



FME® Desktop Manual



Document and Copyright Information

Safe Software Inc. makes no warranty either expressed or implied, including, but not limited to, any implied warranties of merchantability or fitness for a particular purpose regarding these materials, and makes such materials available solely on an "as-is" basis.

In no event shall Safe Software Inc. be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of purchase or use of these materials. The sole and exclusive liability of Safe Software Inc., regardless of the form or action, shall not exceed the purchase price of the materials described herein.

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 (data.vancouver.ca). 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

(http://en.wikipedia.org/wiki/Vancouver_Fire_and_Rescue_Services), used under a Creative Commons Attribution-ShareAlike license

Stanley Park GPS Trail: Used with kind permission of VancouverTrails.com. See <http://www.vancouvertrails.com/trails/stanley-park/>.

Copyright

© 2005–2015 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: services@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:

Desktop Manual

Updated:

January 2015

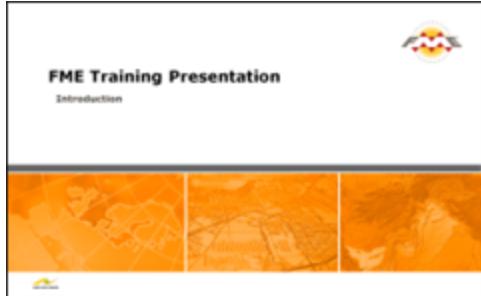
Contents

Chapter 0 - Introduction	7
FME Version	7
Sample Data	7
Course Overview	8
Icons	9
Course Resources	10
Introductions	12
Chapter 1 - Translation Basics	13
What is FME?	13
About FME	13
FME Desktop Components	15
Other FME Products	18
Introduction to FME Workbench	21
Getting Started	27
Introduction to Data Inspection	38
Introduction to the FME Data Inspector	39
Major Components of the FME Data Inspector	39
Using the FME Data Inspector	42
More FME Data Inspector Functionality	51
Translation Previews	64
Module Review	66
Q&A Answers	72
Chapter 2 - Data Transformation	75
What is Data Transformation?	75
Data Transformation Types	75
Structural Transformation	77
Transformation Using Transformers	89
Data Restructuring With Transformers	92
Content Transformation	98
Transformers used in Series	101
Transformers used in Parallel	108
Group-By Processing	114
Data Inspection Using FME Workbench	119
Coordinate System Transformation	122
Module Review	127
Chapter 3 - Best Practice	140
What is Best Practice?	140
Why use Best Practice?	141

Methodology	142
Style	150
Debugging	168
Organization	184
Module Review	190
Chapter 4 - Readers and Writers	203
Key Components	205
Workspaces	213
Readers	218
Removing a Reader	221
Controlling Readers	221
Reader Dataset Parameters	223
Reader Feature Types	225
Removing Feature Types	228
Importing Reader Feature Types	239
Managing Reader Datasets	242
Dealing with Source Feature Types	248
Reader Feature Type Parameters	255
Writers	257
Removing a Writer	259
Controlling Writers	259
Writer Feature Types	261
Removing Feature Types	269
Import Writer Feature Types	270
Writer Feature Type Parameters	271
Module Review	281
Chapter 5 - Practical Transformer Use	294
Transformer Gallery	294
Transformer Searching	297
Most Valuable Transformers	301
Managing Attributes	303
Conditional Filtering	327
Data Joins	341
Constructing Transformer Parameters	357
Module Review	360
Chapter 6 - Course Wrap-Up	371
Product Information and Resources	371
Community Information and Resources	373
Course Feedback	374

Certificates	375
Thank You	376
Congratulations!	377

Chapter 0 - Introduction



This manual is part of a family of FME training materials.

This training course - the FME Desktop Basic Course - is the foundation of all other FME training courses.

It assumes a basic familiarity with the concepts and practices covered by the FME Desktop Tutorial, and covers material that will be required to take almost any other course in the FME training family.

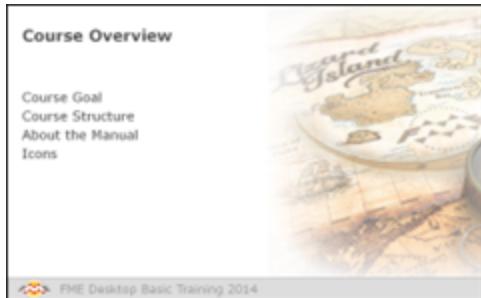
FME Version

This training material is designed specifically for use with FME2015. You may not have some of the functionality described if you use an older version of FME.

Sample Data

The sample data required to carry out the examples and exercises in this document can be obtained from: www.safe.com/fmedata.

Course Overview



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

Course Goal

This training course provides a framework for a basic understanding of FME, upon which a user can base their work. 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 main sections. These sections are:

- Data Translation Basics
- Data Transformation
- FME Best Practice
- Readers and Writers
- Practical Transformer Use

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.

About the Manual

The FME Desktop training manual is yours to keep. It 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.

Icons

In the training manual you may see the following icons...



Additional advice to help apply the knowledge you have learned.



A warning where misuse of FME could lead to difficulties.



Extra challenges for students who are quick to finish and want to take an exercise a little bit further.



A feature new to, or significantly changed in, the most recent version of FME.

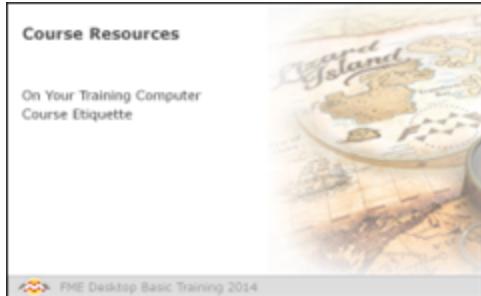


Questions about the course content and how it is applied.



Also, people from the city of Interopolis will also appear from time-to-time to give you advice and dispense FME-related wisdom.

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:\FMEData2015\Data	Datasets used by the City of Interopolis
C:\FMEData2015\Resources	Other resources used in the training
C:\FMEData2015\Workspaces	Workspaces used in the student exercises.
C:\FMEData2015\Output	The location in which to write exercise output.
<documents>\My 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.

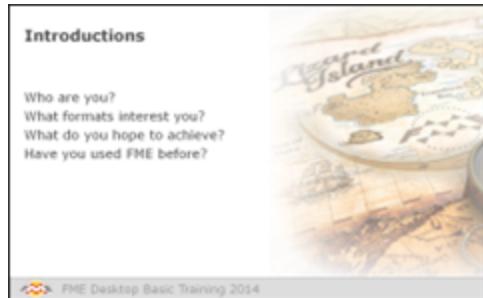
Course Etiquette

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.

Introductions



Safe Software's standalone, web-based, and embedded software products are used by a worldwide network of clients from a wide range of organizations and industries.

Who are you?

What formats interest you?

What do you hope to achieve during this course?

Have you used FME before?

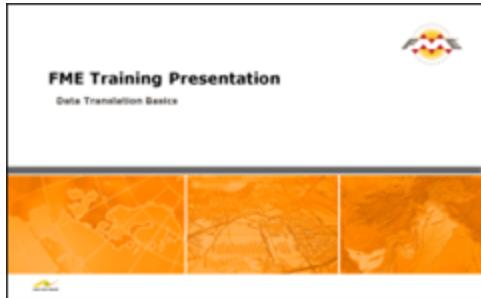
This is your chance to introduce yourself to the rest of the class. You may want to mention the organization you represent, what experience you have using FME, and which formats and functionality you're most interested in.



Please think of the environment and don't print this manual unless you really need to. Using a digital copy or sharing with a colleague will help save paper. And don't forget to recycle printed copies when you are finished with them!



Chapter 1 - Translation Basics



FME is specifically designed for Data Transformation. Its key characteristics illustrate why it is so suitable for this role.

What is FME?

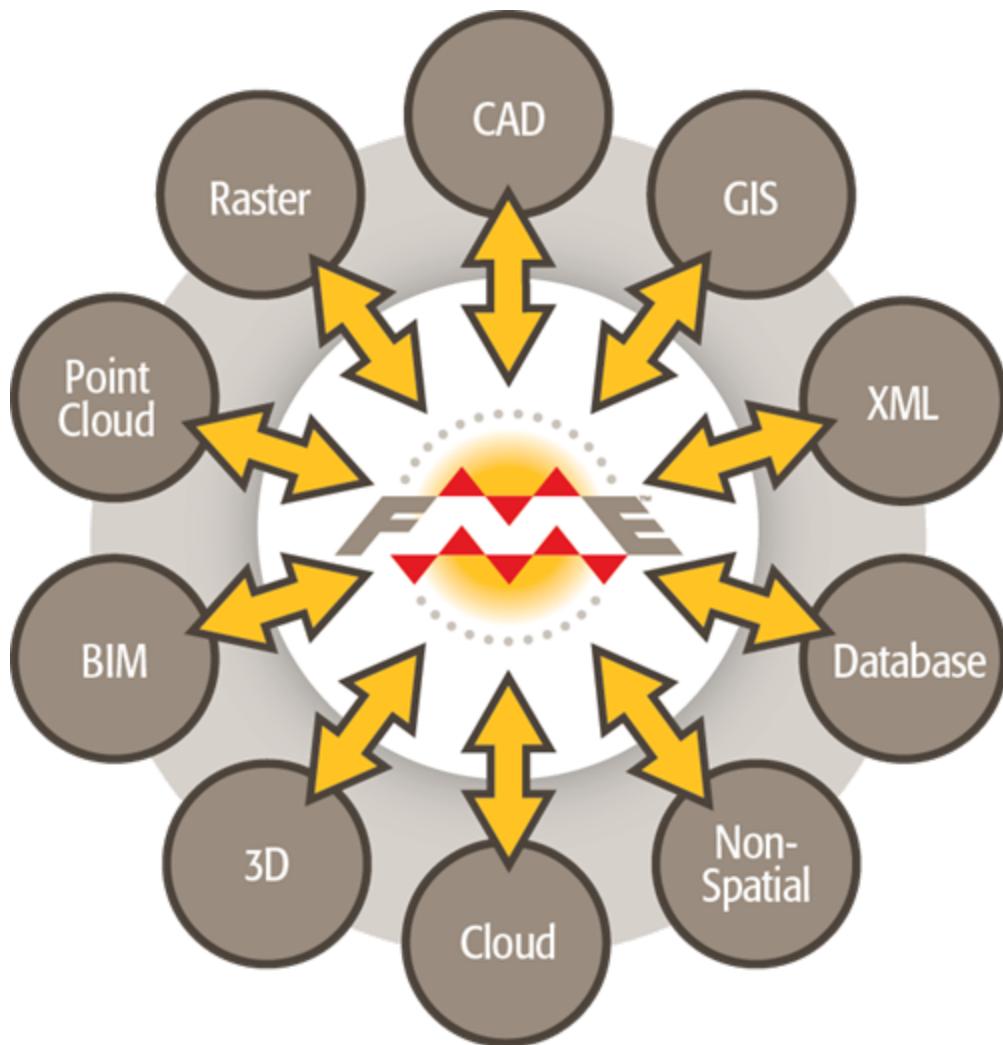
FME (the Feature Manipulation Engine) is a data translation and transformation tool for solving problems of data interoperability. *Interoperability* is about communication; in our case communication by the sharing and distribution of data, and the ability to use that data transparently.

FME is sometimes classed as a Spatial ETL application. ETL stands for *Extract, Transform and Load*. It is a data warehousing tool that extracts data from a source, transforms it to fit the users' needs and loads it into a destination or data warehouse.

While an ETL tool will process the various column types that are in a non-spatial database or system, a Spatial ETL tool must also have the spatial operations - geoprocessing capabilities that change the structure and representation of spatial data – needed to move data from one spatial database or GIS to another.

About FME

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.



This capability is made possible by a rich data model designed to cover all possible geometry and attribute types. When limitations in the destination (output) format cause incompatibility, FME automatically compensates to create a seamless translation process.

FME also provides tremendous transformation functionality, resulting in output that can be much greater than the sum of the inputs. This functionality is exposed to the user through an ingenious and versatile graphic interface.

FME Desktop Components



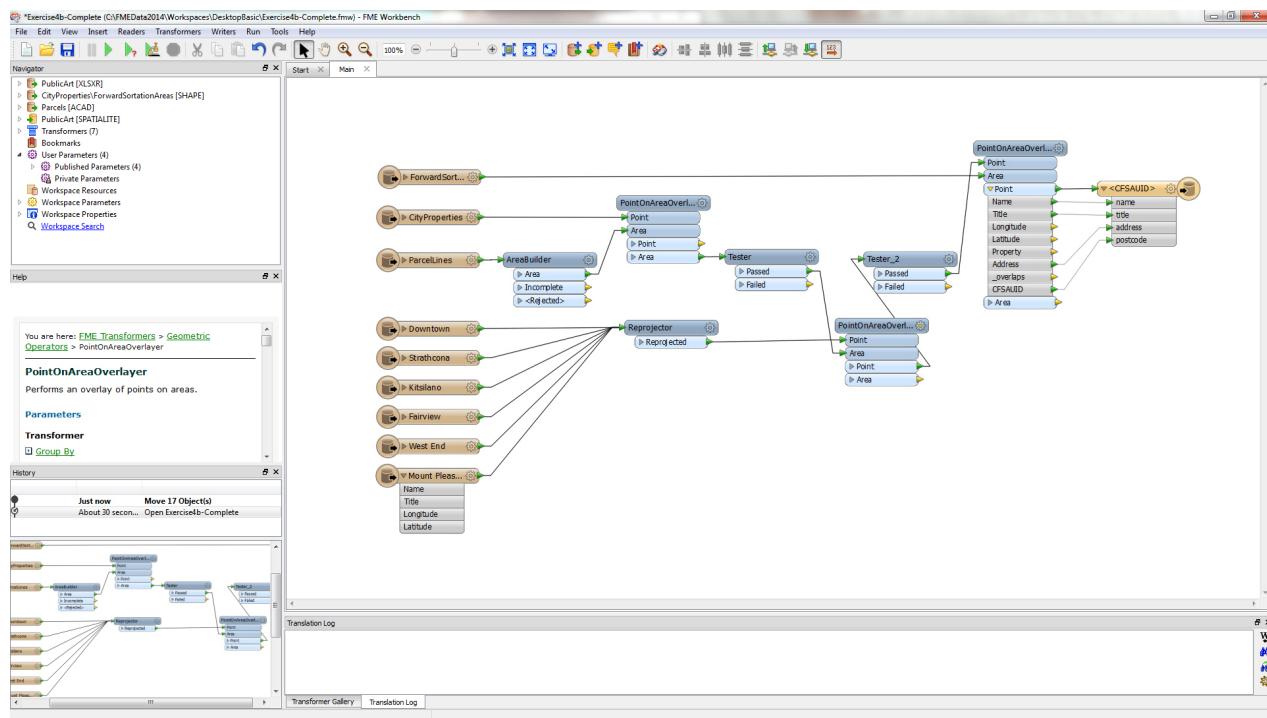
FME comprises a number of spatial data handling components. Everything here is included with every edition of FME Desktop.

FME Applications

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

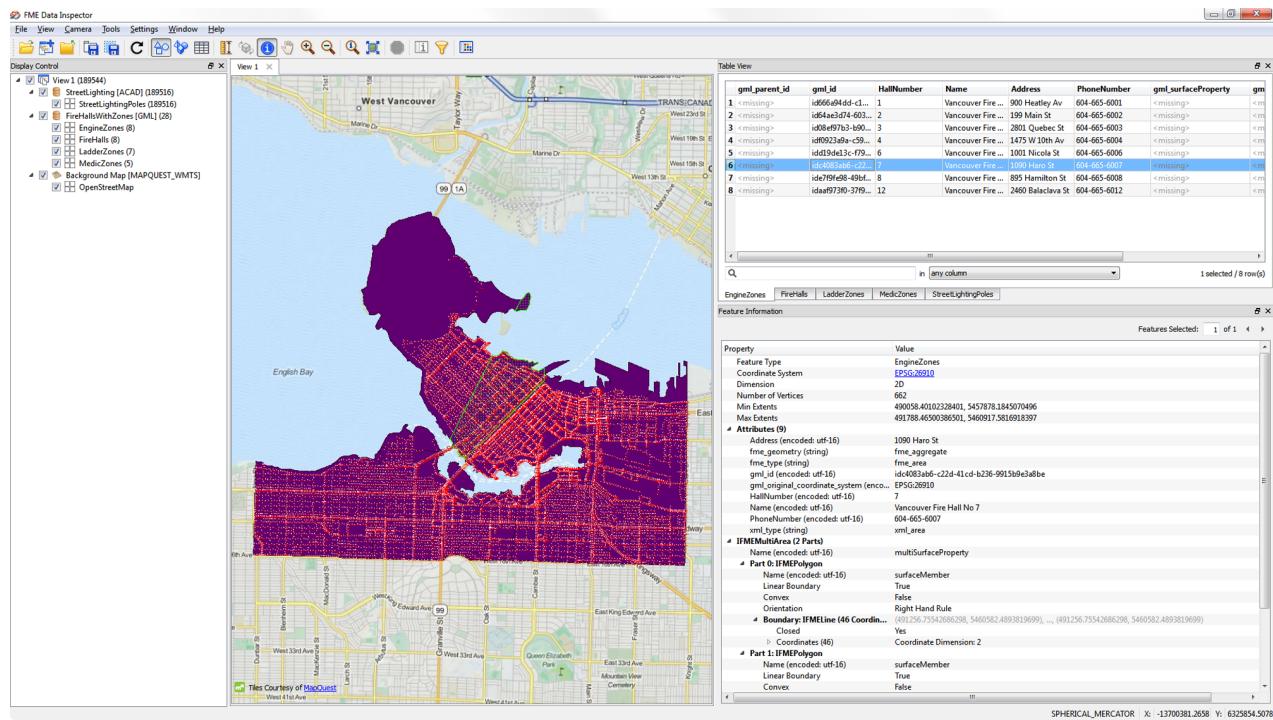
FME Workbench

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

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 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.

Other FME Products



The FME brand name covers other products than FME Desktop.

FME Server

FME Server is a scalable, networked data translation and transformation application.

FME Server:

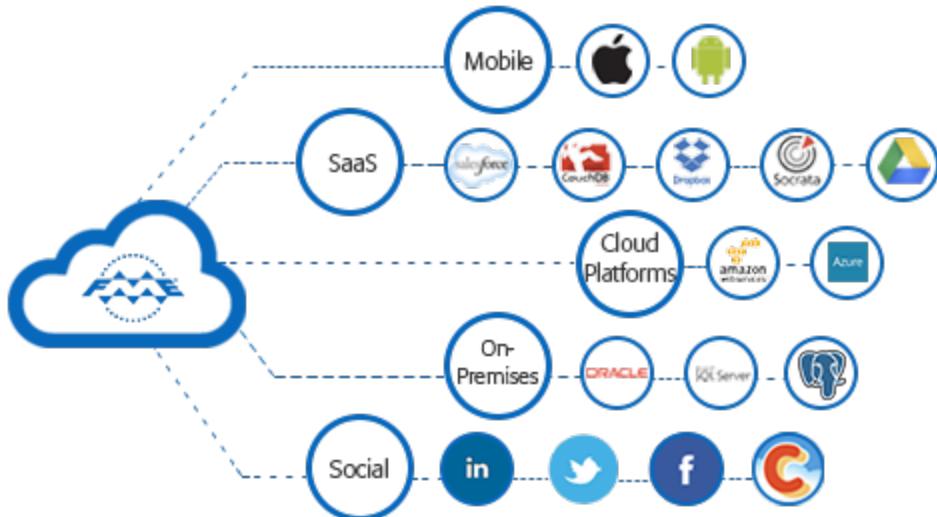
- Allows Desktop users to share translation resources through a repository mechanism
- Allows Desktop users to run resource intensive translations on a dedicated server
- Allows non-FME users to run translations on demand
- Allows translation output to be streamed directly to a chosen spatial application



See <http://fmeserver.com/> for more information.

FME Cloud

FME Cloud is a solution for running FME Server as a web service, on a pre-defined virtual machine, using a "pay-as-you-go" business model.



See <https://fmecloud.com/> for more information



FME Extensions

Besides the basic functionality of FME Desktop, there are a number of optional extra extensions that may be purchased. These extensions add to either the functionality or format reach of the basic FME product.

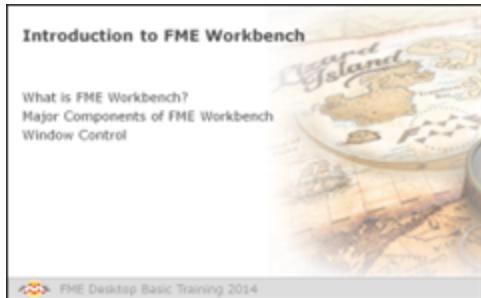
See <http://www.safe.com/solutions/specialty/> for more information.

FME Store

The FME Store is a marketplace for FME components that also add functionality or formats to FME. Offerings include both free and licensed products, provided by either Safe Software or 3rd party partners.

See <http://fmestore.safe.com> for more information.

Introduction to FME Workbench



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

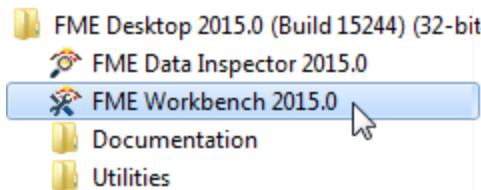
What is FME Workbench?

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.

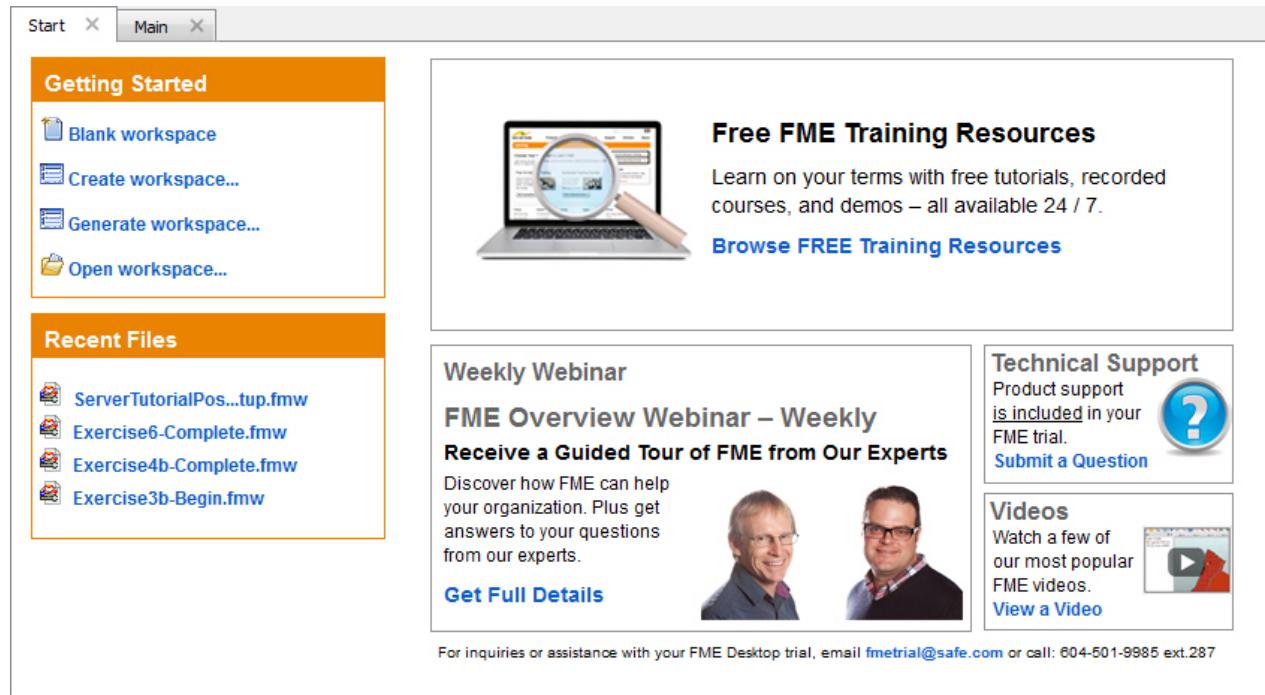
Workbench is fully integrated to interact with other FME Desktop applications such as the FME Data Inspector and other products such as FME Server and FME Cloud.

