing for the 4x scale increase on the DEM, this building is very prominent on the Vancouver skyline!

CONGRATULATIONS

By completing this exercise you proved you know how to:

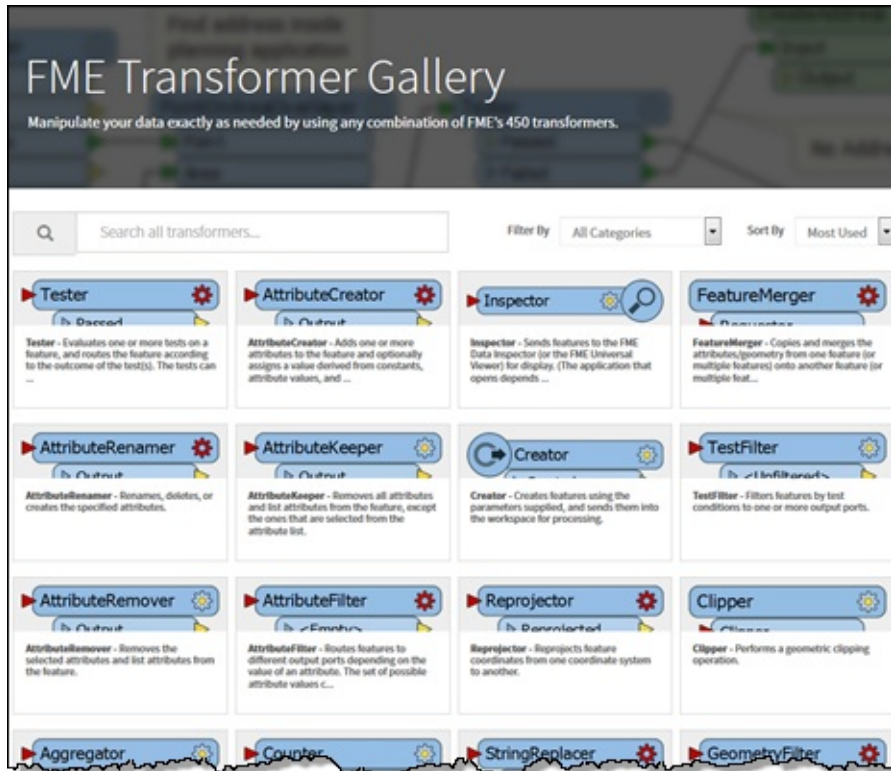
- *Add readers and writers to a workspace*
- *Set reader and writer parameters*
- *Import reader feature types*
- *Read data from a zip file*

And have learned how to:

- *Use the Merge Feature Type option when adding a reader*
- *Convert contours to a 3D surface model*
- *Drape a raster image onto a 3D surface model*

Practical Transformer Use

Even experienced FME users find the full list of transformers a daunting sight. In this section you'll learn to stop worrying and love the transformer gallery.



With over four hundred (400) transformers FME possesses a lot of functionality; probably a lot more than a new user realizes, and much of which would be very useful to them. This section helps find the transformer you need, even if you didn't realize you needed it.

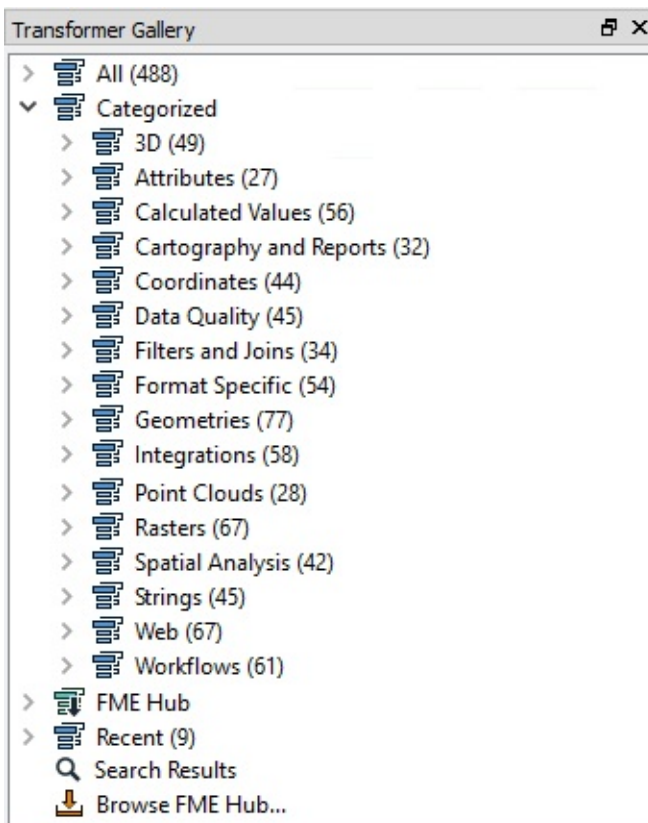
Although the transformer list can look a bit overwhelming, don't panic! The reality is that most users focus on 20-30 transformers that are relevant to their day-to-day workflow. You don't need to know every single transformer to use FME effectively.

Transformer Gallery

The transformer gallery is the obvious place to start looking for transformers. There are a number of ways in which transformers here can be located.

Transformer Categories

Transformer categories are a good starting point from which to explore the transformer list. Transformers are grouped in categories to help find a transformer relevant to the problem at hand.



NEW

All transformers were re-categorized for 2017. Some long-existing categories were dropped and some new ones created. Altogether, the chosen categories are a better reflection of what a transformer affects, are less ambiguous than previous categories, and are better synchronized with categories in other FME resources such as the Knowledge Centre.

Although all of them are important, the most commonly used transformers are found in these categories:

- **Attributes:** Operations for attribute/list management
- **Calculated Values:** Operations that return a calculated value

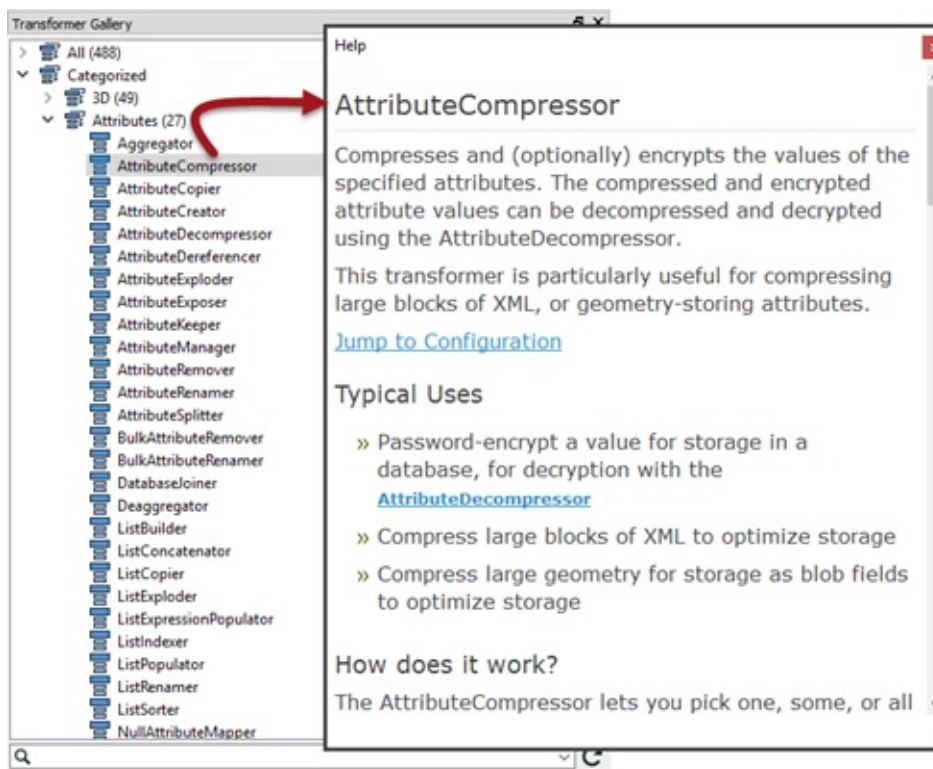
- **Filters and Joins:** Operations for dividing and merging data flows
- **Geometries:** Operations that create geometry or transform it to a different geometry type
- **Spatial Analyses:** Operations that return the result of a spatial analysis
- **Strings:** Operations that manipulate string contents, including dates

Simply click on the expand button to show all transformers within a particular category.

Transformer Help

The FME Workbench Help tool displays information about transformers. Simply click on a transformer and press the F1 key to open the help dialog.

This tool is linked to FME Workbench so that a transformer selected (in the gallery or on the canvas) triggers content to display in the Help tool.



TIP

Another useful - and printable - piece of documentation is the [FME Transformer Reference Guide](#).

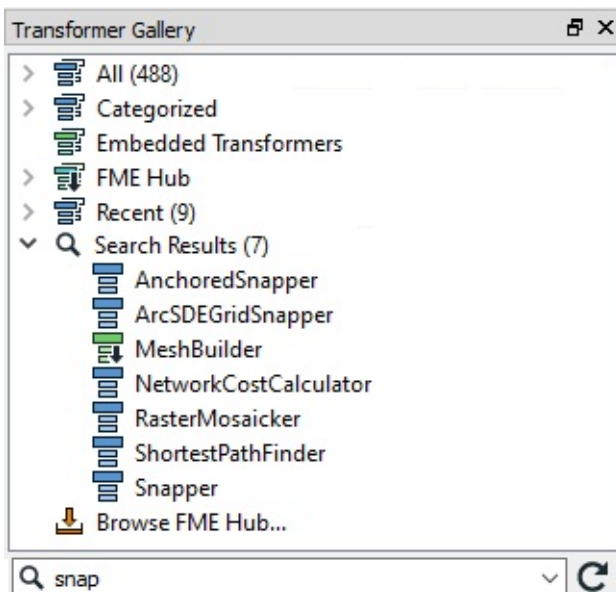
Transformer Searching

There are search functions in both the transformer gallery and Quick Add dialog.

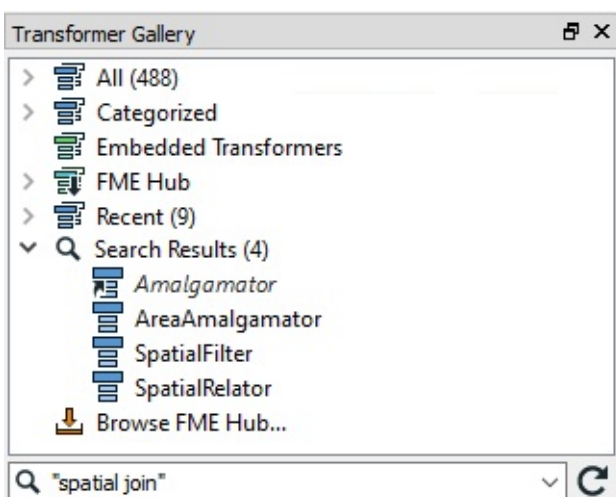
Transformer Gallery Search

To perform a search in the transformer gallery, simply enter the search terms and either press the key or click the search icon (the binoculars icon).

The transformer gallery search searches in both name and description. Therefore a search term may be the exact name of a transformer, or it may be a general keyword referring to functionality in general:

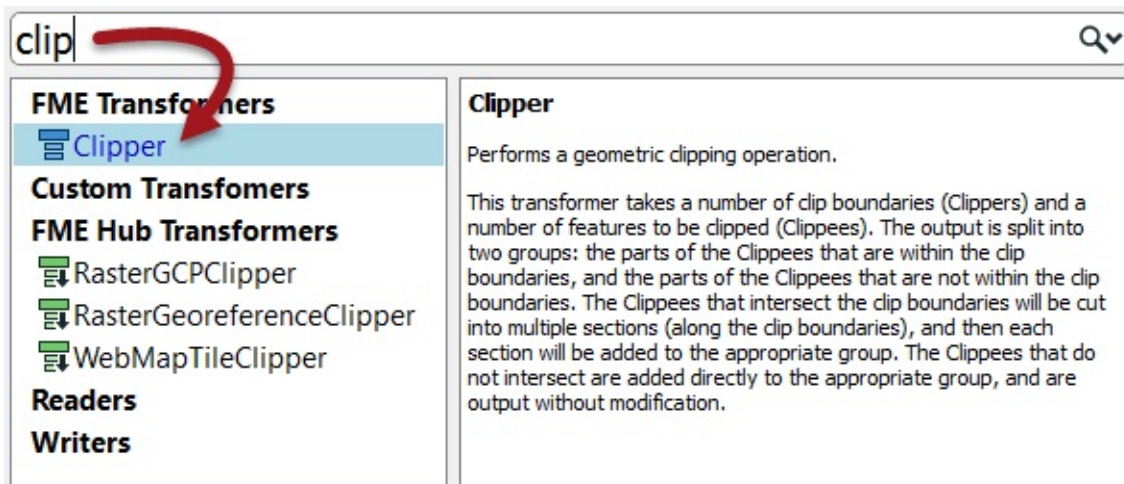


Search terms can either be full or partial words, and may consist of a number of keywords, including quote marks to enclose a single search reference.



Quick Add Search

Quick Add search-terms can also be full or partial words:



clip

FME Transformers

- Clipper**

Custom Transformors

FME Hub Transformers

- RasterGCPClipper
- RasterGeoreferenceClipper
- WebMapTileClipper

Readers

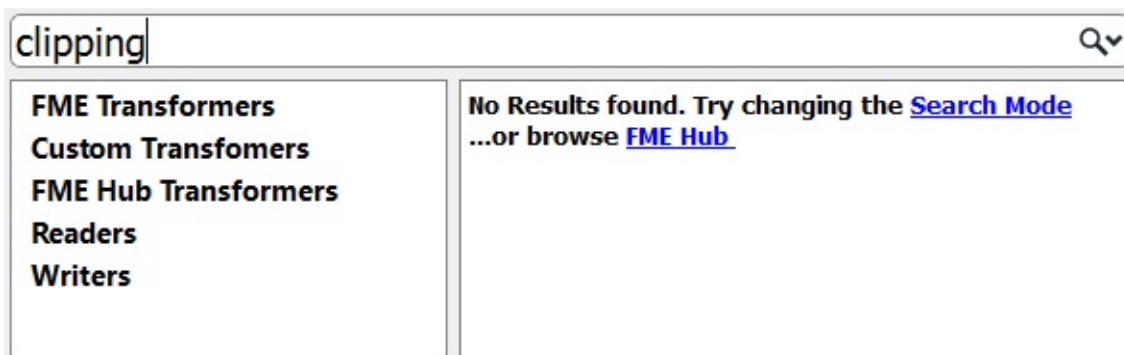
Writers

Clipper

Performs a geometric clipping operation.

This transformer takes a number of clip boundaries (Clippers) and a number of features to be clipped (Clippees). The output is split into two groups: the parts of the Clippees that are within the clip boundaries, and the parts of the Clippees that are not within the clip boundaries. The Clippees that intersect the clip boundaries will be cut into multiple sections (along the clip boundaries), and then each section will be added to the appropriate group. The Clippees that do not intersect are added directly to the appropriate group, and are output without modification.

By default, Quick Add does not look in transformer descriptions, so the search term must be the actual name of a transformer.



clipping

FME Transformers

Custom Transformors

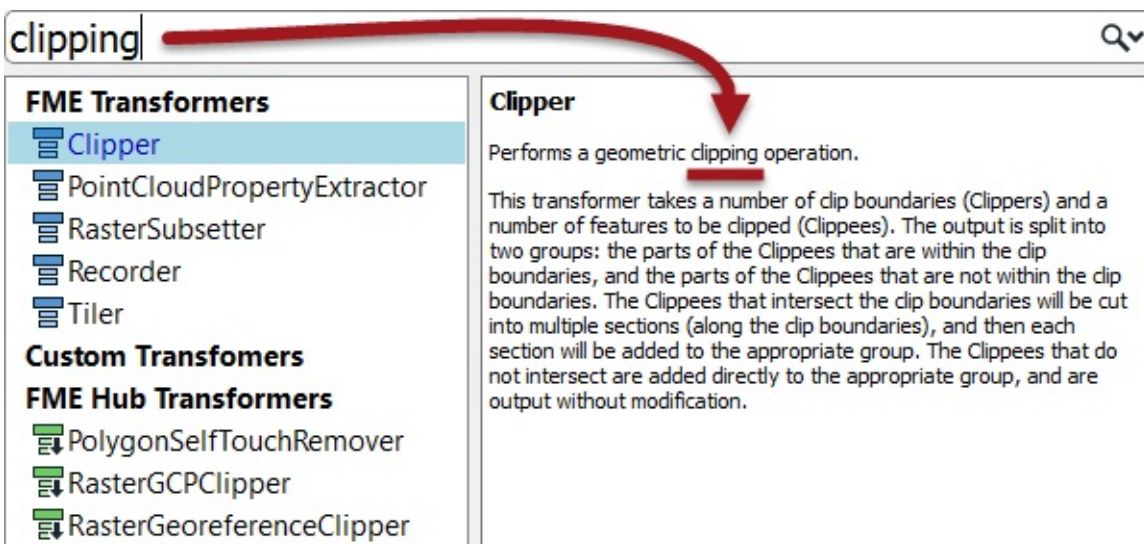
FME Hub Transformers

Readers

Writers

No Results found. Try changing the [Search Mode](#) ...or browse [FME Hub](#)

However, Quick Add will search in the transformer descriptions if you press the <TAB> key.



clipping

FME Transformers

- Clipper**
- PointCloudPropertyExtractor
- RasterSubsetter
- Recorder
- Tiler

Custom Transformors

FME Hub Transformers

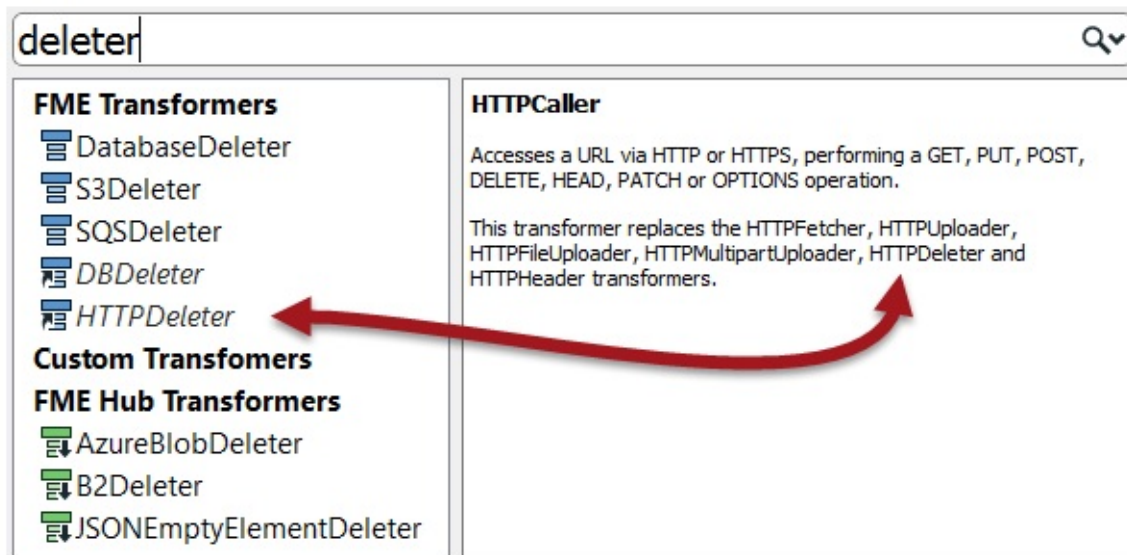
- PolygonSelfTouchRemover
- RasterGCPClipper
- RasterGeoreferenceClipper

Clipper

Performs a geometric clipping operation.

This transformer takes a number of clip boundaries (Clippers) and a number of features to be clipped (Clippees). The output is split into two groups: the parts of the Clippees that are within the clip boundaries, and the parts of the Clippees that are not within the clip boundaries. The Clippees that intersect the clip boundaries will be cut into multiple sections (along the clip boundaries), and then each section will be added to the appropriate group. The Clippees that do not intersect are added directly to the appropriate group, and are output without modification.

Quick Add results include aliases - for example transformers that have an alternative name or which have been renamed - and also include transformers found in the FME Hub.

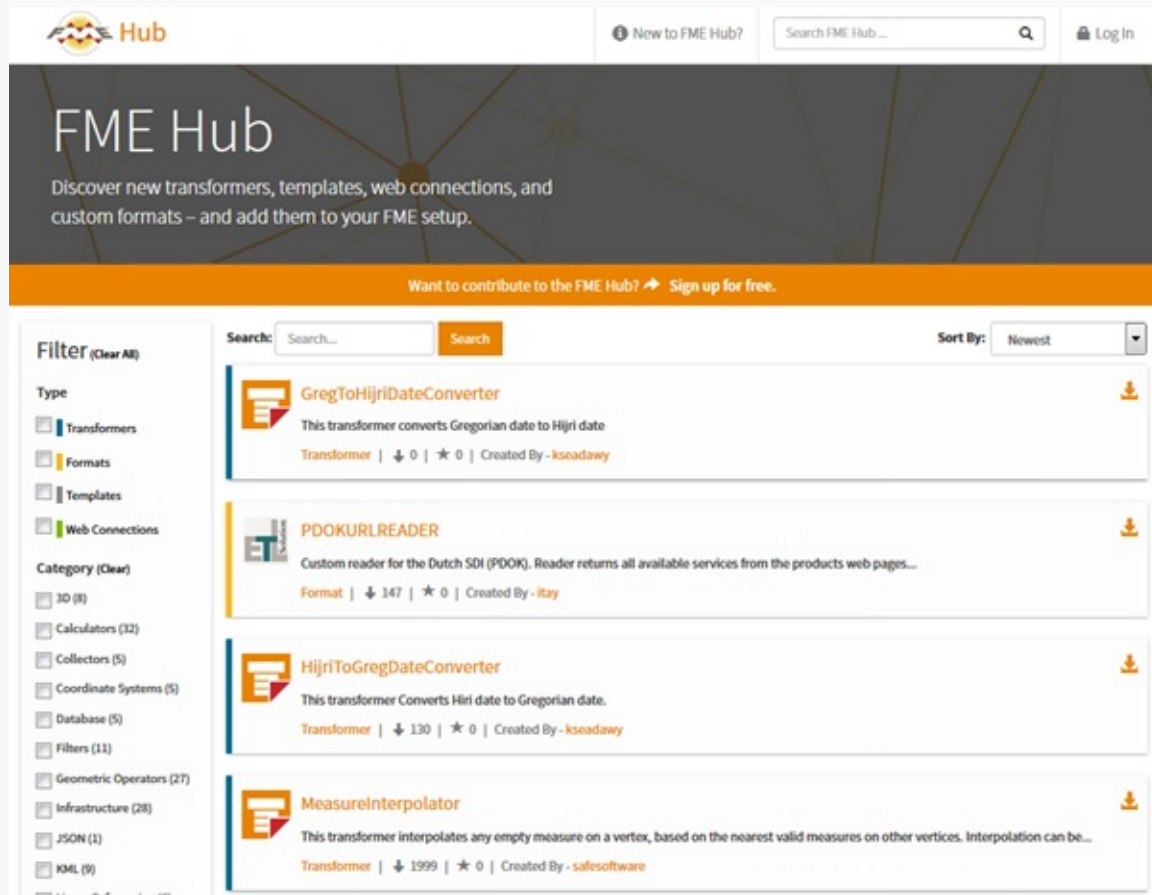


.1 UPDATE

Remember, in 2017.1 the help contents of the Quick Add window are now simplified, but include a clickable link to the full documentation.

Firefighter Mapp says...

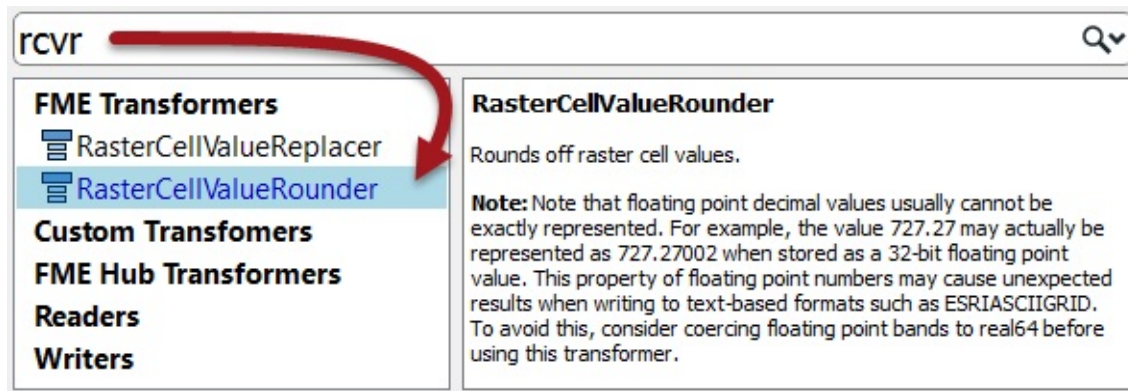
in case you weren't aware, the FME Hub (hub.safe.com) is a facility for sharing FME functionality such as custom transformers, web connections, and formats:



Transformers from the hub are shown in Quick Add with a small, downwards-pointing arrow, to denote that they will be downloaded if selected.

CamelCase

Quick Add also allows the use of CamelCase initials as a shortcut. CamelCase is where a single keyword is made up of several conjoined words, each of which retains an upper case initial; for example AttributeFileWriter (AFW) or ShortestPathFinder (SPF).



Miss Vector says...

Try these questions to see if you can search for transformers.
Which of the following is NOT a category of transformers?

1. Attributes
2. Calculations
3. Data Quality
4. Workflows

Here are four transformers and four categories. Match the transformer to the correct category.

Scenario	Tool
Chopper	Workflows
Terminator	Strings
Matcher	Geometries
DateFormatter	Data Quality

Most Valuable Transformers

If you have a thorough understanding of the most common transformers then you have a good chance of being a very efficient user of FME Workbench.

Anyone can be proficient in FME using only a handful of transformers; if they are the right ones!

The Top 25

The list of transformers on the Safe Software web site is ordered by most-used, calculated from user feedback. Having this information tells us where to direct our development efforts in making improvements, but it also gives users a head-start on knowing which of the (400+) FME transformers they're most likely to need in their work.

The following table (last updated August 2017) provides the list of the most commonly used 25 transformers. The Tester transformer is consistently number one in the list every year, highlighting its importance.

Rank	Transformer	Rank	Transformer
1	Tester	2	AttributeCreator
3	AttributeManager	4	FeatureMerger
5	Creator	6	Inspector
7	AttributeKeeper	8	TestFilter
9	Clipper	10	Reprojector
11	AttributeRenamer	12	Aggregator
13	FeatureReader	14	AttributeFilter
15	VertexCreator	16	AttributeRemover
17	StringReplacer	18	Counter
19	Bufferer	20	StatisticsCalculator
21	SpatialFilter	22	GeometryFilter
23	AttributeExposer	24	Sorter
25	Dissolver		

Besides the obvious transformers for transforming geometry (Clipper, Bufferer, Dissolver) and the obvious transformers for transforming attribute values (StringReplacer, Counter) there are some other distinct groups of transformers.

Managing Attributes

These transformers - mostly named the *Attribute<Something>* - are primarily for managing attributes (creating, renaming, and deleting) for schema mapping purposes. However they can also be used to set new attribute values or update existing ones.

TIP

The AttributeManager is a multi-purpose transformer for carrying out most attribute-related functionality. It is slowly rising up the list, above transformers it replaces, like the AttributeRemover and AttributeRenamer.

Filtering

These transformers - mostly named the *<Something>Filter* - subdivide data as it flows through a workspace. Commonly the filter is a conditional filter, where the decision about which features are output to which connection is decided by some form of test or condition.

Data Joins

Joins are the opposite action to filtering; they are when separate streams of data are combined as they flow through a workspace. Like filtering there is a condition to be met - in this case matching key values - that determine how and where the join takes place.

Managing Attributes

A high proportion of the top 25 transformers are support transformers for managing attributes. These create new attributes, rename them, set values, and delete them.

A key use for these transformers is to rename attributes for the purpose of schema mapping.

Attribute Managing Transformers

The key attribute-management tasks and the transformers that can be used are as follows:

Task	Transformers
Create Attributes	AttributeCreator, AttributeManager
Set Attribute Values	AttributeCopier, AttributeCreator, AttributeManager, AttributeRenamer
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

Many of these transformers can carry out similar operations, and you can see that the `AttributeManager` does so many tasks you can use it almost exclusively.

WARNING

*Don't misunderstand the `BulkAttributeRenamer`. It changes the case - or adds suffixes/prefixes - to the attribute **name**, not the attribute value.*

Lists

A List in FME is a mechanism that allows multiple values per attribute. So, where the attribute *myAttribute* can only contain a single value, the list attribute *myList{0}.myAttribute* can contain multiple values as:

```
myList{0}.myAttribute  
myList{1}.myAttribute  
myList{2}.myAttribute
```

For example, a single polygon representing a forested area, might have a list attribute to record the tree species contained in that area:

```
TreeList{0}.Species = Oak  
TreeList{1}.Species = Chestnut  
TreeList{2}.Species = Ash
```

Various transformers can create lists, others include support for lists, and several are designed to operate specifically on list attributes (for example the ListSorter).

For further reading check out [this article on Attribute Management](#) on the Safe Software blog.

Creating and Setting Attributes

Creating attributes and setting a value are probably the primary attribute function used within FME. When an attribute is created its value can be set in any one of a number of ways.

The transformers capable of creating an attribute - and setting its value - are:

- AttributeCreator
- AttributeManager

The AttributeCopier and AttributeRenamer transformers can set an attribute value, but only if it exists already.

The AttributeManager

For most operations we'll concentrate on the AttributeManager, so here is a quick overview of that transformer.

The AttributeManager parameters dialog has a number of fields: Input Attribute, Output Attribute, Attribute Value, and Action. Uniquely among attribute-handling transformers, it is automatically filled in with the details of the attributes connected to it:

AttributeManager Parameters

Transformer

Transformer Name:

> Advanced: Attribute Value Handling

Attribute Actions

Input Attribute	Output Attribute	Attribute Value	Action
ParkId	ParkId		Do Nothing
ParkName	ParkName		Do Nothing
NeighborhoodName	NeighborhoodName		Do Nothing
DogPark	DogPark		Do Nothing
	<Add new Attribute>		

+ - < > <=>

Filter:

Import...

Help Defaults OK Cancel

The action field can be set by the user, but is also set automatically when a change is made to the other fields.

Manually Create an Attribute

By entering a new attribute name into the Output Attribute field, it will be created in the output.

Attribute Actions

Input Attribute	Output Attribute	Attribute Value	Action
ParkId	ParkId		Do Nothing
ParkName	ParkName		Do Nothing
NeighborhoodName	NeighborhoodName		Do Nothing
DogPark	DogPark		Do Nothing
	City		Set Value
	<Add new Attribute>		

+ - ▲ ▼ ≡ ☰
 Filter: Import ...

The text <Add new Attribute> highlights where a new attribute can be created. By default, when the Attribute Value field is empty, a new attribute has no value.

Set a Fixed Attribute Value

A fixed (or *constant*) value for an attribute can be created by simply entering a value into the Attribute Value field:

Attribute Actions

Input Attribute	Output Attribute	Attribute Value	Action
ParkId	ParkId		Do Nothing
ParkName	ParkName		Do Nothing
NeighborhoodName	NeighborhoodName	<input type="checkbox"/> Kitsilano	Set Value
DogPark	DogPark		Do Nothing
	City	<input type="checkbox"/> Vancouver	Set Value
	<Add new Attribute>		

+ - ▲ ▼ ≡ ☰
 Filter: Import ...

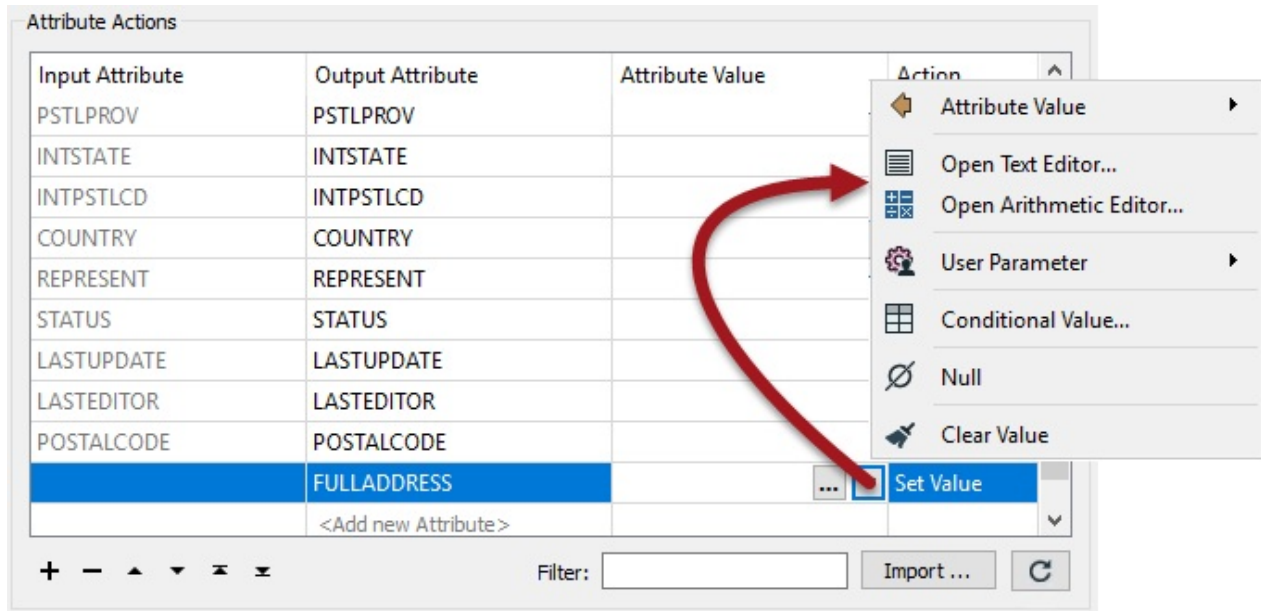
Here, for example, a new attribute called City is being given a fixed value of Vancouver.

However, also note that the existing attribute NeighborhoodName is also being assigned a fixed value. It is being given the value "Kitsilano". Notice how by entering a value into that field, the Action field has automatically changed from "Do Nothing" to "Set Value".

Besides entering set values like this, it's possible to construct an attribute value in a number of different ways.

Constructing Attributes

Besides constant attribute values FME also allows you to construct values using string manipulation and arithmetic calculations. This is achieved using the menu opened by clicking on the arrow in the Attribute Value field:

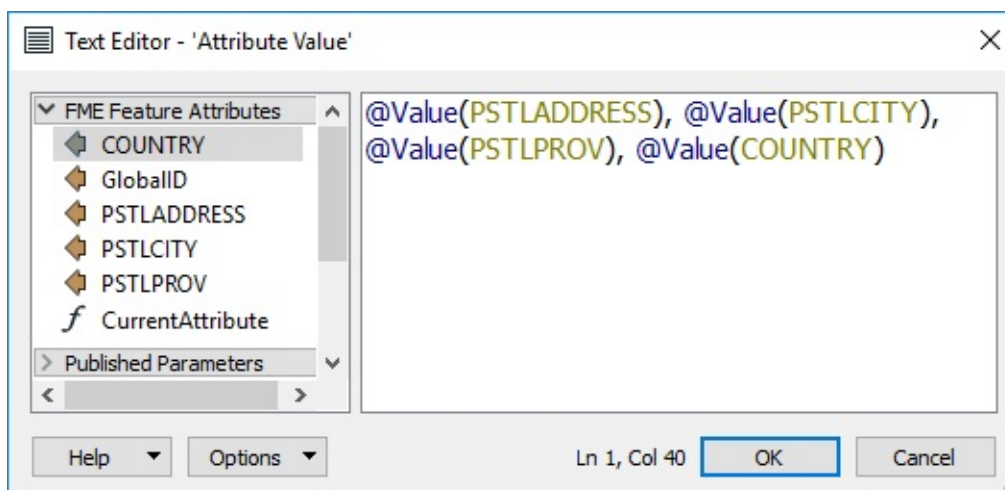


This is very useful because the attribute now no longer is a fixed value: it can be constructed from a mix of existing attributes and parameters. The two main methods are the Text Editor and Arithmetic Editor.

Text Editor

The text editor - as you would expect - allows you to construct a text value. It includes all the usual string-handling functionality you would need, such as concatenation, trimming, padding, and case changing.

The text editor looks like this:



Here the user is constructing a simple address string by concatenating various existing attributes. Notice the menu on the left hand side. Existing attributes are listed here and were added into the string by double-clicking them.

Also notice the other menu options. The most important (for a text editor) are the String Functions:



These are the functions that can be used to manipulate the strings being used. For example, here the user is making sure the attributes being used are trimmed when used:

```
@Trim(@Value(PSTLADDRESS)), @Trim(@Value(PSTLCITY)),  
@Trim(@Value(PSTLPROV)), @Trim(@Value(COUNTRY))
```

NEW

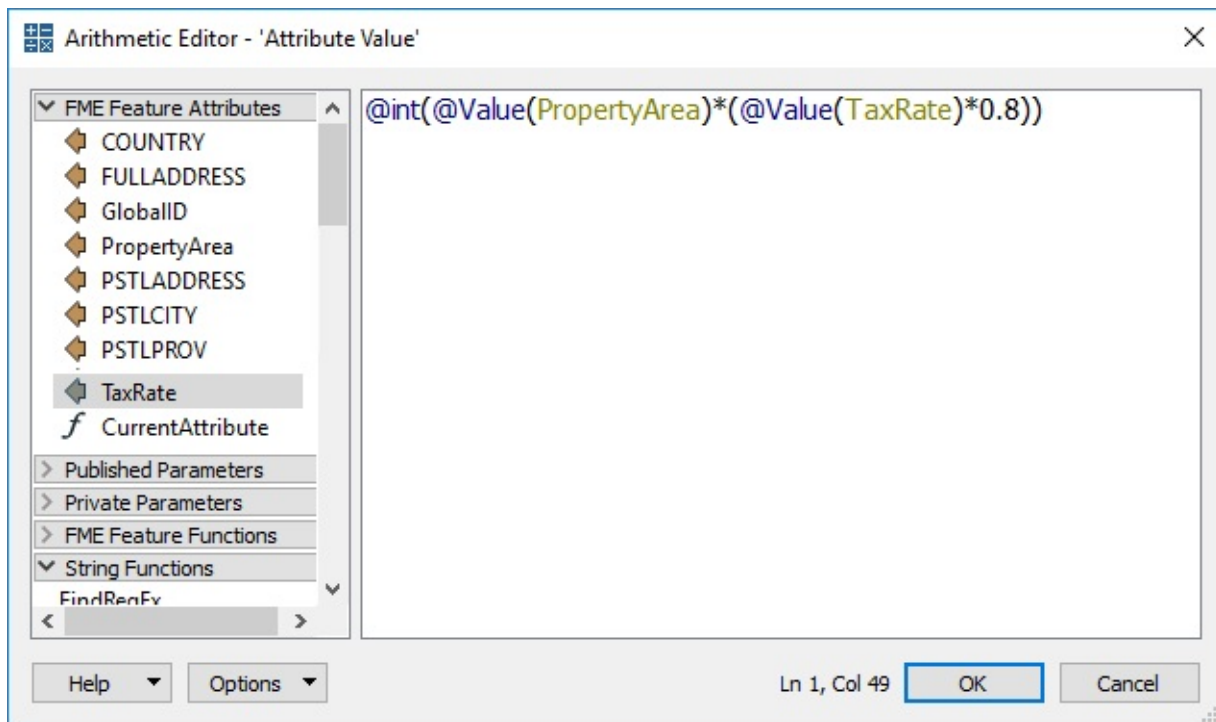
New for 2017 are a series of Date/Time functions in the text editor. These can be used to manipulate dates, times, and datetime strings.

.1 UPDATE

FME 2017.1 introduced additional Date/Time functions for handling TimeZone components.

Arithmetic Editor

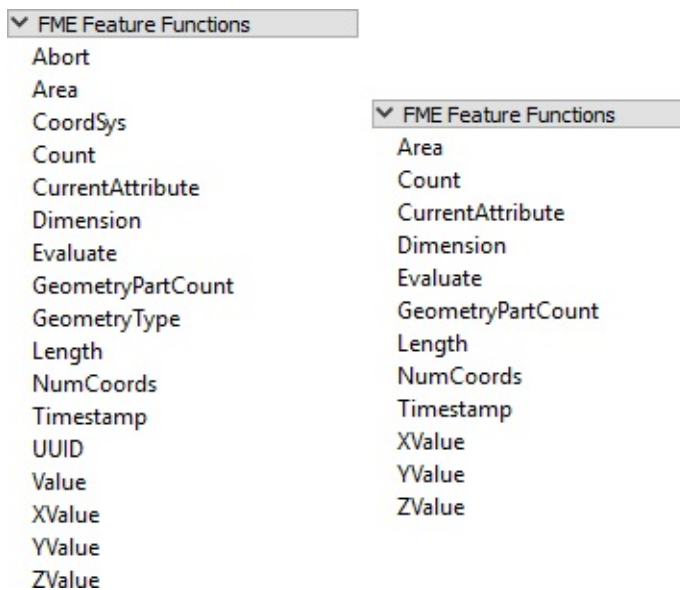
The arithmetic editor is much the same as the text editor, except that whatever is entered into the dialog will be evaluated as an arithmetic expression and a numeric result returned:



Here the user is calculating a resident's tax using a simple equation containing property area and a tax code. As with the text editor, existing attributes and arithmetic functions were obtained from the menu on the left-hand side.

FME Feature Functions

One other item in the menu of both text and arithmetic editors is FME Feature Functions:



These are functions that reach into the very heart of FME's core functionality. They are the building blocks that transformers are built upon; basic functionality that can return values to the editor.

For example, the `@Area()` function returns the area of the current feature (assuming it is a polygon). `@CoordSys()` returns the coordinate system. They are the functional equivalent of the `AreaCalculator` and `CoordinateSystemExtractor` transformers.

Some functions return strings, others return numeric values; therefore the available functions vary depending on whether the text or arithmetic editor is being used. In the screenshot above, the text editor functions are on the left and the arithmetic editor functions on the right. The text editor can use either text or numeric values; the arithmetic editor can only ever accept numeric values.

FME Feature Functions are useful because they allow you to build processing directly into the `AttributeManager`, instead of using a separate transformer.

.1 UPDATE

With the introduction of specific Date/Time functions, the `@Timestamp` function has been removed in FME2017.1. The `@DateTimeNow()` function - or `DateTimeStamper` transformer - can be used instead.

Replacing Other Transformers

Integrated text and arithmetic editors provide a great benefit for workspace creation. They allow attribute-creating functions to be carried out directly in a single transformer.

For example, the `AttributeManager` text editor can be used as a direct replacement for the `StringConcatenator` and `ExpressionEvaluator` transformers.

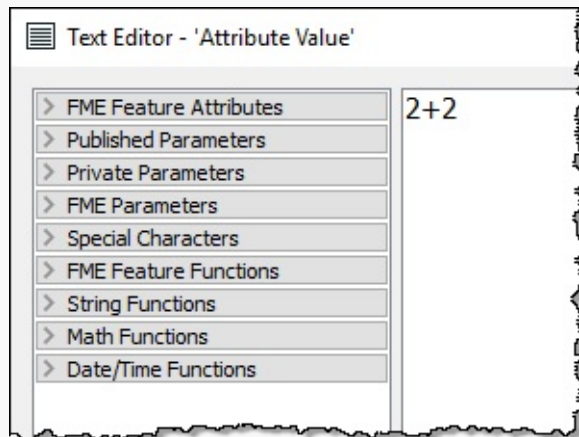
The `AttributeManager` could also replace the `StringPadder` and `AttributeTrimmer` transformers, albeit with a little less user-friendliness. If FME Feature Functions are used inside the editor, this transformer could also technically replace transformers such as the `AreaCalculator`, `LengthCalculator`, `CoordinateCounter`, `TimeStamper`, and many more.

This is usually a good thing. Workspaces will be more compact and well-defined when as many peripheral operations as possible are directly integrated into a single transformer. However, because it's possible for an `AttributeManager` to be carrying out many, many operations, it is also more important to use Best Practice and ensure it has proper annotation.

If an `AttributeManager` is not properly annotated, it isn't possible to determine from looking at the Workbench canvas what action it is carrying out!

Miss Vector says...

Here's a question to see if you are paying attention. Look at this screenshot of an editing dialog and tell me what the value returned to the attribute will be:



1. 2+2
2. 4
3. 4.0
4. Error!

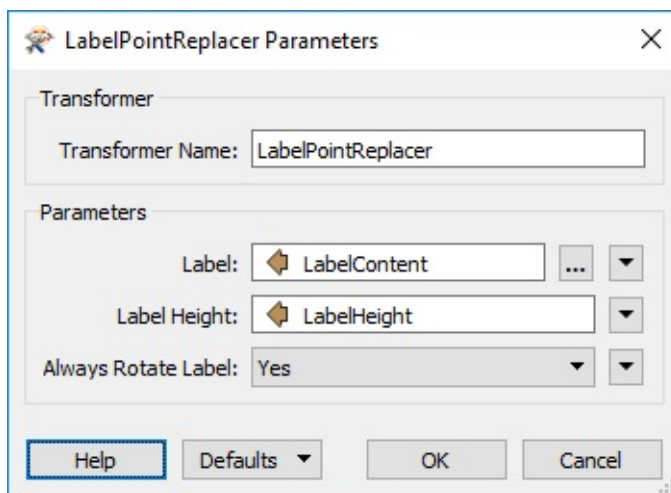
Constructing Parameters

Let's just put attributes to one side for a moment and look at transformer parameters.

Transformer parameters are often set in a fixed way (hard-coded) or set to take on the value of a particular attribute. However, in the same way that attributes can be constructed, text or arithmetic editors can be used to build values for transformer parameters.

Using Attributes for Parameters

As noted, most transformer parameters allow the user to select an attribute value instead of manually entering a fixed value. For example, the LabelPointReplacer can create a label whose contents and height are specified by attribute values:

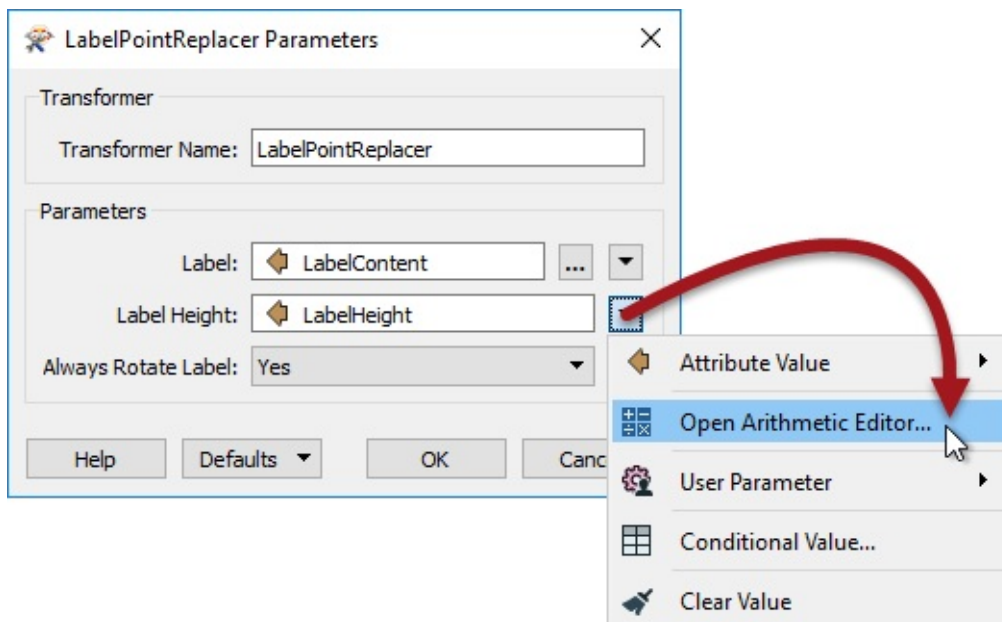


This is very useful because it allows the parameters (for example label size) to get a different value for each feature. An attribute could be read from a source dataset, or calculated using an ExpressionEvaluator, so that one feature creates a label 10 units in height, another creates a label 15 units high, and so on. It is no longer a fixed value.

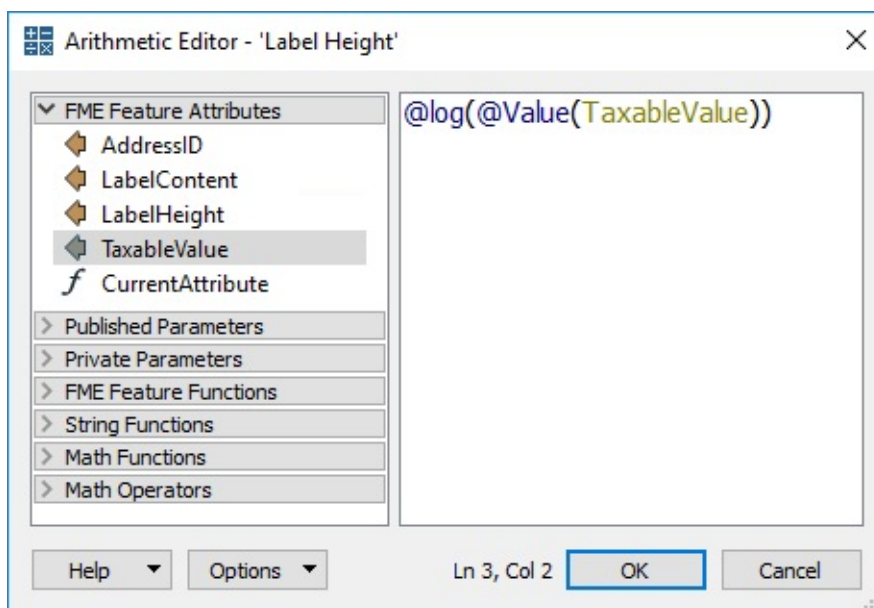
Constructing Parameter Values

If a parameter value needs to be calculated or constructed, instead of using a separate transformer, FME has integrated string and numeric editors built into parameter dialogs.

For example, here the user is choosing to calculate label height using an arithmetic calculator:



The calculator allows the selection and use of FME attributes, other parameters, plus a number of mathematical and string-based functions. For example, here the user has chosen to calculate the height of their labels using the logarithm of the taxable value:



Mr. E. Dict (attorney of FME law) says...

It's a fixed rule that the editor dialogs available depend upon the type of parameter being set. For instance, the Label parameter in a LabelPointReplacer opens in a text editor because the parameter requires a text value. The Label Height parameter opens in an arithmetic editor because that parameter requires a numeric value.

Reducing Workspace Congestion

Like when attribute values are constructed, workspaces are more compact when as many peripheral operations as possible are directly integrated into a single transformer or parameter. However, as with attributes, it's important to use Best Practice and ensure it has proper annotation, else it's difficult for a casual observer to understand what the workspace is meant to do.

Another drawback, specific to parameters, is that you don't also get the information as an attribute to use elsewhere. For example, if you construct a label string in the `LabelPointReplacer`, that string isn't available as an attribute elsewhere in the workspace.

Renaming and Copying Attributes

Renaming and - to a lesser extent - copying attributes are also key attribute functions within FME. When an attribute is renamed it ceases to exist under its prior name; when it is copied it exists both in its new and old names.

The transformers capable of renaming an attribute are:

Transformer	Capability
AttributeCopier	Copy
AttributeCreator	Copy
AttributeManager	Copy and Rename
AttributeRenamer	Rename

Renaming

The fundamental purpose of this transformer is to manually enter a new name for a selected attribute. The old attribute is removed and replaced with the newly named one.

Input Attribute	Output Attribute	Attribute Value	Action
GlobalID	GlobalID		<i>Do Nothing</i>
PSTLADDRESS	PostalAddress		Rename
PSTLCITY	PostalCity	<input type="checkbox"/> Vancouver	Rename
PSTLPROV	PostalProvince		Rename
COUNTRY	Country		Rename
	<Add new Attribute>		

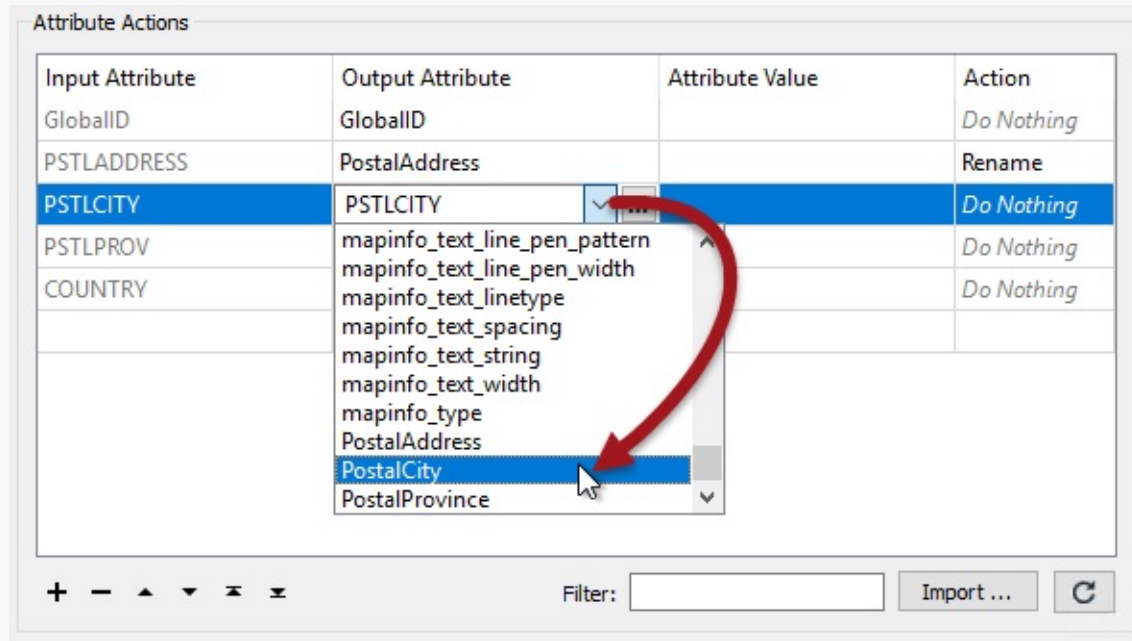
Filter: Import ...

Here the user is renaming a number of fields by entering a different name for the Output Attribute. The Action is automatically set to Rename. Notice that the user is also entering a new constant value for the PSTLCITY/PostalCity attribute.

This type of behaviour is obviously of use when the reader schema ('what we have') needs to be renamed to match the writer schema ('what we want').

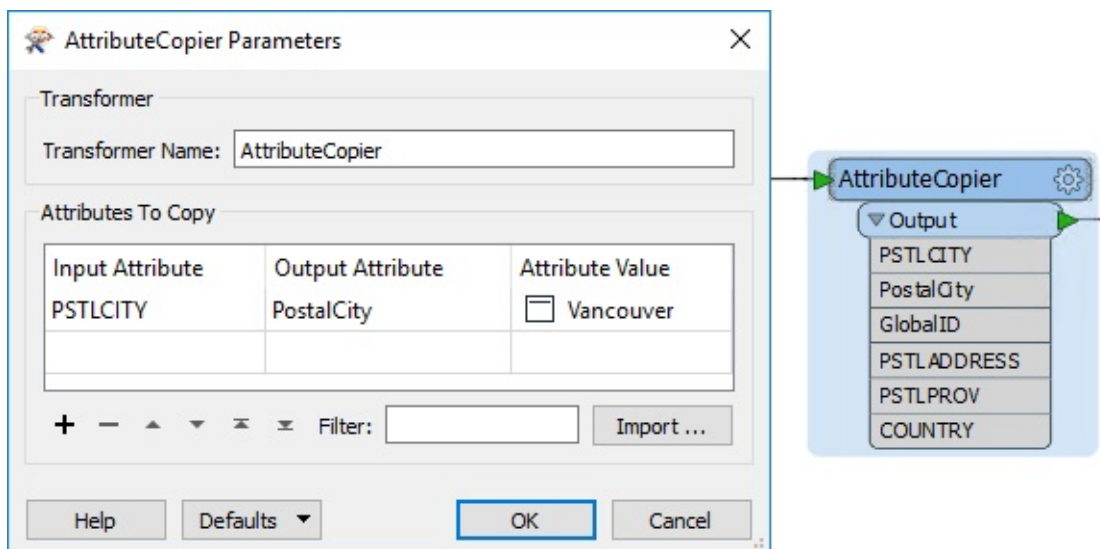
TIP

Although you can manually type a new attribute name into the Output Attribute field, if the transformer is connected to a writer feature type with the correct attributes, its attribute names will be automatically available to select from:



Copying

Depending on the transformer, copying an attribute can be one of two styles.



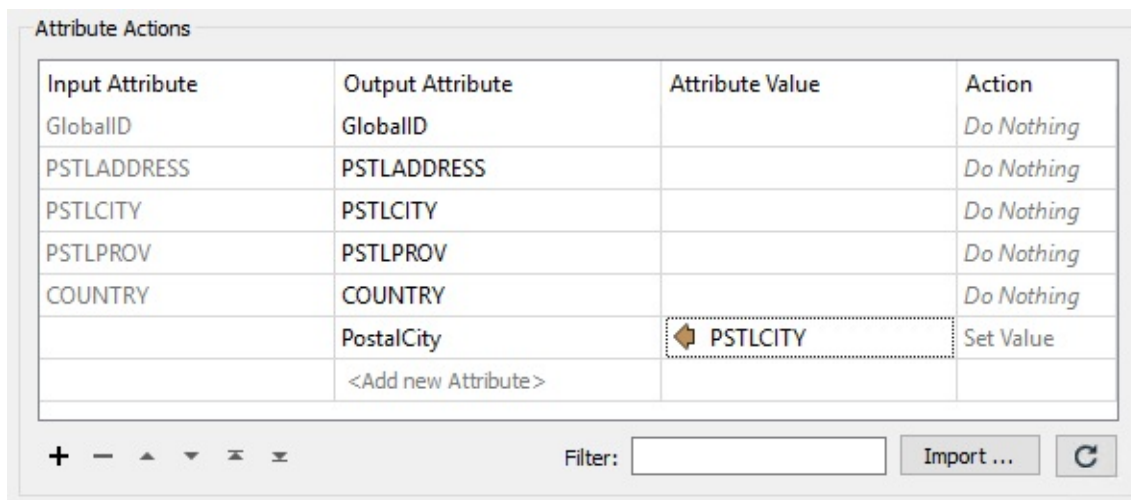
Here the AttributeCopier consists of selecting the existing attribute and entering a new name for it. Again, when connected to a Writer feature type, its schema is available to select from.

Note how both PSTLCITY and PostalCity exist on the output of the transformer, proving that it is copying the attribute rather than renaming it.


TIP

You can (as above) enter a constant attribute value in the AttributeCopier, but in reality it's hardly a copy operation in that case; it's more an attribute creation task.

For other transformers, the setup style is reversed: a new attribute is created and given the value of an existing attribute:



Input Attribute	Output Attribute	Attribute Value	Action
GlobalID	GlobalID		Do Nothing
PSTLADDRESS	PSTLADDRESS		Do Nothing
PSTLCITY	PSTLCITY		Do Nothing
PSTLPROV	PSTLPROV		Do Nothing
COUNTRY	COUNTRY		Do Nothing
	PostalCity	PSTLCITY	Set Value
	<Add new Attribute>		

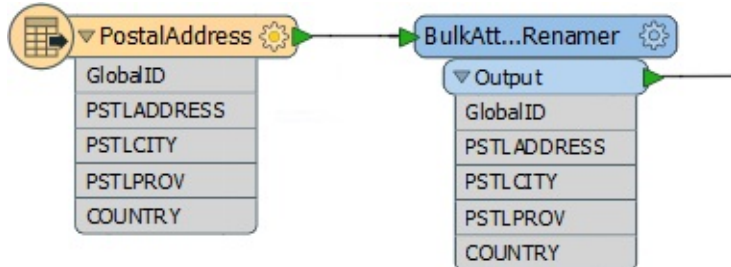
Filter: Import ... 

In this AttributeManager transformer the user creates a new attribute (PostalCity) and assigns it the value from another (PSTLCITY). In effect they have made a copy of the original attribute.

Bulk Attribute Renaming

Usual attribute renaming involves selecting individual attributes to operate on. However, in some cases it's important to be able to carry out the same renaming operation on a large number of attributes.

This scenario is catered for by the BulkAttributeRenamer transformer.



BulkAttributeRenamer

The BulkAttributeRenamer carries out the core function of renaming attributes. But instead of manually specifying each attribute, this transformer lets the user select multiple attributes - or all of them.

When multiple attributes are selected, the action must - of course - carry out the same renaming action on them all. These actions are:

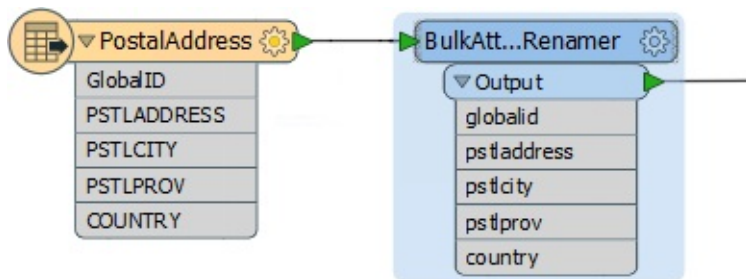
- Add String Prefix
- Add String Suffix
- Remove Prefix String
- Remove Suffix String

- Regular Expression Replace
- String Replace
- Change Case

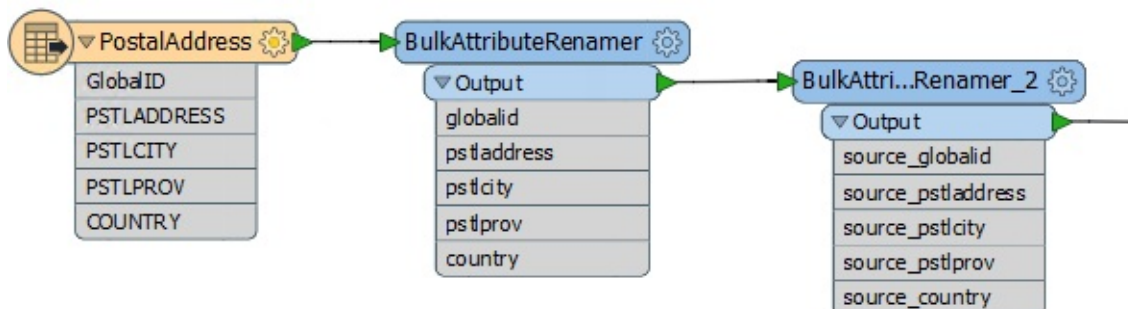
Parameters

Action:	Add String Prefix	▼
Mode:	Add String Prefix	▼
Text To Find:	Add String Suffix	▼
Case Sensitive:	Remove Prefix String	▼
String:	Remove Suffix String	▼
	Regular Expression Replace	▼
	String Replace	▼
	Change Case	▼
Case Change Type:		▼

The power of the transformer is also in its ability to manipulate multiple attributes at once, without having to individually select them all. Here, for example, the incoming attributes are all being renamed to lower case names in order to match a Writer schema that does not support upper case:



Here, for example, the user has converted to lower case and then used a second transformer to add a prefix:



Removing Attributes

Removing attributes is perhaps seen as a less important task in FME. That's because - for a manual attribute schema - only attributes defined in the writer are written to the output; extra attributes that are not required are just ignored.

However, removing attributes does carry useful benefits:

- Removing attributes that aren't required tidies up a workspace and makes it easier to understand
- A workspace is a complex network of objects and schemas. Removing attributes simplifies this network and makes the Workbench interface more responsive
- All data processing incurs costs of time and memory. Removing attributes means less data is being processed and so the FME engine performs faster

TIP

A reader feature type can be used to hide attributes from the schema. This helps tidy a workspace, but does not help improve the translation performance. However, some formats also (in 2017 onwards) have a setting for "Attributes to Read" and using this will help performance.

Transformers that can remove attributes are:

- AttributeKeeper
 - AttributeManager
 - AttributeRemover
 - BulkAttributeRemover
-

Removing Attributes

The AttributeManager and AttributeRemover have the same technique; select an attribute to be removed:

Input Attribute	Output Attribute	Attribute Value	Action
ParkId	ParkId		Do Nothing
ParkName	ParkName		Do Nothing
NeighborhoodName	NeighborhoodName	<input type="checkbox"/> Kitsilano	Set Value
DogPark	DogPark		Do Nothing
	City	<input type="checkbox"/> Vancouver	Set Value
RefParkId	RefParkId		Do Nothing
EWStreet	EWStreet		Remove
NSSStreet	NSSStreet		Remove
Washrooms	Washrooms		Remove
SpecialFeatures	SpecialFeatures		Do Nothing
	<Add new Attribute>		

+ - ▲ ▼ ↕
 Filter: Import ...

Attributes can be removed in the AttributeManager by selecting it and clicking the - button. Alternatively you can change the action field from *Do Nothing* to Remove.

Notice in the above that three attributes have been removed. The output attribute (when selected) shows the name struck out to signify that it is no longer present.

Keeping Attributes

The AttributeKeeper transformer carries out the same function, but approaches it from the opposite direction. It lets the user specify which attributes are **not** to be removed; in other words, this transformer lets the user specify which ones to keep.

So, the AttributeManager should be used where one or two attributes are to be removed, but the majority of them kept. The AttributeKeeper should be used when the majority of attributes are to be removed, and only one or two of them kept.

Bulk Attribute Removal

The BulkAttributeRemover - like the BulkAttributeRenamer - lets the user carry out a process on multiple attributes. In this case, instead of being able to select all attributes, the user enters a string-matching expression in order to define which attributes to remove:

The diagram illustrates the BulkAttributeRemover transformer in a data flow. On the left, a 'Parks' source outputs attributes: ParkId, RefParkId, ParkName, NeighborhoodName, EWStreet, NSSStreet, DogPark, Washrooms, and SpecialFeatures. These are connected to a 'BulkAttributeRemover' transformer. The transformer's output list shows that 'EWStreet', 'NSSStreet', and 'Washrooms' have been removed (indicated by a strikethrough). To the right, the 'BulkAttributeRemover Parameters' window is shown. It has a 'Transformer Name' field set to 'BulkAttributeRemover'. In the 'Parameters' section, the 'Expression to Remove' field contains 'Street\$', with a red arrow pointing to it. The window also includes 'Help', 'Defaults', 'OK', and 'Cancel' buttons.

Here the user is removing all attribute whose name ends in the word "Street".

Exercise 1 Noise Control Laws Project (Addresses)	
Data	Addresses (File Geodatabase)
Overall Goal	Convert a File Geodatabase to Microsoft Excel and map the schema
Demonstrates	Attribute Management for Schema Mapping
Start Workspace	None
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex1-Complete.fmw

City councillors have voted to amend noise control laws and local residents living in affected areas must be informed of these changes.

You have been recommended by your manager to take on the task of finding all affected addresses. There's a tight deadline and at least three city councillors are standing watching you work. The pressure is on and it's up to you to deliver!

This exercise is the first part of the project. You know that the address database for the city is stored in an Esri Geodatabase whose schema matches the Local Government Information Model PostalAddress table.

However, you are told that the software used to carry out automated bulk mailings requires addresses stored in an Excel spreadsheet using a completely different schema.

So, your first task is to create a workspace that converts addresses from Geodatabase to Excel, mapping the schema at the same time.

1) Inspect Data

As usual, the first task is to familiarize yourself with the data. To do this open the following dataset within the FME Data Inspector:

Reader Format	Esri Geodatabase (File Geodb API)
Reader Dataset	C:\FMEDData2017\Data\Addresses\Addresses.gdb

.1 UPDATE

In FME2017.1 the format is now called Esri Geodatabase (File Geodb Open API)

The table that is to be translated is called "PostalAddress." The important thing here is not how the data looks in the graphic display, but more what attributes exist in the Table View window.

Table View

Table: Addresses [FILEGDB] - PostalAddress Columns...

	OBJECTID	GlobalID	OWNERNM1	OWNERNM2	PSTLADDRESS	PSTLCITY	PSTLPROV
1	1	{1C55C207-5A3...	Jake Warnock	<null>	1188 W Pender St	Vancouver	British Columbia
2	2	{8AFE5C37-E0A...	Armand Augustyn	<null>	1661 Ontario St	Vancouver	British Columbia
3	3	{6963B7DB-653...	Lieselotte Cota	<null>	535 Smithe St	Vancouver	British Columbia
4	4	{67133025-D2F6...	Lieselotte Cota	<null>	181 W 1st Av	Vancouver	British Columbia
5	5	{3E5D9415-BDD...	Ernest Ahlgren	<null>	141 W 1st Av	Vancouver	British Columbia
6	6	{E191FAAD-295...	Jim Baskerville	<null>	808 Gore Av	Vancouver	British Columbia
7	7	{9E7BDFEB-228...	Cassandra Bran...	<null>	266 E 15th Av	Vancouver	British Columbia
8	8	{9B11F75F-3443...	Caryl Chinn	<null>	36 Water St	Vancouver	British Columbia

Q in any column 13597 row(s)

Optionally you may wish to locate the file

C:\FMEData2017\Resources\DesktopBasic\AddressSchema.xsd and open it in a text editor (or other XML file viewer). An xsd file describes the schema of an XML/GML dataset, and this file has been created to define what attributes we wish to have in our address Excel dataset.

TIP

To be clear, in this exercise we're going to write to an Excel dataset, setting it up by importing attribute definitions stored in an XML schema document.

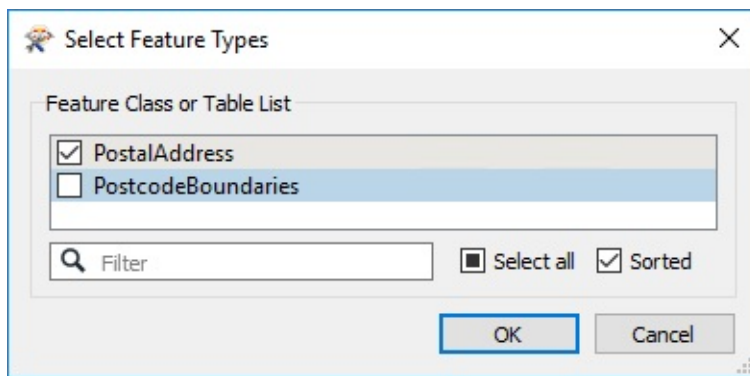
2) Create Workspace

Now that you are familiar with both the source data and the required output schema, start FME Workbench and begin with an empty workspace. Select Readers > Add Reader from the menubar.

Add the Reader with the same values as in the Data Inspector.

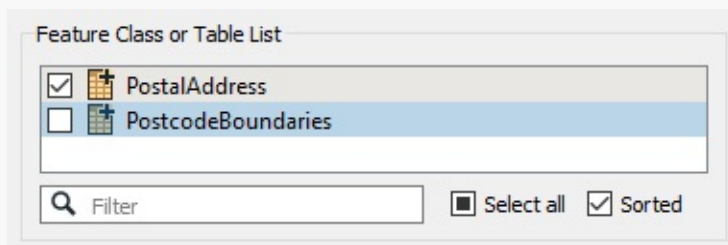
Reader Format	Esri Geodatabase (File Geodb API)
Reader Dataset	C:\FMEData2017\Data\Addresses\Addresses.gdb

When prompted, select only the PostalAddresses table, and no others:



.1 UPDATE

In FME2017.1 the dialog now has some additional icons:

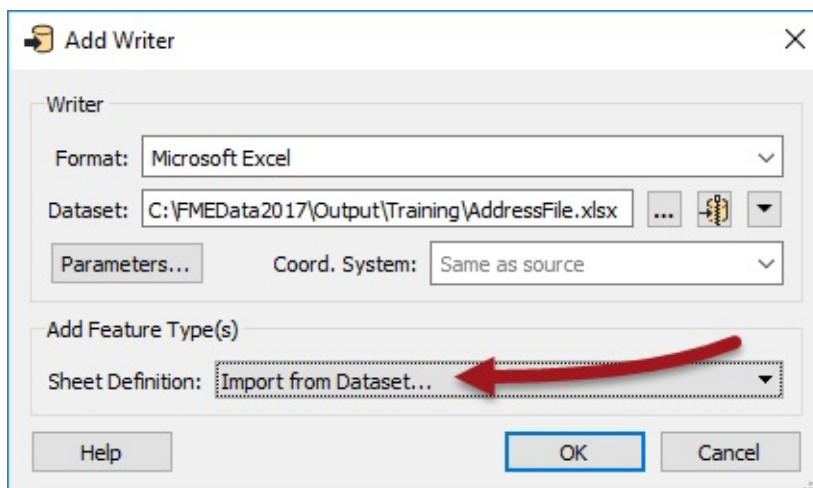


3) Add Writer

Now let's add a Writer to write the output data. Select Writers > Add Writer from the menubar and use the following:

Writer Format	Microsoft Excel
Writer Dataset	C:\FMEData2017\Output\Training\AddressFile.xlsx

Select 'Import from Dataset' in the 'Add Feature Types' section of the dialog:



Then Click OK to add the Writer.

4) Import Feature Types

At this point you are prompted to select the dataset to import a schema definition from. The Dataset can be left as it is or emptied (left blank). It is not the important part:

Reader Format	GML (Geography Markup Language)
Reader Dataset	
Reader Parameters	Application Schema C:\FMEDData2017\Resources\DesktopBasic\AddressSchema.xsd

Application Schema

Ignore Application Schema: No

Application Schema: C:\FMEDData2017\Resources\DesktopBasic\AddressSchema.xsd

Validate GML Dataset File: No

Obtain feature types, fields and data types from: XSD Schema

Ignore xsi:schemaLocation in Dataset: No

Map FeatureCollection: No items selected.

GML Feature Elements:

Click OK to accept the values. The new feature type will be created to match the chosen GML application schema.

The workspace will now look like this:



There are three GML attributes imported by default. These aren't needed, so to clean up the schema open the feature type User Attributes tab and delete them.

While you are in the same dialog, also change the data type of the UpdateDate field from *string* to *datetime*.

5) Add AttributeManager

OK, we now have the reader and writer in place. Now we can start to map the schema from reader to writer. As you'll have noticed, the two do not currently match up very well.

So, place an AttributeManager connected between the two feature types. Its parameters will look like this:

Attribute Actions			
Input Attribute	Output Attribute	Attribute Value	Action
OBJECTID	OBJECTID		<i>Do Nothing</i>
GlobalID	GlobalID		<i>Do Nothing</i>
OWNERNM1	OWNERNM1		<i>Do Nothing</i>
OWNERNM2	OWNERNM2		<i>Do Nothing</i>
PSTLADDRESS	PSTLADDRESS		<i>Do Nothing</i>
PSTLCITY	PSTLCITY		<i>Do Nothing</i>

Firstly let's clear up the Reader schema by removing some of the unwanted attributes.

Click on the following attributes and press the - button to remove them:

- OBJECTID
- GlobalID
- OWNERNM1
- OWNERNM2
- INTSTATE
- INTPSTLCD
- REPRESENT
- STATUS
- LASTUPDATE
- LASTEDITOR

Attribute Actions			
Input Attribute	Output Attribute	Attribute Value	Action
OBJECTID			Remove
GlobalID			Remove
OWNERNM1			Remove
OWNERNM2			Remove
PSTLADDRESS	PSTLADDRESS		<i>Do Nothing</i>
PSTLCITY	PSTLCITY		<i>Do Nothing</i>
PSTLPROV	PSTLPROV		<i>Do Nothing</i>

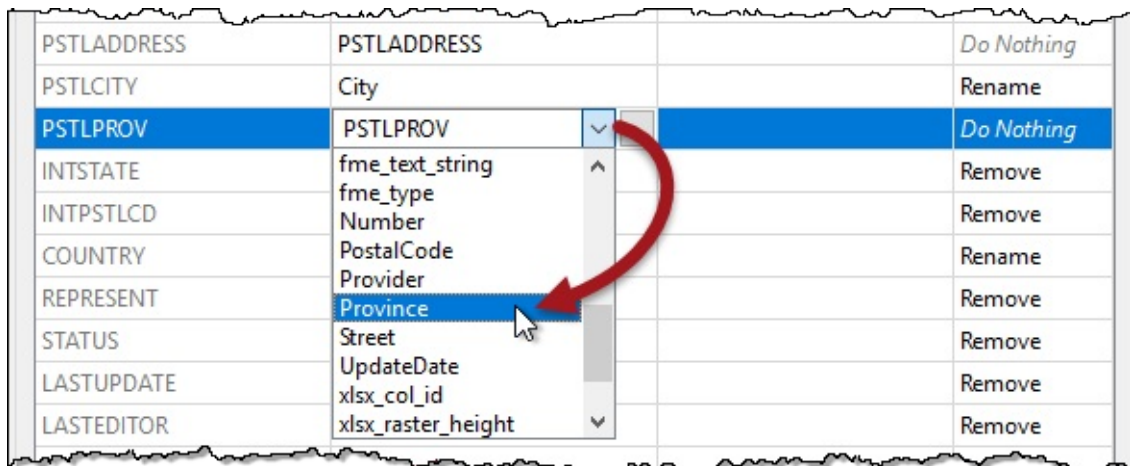
6) Rename Attributes

Several source attributes can be written to the output as they are, but do need renaming first.

In the AttributeManager rename the following:

- PSTLCITY to City
- PSTLPROV to Province
- POSTALCODE to PostalCode

- COUNTRY to Country



PSTLADDRESS	PSTLADDRESS		Do Nothing
PSTLCITY	City		Rename
PSTLPROV	PSTLPROV	▼	Do Nothing
INTSTATE	fme_text_string	▲	Remove
INTPSTLCD	fme_type		Remove
COUNTRY	PostalCode		Rename
REPRESENT	Provider		Remove
STATUS	Province		Remove
LASTUPDATE	Street		Remove
LASTUPDATE	UpdateDate		Remove
LASTUPDATE	xlsx_col_id		Remove
LASTEDITOR	xlsx_raster_height	▼	Remove

If the AttributeManager is connected to the writer feature type then you should be able to select the Output Attribute field from a drop-down list instead of typing it in.

7) Create Attribute (Provider)

Two attributes on the output (Provider and UpdateDate) are new and cannot be copied from the source data. They must be created.

In the AttributeManager create the new attribute "Provider". Because the attribute exists on the output schema, you can again select it from the drop-down list.

Set a fixed value such as your own organization name, "Safe Software", or "City of Interopolis."

8) Create Attribute (UpdateDate)

Now create the new attribute "UpdateDate". Rather than hard-coding a value click on the drop-down arrow in the Attribute Value field and choose Open Text Editor.

In the text editor locate the FME Feature Function called Timestamp and double-click to place it in the editor. By default it creates a datetime in ISO syntax, which is fine for us, so simply click OK to accept this.

Click OK again to close the AttributeManager dialog.

.1 UPDATE

In FME2017.1 the @Timestamp function has been deprecated in favour of the function @DateTimeNow, which you should use instead.

TIP

Why not run the translation now, with Redirect to FME Data Inspector turned on, to see what the result of our efforts so far is?

9) Add AttributeSplitter

Looking at the output schema there are two fields for Number and Street (for example "3305" and "W 10th Av"). However, the source schema condenses that information into one field with <space> characters separating the fields ("3305 W 10th Av"). Therefore we'll have to split the source attribute up in order to match the Writer schema.

Insert an AttributeSplitter transformer. Insert it **before** the AttributeManager - then if there are any actions to carry out on the split attributes we can use the same AttributeManager transformer.

View the AttributeSplitter parameters. Set PSTLADDRESS as the attribute to split and enter a space character into the Delimiter parameter. Ensure that a list name is set in that particular parameter.

Parameters

Attribute to Split: PSTLADDRESS

Delimiter or Format String:

Trim Whitespace: Both

List Name: _list

Drop Empty Parts: No

Click OK to close the dialog. If you run the workspace now you'll see the address as a list attribute:

```
_list{0} (encoded: utf-8) 3305
_list{1} (encoded: utf-8) W
_list{2} (encoded: utf-8) 10th
_list{3} (encoded: utf-8) Av
```

Remember a list attribute is one that can store multiple values under a single name (_list).

10) Rename Attribute

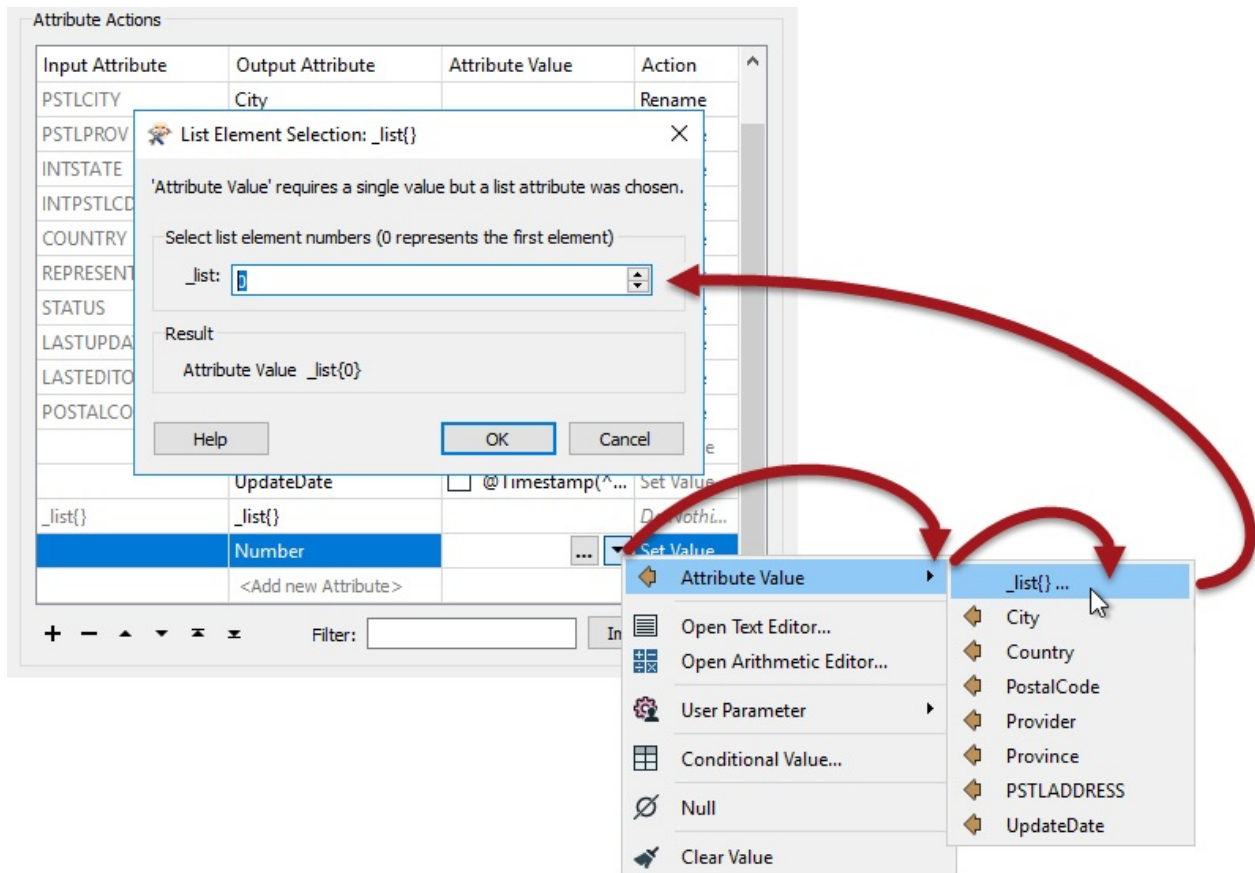
Now let's handle the Number field in the output. Go back to the AttributeManager parameters.

Notice that there is now an entry for the list attribute called _list{}. However, this is just the list attribute *"in general"* - it isn't showing each element (value) in the list.

What we need to do is create a new attribute and copy the list element we want into it. So, in the Output Attribute field create a new attribute called Number by selecting it from the drop-down list.

For the Attribute Value field click the drop-down arrow and select Attribute Value > _list{}

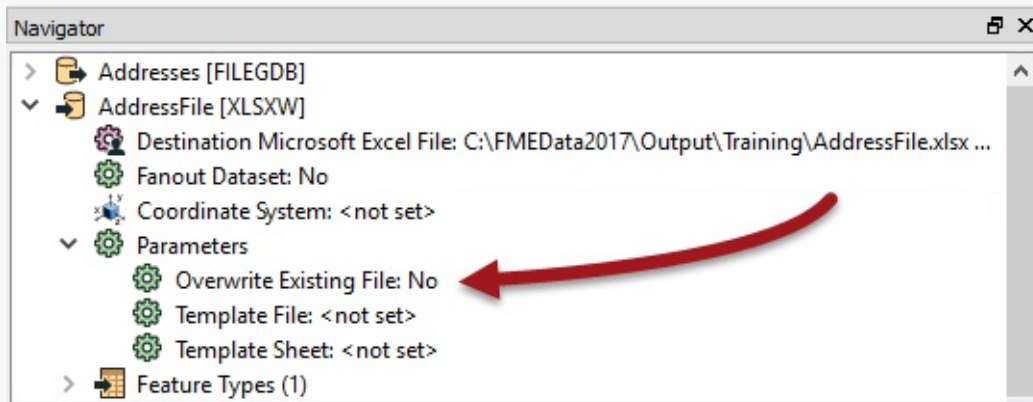
You will now be prompted to select the element in the list. Ensure it is set to zero (0) and click OK.



Click Apply/OK to confirm the changes and run the workspace to ensure the number is being copied.

WARNING

The Excel writer has a parameter called **Overwrite Existing File**, which by default is set to **No**.



This will be a problem if you are writing to the dataset rather than using the Redirect to Data Inspector setting, because each run will append data rather than overwriting.

So this is probably a good time to change the parameter to Yes, just in case you need to run the workspace more than once!

11) Construct Attribute

The final step is to recreate the Street attribute, without it being prefixed by the address number.

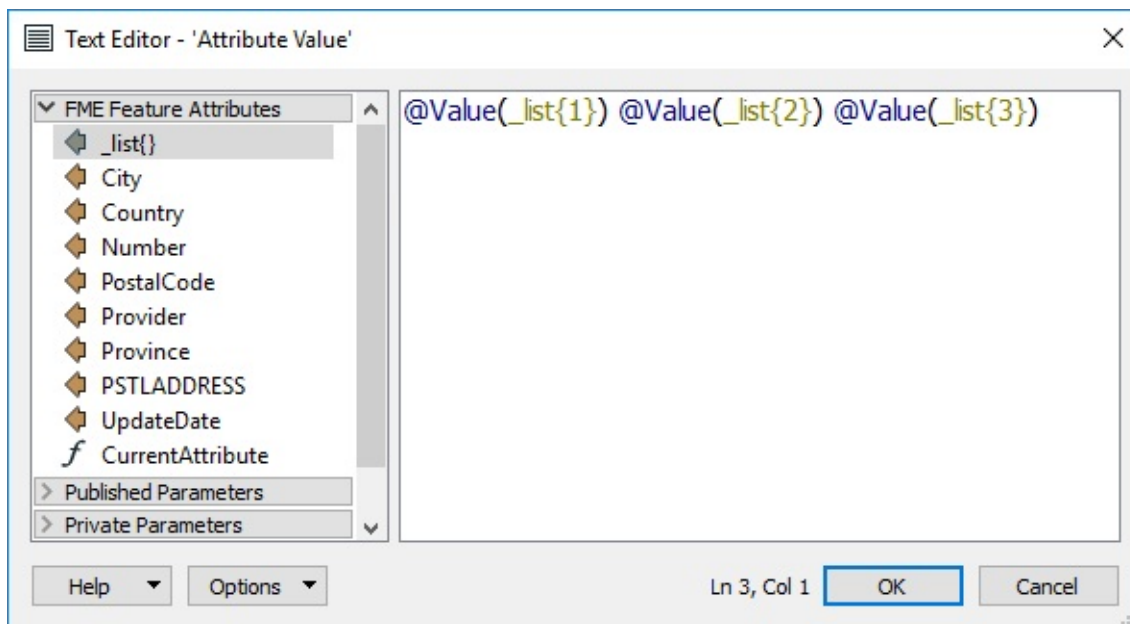
View the AttributeManager parameters again. This time in the Output Attribute field create a new attribute called Street by selecting it from the list.

Unlike the Number field, we want to create this value by concatenating several list elements. So click the drop down arrow in the Attribute Value field and choose Open Text Editor.

Locate `_list{}` in the FME Feature Attributes menu and carry out the following steps:

- Double-click `_list{}` and when prompted select element 1
- Press the spacebar to enter a `<space>` character
- Double-click `_list{}` and when prompted select element 2
- Press the spacebar to enter a `<space>` character
- Double-click `_list{}` and when prompted select element 3

The dialog will now look like this:



In this way we will have concatenated all parts of the street name back together, for example:

"W"+"17th"+"St" becomes "W 17th St"

We're assuming that no street name has more than three parts to it, but that's reasonable for our example.

12) Run Workspace

Save and then run the workspace (remember to turn off any Redirect option and ensure Overwrite Existing File is set to Yes!)

Open the containing folder to check the output has been written. Inspect the data in the FME Data Inspector. The output (in the Table View window) should look like this:

Table: AddressFile [XLSXR] - PostalAddress									
	Number	Street	City	Province	PostalCode	Country	Provider	UpdateDate	Even
1	1188	W Pender St	Vancouver	British Columbia	V6E4V5	Canada	Safe Software	20170124090044	<m
2	1661	Ontario St	Vancouver	British Columbia	V5Y2Q7	Canada	Safe Software	20170124090044	<m
3	535	Smithe St	Vancouver	British Columbia	V6B8E1	Canada	Safe Software	20170124090044	<m
4	181	W 1st Av	Vancouver	British Columbia	V5Y4W5	Canada	Safe Software	20170124090044	<m
5	141	W 1st Av	Vancouver	British Columbia	V5Y0W9	Canada	Safe Software	20170124090044	<m
6	808	Gore Av	Vancouver	British Columbia	V6A2T7	Canada	Safe Software	20170124090044	<m
7	266	E 15th Av	Vancouver	British Columbia	V5T3S6	Canada	Safe Software	20170124090044	<m
8	36	Water St	Vancouver	British Columbia	V6B8Y1	Canada	Safe Software	20170124090044	<m
9	2762	W 3rd Av	Vancouver	British Columbia	V6K6N3	Canada	Safe Software	20170124090044	<m
10	989	Main St	Vancouver	British Columbia	V6A0Z9	Canada	Safe Software	20170124090044	<m

13597 row(s)

TIP

The Event field is still empty at this point, but will be completed in a subsequent exercise.

CONGRATULATIONS

By completing this exercise you have learned how to:

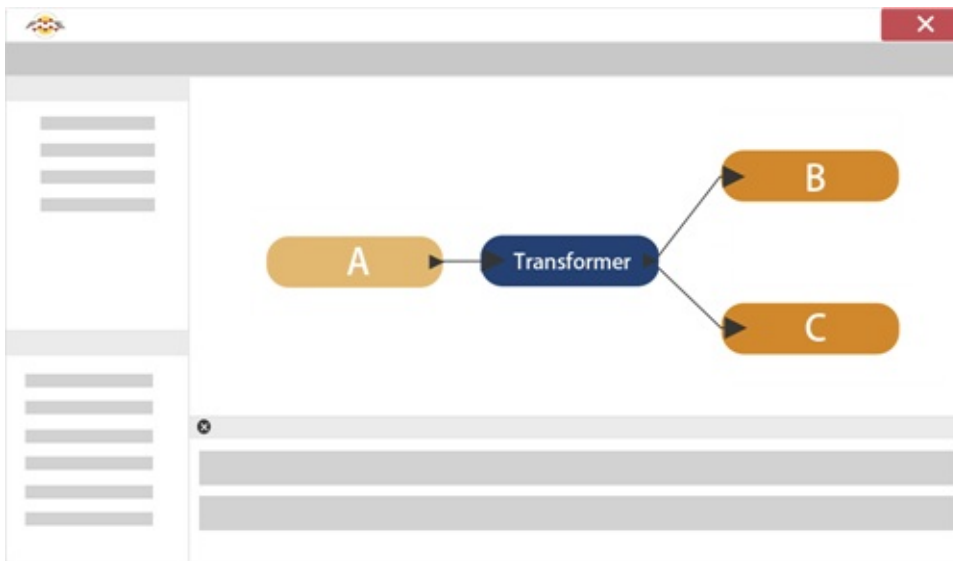
- *Use the AttributeManager transformer to create, delete, and rename attributes*
- *Use the AttributeSplitter to split attributes into a list attribute*
- *Handle list attributes in the AttributeManager*
- *Use an FME function in the AttributeManager text editor*
- *Import feature types from one format (an XML schema document) for use in another format (Excel)*

Conditional Filtering

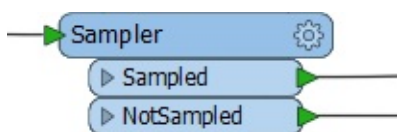
Transformers that filter don't transform data content, yet surveys show that they're the most commonly used type of transformer there is!

What is Filtering?

Filtering is the technique of subdividing data as it flows through a workspace. It's the case where there are multiple output connections from a transformer, each of which carries data to be processed in a different way. Here (for example) incoming stream A is filtered into two new streams, B and C:



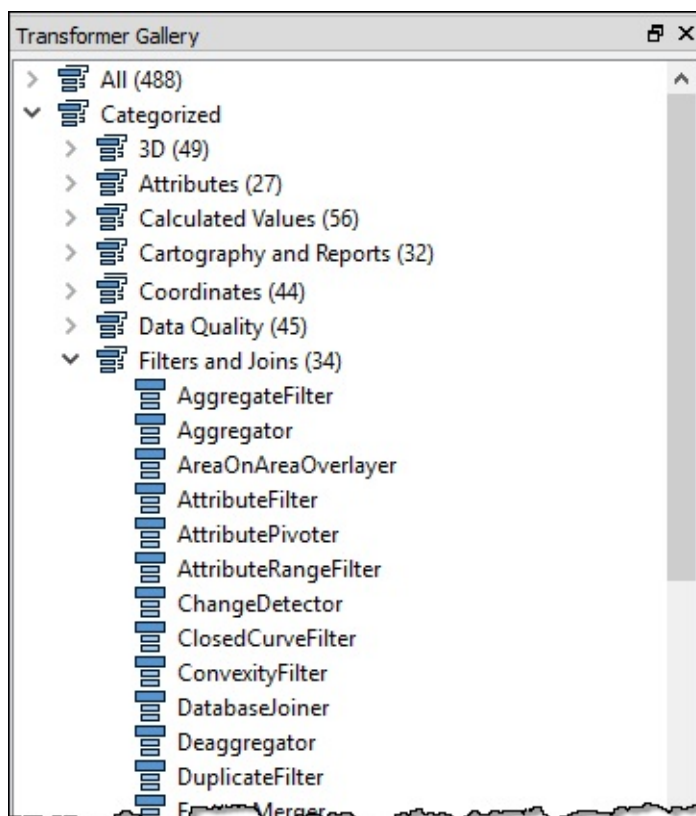
A filtering transformer may be any transformer with multiple output ports, such as the GeometryFilter or Sampler transformers, the latter of which create a sample selection of data and separates it out through Sampled and NotSampled output ports.



However, these are mostly in-built, fixed tests. Conditional filtering is where the decision about which features are output to which connection is decided by some form of *user-defined* test or condition. The Tester transformer is the most obvious example of this. It carries out a test and has different output ports for features that pass and fail the test.

Transformers that Filter

Many transformers in the Filter and Joins category carry out these tests and redirect data according to the results.



Although the Tester transformer is the most used of this category, there are many other transformers such as the TestFilter, GeometryFilter, AttributeFilter, SpatialFilter, and Sampler.

Miss Vector says...

Time for a quick question. How many filtering transformers in the Filter and Joins category appear in the top-25 Most-Valuable Transformers list?

1. Two (2)
2. Four (4)
3. Six (6)
4. Eight (8)

The Tester and TestFilter Transformers

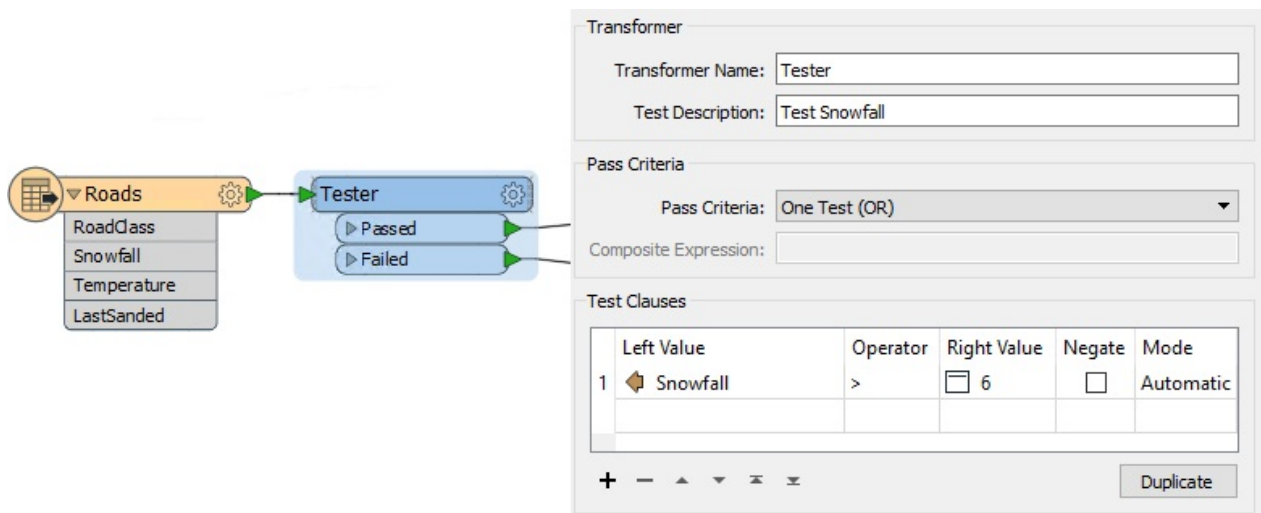
The Tester and TestFilter are the two key transformers for conditional filtering. They are for tests on attribute values.

Tester

The Tester transformer (number 1 in the top 25) is generally for single tests that produce a Yes/No result.

For example, here we wish to make a decision whether to send out snow plows (ploughs) based on a series of criteria:

- Has there been more than six inches of snowfall?
- Is this a major road?
- Is the temperature less than zero degrees Celsius?
- Was sand last applied more than 24 hours ago?



Here, features will pass if the value of their Snowfall attribute is greater than six. If it is less than (or equal to) six, the feature will fail the test.

The Tester also allows the combination of multiple tests, where a user can combine any number of clauses using an AND and OR statement. For example, if I was trying to determine whether to send a snow plough to a particular road I might ask:

- Has there been more than six inches of snowfall AND is the temperature less than zero degrees Celsius?
- Is this a major road AND (has there been more than six inches of snow OR (is the temperature less than zero AND was sand last applied more than 24 hours ago))?

Pass Criteria

Pass Criteria: Composite Test

Composite Expression: 2 AND (1 OR (3 AND 4))

Test Clauses

	Left Value	Operator	Right Value	Negate	Mode
1	Snowfall	>	6	<input type="checkbox"/>	Automatic
2	RoadClass	=	Major	<input type="checkbox"/>	Automatic
3	Temperature	<	0	<input type="checkbox"/>	Automatic
4	LastSanded	>	24	<input type="checkbox"/>	Automatic

+ - < > & ¬

Duplicate

When I have multiple tests I control them using the Pass Criteria field. A mix of AND/OR clauses requires a Composite Test, as shown above. But - however complex the test becomes - it still results in a single Yes/No result; features will either pass or fail this set of tests.

Notice we aren't restricted to simple tests of equality ($A=B$); in the above there are also "greater than" and "less than" tests. That's because there are many different operators available for use in a test clause.

Operators

The list of operators available in the Tester transformer (or in many of the other locations that make use of the Tester dialog) looks like this:

Operator	Right Value
>	6
=	
!=	
<	
>	
<=	
>=	
In	
In range	
Like	
Contains Regex	
Contains	
Begins With	
Ends With	
Type is	
Encodable in	
Attribute has a value	
Attribute Is Null	
Attribute Is Empty String	
Attribute Is Missing	

Besides the usual operators, there are also some based on a SQL where clause. These include:

- In

- Between
- Like
- Contains Regex
- Contains
- Begins With
- Ends With

...plus there are other tests that check for the existence of attributes and values:

- Attribute has a value
 - Attribute is Null
 - Attribute is Empty String
 - Attribute is Missing
-

TIP

"Attribute has a value" is the opposite of the three other tests; i.e. this attribute is not Null, AND it is not an empty string, AND it is not missing. Incidentally, "missing" means the attribute does not exist at all on the feature being tested.

TIP

"Contains Regex" means only part of the string needs to match. For example...

*Attribute Value: abcd
Search String: ^ab
Contains Regex: Passed*

i.e. the entire string doesn't need to match.

TestFilter

The TestFilter (#8 in the top 25) is essentially a way to combine multiple Tester transformers into one. Instead of the Tester's single Passed and Failed ports, you can create a passed port for each condition (it does not need to be called "Passed") with failed features going on to the next test. There is also an output port for features that fail all of the test conditions.

The TestFilter is very similar to the CASE or SWITCH command in programming or scripting languages.

Sister Intuitive says...

The TestFilter is very good for filtering a feature by a set of cascading conditions, for example here are a set of tests to again determine whether to send out a snow plough:

- *Has there been more than six inches of snowfall?*
- *Has there been more than four inches of snowfall AND is this a major road?*
- *Is the temperature less than zero degrees Celsius AND was sand last applied more than 24 hours ago?*

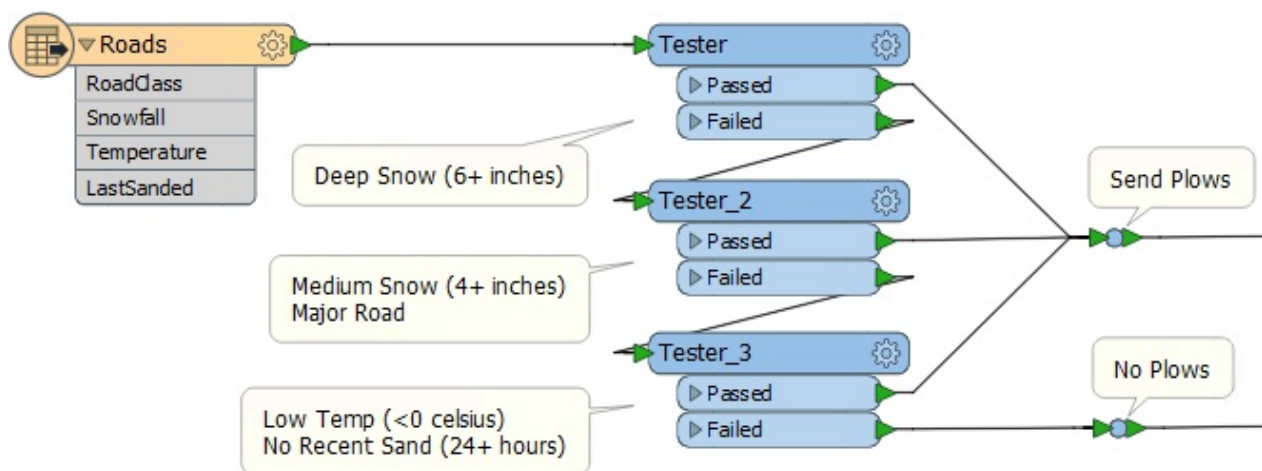
It's a set of cascading tests, because if there has been more than six inches of snow, the ploughs are sent out anyway; you don't need to test any other criteria. So the test order can be very important. If every test is a fail, then the plows are not sent out.

Also notice that you can include composite tests (those with ANDs or ORs in them).