Starting FME Workbench

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



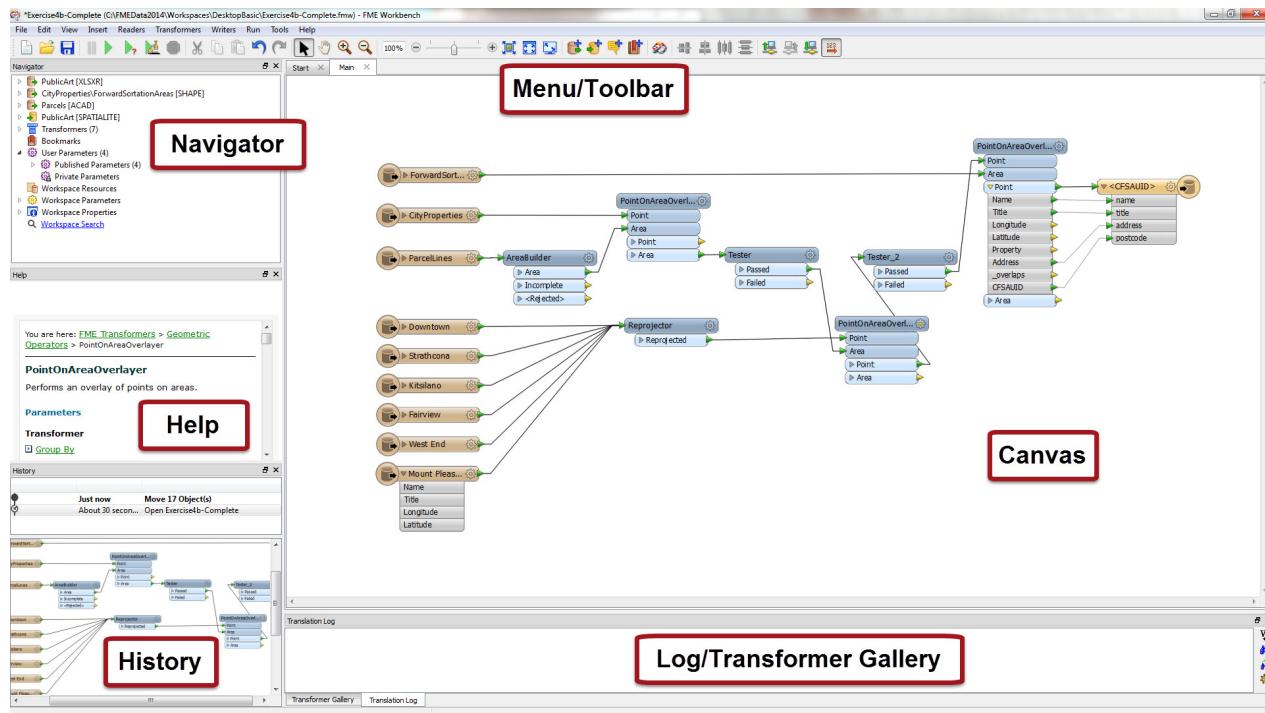
The startup tab links to a live web page and therefore the display will change over time as new information and resources are shown.



A second tab – Main – displays a canvas where the actual translation will be graphically defined.

Major Components of FME Workbench

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



Menu bar and Toolbar

The menu bar and toolbar contain a number of tools: for example, tools for navigating around the workspace, 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...

To be precise the application itself is called 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.

Translation Log

The log window (translation log) shows a report on translation results. Information includes any warning or error messages, translation status, length of translation, and number of features processed.



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.

Window Control

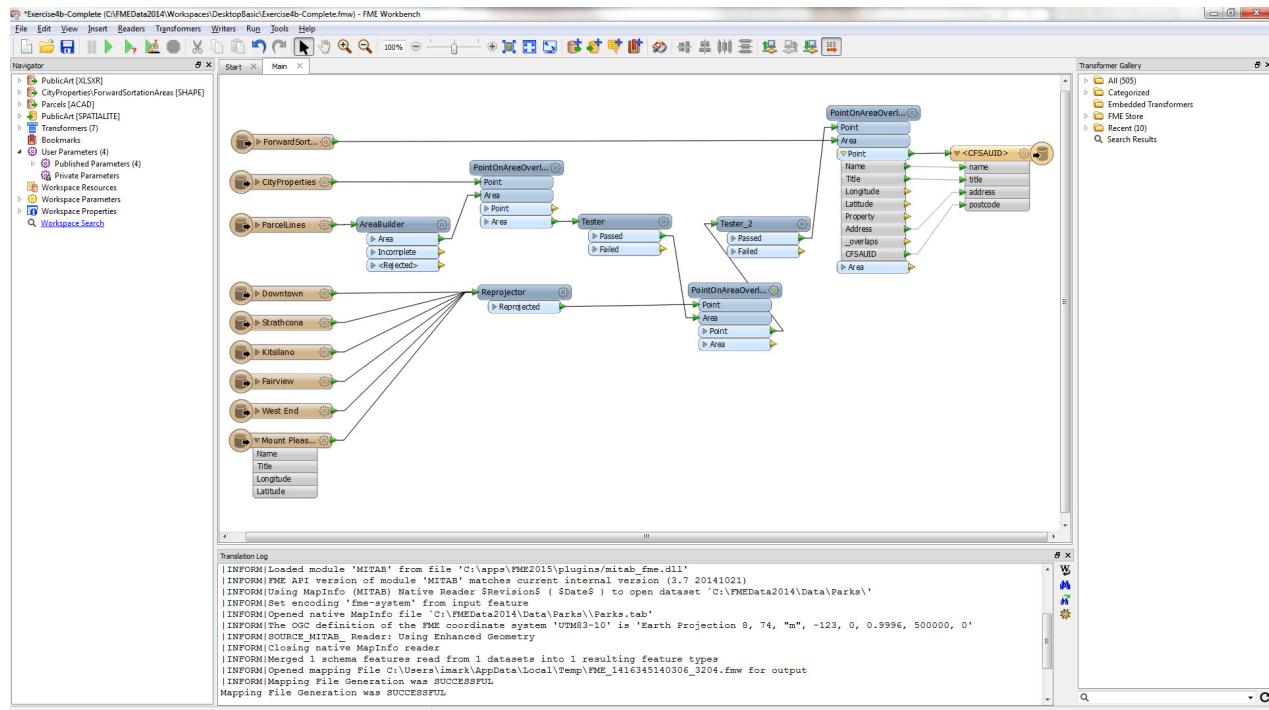
All windows in the Workbench interface can be detached from a fixed position and deposited in a custom location by clicking on the frame of the window and dragging it into a new position. The windows can even float outside of the main Workbench window.

Windows can be docked within Workbench by dragging them onto the Workbench window frame. Windows can be docked to either the left, right, upper or lower boundaries of the Workbench frame.

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

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

This user has opted to dock the Navigator and Transformer Gallery on the left and right sides of Workbench respectively. The Log window is docked in its traditional position at the bottom of the window and the Help and History windows have been closed altogether.



FireFighter Mapp says...

"Here's a hot tip for you. Don't feel put out by a lack of canvas space. Pressing F11 instantly dispatches the lesser-used windows to one side and expands the canvas window to an alarming size!"



Miss Vector says...

'Attention please! It's time for a quiz to see what you've learned so far. Turn to a fellow student and answer these questions between you.'

FME Desktop 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

Which of the following applications are parts of FME Desktop?

- 1) FME Workbench
- 2) FME Server
- 3) FME Quick Translator
- 4) FME Data Inspector

Which of the following windows are on the Workbench interface?

- 1) Navigator
- 2) Transformer Gallery
- 3) Log Window
- 4) Display Control Window

Getting Started



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

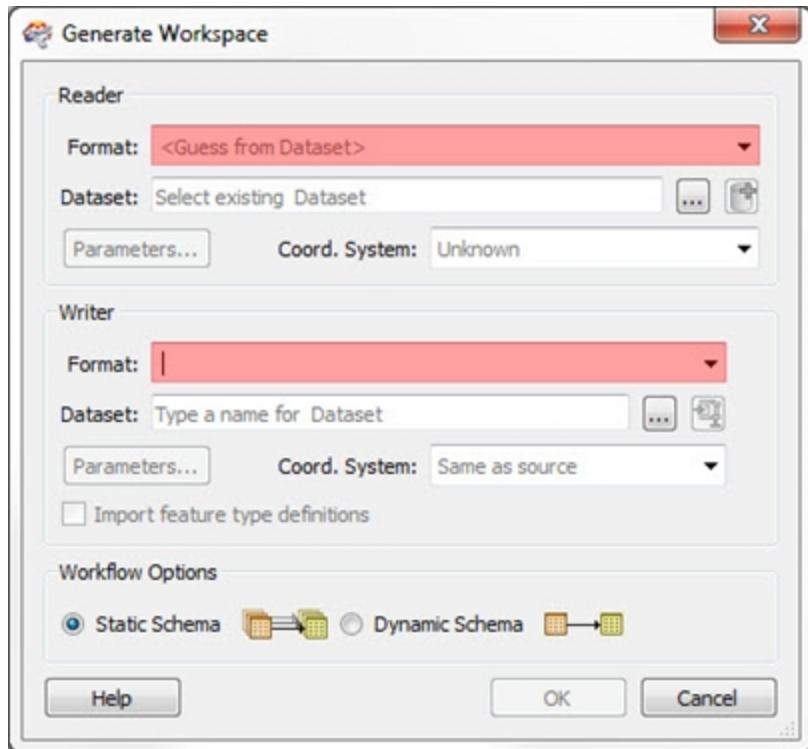
The Start Tab in FME Workbench includes a section called "Getting Started." This section has links to the different ways in which a workspace can be created. The simplest method is Generate Workspace.



Generate Workspace Dialog

The Generate Workspace dialog condenses all the choices to be made into a single dialog box.

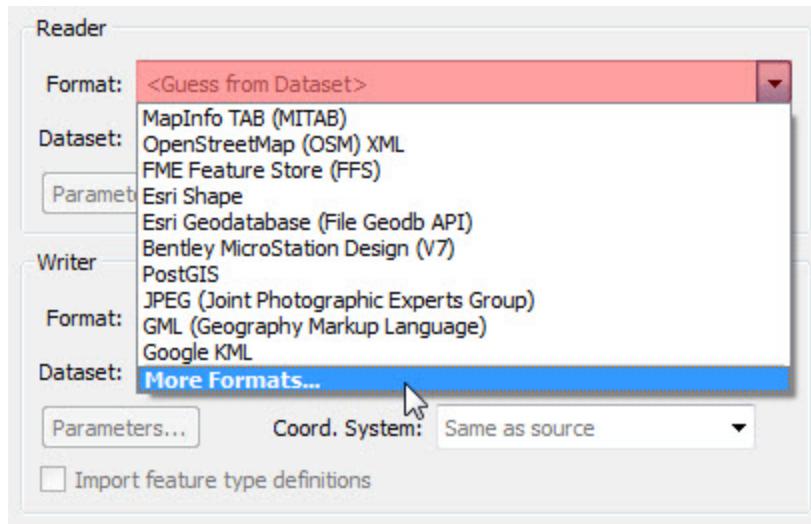
The Generate Workspace dialog has fields for defining the format and location of both the data to be read, and the data to be written.



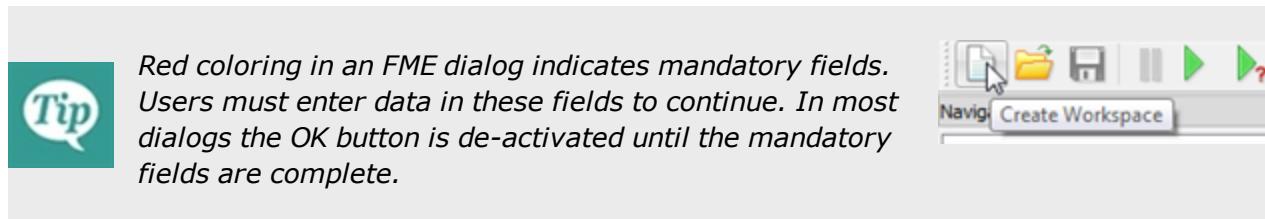
All format selection fields in FME are both a pull-down menu and a text entry field. The drop-down list shows the last ten formats used, so favourite formats are instantly available.

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.

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.

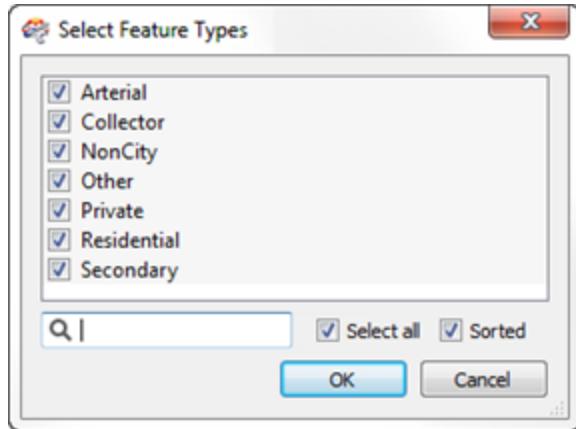


Feature Types Dialog

Whenever a source dataset contains multiple layers the user is 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 within the workspace.



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.

The New Workspace

A new workspace reads from left to right, from the layers in a source (Reader), through a dataflow to the layers in a destination (Writer). One could also think of these as the Extract-Transform-Load stages of a spatial ETL process.

A new workspace resembles this example.



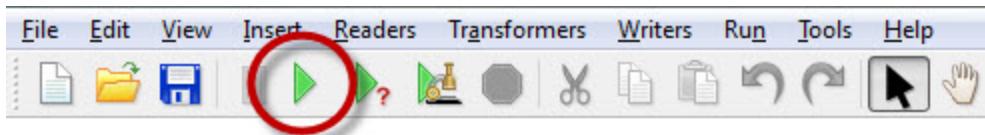
Arrows denote the direction of data flow, from source to destination.



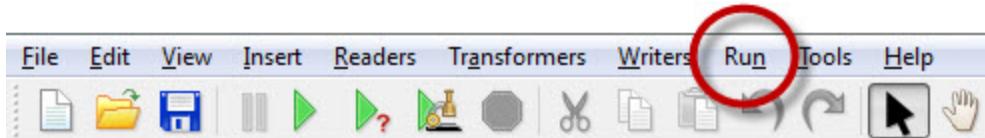
Terminology: In most cases FME uses the terms 'Reader' and 'Writer' instead of 'Source' and 'Destination.' A later chapter explains the why. For now, just be aware that a Reader reads datasets and a Writer writes datasets, and these terms are analogous to source/destination and input/output.

Running the Workspace

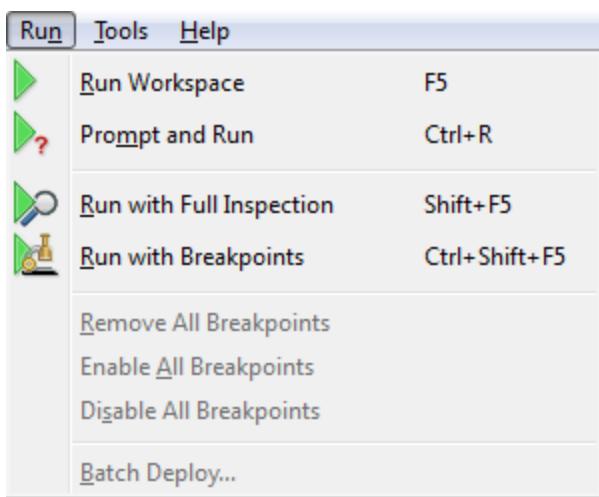
The green arrow (or 'play' button) on the Workbench toolbar starts a translation.



There are also options under Run on the menubar.

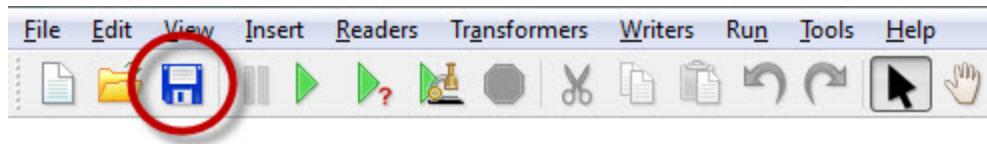


The Run menu with run options include shortcut keys that can be used – the F5 key to simply run a translation and Ctrl+R to prompt and run a translation.



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:



Again there are 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."



Firefighter Mapp says...

'File > Open Recent' shows a list of previously used workspaces. This list is expandable to show up to a towering 15 entries.'



The 'Run' option carries out a translation using the same parameters and settings used previously. The 'Prompt and Run' option prompts for new values for parameters and settings.

Regardless of this, however, the 'Run' option must still prompt for parameters that have not yet been filled in and don't have default values.

Workspace Results

After running a workspace related information and statistics are found 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
|=====
|Arterial           957
|Collector          8
|NonCity           26
|Other              43
|Private            191
|Residential        2507
|Secondary          263
|=====
|Total Features Read    3995
|=====

|=====
|          Features Written Summary
|=====
|Arterial           957
|Collector          8
|NonCity           26
|Other              43
|Private            191
|Residential        2507
|Secondary          263
|=====
|Total Features Written 3995
|=====

|DESIGN READER: Closing DGN V8 file
|Translation was SUCCESSFUL with 7 warning(s) (3995 feature(s) output)
|FME Session Duration: 1.1 seconds. (CPU: 0.8s user, 0.3s system)
```

In this example the log file reveals that 3995 features were read from a MicroStation DGN file.

These features were written to a MapInfo TAB dataset..

The overall process was a success (with some warnings).

The elapsed time for the translation was 1.1 seconds.

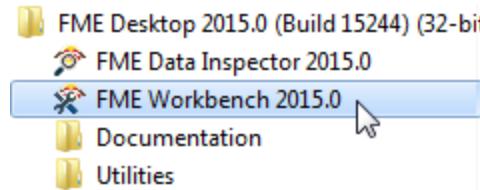
Basic Workspace Creation

Exercise 1a Basic Workspace Creation	
Scenario	FME Workspace Author
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
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise1a-Complete.fmw</i>

Let's start using FME to do a simple file format translation. We've outlined all of the actions you need to take – although FME's interface is so intuitive you might be able to carry out the exercise without the need for 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 2015.0 > FME Workbench 2015.0**.



2) Begin Setup

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



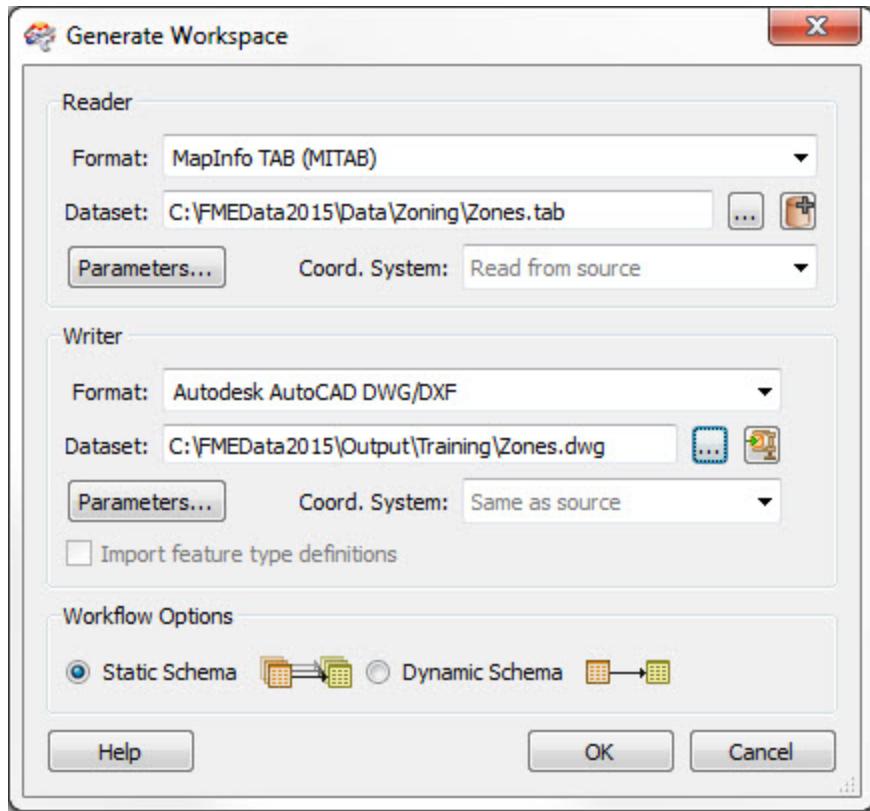
3) Set up Translation

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:\FMEData2015\Data\Zoning\Zones.tab
Writer Format	Autodesk AutoCAD DWG/DXF
Writer Dataset	C:\FMEData2015\Output\Training\Zones.dwg

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 Workspace

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



Click the arrow icon on each object to expose a list of attributes in the data.

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 report 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: 0.9 seconds. (CPU: 0.5s user, 0.3s system)
|END - ProcessID: 11728, peak memory usage: 105976 kB, current memory usage: 105872 kB
```

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.

Name	Date modified	Type	Size
 Zones.dwg	19/11/2014 10:31 AM	DWG TrueView Drawing	187 KB

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! You have now:

- Created and run an FME workspace

Introduction to Data Inspection



Inspecting data prior to, during, or after the translation is a helpful way to verify that the process is operating as expected.

What is 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 facets to spatial data may be inspected, including the following:

- **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?
- **Data Format:** Is the data in the expected format?
- **Data Schema:** Is the data subdivided into the correct layers, categories or classes?
- **Data Quantity:** Does the data contain the correct number of features?
- **Process Output:** Has the translation process restructured the data as expected?



Chef Bimm says...

'I have a great recipe for loading CAD files into a Building Information Model. Previewing the ingredients... I mean data... 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.'

Introduction to the FME Data Inspector



The FME Data Inspector is a utility for viewing and examining both spatial and tabular datasets.

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 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.

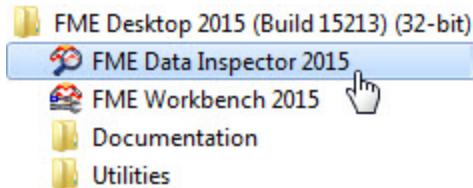
The FME Data Inspector is closely tied to FME Workbench so that Workbench can send data directly to the Inspector.

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.

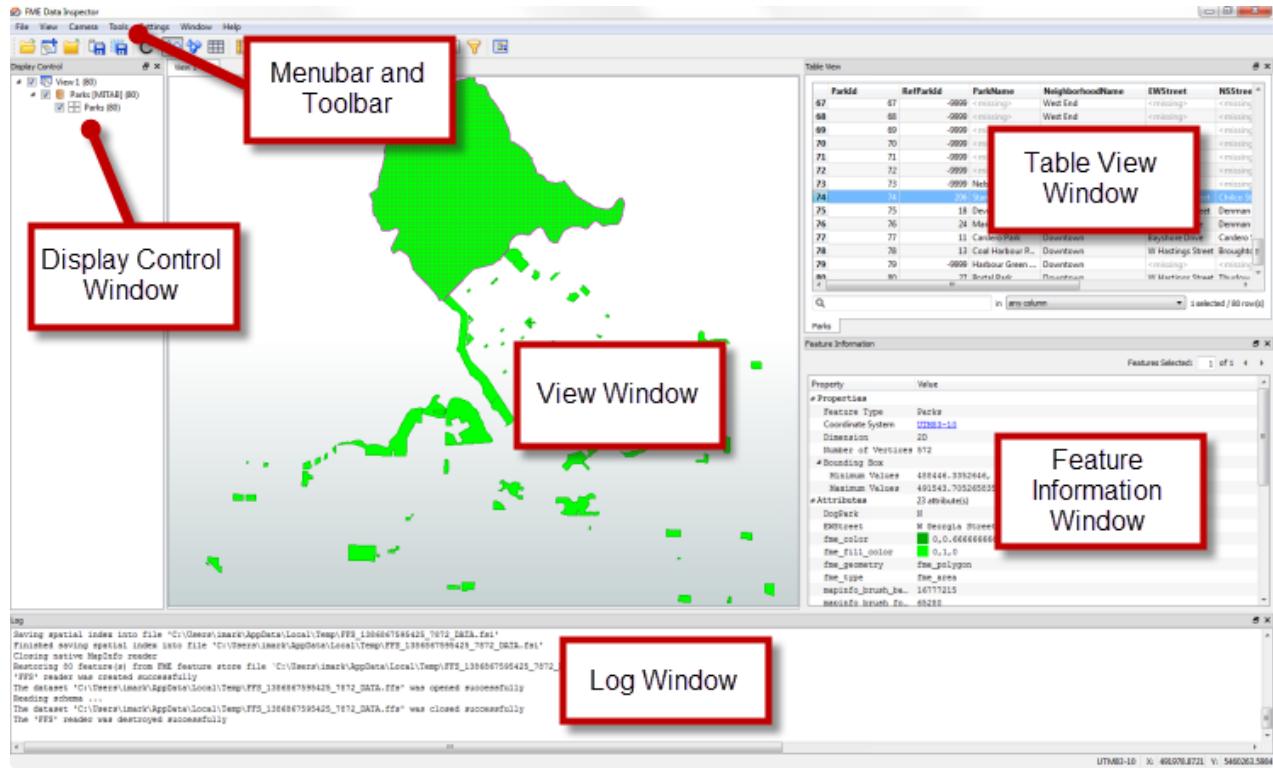
Starting the FME Data Inspector

To start the FME Data Inspector, from the Windows start menu click **FME Desktop 2015**, then on the sub-menu click **FME Data Inspector 2015**.



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.

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).

Log Window

The Log window 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.

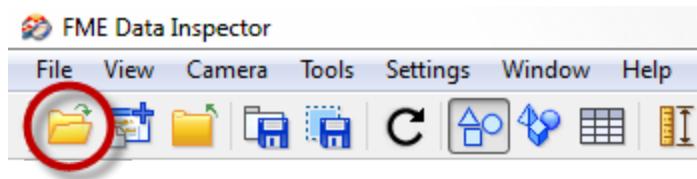
Viewing Data

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.

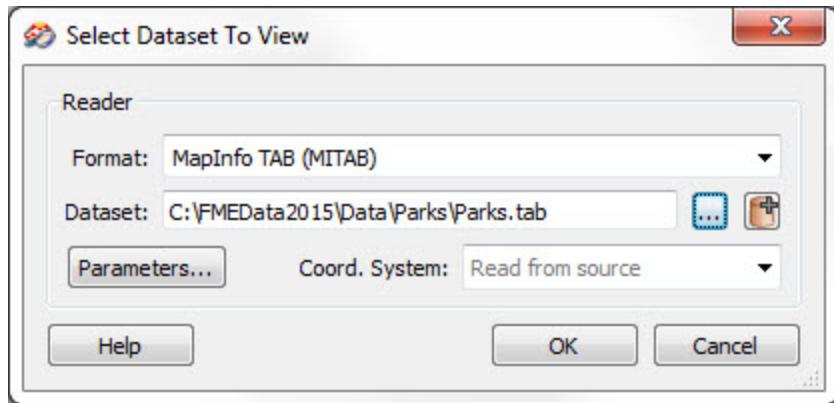
- Selecting **File > Open Dataset** from the menu bar



- Selecting the toolbar button Open Dataset.
- Dragging and Dropping a file onto any window (*except the View window*)
- 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.'

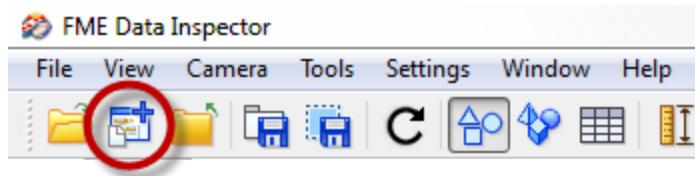
All of these methods cause a dialog to open in the FME Data Inspector in which to define the dataset to view.



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*

Inspecting Data

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



Windowing tools are:

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

Querying tools are

- Query individual feature(s)



- Measure a distance within a View window



By default the Query tool is active when you start the FME Data Inspector. Clicking the toolbar Query button at this point only turns the tool off.

The results of a feature query are shown in the 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

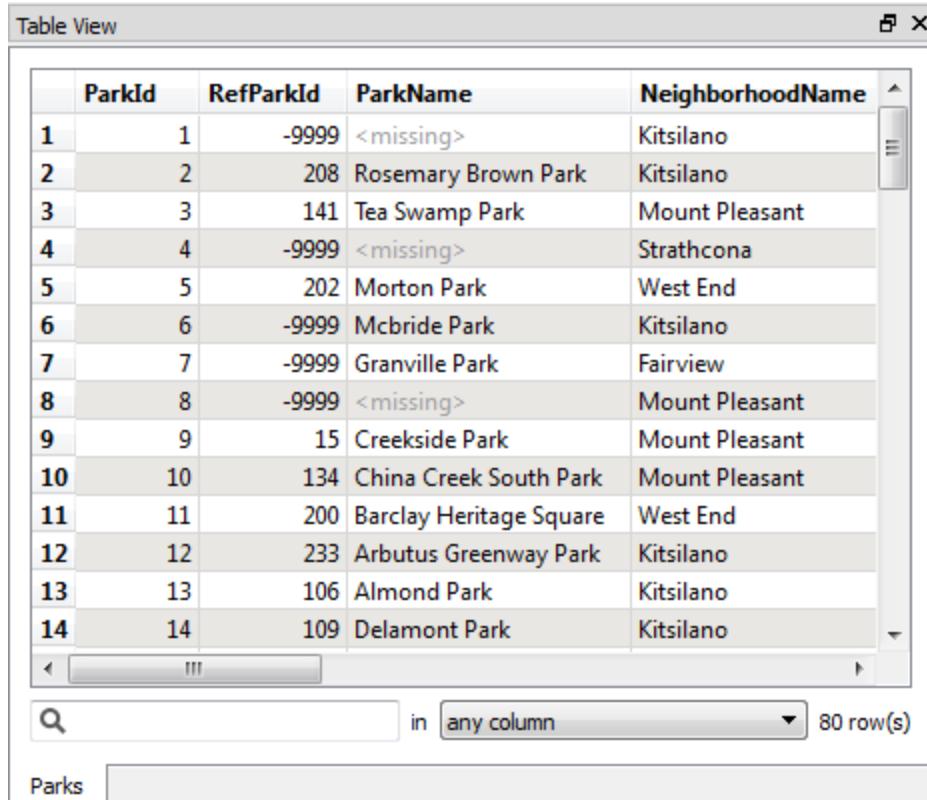
Features Selected: 1 of 1

Property	Value
Feature Type	Parks
Coordinate System	UTM83-10
Dimension	2D
Number of Vertices	572
Min Extents	488446.33526460017, 5459665.4108974617
Max Extents	491543.70526583528, 5462384.0675227698
Attributes (23)	
DogPark (encoded: fme-system)	N
EWStreet (encoded: fme-system)	W Georgia Street
fme_color (string)	 0,0,6666666666666667,0
fme_fill_color (string)	 0,1,0
fme_geometry (string)	fme_polygon
fme_type (string)	fme_area
mapinfo_brush_background (32 bit unsigned int)	16777215
mapinfo_brush_foreground (32 bit unsigned int...)	65280
mapinfo_brush_pattern (32 bit integer)	2
mapinfo_brush_transparent (string)	false
mapinfo_centroid_x (64 bit real)	489995.0202652177
mapinfo_centroid_y (64 bit real)	5461024.7342111338
mapinfo_pen_color (32 bit unsigned integer)	43520
mapinfo_pen_pattern (32 bit integer)	2
mapinfo_pen_width (32 bit integer)	1
mapinfo_type (string)	mapinfo_region
NeighborhoodName (encoded: fme-system)	West End
NSStreet (encoded: fme-system)	Chilco Street
ParkId (string)	74
ParkName (encoded: fme-system)	Stanley Park
RefParkId (string)	206
SpecialFeatures (encoded: fme-system)	Y
Washrooms (encoded: fme-system)	Y
IFMEPolygon	
Linear Boundary	Yes
Convex	No
Orientation	Right Hand Rule
Boundary: IFMELine (572 Coordinates)	
Closed	Closed In 2D
Coordinates (572)	
0	491042.11880315107, 5460549.0710309977
1	490995.1587198851, 5460561.4685075274
2	490991.53054839483, 5460562.1883610031
3	490970.70979154733, 5460565.227742346
4	490949.70762612531, 5460566.4474940691
5	490928.6807231704, 5460565.8276202427
6	490907.78575372422, 5460563.3781188317
7	490896.81053496618, 5460560.4887069631
8	490877.57298022351, 5460553.9900297495
9	490858.97035549162, 5460545.841688321
10 - 561	Show All or 20 , 200 more
562	491064.51451625914, 5460535.7237477992
563	491061.59548737836, 5460536.2836338356
564	491058.73417940759, 5460537.0834710309
565	491055.96357572416, 5460538.1432553148
566	491053.2836763279, 5460539.4329927927
567	491050.73571044044, 5460540.9526834637
568	491048.3196780617, 5460542.6923293639
569	491046.06856256892, 5460544.6319345627
570	491043.9988556506, 5460546.7615010962
571	491042.11880315107, 5460549.0710309977

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.

Also available is a window called the Table View.



The screenshot shows the 'Table View' window with the title bar 'Table View'. The main area is a grid table with columns: ParkId, RefParkId, ParkName, and NeighborhoodName. The data rows are numbered 1 to 14. Row 1 has ParkId 1, RefParkId -9999, ParkName <missing>, and NeighborhoodName Kitsilano. Row 2 has ParkId 2, RefParkId 208, ParkName Rosemary Brown Park, and NeighborhoodName Kitsilano. Row 3 has ParkId 3, RefParkId 141, ParkName Tea Swamp Park, and NeighborhoodName Mount Pleasant. Row 4 has ParkId 4, RefParkId -9999, ParkName <missing>, and NeighborhoodName Strathcona. Row 5 has ParkId 5, RefParkId 202, ParkName Morton Park, and NeighborhoodName West End. Row 6 has ParkId 6, RefParkId -9999, ParkName McBride Park, and NeighborhoodName Kitsilano. Row 7 has ParkId 7, RefParkId -9999, ParkName Granville Park, and NeighborhoodName Fairview. Row 8 has ParkId 8, RefParkId -9999, ParkName <missing>, and NeighborhoodName Mount Pleasant. Row 9 has ParkId 9, RefParkId 15, ParkName Creekside Park, and NeighborhoodName Mount Pleasant. Row 10 has ParkId 10, RefParkId 134, ParkName China Creek South Park, and NeighborhoodName Mount Pleasant. Row 11 has ParkId 11, RefParkId 200, ParkName Barclay Heritage Square, and NeighborhoodName West End. Row 12 has ParkId 12, RefParkId 233, ParkName Arbutus Greenway Park, and NeighborhoodName Kitsilano. Row 13 has ParkId 13, RefParkId 106, ParkName Almond Park, and NeighborhoodName Kitsilano. Row 14 has ParkId 14, RefParkId 109, ParkName Delamont Park, and NeighborhoodName Kitsilano. At the bottom of the window, there is a search bar with a magnifying glass icon and the text 'in any column', a dropdown menu set to '80 row(s)', and a tab labeled 'Parks'.

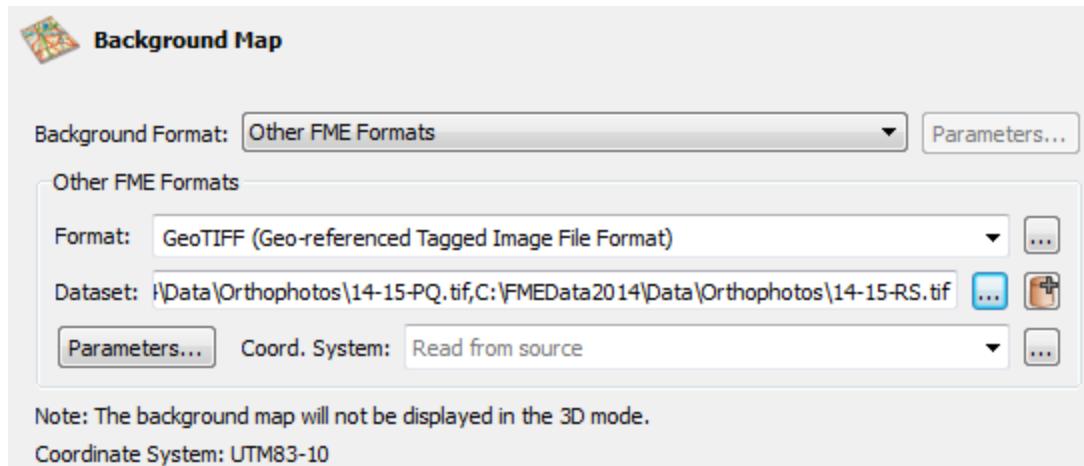
ParkId	RefParkId	ParkName	NeighborhoodName
1	1	-9999 <missing>	Kitsilano
2	2	208 Rosemary Brown Park	Kitsilano
3	3	141 Tea Swamp Park	Mount Pleasant
4	4	-9999 <missing>	Strathcona
5	5	202 Morton Park	West End
6	6	-9999 McBride Park	Kitsilano
7	7	-9999 Granville Park	Fairview
8	8	-9999 <missing>	Mount Pleasant
9	9	15 Creekside Park	Mount Pleasant
10	10	134 China Creek South Park	Mount Pleasant
11	11	200 Barclay Heritage Square	West End
12	12	233 Arbutus Greenway Park	Kitsilano
13	13	106 Almond Park	Kitsilano
14	14	109 Delamont Park	Kitsilano

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.

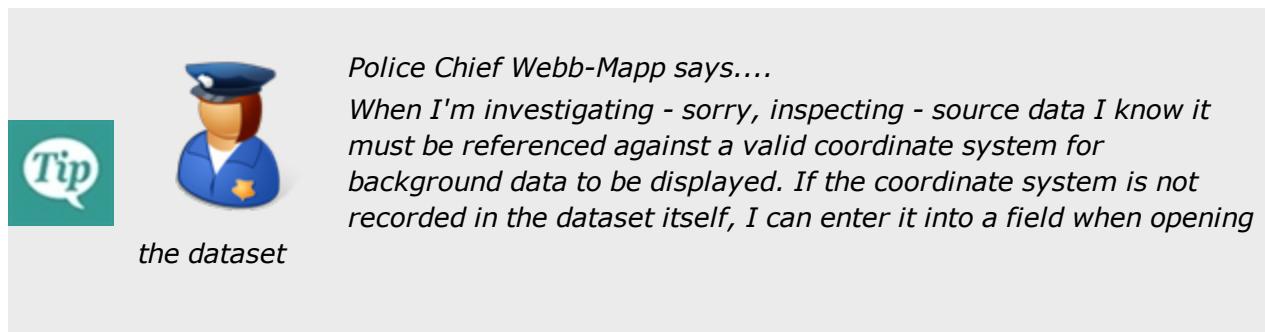
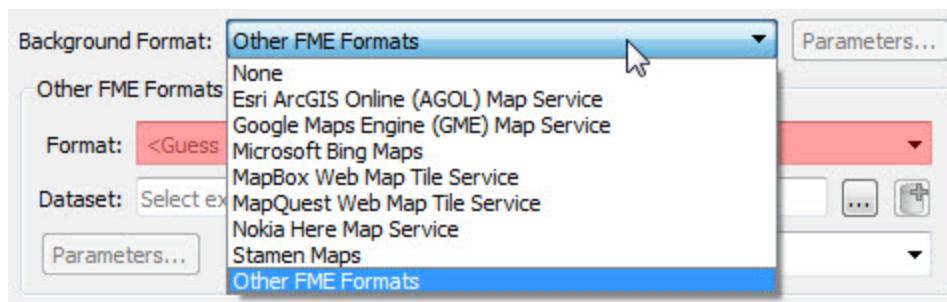
Background Maps

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:





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.

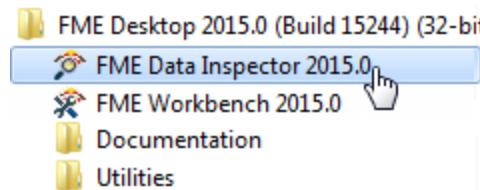
Basic Data Inspection

Exercise 1b Basic Data Inspection	
Scenario	FME Workspace Author
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	None
Finished Workspace	None

Now let's see how the FME Data Inspector interface works by inspecting the output from the quick translation in Exercise 1a.

1) Start FME Data Inspector

Start FME Data Inspector by selecting it from the Windows start menu. You'll find it under **Start > All Programs > FME Desktop 2015.0 > FME Data Inspector 2015.0**.



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:

Format Autodesk AutoCAD DWG/DXF
Dataset C:\FMEData2015\Output\Training\Zones.dwg

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.



FME makes it easy to locate either input or output data from within FME Workbench. Simply right-click on any Reader or Writer feature type and choose the option "Open Containing Folder."



Congratulations! You have now:

- Inspected the output from an FME workspace translation

More FME Data Inspector Functionality

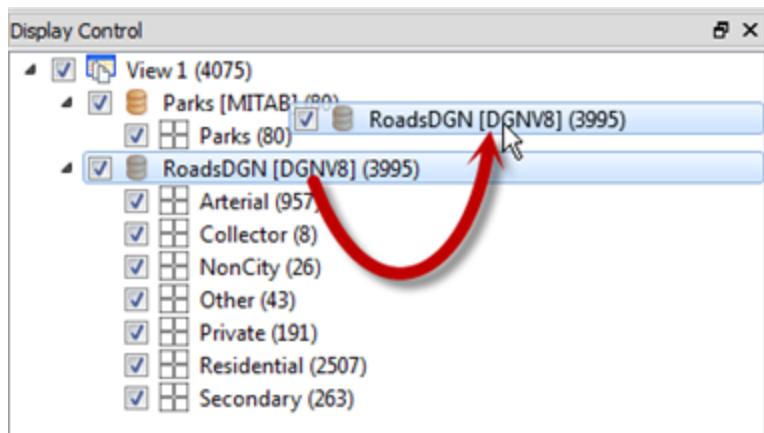


The FME Data Inspector has a number of controls to assist in showing the data in an orderly manner.

Display Control

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

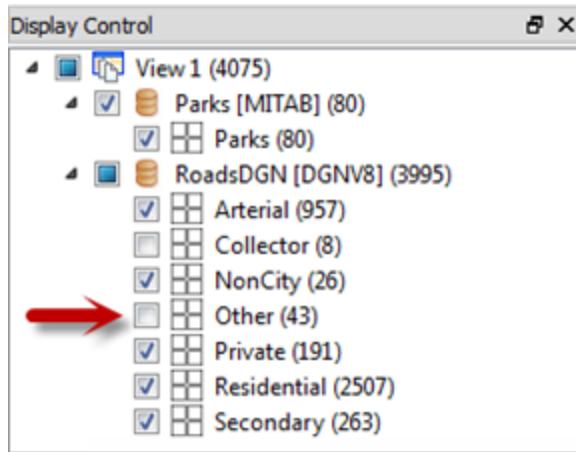
Each dataset and feature type can be dragged above any other to promote its display order in the View window.



*Feature Types can only be ordered within their container dataset.
For example, "Private" cannot be promoted above "Parks" in the display, without the entire Roads dataset being promoted above Parks.*

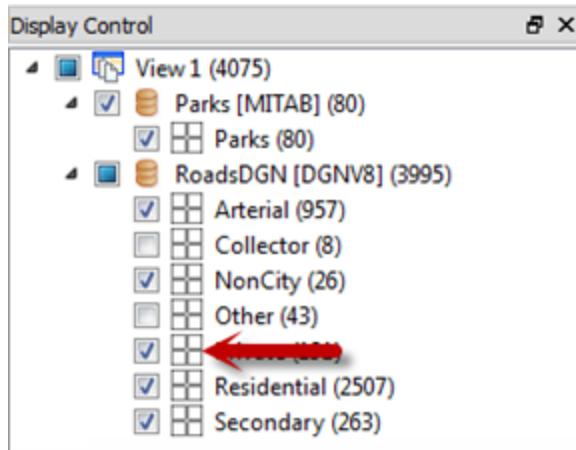
Display Status

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

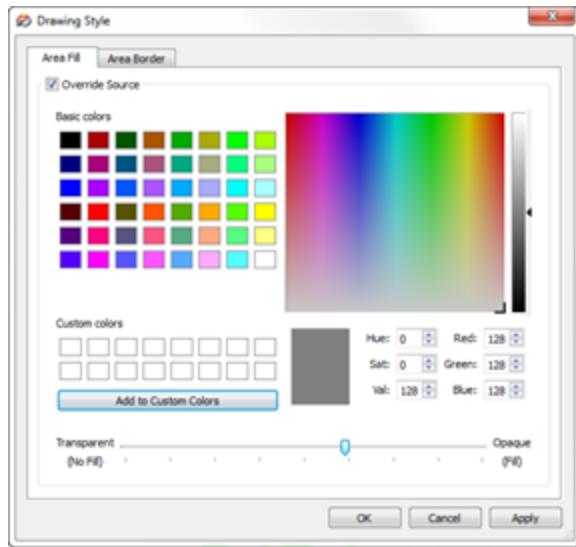


Style

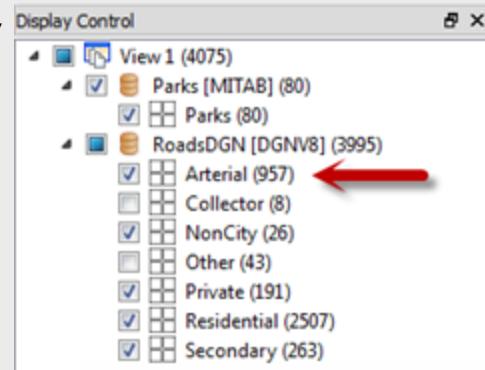
Each feature type can be assigned a different color or style that applies to its geometries. To access this functionality click on the style icon for that feature type in the Display Control window:



The Drawing Style dialog allows you to set the color of a feature and its degree of transparency. The exact settings available depend upon the type of geometry being set; for example a polygon feature will have settings for both fill and border:



Each Feature Type in the Display Control window is tagged with the number of features it contains, in parentheses after the feature type name. Thus we can see that there are 957 arterial road links in the city of Vancouver.



Miscellaneous Viewer Functionality

The FME Data Inspector has a number of miscellaneous functions that help users navigate through data, investigate data, and even translate data to a different format!

Shift and Ctrl Key Functions

Press the Shift key on the keyboard and it will activate the zoom-in tool in the Viewer.

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.

Save Tools

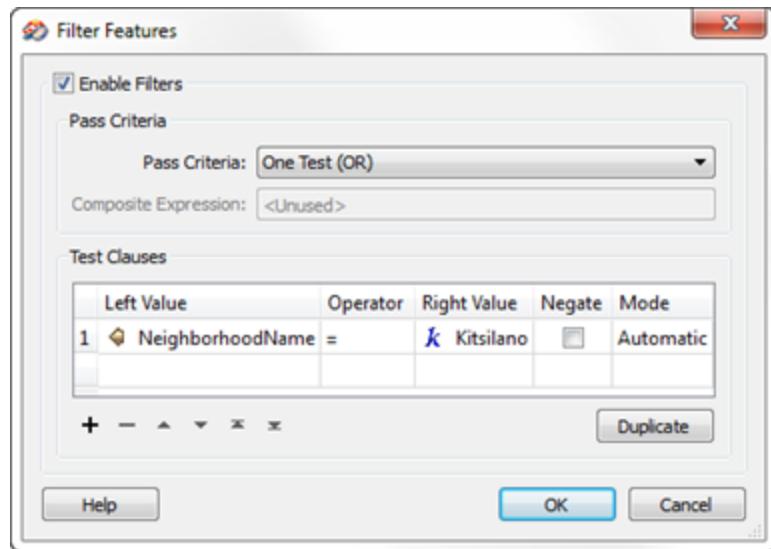
The FME Data Inspector save tool lets you save whatever data is currently being displayed in the view window. The data can be saved in any FME-supported format of your choice.