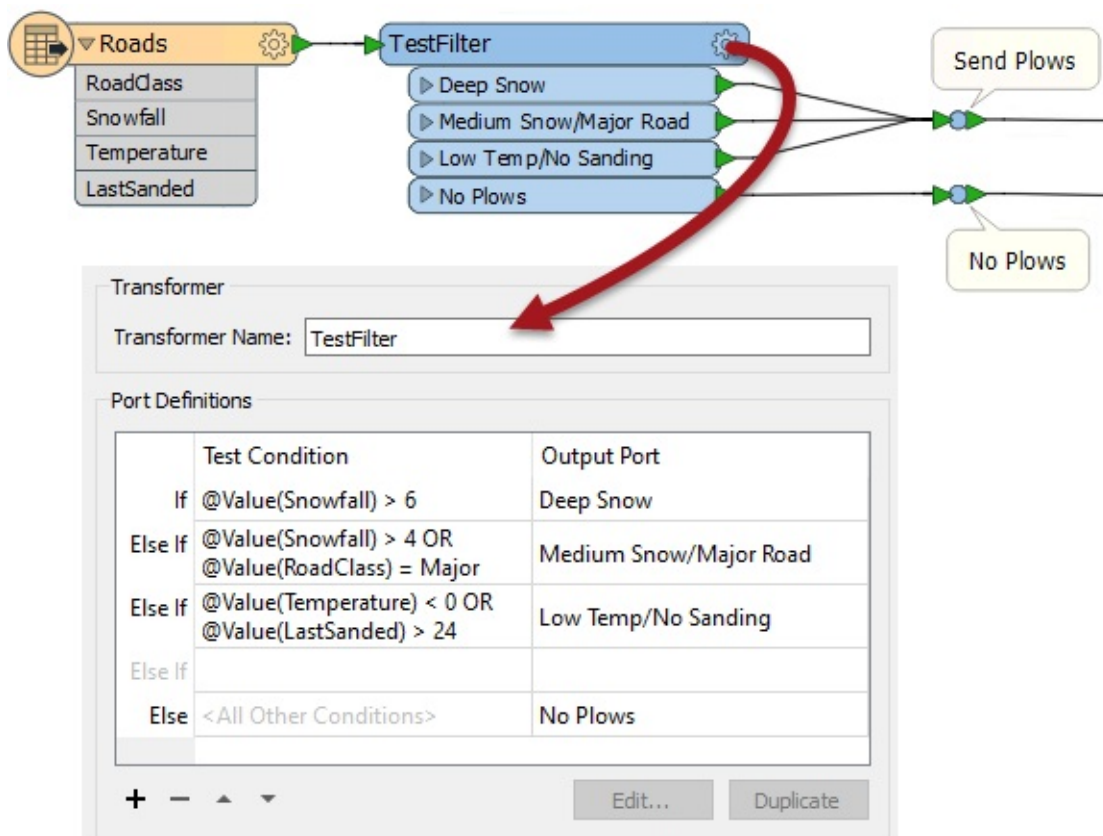
The TestFilter has the full set of operators available with the Tester such as equals, greater than, less than, and so forth. Each condition is tested in turn.

Features that pass are output through the matching output port. Features that fail are sent on to the next condition in the list. Therefore it's very important to get the conditions in the correct order.

In this example the user has used three Tester transformers and multiple connections (using the above logic) to determine whether to send the snow plows:



With the TestFilter, the three Testers are now replaced with one single transformer and there are fewer connections:



Also notice how the TestFilter output ports have custom naming. This is another advantage to this transformer.

Chef Bimm says...

Because the TestFilter can carry out a single test (as well as multiple ones) it's possible to use it exclusively instead of the Tester transformer.

Other Key Filter Transformers

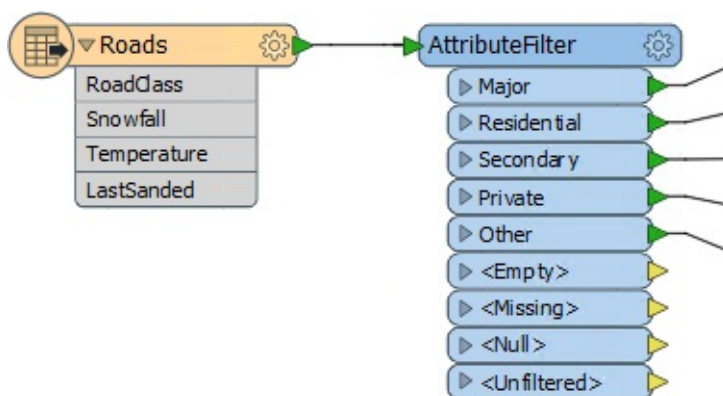
The Tester and TestFilter are not the only useful filter transformers.

AttributeFilter

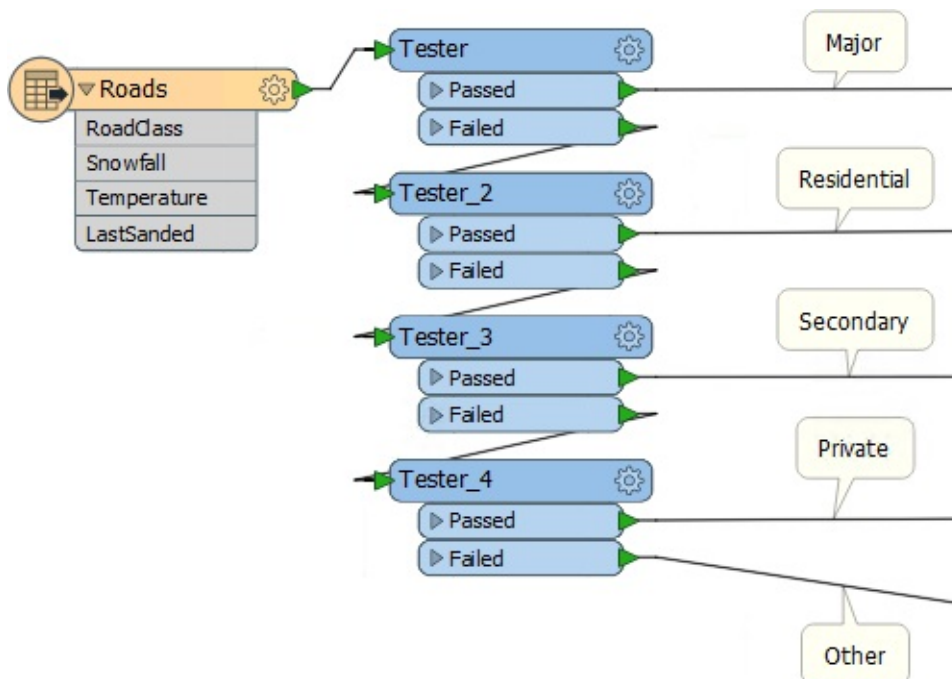
The AttributeFilter transformer (#14 in the top 25) directs features on the basis of values in a chosen attribute. It is best for testing many values for a single attribute, for example:

- Is that road a Primary, Secondary, Residential, Private, or Other type of road?
- Is the forecast for sun, rain, snow, or fog?

In this example features are divided into different streams depending on the value of a RoadClass attribute.



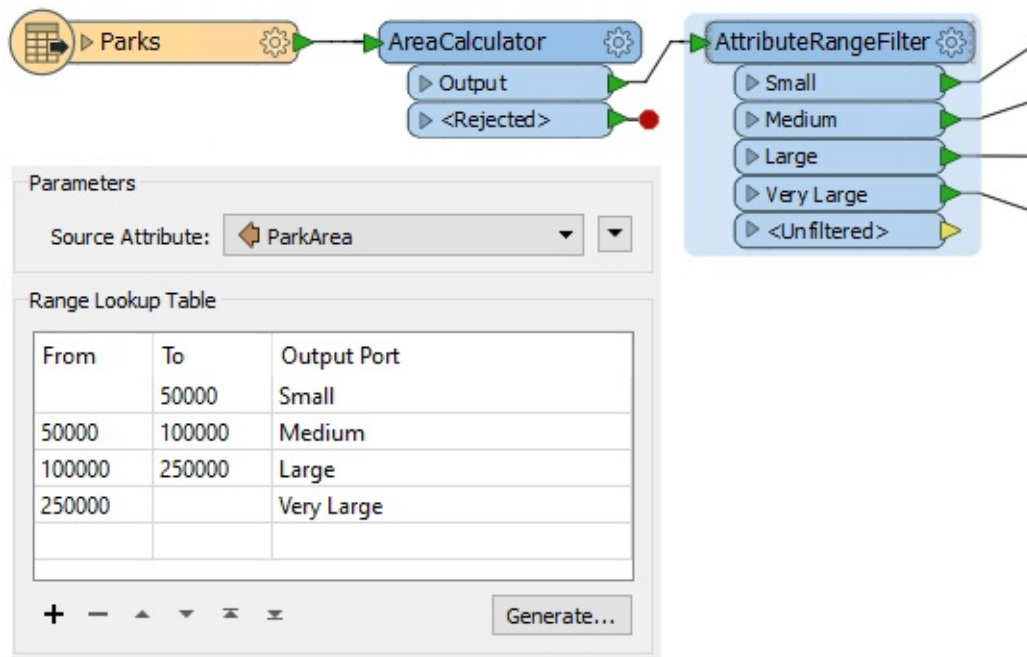
In a workspace like this the repetition of Tester transformers indicates an AttributeFilter transformer might be a better option:



The AttributeFilter's only "operator" is to find equivalency, so you would rarely use it for arithmetical tests.

AttributeRangeFilter

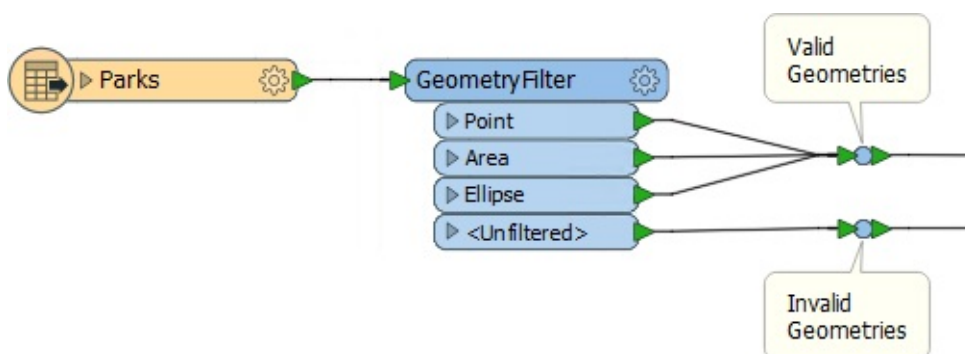
The AttributeRangeFilter carries out the same operation as the AttributeFilter, except that it can handle a range of numeric values instead of just a simple one-to-one match.



The AttributeRangeFilter parameters dialog has the option to generate ranges automatically from a set of user-defined extents.

GeometryFilter

The GeometryFilter (#22 in the top 25) directs features on the basis of geometry type; for example, point, line, area, ellipse:



The GeometryFilter is useful for:

- Filtering out unwanted geometry types; for example removing non-linear features before using an AreaBuilder transformer
- Validating geometry against a list of permitted types; for example where the dataset is constrained to either point or area features (above)
- Dividing up geometry types to write to separate destination Feature Types; for example, when writing to a geometry-restricted format such as Esri Shapefile

Confused from Interopolis says...

Dear Aunt Interop

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?

Aunt Interop says...

Dear Confused

The best solution is to check out these two articles on the Safe Software blog:

- [Conditional Processing in FME](#)
- [A Simple Guide to FME Filter Transformers](#)

There's also a useful table I put together:

	Single Test		Multiple Tests		Test Type		C
	Single Clause	Multi Clause	Single Clause	Multi Clause	String	Numeric	
Tester	Y	Y	–	–	Y	Y	
TestFilter	Y	Y	Y	Y	Y	Y	
AttributeFilter	Y	Y	–	–	Y	–	
AttributeRangeFilter	Y	Y	–	–	–	Y	

Exercise 2 Noise Control Laws Project (Spatial Filtering)	
Data	Addresses (File Geodatabase) Zoning (MapInfo TAB) Roads (AutoCAD DWG)
Overall Goal	To find all residential addresses within 50 meters of an arterial highway
Demonstrates	Methods of conditional filtering
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex2-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex2-Complete.fmw

As you know, city councillors have voted to amend noise control laws and local residents living in affected areas must be informed of these changes.

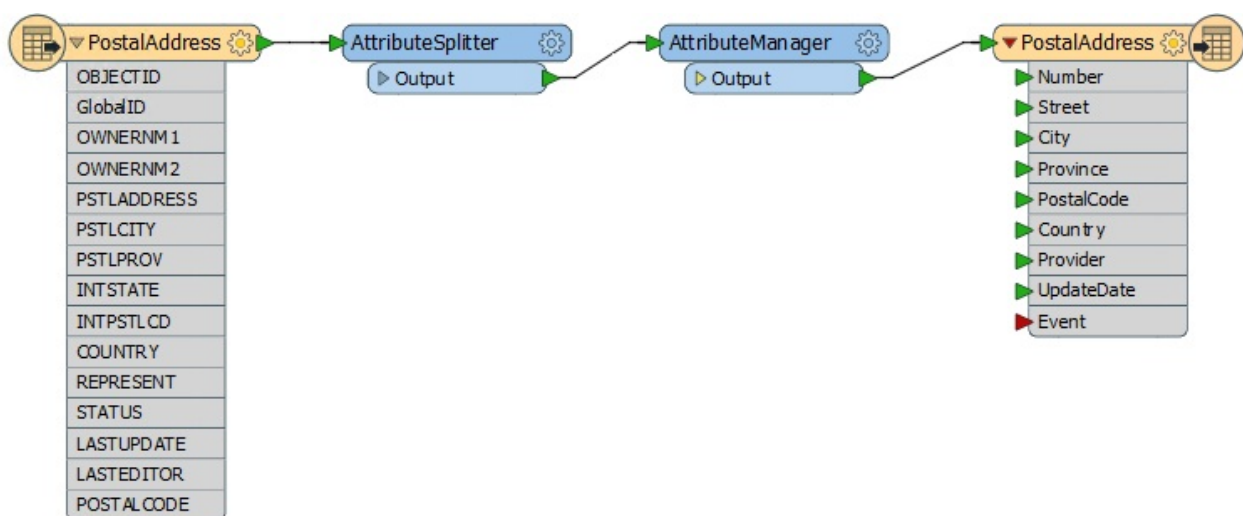
You have been recommended by your manager to take on the task and there's a tight deadline.

In the first part of the project you created a workspace to convert addresses from Geodatabase to Excel, mapping the schema at the same time.

This exercise is the second part of the project: locating all affected residents. You must locate all single-family residences within 50 metres of a major highway and filter out all others from the stream of address data.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 1. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex2-Begin.fmw



The workspace already has a reader to read addresses, transformers to edit the address schema, and a writer to write data to an Excel spreadsheet.

2) Add Reader (Roads data)

Use Readers > Add Reader to add a reader for the roads data. The roads data will be used to determine distance from an arterial route.

Reader Format	Autodesk AutoCAD DWG/DXF
Reader Dataset	C:\FMEDData2017\Data\Transportation\CompleteRoads.dwg

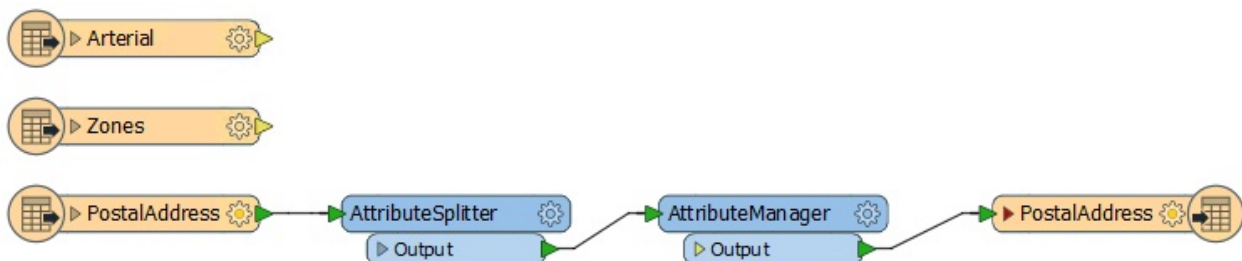
When prompted, select only the feature type (layer) called Arterial.

3) Add Reader (Zoning data)

Use Readers > Add Reader to add a reader for zoning data. The zoning data will be used to determine whether an address is single-family residential or not.

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEDData2017\Data\Zoning\Zones.tab

Ensure the feature types on the canvas are laid out in the order Arterial - Zones - PostalAddress; with attribute lists hidden the workspace will now look like this:



Feel free to inspect all of the source data to familiarize yourself with the contents.

4) Add Tester Transformer

Add a Tester transformer to the Zoning feature type.

This Tester will be used to filter residential zones from the other zoning areas. All single-family residential zones will start with RS, so the Tester should be set up like this:

Test Clauses					
	Left Value	Operator	Right Value	Negate	Mode
1	ZoneName	Begins With	RS	<input type="checkbox"/>	Automatic
<div> + - ▲ ▼ ↔ ↕ </div> <div>Duplicate</div>					

The important thing is to set up the tests using the “Begins With” predicate.

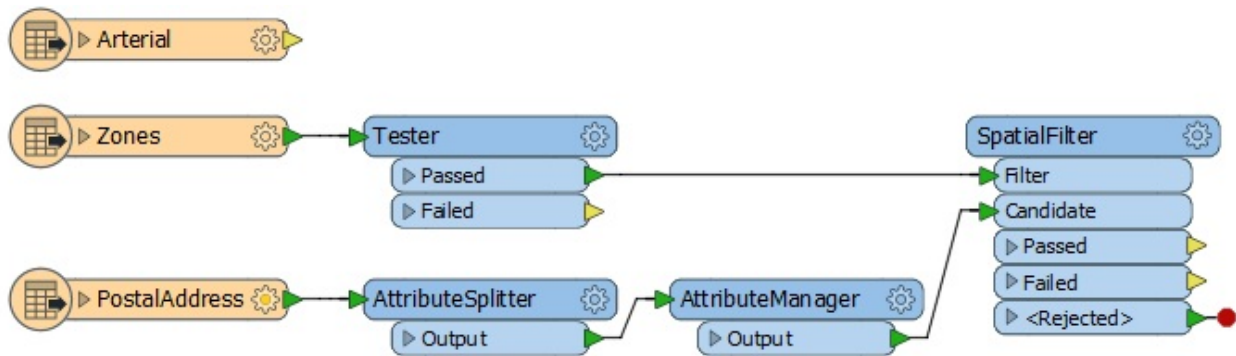
Miss Vector says...

So... why the Tester? Why not use the AttributeFilter? Do you know?

5) Add SpatialFilter

Add a SpatialFilter transformer to the workspace. This will be used to test each address to show whether that address falls inside one of the filtered residential zones.

- Connect the Tester:Passed output to the SpatialFilter:Filter port
- Connect the AttributeManager:Output port to the SpatialFilter:Candidate port



6) Set up SpatialFilter

Set up the SpatialFilter parameters as follows:

Filter Type	Multiple Filters	There are multiple zoning areas
Pass Criteria	Pass Against One Filter	A single address cannot be in all zones
Spatial Predicates to Test	Filter Contains Candidate	Find addresses that the zones contain

Tests

Filter Type: Multiple Filters

Pass Criteria: Pass Against One Filter

Support Mode: Support Aggregates

Spatial Predicates to Test: "Filter Contains Candidate"

Use Bounding Box: No

Curve Boundary Rule: Default Rule

7) Add Inspectors

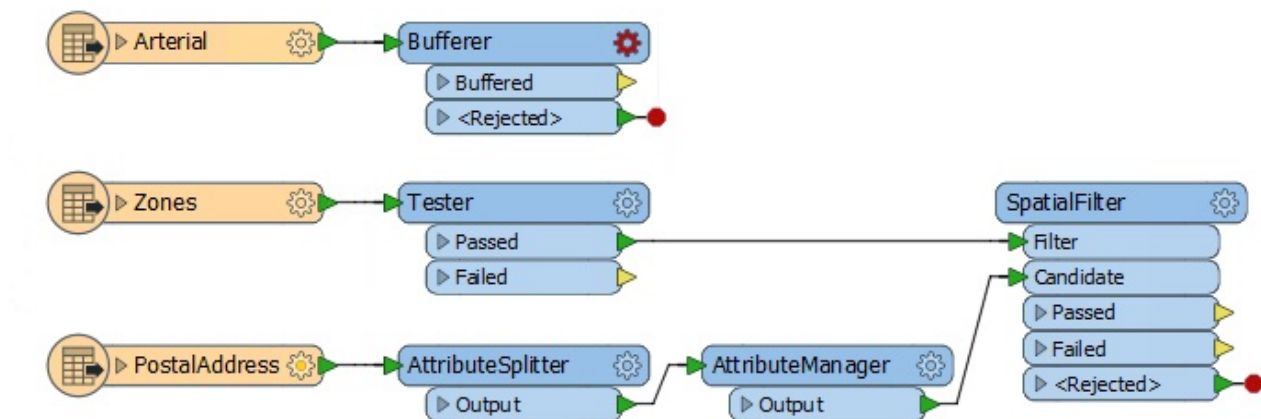
Now let's test what we have so far. Add Inspectors to the SpatialFilter Passed and Failed output ports, and the Tester:Passed port.

Save and run the workspace. Inspect the output to prove that it has worked as expected. The only area features will be the residential zones, and SpatialFilter:Passed (address) features will fall inside these areas.

8) Add Bufferer

Now we can determine which of the filtered addresses fall within 50 metres of an arterial route.

The SpatialFilter does not have a test for "within X distance" therefore we'll have to set that up a little differently. Add a Bufferer transformer to the workspace. Connect it to the Arterial roadsdata:



Set the Bufferer Buffer Amount parameter to be 50.

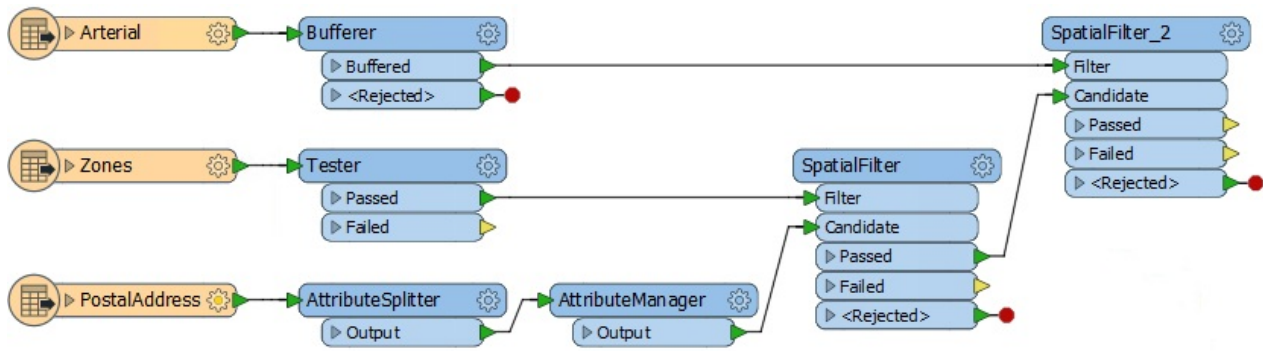
TIP

Optionally you can add a Dissolver transformer after the Bufferer, to merge all the buffer features together.

The results of the translation will be the same (in terms of addresses selected) but the data will look better in the FME Data Inspector.

9) Add SpatialFilter

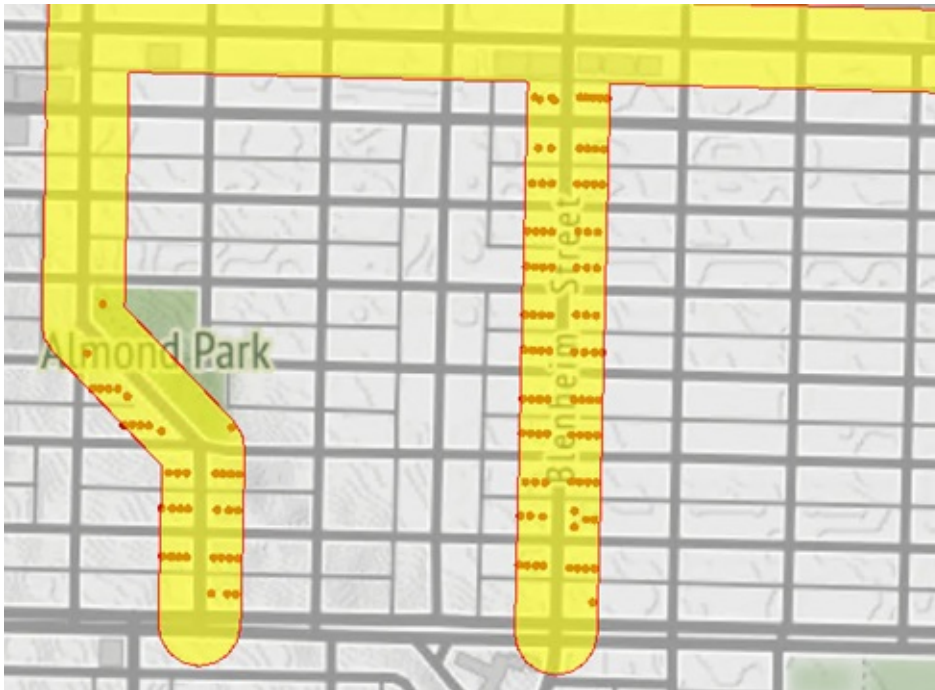
Add a second SpatialFilter transformer. The buffered arterial routes are the Filter. The prefiltered addresses are the Candidates:



As before, set the parameters to test for candidates contained by a filter, using multiple filters but only a single pass is required.

10) Run Workspace

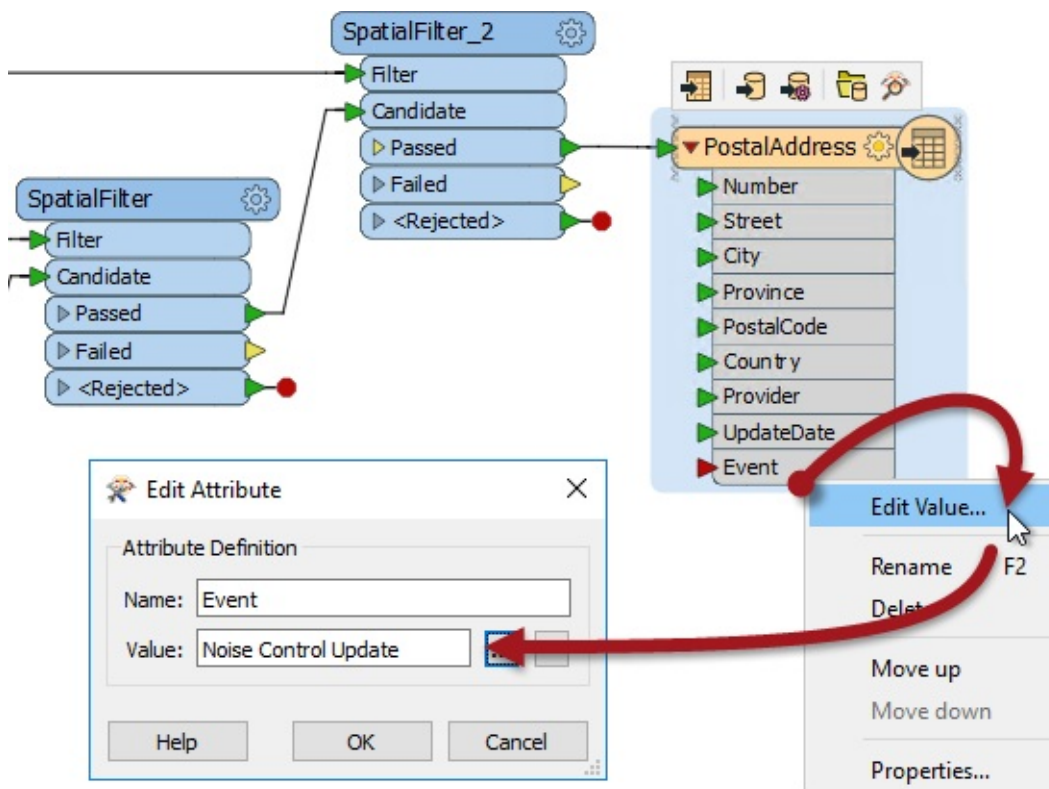
Attach some Inspector transformers to show you the output from various transformers. Save and run the workspace. The output (zoomed in) should look like this:



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

11) Connect Output

As a final step, reconnect the Excel spreadsheet output, and set a fixed value for the Event field:



Re-run the workspace and check the output to confirm the dataset has been written correctly. There should be 148 records in the spreadsheet, ready to send to the administration department for a bulk mailing.

CONGRATULATIONS

By completing this exercise you have learned how to:

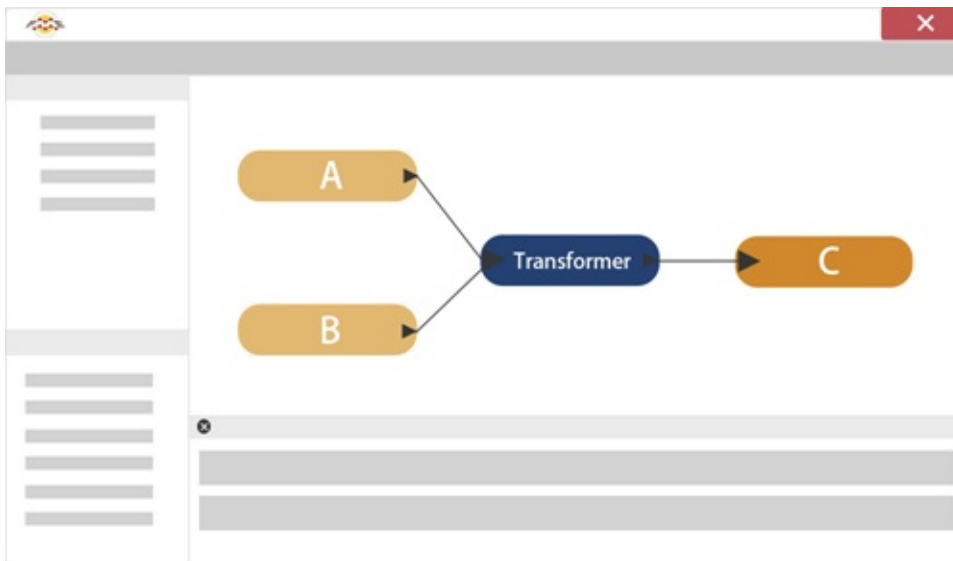
- Use the *Tester* transformer to filter by an attribute value
- Use the *SpatialFilter* transformer to filter by geometry
- Use the *Bufferer* transformer to set up a "within x distance of" test

Data Joins

Transformers that join data together are also very commonly used in FME. It's so related to filtering that the transformer category is called Filters and Joins.

What is a Data Join?

Whereas Filter transformers divide data into different streams, other transformers bring data streams together, merging the data according to a set of user-defined conditions. Here (for example) incoming streams A and B are joined together into a new stream, C:



To merge data in FME Workbench it is necessary to do more than just draw two connections into the same input port; that will only combine the data into a single stream, not fuse it together. To truly merge data it is necessary to define a relationship for the basis of the join, and this is done with one of a number of transformers.

These transformers allow you to merge not just data that is being processed by the workspace, but provide the ability to form a join against a database or other external dataset.

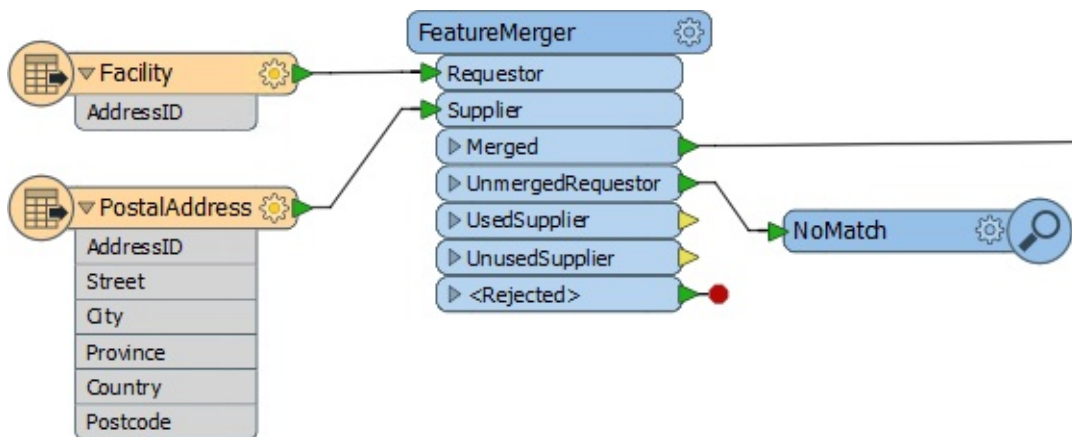
Joins in FME can either be based on matching attribute values (DatabaseJoiner or FeatureMerger), or they can be based on a spatial relationship such as an overlap between features or proximity from one feature to another (NeighborFinder or SpatialRelator).

Attribute Join Transformers

These are transformers that join data on the basis of a matching attribute value.

FeatureMerger

The FeatureMerger is the primary transformer for joining two streams of data within a workspace. This is achieved on the basis of one or more matching attribute values (keys).



Here, for example, a dataset of facilities has an AddressID number, but no address. The FeatureMerger is being used to combine data from an address table into the facilities data.

Of interest for QA reasons is the UnmergedRequestor port. Here facilities that did not have a matching AddressID are sent to a Inspector transformer to record the fact that there was no match. These records could then be checked to ensure they have a valid AddressID.

The parameters dialog for the FeatureMerger looks like this:

Join On

Requestor	Supplier	Comparison Mode
AddressID	AddressID	Automatic

+ -

Merge Parameters

Feature Merge Type: Attributes and Geometry

Process Duplicate Suppliers: No

Reject Null and Missing Keys: No

Geometry Merge Type:

Connect Z Mode:

Number of Suppliers Attribute:

☐ Attribute Accumulation

Accumulation Mode: Merge Supplier

Conflict Resolution: Use Requestor

Ignore Nulls: No

Prefix:

☐ Generate List

Notice how the AddressID attribute from each set of data is being used to merge the features together. Also notice that the merge can (and in this case does) include both attributes and geometry; i.e. the "requestor" can be updated with the geometry of the "supplier."

Attribute Accumulation parameters allow you to determine what to do if the requestor already has an attribute of the same name as one being supplied to it (for example, if the Facility records already had a Postcode field, which should take precedence?)

TIP

Another useful parameter is called Number of Suppliers Attribute. This attribute will be used to record the number of suppliers that were matched to any one requestor (assuming Process Duplicate Suppliers is turned on).

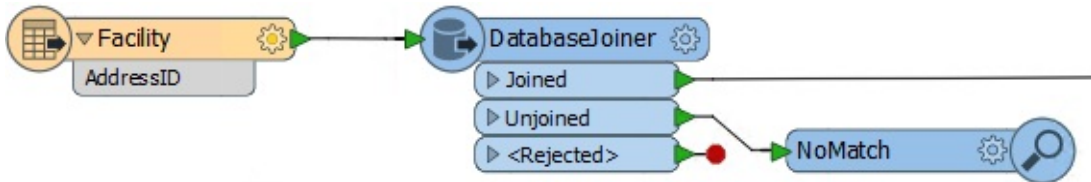
DatabaseJoiner

The DatabaseJoiner transformer is similar to the FeatureMerger, but instead of merging two streams of features, it merges one stream of features with data from an external database.

NEW

This transformer was renamed in FME2017 from Joiner to DatabaseJoiner, to better reflect what it does.

It also received a new output port for collecting unjoined features.



Here is the same example as for the FeatureMerger above. In this case the facilities features are obtaining address data directly from an address table in a PostGIS database.

The parameters dialog for the DatabaseJoiner looks like this:

Reader

Format:

Dataset:

Parameters... Coord. System:

Join

Table:

Feature Attribute	Table Field
Join On: <input type="text" value="AddressID"/>	<input type="text" value="PostalAddressId"/>

+ -

Fields to Add:

Cardinality:

Multiple Matches:

Joined List Name:

Merge Attributes

Accumulation Mode:

Conflict Resolution:

Prefix:

Again, AddressID is being used from the source dataset to facilitate a merge between the two sets of data, with PostalAddressId being the database field used.

First-Officer Transformer says...

The DatabaseJoiner has a number of advantages over the FeatureMerger. Firstly it has parameters to control how multiple matches are handled, as well as parameters for optimizing the database query.

Secondly, it allows features to be joined without having to read the entire dataset into a workspace. FME can just query the database and select the individual records it needs. This can improve performance greatly.

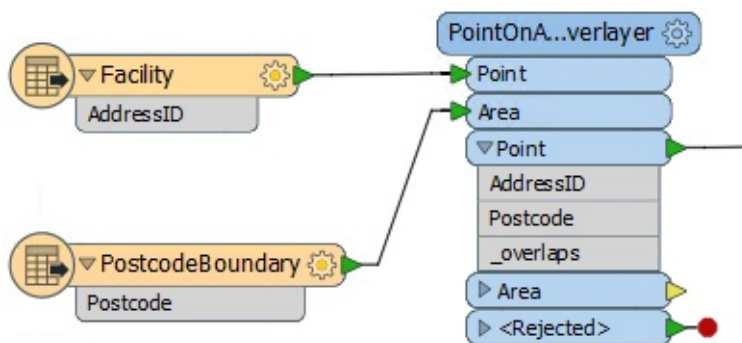
It does, of course, require the supplier records to be stored in an appropriate database format!

Spatial Join Transformers

These are transformers that join data on the basis of a spatial relationship. There are many of these in FME Workbench, but the following are some of the key ones.

Overlayers

There are a number of different "overlayer" transformers, each handling a different form of overlay. For example the PointOnAreaOverlayer carries out a spatial join on points that fall inside area (polygon) features. As the help explains, "Each point receives the attributes of the area(s) it is contained in, and each containing area receives the attributes of each point it contains."



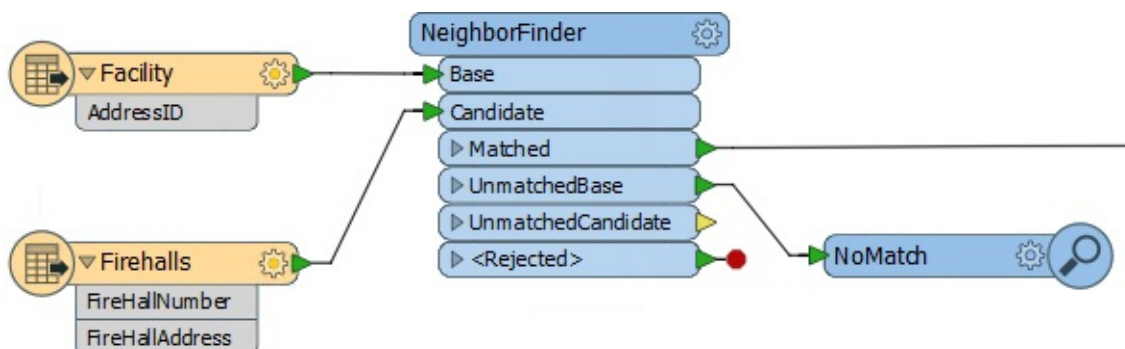
Here the facility features are being provided with a postal code depending on which postal code polygon feature they fall inside.

The "_overlaps" attribute is another useful outcome of this transformer. It tells us how many polygons each facility fell inside; in this case it would be an easy way to spot the problem of a facility falling inside more than one postal code.

Conversely, the Area output would have an "_overlaps" attribute that would tell us how many facilities fell inside each postal code.

NeighborFinder

The NeighborFinder transformer carries out a spatial join based on a proximity relationship. Here the NeighborFinder is being used to identify the closest fire hall to each facility:

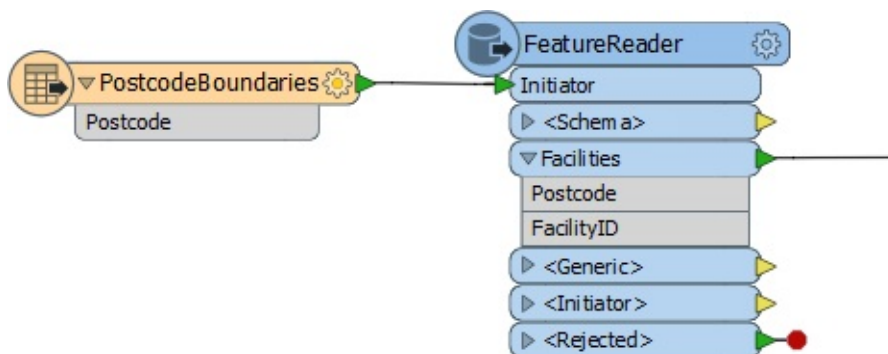


The FireHallId and FireHallAddress attributes will be merged onto each Facility feature along with a number of useful attributes recording the X/Y coordinate, direction, and distance of the closest fire hall.

The parameters of the NeighborFinder includes the ability to specify a maximum distance for the relationship, or the maximum number of neighbors to find.

FeatureReader

The FeatureReader is the spatial equivalent of the DatabaseJoiner transformer. It reads from an external dataset and forms a match based on a spatial relationship between the initiating feature and features in that dataset.



One difference is that the output is not the original feature, but the queried feature; hence the name FeatureReader.

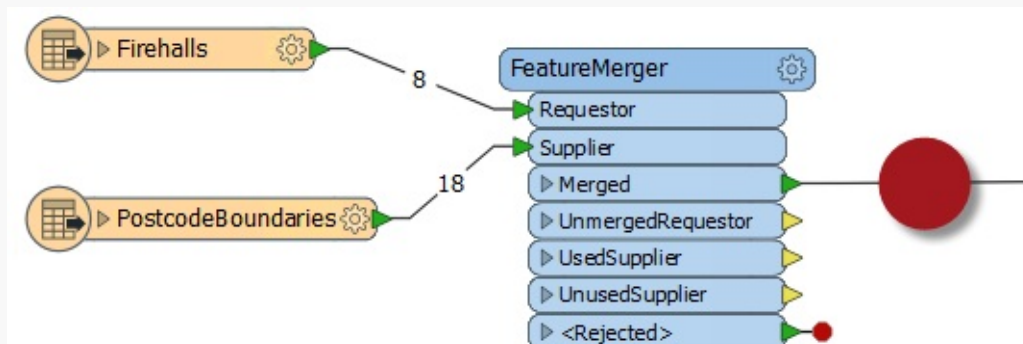
For example, here the FeatureReader is being used to carry out the same overlay of facility and postal code. The Postcode features are read into the workspace and used as a means to spatially query facilities. The facilities are retrieved with the attributes of the postcode feature they fall inside.

SpatialFilter

The SpatialFilter - as its name suggests - filters data according to a spatial relationship. However, it does also merge attributes from one feature to another, therefore can be said to be a type of Spatial Join.

Miss Vector says...

Here is a question on data joins. Look at the following screenshot, then answer how many features will appear in the output connection...



1. Eight (8)
2. Eighteen (18)
3. Twenty-six (26)
4. Can't tell

Module Review

This module was designed to introduce you to a wider range of FME transformers, plus a number of techniques for applying transformers more efficiently.

What You Should Have Learned from this Module

The following are key points to be learned from this session:

Theory

- There are distinct groups of transformers that do work other than transforming data attributes or geometry
- Integrated functionality allows the author to replace support transformers with tools built-into operational transformers
- A large proportion of the most-used transformers are related to attribute-handling
- Filtering is the act of dividing data. Conditional Filtering is the act of dividing data on the basis of a test or condition
- Data Joins are carried out by transformers that merge data together, from within Workbench or from external data sources

FME Skills

- The ability to locate a transformer to carry out a particular task, without knowing about that transformer in advance
- The ability to build strings and calculate arithmetic values using integrated tools
- The ability to use common transformers for attribute management
- The ability to use transformers for filtering and dividing data
- The ability to use transformers for merging data together

Further Reading

For further reading why not browse blog articles for particular transformers such as the [TestFilter](#), [AttributeCreator](#), or [FeatureMerger](#)?

Questions

Here are the answers to the questions in this chapter.

Miss Vector says...

Which of the following is NOT a category of transformers?

1. *Attributes*
2. **Calculations**
3. *Data Quality*
4. *Workflows*

Miss Vector says...

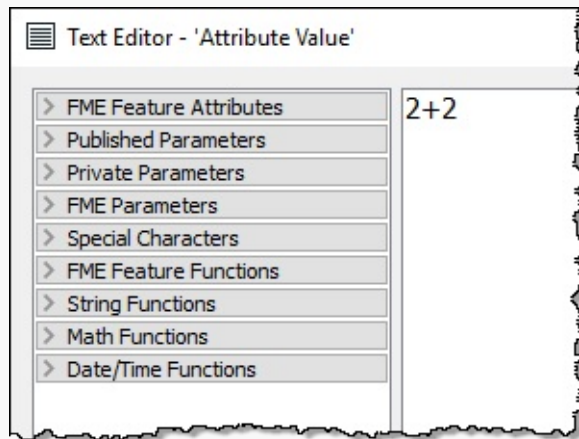
Here are four transformers and four categories. Match the transformer to the correct category.

Scenario	Tool
Chopper	Geometries
Terminator	Workflows
Matcher	Data Quality
DateFormatter	Strings

Did you look through the transformer gallery to find these? The quicker way is to look at the help page for the transformer. While we are on the subject, transformers can belong to more than one category (though not all do).

Miss Vector says...

Look at this screenshot of an editing dialog and tell me what the value returned to the attribute will be:



1. **2+2**
2. 4
3. 4.0
4. Error!

The key is to notice that the header of this dialog says "String Editor". Therefore the value returned to this attribute will be the literal string "2+2". If the user wishes to add 2+2 to get 4, they should have used the arithmetic editor!

Miss Vector says...

How many of the transformers in the Filters and Joins category appear in the top-25 Most-Valuable Transformers list?

1. Two (2)
2. Four (4)
3. Six (6)
4. **Eight (8)**

As of August 2017 (and FME 2017.1) there are eight. They are the Tester (1st), FeatureMerger (4th), TestFilter (8th), Aggregator (12th), FeatureReader (13th), AttributeFilter (14th), SpatialFilter (21st), and GeometryFilter (22nd)

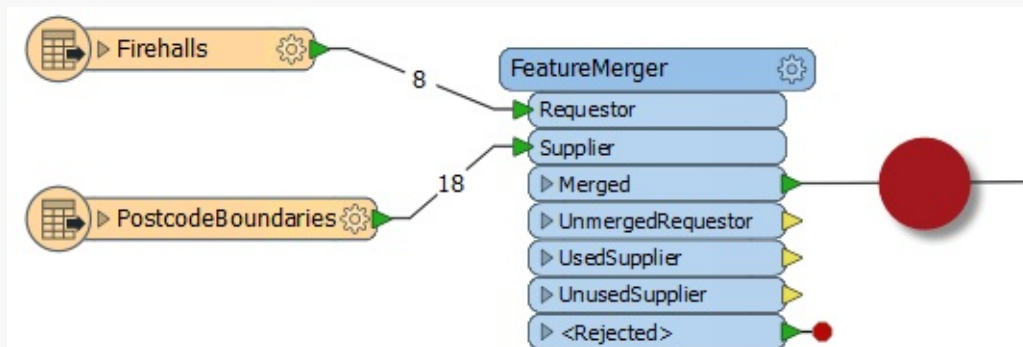
Miss Vector says...

So... why the Tester? Why not use the AttributeFilter?

Because we only need to test for one value in a simple Yes/No format. The AttributeFilter is better for testing multiple values. Also the AttributeFilter doesn't let us do "Begins With" tests.

Miss Vector says...

Look at the following screenshot, then answer how many features will appear in the output connection...



1. Eight (8)
2. Eighteen (18)
3. Twenty-six (26)
4. **Can't tell**

It's impossible to tell from the screenshot, because you don't know how many attribute values will be a match. Because there are eight firehalls it will be anywhere from zero to eight, but that's all we can tell.

Miss Vector says...

Why do we use the `StringCaseChanger` on the roads data (to change it to `UPPERCASE`) rather than on the crime data (to change it to `TitleCase`)?

Because uppercase is easier to match and has no risk of bad data. If a block was wrongly labelled W Georgia ST in the original data, a title case match would fail. An uppercase conversion would not.

Also, be sure not to confuse the `StringCaseChanger` (changes attribute values) with the `BulkAttributeRenamer` (changes attribute names)!

Exercise 3 Noise Control Laws Project (Crime Data Joins)	
Data	Crime Statistics (CSV)
Overall Goal	Carry out a join between crime statistics and address features
Demonstrates	Attribute-Based Joins
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex3-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex3-Complete.fmw

As you know, city councillors have voted to amend noise control laws and local residents living in affected areas were informed of these changes.

In the first part of the project you created a workspace to convert addresses from Geodatabase to Excel, mapping the schema at the same time. In the second part of the project you continued the workspace to locate all single-family residences within 50 metres of a major highway and filter out all others from the stream of address data.

Now a data journalist with a national newspaper is concerned that the relaxation of noise control laws may lead to more crime in the city. They have therefore made a request for recent crime figures for each of the affected addresses. They intend to compare this against future data to see if their theory is correct.

This is a major test of the city's open data policy and there's no question of not complying. However, a crisis arises as the current datasets for crime (CSV, non-spatial) is not joined to the address database in any way.

So, for the final part of this project you must take the existing noise control workspace and amend it to incorporate crime statistics.

Pull this off and you will be a spatial superhero!

1) Inspect Source Data (Crime)

The first task is to familiarize yourself with the source data. To do this open the following dataset within the FME Data Inspector or a simple text editor.

Reader Format	Comma Separated Value (CSV)
Reader Dataset	C:\FMEDData2017\Data\Emergency\Crime2011.csv
Reader Parameters	Fields:Delimiter Character: , (Comma) Fields:Field Names Line: 1

The data will look like this in the Data Inspector Table View window:

	CrimelD	Type	Month	Block
1	5	Theft From Auto Over \$5000	1	7XX W GEORGIA ST
2	9	Theft From Auto Over \$5000	1	17XX BARCLAY ST
3	10	Theft From Auto Over \$5000	2	9XX BURRARD ST
4	12	Theft From Auto Over \$5000	2	7XX DUNSMUIR ST
5	17	Theft From Auto Over \$5000	2	1XX KEEFER ST
6	18	Theft From Auto Over \$5000	2	1XX KEEFER ST
7	19	Theft From Auto Over \$5000	2	7XX W GEORGIA ST
8	20	Theft From Auto Over \$5000	2	1XX W CORDOVA ST

Notice how there is no spatial data as such, but there is a block number.

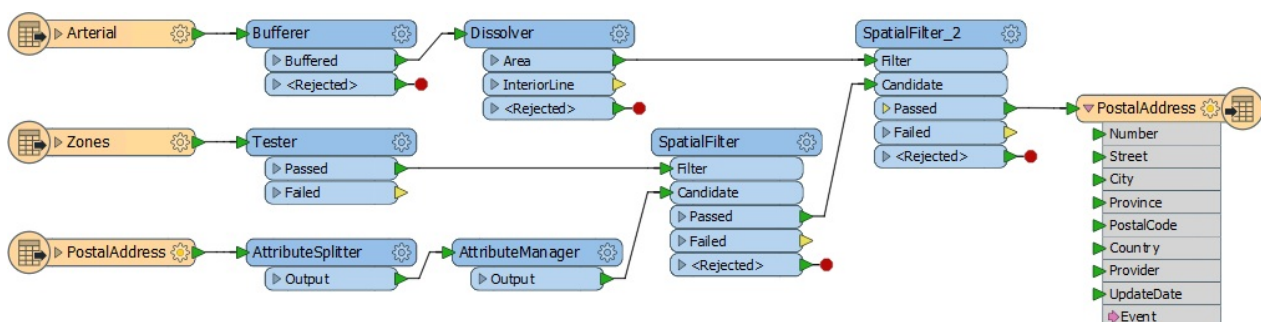
Police-Chief Webb-Mapp says...

Crime?! In my city? I think not. But if there was... be aware that 7XX W Georgia Street means the seventh block on Georgia Street west of Ontario Street and covers building numbers 700-800. 7XX E Georgia Street would be 14 blocks away, the seventh block east of Ontario. Got it?

2) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2. Alternatively you can open C:\FMEDData2017\Workspaces\DesktopBasic\Transformers-Ex3-Begin.fmw

The workspace is already set up to read addresses, filter them spatially, and write them to an Excel spreadsheet.