Simply select **File > Save Data As** to open the prompt for saving data.

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 that time. This functionality is activated by **Tools > Filter Features** on the menubar.

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.



The FME Data Inspector

Exercise 1c The FME Data Inspector	
Scenario	FME Workspace Author
Data	Parks (MapInfo TAB) Fire Halls (GML - Geographic Markup Language) Neighborhoods (Google Earth 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
Finished Workspace	None

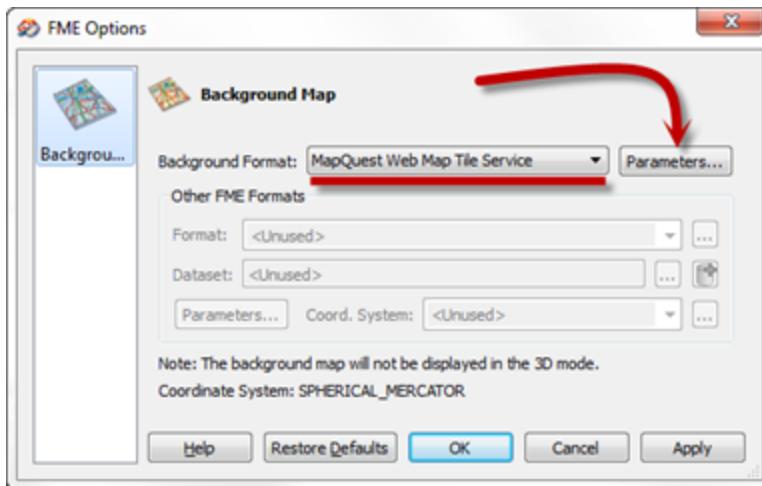
In this exercise, imagine that you are a GIS technician working for a city planning department. The mayor has asked you to help him use FME to inspect some data, to pick a neighborhood for him to buy a new house in!

1) Start FME Data Inspector

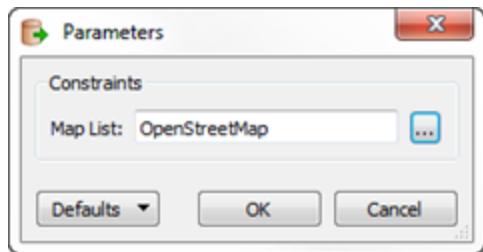
2) Set Background

When inspecting data it will help to have a background map to provide a sense of location. 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 of MapQuest Web Map Tile Service:



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



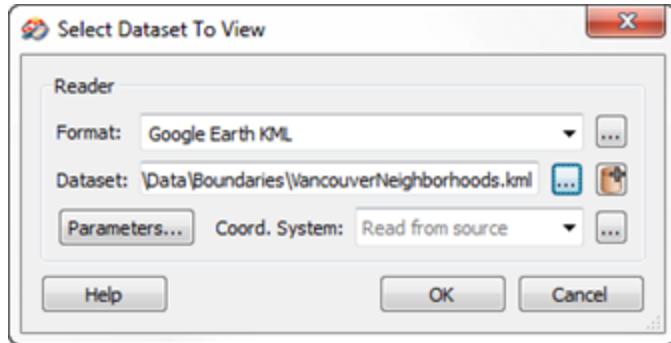
Click OK and OK again to close these dialogs.

3) Add Data

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

Format Google Earth KML

Dataset C:\FMEData2015\Data\Boundaries\VancouverNeighborhoods.kml



Click OK to close the dialog and add the data.

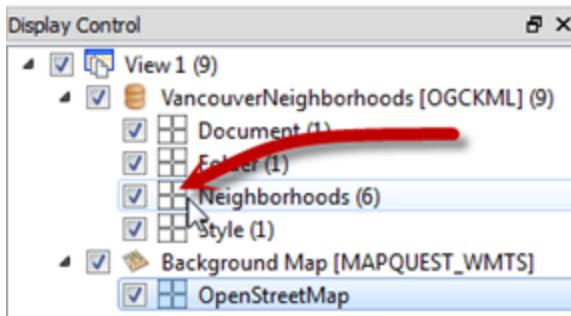
4) Set Data Symbology

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

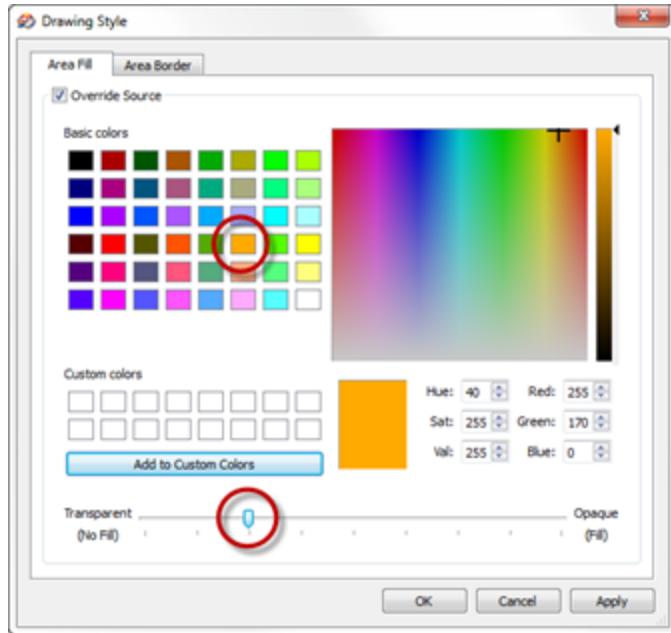


MapQuest: Tiles Courtesy of MapQuest

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).



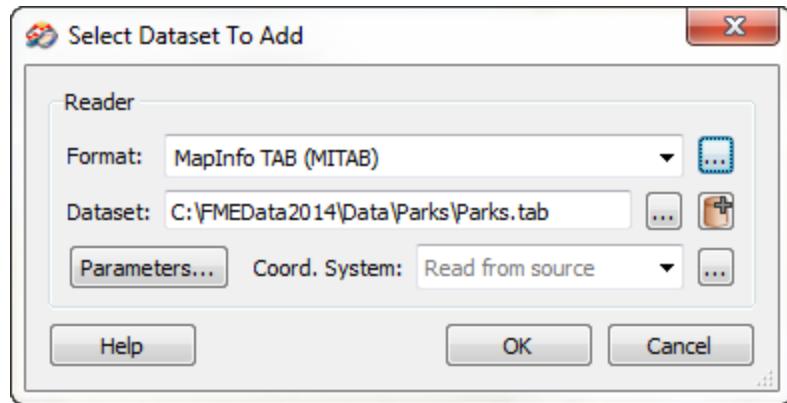
Click OK to accept the parameters and close the dialog. The background map will now be visible through the neighborhood data.

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:

Format	MapInfo TAB (MITAB)
Dataset	C:\FMEData2015\Data\Parks\Parks.tab

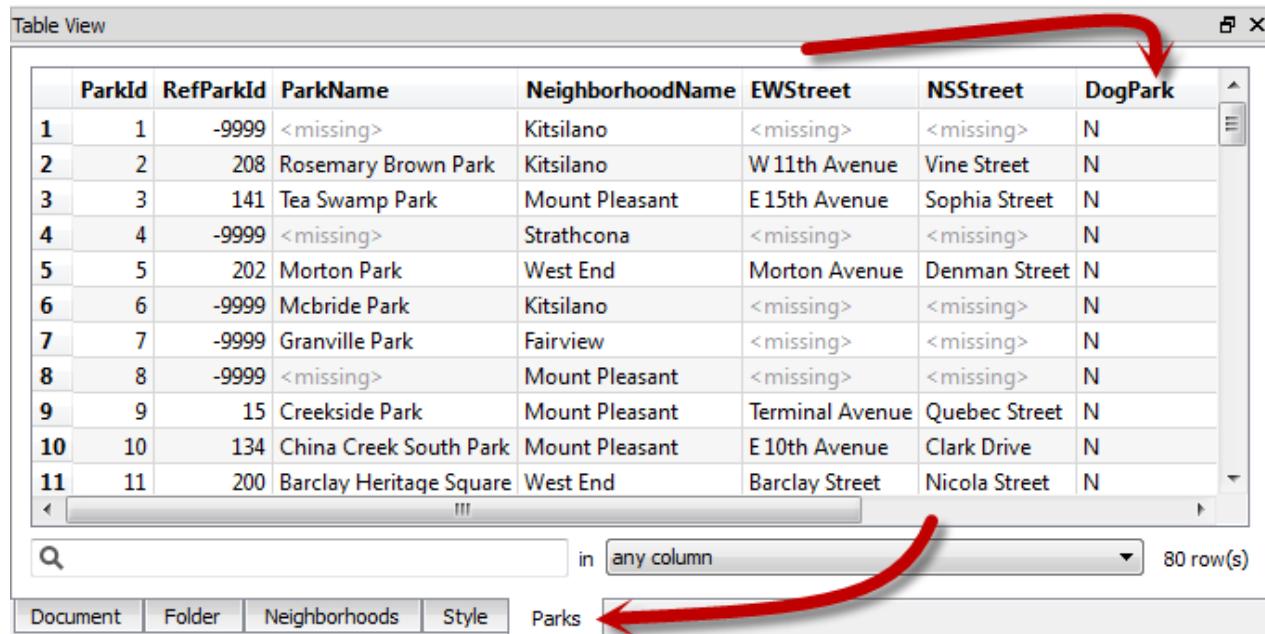


Click OK to close the dialog and add the data.

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.



	ParkId	RefParkId	ParkName	NeighborhoodName	EWStreet	NSStreet	DogPark
1	1	-9999	<missing>	Kitsilano	<missing>	<missing>	N
2	2	208	Rosemary Brown Park	Kitsilano	W 11th Avenue	Vine Street	N
3	3	141	Tea Swamp Park	Mount Pleasant	E 15th Avenue	Sophia Street	N
4	4	-9999	<missing>	Strathcona	<missing>	<missing>	N
5	5	202	Morton Park	West End	Morton Avenue	Denman Street	N
6	6	-9999	Mcbride Park	Kitsilano	<missing>	<missing>	N
7	7	-9999	Granville Park	Fairview	<missing>	<missing>	N
8	8	-9999	<missing>	Mount Pleasant	<missing>	<missing>	N
9	9	15	Creekside Park	Mount Pleasant	Terminal Avenue	Quebec Street	N
10	10	134	China Creek South Park	Mount Pleasant	E 10th Avenue	Clark Drive	N
11	11	200	Barclay Heritage Square	West End	Barclay Street	Nicola Street	N

80 row(s)

Document Folder Neighborhoods Style Parks

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.

Now, clicking on a feature will highlight it on the display window, but 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).

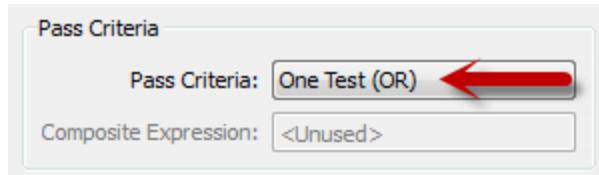
Test Clauses					
	Left Value	Operator	Right Value	Negate	Mode
1	DogPark	=	k Y	<input type="checkbox"/>	Automatic
2		=		<input type="checkbox"/>	Automatic
				<input type="button" value="Duplicate"/>	
<input type="button" value="+"/> <input type="button" value="-"/> <input type="button" value="▲"/> <input type="button" value="▼"/> <input type="button" value="✖"/> <input type="button" value="✖"/>					

Filtering in the Data Inspector is applied to all visible data, 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)

Test Clauses					
	Left Value	Operator	Right Value	Negate	Mode
1	DogPark	=	k Y	<input type="checkbox"/>	Automatic
2	NeighborhoodID	>	k 0	<input type="checkbox"/>	Automatic
3		=		<input type="checkbox"/>	Automatic
				<input type="button" value="Duplicate"/>	
<input type="button" value="+"/> <input type="button" value="-"/> <input type="button" value="▲"/> <input type="button" value="▼"/> <input type="button" value="✖"/> <input type="button" value="✖"/>					

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



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.



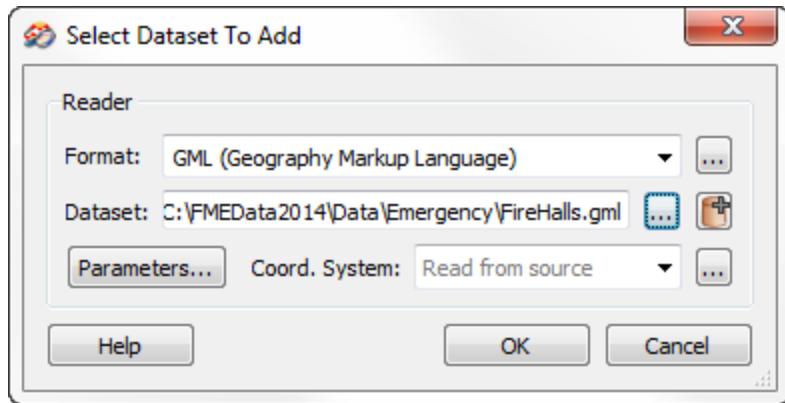
MapQuest: Tiles Courtesy of MapQuest

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:

Format	GML (Geography Markup Language)
Dataset	C:\FMEData2015\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).

Test Clauses					
	Left Value	Operator	Right Value	Negate	Mode
1	DogPark	=	k Y	<input type="checkbox"/>	Automatic
2	Neighborho...	>	k 0	<input type="checkbox"/>	Automatic
3	Rescue	=	k Y	<input type="checkbox"/>	Automatic

+ - ▲ ▼ × × Duplicate

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! You have now:

- Set a background map for the FME Data Inspector
- Opened a dataset for inspection and added more data to the same view
- Set symbology including color and transparency
- Queried and sorted data in the Table View window
- Filtered data using a filtering tool

Translation Previews



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

Redirect to Inspection Application

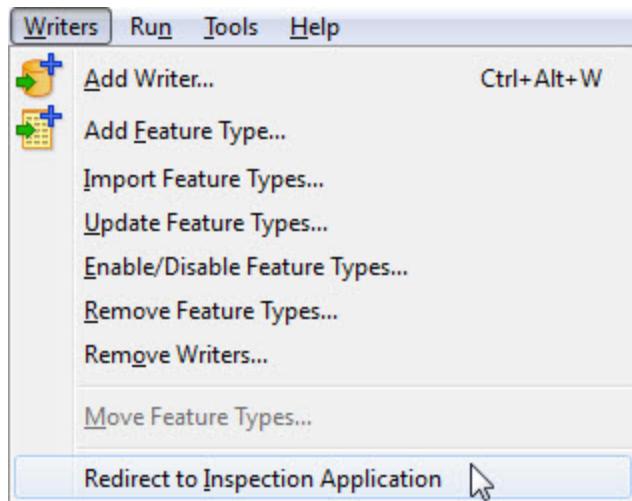
In some cases it's desirable to inspect output data, but undesirable to actually have to write the data to do so. In other words, you want to preview what the output of a translation will be.

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 'Redirect to Inspection Application' setting.

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.

The simplest way to turn on this ability is to select **Writers > Redirect to Inspection Application** on the Workbench menu bar.



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

Here a user is about to activate the Redirect to Inspection Application setting. Absence of a checkmark shows it is not already set.



An embarrassing problem occurs when a user forgets to deactivate the setting and does not understand why no output datasets are being written. To help combat this issue, the FME Log window includes a distinctive warning message when data is being redirected.

Notice how the redirection message in the FME Log window reports that zero features have been written to the output datasets.

```
=====
Features Written Summary
=====
Total Features Written          0
=====
-- *** All writer output has been redirected to the Inspector application ***
--
```

Module Review



This module was designed to introduce you to FME and to investigate 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; 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.

Exercise

Now let's prove what you have learned by carrying out an exercise on basic data translation and data inspection.

Data Translation Basics

Exercise 1d Data Translation Basics	
Scenario	FME Workspace Author
Data	Community Mapping/Food Vendors (Esri Geodatabase)
Overall Goal	Create a GPS-compatible dataset of Food Vendors
Demonstrates	Basic Data Translation and Data Inspection
Start Workspace	None
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise1d-Complete.fmw</i>

In this exercise, imagine that you are a GIS technician working for a city planning department. 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. You have been asked to produce the data to be used in this scheme.

1) Start FME Workbench

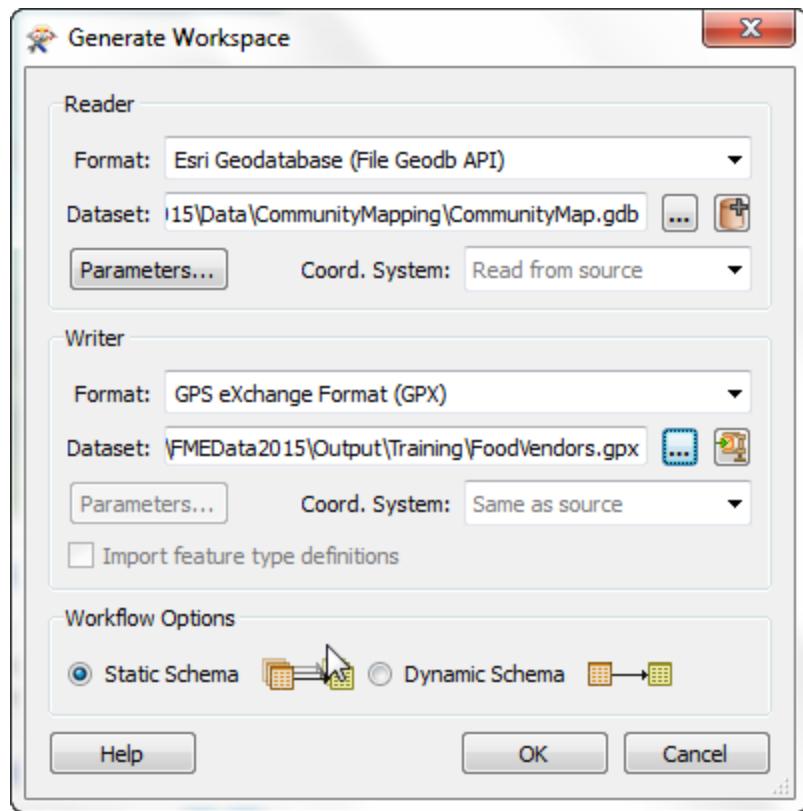
Start FME Workbench. In the Getting Started section of the Start window, choose the option to Generate Workspace. When prompted generate a translation with the following parameters:

Reader Format Esri Geodatabase (File Geodb API)

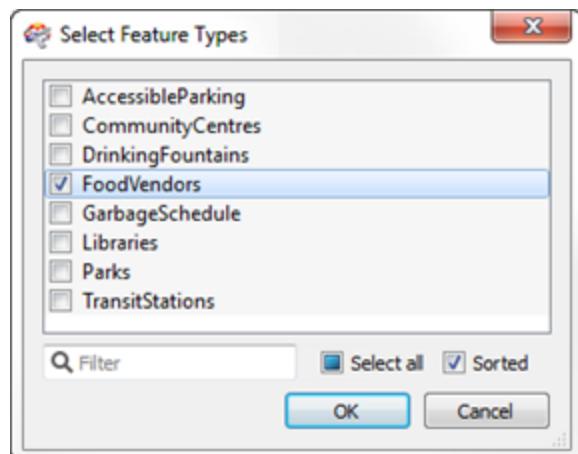
Reader Dataset *C:\FMEData2015\Data\CommunityMapping\CommunityMap.gdb*

Writer Format GPS eXchange Format (GPX)

Writer Dataset *C:\FMEData2015\Output\Training\FoodVendors.gpx*



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

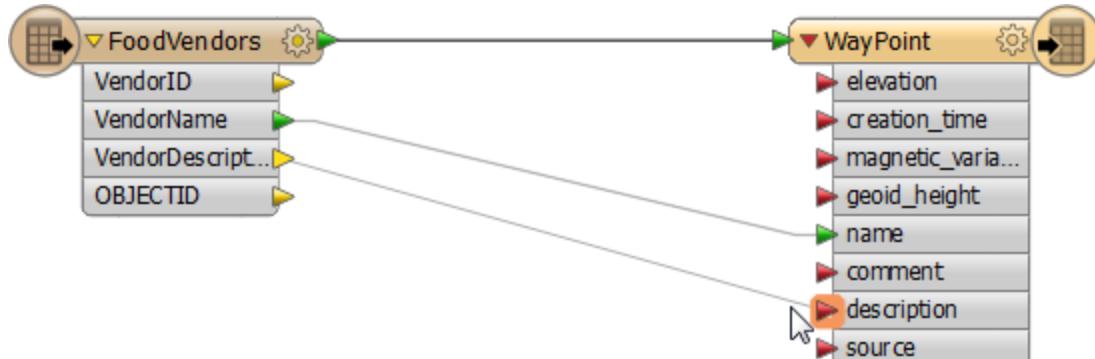
When first created, the Reader and Writer are not connected in this workspace. Connect them by dragging a connection from the output port of the Reader to the input port of the Writer object labelled WayPoint:



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 desc:



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

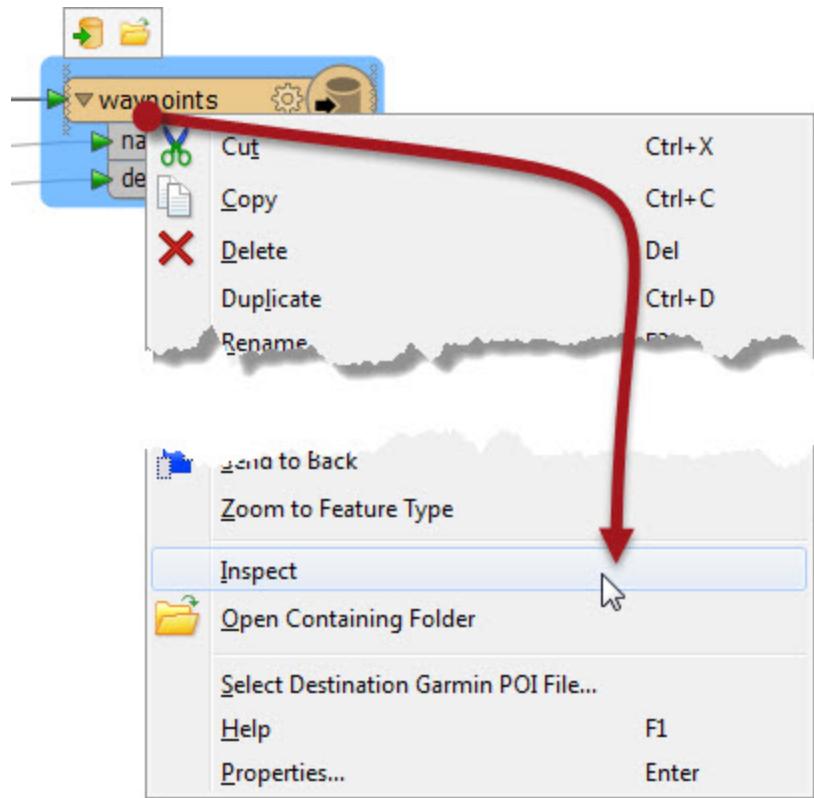
3) Run Workspace

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
|=====
|waypoints                                      91
|=====
|Total Features Written                         91
|=====
|Translation was SUCCESSFUL with 0 warning(s) (91 feature(s) output)
|FME Session Duration: 0.8 seconds. (CPU: 0.3s user, 0.2s system)
```

4) Inspect Data

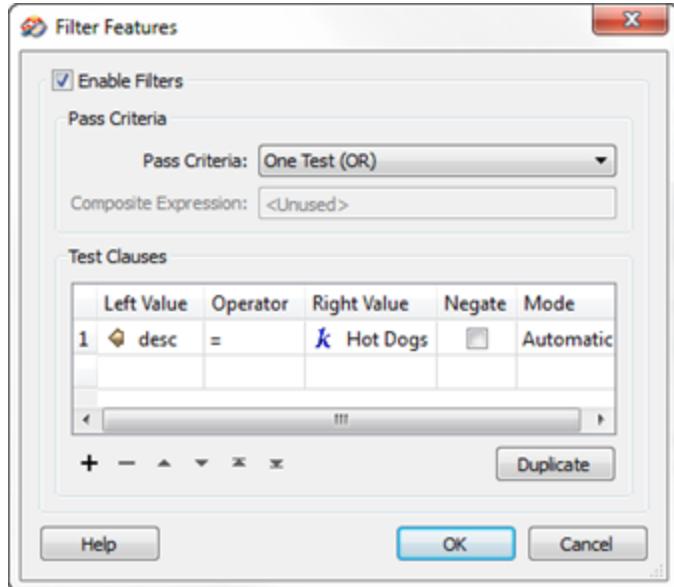
Right-click on the Writer object in the Workbench canvas. Choose the option to Inspect.



This will open the dataset in the FME Data Inspector, where you can verify it is all 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! You have now completed the exercise for this chapter.

Q&A Answers



Miss Vector says...

Here are the test answers. Anything less than 3 out of 3 and you'll find yourself reviewing the section again during lunch!

FME Desktop 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*

✓

Which of the following applications are parts of FME Desktop?

- 1) FME Workbench
- 2) FME Server
- 3) FME Universal Translator
- 4) FME Data Inspector

✓
✓
✓

Which of the following are windows in the Workbench interface?

- 1) Navigator
- 2) Transformer Gallery
- 3) Log Window
- 4) Display Control Window

✓
✓
✓



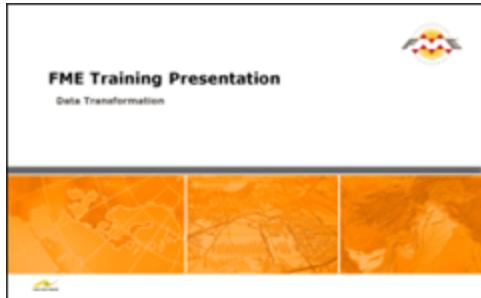
*Why might it be useful to set the data format before browsing for the source data?
Try browsing for a dataset before setting the format type and see if you can detect the difference.*

Answer -

When you set the format and then browse for a dataset FME will show only datasets of the chosen format, making the selection task easier.

When you browse for a dataset before setting the format, FME will show all files.

Chapter 2 - 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.

Sometimes FME automatically manipulates the data, as part of its **semantic** capabilities, renaming attributes, adjusting geometry, and splitting up data into several layers in order to ensure that the output from a Format Translation meets the specification of the destination format.

Data Transformation Types

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

Structural Transformation

This type of transformation is perhaps better called 'reorganization'. It refers to the ability of FME to channel data from source to destination in an almost infinite number of arrangements. This includes the ability to merge data, 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

This type of 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..

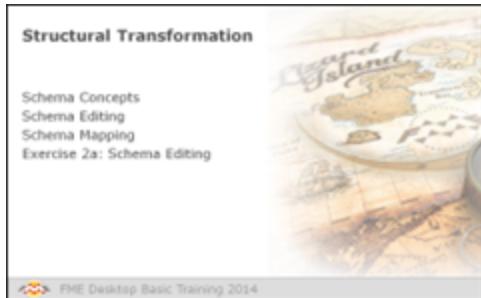
"Before you start on transformation, here's a challenge for you!

According to user feedback, the most common format translation defined with FME is from Esri Shape to which format?

- *Geodatabase*
- *AutoCAD DXF/DWG*
- *Esri Shape*
- *Google Earth KML*

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. The term Data Model may be more familiar, but at Safe Software we stick to the term 'schema'.

Each dataset has its own unique structure (schema) that includes feature types (layers), permitted geometries, user-defined attributes, and other rules that define or restrict its content.

How Does FME Represent Schemas?

When a new workspace is created, FME scans all of the source datasets. From this it creates a visual representation of the data's schema on the left side of the canvas. On the right side, it creates a visual representation of how this schema will be duplicated in the chosen output format.

Here are source and destination schemas as they are represented in Workbench.



Each object on the canvas is a separate Feature Type within a dataset.

The workspace reads from left to right.

At this point, the Reader schema represents '*what we have*'; that is, FME's view of the source datasets. The Writer schema represents '*what we want*'; the data required by the user.

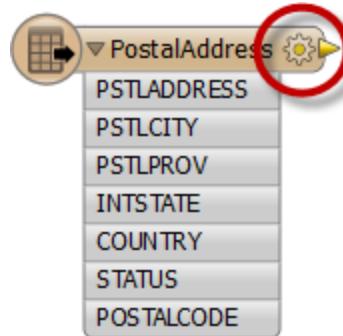
By default, the Writer schema is a mirror image of the source; differences only occur when demanded by limitations of the selected destination format. This allows Quick Translation to occur with no further editing of the translation by the user.

Viewing the Schema in FME Workbench

A schema goes beyond what can be seen on the workspace canvas; there are other components in various dialogs that also represent the structure of a dataset.

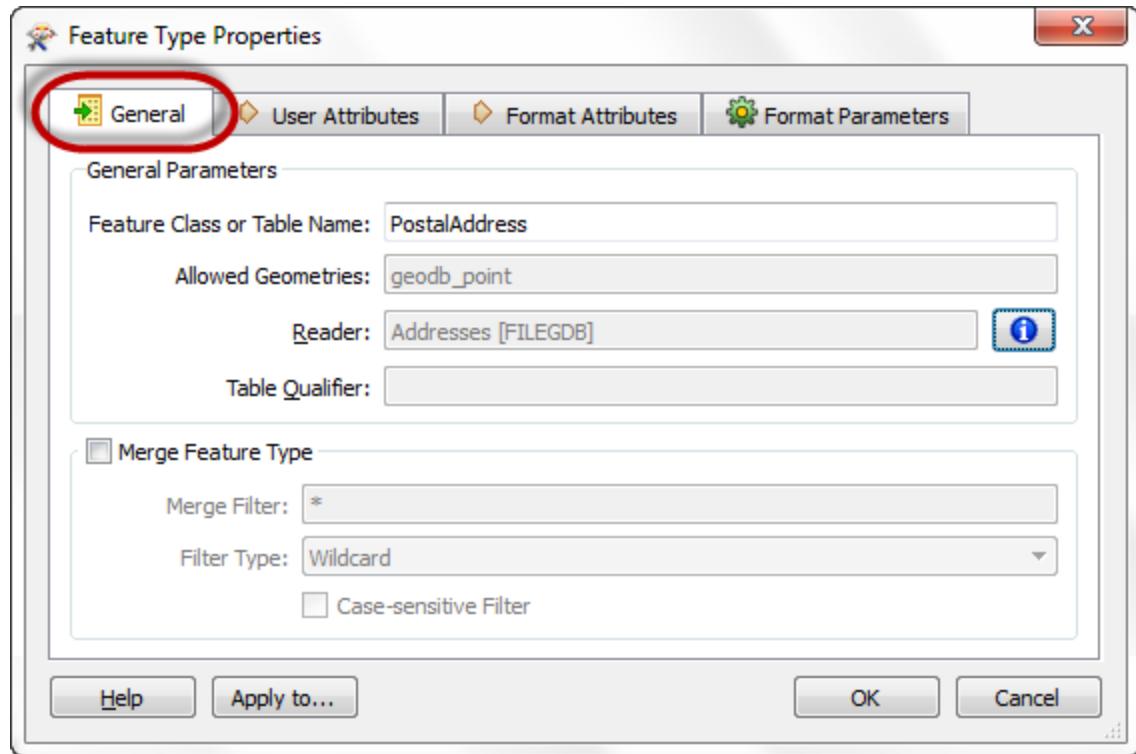
Some parts of the schema relate specifically to a single feature type only. Attributes are one such component. These components are shown in the Properties dialog of a feature type.

The Properties dialog is opened by clicking the Properties Browse button at the right side of the feature type.



The Feature Type Properties dialog has a number of tabs that show information.

Under the General tab there is the name of the feature type, in this case *PostalAddress*.

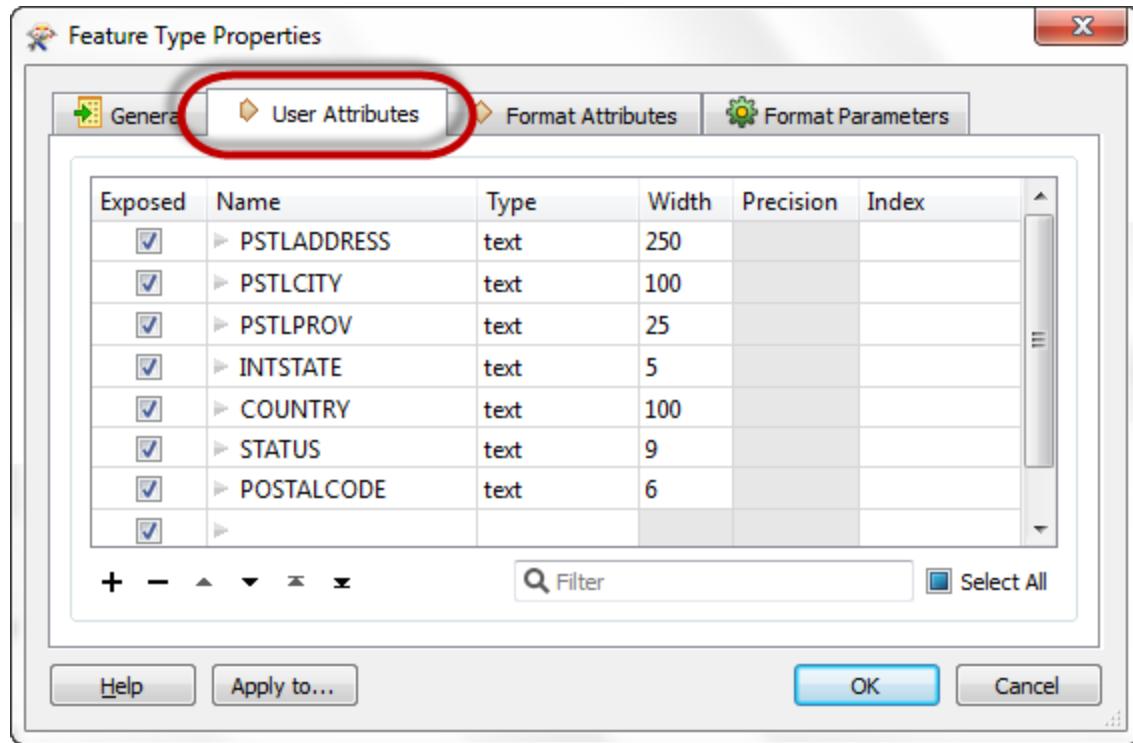


Allowed geometry types and the name of the parent dataset for the feature types are shown as well.



FME 2015 dialogs have been updated with format-specific terminology. That is, the term "Feature Type" is automatically replaced with the appropriate term for the current format. In the previous screenshot a label states "Feature Class or Table Name" because the format is a Geodatabase. If the format were AutoCAD (for example) then the label would state "Layer Name."

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



This example shows a Reader feature type. You may find these greyed out by default, since source attributes cannot be edited because they represent the physical schema of the data. If they were changed, the schema would no longer match the Reader dataset.



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.

Schema Editing

As noted, initially the Writer schema in a workspace is a mirror image of the source. However, in many cases the user requires the output to have a different data structure.

Schema Editing is the process of altering the destination schema to customize the structure of the output data. One good example is renaming an attribute field in the output. After editing, the source schema still represents '*what we have*', but the destination schema now truly does represent '*what we want*'.

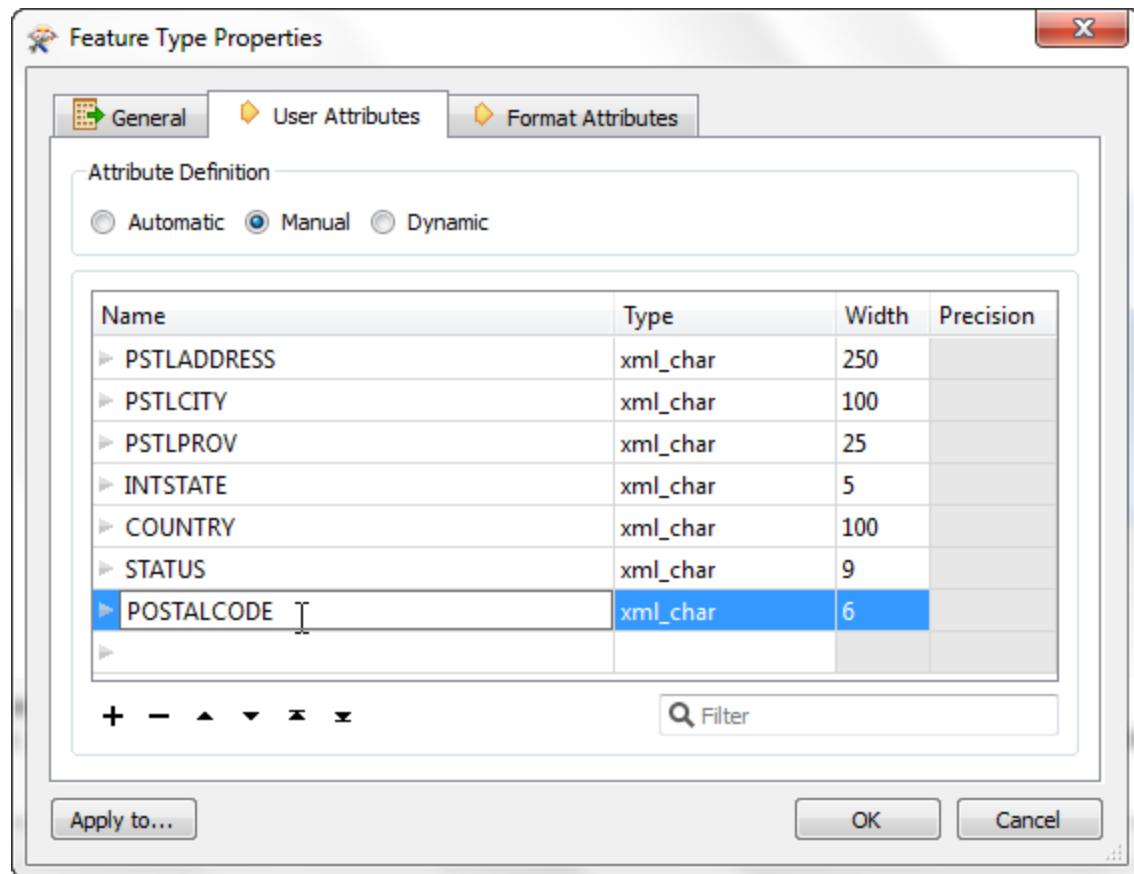
Editable Components

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.



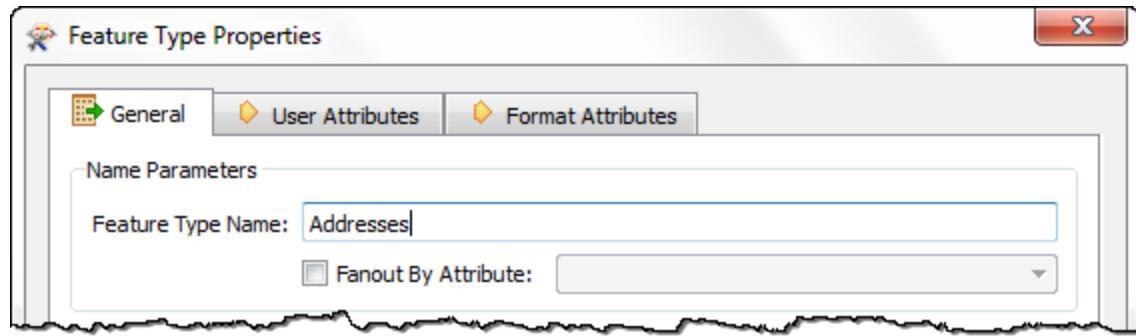
Attribute Type Changes

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

Feature Type Renaming

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



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.

Schema Mapping

Schema Mapping forms the basis for data restructuring.

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.

Feature Mapping

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

Attribute Mapping

Attribute Mapping is the process of connecting source attributes to destination attributes.



Ms. Analyst says...

"You can think of Schema Editing and Mapping as 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.

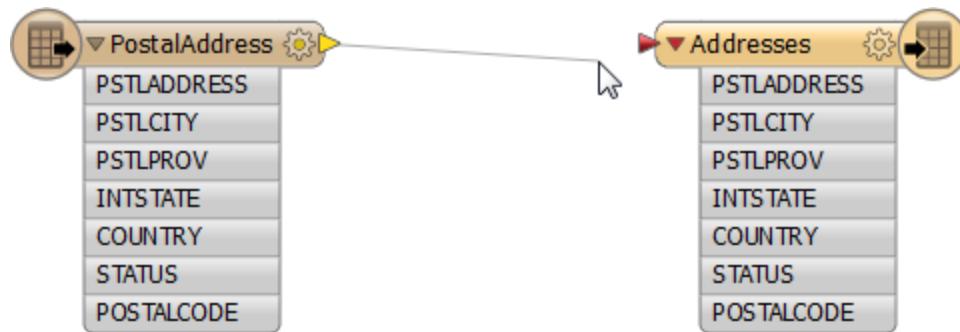
The same holds true for spatial data restructuring: it's the act of reorganizing data to make it more usable."

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.

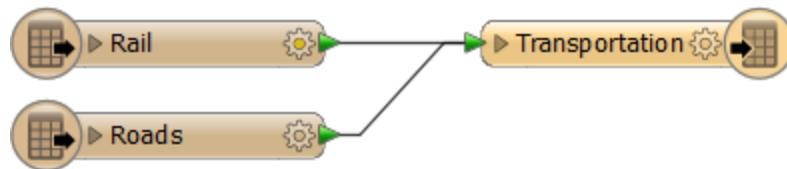
Feature Mapping in FME Workbench

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.



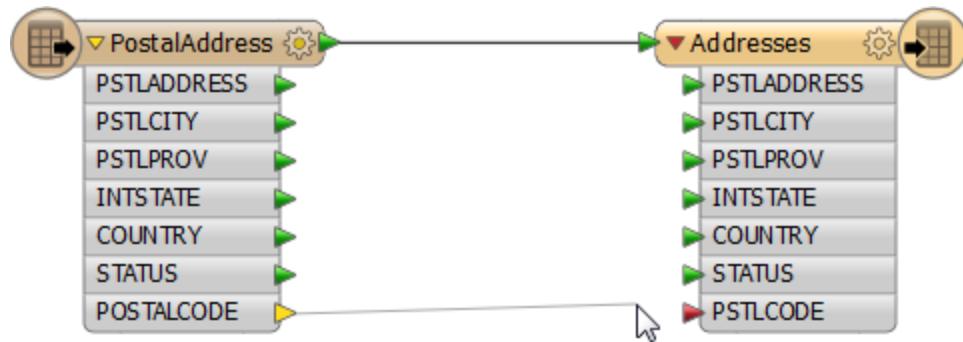
Here, a user wishes to create a single layer called Transportation and so is merging two input feature types (Roads and Rail) into one output feature type (Transportation).



Attribute Mapping in FME Workbench

Attribute Mapping is performed by clicking the output port of a source attribute, dragging the arrowhead to the input port of a destination 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 source attribute that's unconnected to a destination. Red ports indicate a destination attribute that's unconnected to a source.

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 source and destination are connected automatically, even though a connecting line might not be visible; the port color is the key.

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

It is permitted to map '*what we have*' to '*what we want*' in any way that is desired.

Schema Editing

Exercise 2a Schema Editing	
Scenario	FME user; City of Interopolis, Planning Department
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
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2a-Complete.fmw

In this example, imagine that you are a GIS technician working for a city planning department. 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 are to 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 the information that follows.

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2015\Data\Parks\Parks.tab
Writer Format	MapInfo TAB (MITAB)(yes – here we write back to the same format!)
Writer Dataset	C:\FMEData2015\Output\Training

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.

Open the Feature Type Properties dialog for the **writer** feature type by clicking the Properties button. Click the User Attributes tab to open a list of the destination attributes. It will look like this:

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

In turn, carry out the following actions:

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

After these edits the attribute list should look like this:

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

3) Rename Feature Type

Now click back on the General tab.

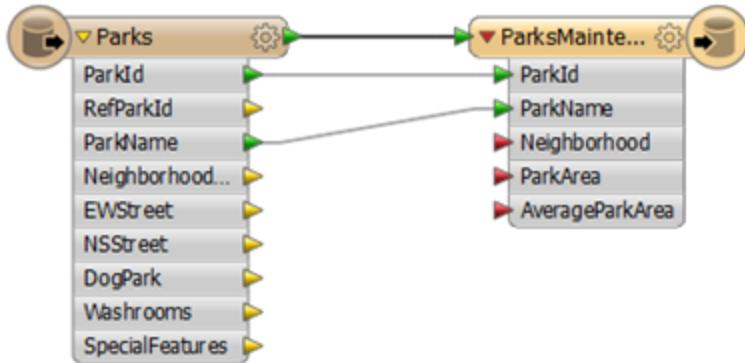
Click in the field labelled Table Name (remember this label is format-specific and in MapInfo we deal with "tables") and change the name from *Parks* to *ParksMaintenanceData*.

Click **OK** to accept these changes.

Now when the workspace is run the output will be named *ParksMaintenanceData.tab*

4) Un-Expose Source Attributes

The workspace will now look like this:



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.

Open the Feature Type Properties dialog for the **Reader** feature type by clicking the Properties button. 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		
<input checked="" type="checkbox"/>	►				

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

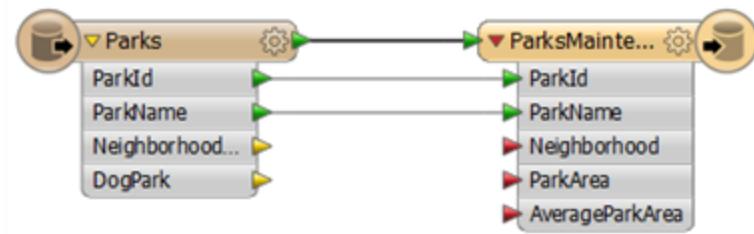
- RefParkID
- EWStreet
- NSStreet
- Washrooms
- SpecialFeatures

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

Click OK to close the dialog.

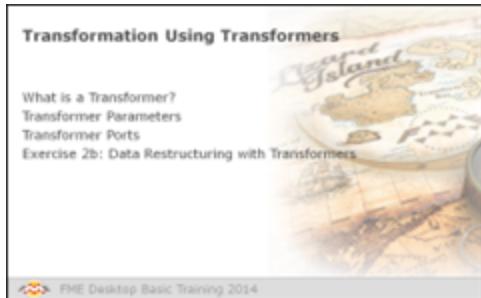
5) Save the Workspace

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



There is nothing yet to map to ParkArea or AverageParkArea because values for them do not yet exist, and Neighborhood is still unconnected, but you can still run the workspace just to see what the output looks like.

Transformation Using Transformers



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

What is a Transformer?

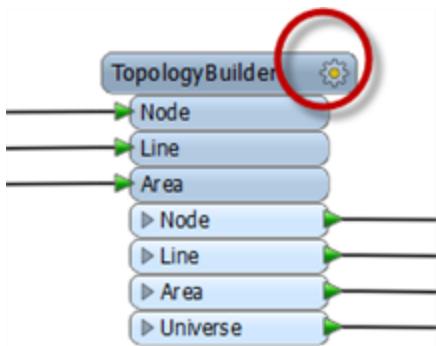
As the name suggests, a transformer is an FME Workbench object that carries out transformation of features. A number of different transformers exist to carry out different operations.