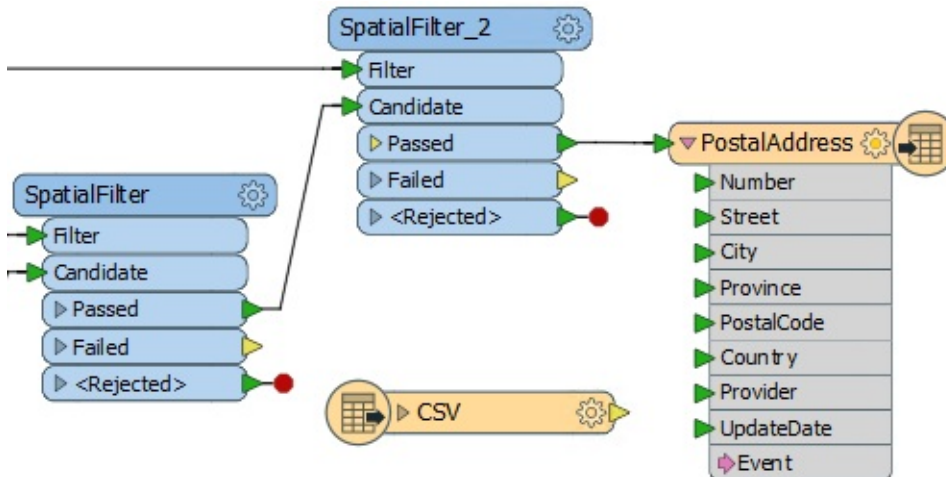
Notice that each address has a number (not a block ID like "7XX"). Another difference is that the road data is stored in Title case ("W Georgia St") in the roads dataset, whereas the crime dataset is upper case ("W GEORGIA ST").

Both of these will make it harder to join the two sets of data together.

3) Add Readers

Now let's start working with this data.

Add a reader to the workspace using Readers > Add Reader from the menubar. This reader should be used to read the crime (CSV) data. Be sure to use the same parameters as specified for the Data Inspector.



4) Add StringReplacer

To merge the data we need to reduce the address number to a block number that matches the crime data in structure; for example we need 74XX instead of 7445.

So, add a StringReplacer transformer and connect it to SpatialFilter_2:Passed port.

Set the following parameters:

Attributes	Number
Mode	Replace Regular Expression
Text to Replace	..\$
Replacement Text	XX

The text to replace (..\$) means replace the last two characters of the string, and they are replaced with XX to match the crime data.

Parameters

Attributes: ... ▼

Mode: ▼ ▼

Case Sensitive: ▼ ▼

Text To Replace: ... ▼

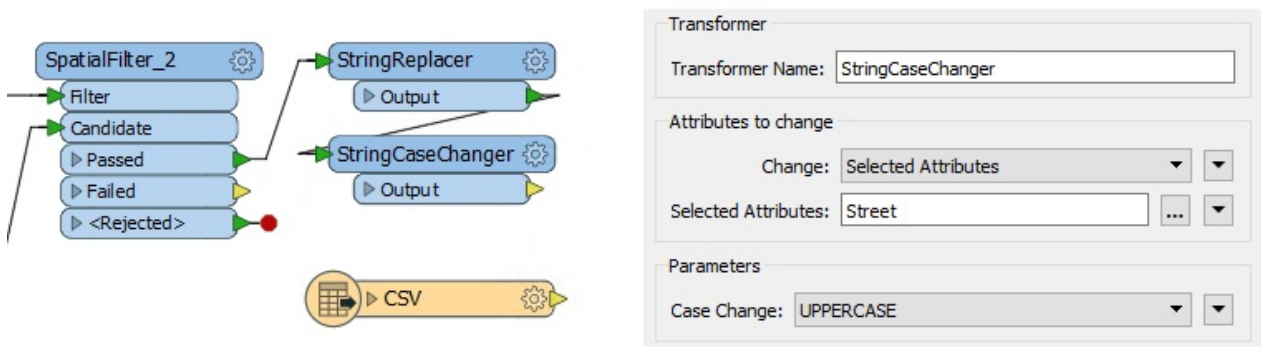
Replacement Text: ▼

This is a regular expression so be sure to set that as the mode. If you wish, attach an Inspector and run the workspace to ensure the transformer is working as expected.

5) Add StringCaseChanger

The other difference in crime/road data was in UPPER/Title case street names. This disparity can be fixed with a StringCaseChanger transformer.

Add a StringCaseChanger transformer after the StringReplacer and set the parameters to change the value of Street to upper case:



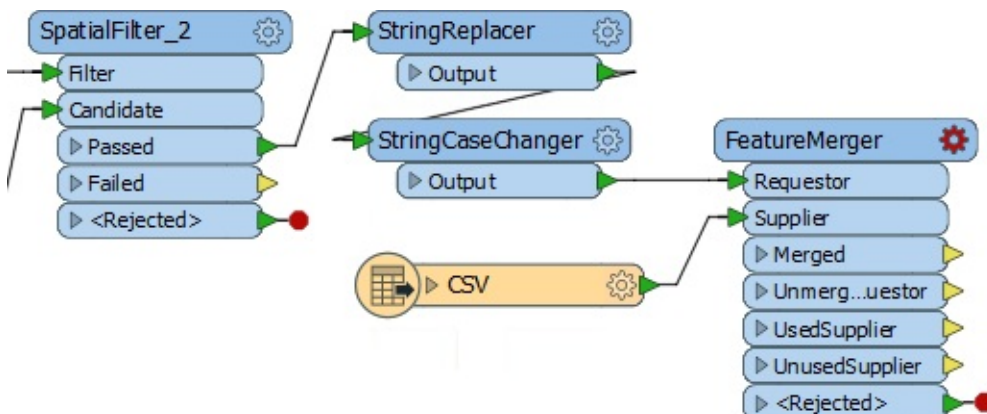
Miss Vector says...

So, answer me this. Why do we use the StringCaseChanger on the address data (to UPPERCASE) rather than changing the crime data (to TitleCase)? Do you know?

6) Add FeatureMerger

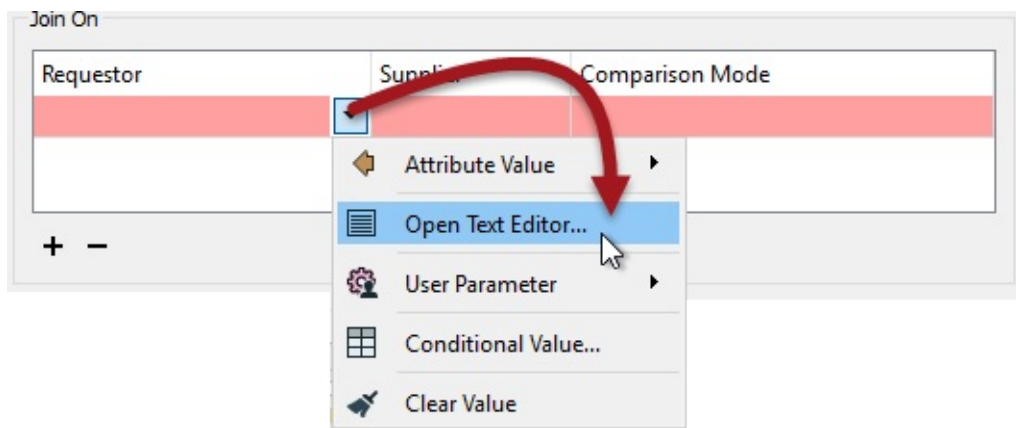
Now we've sorted out the structure of our join keys we can merge the data together with a FeatureMerger.

Add a FeatureMerger to the canvas. Connect the address data as the Requestor and the crime data as the Supplier (we wish to supply the addresses with crime statistics):



Check the parameters for the FeatureMerger.

For the Join On:Requestor parameter, click in the field, select the drop-down arrow, and choose Open Text Editor:

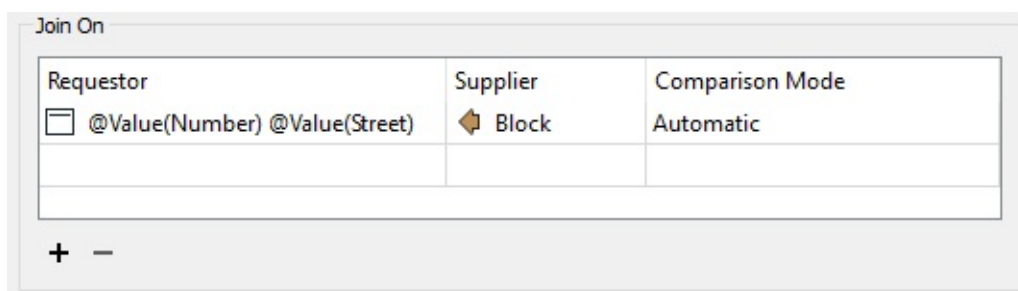


In the Text Editor dialog enter (select):

```
@Trim(@Value(Number) @Value(Street))
```

This will match the structure of the crime data (be sure to include a space character between the two attributes). The Trim function is there to ensure there are no excess spaces on those attributes.

Now back in the FeatureMerger parameters select Block as the Join On:Supplier attribute:



Dr Workbench says...

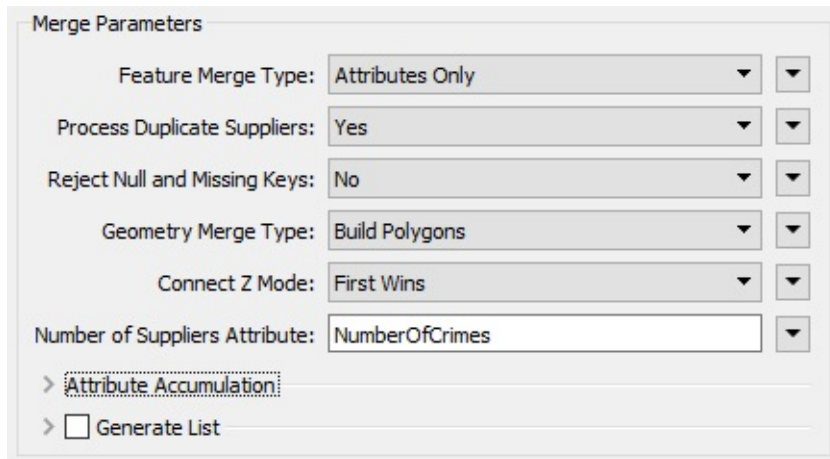
If you are sharp, you may have noticed that the Requestor and Supplier parameters can be defined within a text edit window using functions like @ReplaceString and @UpperCase to replace the previously added transformers.

Now let's set the rest of the FeatureMerger parameters.

Feature Merge Type	Attributes Only
Process Duplicate Suppliers	Yes
Number Of Suppliers Attribute	NumberofCrimes

" the Feature Merge Type should be Attributes Only. Because we can expect multiple crimes per block the parameter Process Duplicate Suppliers should be set to Yes. Set the Number of Suppliers Attribute to be "NumberofCrimes" - this will create an attribute of that name

recording how many crimes occurred in that block.



Merge Parameters

Feature Merge Type: Attributes Only

Process Duplicate Suppliers: Yes

Reject Null and Missing Keys: No

Geometry Merge Type: Build Polygons

Connect Z Mode: First Wins

Number of Suppliers Attribute: NumberOfCrimes

> ☒ Attribute Accumulation

> ☐ Generate List

Click Accept/OK to confirm the changes.

7) Add Inspectors

Add Inspector transformers to the Merged and UnmergedRequestor output ports. This will give us the address data with crime info attached, including addresses where there were no crimes (about half of them).

Save the workspace and then run the translation. The NumberOfCrimes attribute will tell you how many crimes took place in the block on which the address is located.

TIP

If you want to see a list of crimes that took place in each block, open the FeatureMerger parameters and check the box to Generate List. Now re-run the workspace and query a feature (the list won't show up in the table view window).

8) Write Data

Now we know the workspace is performing correctly remove all Inspectors. Connect the Merged and Unmerged ports to the output feature type. Update the output schema to record the NumberOfCrimes attribute.

Rerun the workspace and check the results in the Data Inspector. The data will include the number of crimes and the reworking of the attributes means that individual addresses have been anonymized as well, which is good since this data is being made public.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Pre-process data to get join keys with a matching structure*
- *Join non-spatial data with a join key in the FeatureMerger*

Best Practice

If a workspace runs to completion, and produces the output you want, it can't be bad, can it? Well, yes it can. It's not enough just to put together a functioning workspace, it's also important to use FME in a manner that is both efficient and scalable.

What is Best Practice?

In general terms Best Practice means the best way of doing something; in other words, carrying out a task in the most effective and efficient manner.

Despite the word 'best', we're not presuming the ideas here will meet every need and occasion. The best description of this concept I've heard – and one that fits well here – is:

“a very good practice to consider in this situation based on past experience and analysis”

Why Use Best Practice?

Best Practice in FME can help a user to...

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

Dr. Workbench says...

I'm Dr. Workbench, licensed to practice medicine on FME workspaces.

I learned about Best Practice the hard way, when I was tasked with surgery on a set of someone else's workspaces. They were so badly organized the whole operation took me three times as long as it should have, and ruined my plans to spend the afternoon playing golf!

In this chapter we'll cover three different categories of Best Practice.

Style

This section is a guide to the preferred design for workspaces. The correct style makes a workspace easier to interpret, particularly in the long run when the author might return to it after a period of inactivity.

Methodology

This section covers which techniques make efficient use of Workbench and its components - and which don't! Using Workbench the right way makes for a more productive and efficient experience. It will also cover development techniques like incremental changes and prototyping.

Debugging

This section covers tools and methods to help identify and fix problems in translations.

Style

Style is perhaps the most obvious component of FME Best Practice. You can tell at a glance when a workspace is well-styled and when it is not. But style is more than just looks; a properly designed workspace provides many benefits as it is further developed and edited in the future.

An FME Workspace Style Guide

A good style of design makes it easier to navigate and understand an existing workspace. This is important when workspaces might need to be edited by other users, or when you intend to make edits yourself at a later date.

Ms. Analyst says...

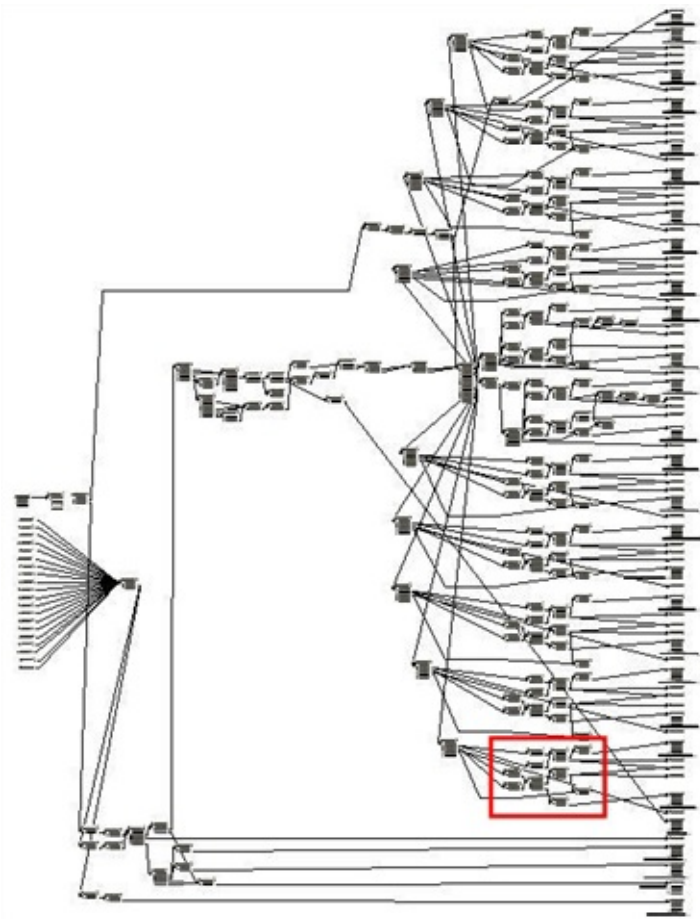
It's very much like a computer programmer adding comments to their code to explain their actions. As an example, nearly 30% of FME's codebase consists of comment lines.

Specifically, a good style can help a user to...

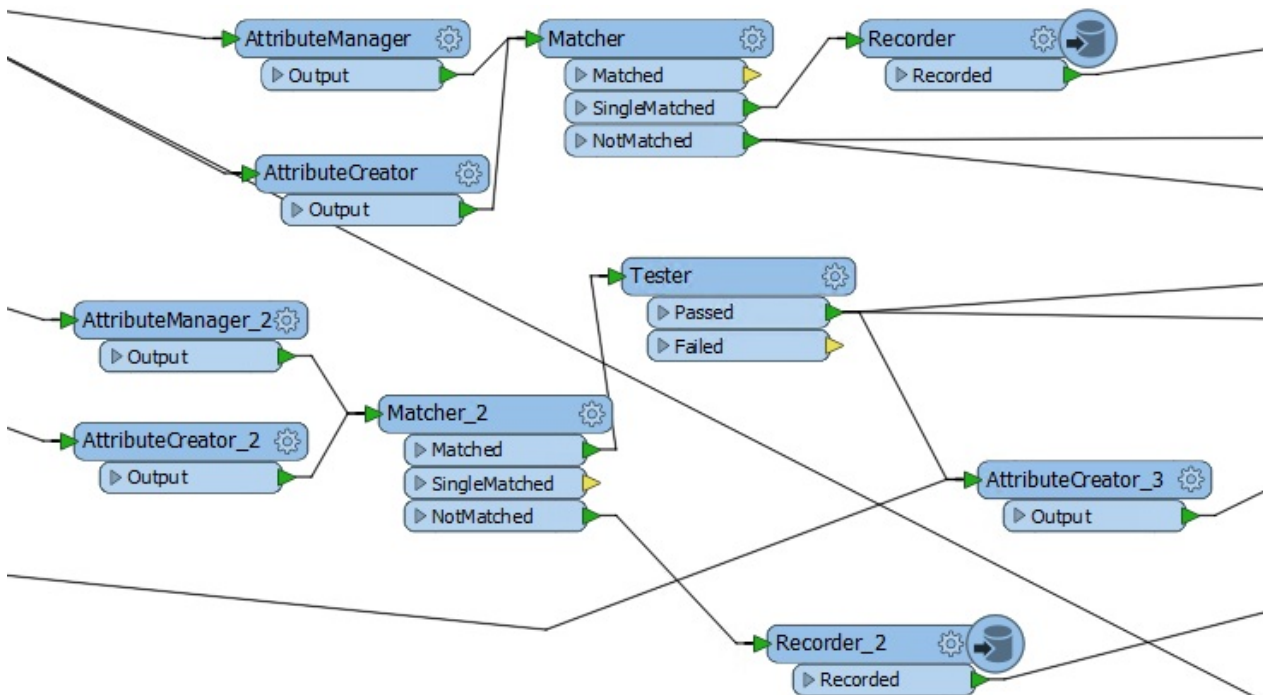
- Distinctly define different sections or components of a workspace
- Quickly navigate to a specified section or particular transformer
- Pass a workspace on to another user for editing
- Rename workspaces and content with a more explanatory title

Example of Poor Design

You need proof? Well, would you want to be given the task of editing this workspace?



Can you even tell what this section of workspace is doing?



Aunt Interop says...

I'm Aunt Interop. I write a blog helping out users with their various FME problems.

As I always say, size doesn't matter! You should always use Best Practice, whether it's a small workspace or training exercise, or a large-scale project. Getting into the habit helps make your smaller projects scalable.

If you don't design a workspace well from the very start, it will just become harder and harder to make edits as you work on it.

Annotating Workspaces

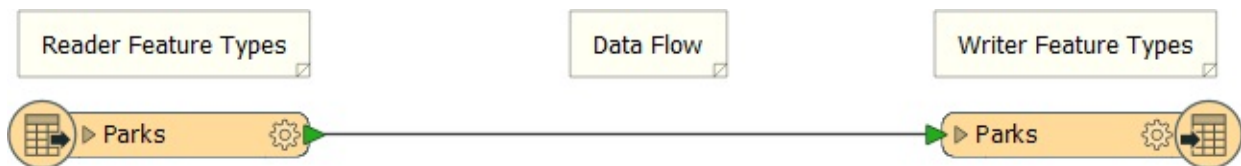
Annotation is a key method for a clear and comprehensible design.

Annotation helps other users understand what is supposed to be happening in the translation and also helps the creator when returning to a workspace after a long interval (take it from me that this is especially important!)

There are a number of different types of annotation that can be applied to a workspace.

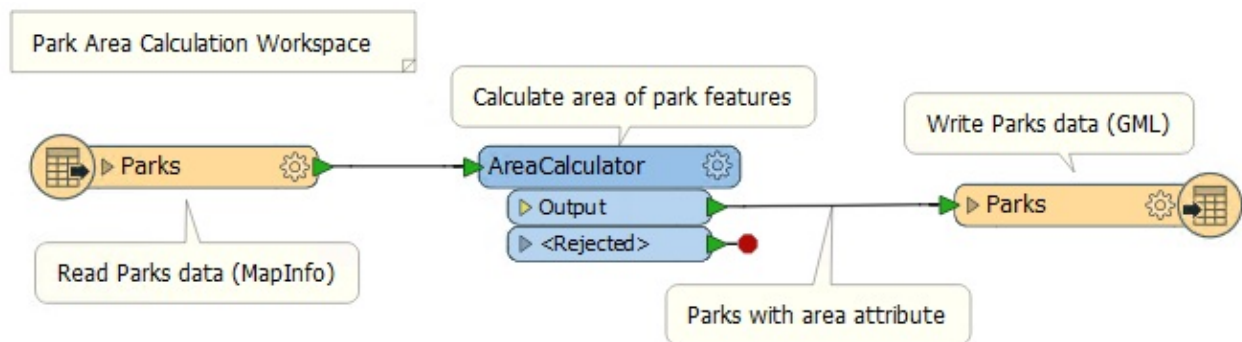
Header Annotation

By default FME adds three annotations to a newly-created workspace. The annotations are basic comments that indicate the source data, the transformation, and the destination data. They can be deleted or, optionally, left ungenerated, and therefore may not appear in every workspace.



User Annotation

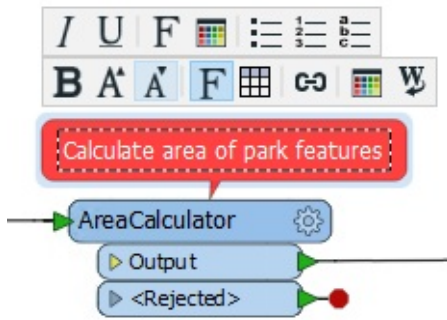
User annotation is a comment created by the user. It can be connected to a workspace object (transformer or feature type), can be connected to a workspace connection, or can float freely within the workspace.



To create floating user annotation, right-click the canvas and select Insert Annotation.

To create attached user annotation, right-click a workspace object and select Add Annotation, or use the shortcut Ctrl+K.

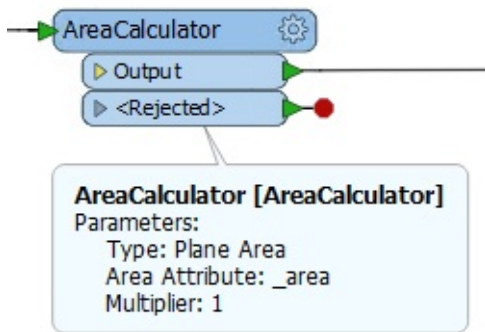
When you place an annotation you have the opportunity to change the font style, font size, and background color; plus you can also add hyperlinks, bullet points, and tables.



Summary Annotation

Summary annotation is an FME-generated comment that provides information about any object within the workspace. This item can be a source or destination feature type, or a transformer.

Summary annotation is always colored blue to distinguish it from other annotation. It's always connected to the item to which it relates and cannot be detached.

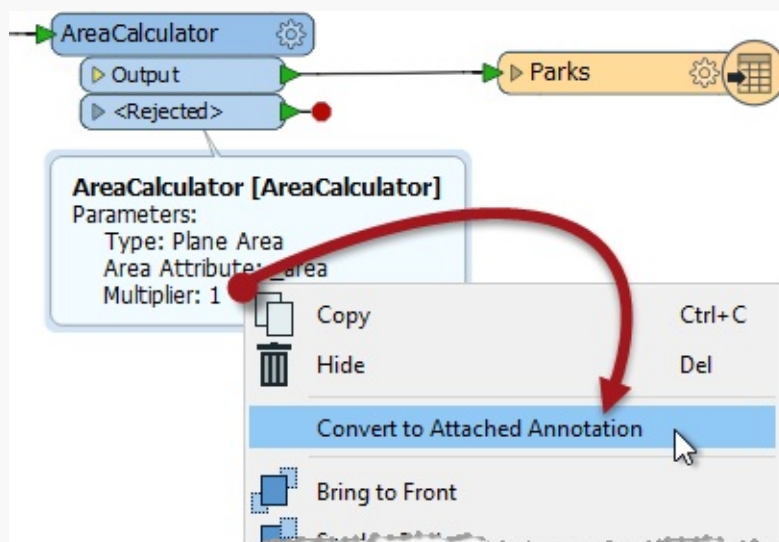


The nice thing about Summary Annotation is that it automatically updates in response to changes. That makes it very useful for checking transformer parameters (or reader/writer schemas) at a quick glance. It's particularly useful in situations where the parameters are set through a wizard and are more awkward to check (for example, the SchemaMapper or FMEServerJobSubmitter transformers).

NEW

A good idea is to use summary annotation to show **what** actions are taking place; but use user annotation to clarify **why** an action is being carried out.

However, in FME2017 you can convert a summary annotation to a user (attached) annotation by using this context menu option:



This allows you to extract the information from a summary annotation, but edit it as you would a user annotation. Note that a converted summary annotation no longer updates automatically!

Annotation Options

Right-click on an annotation object and you will find its context menu includes a number of options to control and edit that object, such as:

- Set Background Color
- Word Wrap
- Follow Attached Object
- Attach to Feature Type
- Attach to Transformer
- Attach to Connection
- Detach

Bookmarks

A bookmark, like its real-world namesake, is a means of putting a marker down for easy access.

With FME the bookmark covers an area of workspace that is usually carrying out a specific task, so a user can pick it out of a larger set of transformers and move to it with relative ease.

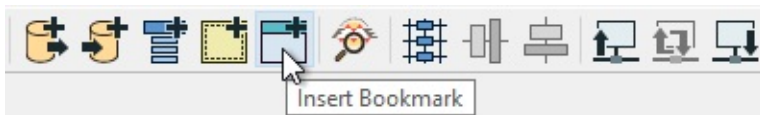
Why use Bookmarks?

Bookmarks play an important role in a well-styled workspace for a number of reasons, including these.

- Sectioning: As a way to divide a workspace into different - clearly marked - sections
- Access: As a marker for quick access to a certain section of workspace
- Editing: As a means to move groups of transformers at a time

Adding a Bookmark

To add a bookmark, click the Bookmark icon on the toolbar.



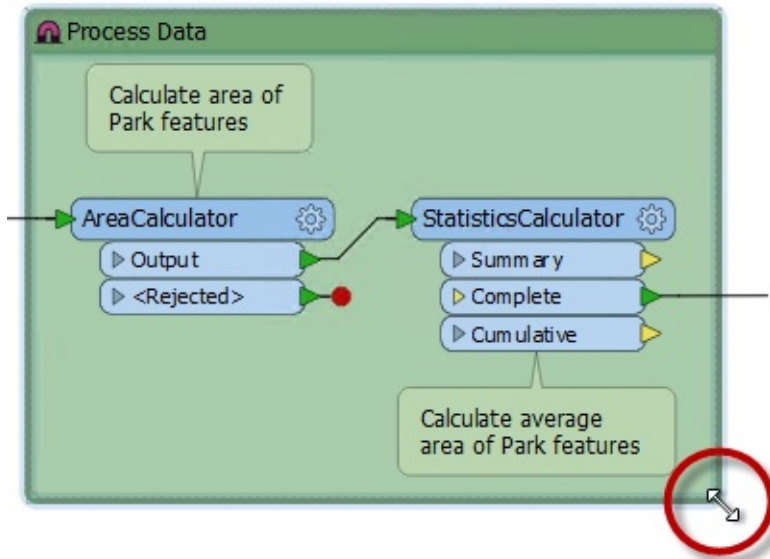
Whereas a traditional bookmark marks just a single page in a book, the FME bookmark can cover a wide area of the canvas. A single workspace can be divided into different sections by applying multiple bookmarks.

TIP

If any objects on the workspace canvas are selected when a bookmark is created, the bookmark is automatically expanded to include those items.

Resizing and Editing a Bookmark

To resize a bookmark simply hover over a corner or edge and then drag the cursor to change the bookmark size or shape.



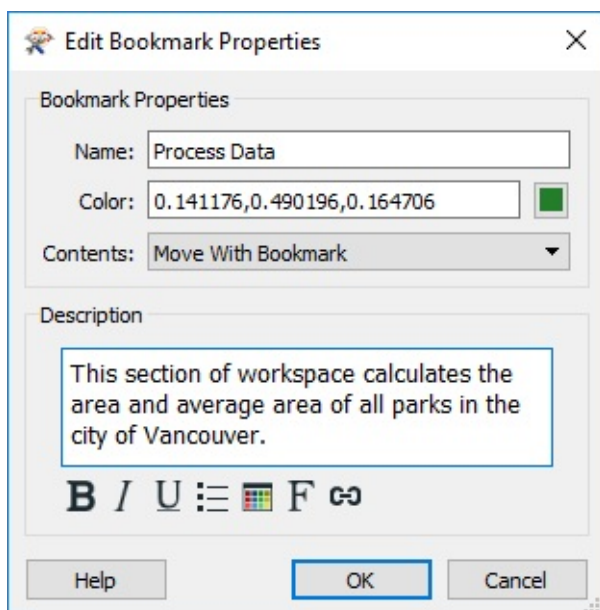
TIP

Bookmarks are shown either as a frame around white-space or filled with color.

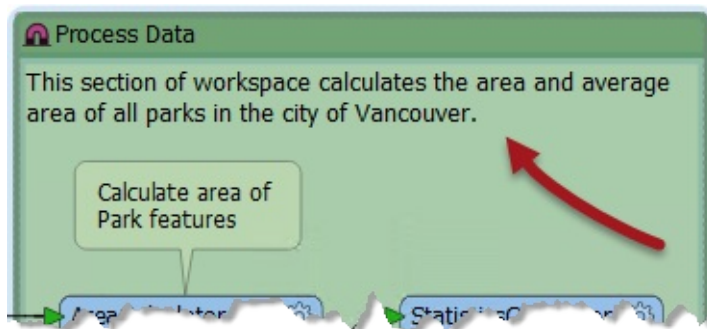
Tools > FME Options on the menu bar opens a dialog with a number of sections, one of which (Workbench) has an option to have color-filled bookmarks.

Bookmark Properties

Double-click on the header part of a bookmark to open the bookmark properties dialog:



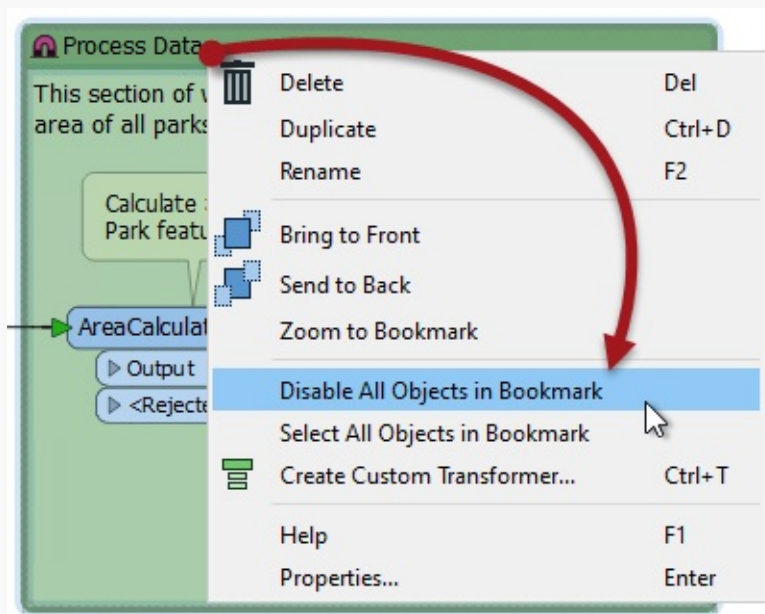
Here you can change both the name and color of the bookmark, and add a description that appears at the top of the bookmark area:



These properties can therefore make your bookmarks both more visually appealing, but also provide extra information as a form of annotation.

TIP

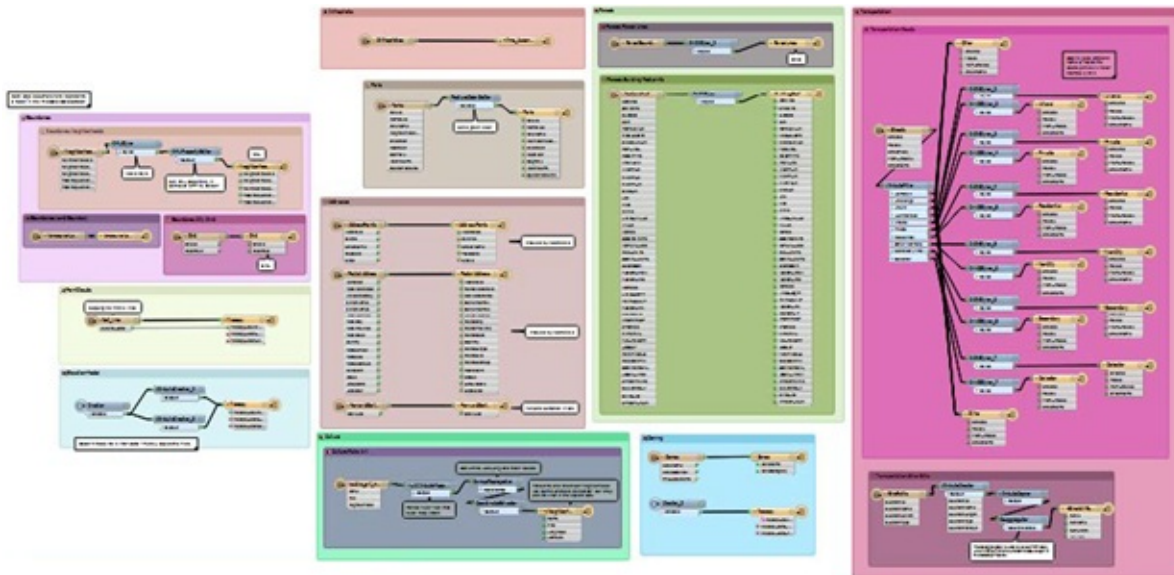
Right-clicking on a bookmark allows you to select all objects within the bookmark, or to disable all of those objects, making it useful for testing purposes.



Bookmarks for Sectioning

A bookmark is a great way of indicating that a particular section of a workspace is for a particular purpose. By subdividing a workspace in this way, the layout is often a lot easier to follow.

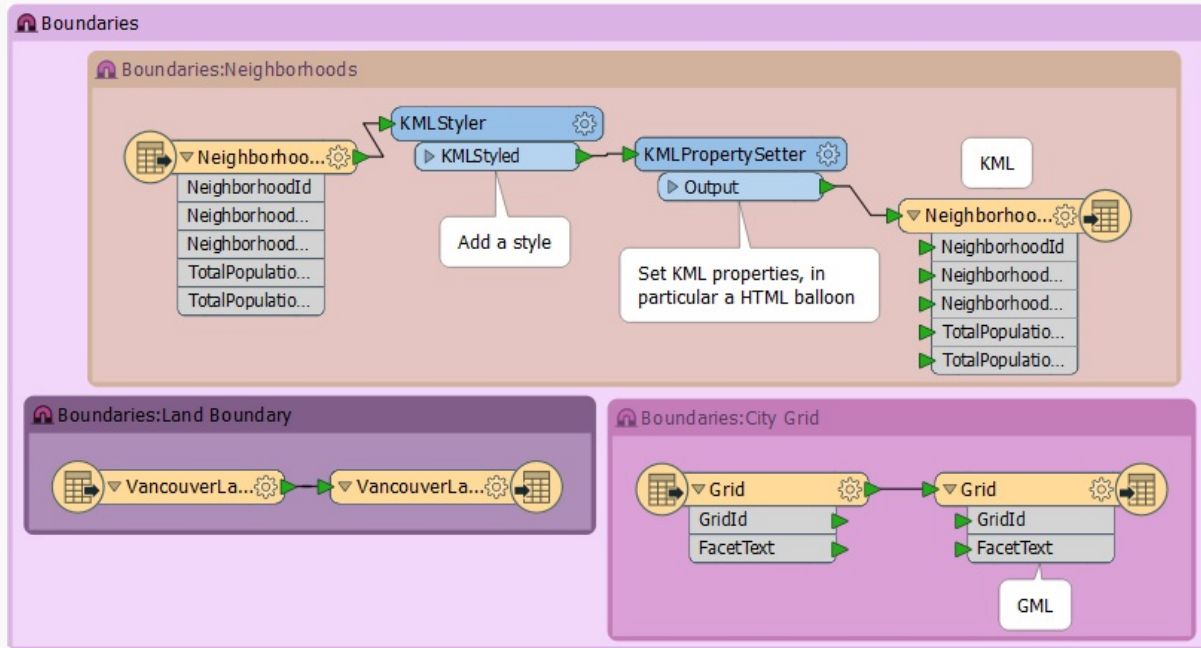
As one user has put it, bookmarks are like paragraphs for your workspace!



The above workspace illustrates nicely how to mark up different sections of a workspace using Bookmarks.

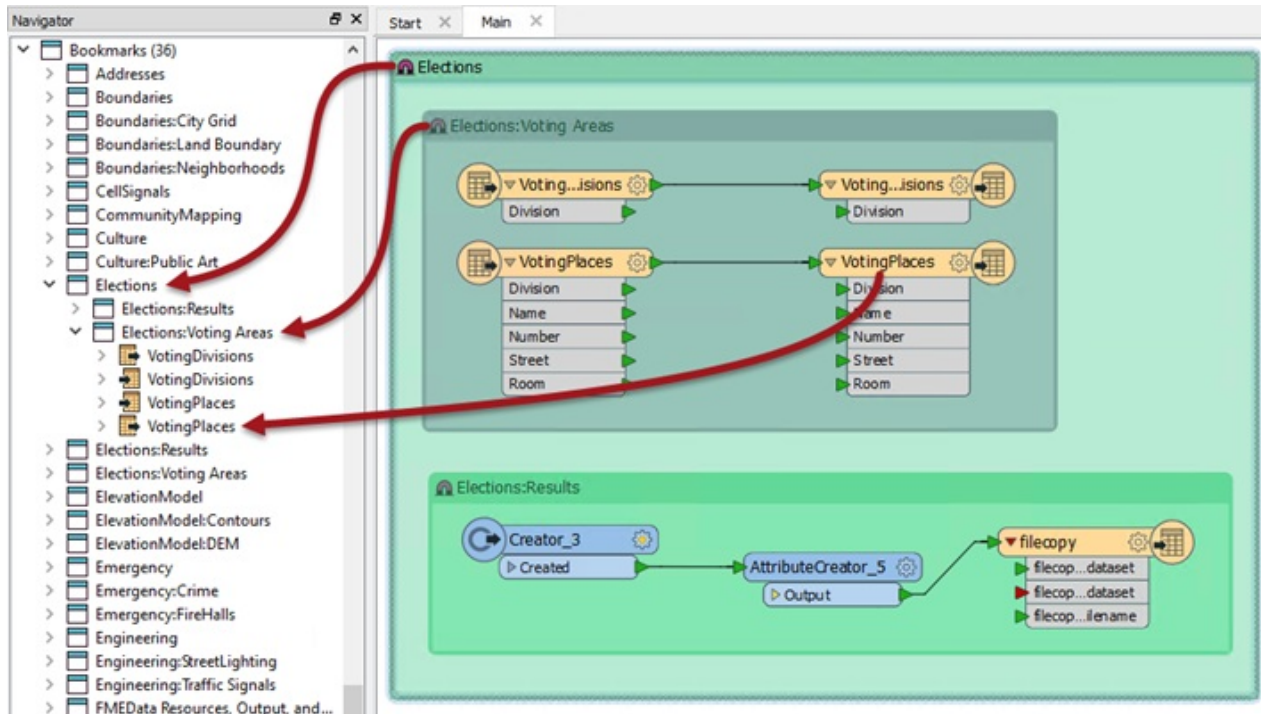
Mr E. Dict (Attorney of FME Law) says...

In my considered opinion, it's perfectly legal to nest bookmarks. Nesting means to place one bookmark inside another. In this manner each 'section' of a workspace can be divided into subsections.



Bookmarks for Quick Access

Bookmarks are listed on the Workbench Navigator. Each bookmark is depicted as a folder and can be expanded to show its contents. This may include feature types, transformers, or other - nested - bookmarks:



Clicking a bookmark in the Navigator selects that bookmark and brings it into view.

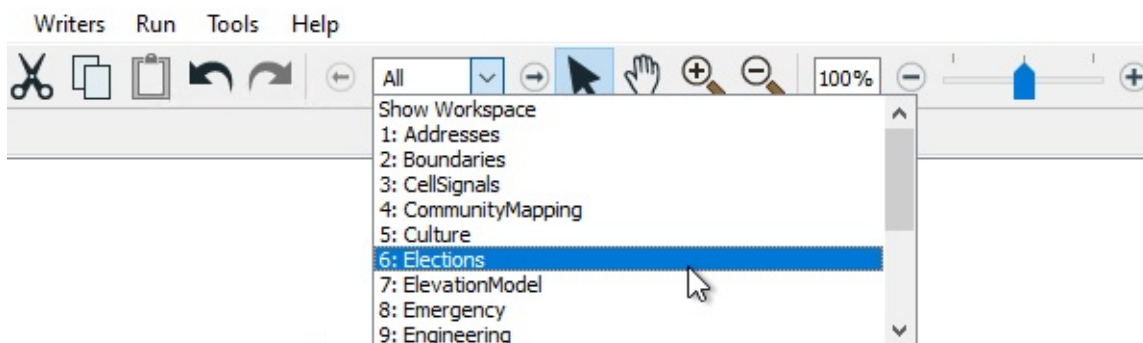
Double-clicking it selects the bookmark and brings it into the centre of the view.

So, when bookmarks have been used to divide a workspace into sections, they can also be used to navigate between different parts of that workspace.

In this way bookmarks are like the chapter headings in a book!

Workspace Presentation

Bookmarks can also be made to appear on the FME Workbench toolbar:



Besides being a way to quickly access bookmarks, this tool can be used as what is known as "Presentation Mode" in FME Workbench. By clicking the arrow button (or pressing the keyboard spacebar) you flip from bookmark to bookmark, using animation, in a way that would be very useful when showing the workspace as part of a presentation.

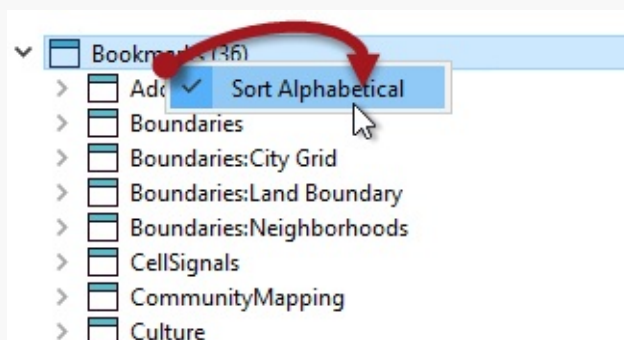
TIP

To access the functionality you need to make sure it is added to the toolbar. You can do this by right-clicking on the toolbar and using the customize option.

Dr. Workbench says...

The order of bookmarks in that window is alphabetical and that might not always be the same order that you wish to present a workspace.

In that case, right-click on Bookmarks in the Navigator window and turn off the default option to "Sort Alphabetically".



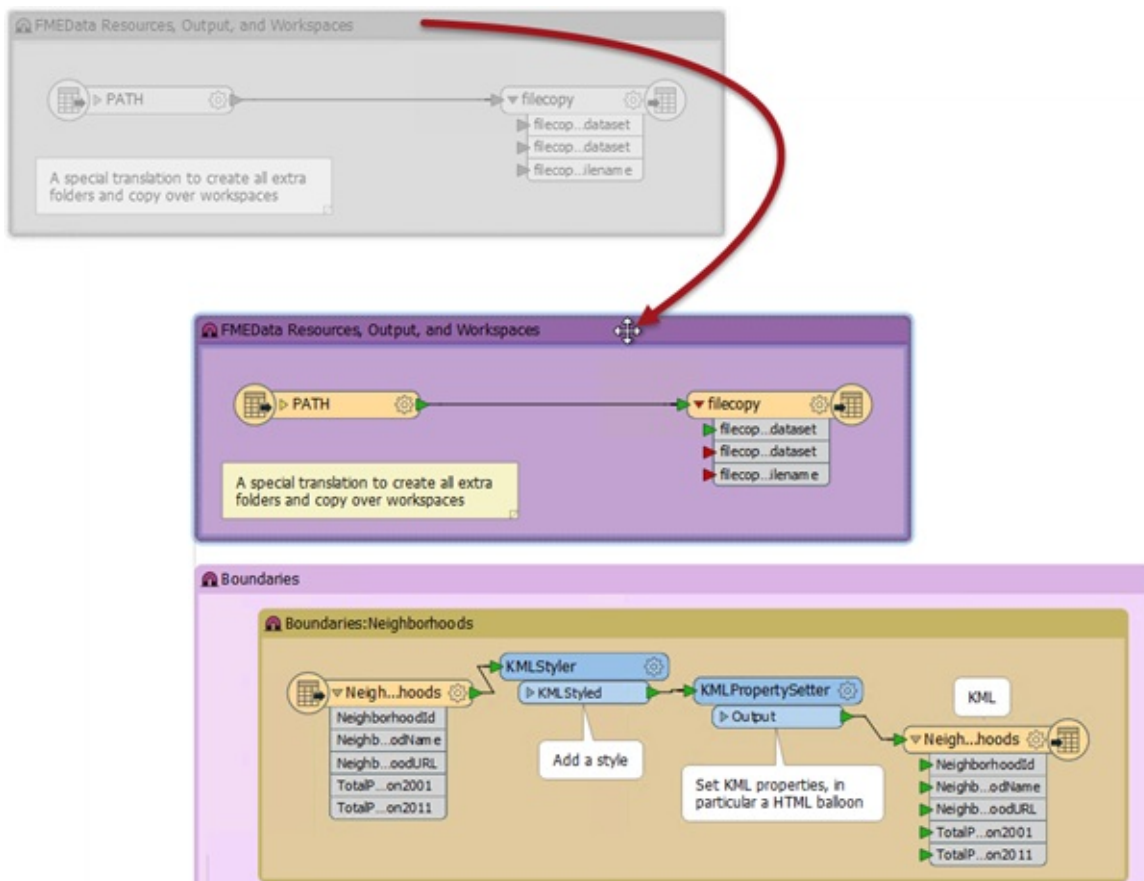
Bookmarks can then be dragged up and down in the Navigator window to give the correct order. Additionally, a new option on the bookmark Properties dialog allows you to exclude specific bookmarks from the Bookmark Navigator. Nested bookmarks are excluded by default, so must be turned on to be included in the navigator.

For more information see this [article on bookmarks](#) on the Safe Software blog.

Bookmarks for Editing

Bookmarks define a section of workspace containing a number of objects. When editing a workspace without bookmarks, moving objects is done by selecting the object or objects and dragging them to a new position.

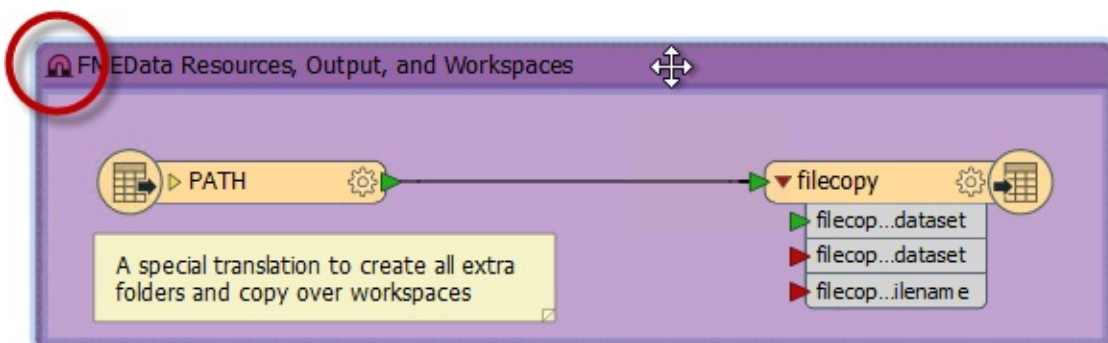
However, when a workspace is divided by bookmarks, objects can be moved by simply dragging the bookmark to a new position. When an object is located inside the bookmark, it moves as the bookmark does.



Using this technique large groups of objects can be moved about the workspace canvas, in order to create a clearer layout.

The Magnet Icon

Each bookmark in FME has a *magnet* icon in its top-left corner. Clicking on the icon toggles it from active to inactive (and vice versa).



An inactive magnet (colored grey) allows the bookmark to be moved without disturbing the contents. It's useful when you need to reposition the bookmark without also moving its contents.

Miss Vector says...

Here are some Best Practice questions. You do want to be the best, don't you?

Which of the following is NOT a method of creating a bookmark?

- 1. Click the Insert Bookmark button on the toolbar*
- 2. Select a transformer, right click, choose Create Bookmark*
- 3. Select multiple transformers, right click, choose Create Bookmark*
- 4. Use the Ctrl+B shortcut*

Which of these is NOT a type of annotation in FME Workbench?

- 1. Header Annotation*
- 2. Parameter Annotation*
- 3. User Annotation*
- 4. Summary Annotation*

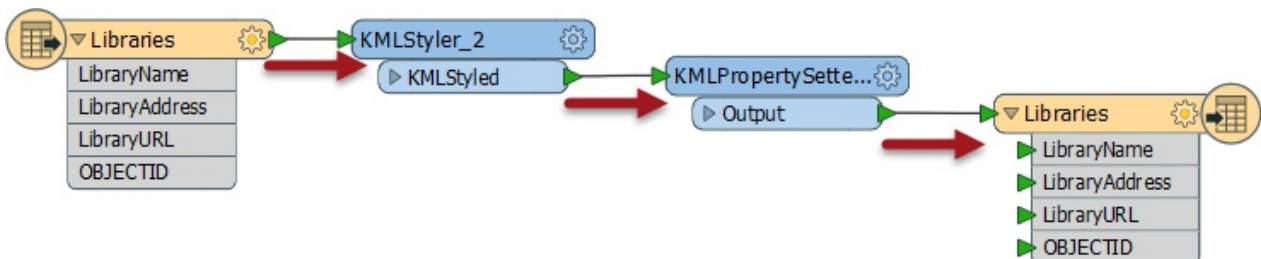
Object Layout

The positioning of workspace objects and the care taken in connecting them can really make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

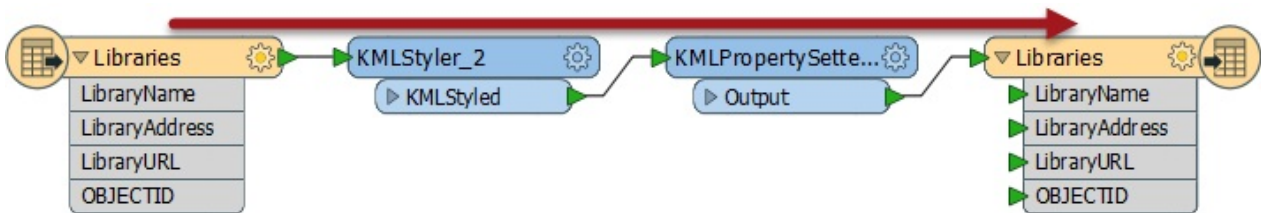
As a user once told us, a neat design inspires confidence from your clients. In fact, you should always create neat designs - even if you don't think the client will see it - because it demonstrates professionalism.