Transformers usually appear in the canvas window as rectangular, light-blue objects, although there are some transformers that vary from the norm with special shapes and colors.

Transformers are connected somewhere between the source and destination feature types, so that data flows from the reader feature types, through a transformation process, and on to the writer.

Transformer Parameters

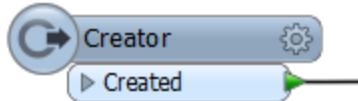
Each transformer may have a number of parameters – also known as settings. Access the settings by clicking the Properties button at the top right of each transformer.



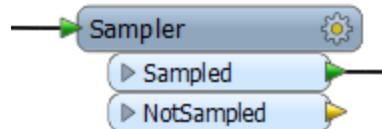
Color-Coded Properties Buttons

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

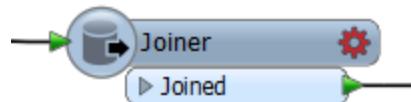
A **blue** Properties button indicates that the default transformer settings have been checked and amended as required, and that the transformer is ready to use.



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



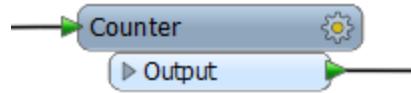
A **red** Properties button indicates that there is at least one setting for which FME cannot supply a default value. The setting must be provided with a value before the transformer can be used.



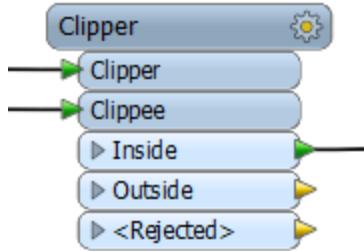
Transformer Ports

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

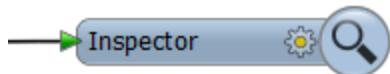
This *Counter* transformer has a single input and output port.



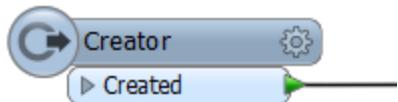
This *Clipper* has multiple input and output ports. Notice too that not all of them are – or need to be – connected.



This *Inspector* has just a single input port...



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



Data Restructuring With Transformers

Exercise 2b Data Restructuring with Transformers	
Scenario	FME Workspace Author
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
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2b-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2b-Complete.fmw

In this exercise, imagine that you are a GIS technician working for a city planning department. 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 are to use FME to provide a dataset of this information.

We'll now continue the previous example by filtering 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.

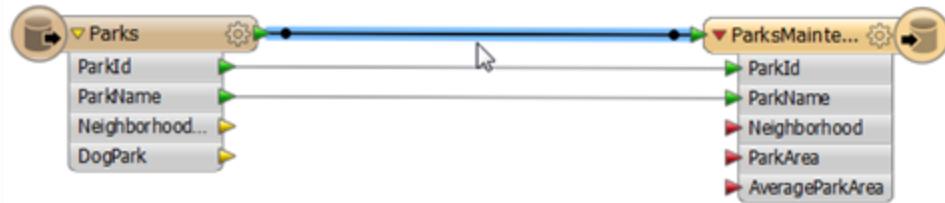
1) Start FME Workbench

Start FME Workbench and open the workspace from Exercise 2a. Alternatively you can open C:\FMEData2015\Workspaces\DesktopBasic\Example2b-Begin.fmw.

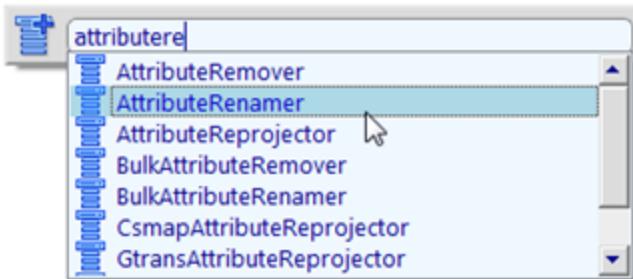
2) Add Transformer

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 called the AttributeRenamer.

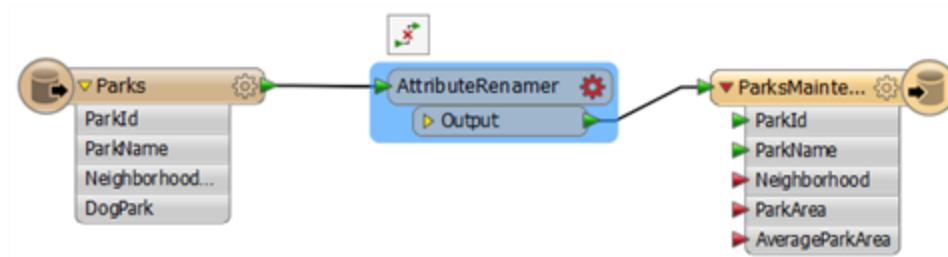
Therefore click on the feature connection from Reader to Writer feature type:



Start to type the phrase "AttributeRenamer." 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 AttributeRenamer, select it from the dialog:



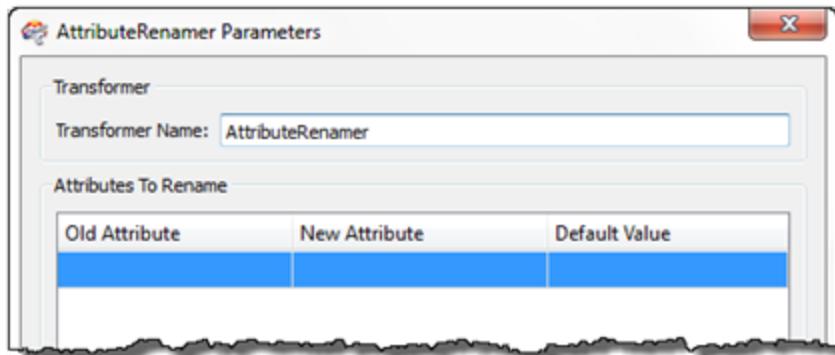
This will place an AttributeRenamer transformer like so:



Notice that the parameters button is colored red because there are parameters that need to be set for this transformer.

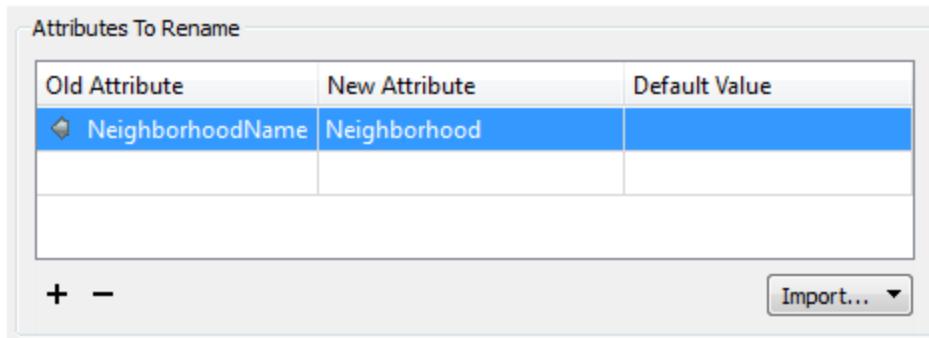
3) Set Parameters

Click the parameters button on the AttributeRenamer transformer to open the parameters dialog. It will look like this:



In the Old Attribute field, click in the field, click on the drop down arrow, and from the list that is revealed choose NeighborhoodName as the attribute to rename.

In the New Attribute field, repeat the process but instead choose Neighborhood. The dialog will now look like this:

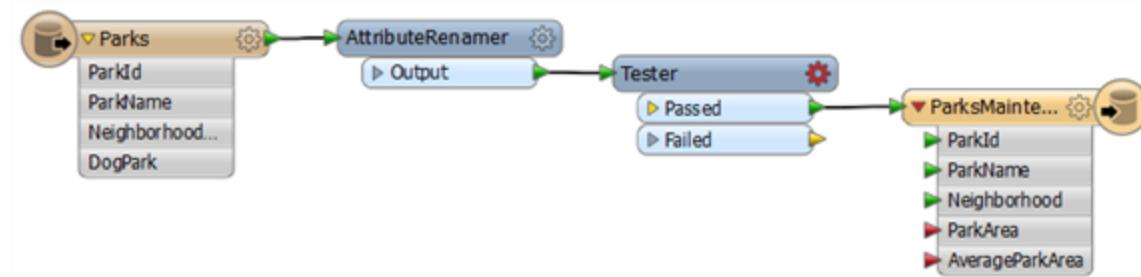


Click OK to close the dialog. Now in the Workbench canvas window you will see the Neighborhood attribute is flagged with a green arrow, to confirm that a value is being provided to that attribute.

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 AttributeRenamer output port to the ParksMaintenanceData feature type on the Writer.

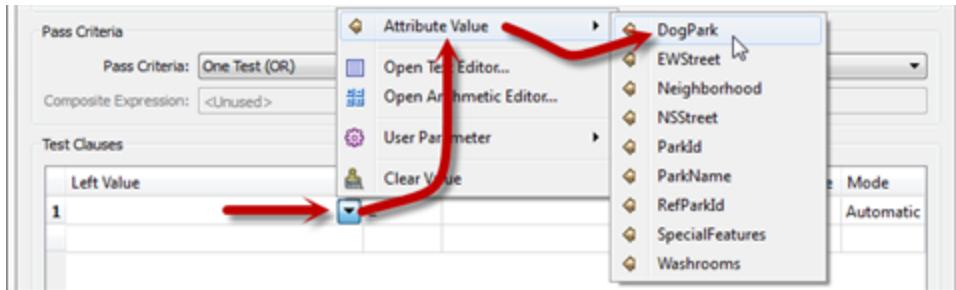
Start typing the word Tester. When you spot the Tester transformer click on it to add one to the workspace. The workspace will now look like this:



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

5) Set Parameters

Click the parameters button on the Tester transformer to open the parameters dialog. Double-click in the Left Value field and from there click the down arrow and choose **Attribute Value > DogPark**.



For the operator choose the equals sign (=) and for the Right Value click into the field and type the value N.

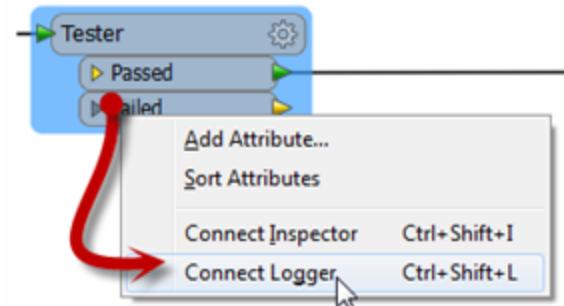
Left Value	Operator	Right Value	Negate	Mode
1 ♦ DogPark	=	<i>N</i>	<input type="checkbox"/>	Automatic

Click OK to accept the values and close the dialog.

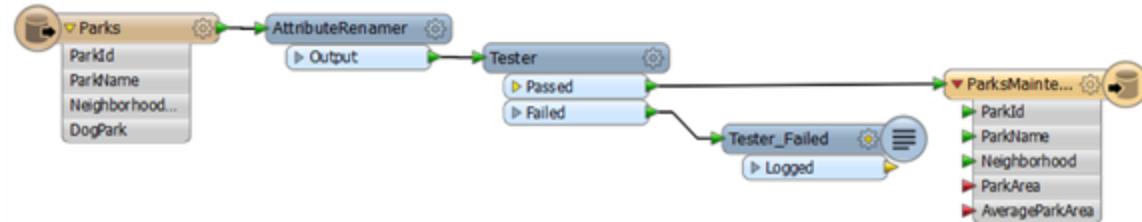
6) Add Transformer

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.



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! You have now:

- Carried out schema mapping with the AttributeRenamer transformer
- Filtered data using the Tester transformer
- Recorded data with the Logger transformer



Miss Vector says...

'Surprise! It's time for another quiz to check your progress.'

Turn to a fellow student and answer these questions between you.'

The ability to manipulate data during translation is called:

- 1) Translation
- 2) Transmogrification
- 3) Transfiguration
- 4) Transformation

The source and destination parts of a schema show:

- 1) What we have/What we want
- 2) What we did/What we should have done
- 3) What we're adding/What we're removing
- 4) Where my data was/Where it is now

Deciding how the source schema connects to the destination is called:

- 1) Schema Mapping
- 2) Schema Connecting
- 3) Schema Joins
- 4) Schema Concatenating

Which of these isn't a connection arrow color on an FME schema?

- 1) Red
- 2) Green
- 3) Blue
- 4) Yellow

Content Transformation

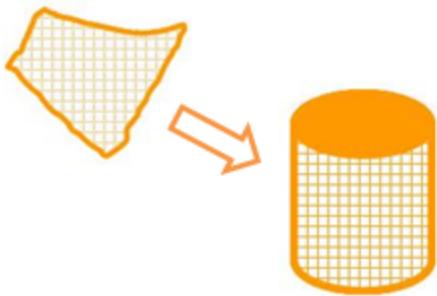


Content transformations are those that operate on the geometry or attribute content of a dataset.

What is a Feature?

A feature in FME is an individual item within the translation.

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.



A feature in FME is the fundamental (that is, smallest) unit of FME data. Features in FME have a flexible, generic representation; in other words they have a basic FME definition that is unrelated to the format from which they originated.

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



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"

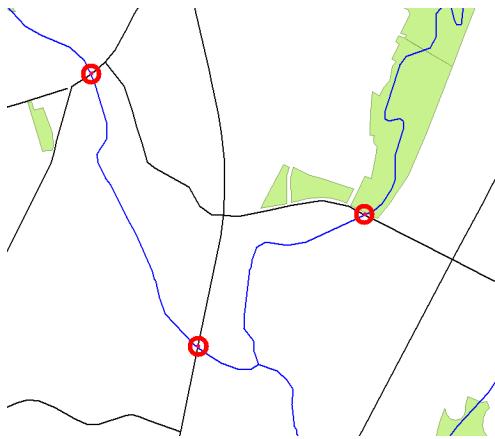


Geometric Transformation

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

Transformers used in Series



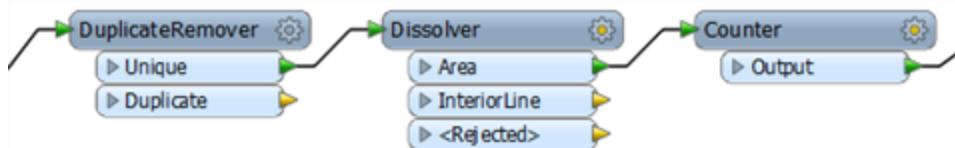
Much like a set of components in an electrical circuit, a series of Workbench Transformers can be connected together to have a cumulative effect on a set of data.

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 *DuplicateRemover* 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 2c Content Transformation with Transformers

Scenario	FME user; City of Interopolis, Planning Department
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
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2c-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2c-Complete.fmw

In this example, imagine that you are a GIS technician working for a city planning department. 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 are to use FME to provide a dataset of this information.

The previous two examples set up the project schema, now we'll continue by calculating the size and average size of each park, and ensuring that information is correctly mapped to the destination schema.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2b.

Alternatively you can open `C:\FMEData2015\Workspaces\DesktopBasic\Exercise2c-Begin.fmw`

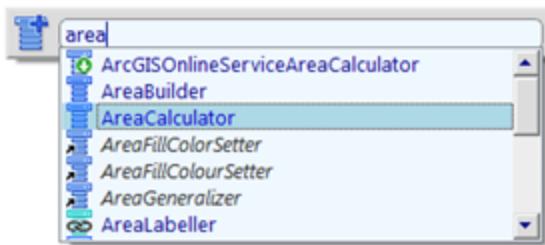
2) Add an AreaCalculator Transformer

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 Parks. Start typing the letters "areac". You will see the Quick Add list of matching transformers appear beneath.

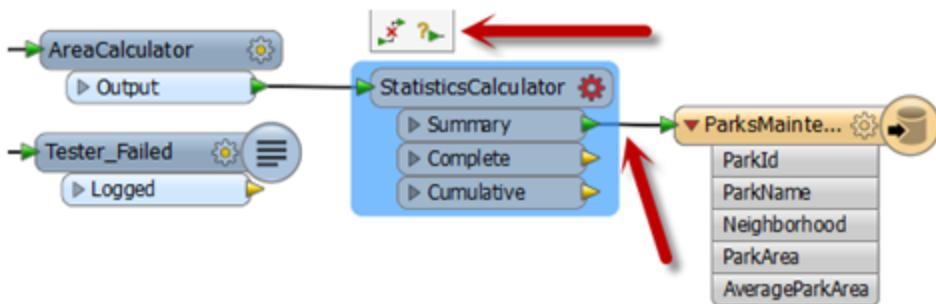
Select the transformer named *AreaCalculator*



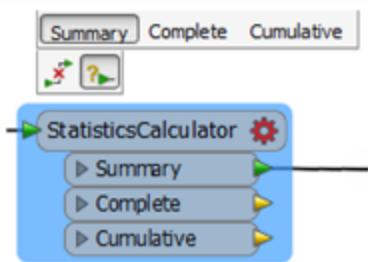
3) Add a StatisticsCalculator Transformer

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

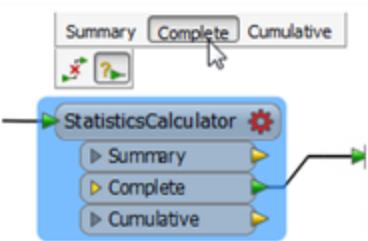
But do not click anything else yet! The transformer will now look like this:



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:

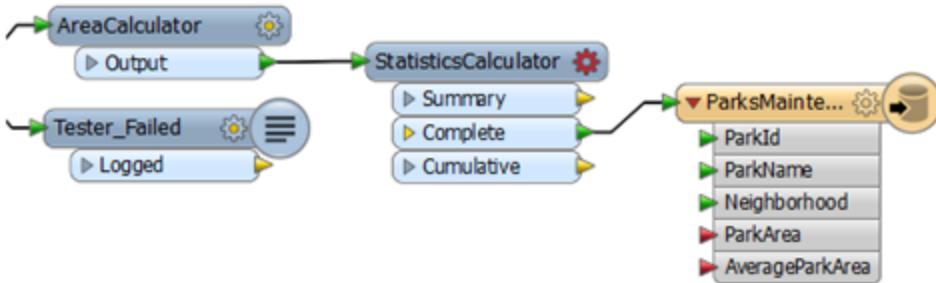


Click on the Summary port entry to disconnect that, and then on the Complete port entry to connect that:



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

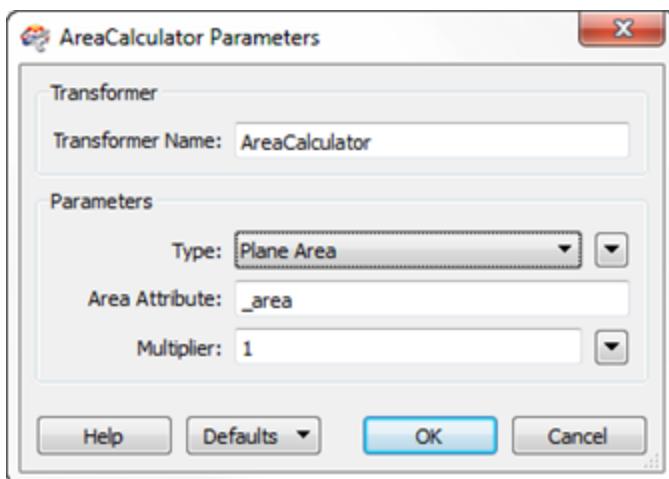
The latter part of the workspace now looks like this:



4) Check AreaCalculator Settings

A yellow icon indicates the *AreaCalculator* has parameters that need to be checked.

Open the *AreaCalculator* transformer Parameters dialog.



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.

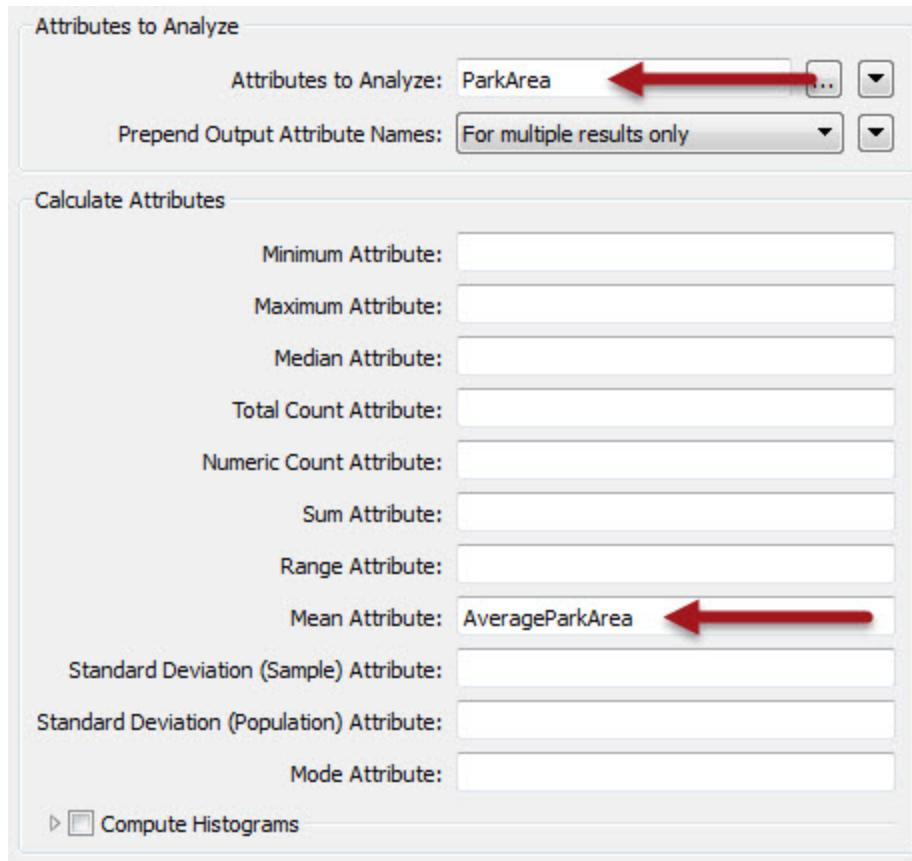
Notice that the attribute on the Writer feature type is now flagged as connected.

5) Check StatisticsCalculator Settings

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

Open the *StatisticsCalculator* transformer's Parameters dialog.

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



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)

Finally, click **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

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

any column 73 row(s)

ParksMaintenanceData

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

7) Save the Workspace

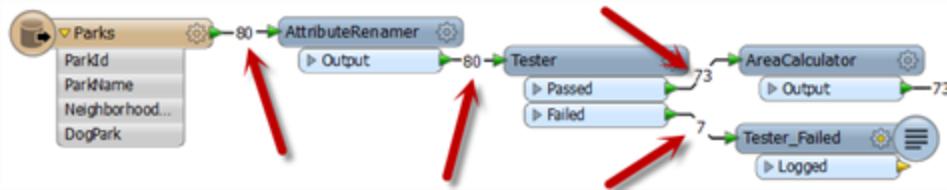
Save the workspace – it will be completed in further examples.



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.

Feature Count Display – Serial Transformers

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?

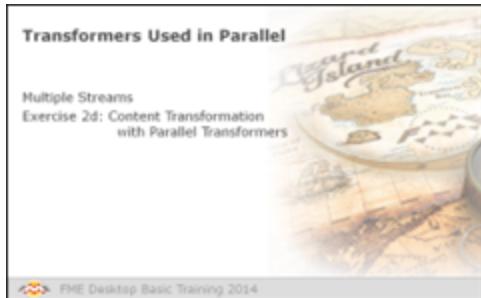


The final feature counts show that 80 features were read, 73 passed the Tester while 7 failed (for being dog parks) and went on to the Logger.

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 create multiple processing streams within a workflow or to bring several streams together into one.



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

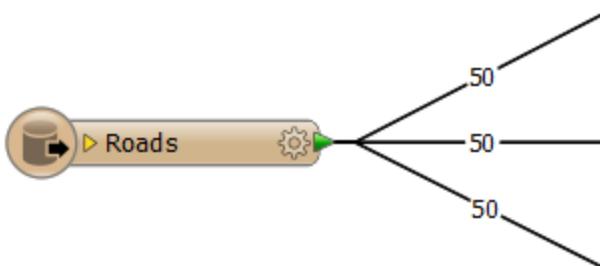
Creating Multiple Streams

Splitting data occurs with any transformer with multiple output ports (a Tester transformer is a good illustration of this) but can also be achieved by simply making multiple connections out of a single output port.

The difference is that when using multiple ports the data is being split amongst the different streams.

However, when using multiple connections from a single port, the data is being duplicated rather than being split. In effect it creates a set of data for each connection.

Here, 50 road features are being read but, because there are multiple connections from the feature type, multiple copies of the data are being created.

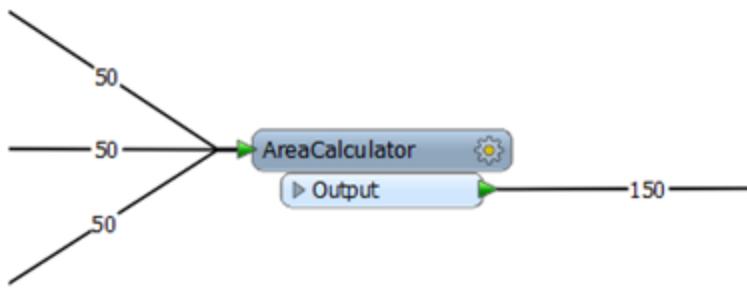


Multiple streams are useful when a user needs to process the same data, but in a number of different ways.

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.