Object Layout

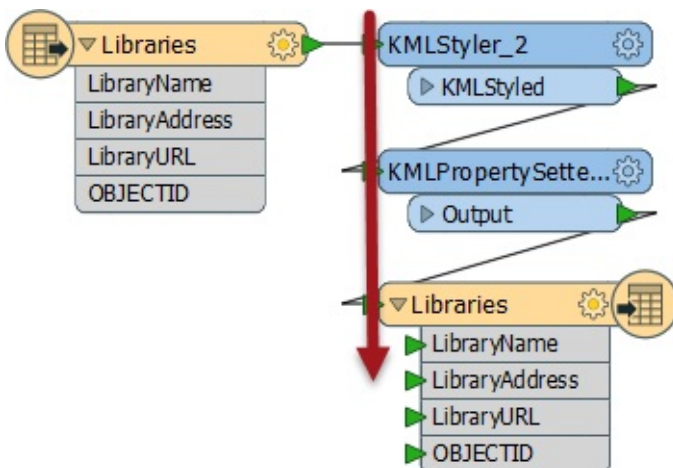
Layout methods vary from user to user. Some users like to line up objects so that all *connections* are horizontal:



Others prefer the tops of *objects* to be aligned horizontally, with angled connections:



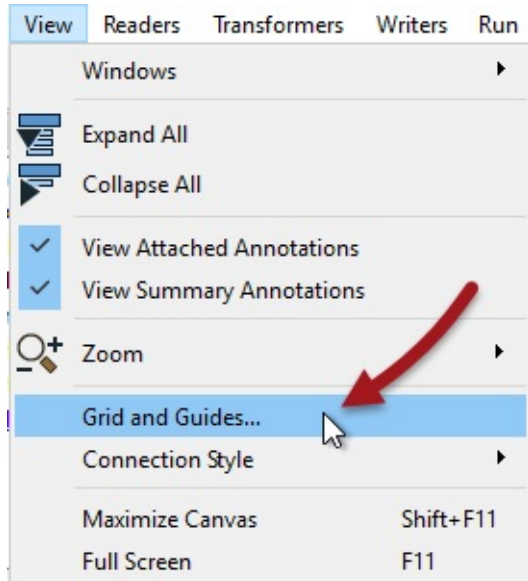
Some prefer to align object edges vertically:



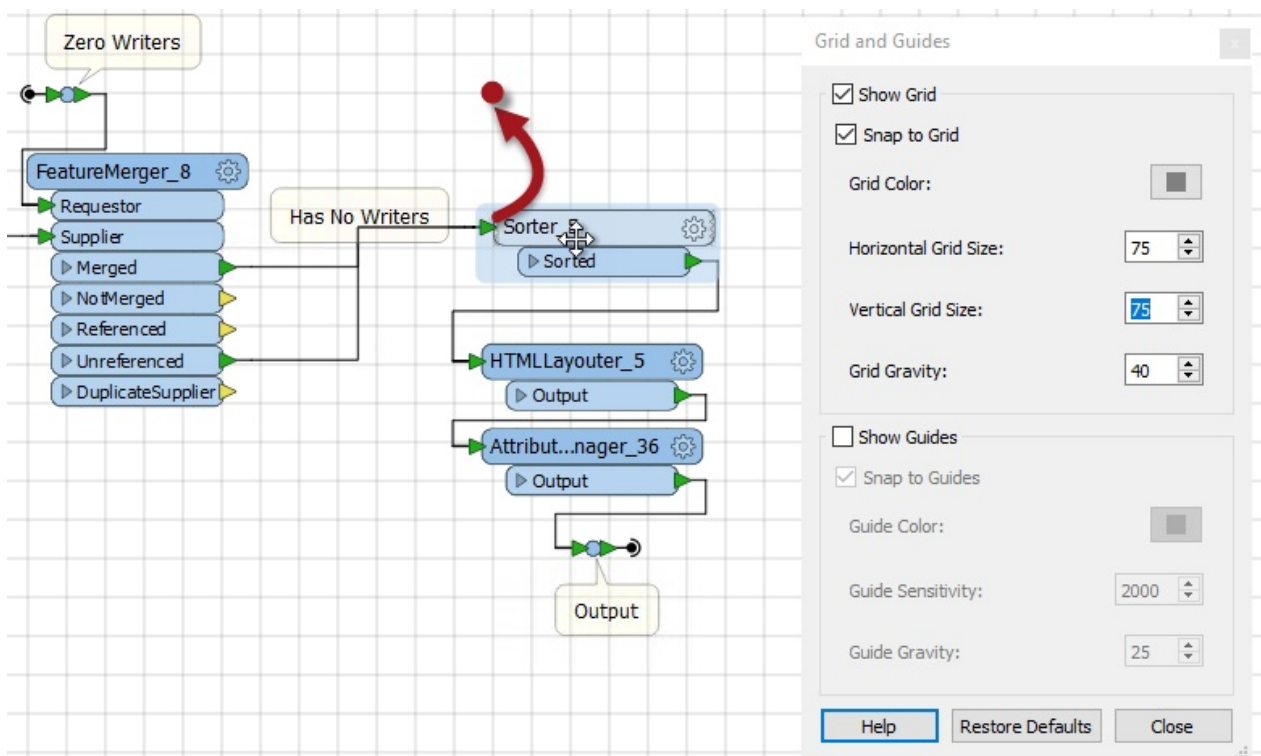
The style used is more a personal preference than a definite rule, but what is important is consistency. A workspace that has no obvious layout style, or an inconsistent one, does not inspire confidence in the author's abilities!

Grid and Guides

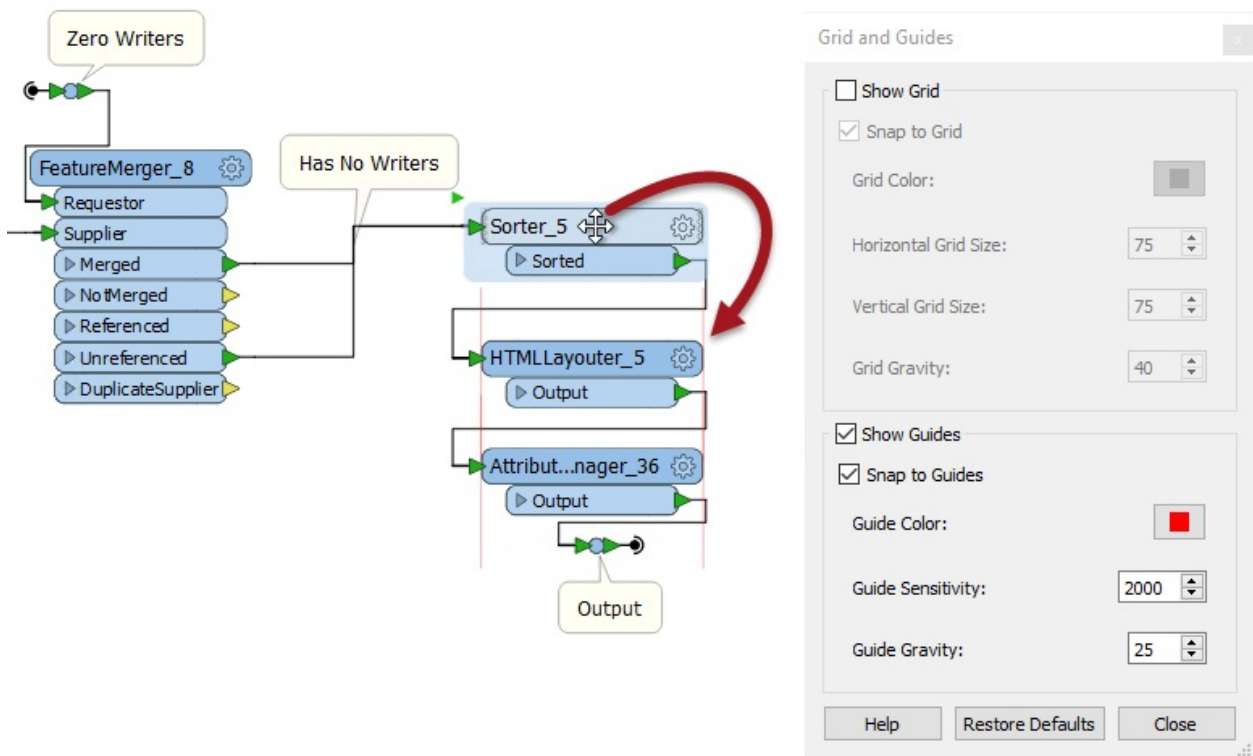
Grids and Guides are a tool to help align workspace objects in a neat and tidy way. This functionality is accessed through View > Grid and Guides on the Workbench menubar.



Show Grid causes a grid of lines to be displayed on the Workbench canvas. Snap to Grid causes all objects - such as the summary annotation highlighted - to snap onto the intersection of grid lines when moved. In this way objects can be more easily lined up.



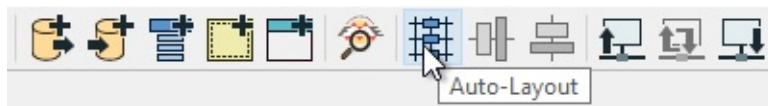
Show Guides causes guidelines to be displayed on the Workbench canvas whenever an object is moved, and lines up approximately to another canvas object. Snap to Guides allows an object to be snapped onto a highlighted guideline.



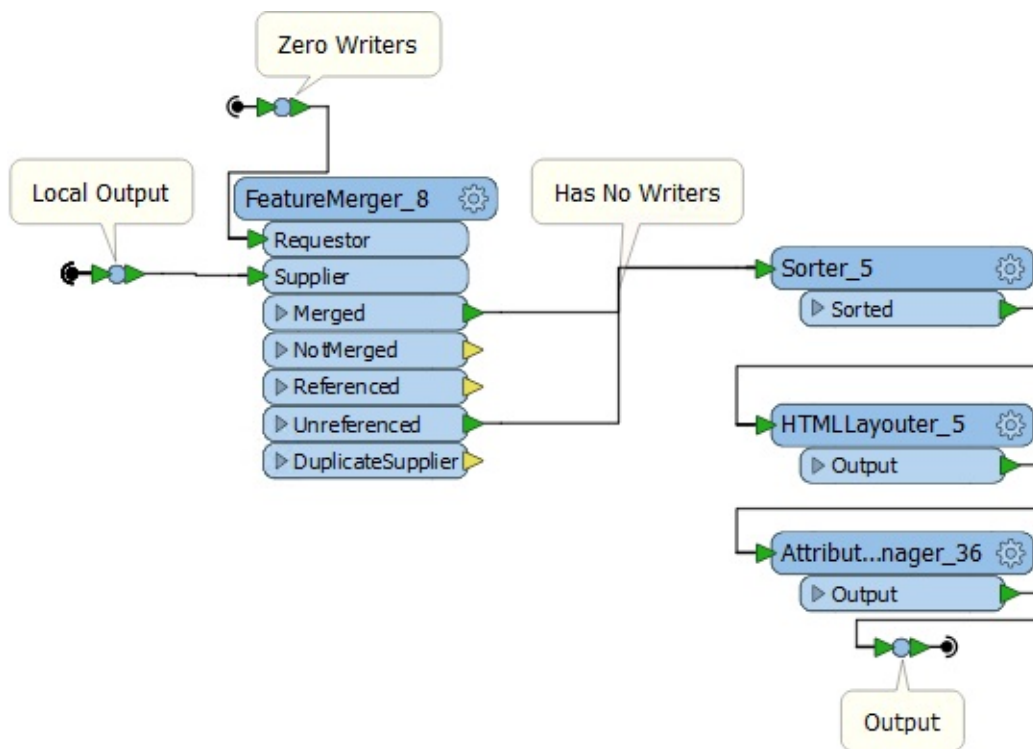
These two tools make it very simple for workspace objects to be aligned in a pleasing style.

Autolayout

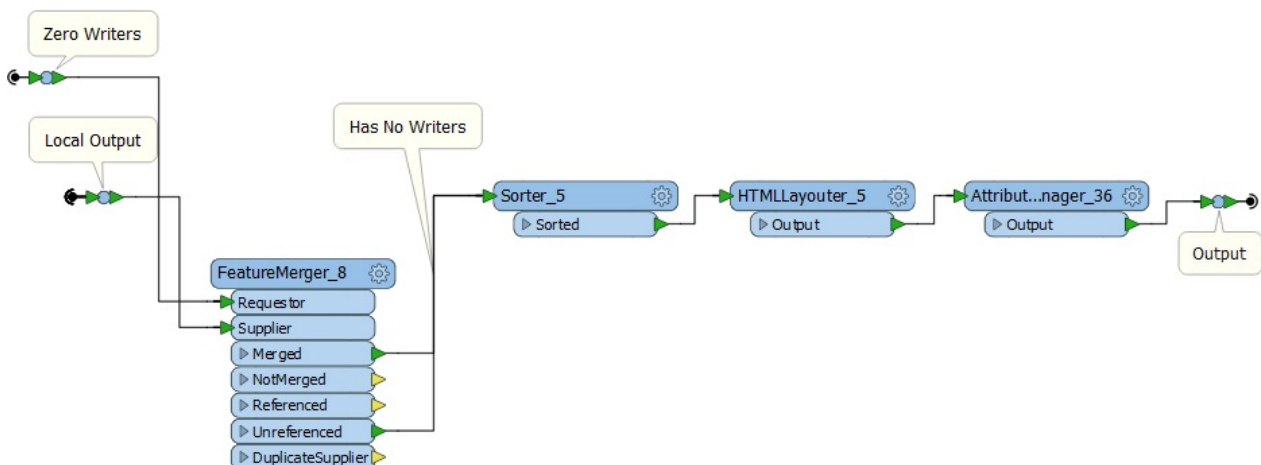
The Autolayout tool appears on the toolbar of FME Workbench:



Clicking the toolbar button will layout either all of the workspace, or just objects that are currently selected. For example, if this section of workspace is selected:



The Autolayout tool reorganizes the contents to this:



As you can see, the autolayout tends to use a horizontal pattern, with the tops of objects aligned. Therefore it's better to select groups of transformers at a time, when using this tool, rather than trying to lay out the entire workspace in a single action.

NEW

Autolayout is new for FME2017... well, it used to exist in older FME Versions (2013 and earlier) and has now been brought back in a new, improved form.

In general, the layout algorithm is better than it used to be, but it still can't compare with taking the time and effort to manually organize your object layout.

Connection Style

It's also worth noting that object positioning is only part of a good layout. The other key part is the connection style.

As with the positioning of workspace objects, the care taken in connecting them can really make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

Connection Styles

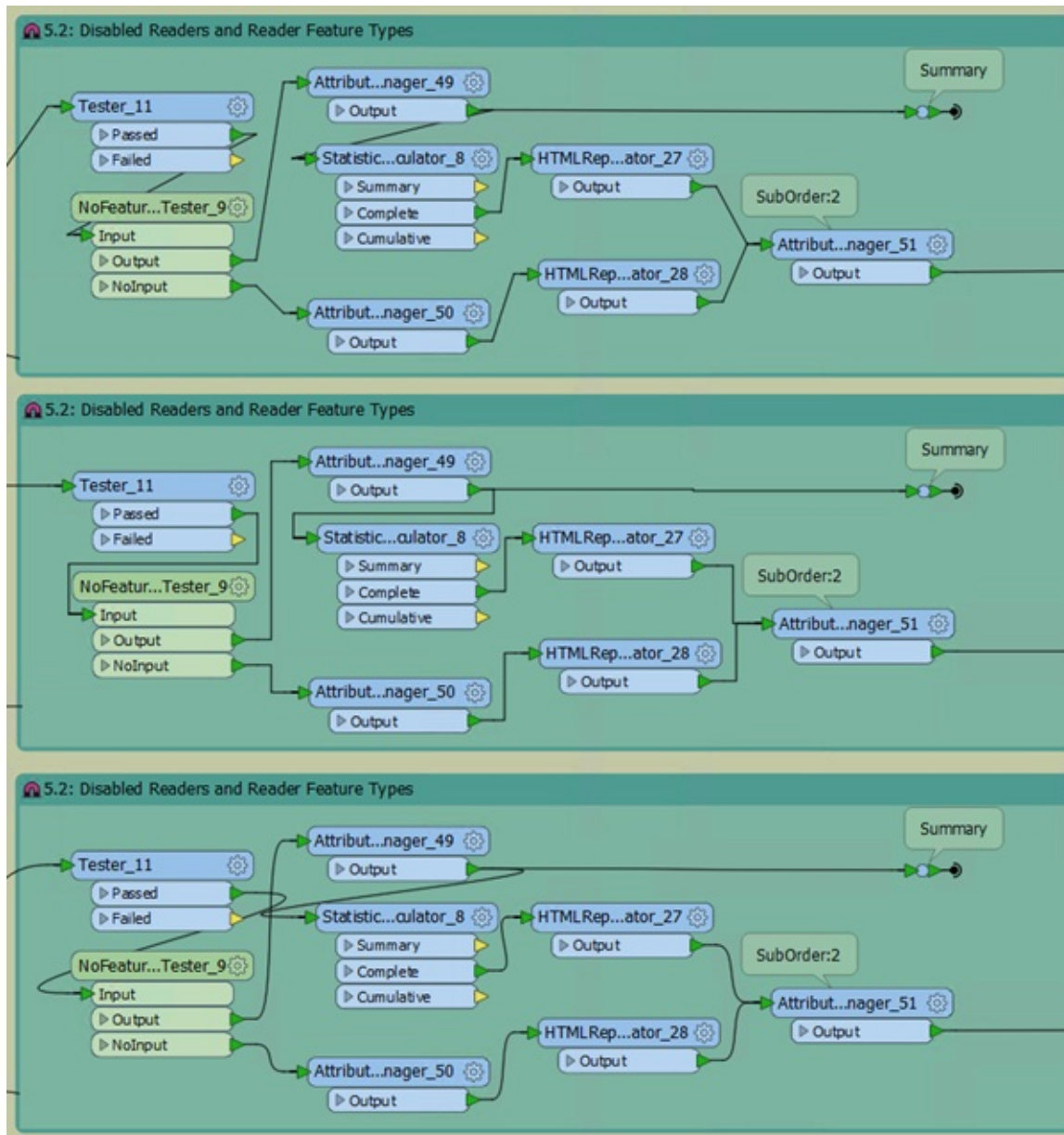
Connections are the lines between objects on the workspace canvas. There are three different styles of connection that you can create in Workbench.

- Straight: The original connection style; a straight line between two objects
- Squared: A style that evokes a Manhattan skyline through squared connections
- Curved: A style that changes straight lines to curved

NEW

These different styles are completely new for FME2017

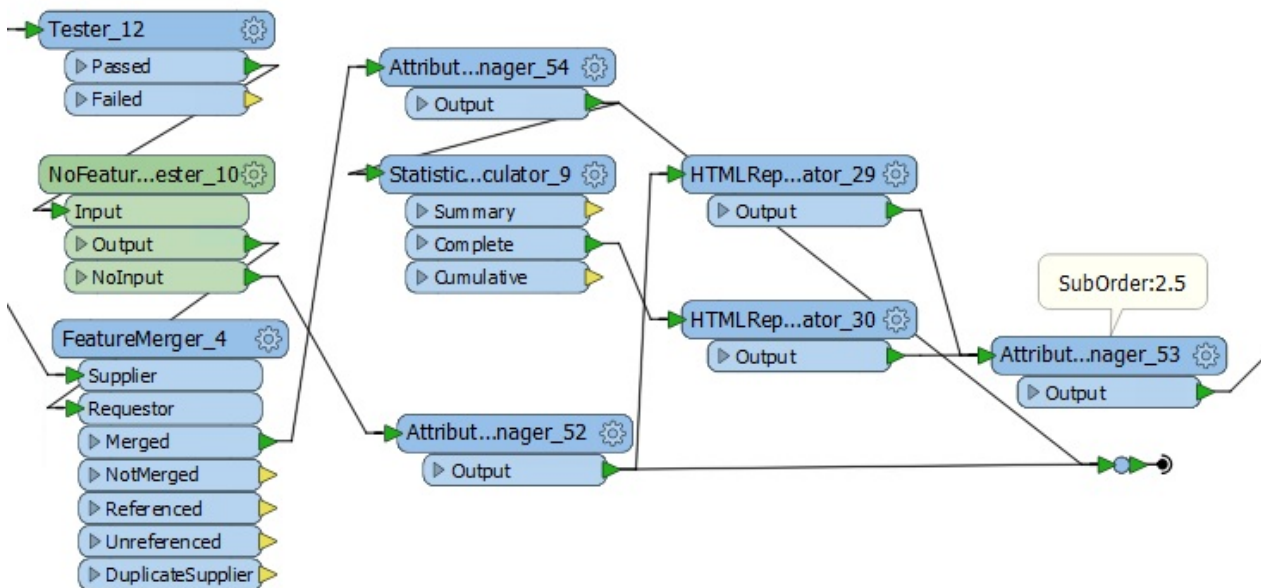
This image shows a comparison of the three styles. You can switch between styles using either the View menu, the FME Options menu, or the shortcut Ctrl+Shift+C.



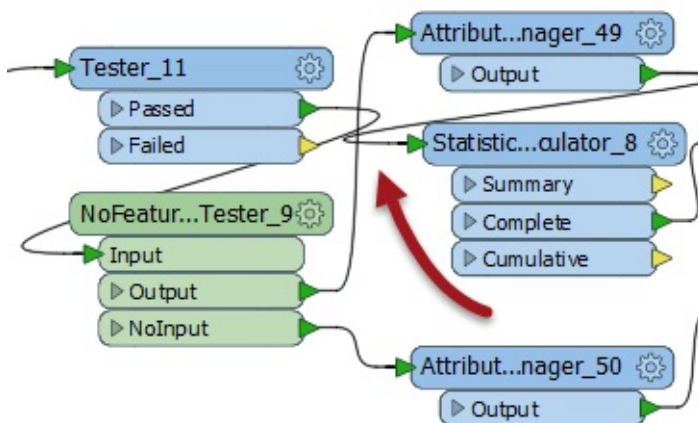
Once more, there is no right or wrong choice about which style to use; it is more a personal preference. However, the best overall look is usually defined by an object layout that will be subtly different for each style, as the following shows.

Non-Overlapping Connections

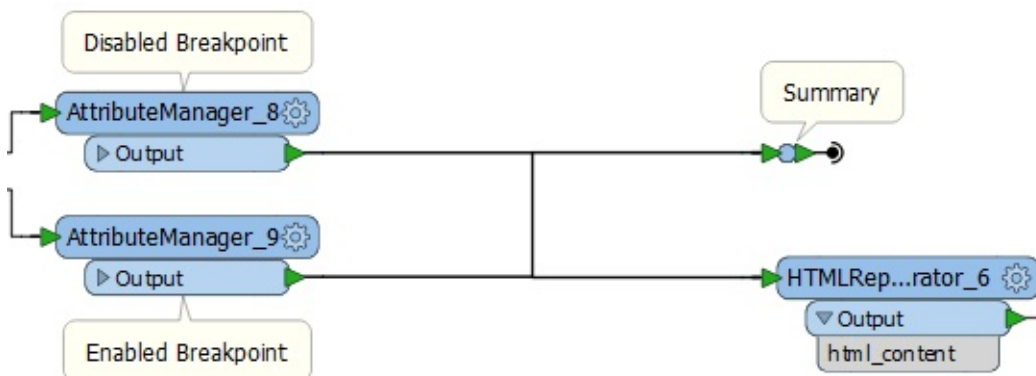
One of the most obvious failings of a workspace design is to have connections that cross over each other, for example like this:



So overlapping connections should always be avoided. However, sometimes the choice of connection style can be more likely to lead to this happening. For example curved connections tend to cross over more than straight ones:



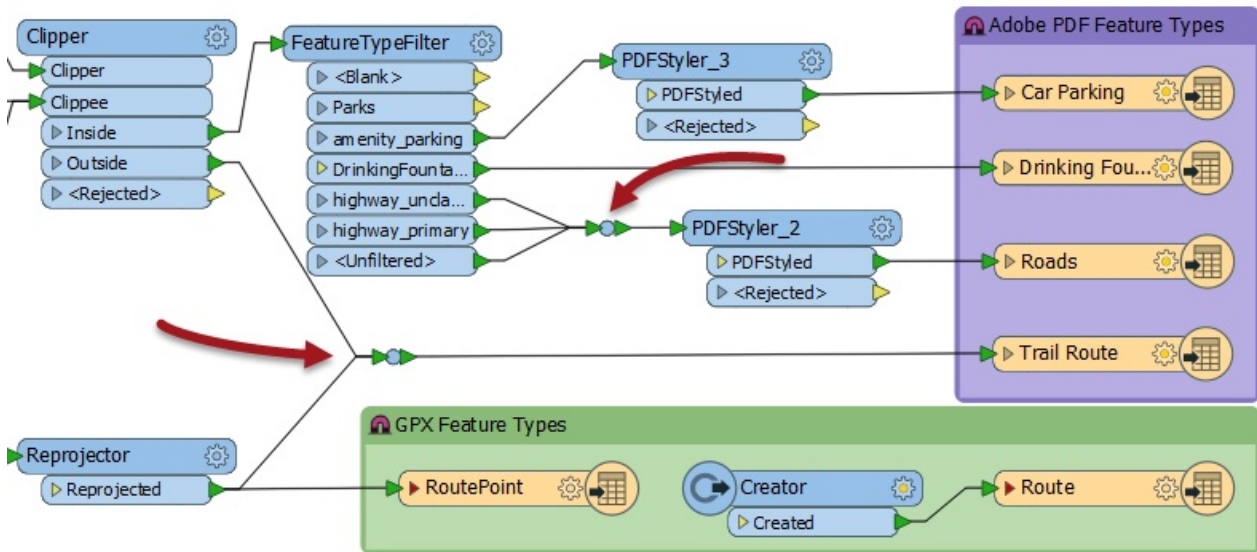
...and squared connections can sometimes cross in ways that are difficult to decipher:



Because these issues can spring up when you switch connection style, and would need objects moving to clear them up, it's wise to choose a particular connection style and layout technique and stick with it.

Junctions

There is one transformer in FME Workbench that is designed to be used to enhance the layout of objects and connections. That transformer is called the **Junction**.

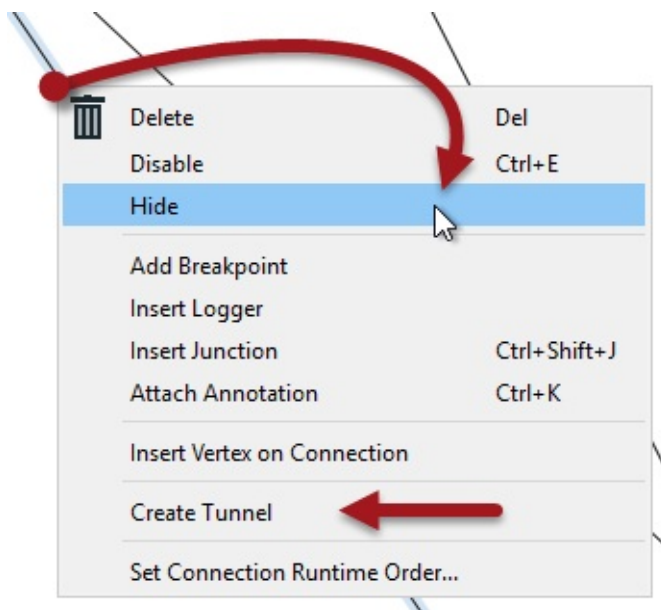


This transformer is a small, node-like object, that carries out no function on the data, but is instead used to tidy connections within a workspace - as in the above screenshot. This makes it an excellent tool for best practice.

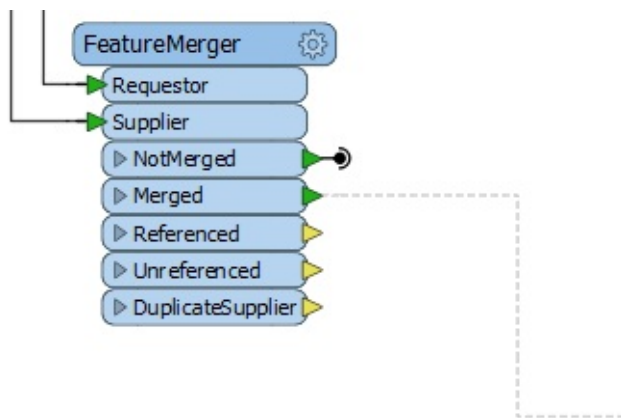
As with any other transformer, a junction can be connected to an Inspector or Logger, and it can have annotation objects attached to it. It also works with both Quick Add and Drag/Connect functionality.

Hidden Connections and Tunnels

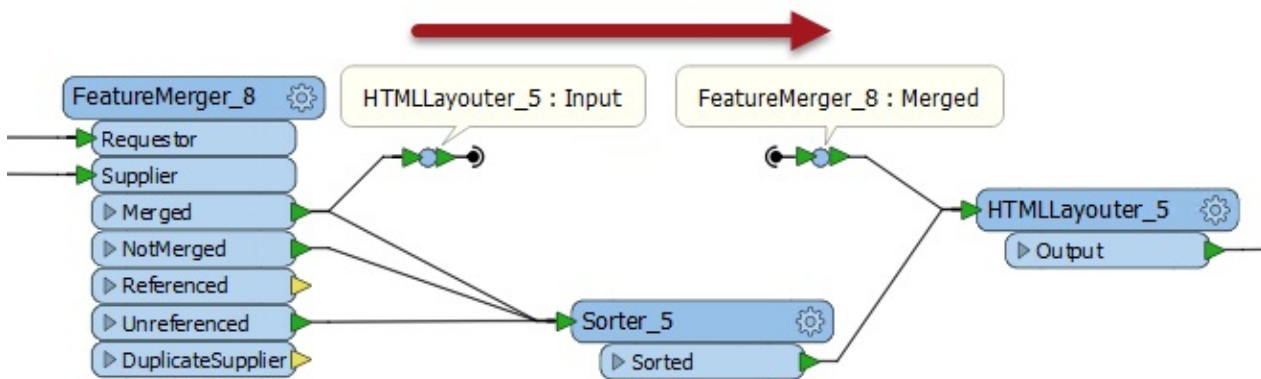
The ability to hide connections is especially useful for avoiding overlaps. To hide a connection, right-click on it and choose the option to Hide:



A hidden connection is represented by a 'transmitter' icon, or by a greyed-out dashed line when the object at one end of the connection is selected:



The other available option is "Create Tunnel". This creates a hidden connection with the addition of an annotated junction transformer at each end:



A tunnel makes a hidden connection slightly more obvious, plus allows for annotation at each end.

Revisualizing Hidden Connections

To view hidden connections, simply click on an object at either end. The connection is highlighted as a greyed-out dashed line.

To return a connection back to view, right-click an object to which it is connected and choose Show Connection(s).

For more information on Tunnels and Junctions see [this blog post](#).

Exercise 1 Applying the Style Guide

Data	City Parks (MapInfo TAB)
Overall Goal	Clean up workspace and apply concepts of style guide
Demonstrates	Style Best Practice
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex1-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex1-Complete.fmw

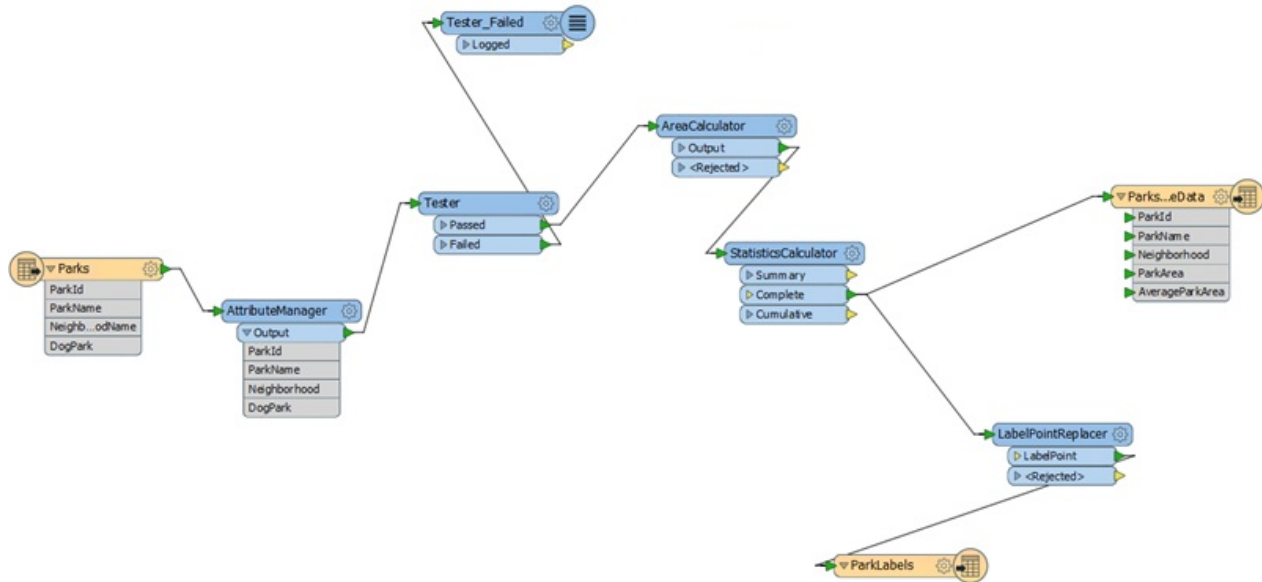
“A good looking well-organized workspace gives the customer the feeling that you have done quality work.”

An FME author (it might have been you) created a workspace to process data for a grounds maintenance project. The results of the workspace are fine and everyone is happy. However, the workspace was created before you knew about the FME style guide. So let's apply a little bit of styling to this project.

1) Open the Workspace

Open the workspace for the grounds maintenance project. You may have already created it, or you can use a pre-defined version from

C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex1-Begin.fmw

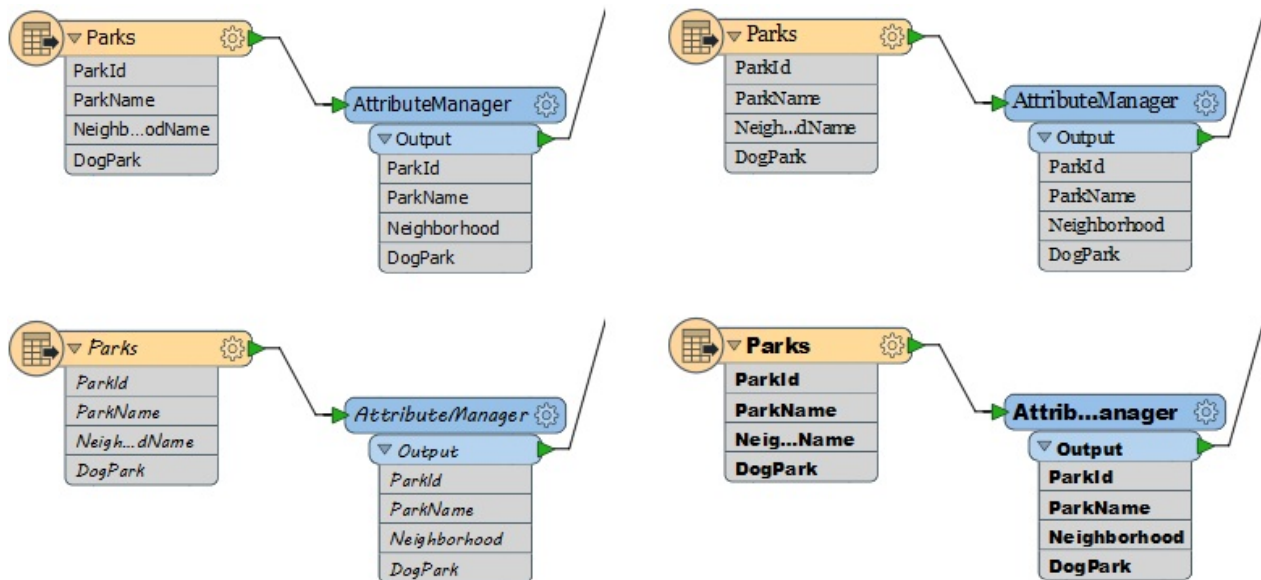


2) Check FME Options

Perhaps the first thing to do is check what FME Options exist to make creating a well-styled workspace easier.

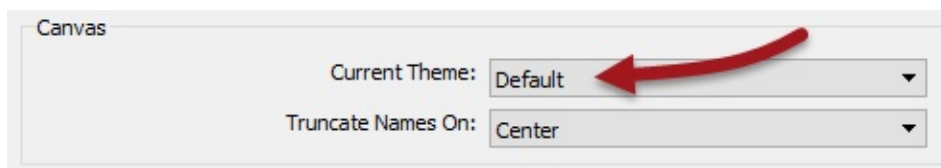
Select Tools > FME Options from the menubar. Click on the Appearance icon on the left of the FME Options dialog.

Notice how it is possible to change the font for the workspace canvas, and the log window, so - for example - you can change the workspace to look like any of these:



The important thing to remember is that these are FME options, not workspace options, and changes made here will only affect YOUR installation of FME. If you save a workspace the font used is not saved and will not be applied if another user opens that workspace file on their FME installation.

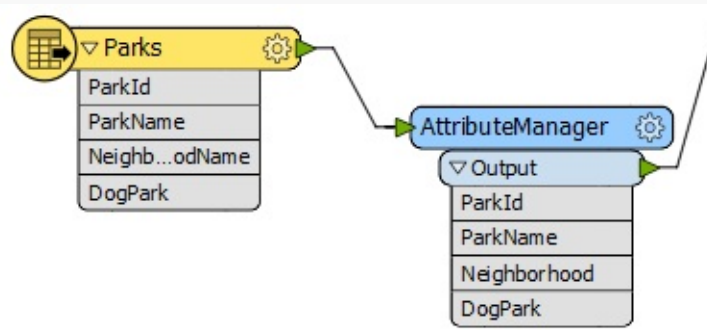
Another option in this dialog allows you to change the theme used by Workbench:



A theme is a set of predefined colors for objects and dialogs within FME Workbench.

NEW

FME2017 introduces a theme designed to assist users with Deuteranopia (red-green color blindness):



If you experience a different form of color blindness - Tritanopia or Protanopia - then let us know and we can increase the priority for implementing additional themes.

Click on the Workbench icon in the FME Options dialog and you will find options for drawing bookmarks with a filled background, and options for annotation defaults:



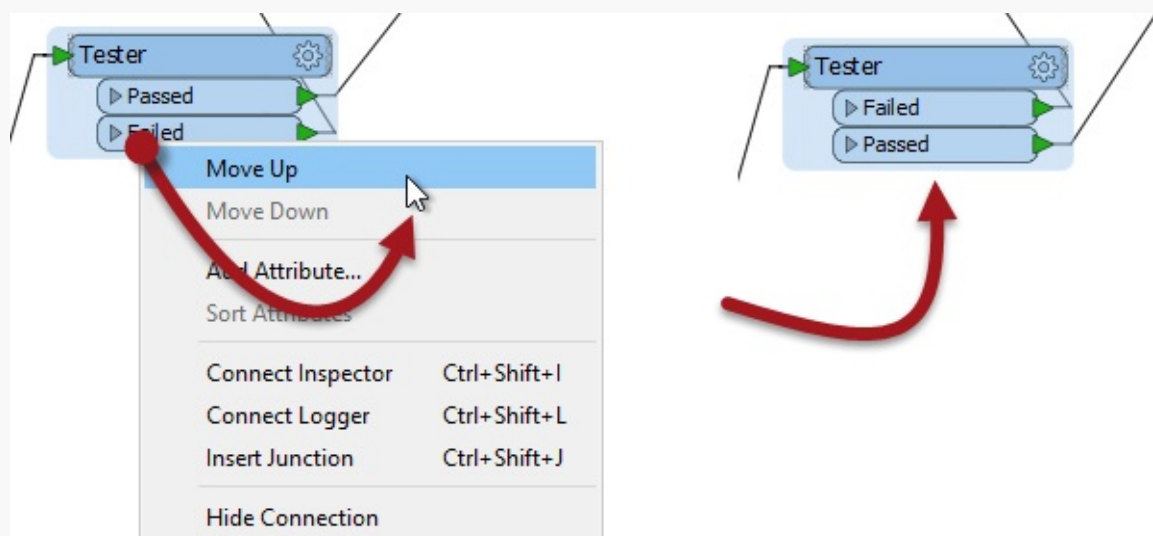
3) Tidy the Workspace

Having experimented with some FME Options, now let's tidy the workspace layout and setup using the FME Style Guide covered in the previous pages.

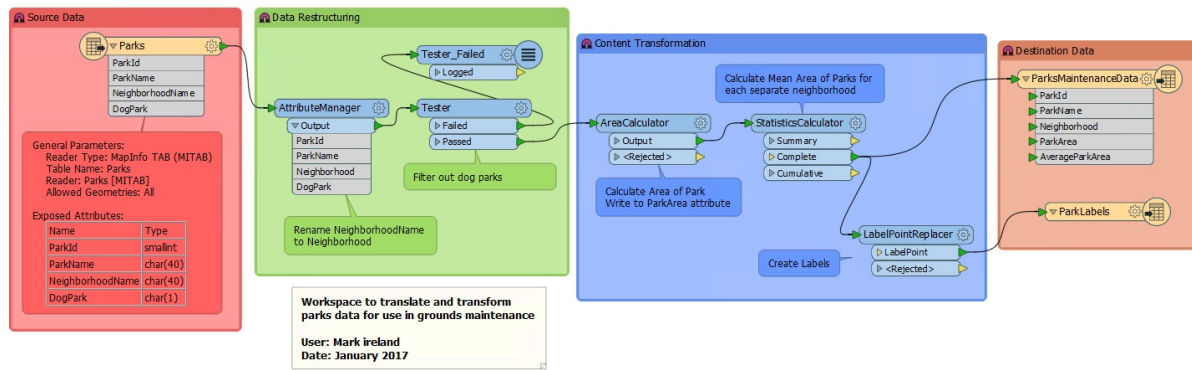
Don't forget to use annotation and bookmarks, so that future users of the workspace will be able to tell at a glance what the workspace is supposed to do. Pick a suitable object layout style that matches the connection style that you will use. Also consider using the Junction transformer to tidy up potentially overlapping connections.

TIP

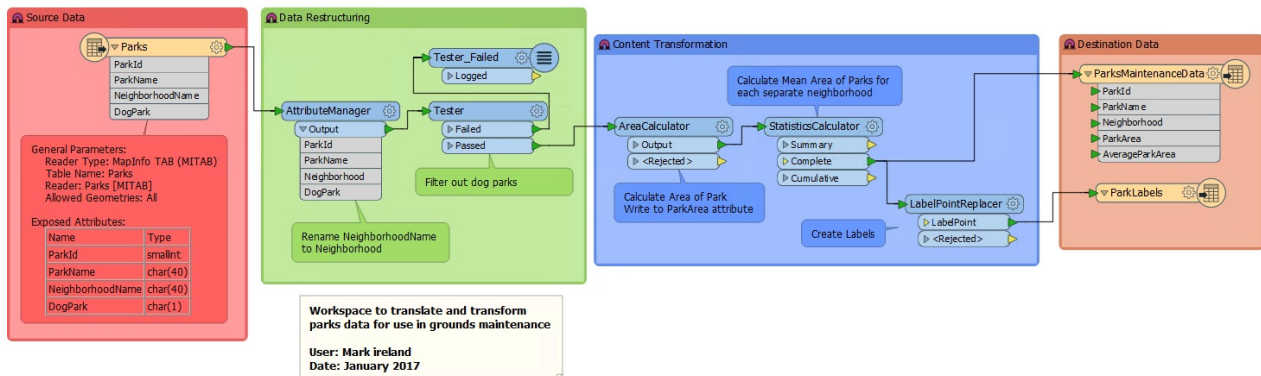
Overlapping connections can also be cleaned up by rearranging the order of transformer output ports:



By the time you have finished, the workspace could look something like this:



Or even this:



Notice a subtle layout change to the LabelPointReplacer to better handle squared style connections.

CONGRATULATIONS

By completing this exercise you have learned how to:

- Use annotations to clarify the processes taking place in a workspace
- Use bookmarks to turn a single workspace into defined sections
- Rename transformers to make their purpose more clear
- Avoid poor design choices like overlapping connections

Methodology

Methodology is all about the tools and processes you use to create an FME project.

One aspect of this is using the tools you have in the way they were intended. If you have experience of software development you might be aware of the concept of **design patterns**. According to Wikipedia:

Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system

Design patterns make for a simplified and efficient development process. Likewise, using Workbench the right way makes for a more productive and efficient FME experience.

Additionally, there are other project management processes that you might want to consider using for FME projects, such as prototyping, versioning, re-usability, and peer reviews.

Prototyping

In the classical sense, prototyping means creating an incomplete application as a way to evaluate the feasibility of a project.

Here we'll stretch the definition to mean how to build a complex FME project incrementally; starting with an empty workspace and building it piece by piece to deliver a result that matches the final specification.

Incremental Development

The key development technique for FME workspaces is [incremental updates](#).

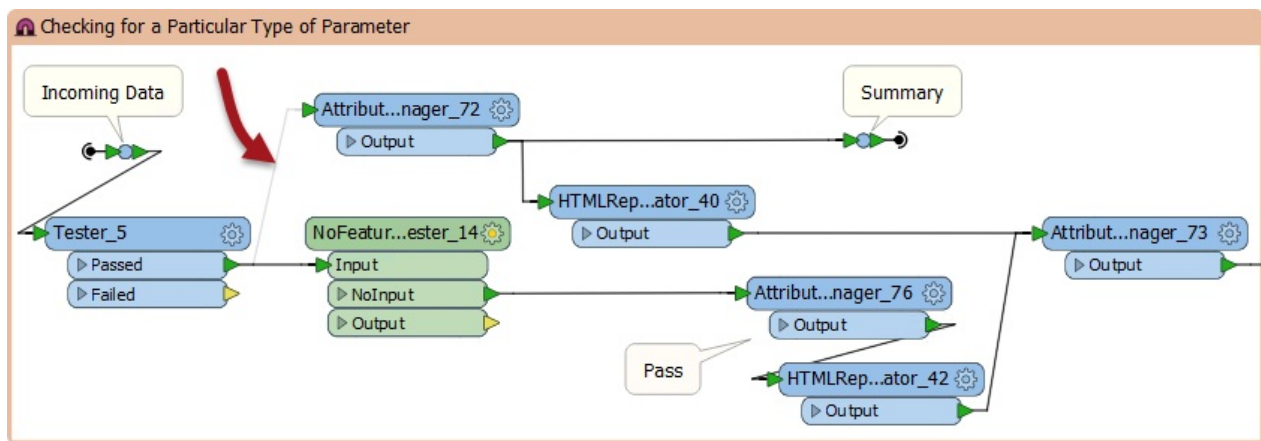
The steps to this technique are:

- Plan your project as a series of small sections, each of which would fit inside a bookmark in FME
 - Design and implement a section in FME Workbench. It should ideally be between 3-10 transformers.
 - Test each section immediately after it is completed. That way you can identify problems at an early stage, and identify them more easily because only a few changes are being made in any increment.
 - Repeat the process, saving the workspace and testing it whenever you've added a new section
-

WARNING

It can be all too easy to start developing a workspace and forget to save it at all! FME keeps a recover file as soon as the workspace is saved for the first time, but until then you are running the risk of an irretrievable loss.

Although a range of 3-10 transformers is an arbitrary number, the more transformers you add the more difficult it would be to identify the source of any problems. Beyond ten transformers is the point at which you should consider chopping that process into smaller sections.



Here a user is testing a newly added section of workspace. By disabling the connection to a parallel section the user is not re-testing part of the workspace that they already know works.

TIP

Another common scenario is to start creating a section of workspace, only to find a limitation or restriction that requires fixing further back in the workspace.

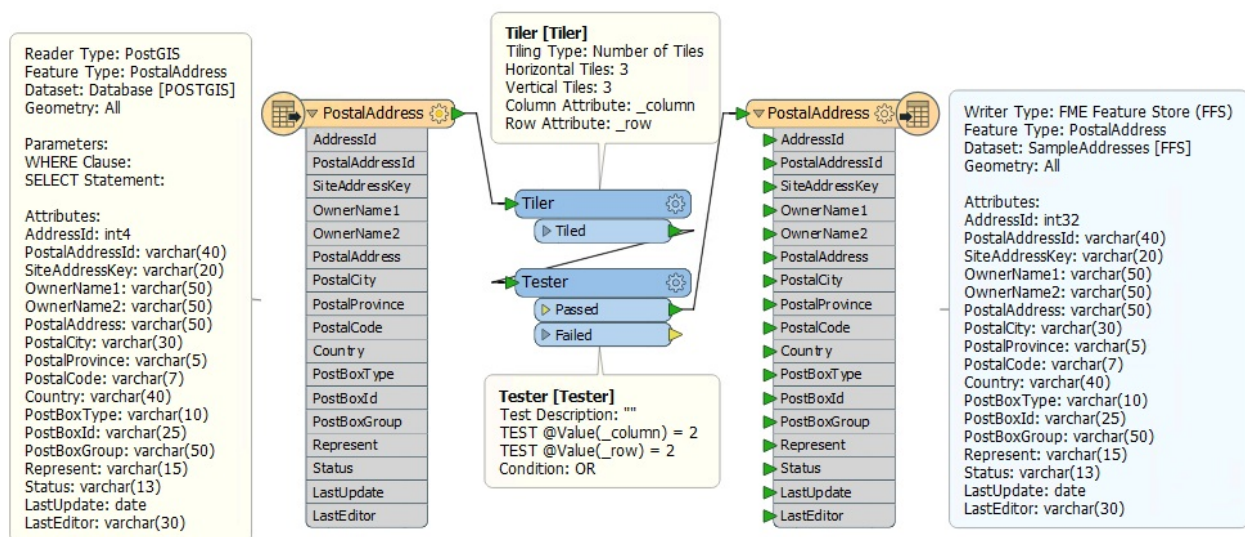
For example, you might find a transformer that you use is not compatible with aggregated data features like multi-polygons, and so you must add a section further back in the workspace to deal with aggregates before they reach this step.

Up until now, exercises in this training have tended to lead you through the required steps to a solution, as if you had perfect knowledge of what is required in advance. From now on exercises will follow the more common approach, where you sometimes must revisit earlier sections that you thought were already complete.

Source Data

When the FME project is large and complex, it's likely that the source data will be large and complex too. Here are some suggestions for how to handle that when developing a project.

- You should - as noted above - create a workspace in small increments, testing it after each section is created. However, you should not be using the entirety of a large source dataset to do so. It's better to use a sample of the source data, preferably a sample that is geographically continuous, and do your testing on that.
- Rather than read a sample of data from the source, it can be better to extract that data into a separate dataset and use that for testing. FME's own format - FFS - is a good choice of format to store this sample dataset. That's because the data is stored using FME's comprehensive data model, and is also in the same structure that FME would use when reading from the actual dataset.
- Databases are a case where extracting a source sample is particularly important. There's no need to be waiting for network traffic and a database response in order to test a small section of workspace.



Here the workspace author is extracting a section of source data by reading from a database, splitting it into tiles, and writing just one tile to the FFS format. This one tile can be used for prototyping a solution in a way that is representative of the entire source database table.

TIP

If you have colleagues who also use FME, why not ask them to review your work? All code that goes into the FME product line is reviewed before it is committed, so there is no reason why your workspaces should not be too.

A reviewer should check to see if there are any mistakes in the workspace logic, if there are any obvious improvements that could be made, or if there are any alternative methods to your approach that would bring performance benefits.

Separate Environments

Large projects are often multi-stage processes; for example Development, Testing, Acceptance, and Production (DTAP).

Ideally, when creating a project, you will have a separate environment for each stage. This includes a different installation of FME, a separate database, and any other aspects of your project.

When transferring an FME project from one environment to another, some things you need to check and possibly adjust are:

- System credentials and accounts for accessing resources such as filesystems and databases
- Environment variables used inside a workspace, for example as source data paths
- URLs that point to web services inside a particular environment
- Database connections that point to a database inside a particular environment
- Custom code (in Python for example) that references resources that are environment dependent

Version Control

The easiest mistake to make with a large FME project is to keep working on a single workspace file. There are two problems with this:

- Faults cannot be easily tracked because there is no record of what has changed and when
- It is not easy to create different versions for different platforms
- If the workspace file is lost or corrupted then the entire project is lost

Therefore, it is better to keep versioned workspaces, where a different copy is kept for each set of revisions.

In fact, it is a good idea to keep and version all materials related to an FME project, including:

- Workspace files
- Python files
- Log Files
- Source Datasets

WARNING

It's better to not store any information that is personal or that includes passwords. Also there's no need to store temporary files.

Reusing Resources

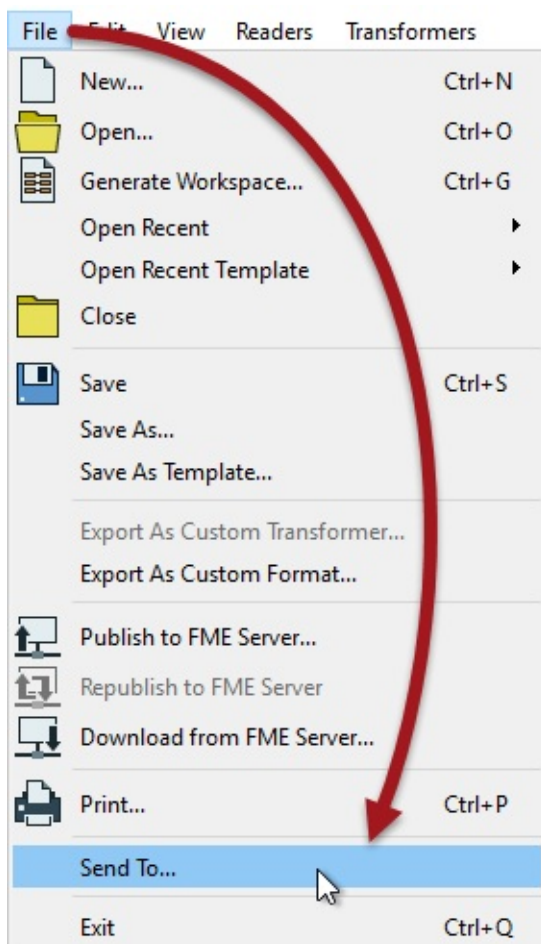
In software development, reusing is the use of an existing component within future product developments.

In FME there are many different components to a translation and so there is a great capacity for reuse. It helps in creating consistent designs among a large group of employees and improves both efficiency and reliability.

Resources that may be shared include workspaces, custom transformers, custom formats, custom coordinate systems, and templates.

Shared Files

The most basic method of sharing is simply giving someone else your workspace file. A function exists on the File menu in Workbench to do just that:



But - as Wikipedia notes - reuse is better when you formalize the practice of reuse by integrating processes and activities into the product lifecycle. To enable this there are other capabilities in FME such as shared folders and templates.

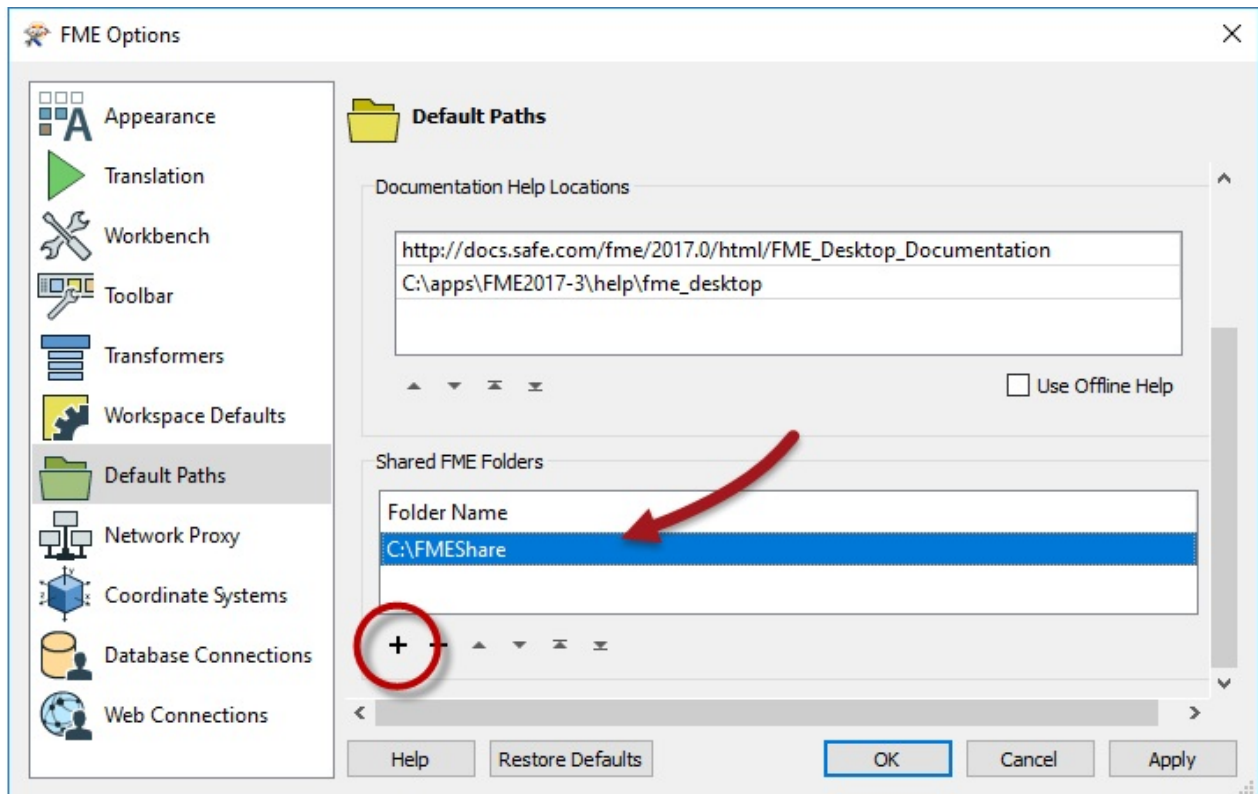
Shared Resource Folders

A more advanced method of sharing is through a Shared Resource Folder. FME is able to identify resources stored in these folders, and use them directly within a translation.

A shared resource folder can be used by just one person, or many.

Using a shared folder is as simple as defining it using Tools > FME Options > Default Paths.

By specifying a location in this option, FME automatically searches for and uses any shared resources that are stored in this folder.



Resources that can be shared in this way include custom coordinate systems, custom transformers, custom FME themes, and workspace template files.

TIP

Use a shared folder on your network as a shared resource folder when you have several FME authors who all need access to the same resources, or just as a location to store your own files.

<user>/<documents>/<FME> is a shared resource folder created and used by default, without having to define it within the options dialog.

Templates

FME Templates are a particularly useful form of reusing resources. Like similarly-named items in other software, FME templates are a means of creating a workspace with/from a pre-designed format and structure.

The closest analogy is to think of templates as a blueprint design.

They have their own extension (*.fmwt) and their own storage folder (<user>/FME/Templates).

Perhaps the most interesting thing about FME templates is that they can include source datasets within the file. This way both a workspace example, and the data required to run it, can be bundled together and provided to another user.

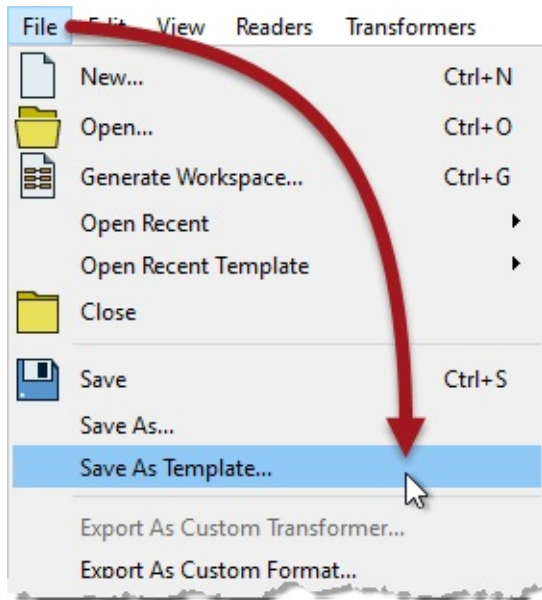
What Are Templates For?

Templates have potentially very many uses. The most obvious ones are:

- Sharing a complete translation
 - Wrapping up source data and workspace inside a single file makes it a very elegant way of providing a complete set of translation files to another user.
 - Sharing pre-defined data
 - When a series of workspaces are all to use the same source or destination data, a template allows the author to duplicate Readers and Writers without having to recreate the workspace each time.
 - Sharing a pre-defined transformation
 - Templates are a great way to store a set of processing tasks for re-use. The end-user can simply create a workspace from the template and add their own readers and writers.
 - Sharing an example translation
 - A template is also a great way to share demos and examples in a way that another user can run immediately.
-

Creating Templates

Templates are created by saving a workspace using File > Save As Template on the menubar:



Miss Vector says...

Templates are very useful functionality. You really should be aware of how to create and use them.

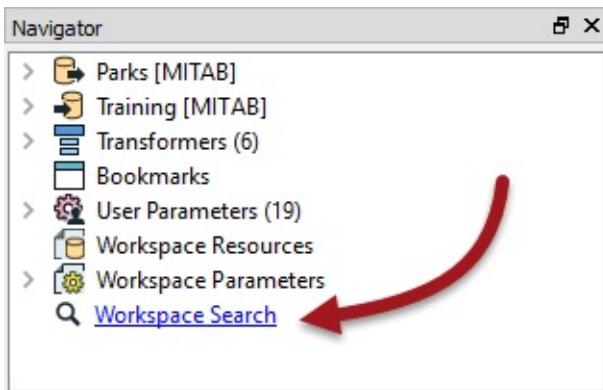
What ways can a template be opened for use in FME Workbench (there might be more than one correct answer):

- 1. Open it using File > Open*
- 2. Open it using File > Open Recent Template*
- 3. Double-click the fmwt file in Windows Explorer*
- 4. Use Create Workspace from Template in the Getting Started part of the start tab*

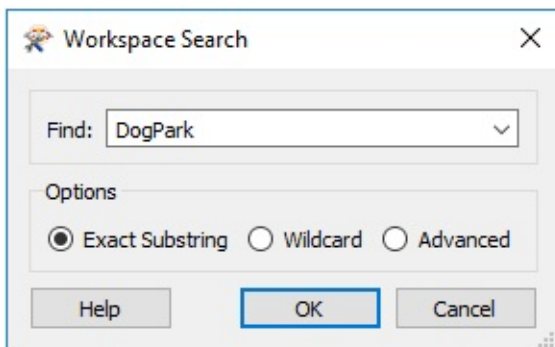
Workspace Searching

FME Workbench includes a search tool for locating objects in a workspace according to a set of search text. This is useful in two scenarios: a) editing large workspaces and b) debugging

The workspace search function is accessed through a hyperlink at the foot of the Navigator window:



Clicking this link opens up a text dialog in which to enter search terms. The results of the search include any attribute names, feature types, transformers, parameter names, and parameter values that include the search term entered.

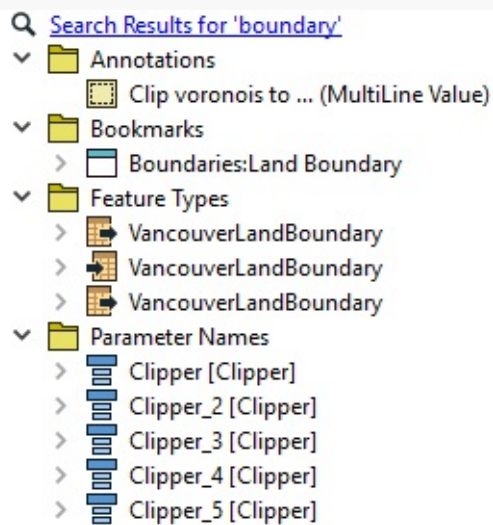


Searching is particularly useful when you wish to make an edit in a large workspace, especially when the same component (e.g. transformer) is used throughout the workspace. For example, you wish to find where all instances of an AttributeManager transformer are used.

It is also useful in debugging when you wish to find where a particular item is set or altered. For example, you might search for a particular attribute that has a different result in the output to what was expected. It might be that it is being set in an unexpected place in the workspace.

Chef Bimm says...

What I especially like about the search tool is that it returns bookmarks and annotations too. Delicious!



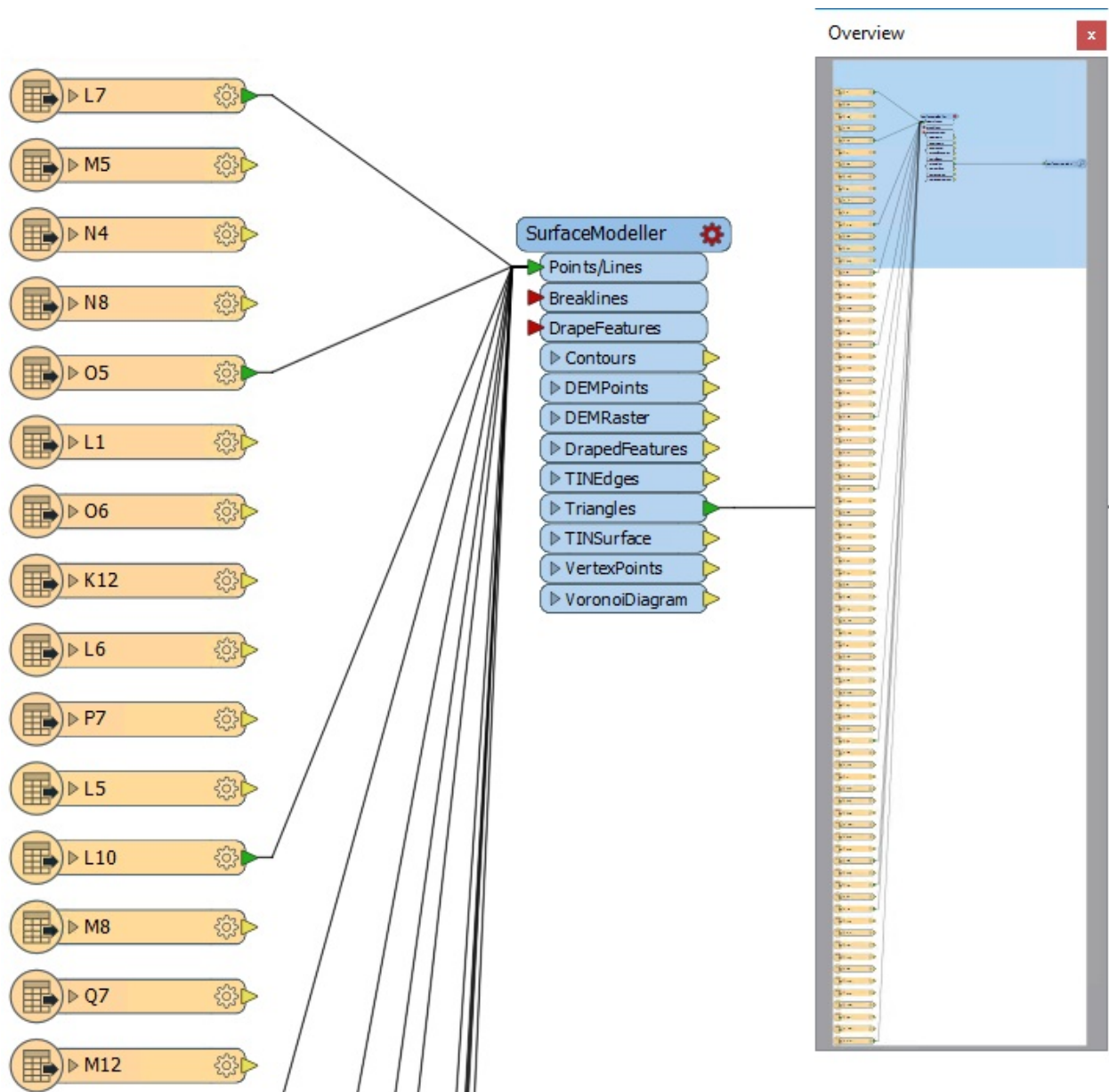
Format Best Practices

Format best practices involve setting up readers and writers both quickly and without cluttering the canvas.

Minimizing Feature Type Objects

A schema definition may contain many, many feature types. If these are all chosen for translation then the workspace can become extremely unwieldy and poorly laid out.

Here a user has 80 feature types, only a handful of which are being used:



Remember, when creating a workspace you'll be prompted which feature types to add to the translation. Don't add feature types you don't need, as it will cause cluttering of the canvas.

TIP

An excess of attributes can also cause workspace clutter. Don't forget you can remove attributes on the reader feature type and make the workspace clearer and tidier than if the entire set of attributes was still exposed.

Database Connections

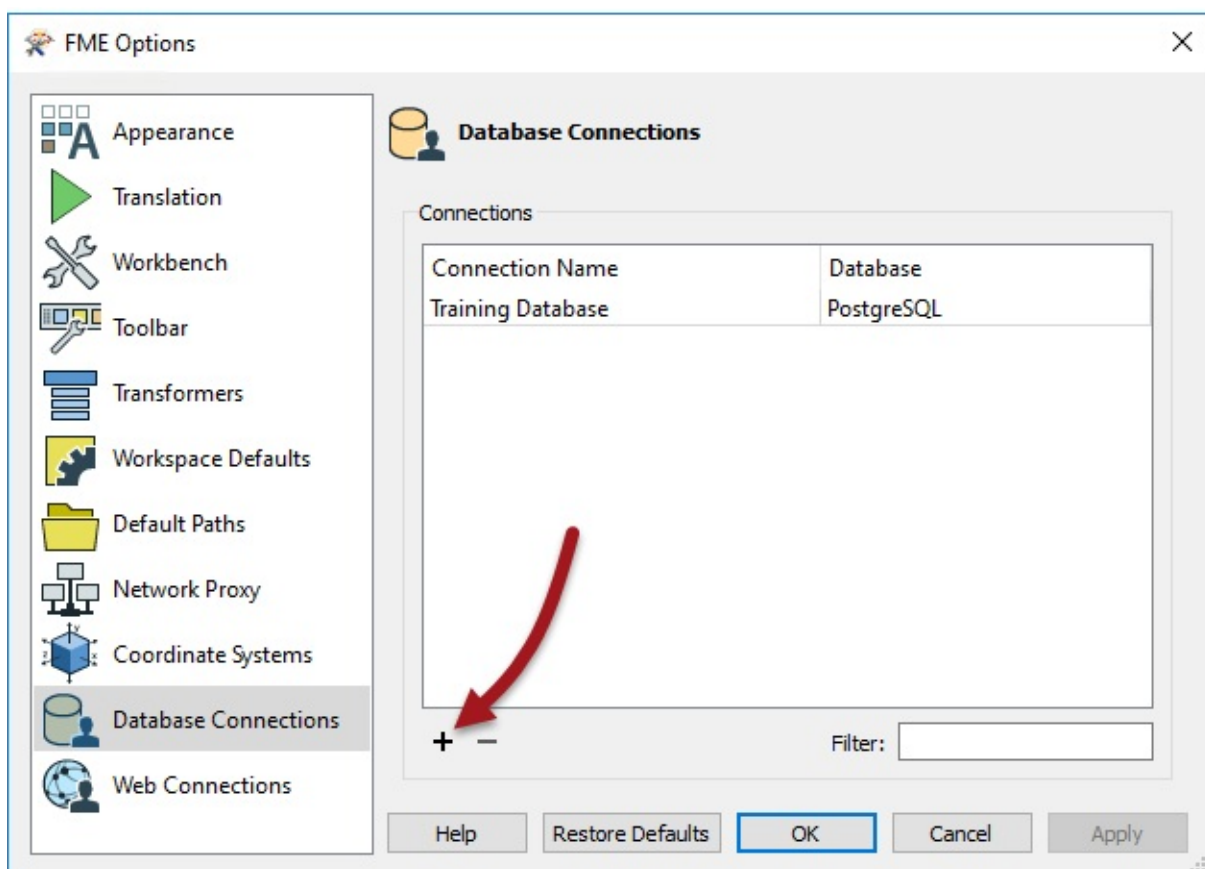
Database formats are an example of a repeating pattern, where the same connection and authentication parameters need to be entered whenever a database reader or writer is added, or a connection is required inside a transformer.

To simplify this process, use a tool called Database Connections.

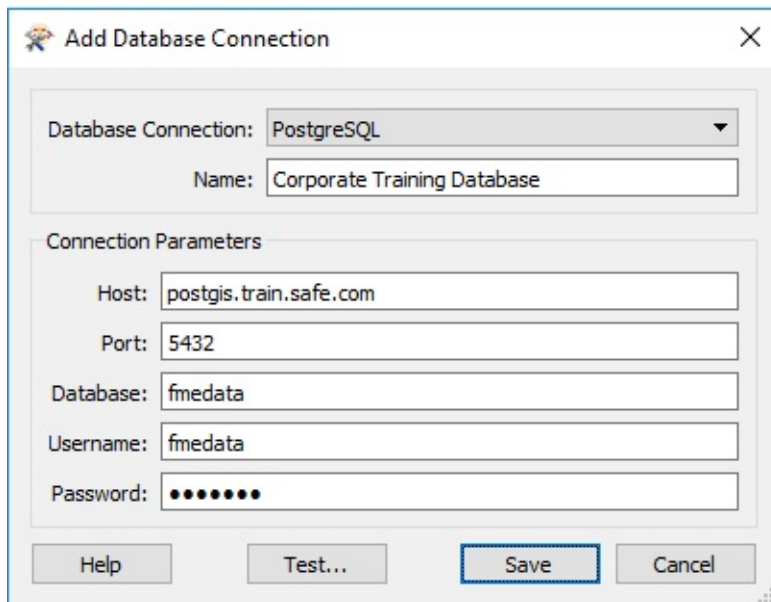
Database Connections are a pre-defined set of connection and authentication parameters, stored under a single name. Once created, all FME readers and writers (and transformers) will accept a connection name in lieu of the actual parameters.

Creating a Database Connection

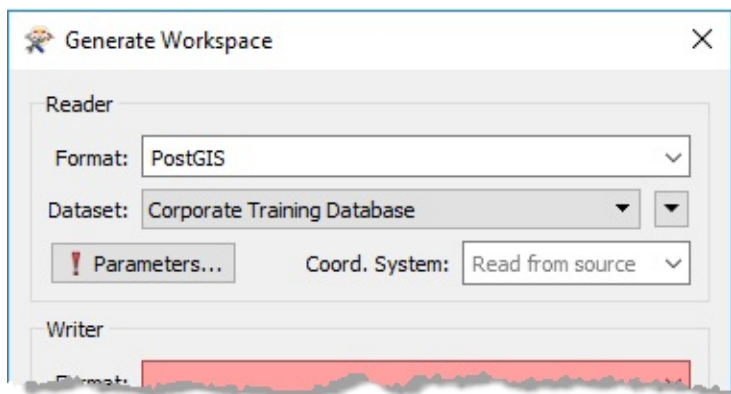
To create a Database Connection first select Tools > FME Options from the menubar and then click on the Database Connections icon. A new connection is created by clicking the + button:



A new connection is defined by selecting the database type, entering connection parameters, and giving the connection a name:



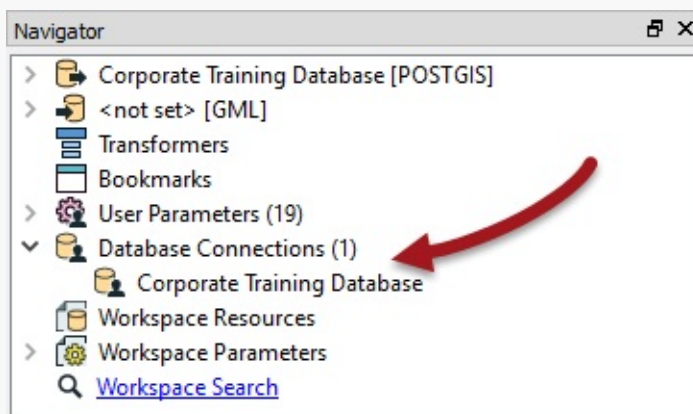
Now, whenever a workspace involves a database-related Reader/Writer, it's possible to simply use this connection instead of re-entering all the usual parameters:



The benefit here is not only efficient workspace authoring, there is also a security benefit. If you embed the connection parameters (i.e. enter them for each reader/writer) then they are stored inside the workspace file. This could be a security risk if you send the workspace to someone. But FME Database Connections are stored securely on your system, outside of the workspace, and can never be accessed by someone else.

TIP

Database connections are also listed in the Navigator window once they are used:



Right-clicking a connection here also displays a context-menu with the ability to edit, test, or embed that connection; and to show where in the workspace it is used.

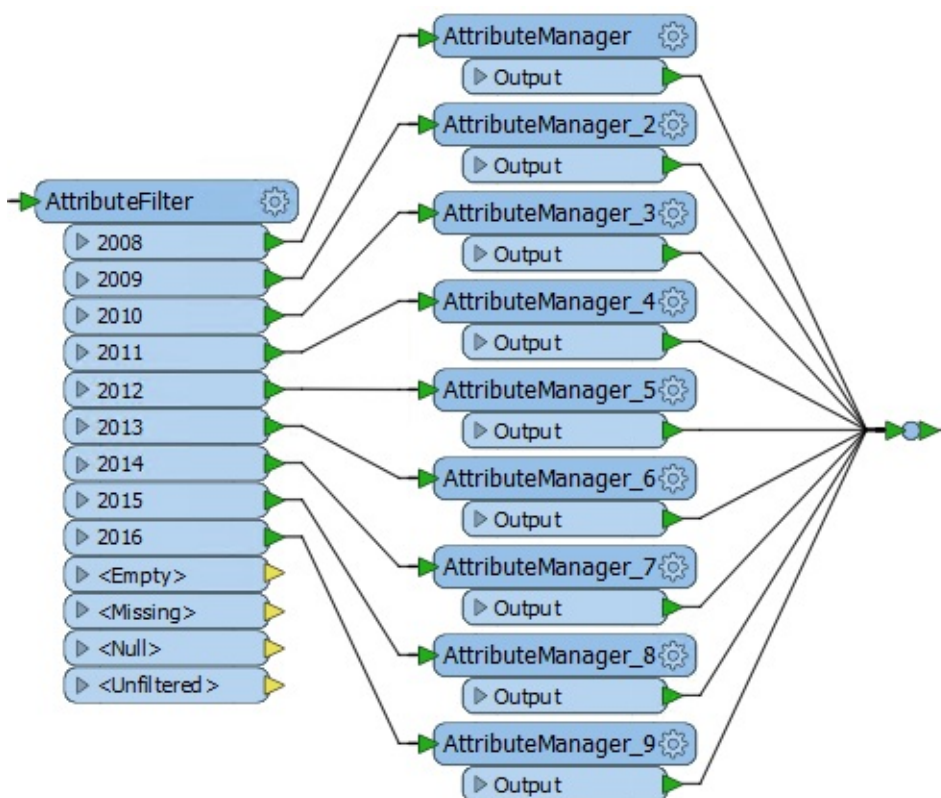
Transformer Best Practices

With FME's huge number of transformers it's very difficult to say exactly which transformers to use and when. However, it's much easier to identify when something is wrong, with design weaknesses that may lead to future difficulties.

Duplicated Transformers

If a workspace duplicates the same transformer again and again – maybe creating multiple streams to do so – then it is probable that the workspace has been designed very inefficiently.

The multiple attribute transformers here indicate there is a problem.



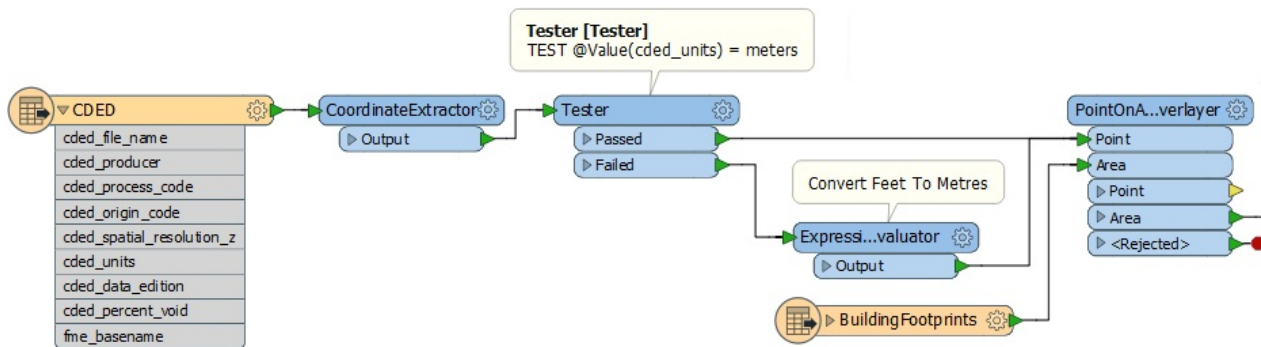
Here a single AttributeValueMapper transformer could be used to replace all of the above transformers (both AttributeFilter and AttributeManagers).

Error Trapping

Error trapping is a way to design a workspace such that unexpected data does not cause the workspace to fail. Basically you try to foresee the sort of data problems that might arise and build in methods to handle them.

Error trapping can be as simple as adding a test or filter transformer to weed out bad features, or it can be more complex and include ways to process data in different ways depending on different circumstances.

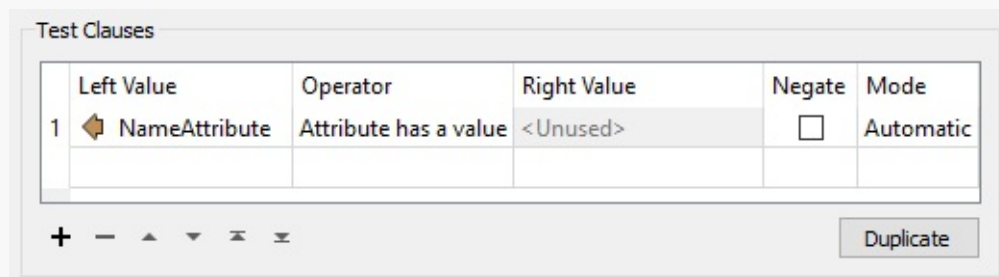
Here a user is reading a surface model and assigning Z values to building footprints. The important part is that they are testing what units the dataset is in (cded_units = meters|feet) and converting where the units are not what they want:



This workspace can be depended on to produce the correct results, no matter whether the source data is in metres or feet.

TIP

The Tester transformer has an operator for testing whether an attribute has a value:



This is very useful for error trapping, to test whether an attribute has a value before trying to use it as the source for a parameter.

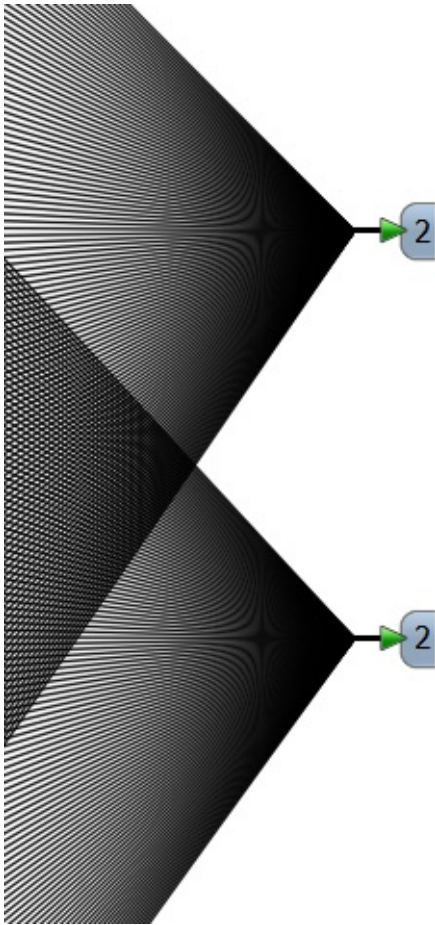
Complexity

A workspace does not need to be complex to be a good project; it can carry out a complex process in a simple way.

These are clues that there might be a problem:

- **Low Level Complexity:** When a workspace author uses FME functions and factories inside a workspace
- **Excess Scripting:** When a workspace contains more scripting (in Python or Tcl) than it does FME transformers
- **Connection Density:** When workspace connections are so dense they form a moiré pattern!
- **Multiple Workspaces:** This workspace calls that one, which calls this one, which runs Python to call that one...

The connections in this workspace are so dense they could form a black hole! If your workspace looks like this, it might be time to look at how it can be improved:



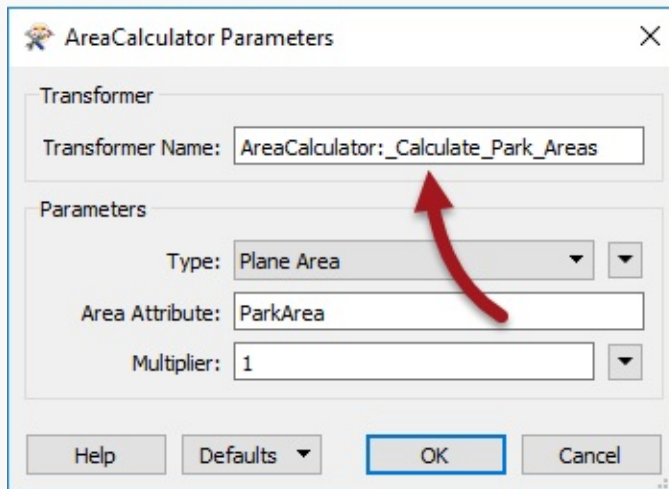
Firefighter Mapp says...

It's better to be safe than sorry. You don't want to get burned by bad designs. If you're unsure about a workspace, consult [other FME users](#), or contact the [Safe Software support team](#) for advice.

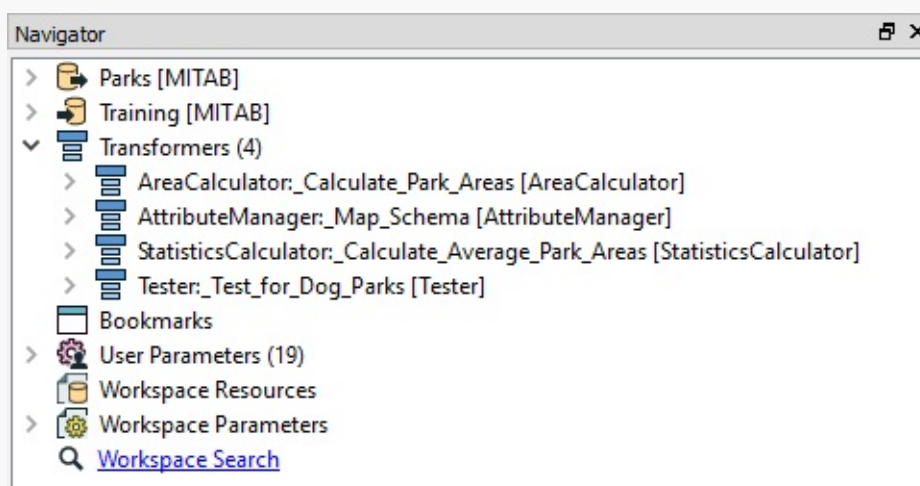
Also, be sure to check out this [blog post on FME and Code Smells](#).

TIP

Did you notice that the Properties dialog for each transformer contains a field where the transformer may be renamed to a more meaningful title?



This can help identify which transformer carries out which action, both on the canvas and in the navigator window:



Users who wish to keep the original name on the canvas often use annotation instead, or use the transformer type as a prefix in any new name.

Exercise 2 Design Patterns	
Data	Orthophoto images (GeoTIFF), Roads (AutoCAD DWG)
Overall Goal	Create a workspace to overlay road features onto raster images
Demonstrates	Methodology Best Practice
Start Workspace	None
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex2-Complete.fmw

A mapping project in the city requires a dataset of roads laid down on a raster background. The data should cover the neighborhoods of Downtown and the West End, and the road features should only be of the types Arterial, Residential, Secondary, and Other. Each type of road feature should be depicted differently in some way, so as to make their type visually recognizable.

As the resident spatial data expert, you've been asked to create this dataset. The source data consists of a lot of files and layers, and the FME workspace will include a number of operations that might be carried out by different transformers; but luckily you've just learned all about proper methodology in FME and are looking forward to applying that knowledge on this project.

1) Make an Initial Assessment

Let's first consider the steps we know are needed for the project.

- You need to read a set of GeoTIFF files
 - There are a lot of files so while prototyping the workspace you may wish to cut back on the amount of data being used to test the solution.
 - The area of the project is not the entire city (just two neighborhoods) so you will have to find a way to avoid reading excess files in the final solution.
- You need to read an AutoCAD DWG file of roads
 - The project requires only four road types, so not all source data layers are required
 - Again, you will need to cut the data back to the required area
- You need to differentiate road types in some way
 - The most obvious visual difference would be colour, but size might also work. The transformer to use will differ depending on what technique you decide to use.
- You need to overlay the road features on to the raster data
 - This is not something you've had to do before, but you know there must be a transformer to do so!

2) Inspect Source Data

Use a file browser to explore the folder C:\FMEDData2017\Data\Orthophotos. This is where the raster data is kept. The name of each file denotes a tilename in a grid of tiles, but Unless you know what that grid is you won't be able to select a single file that you know covers the Downtown or West End areas.

So, open all the raster tiles in the FME Data Inspector.

TIP

Here's a free tip for you: when opening the raster data, check the parameters dialog and set Feature Type Names to come From File Name. That way you'll be able to see each file separately, not all joined together.

Pick a single raster tile that appears in the West End or Downtown areas of the city. File 08_09_NO would be a good choice, but you can use whatever file you think correct.

3) Add GeoTIFF Reader to Workbench

Start FME Workbench and begin the project by adding a reader to read the chosen GeoTIFF file. Do this by selecting Readers > Add Reader from the menubar. The parameters are as follows:

Reader Format	GeoTIFF (Geo-referenced Tagged Image File Format)
Reader Dataset	C:\FMEData2017\Data\Orthophotos\99-99-XX.tif

Your workspace will now look like this:



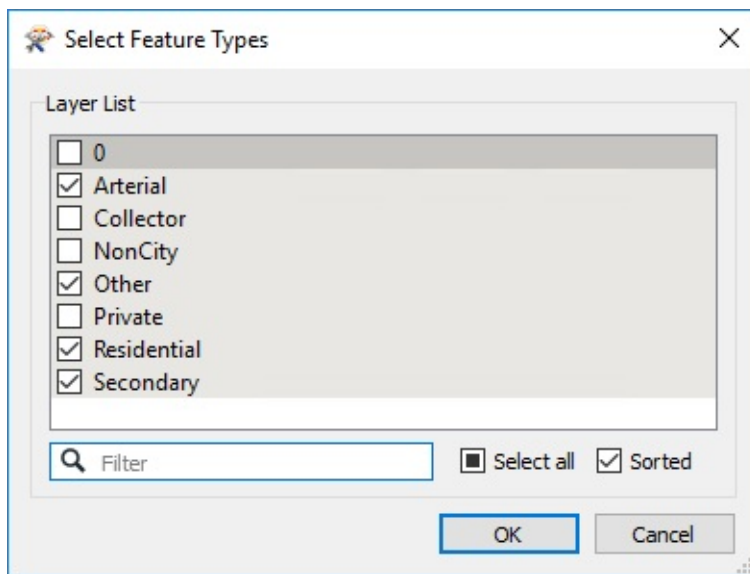
A simple start, but a good one.

4) Add AutoCAD Reader to Workbench

Now add a reader (Readers > Add Reader) to read the roads data as follows:

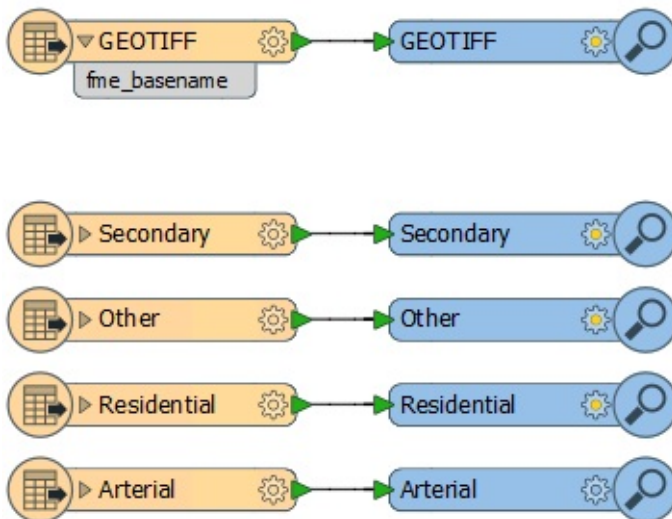
Reader Format	Autodesk AutoCAD DWG/DXF
Reader Dataset	C:\FMEData2017\Data\Transportation\CompleteRoads.dwg

You will be prompted which feature types (layers) you wish to read from the data. Choose only the required types (Arterial, Other, Residential, Secondary)



5) Test Workspace

Add an Inspector transformer to each of the source feature types. You can do that by selecting all the objects, right-clicking, and choosing Connect Inspectors. The workspace will look like this:



Now run the workspace and inspect the output.



Dr. Workbench says...

If you get a message pop-up (or appear in the log window) about features not matching, don't worry. It isn't an error message. It's a warning that you left layers from the DWG dataset out of the workspace. If you haven't covered this message so far in the training, you'll probably cover it very shortly.

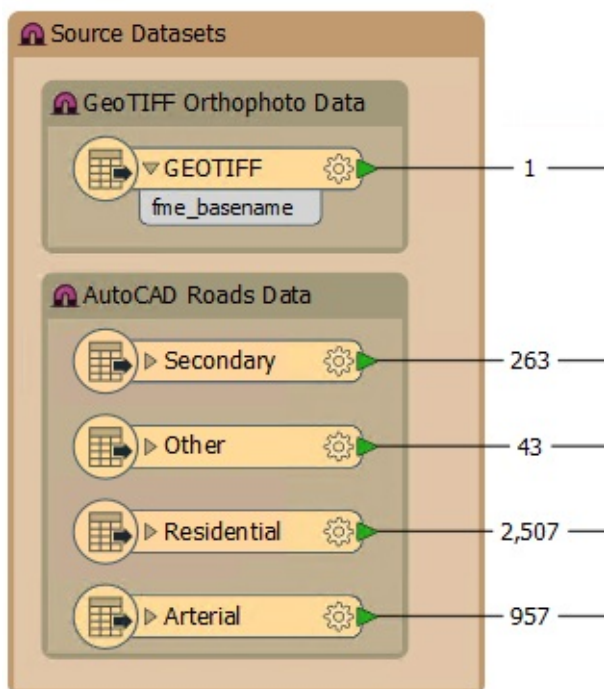
Although this is an early stage, the output is still important. It shows that we are reading data correctly, that we have only the required road layers, and that we have a raster dataset that overlaps the roads. The latter point is especially important. It shows that we aren't trying to overlap two datasets in different coordinate systems.

Additionally, by querying a road feature we can see that it is still vector data, and isn't yet being overlaid onto the raster as we need.

6) Add Style

Now that we have the first section set up, we can mark it up with a bookmark. So, add a bookmark around the reader feature type objects and give it a suitable name.

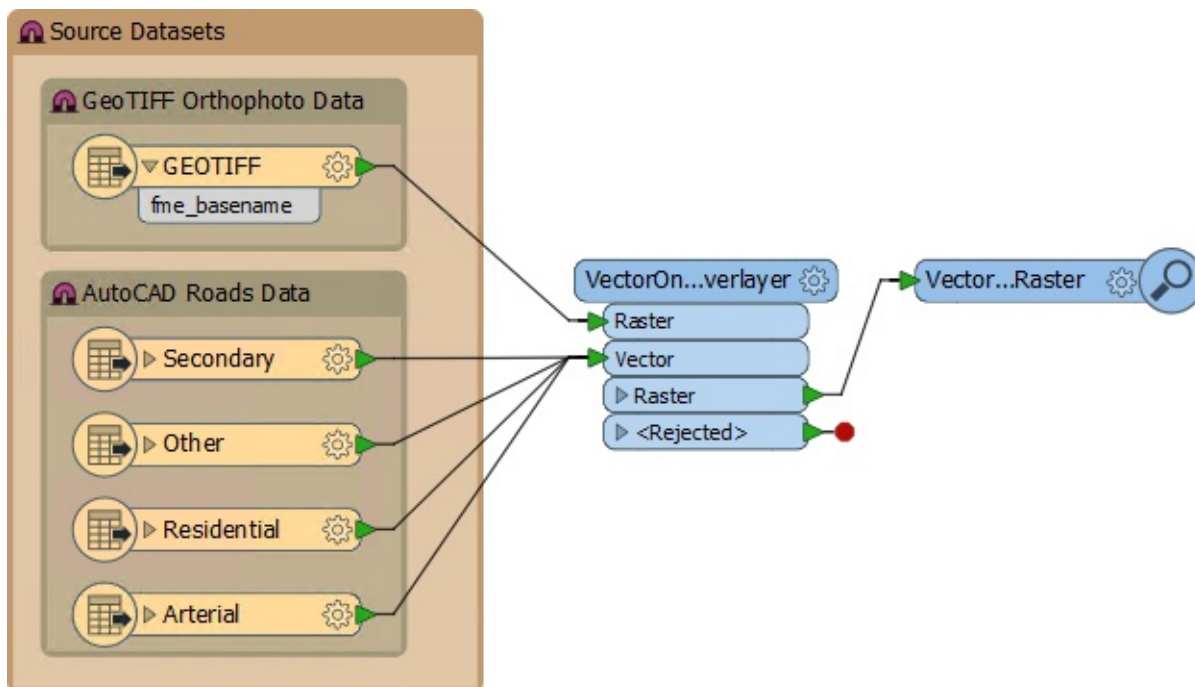
You may even create nested bookmarks to denote which object is from which reader:



7) Add VectorOnRasterOverlayer Transformer

The next logical step is to try and overlay the vector data onto the raster. So, add a VectorOnRasterOverlayer transformer to the workspace.

Arrange the feature types or transformer ports so as to avoid overlapping connections. Add an Inspector transformer:



Now run the workspace. Zoom in to the overlaid data and you will see that the road features are now imprinted into the raster data as we require:



So now that transformer is tested and proved to work as expected.

The really useful aspect is that vector data outside of the raster area was simply discarded. This means we don't need to set up a separate clip operation on the vector data; the VectorOnRasterOverlayer will clip it for us.

8) Add Style

The VectorOnRasterOverlayer is just a single transformer, so it doesn't really need a bookmark around it (not for Sectioning the workspace, and the project is not going to be big enough to bookmark a single transformer for Access or Editing purposes).

However, you should probably add an annotation to it, to denote what is going on.

9) Add Vector Symbology

Although the vector data has been overlaid correctly, we haven't yet symbolized it in such a way that each data type is distinguishable from the other.

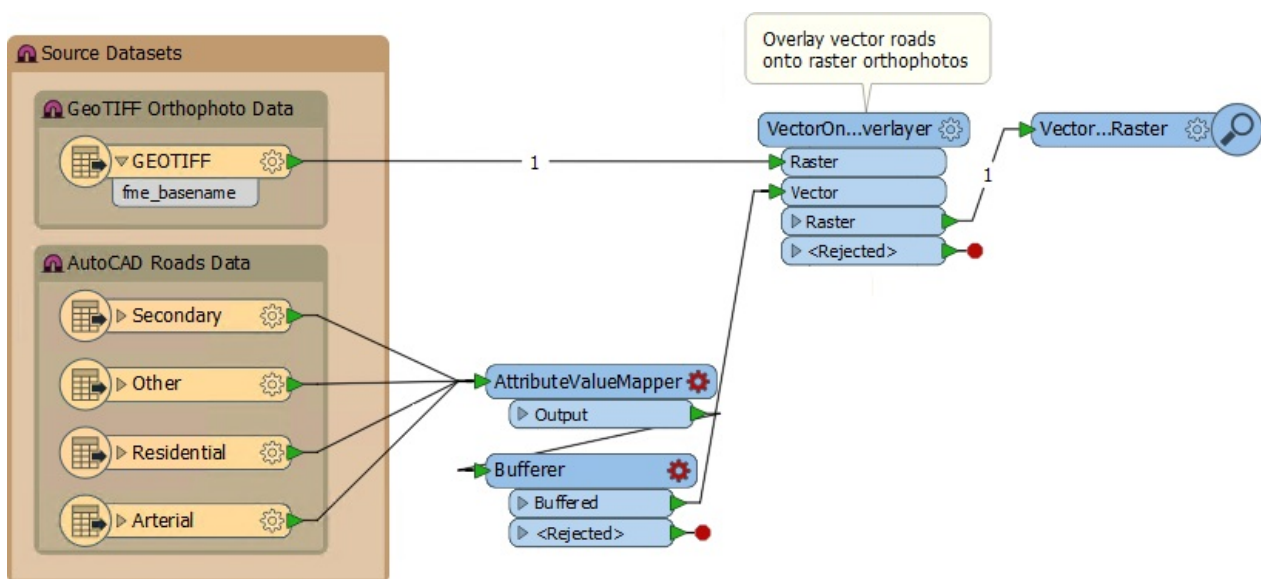
Let's do this on the basis of size. We can create a buffer around the roads where the buffer size is dependent on the type of road:

Road Type	Buffer Size
Residential	2
Other	5
Arterial	5
Secondary	3

The *simplest* way is to put a Bufferer transformer for each road layer/type; but the simplest way is not necessarily the best. Here it would mean duplicating the Bufferer transformer multiple times.

What we'll do is set the buffer size as an attribute depending on the road type, then use that attribute in the Bufferer transformer parameters.

So, add an AttributeValueMapper transformer, followed by a Bufferer transformer. All road types should connect to the AttributeValueMapper and the Bufferer output will connect to the VectorOnRasterOverlayer:Vector input port:

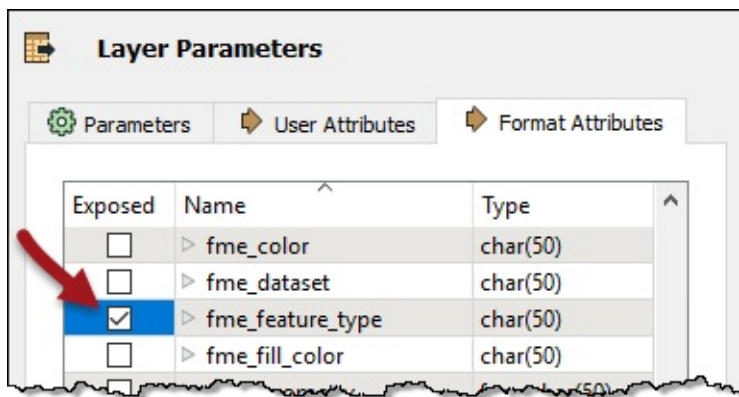


10) Set Parameters

If you view the parameters for the AttributeValueMapper (either in the Parameter Editor window or the transformer's own parameters dialog) you'll see the Source Attribute parameter cannot be set because we don't have any attributes. What we need is an attribute that tells us what the original layer was, and that is an **FME attribute** called *fme_feature_type*.

This is a good example of having to go back and change something - in this case a reader - to help in a later operation.

So, click on one of the Road feature type objects and in the Parameter Editor window click the Format Attributes tab. Put a check mark next to the attribute *fme_feature_type*:



Click Apply. Now check the AttributeValueMapper parameters again. Set:

Source Attribute	fme_feature_type
Destination Attribute	RoadSize

And then in the mapping fields set:

Source Value	Destination Value
Residential	2
Other	5
Arterial	5
Secondary	3

Transformer

Transformer Name:

Attribute Selection

Source Attribute:

Destination Attribute:

Default Value:

Value Map

Mapping Direction:

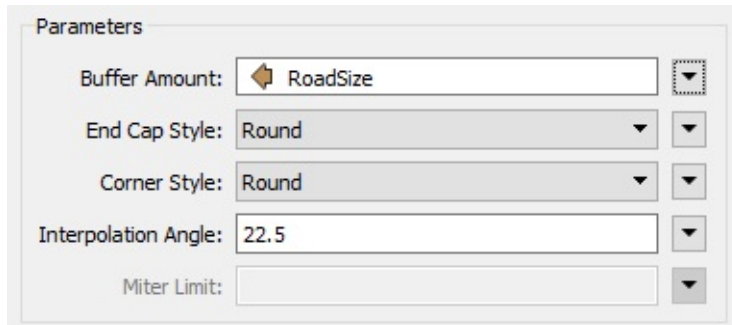
Source Value	Destination Value
<input type="checkbox"/> Residential	<input type="checkbox"/> 2
<input type="checkbox"/> Other	<input type="checkbox"/> 5
<input type="checkbox"/> Arterial	<input type="checkbox"/> 5
<input type="checkbox"/> Secondary	<input type="checkbox"/> 3

+ - ▲ ▼ ↕ ⇄


Then click Apply/OK to apply these changes.


11) Set Parameters


Now view the Bufferer parameters. This is a lot easier. Simply set the Buffer Amount parameter to the value of the RoadSize attribute:





Parameters

Buffer Amount: 

End Cap Style: 

Corner Style: 

Interpolation Angle: 

Miter Limit: 

Again click Apply/OK to confirm the parameter changes.

12) Add Style

You know the drill! Apply some style to the workspace by adding a bookmark around the two newly added transformers!

13) Run Workspace

Save and run the workspace. You now have the results you are looking for:



14) Apply Full Dataset

So, our prototype is now complete. All we need do is apply it to all of the tiles in the two suggested neighborhoods.

There are multiple ways of doing this, some more complex than we want to cover here (using the FeatureReader transformer, if you must know)! For this exercise, we'll do something very simple.