Here, three streams of 50 features each, converge upon a single AreaCalculator:INPUT port.



Notice how no merging has taken place; the data simply accumulates into 150 output features.

To carry out actual merging of data requires a specific transformer such as the *FeatureMerger*.

Exercise 2d Content Transformation with Parallel Transformers	
Scenario	FME user; City of Interopolis, Planning Department
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
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2dBegin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2d-Complete.fmw C:\FMEData2015\Workspaces\DesktopBasic\Exercise2d-Complete-Advanced.fmw

In this example, imagine that you are a GIS technician working for a city planning department. 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 are to use FME to provide a dataset of this information.

Now 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 2c.

Alternatively you can open `C:\FMEData2015\Workspaces\DesktopBasic\Exercise2dBegin.fmw`

Exercise 2c measured park areas with the *AreaCalculator*. Now the FME user has been asked to add this information as labels to the output dataset.

This can be achieved using the *LabelPointReplacer* transformer.

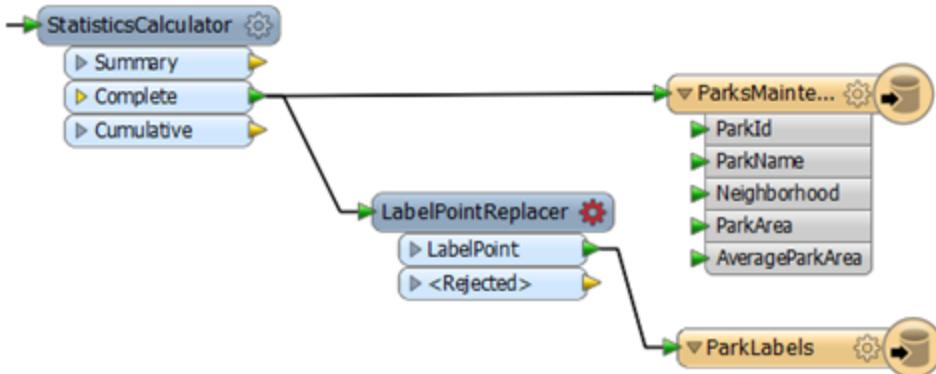
2) Place a LabelPointReplacer Transformer

Click onto a blank area of canvas. Type "LabelPointReplacer" 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.

3) Create new writer Feature Type

Right-click the Writer feature type and choose the option **Duplicate**. This creates a new feature type (layer) in the output dataset.

Open the Feature Type properties dialog. Rename the new type to *ParkLabels*. In the User Attributes tab delete all of the existing user attributes.



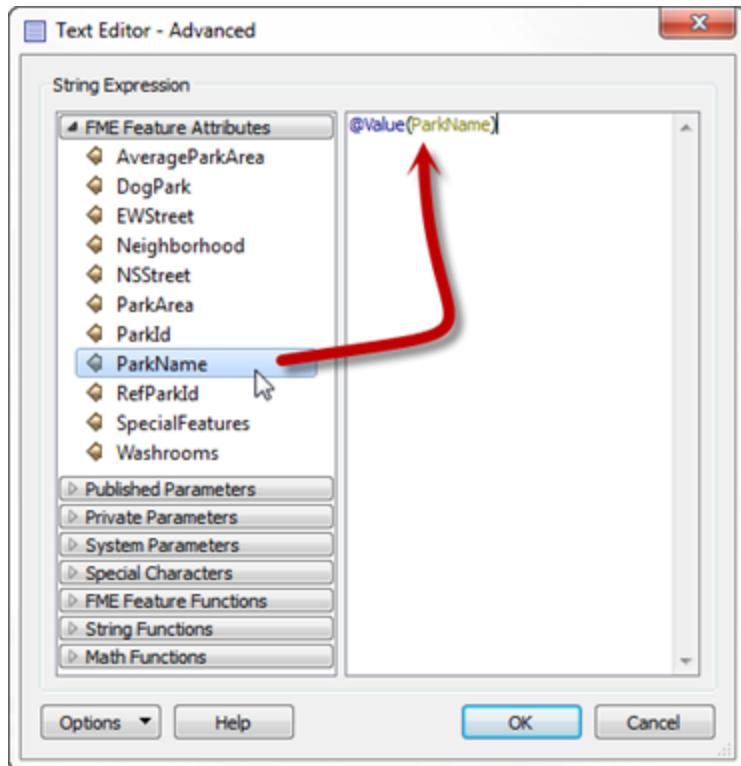
Make a new connection from the *LabelPointReplacer* to the new feature type.

4) Check Transformer Parameters

Open the parameters dialog for the *LabelPointReplacer* transformer.

Click the browse button to the right of the label field to open an advanced text editor. We want the label to include the park name and area, with a notation for the units too.

So, firstly double click ParkName in the list of attributes to the left:

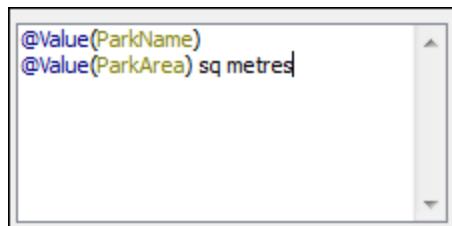


Now press the Enter/Return key to insert a new line.

Next double-click the ParkArea attribute to add that.

Finally type "sq metres" after the ParkArea attribute.

The dialog should now look like this:



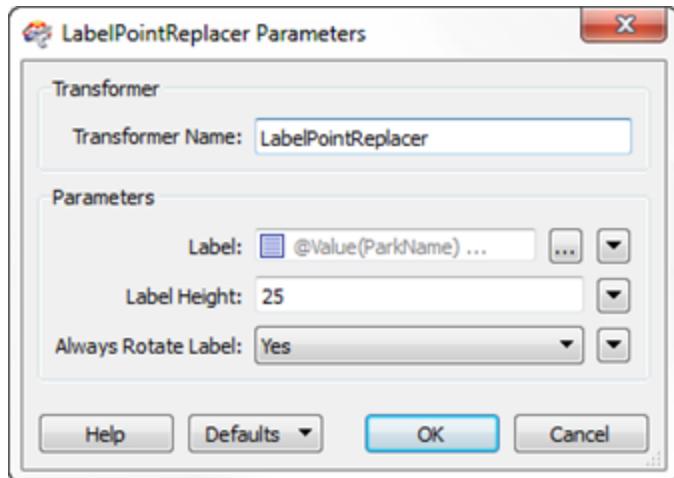
Click OK to close this dialog.

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.



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". If in doubt, a tooltip is often provided to point the way.



5) Run the Translation

Run the translation and inspect the output.

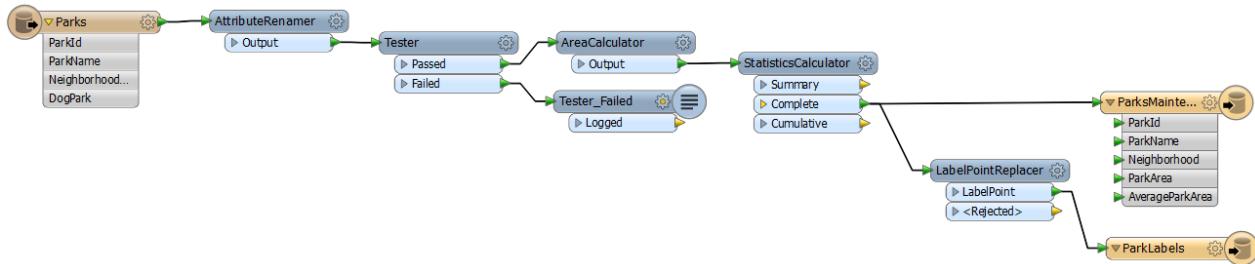
Notice that the output is in two layers in two files. Use the FME Data Inspector to open both output files in the same view.



MapQuest: Tiles Courtesy of MapQuest

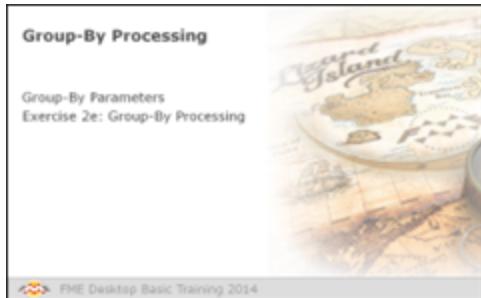
6) Save the Workspace

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



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.

Group-By Processing



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

FME transformers can 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) but 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." By default the group is ALL features entering the transformer. However, the "Group By" parameter in a transformer lets us define several groups based upon the value of an attribute.



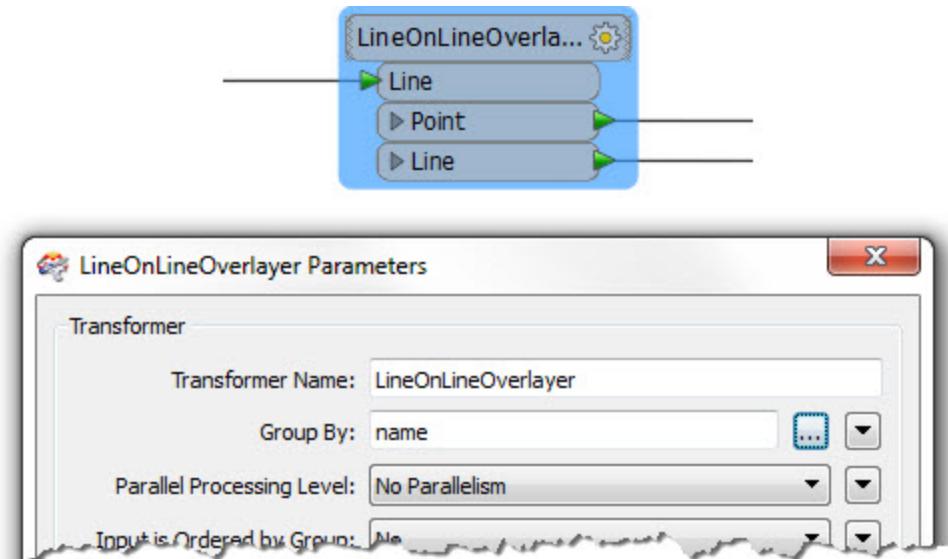
Mr. Statistics-Calculator, CFO, says...

"To illustrate the point consider calculating the average age of everyone in the room. This is the default group.

Now divide everyone up on into men and women. You have made a set of two groups and can calculate average age per gender.

This is the same as a group-by in FME."

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.

Exercise 2e Group-By Processing	
Demonstrates	Content Transformation, Group-By Processing
Scenario	FME user; City of Interopolis, Planning Department
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
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2e-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2e-Complete.fmw

In this example, imagine that you are a GIS technician working for a city planning department. 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 are to use FME to provide a dataset of this information.

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.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2d.

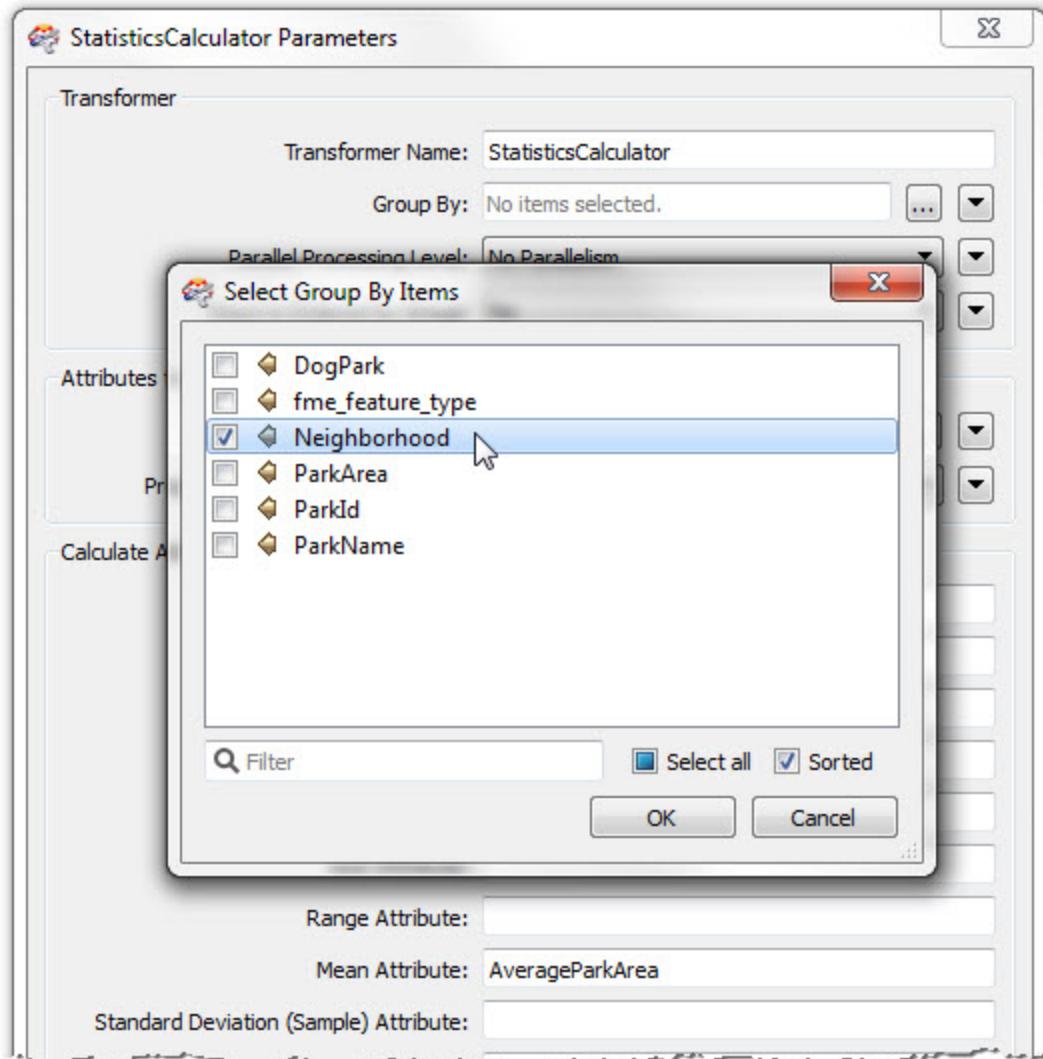
Alternatively you can open *C:\FMEData2015\Workspaces\DesktopBasic\Exercise2eBegin.fmw*

2) Set Group-By in StatisticsCalculator

This is a really simple task to do. Open the parameters dialog for the StatisticsCalculator transformer.

Click the 'browse' button next to the Group By parameter.

Select the attribute called Neighborhood and click OK until all dialogs are closed.



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:

ParkId	ParkName	Neighborhood	ParkArea	AverageParkArea
58	47 Point Grey Park Site at ...	Kitsilano	2190.28809109182	23986.3438250703
39	48 Hadden Park	Kitsilano	149477.314458903	23986.3438250703
40	49 Maritime Museum	Kitsilano	5722.22169147615	23986.3438250703
41	51 Vanier Park	Kitsilano	184668.339136154	23986.3438250703
42	3 Tea Swamp Park	Mount Pleasant	2631.26398618223	11660.2527620148
43	8 <missing>	Mount Pleasant	3123.72307219952	11660.2527620148
44	9 Creekside Park	Mount Pleasant	23975.705264418	11660.2527620148
45	10 China Creek South Park	Mount Pleasant	14750.3802947434	11660.2527620148
46	17 Mount Pleasant Park	Mount Pleasant	11177.7118351605	11660.2527620148
47	18 Major Matthews Park	Mount Pleasant	559.218942559321	11660.2527620148
48	19 Jonathan Rogers Park	Mount Pleasant	14022.1597369827	11660.2527620148
49	23 Robson Park	Mount Pleasant	15494.1787816619	11660.2527620148
50	24 Guelph Park	Mount Pleasant	10438.1870827894	11660.2527620148
51	25 Carolina Park	Mount Pleasant	373.699233488392	11660.2527620148
52	26 China Creek North Park	Mount Pleasant	31716.5521519775	11660.2527620148
53	4 <missing>	Strathcona	1984.83635717903	10196.9647106941
54	20 Thornton Park	Strathcona	14343.4149849573	10196.9647106941
55	27 Evans Yard Park	Strathcona	18480.8892025722	10196.9647106941

Data Inspection Using FME Workbench



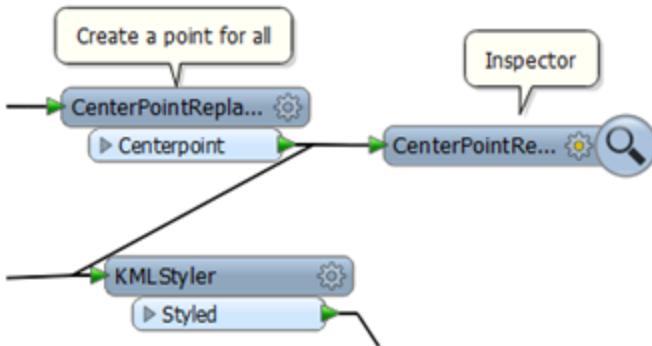
**Besides 'Redirect to Inspection Application',
Workbench can route data to the FME
Data Inspector from individual transformers.**

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.

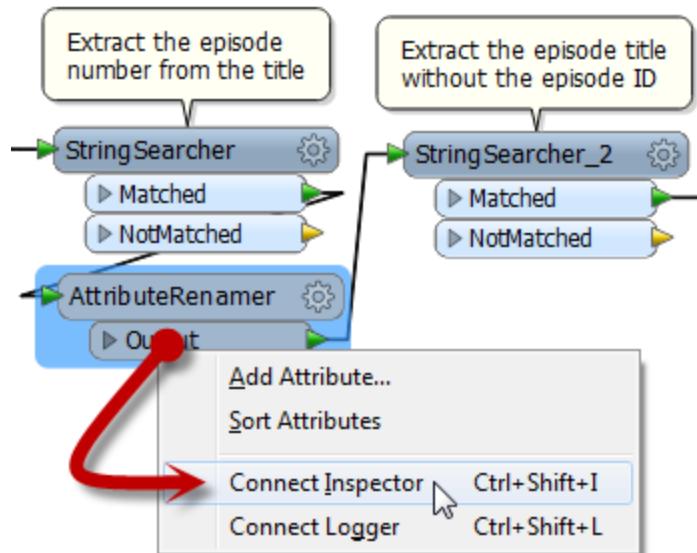
An *Inspector* transformer differs from the Redirect to Inspection Application setting because the transformer can be applied at any point in a translation (not just at the very end) and does not prevent the data being output to the writer. It also lets a user be more selective about which features should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

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



Placing an Inspector Transformer

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

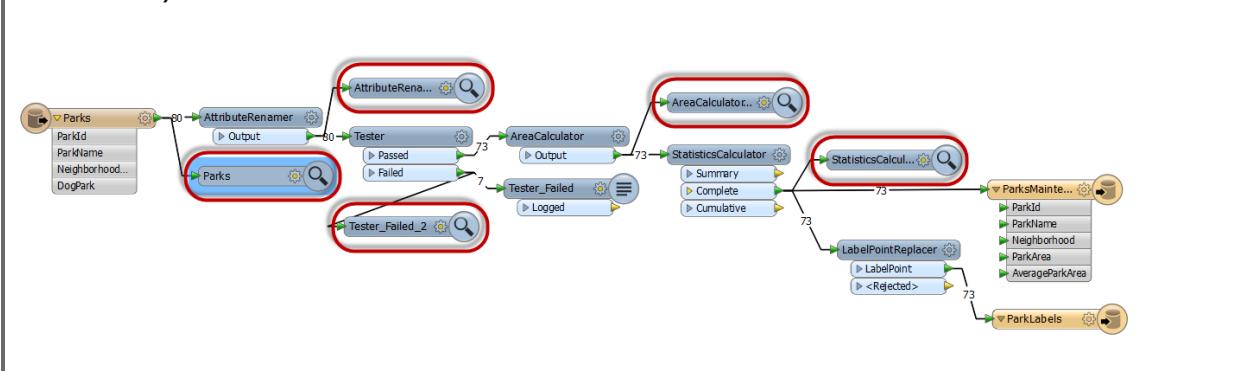


Here the user right-clicks the Output port of an AttributeRenamer and chooses the option **Connect Inspector**.

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



Mr. E. Dict, (Attorney of FME Law) says...
"Pursuant to clause 4a-(27) in your training course agreement, you are required to re-open the workspace from the previous example and add Inspector transformers to selected objects to practice using this functionality."



The diagram shows a workspace flow with several objects highlighted by red circles:

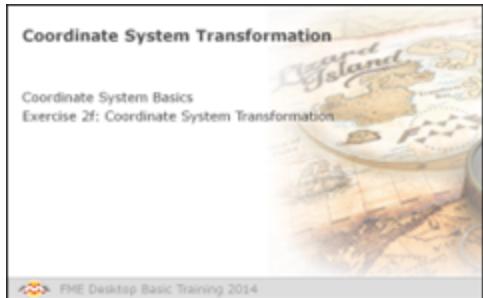
- An input source labeled "Parks" (containing fields ParkId, ParkName, Neighborhood, DogPark).
- An "AttributeRenamer" transformer.
- A "Tester" transformer.
- A "Parks" object (highlighted again).
- A "Tester_Failed_2" transformer.
- An "AreaCalculator" transformer.
- A "StatisticsCalculator" transformer.
- A "LabelPointReplacer" transformer.
- An output target labeled "ParksMaintenance" (containing fields ParkId, ParkName, Neighborhood, ParkArea, AverageParkArea).
- A final output object labeled "ParkLabels".

Connections between these objects are shown with arrows, and numerical values (e.g., 60, 73, 7) are placed near some connections.



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!

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.

Coordinate System Basics

Each feature that is processed by FME is coordinate system aware; that is, it knows what coordinate system it belongs to at all times. This helps prevent confusion when reading multiple datasets that belong to different coordinate systems.

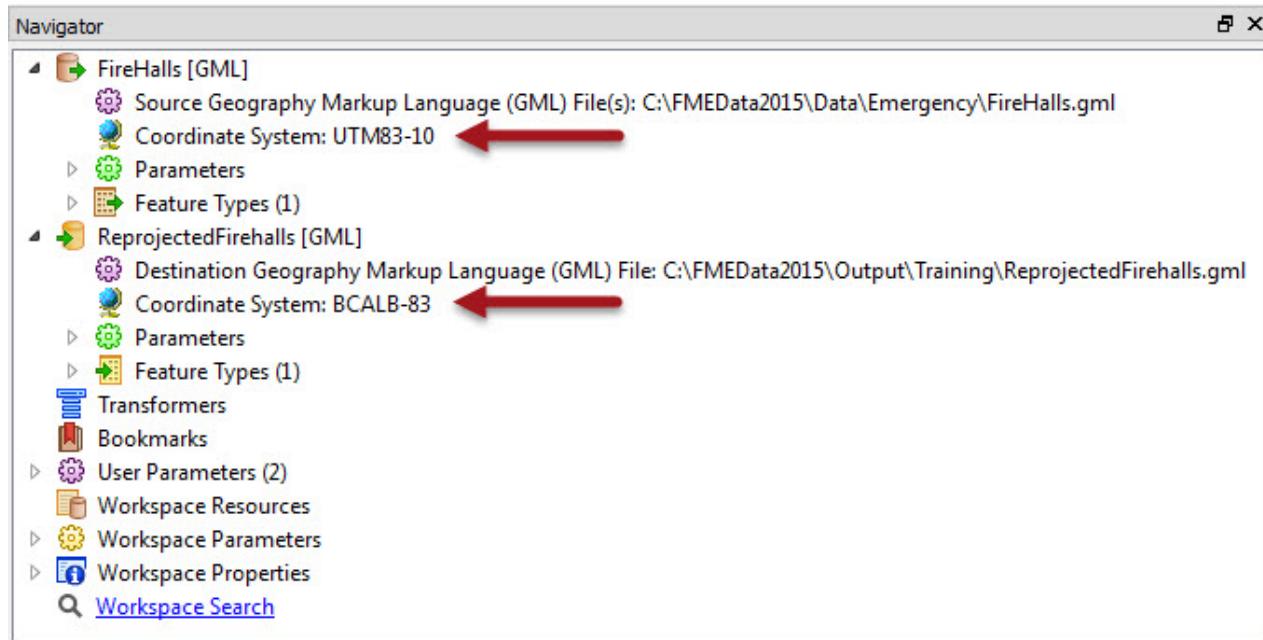


Some users call this location of data a 'projection,' but projection is just one component of a definition within space. A true definition includes projection, datum, ellipsoid, units, and sometimes a quadrant, which together is called a 'Coordinate System.'

Coordinate System Settings

Each source reader and destination writer within FME is assigned a coordinate system. That coordinate system is set in the navigation pane of Workbench.

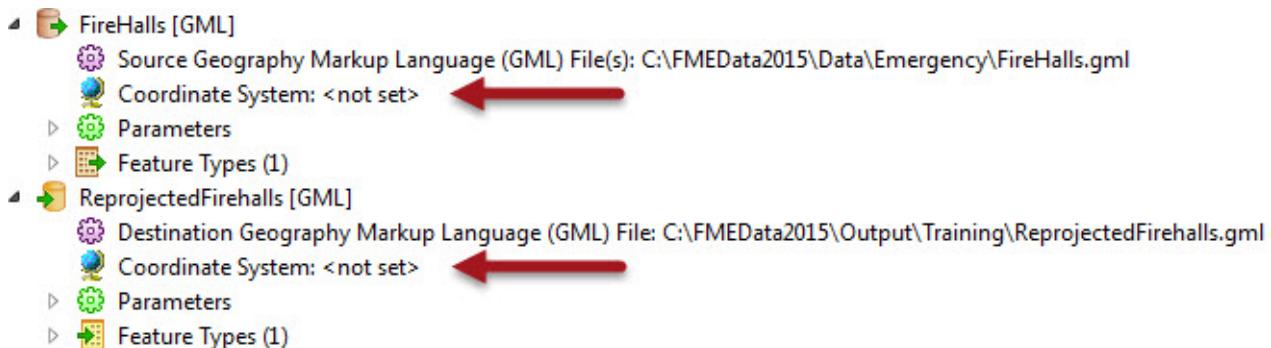
Here the source coordinate system has been defined as UTM83-10 and the destination as BCALB-83.



When a translation is carried out where the source and the destination coordinate systems differ in this way, FME automatically restructures the data, at the end of the translation, so that the output is in the correct location.

Automatic Detection of Coordinate Systems

Some data formats are capable of containing information about the coordinate system in which they are held. Shape format is one example of this. FME can be set to detect any such information.



Because the source Reader coordinate system is marked <not set>, FME will try to determine the coordinate system from the source dataset.

Because the destination 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 source data.

Exercise 2f Basic Reprojection	
Scenario	FME user; City of Interopolis, Planning Department
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, Reprojection
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2f-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2f-Complete.fmw

In this example, imagine that you are a GIS technician working for a city planning department. 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 are to use FME to provide a dataset of this information.

The parks team has decided that the output data should be in an Albers Equal Area projection (coordinate system = BCALB-83)

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 2e.

Alternatively you can open C:\FMEData2015\Workspaces\DesktopBasic\Exercise2fBegin.fmw.

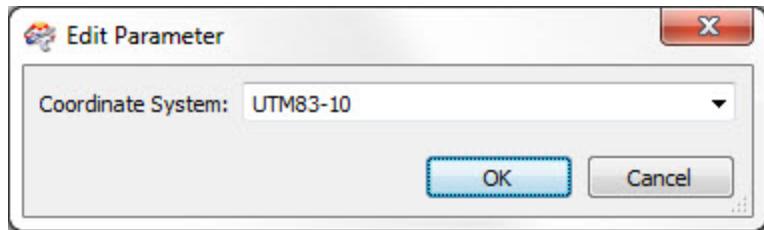
2) Edit Reader Coordinate System

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.



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

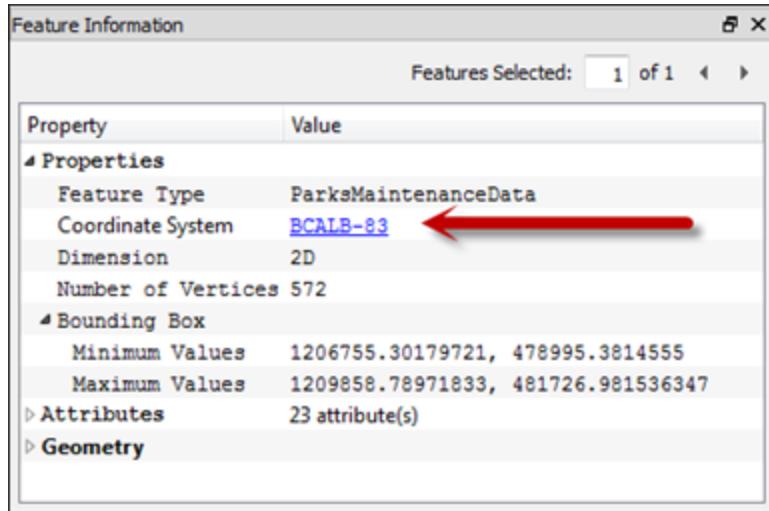
Save and then run the workspace.

In the log file you should be able to find...