In the FME Data Inspector ensure you have background maps turned on. Now open all of the orthophoto images (you may still have them all open from step 2). Then add the following dataset to the same view:

Reader Format	Google KML
Reader Dataset	C:\FMEDData2017\Data\Boundaries\VancouverNeighborhoods.kml

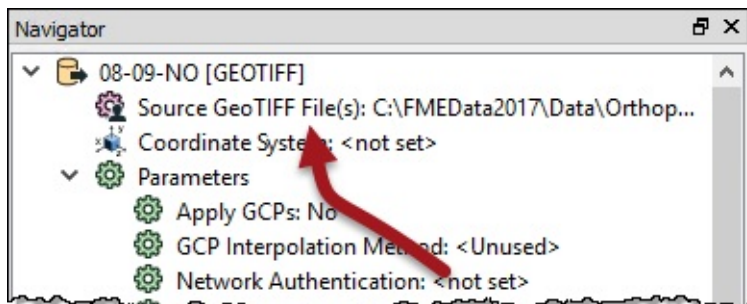
***NB:** You need to have the background maps turned on because these datasets are in different coordinate systems and would otherwise not appear properly georeferenced.*

From this you ought to be able to identify which tiles overlap the West End and Downtown neighborhoods. Make a list of them, or you can use this list:

- 04_05_LM
- 04_05_NO
- 04_05_PQ
- 06_07_LM
- 06_07_NO
- 06_07_PQ
- 08_09_LM
- 08_09_NO
- 08_09_PQ
- 10_11_LM
- 10_11_NO
- 10_11_PQ

15) Set Source Datasets

Back in FME Workbench look in the Navigator window under the GEOTIFF reader for the parameter *Source GeoTIFF File(s)*:



Double-click the parameter and then use the browse button to select the files in the above list. Click OK to accept the changes.

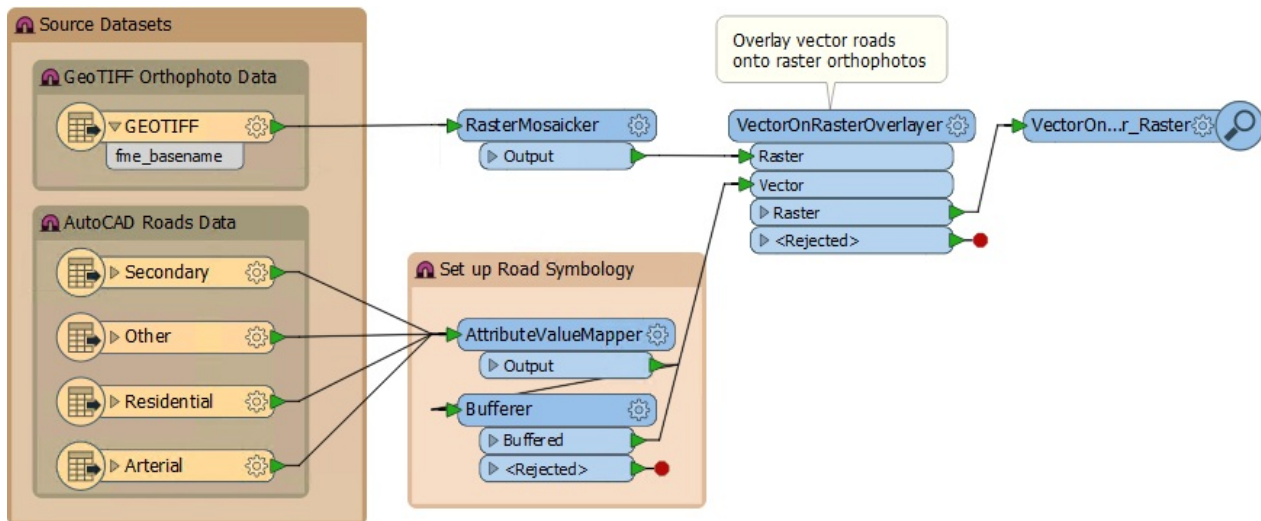
16) Mosaic Data

Save and re-run the workspace. You will find it fails with an error, like so:

```
VectorOnRasterOverlayer(VectorToRasterFactory): Too many raster
features input.
Each group may only have a single background raster
```

The problem is that the VectorOnRasterOverlayer only permits one raster feature for each set of vectors. What we need to do is join together all the input tiles (mosaic them) into one feature.

So, add a RasterMosaicker transformer before the VectorOnRasterOverlayer:



This is another good example of having to go back and change something - in this case add a new transformer - to help in a later operation.

Now re-run the workspace. The result will look like this:



17) Test Workspace

Email your completed workspace to BestPractice@safe.com. In reply you will receive a report that analyzes your workspace for best practice techniques. If necessary, update the workspace to fix any errors or warnings that are included in the report.

CONGRATULATIONS

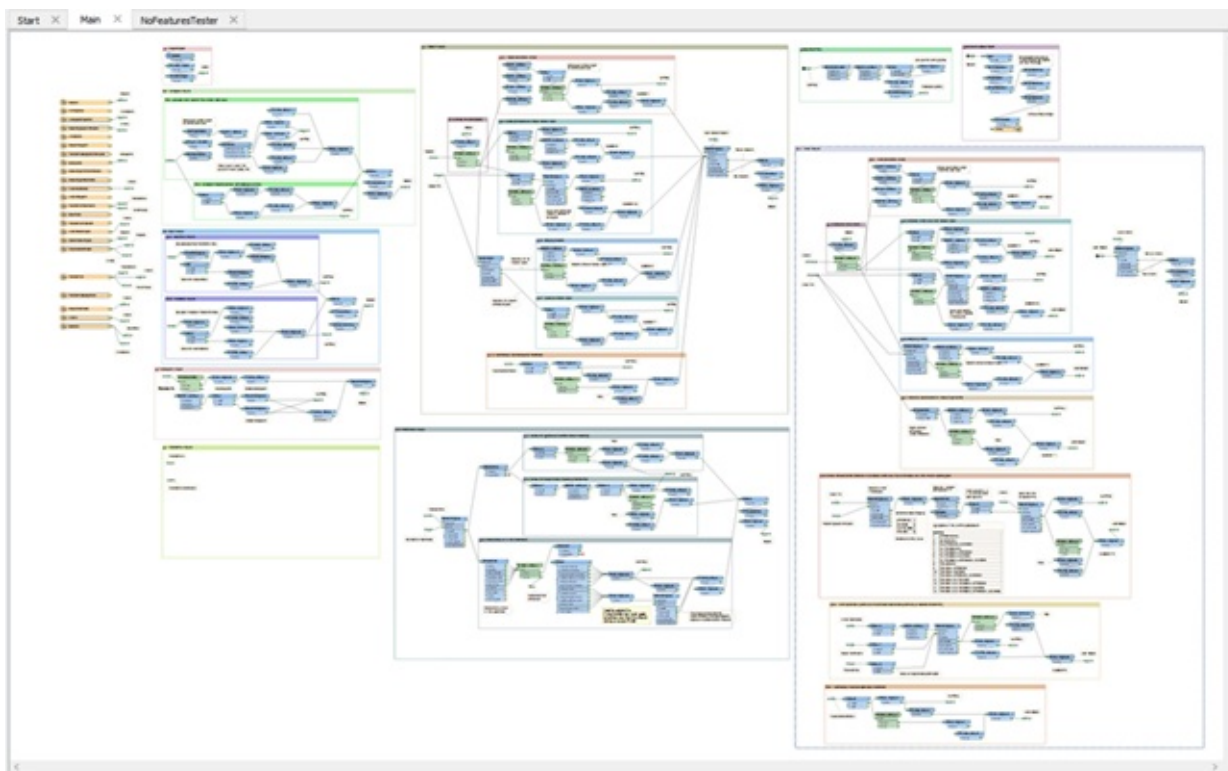
By completing this exercise you have learned how to:

- *Assess the steps required to carry out a particular project*
- *Create a workspace incrementally, applying style and testing it after each increment*
- *Select and use a sample set of data for prototyping a workspace*
- *Handle data with excessive layers by leaving them out of the workspace*
- *Overlay vector data onto raster (not the main point of the exercise, but useful to know)*
- *Set up a transformer parameter by creating a mapped attribute value*
- *Apply a prototype workspace to a full dataset*
- *Backtrack to an earlier section of workspace to solve a problem that occurs in a later section*

Debugging

Even skilled FME users seldom produce correct results with the first version of a new workspace.

In the event of an error message or unexpected output, a user must 'debug' the workspace to find what error has been introduced into the workflow definition. And when the workspace is very large, like this:



...it's very important to be aware of the debugging techniques available in FME.

Debugging Order

It's important to use the various debugging steps in the correct order. For example, it's not particularly useful to check the output for problems (and edit your workspace to track them down) when a warning message in the log window already highlights and explains the issue.

Generally you would first look for signs of problems and then attempt to track down the issues that caused them. Therefore a logical order would be:

- Set the log file parameters before running the translation
- Run the translation
- Examine the log for warnings and errors
- Inspect the output datasets, if the translation wrote any
- Check feature counts in the Workbench canvas to locate where problems occurred
- Check reader, writer, or transformer parameters at the point of failure
- Test the workspace by adding Inspector transformers to view data at different points
- If necessary, run the workspace in feature debug mode

However, sometimes problems in the output are not highlighted by errors or warnings in the log - for example those caused by misuse of parameters or incorrect connections. Therefore you might have to revisit the log and/or feature counts after inspecting the output.

Logging

When you run a workspace FME writes to both the FME Workbench log window and a log file.

These logs contain a record of all stages and processes within a translation. The contents may appear complex, but such information is vital for interpreting a workspace's actions.

Sister Intuitive says...

*Why are logging parameters the first part of debugging? Because if you don't know where the log file is - or haven't set it up correctly **before** running the workspace - it will not be helpful in debugging any issues that arise!*

Log File Location

The log file is (by default) written to the same folder as the workspace with the name `<workspace name>.log`. However, it can also be set using a parameter in the Navigator window in Workbench.

Message Types

There are a number of different message types that show in the log window including:

Error: An error, denoted in the log by the term **ERROR**, indicates that a problem has caused FME to terminate processing. For example, FME is unable to write the output dataset because of incorrect user permissions.

Warning: A warning, denoted by the term **WARN**, indicates a processing problem. The problem is sufficiently minor to allow FME to complete the translation, but the output may be adversely affected and should be checked. For example, FME is unable to write features because their geometry is incompatible with the Writer format. The features will be dropped from the translation and a warning issued in the log.

Information: Information messages, denoted by the term **INFORM**, indicate a piece of information that may help a user determine whether their translation has been processed correctly. For example, FME sometimes log confirmation of a particular dataset parameter, such as coordinate system.

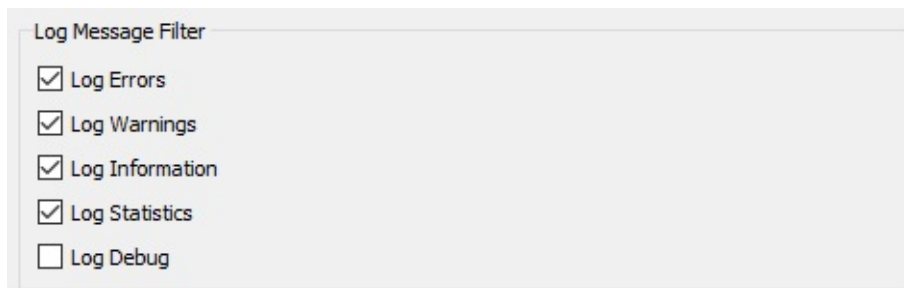
Statistics: Statistics messages, denoted by the term **STATS**, provide information on various numbers relating to the translation; for example the number of features read from a source dataset, and the time taken to do so.

Sister Intuitive says...

You can search for different types of messages by clicking in the log window and pressing Ctrl+F to open a search dialog.

Log Options

Workbench has the ability to filter different message types from the log window. This is done using Tools > FME Options > Translation.



Turning off STATS and INFORM log messages helps to highlight WARN and ERROR messages in the log so there is less chance of missing them; though it does feel strange to watch a translation take place without the log window scrolling past as usual!

The **Log Debug** option forces FME to show additional messages (of any of the usual types) that depict what is going on at a lower level in the FME engine. This option can be helpful when trying to debug a workspace problem, and particularly useful for logging information about HTTP connections.

WARNING

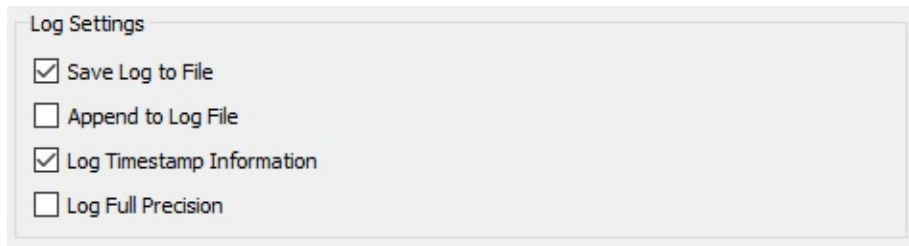
*Logging debug messages is **NOT** a good idea unless you already have a problem you are trying to track down. That's because it includes errors and warnings that are part of the natural process of a workspace.*

For example, this message looks bad, but it is really a common debug message in successful translations:

ERROR |BADNEWS: File C:\apps\FME\metafile\MULTI_WRITER.fmf could not be opened (tabrdr.cpp:1037)

Log Timings

Besides log message types, there are other useful options under Tools > FME Options > Translation.



One of these options is the ability to turn log timestamp information on or off.

Log timestamps indicate the absolute date and time for each step of the translation process. They also show the time taken by FME to process the previous stage and the cumulative time taken to reach that point in the translation.

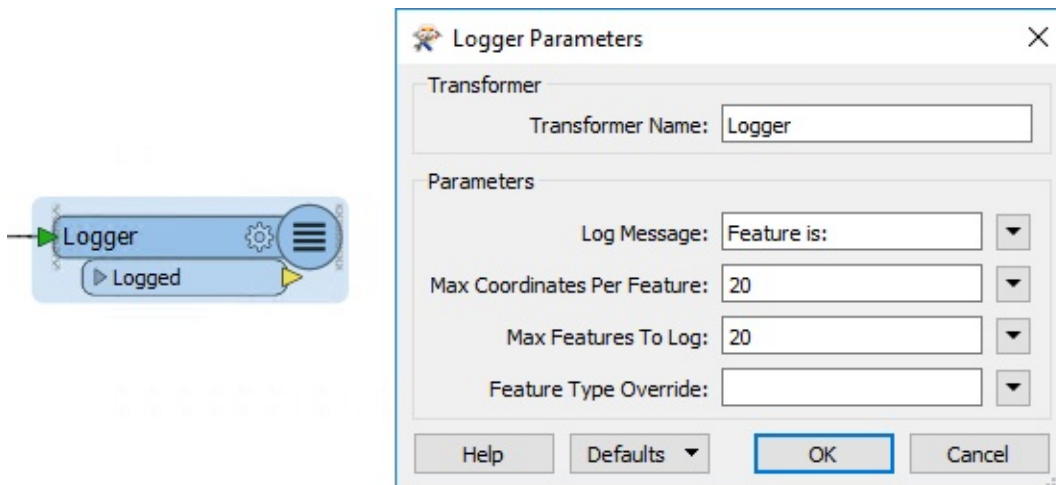
Mr. Statistics-Calculator says...

What kind of statistician would I be if I didn't like numbers? Even so, I recommend you to turn on log timings (they are not turned on by default) and keep them turned on. They do add to the amount of content in the window, but 97.8% of users find them incredibly useful when trying to debug a poorly-performing translation.

Logger Transformer

Besides FME writing information to the log window, the user can force information about a feature to be logged using the Logger transformer.

This transformer is extremely useful for debugging purposes, and has a number of parameters:



The 'Log Message:' parameter is especially important. When a feature is logged, the message entered into that parameter appears with the feature in the log window. Setting a unique message helps you locate logged features with the log search function.

Spatial Log File

Besides writing the log to a text file (<workspace name>.log) FME also writes a spatial log. This is a dataset of features that have been mentioned in the log - either because of a warning from FME or by use of the Logger transformer.

	PreviousSundayFinder.fmw Type: FMW File	Date modified: 17/02/2017 5:21 PM Size: 52.3 KB
	PreviousSundayFinder.log Type: Text Document	Date modified: 17/02/2017 5:20 PM Size: 15.3 KB
	PreviousSundayFinder_log.ffs Type: FFS File	Date modified: 17/02/2017 5:20 PM Size: 1.47 KB



The log is an FFS (FME Feature Store) format dataset. It can be opened within the FME Data Inspector to inspect the features and identify the problems that caused them to be rejected.

This is a much easier process than trying to locate the same feature within the full source dataset.

Interpreting the Log Window

It can't be emphasized enough that the log window is **THE** most important place to look for information when a translation does not complete as expected - or to be sure that a translation has completed as expected.

Here are some suggestions for interpreting what you see in the log window.

Check for Errors

If an ERROR occurs, it is likely that the translation will be halted. There will be a lot of red text and some terminating statements such as:

```
Program Terminating
Translation FAILED.
```

There may be several ERROR messages, so scroll back up the log window to try and identify the first of these, which is likely to be the root cause of the problem. For example this:

```
ERROR |Error connecting to PostgreSQL database(host='postgis.train.safe.com',
port='5432', dbname='fmedata', user='fmedata', password='*'): 'FATAL: password
authentication failed for user "fmedata" FATAL: password authentication failed for user
"fmedata"
```

...is an obvious problem with authenticating a database connection.

Check for Warnings

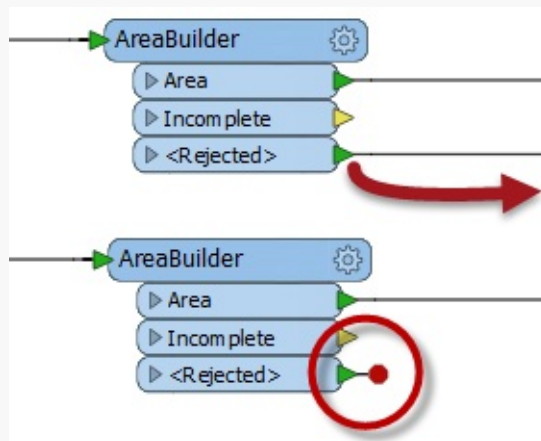
Even when a translation succeeds, it's important to check the log for the following comment:

```
Translation was SUCCESSFUL with X warning(s)
```

If there are any warnings (i.e. if $X > 0$) then use the search option to look for the word WARN. Any warning messages might have important consequences for the quality of the output data.

TIP

More and more FME transformers favour a <Rejected> port, rather than dropping the feature and logging it. This allows the workspace author to deal with bad features immediately, rather than having to handle them after the workspace has finished.



However, if a feature is rejected without being handled, then the translation will terminate.

Check Timestamps

Timestamps in the log have this format:

Absolute Date	Absolute Time	Total FME Time	FME Time for this Step
2017-05-11	13:27:07	0.2	0.1

The time taken for the process being logged is the "Time for this Step". Often this is rounded down to 0.0. The Total FME Time is, literally, the amount of time that FME has been actively processing.

Sometimes, this total appears at odds with the absolute date/time:

Absolute Date	Absolute Time	Total FME Time	FME Time for this Step
2017-05-11	13:27:07	0.2	0.1
2017-05-11	13:29:30	68.4	68.3

Here the step took 68.3 seconds, but the difference between the two absolute times is 123 seconds!

The absolute start and end times can differ from FME processing time because non-FME processes, such as a database query, add to the absolute time taken without adding to the FME processing time.

Therefore the log can provide an indication of the efficiency of external processes; for example, database reading. A slow database read would imply database indexing needs to be improved.

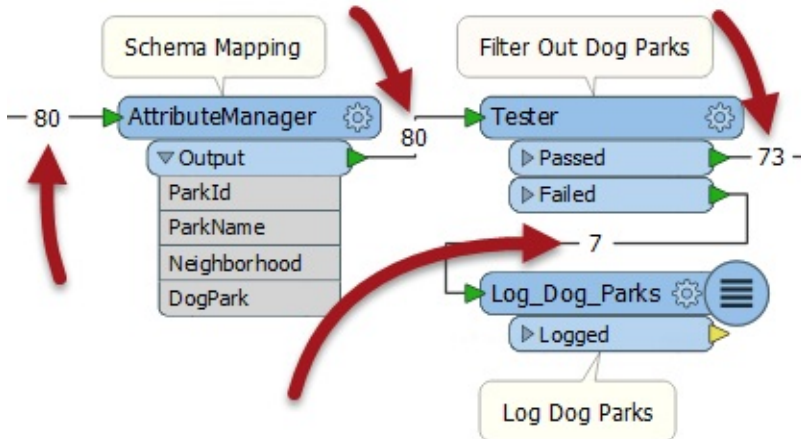
Sequence

The FME engine is set up to process features as efficiently as possible. In most cases that is not a single process that takes place in the same order as the transformers appear on the Workbench canvas.

Because of that, it's unlikely the log window will show features processing in the order you would expect. Don't be confused if the order of actions in the log file doesn't match the layout of the workspace.

Feature Counts

A workspace **Feature Count** refers to the numbers shown on each connection once a translation is complete:



When a log file shows a translation that ended in an error or where the number of output features was not what was expected, then the Feature Count values shown on each connection can help to diagnose where the error occurred.

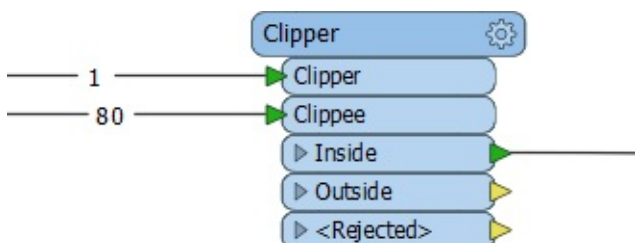
In the above screenshot, if you were expecting 74 output features, but only got 73, then you would need to tract the feature counts to find where that missing feature went (it appears to unexpectedly fail the Tester tests).

Incorrect Output

When the number of output features is incorrect, then there are several things to check.

If you get zero output, and the feature counts show that all features entered a transformer, but none emerged, then you can be fairly confident that the transformer is the cause of the problem.

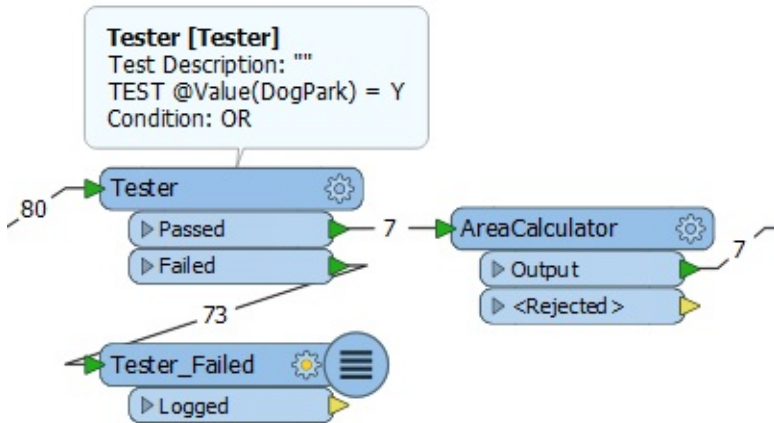
Here, for example, 80 features enter the Clipper transformer (to be clipped against a single boundary) but none emerge.



Here the user will want to inspect the data as it enters the transformer as it's possible that Clipper and Clippees don't occupy the same coordinate system, hence one does not fall inside the other. The data is not rejected, because it is not invalid data; it merely does not pass the test expected.

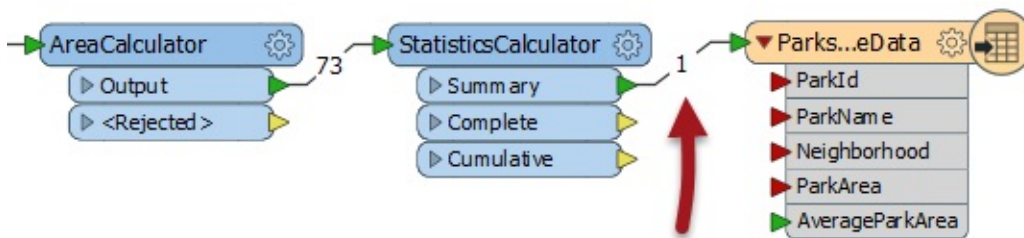
Another common cause of missing features is the wrong output port being connected. For example, a user might get confused over the logic in a Tester transformer, and connect the Passed port when in fact the required data emerges from the Failed port.

Here - for example - the workspace author has set up a test for DogPark=Y, and kept the Passed features:



In reality they want to either keep the Failed features or set the test to DogPark=N.

Similarly, this user has connected the StatisticsCalculator Summary output port, when they really wanted the Complete port connected:

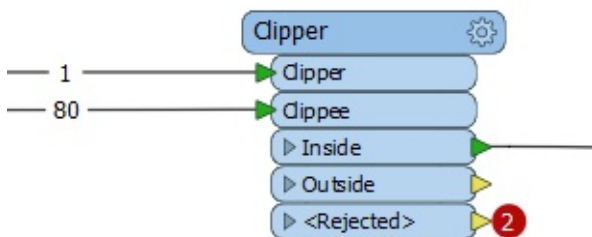


The red attribute connectors on the writer feature type are a good indication that something is going wrong.

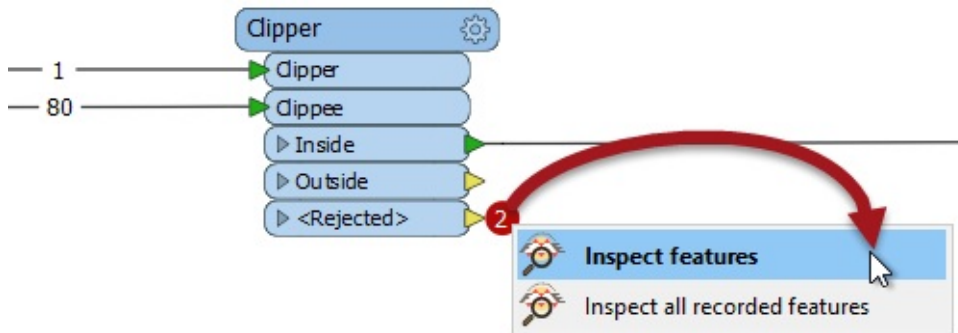
Rejected Features

Sometimes when features go missing they are being rejected by a transformer. Many transformers include a <Rejected> port to output these invalid features.

To help in debugging, rejected features are automatically counted on a <Rejected> port, even if there is no Logger or other transformer attached:



Additionally, the rejected features are also saved as a temporary dataset, so right-clicking (or double-clicking) allows the user to inspect them in the Data Inspector, whether or not the features were expected.



WARNING

The <Rejected> ports on transformers carry out one of two actions; they can either record all rejected features or they can stop the translation after the first rejected feature.

This action is controlled by a setting in the Navigator window (Workspace Parameters > Translation > Rejected Feature Handling). More information appears in the chapter on Translation Components.

Inspecting Output

Inspecting the output of a translation is simply a case of viewing it in either the FME Data Inspector or the application in which the data is intended to be used.

It should be straightforward to check a dataset and see if any of its components are incorrect. However, it is important not to jump to conclusions about *where* a problem occurred.

Generally, issues in an output dataset can occur at one of three operations:

- The data was incorrectly transformed in FME Workbench
- The data was incorrectly written by FME Workbench
- The data is being incorrectly interpreted by another application

For example. You open an FME-generated dataset in application X and find that "Parc national des Pyrénées" is rendered as "Parc national des Pyr?n?es". Obviously an encoding problem has occurred, but where?

Maybe the data was incorrectly encoded before FME read it? Or maybe the data was correct but FME wrote it using the wrong encoding? Or maybe FME wrote the data correctly, but application X does not support that form of encoding?

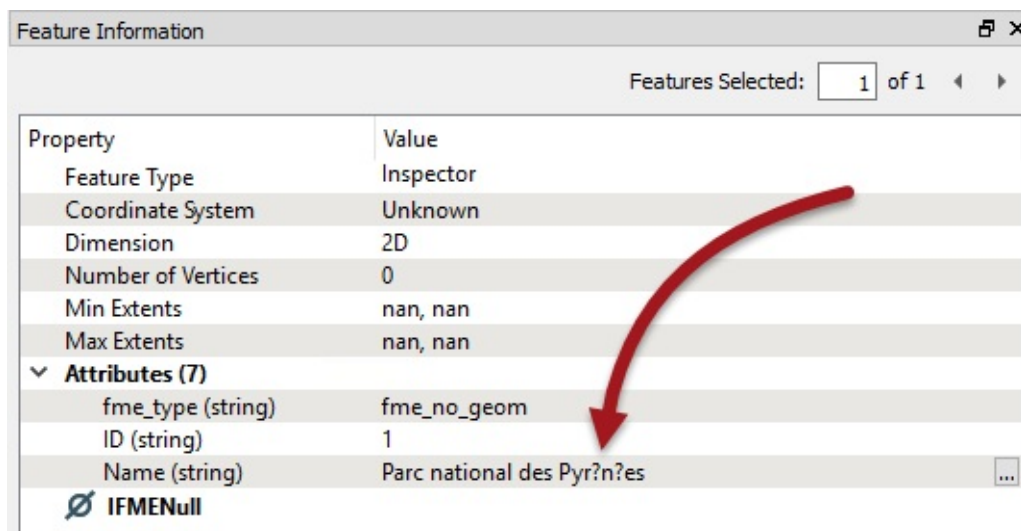
So, the fact that data appears incorrect in another application is no real indication of where that problem was introduced. In this scenario it's best to check the data in the following order:

Inspect the "Pre-Output" Data

Because FME has the option *Redirect to FME Data Inspector* (and the option to *Run with Full Inspection*) it's possible to look at the data before it is written to the output dataset.

So, if a problem occurs in the output, first re-run the translation with one of these options and check whether the data is correct before FME tried to write it.

In the above example, if the Data Inspector shows ? characters at this point:



...then the data was bad before FME even tried to write it. Go back to the source data and check if that is even correct, before checking the workspace to see if encoding might have been affected in there.


TIP

Be aware that "Run with Full Inspection" will take up more time and system resources, and also needs all Writers disabled separately to avoid still writing data.

Inspect the Output Dataset in FME

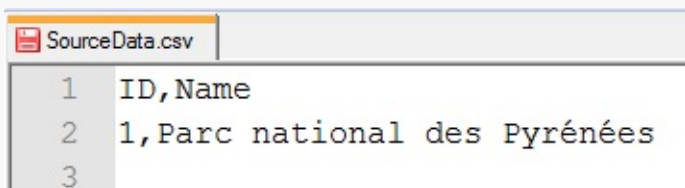
If the "pre-output" data is correct then any issue is not in the data transformation; therefore it might be that it is being written incorrectly.


So, open the output dataset in the FME Data Inspector. This will show the data as FME wrote and interpreted it. If the data is incorrect here then the problem likely occurred during writing of the data.

In the above example, if the Data Inspector shows  characters at this point, then the data has been mangled when it was being written. Either FME has a limitation in that writer or the workspace author has set an encoding parameter incorrectly on the writer.

TIP

Another technique - for non-binary formats - is to open the dataset in a text editor for inspection. This can give you absolutely definitive proof of what has been written to the output file, without the possibility of it being re-interpreted by a different software application.




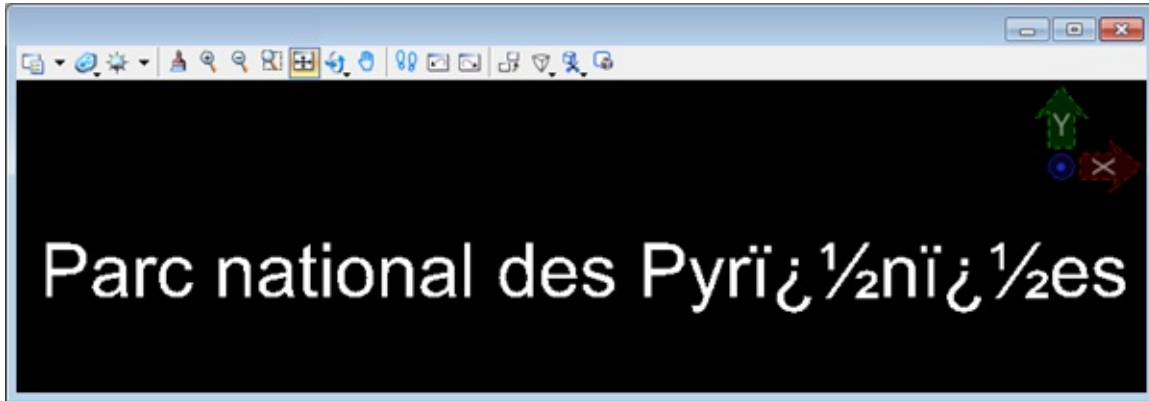
This proves this dataset contains the correct encoding (and is being viewed on a computer that supports it). Of course you would want to check whether your text editor (and computer) could view accented characters before assuming that  was a problem in the data and not the text editor!

Inspect the Output Dataset in Another Application

If the "pre-output" data was correct, and the dataset still looked correct in the FME Data Inspector, then it might be that the intended application is not viewing the data correctly.

So, open the output dataset in the application in which it is intended to be used. If the data is only incorrect here then the problem is more likely to be with how the application interprets the data.

In the above example, if the data was correct within FME, and it written correctly by FME (especially when a text editor proves it) then suspicion falls upon the application that is showing  characters:



That would be particularly true if the format was non-native to that application (for example, reading a Geodatabase outside of an Esri product).

Sister Intuitive says...

All of the above techniques are for narrowing down where an error might have occurred, but it doesn't rule out any other cause.

For example, a dataset looks correct on application X, but crashes application Y. Does that mean application Y has a problem? On one hand yes (it shouldn't crash) but maybe application X is just more tolerant of bad data?

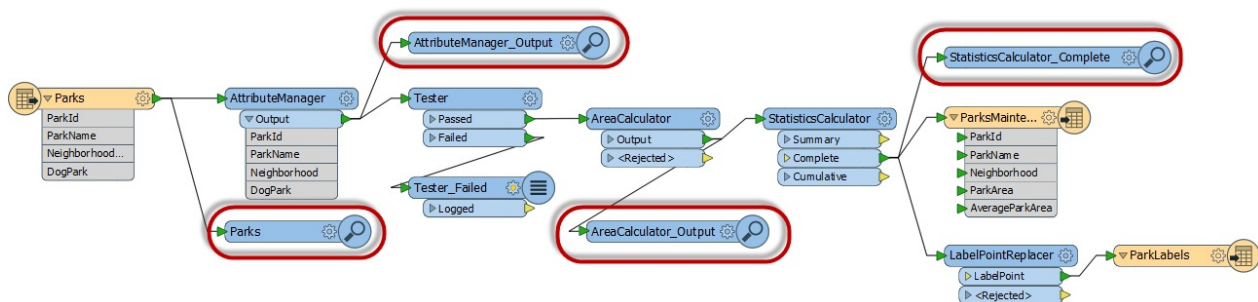
In short, these techniques identify where to investigate first, but won't provide an absolute answer by themselves.

Workspace Testing

Once a problem in the output has been narrowed down to the workspace that produced it, it's necessary to identify where that error occurred. To do so often means re-running the workspace in various ways to narrow down further the location of the issue.

Testing a Workspace

The most common way to narrow down a problem is to run it with Inspector transformers attached at specific points.



Ideally you have already got a good idea of which section of workspace - or which transformers - might be at fault; so only attach Inspectors to those parts you suspect.

Re-run the workspace and inspect each set of data to determine at which point the problem occurred.

TIP

It's certainly quicker to use "Run with Full Inspection" than to manually add Inspector transformers. However, "Run with Full Inspection" causes a workspace to run more slowly, and at some point the time saved in setup is just being lost in run-time.

Testing a Section in Isolation

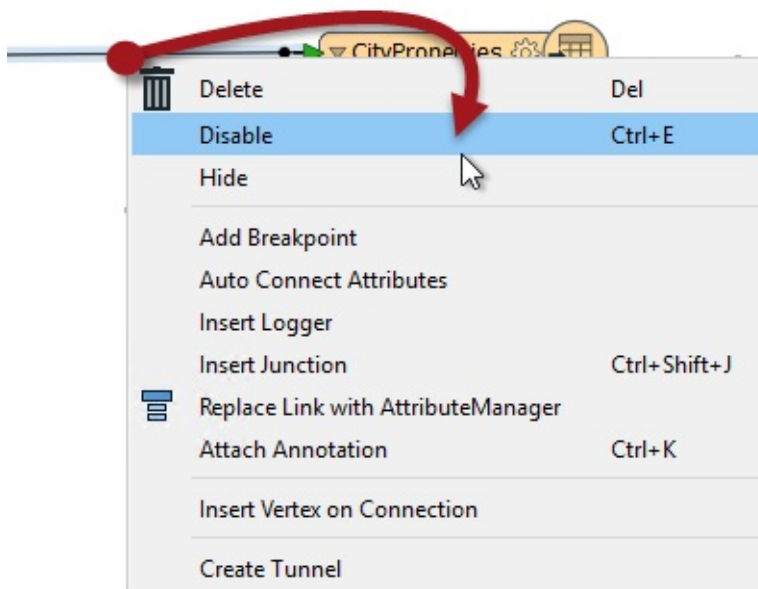
Testing and debugging a large workspace is easier when it's possible to isolate sections and test them separately. This is done by disabling connections to the parts that do not need testing.

TIP

This is also a useful technique for where a failure halts the translation. That's because a failure can stop a translation before data is sent through Inspector transformers to the Data Inspector!

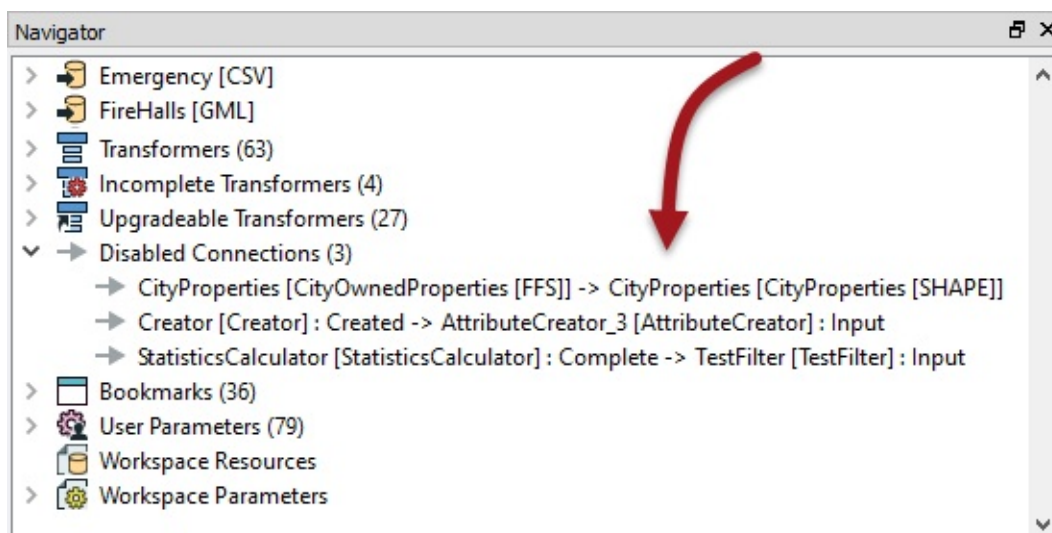
In these scenarios you need to stop data reaching the failure point, in order to view that data in the Data Inspector.

A connection is disabled by right-clicking it and choosing the option to Disable (or selecting it and using the shortcut Ctrl+E):

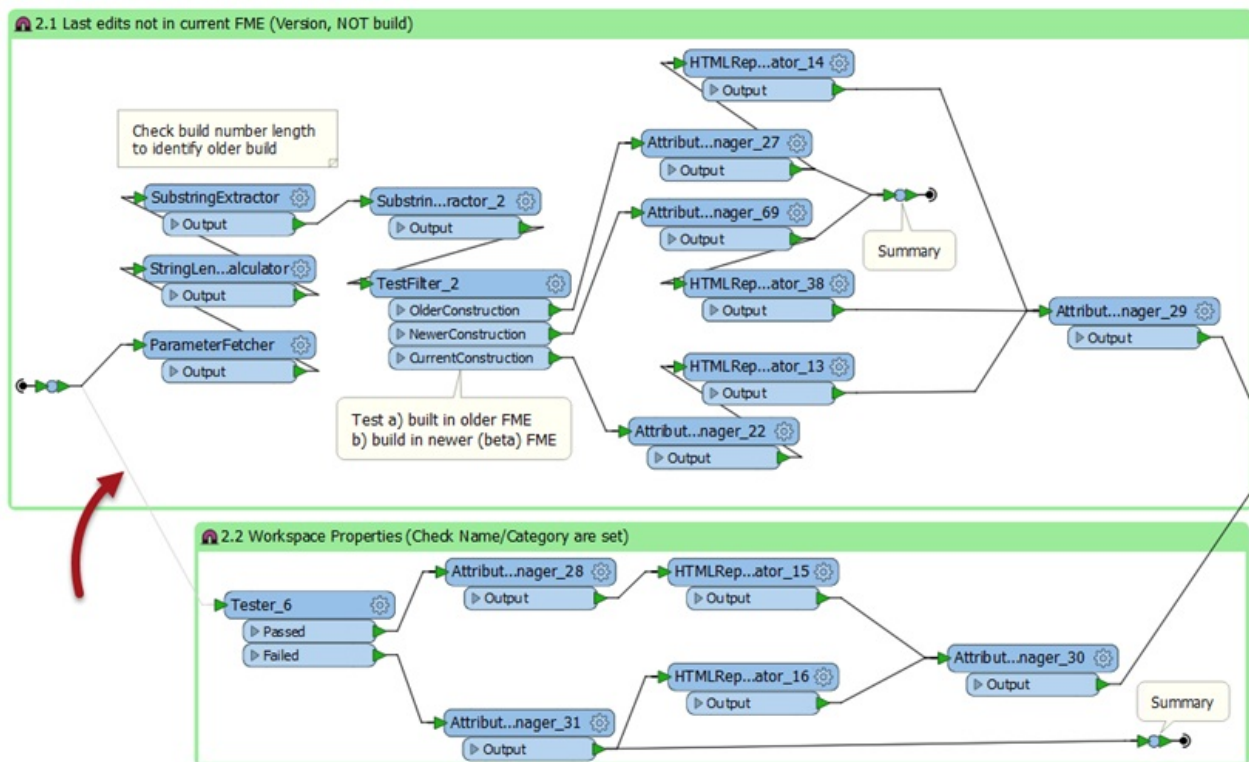


A disabled connection is rendered inoperative in much the same way as if it had been deleted, and no features will pass through.

Disabled connections are listed in the Navigator window. Clicking them will move the canvas display to the location of that disabled connection.



Here the user has chosen to disable a connection so that they can test the upper part of the workspace without the lower part running as well:



This speeds up the translation so that testing of the required section runs quicker, but also removes the possibility that the lower part would be responsible for any problems with the output.

Testing with Python/SQL Transformers

Testing with Inspector transformers is not a useful technique where the problem occurs inside a multi-line command; for example a script inside a PythonCaller or SQLExecutor transformer.

In these scenarios you would need to debug the script in other ways. Because FME doesn't include a full development environment for scripting languages, you would likely need to strip down the script to individual components, and build it back up, testing each part at a time until you locate the issue.

Miss Vector says...

For testing purposes it's much easier to disable certain parts of a workspace when you've properly divided it up into sections with bookmarks. So here's a question for you:

It's possible to disable other objects besides connections. Can you pick out which of these objects (there may be more than one) can be disabled in Workbench?

- 1. Transformers*
- 2. Feature Types*
- 3. Annotation*
- 4. Bookmarks*

Exercise 3 Debugging a Workspace

Data	GPS Roads data (JSON)
Overall Goal	Fix the bugs in someone else's workspace
Demonstrates	Debugging Best Practice
Start Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex3-Begin.fmw
End Workspace	C:\FMEDData2017\Workspaces\DesktopBasic\BestPractice-Ex3-Complete.fmw

As President of your local FME user group, you are responsible for checking projects that are submitted for presentation at your monthly meetings. One day you receive this in an email:

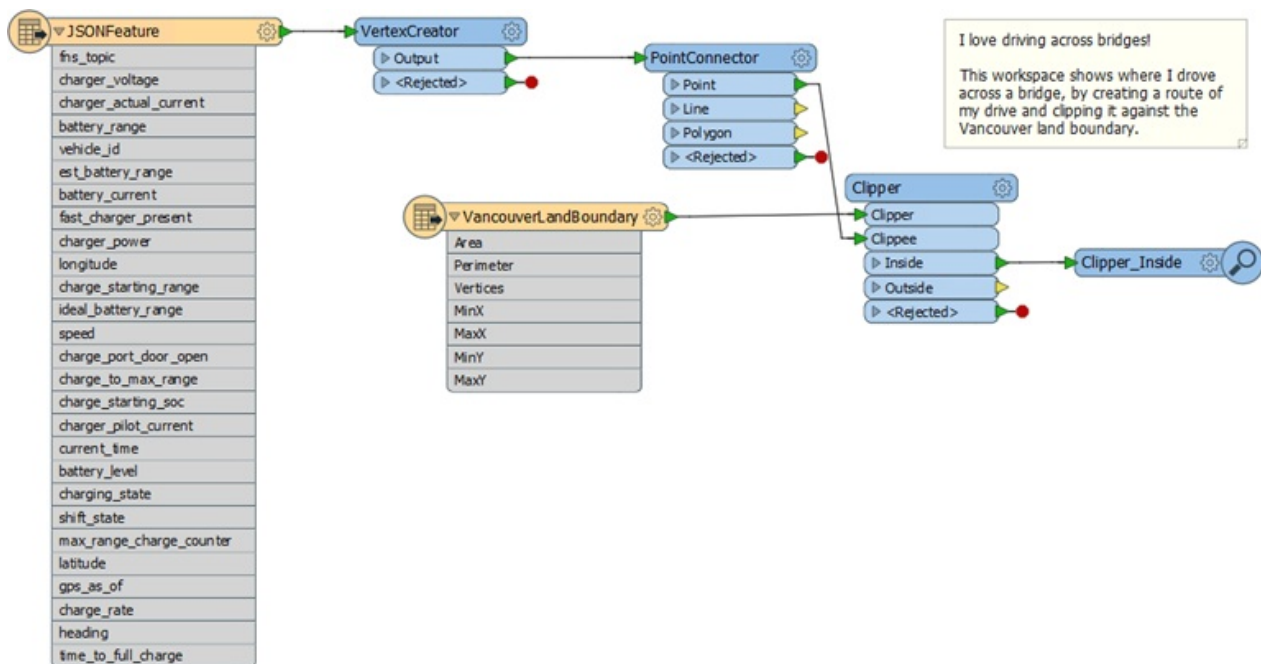
Mr Flibble says...

I love driving across bridges! This workspace takes a set of GPS points, converts them to road lines, and then shows where I drove across a bridge by clipping it against the Vancouver land boundary.

Unfortunately Mr Flibble is seriously in error, and has produced a very poor workspace. Rather than turn down his presentation, let's help him debug the problems so it is worth sharing.

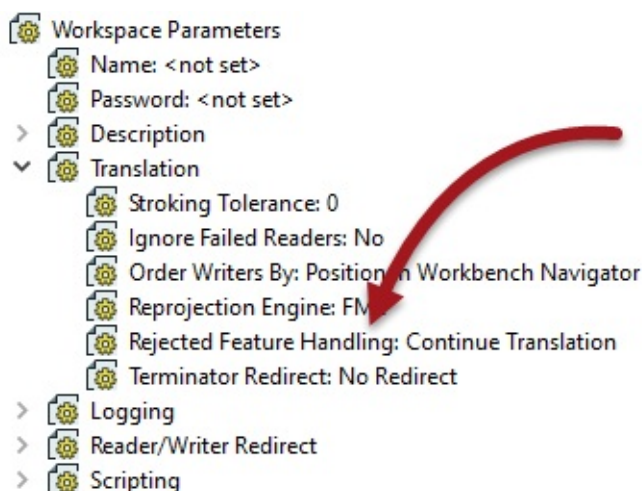
1) Set Rejected Feature Handling

Start FME Workbench and open the starting workspace. It looks like this:

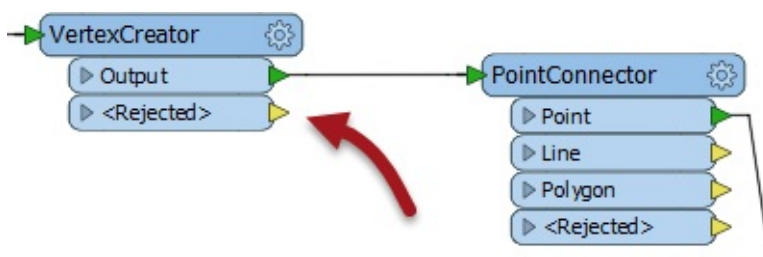


Because we're trying to test a workspace - we're not trying to pick up on errors that occur in production - it's probably better to have the Rejected Feature Handling not terminate the translation just because a feature has been rejected. We want the workspace to continue and find what other issues might occur.

So, check the Workspace Parameters section of the Navigator window and set the Rejected Feature Handling parameter to Continue Translation:



Now the Rejected ports on transformers will lose their "stop" icon and look like this:



2) Debug Workspace

Run/debug the workspace to track down all the problems in it (I count five). Remember the order of debugging operations should be something like the following:

- Set the log file parameters before running the translation
- Run the translation. Examine the log for warnings and errors
- Inspect the output datasets, if the translation wrote any
- Check feature counts in the Workbench canvas to locate where problems occurred
- Check reader, writer, or transformer parameters at the point of failure
- Test the workspace by adding Inspector transformers to view data at different points
- If necessary, run the workspace in feature debug mode

Be sure to fix any problems that you find!

Once the workspace is fixed, the output should look like this in the FME Data Inspector:



CONGRATULATIONS

By completing this exercise you have learned how to:

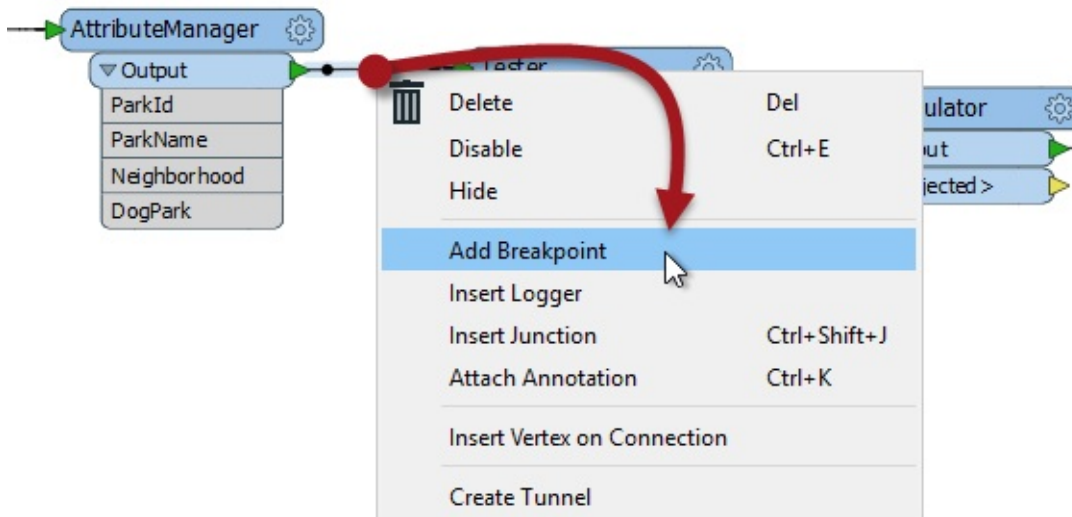
- *Set the Rejected Feature parameter*
- *Read the translation counts to locate a workspace problem*
- *Used Inspector transformers to identify problems*
- *Interpret a log file to ensure everything is correct*

Feature Debugging

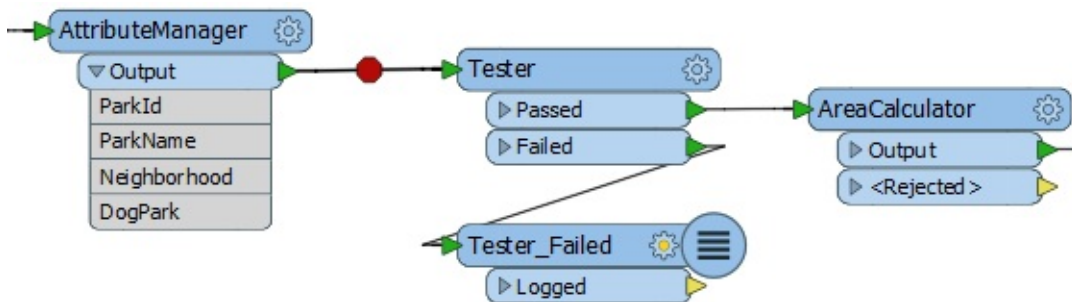
Feature Debugging is a tool that allows individual features to be inspected, one-by-one, during a translation. As might be imagined, this is very useful for debugging purposes.

Feature Debugging is triggered by "breakpoints"; workspace connections that are flagged by the user as a location where features should be inspected.

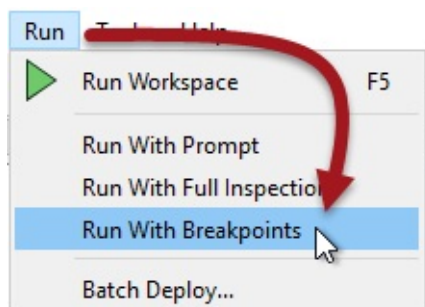
Here a user wishes to inspect data after processing by the AttributeRenamer transformer. A right-click on the connection and selection of Add Breakpoint is used to set it up:



The connection is highlighted in a darker black color with a red "stop" sign, to denote its new status:

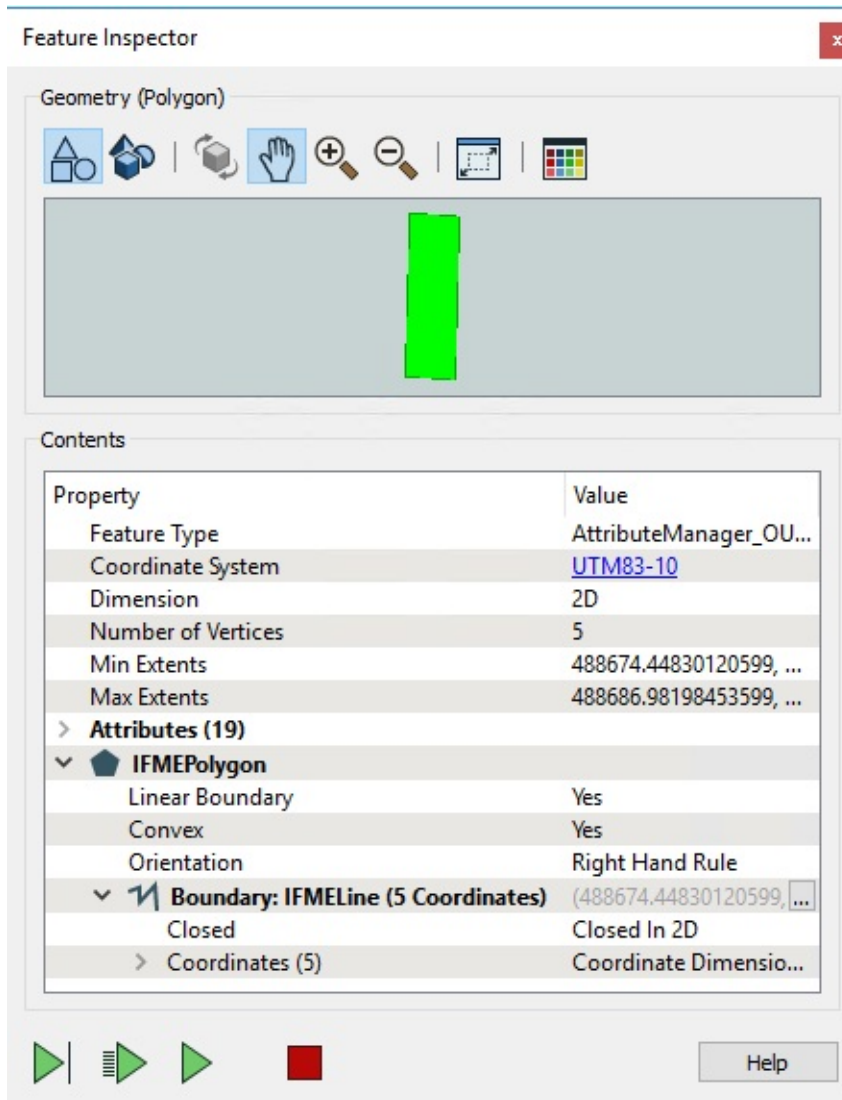


Now the workspace is run using "Run with Breakpoints":







When the first feature arrives at the breakpoint, the translation is temporarily paused and information about the feature displayed in a Feature Inspector window.

The upper part of the window shows a graphic representation of the feature; the lower part lists properties such as Feature Type and Coordinate System; plus attribute and geometry information.

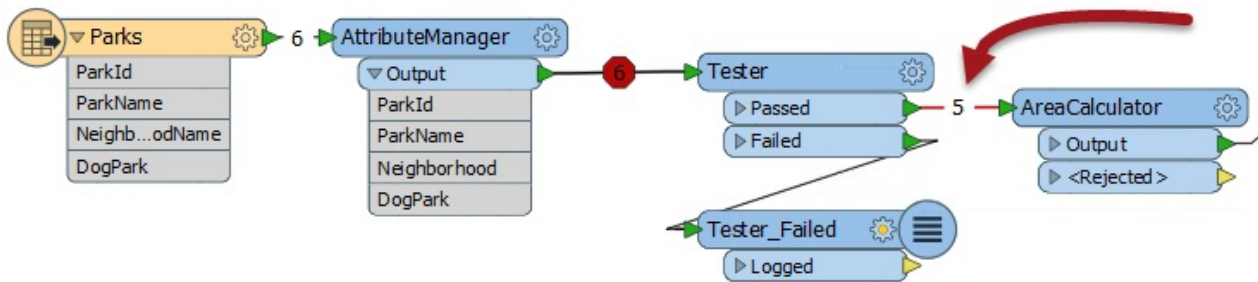


There are four buttons at the foot of the Feature Inspector window:

Button	Operation	Description
	Step to Next Connection	This tool steps through the workspace one transformer at a time, showing the status of a feature as it is processed.
	Step to Next Breakpoint	This tool re-starts the translation, stopping the next time a feature reaches an inspection point.
	Continue Translation	This tool re-starts the translation, ignoring all further breakpoints.
	Stop Translation	This tool stops the translation.

The currently active connection is highlighted red to show it is the location where the translation is currently paused.

The current connection might be different to the original breakpoint when the "Step to Next Connection" tool has been used.



TIP

Use Feature Debugging when the output from a transformation is wrong and you can't tell why, or when you suspect one particular feature is causing a problem. It's likely to help less when the problem is a crash or ERROR in the log window.

Miss Vector says...

Now you've learned about Feature Debugging, why not try the previous exercise again, this time using these techniques to show what happens step, by step?

Module Review

This module was designed to help you use FME Workbench in the most efficient manner, to effectively manage FME related projects, and to ensure those projects are scalable and portable.

What You Should Have Learned from this Module

The following are key points to be learned from this session:

Theory

- Best Practice is the act of using FME in a manner that is efficient, but also easily comprehensible by other FME users.
- Best Practice in FME can be divided into Style, Methodology, and Debugging.

FME Skills

- The ability to use Workbench in the right way, so as to provide maximum efficiency and productivity.
- The ability to use bookmarks, annotation, and workspace settings to apply a well-designed workspace style.
- The ability to use FME as part of a larger project, in an organized, efficient, and scalable manner.
- The ability to interpret an FME log file at a basic level and to use debugging techniques to locate problems in a translation.

Further Reading

For further reading why not browse [articles tagged with Best Practice](#) on our blog?

Questions

Here are the answers to the questions in this chapter.

Miss Vector says...

Which of the following is NOT a method of creating a bookmark?

- 1. Click the Insert Bookmark button on the toolbar*
 - 2. Select a transformer, right click, choose Create Bookmark**
 - 3. Select multiple transformers, right click, choose Create Bookmark*
 - 4. Use the Ctrl+B shortcut*
-

Miss Vector says...

Which of these is NOT a type of annotation in FME Workbench?

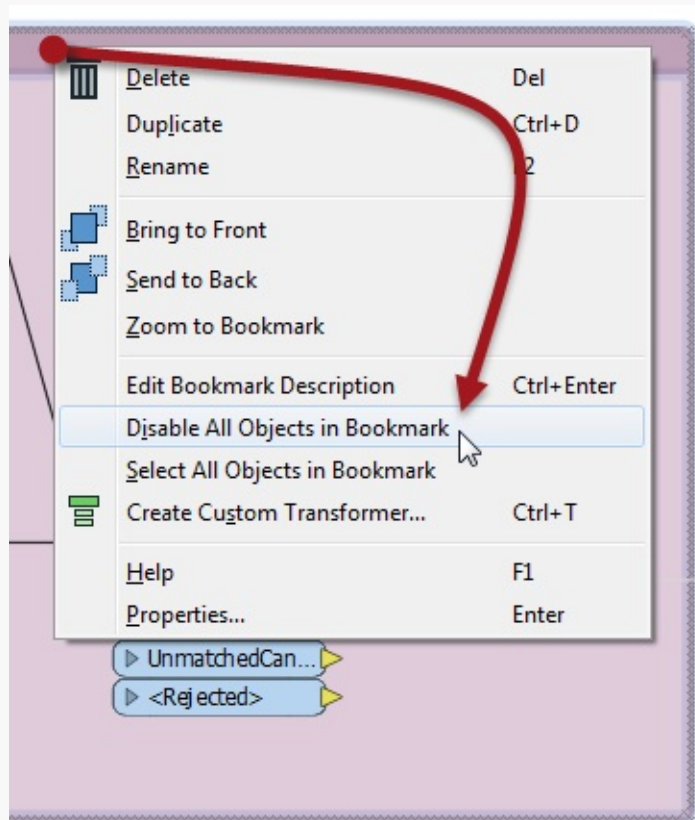
- 1. Header Annotation*
 - 2. Parameter Annotation**
 - 3. User Annotation*
 - 4. Summary Annotation*
-

Miss Vector says...

It's possible to disable other objects besides connections. Can you pick out which of these objects (there may be more than one) can be disabled in Workbench?

- 1. Transformers**
- 2. Feature Types**
3. Annotation
- 4. Bookmarks**

OK, technically you can't disable a bookmark, but you can choose to disable all the objects within it.



Miss Vector says...

What ways can a template be opened for use in FME Workbench (there might be more than one correct answer):

- 1. Open it using File > Open***
- 2. Open it using File > Open Recent Template***
- 3. Double-click the fmwt file in Windows Explorer***
- 4. Use Create Workspace from Template in the Getting Started part of the start tab***

Yes! Every one of those is a valid method of using a template file!

Exercise 4 FME Hackathon

Data	Roads (Autodesk AutoCAD DWG and/or PostGIS)
Overall Goal	Find the shortest route from the hackathon to an Italian Cafe
Demonstrates	Data Translation, Transformation, and Best Practice
Start Workspace	None
End Workspace	C:\FMEData2017\Workspaces\DesktopBasic\BestPractice-Ex4-Complete.fmw

A regional GIS group is holding an FME Hackathon and you have been invited to take part.

You have been provided with a set of source data and asked to create a useful project from it. You decide that it would be interesting to produce a tool that maps the route from the hackathon venue to a cafe the where a group get-together will be held that evening.

So, your task is to use the data available to you to calculate the best route from the convention centre to the cafe, and to write out that data to GPX format so folk can use it in their GPS/mobile device.

1) Create Database Connection

The source data has been provided in a PostGIS database; therefore our first task should be to create a connection to it. That way we can use the connection instead of having to enter connection parameters.

In a web browser visit <http://fme.ly/database> - this will show the parameters for a PostGIS database running on Amazon RDS.

In Workbench, select Tools > FME Options from the menubar

Click on the icon for the Database Connections category, then click the [+] button to create a new connection. In the "Add Database Connection" dialog, first select PostgreSQL as the database type. Then enter the connection parameters obtained through the web browser.

The screenshot shows the 'Add Database Connection' dialog box. The 'Database Connection' dropdown is set to 'PostgreSQL'. The 'Name' field contains 'Hackathon PostGIS Database'. The 'Connection Parameters' section includes the following fields: 'Host' (postgis.train.safe.com), 'Port' (5432), 'Database' (fmedata), 'Username' (fmedata), and 'Password' (masked with dots). At the bottom, there are four buttons: 'Help', 'Test...', 'Save', and 'Cancel'.

Give the connection a name and click Save. Then click OK to close the FME Options dialog.

TIP

*The completed workspace for this exercise uses a database connection called **Hackathon PostGIS Database***

If you wish to open/use this workspace, you should create your connection with the same name. That way when you open the workspace it will automatically find a matching connection.

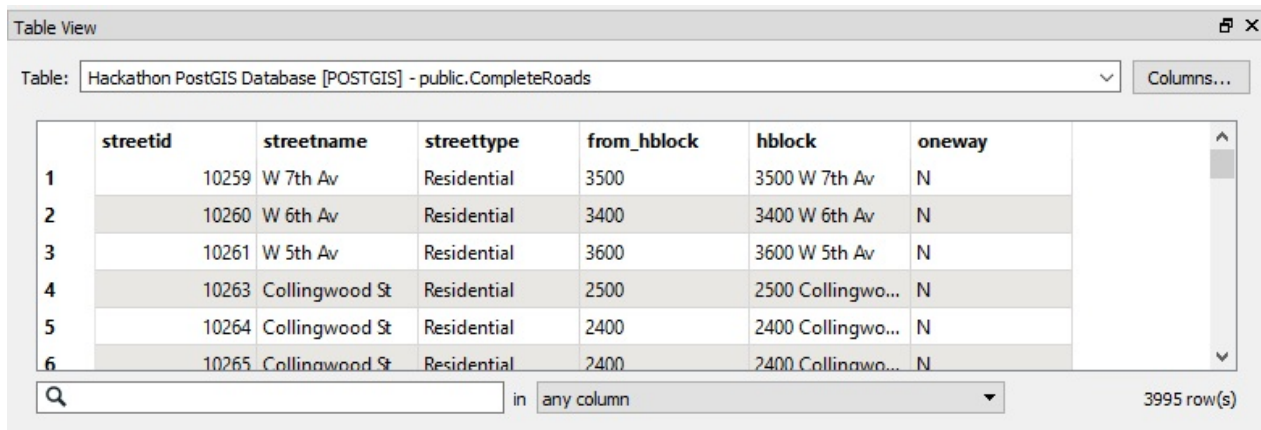
This is a good illustration of the importance of naming conventions for database (and web) connections.

2) Inspect Data

Start the FME Data Inspector to inspect the dataset we will be using. Select File > Open Dataset and, when prompted, enter the following:

Reader Format	PostGIS
Reader Dataset	Hackathon PostGIS Database
Parameters	Table List: public.CompleteRoads

Click OK to close the dialogs and open the data. You will see a set of road features each of which has a set of attributes. One attribute specifies whether a feature represents a one-way street. It's important to know this if we want to calculate a route that is actually legal!



	streetid	streetname	streettype	from_hblock	hblock	oneway
1	10259	W 7th Av	Residential	3500	3500 W 7th Av	N
2	10260	W 6th Av	Residential	3400	3400 W 6th Av	N
3	10261	W 5th Av	Residential	3600	3600 W 5th Av	N
4	10263	Collingwood St	Residential	2500	2500 Collingwo...	N
5	10264	Collingwood St	Residential	2400	2400 Collingwo...	N
6	10265	Collingwood St	Residential	2400	2400 Collingwo...	N

Table: Hackathon PostGIS Database [POSTGIS] - public.CompleteRoads Columns...

3995 row(s)

NB: If you have any problems using the PostGIS database - for example connectivity problems with a firewall - then the following AutoCAD dataset can be substituted with very few changes required:

Reader Format	Autodesk AutoCAD DWG/DXF
Reader Dataset	C:\FMEDData2017\Data\Transportation\CompleteRoads.dwg

3) Start Workbench

Start Workbench and use the option to Generate a workspace.

Reader Format	PostGIS
Reader Dataset	Hackathon PostGIS Database
Parameters	Table List: public.CompleteRoads
Writer Format	GPS eXchange Format (GPX)
Writer Dataset	C:\FMEDData2017\Output\Training\Route.gpx

The workspace will look like this:



Remember, GPX is a fixed-schema format, hence the six different writer feature types that are automatically created. For the sake of Best Practice, you may want to put a bookmark around these features.

4) Add ShortestPathFinder

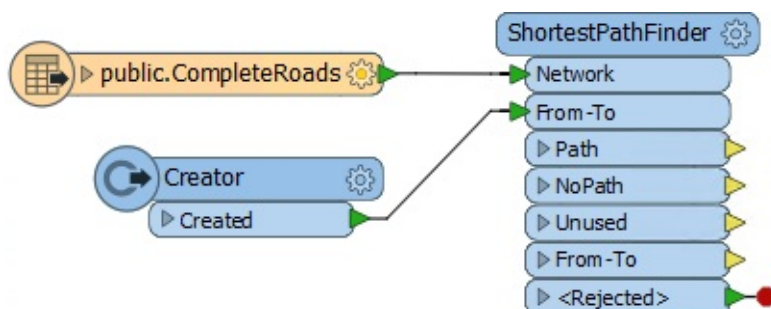
Now we need to start calculating a route. The obvious first step is to add a ShortestPathFinder transformer, which is how we can calculate our route.

So, add a ShortestPathFinder transformer. Connect public.CompleteRoads to the Network port.

5) Add Creator

The other input port on the ShortestPathFinder is for the From-To path (the start and end points of our journey). There are many ways to create this - or even accept input from a web map - but here we'll just manually create a feature with the Creator transformer.

So, add a Creator transformer and connect it to the From-To port:

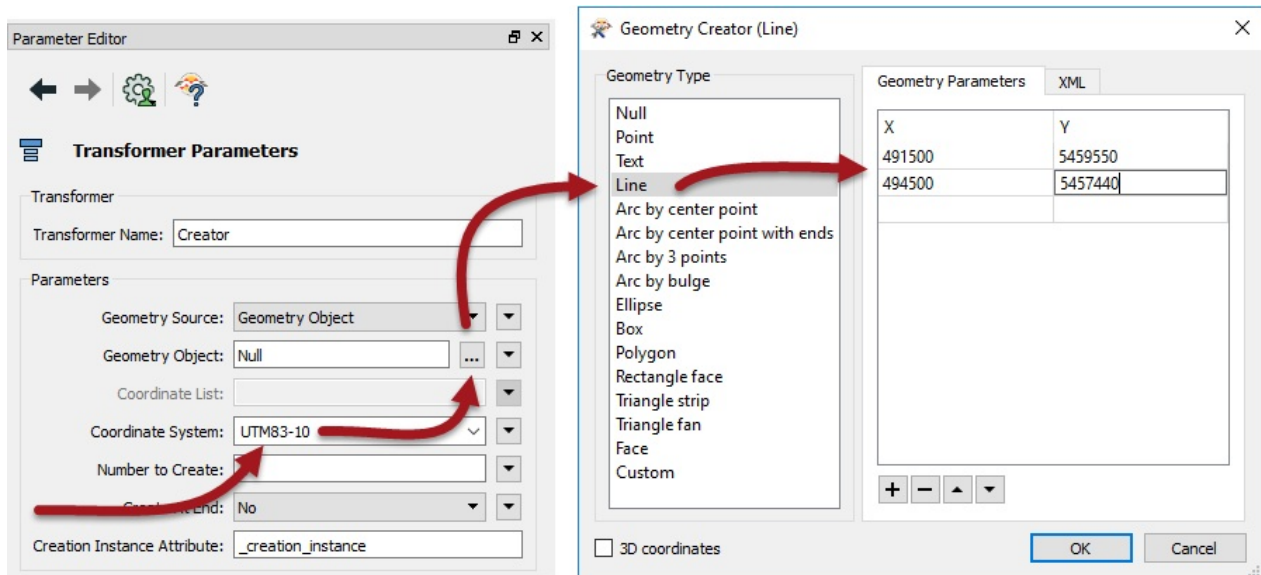


View the Creator's parameters.

Firstly enter UTM83-10 as the coordinate system of the data we are about to create. For the Geometry Object parameter, click the [...] browse button to the right to open a geometry-creation dialog. Select Line as the geometry type and enter the following coordinates:

```
X 491500 Y 5459550
X 494500 Y 5457440
```

The first coordinate is that of the hackathon venue and the second is the closest point in our network to the cafe we're going to visit.

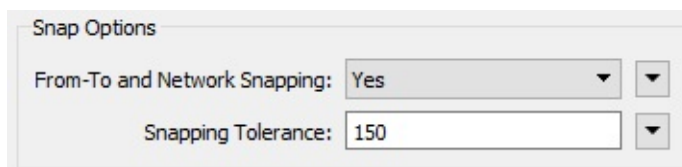


Click the OK button to close the dialog and the Accept/OK button to confirm the changes.

6) Check ShortestPathFinder Parameters

The coordinates of the feature we've added might not sit exactly on the road network. To get around this issue there are parameters we can use in the ShortestPathFinder.

So, inspect the ShortestPathFinder parameters. Under Snap Options set **From-To and Network Snapping** to Yes and enter 150 as the **Snapping Tolerance**:

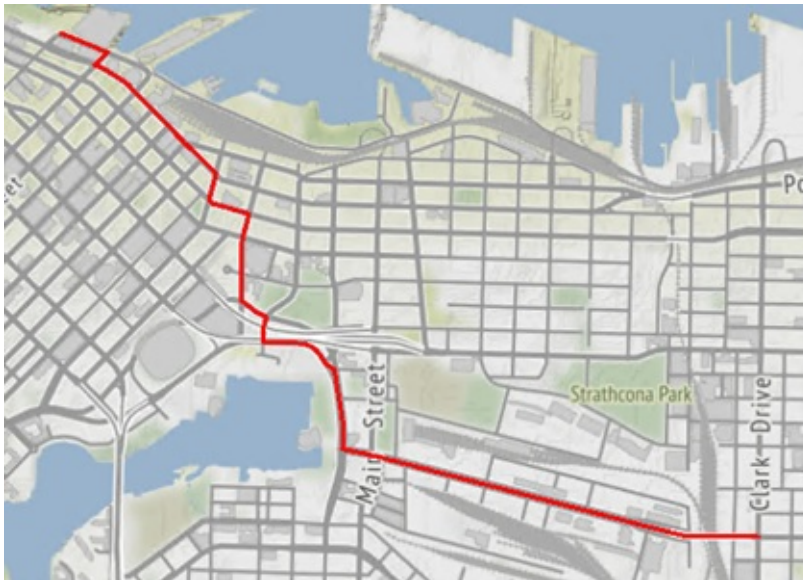


Also notice that there are parameters for network costs - we'll be making use of those later.

7) Run Workspace

Add some Inspector transformers after the ShortestPathFinder and run the workspace to prove it is working up to this point.

If all has gone well, the output will look like this, with a route defined:



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

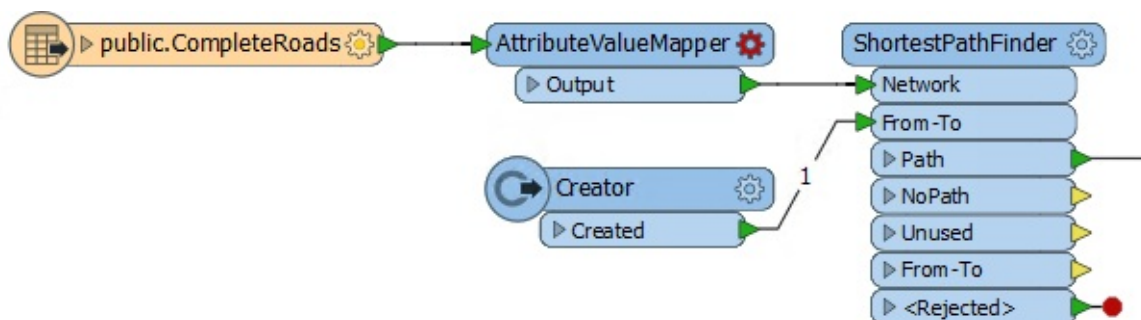
Of course, if all has not gone correctly, you must make use of Inspector and Logger transformers, plus Feature Debugging, to try and locate the error!

8) Add AttributeValueMapper

The result looks fine, but there are some things we are yet uncertain about: for example what if the route uses slower, residential roads? We can force the route to prefer arterial routes by applying a different cost to each road feature.

The cost will depend on the road type. In essence we are mapping road type to cost and the way to do that is with an AttributeValueMapper transformer.

So, add an AttributeValueMapper transformer to the workspace, between the CompleteRoads feature type and the ShortestPathFinder:Network port:



9) Edit AttributeValueMapper

Inspect the AttributeValueMapper's parameters. Enter the following values:

Source Attribute	roadtype
Destination Attribute	Cost
Default Value	2

Attribute Selection

Source Attribute:

Destination Attribute:

Default Value:

Now, underneath those parameters, we'll map some data.

Source Value	Destination Value
Arterial	1
Residential	3

Value Map

Mapping Direction:

Source Value	Destination Value
<input type="checkbox"/> Arterial	<input type="checkbox"/> 1
<input type="checkbox"/> Residential	<input type="checkbox"/> 3
<input type="checkbox"/>	<input type="checkbox"/>

+ - < > < >

Import...

Basically, if the route is arterial (a main road) it will get a cost of 1; residential routes will get a cost of 3 and all other types will get a cost of 2 (because that's the default value). Click Accept/OK to confirm the parameters.

10) Apply Costs

Now we have to apply the costs we have just created. Inspect the ShortestPathFinder's parameters. Enter the following values:

Cost Type	By Two Attributes
Forward Cost Attribute	Cost
Reverse Cost Attribute	Cost

TIP

Why "Two Attributes"? That's because with only a forward cost, I could only ever travel along a stretch of road in the same direction as the vertices are arranged in. Since I don't want to avoid roads based on their vertex direction, using two attributes tells FME the cost is the same in both directions.

Now re-run the workspace to see if there is any difference in the result. It should look like this:



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

So the cost weighting has obviously made a difference. But there is a problem with this result...

Ms. Analyst says...

*The route is taking a longer path this time, and I can see a reason why that might be: cost is being used to weight routes **instead** of the distance, not as well as.*

To explain this, let's say I want to travel from A to B. There is a single residential road feature that starts at A and ends at B, with a route distance of 1.5 kilometres.

There is also a single arterial road feature that starts at A and ends at B. Rather extremely, it loops around the dark side of the moon with a route distance of 768,000 kilometres.

Currently our solution would choose the 768,000 kilometre trip, because it has a cost of "1" compared to the residential route cost of "3"!

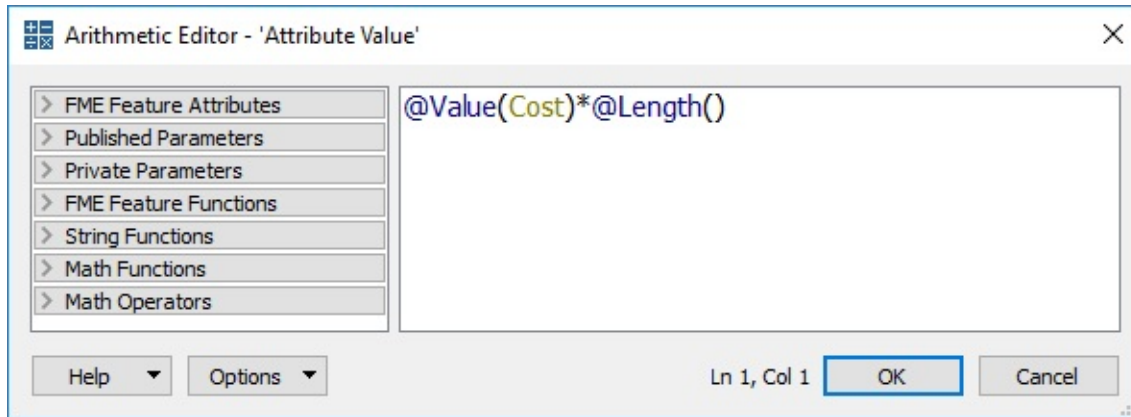
Plainly the exercise data here is nowhere near as ridiculous, but it's equally plain that the route might not be the optimum until distance is factored back into the result.

11) Add AttributeManager

Add an AttributeManager transformer between the AttributeValueMapper:Output port and the ShortestPathFinder:Network port and view the parameters.

In the value field for the Cost attribute, click the dropdown arrow and select Open Arithmetic Editor. In that dialog enter the expression:

```
@Value(Cost)*@Length()
```



In short, we're now multiplying cost by road length to give us a combined weighting.

Re-run the translation to see if it makes a difference:



Map tiles by [Stamen Design](#), under [CC-BY-3.0](#). Data by [OpenStreetMap](#), under [CC-BY-SA](#).

Yes, it does, proving that the route was longer than it could be. Of course, the expression we've used is again very subjective, and could be made more complex to give a better result (we could try a logarithmic scale to see what that produced).

12) Edit AttributeManager

Regardless of our expression, there's another last snag to cover... one-way streets. Currently we have no solution in place to prevent us driving the wrong direction.

Luckily each one-way street is flagged with an attribute, and has its vertices ordered in the direction of permitted travel, so we know which way to avoid. Let's use this information to solve that problem.

We'll need to calculate different cost attributes for each direction now, although the value will only differ when a one-way street is involved. There are - as usual - multiple ways to handle this in FME; let's go with a moderately easy one.

View the AttributeManager parameters again. This time create a new Output Attribute called ReverseCost. In the value field click the drop down arrow and choose Conditional Value.

TIP

Conditional Values are those that are set dependent on a test condition. It's like incorporating a Tester into the AttributeManager. These are covered in more detail in the FME Desktop Advanced training course.

In the definition dialog that opens, double-click in the first "If" row and a Test Condition dialog opens. In here set up a test for oneway = Y. For the Output Value (bottom of the dialog) enter the value 9999 (i.e. the cost of travelling the wrong way is *really* expensive)!

Test Conditions

Pass Criteria

Pass Criteria: One Test (OR)

Composite Expression:

Test Clauses

	Left Value	Operator	Right Value	Negate	Mode
1	oneway	=	Y	<input type="checkbox"/>	Automatic

+ - < > < >

Duplicate

Output Value

Output Value: 9999

Help OK Cancel

Click OK to close that dialog. Back in the previous dialog, double-click where it says <No Action> choose the dropdown arrow and select the attribute Cost:

Condition Statement

	Test Condition	Output Value
If	@Value(oneway) = Y	9999
Else If		
Else	<All Other Conditions>	Cost

+ - ▲ ▼ Edit... Duplicate

Click OK again twice more to close these dialogs.

What we've essentially done is say that if this is a one-way street, set a prohibitively high reverse cost, otherwise just set the usual forward cost.

13) Apply Cost

One last change. Check the ShortestPathFinder parameters and change the ReverseCost attribute from Cost to ReverseCost:

Parameters

Cost Type: By Two Attributes

Forward Cost Attribute: Cost

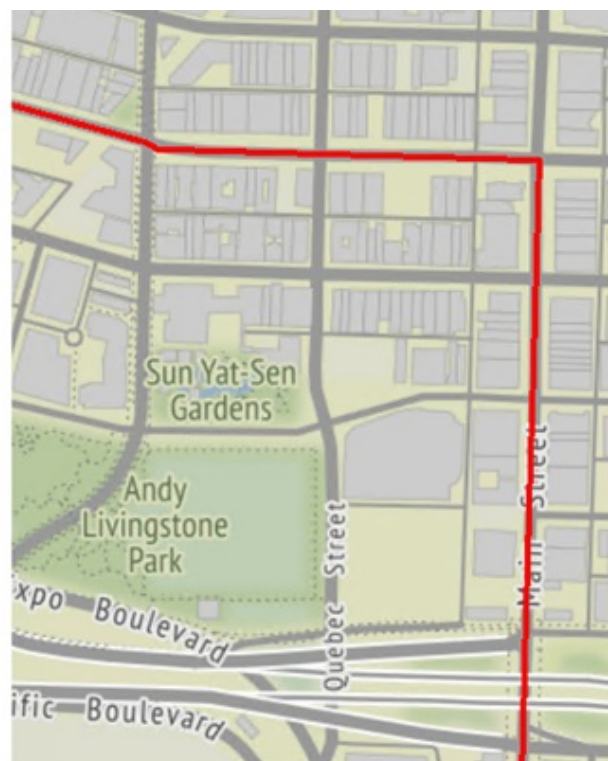
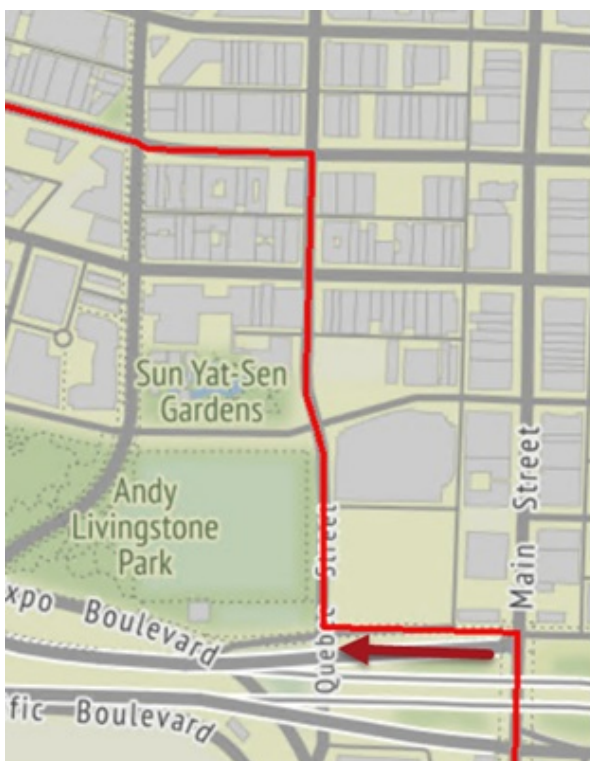
Reverse Cost Attribute: ReverseCost

Allow U-Turns: Yes

Attribute Accumulation

> ☐ Generate List

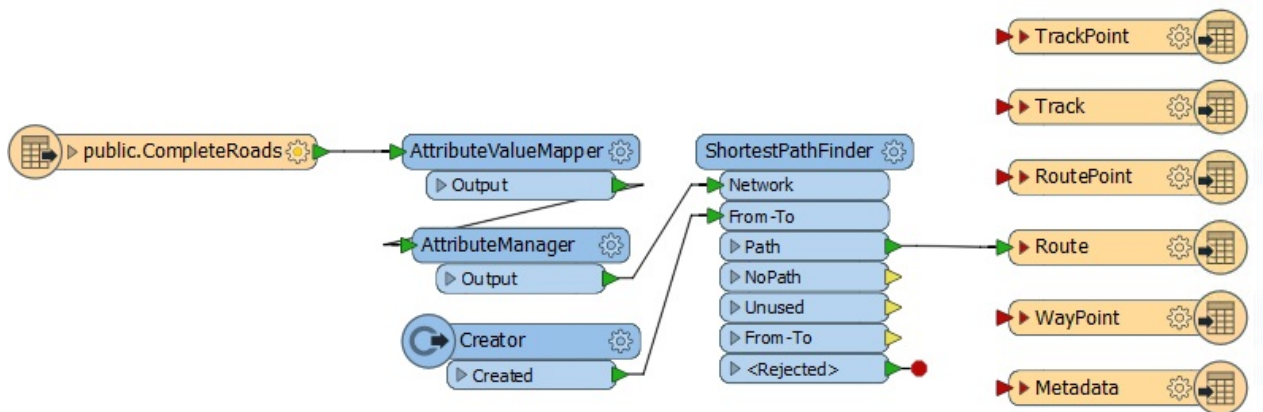
Now we're ready to go. Re-run the workspace and check the output:



There is at least one change (above) caused by a route that previously travelled the wrong-way along a one-way street!

14) Connect Schema

Oh! Don't forget to remove the Inspector transformers and connect the Path port to the Route output feature type:



Now run the workspace, upload the data to your GPS device, and you are ready to go!

Advanced Exercise

Not really advanced, but you did use Best Practice throughout, right? I mean, you have bookmarks and annotations where needed, and no overlapping connections? If not, well you might want to fix that!

CONGRATULATIONS

By completing this exercise you proved you know how to:

- *Create and use an FME database connection*
- *Create a prototype FME workspace using a variety of transformers*
- *Use debugging techniques to find any problems encountered in an exercise*
- *Use a good style for developing workspaces*

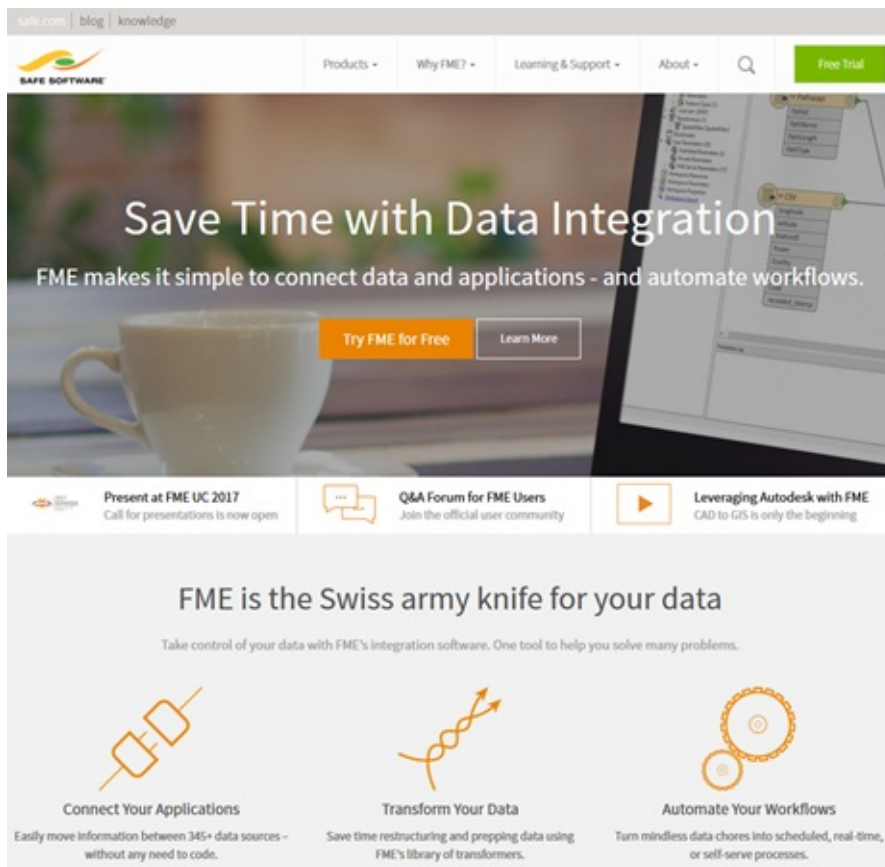
Course Wrap-Up

Although your FME training is now at an end, there is a good supply of expert information available for future assistance.

Product Information and Resources

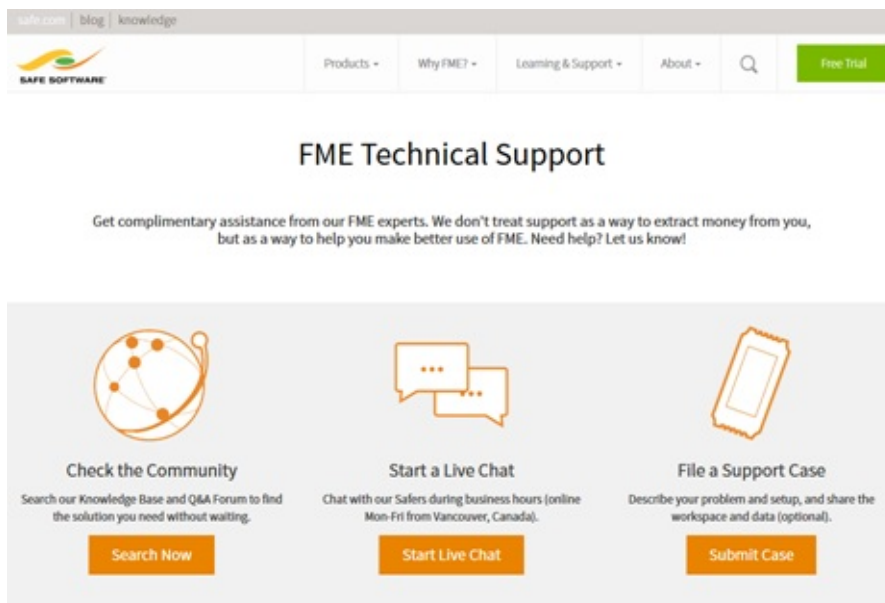
Safe Software Web Site

The [Safe Software web site](#) is the official information source for all things FME. It includes information on FME products, Safe Software services, FME solutions, FME support and Safe Software itself.



Safe Support Team

Behind FME are passionate, fun, and knowledgeable experts, ready to help you succeed, with [a support team](#) philosophy built on the principle of knowledge transfer.



You can request product support through a Support Case (web/email) or using a Live Chat.

Your Local Partner

Safe Software has partners and resellers around the world to provide expertise and services in your region and your language.

You can find a list of official partners on the [Safe Software Partners Page](#).



Safe Software Blog

The [Safe Software blog](#) provides technical information about FME, articles about customers' use cases, and general thoughts on spatial data interoperability.

The Safe Software Blog

About Data
About FME
About Our Customers

About FME | December 15, 2016 | by Claude Vissier

Improving FME Server Performance with NGINX

All new FME Server 2017.0+ and 2016.1.3 instances are now running behind a NGINX reverse proxy in FME Cloud. As a developer on the FME Cloud team, this is something I have been hoping to achieve for a while and with 2016.1.3+ all the pieces are finally in place. In this blog post, I would [...]

[READ MORE](#)

Comment
Share

About FME | December 7, 2016 | by Mark Ireland

FME Parallel Processing: Tips and tricks for generating groups

I recently taught the performance chapter of our FME Desktop advanced training course and got into a conversation with a student about creative ways to use parallel processing. Some of the ideas we came up with about generating 'groups' were so interesting I thought I would share them. I hope you find them of use.

[READ MORE](#)

Comment
Share

About Data | December 5, 2016 | by Tiana Warner

The GeoGeek's Guide to Planet's Satellite Imagery

Small satellite technology is completely changing the world. Here's what Planet is doing to revolutionize Earth Observation data, and how you can leverage satellite trends to get ahead in your industry. Satellites hit a turning point a few years back, when an upsurge in the number being built culminated in 94 launches over the span [...]

[READ MORE](#)

Comment
Share

FME Manuals and Documentation

Use the Help function in FME Workbench to access help and other documentation for FME Desktop. Alternatively, look on our web site under the [Knowledge Center](#) section.

safe.com | blog | knowledge

Questions
Articles
Ideas
Documentation

FME Documentation

FME Desktop

FME Desktop Administrator's Guide

Find out how to install and license your version of FME Desktop, and perform other administrative tasks. (PDF Version)

Getting Started with FME Desktop

Get up and running with FME by browsing through this handy guide. (PDF Version)

FME Readers and Writers

A detailed technical guide to the many reader and writer formats available in FME.

FME Workbench

A guide to FME's primary graphical tool for creating and running data transformations.

FME Transformer Reference Guide (PDF)

A quick reference describing each transformer's functionality.

FME Workbench Transformers

A detailed technical guide to the many transformers available in FME Workbench.

FME Data Inspector

A guide to FME's graphical tool for inspecting transformation results and other datasets.

FME Integration Console

Find out how to extend your "FME-ready" third-party applications so they will integrate with FME Desktop.

FME Quick Translator

Use this tool to perform simple, automatic data conversions.

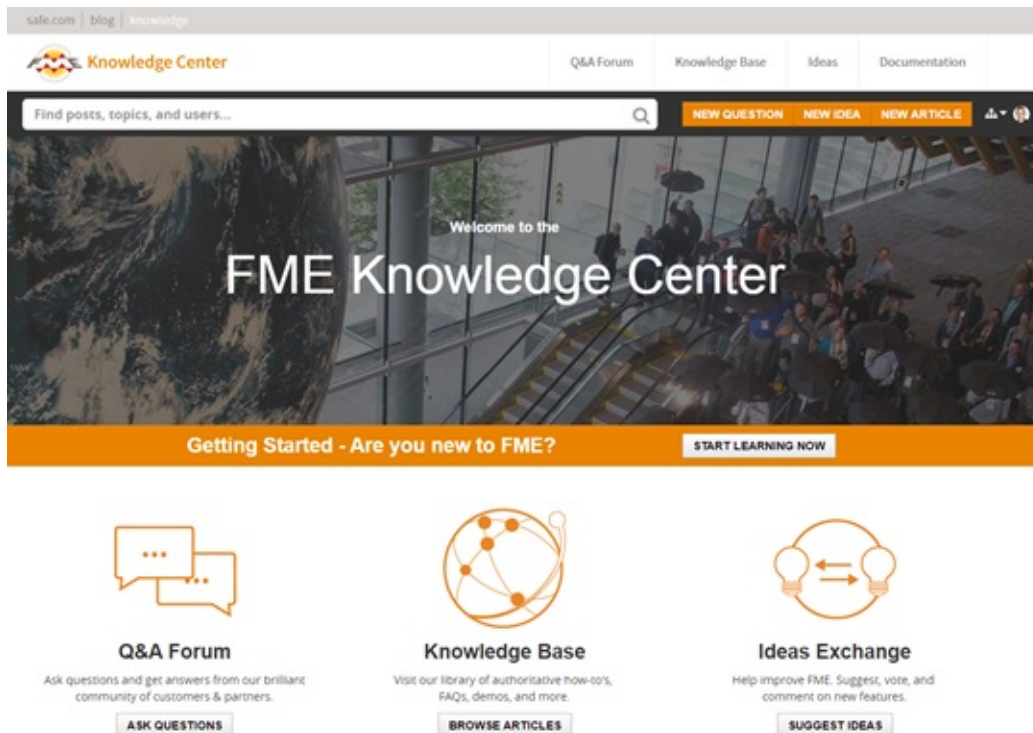
FME Desktop
FME Server
FME Cloud
FME Technical Reference

Community Information and Resources

Safe Software actively promotes users of FME to become part of the FME Community.

The FME Knowledge Center

The **FME Knowledge Center** is our community web site - a one-stop shop for all community resources, plus tools for browsing documentation and downloads.



Knowledge Base

The FME Knowledge Base contains a wealth of information; including tips, tricks, examples, and FAQs. There are sections on both FME Desktop and FME Server, with articles on topics from installation and licensing to the most advanced translation and transformation tasks.

Q&A Forum

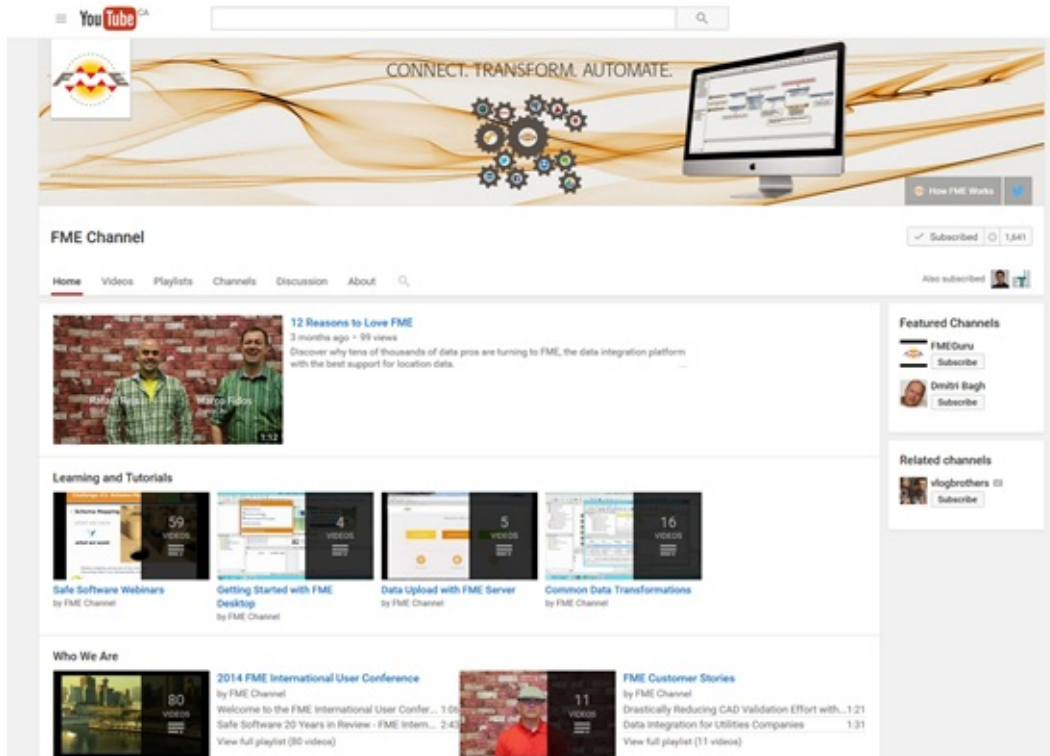
FME community members post FME-related messages, ask questions, and share in answering other users' questions. Members earn "reputation" and "badges" and there is a leaderboard of the top-participating users. Come and see how they can help with your FME projects!

Ideas Exchange

FME development is very much user-driven. The Ideas Exchange gives users the chance to post their ideas for new FME functionality, or improvements to existing functionality, and allows everyone to vote on the proposed ideas. The more votes an idea gets, the more likely it is to be implemented!

The FME Channel

This [FME YouTube channel](#) is for those demos that can only be properly appreciated through a screencast or movie. Besides this there are a host of explanatory and helpful movies, including recordings of most training and tutorials.



Feedback and Certificates

The format of this training course undergoes regular changes prompted by comments and feedback from previous courses.

Course Feedback

Miss Vector says...

There's one final set of questions – and this time you'll be telling me if the answers are correct or not!

Safe Software greatly values feedback from training course attendees and our feedback form is your chance to tell us what you really think about how well we're meeting your training goals.

You can fill in [the feedback form](#) now, but you'll also be reminded by email shortly after your course. Safe Software's partners who carry out training may ask that you fill in a separate form, but you can also use the official Safe Software form if you wish.

Certificates

Mr. E. Dict, (Attorney of FME Law)says...

In order to prove you have taken this training course, a certificate will be emailed automatically to anyone who was logged on for the duration of Safe Software hosted courses.

Thank You

Thank you for attending this FME training course.



Congratulations!

As a reward for reading this far, here's a little challenge for you to try out.

Various folk have something to say to you, but can you figure out what it is? Maybe FME can help?

Miss Vector says...

*.6 si ecnetnes siht rof rebmun edoc ehT .rebmun edoc a dnif ot gnidoced
ralimis sdeen ecnetnes hcaE .ylreporp ti tamrof ot EMF esu ot si egnellahc
eht ,yawyna daer ylbaborp dluoc uoy ecnetnes eno si siht hguhtlA
.snoitalutargnoC*

Dr Workbench says...

*57656C6C20646F6E652E20596F75207265636F676E697A65642074686973
2061732068657820656E636F64656420746578742E204920686F706520796
F75206465636F64656420697420776974682074686520546578744465636F
646572207472616E73666F726D65722E20427920746865207761792C2074
686520636F6465206E756D6265722066726F6D206D652069732039*

Sister Intuitive says...

*11114604017115716504014715716404016415014504016016214516615115
71651630401631451561641451561431450401511640401631501571651541
44040150141166145040142145145156040152165163164040141163040145
14116317104016415704014414514315714414504016415015116304015714
31641411540401641451701640401621451601621451631451561641411641
51157156040165163151156147040164150145040124145170164104145143
15714414516204016416214115616314615716215514516205604012415015
1163040143157144145040156165155142145162040151163040061*

Police Chief Webb-Mapp says...

Lbh ner tbbq, nera'g lbh. Gur pbqr ahzore sbe guvf fragrapr vf 3. Bs pbhefr, gung ahzore nccrnef va pyrne grkg, ohg bayl lbh jvyy xabj gur cnffjbeq gb gur arkg pyhr vf gur svefg sbhe pbqrf pbagnpgrangrq gbtrgure!

Miss Vector says...

C:\FMEDData2017\Resources\Challenge\FinalChallenge.ffs