```
FME Configuration: Source coordinate system set to `UTM83-10'
```

```
FME Configuration: Destination coordinate system set to `BCALB-83'
```

5) Inspect the Output



Open the FME Data Inspector. Choose **Tools > FME Options** and turn off the background map. If the background map is activated then the data is automatically reprojected to match, and this would not help us verify the results of the translation.

Open the newly reprojected dataset. Query a feature. The Feature Information window should report that the data is now in BCALB-83



Reprojection can also take place using transformers – in fact this might be considered the better method because the transformers include extra parameters for controlling the reprojection.

Module Review



This module was designed to introduce you to the concept of Data Transformation and to learn 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 restructure data by adjusting the schema mapping, both manually and through transformers.
- The ability to locate transformers in Workbench and place them in the processing stream (by all methods) to transform data content.
- The ability to define feature groups using the Group-By option.
- The ability to reproject data using Workbench.


Answers
Q&A


Miss Vector says...

"Here are the test answers. Don't you dare get #1 wrong!"

FME's ability to manipulate data is called:

- 1) Translation
- 2) Transmogrification
- 3) Transfiguration
- 4) Transformation

✓

The source and destination parts of a schema show:

- 1) What we have/What we want
- 2) What we did/What we should have done
- 3) What we're adding/What we're removing
- 4) Where my data was/Where it is now

✓

Deciding how the source schema connects to the destination is called:

- 1) Schema Mapping
- 2) Schema Connecting
- 3) Schema Joins
- 4) Schema Concatenating

✓

Which of these isn't a color of connection arrow on an FME schema?

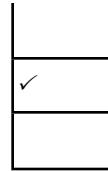
- 1) Red

--

2) Green

3) Blue

4) Yellow



Mr. Flibble's challenge:

The most common FME translation is from Esri Shape to..... Esri Shape.

In other words, users are using FME for the transformation instead of the translation capabilities!



Mr. CAD says...

'Like many people, I bought FME solely for format translations. But the Data Transformation tools opened a flood of new possibilities! I now use FME to transform data, even when it doesn't need a format change!'

Data Transformation

Exercise 2g Data Transformation	
Scenario	FME Workspace Author
Data	Election Mapping (GML) Election Statistics (Excel)
Overall Goal	Map statistics of voting patterns
Demonstrates	Basic Data Transformation
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2g-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2g-Complete.fmw
	C:\FMEData2015\Workspaces\DesktopBasic\Exercise2g-CompleteAdvanced.fmw

In this exercise, the municipal elections officer has asked for your help in identifying voting divisions that had a low turnout at the last election, or that had difficulty understanding the voting process. The results should be presented in Google Earth 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:\FMEData2015\Data\Elections\ElectionVoting.gml
Reader Format	Microsoft Excel
Reader Dataset	C:\FMEData2015\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

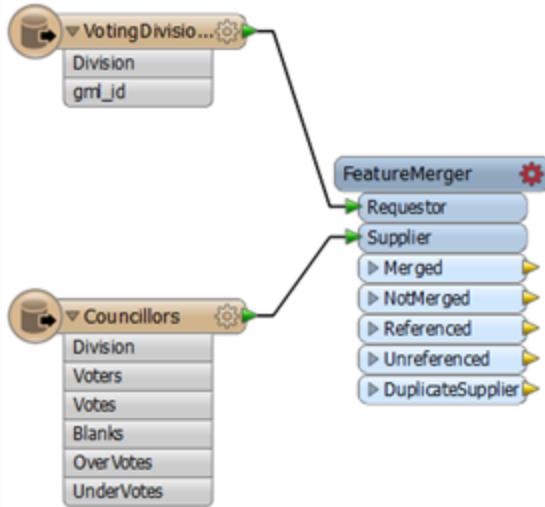
Start Workbench and open the starting workspace. It already has Readers and Writers added to handle the data; all we need to do is carry out the transformation:



3) Add FeatureMerger

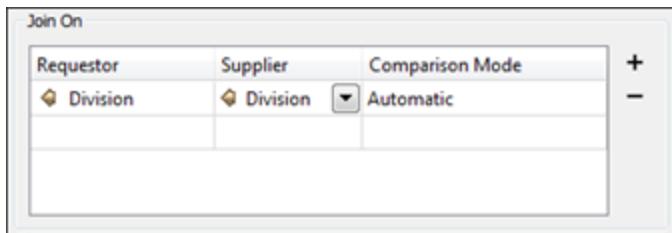
The first task is to merge the statistical election data onto the actual features. We'll use a FeatureMerger transformer to do this (more on this transformer appears later in this course).

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



4) Set Parameters

Click the cog-wheel icon to open the FeatureMerger parameters dialog. For both the Requestor and Supplier join fields, click the drop-down arrow and choose the Division attribute. This is the common key by which our data is merged:



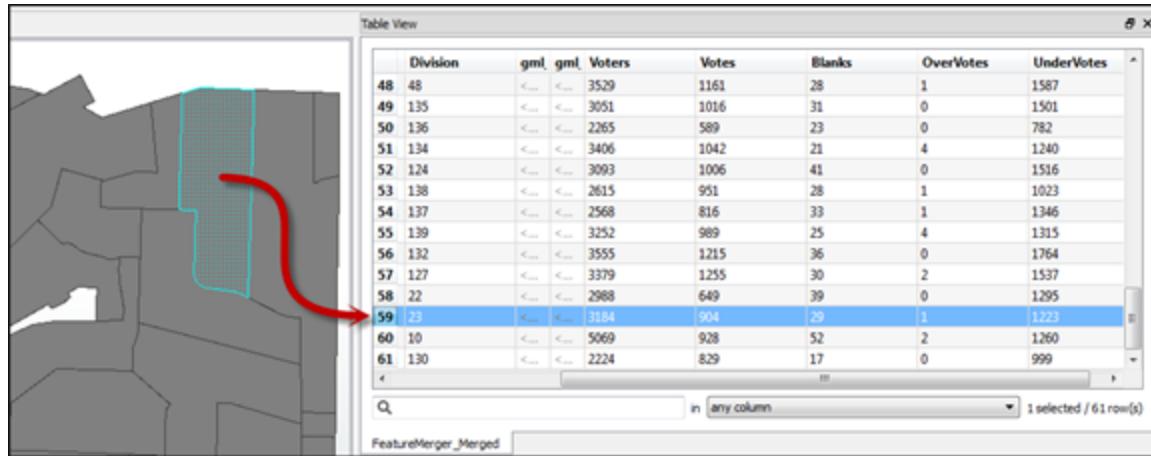
Click the OK key to close the dialog.

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:



6) Add ExpressionEvaluator

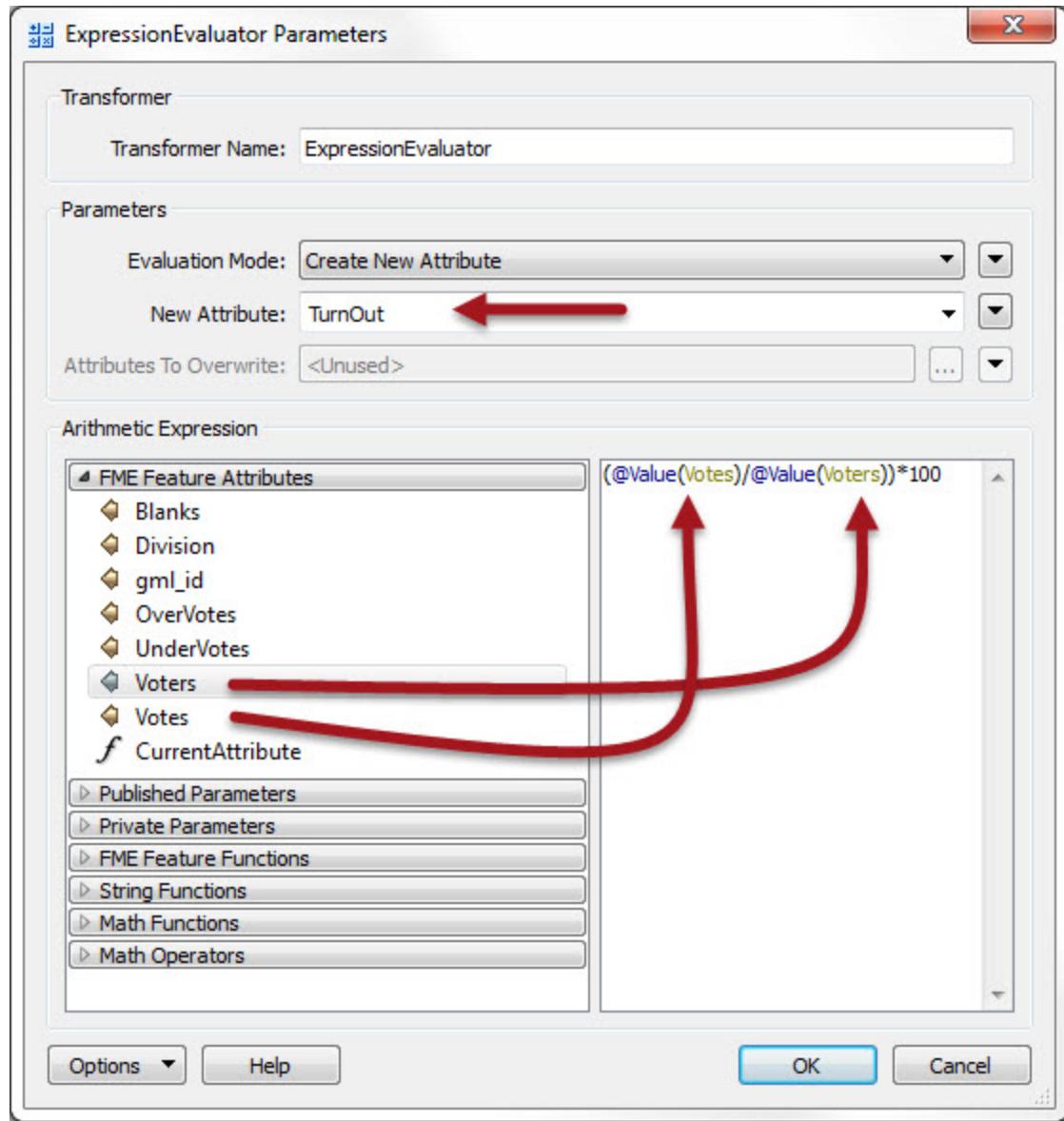
Now that we have the numbers we need, we can start to calculate some statistics. To do this we'll use an ExpressionEvaluator transformer to first calculate voter turnout percentage for each division. Place an ExpressionEvaluator transformer after the FeatureMerger - connect it to the FeatureMerger's Merged output port - and open the parameters dialog.

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 OK to close the dialog. If you wish you can reconnect an Inspector and re-run the translation, to see what the result is.

7) Add ExpressionEvaluator

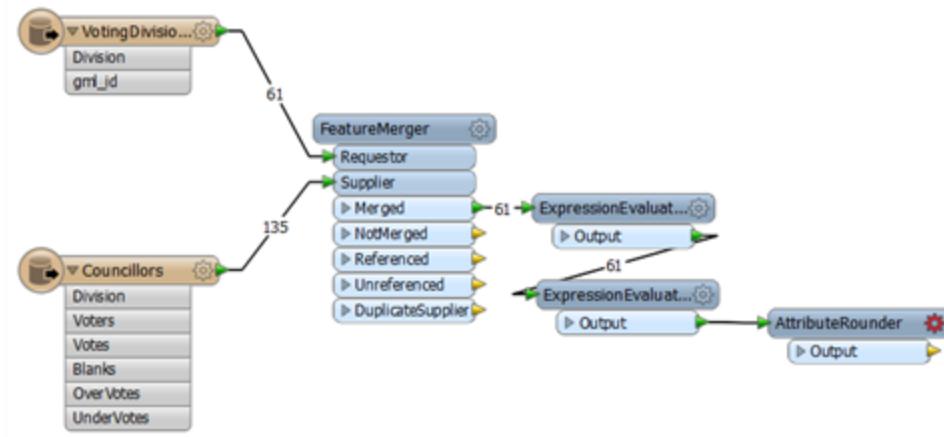
Using a similar technique, use an ExpressionEvaluator to calculate the number of UnderVotes per voter and put it in an attribute that matches the output schema. The expression will be something like:

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

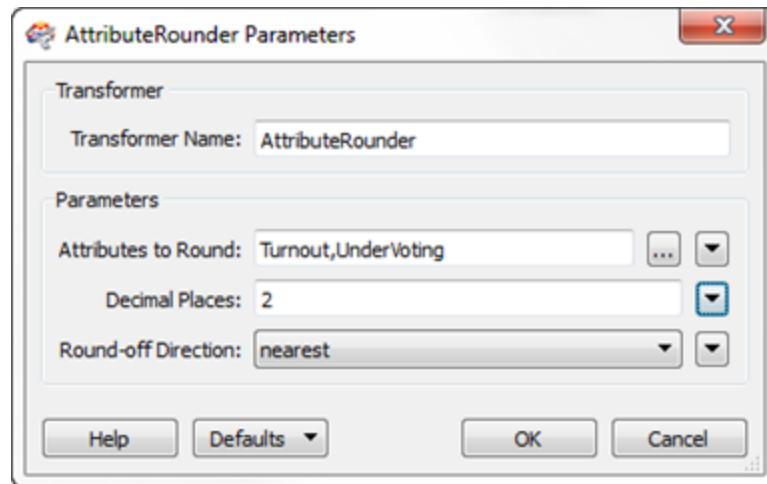
8) Add AttributeRounder

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.



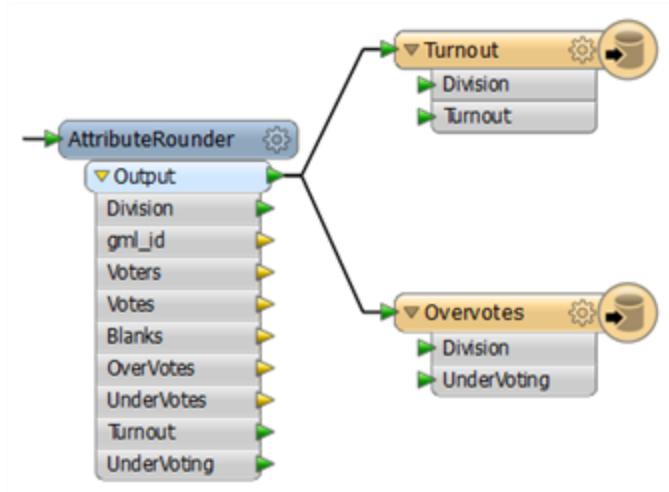
Open the parameters dialog. Under Attributes to Round select the newly created Turnout and UnderVoting attributes. Set the number of decimal places to 2:



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:



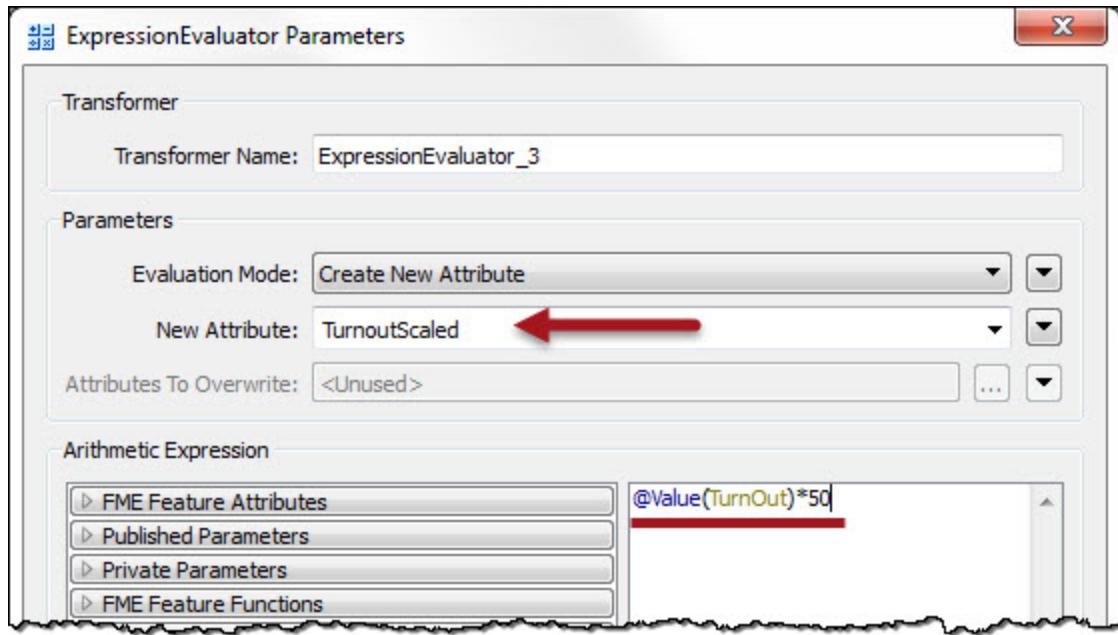
Run the workspace and examine the output in Google Earth to prove it has the correct attributes and is in the correct location.



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.

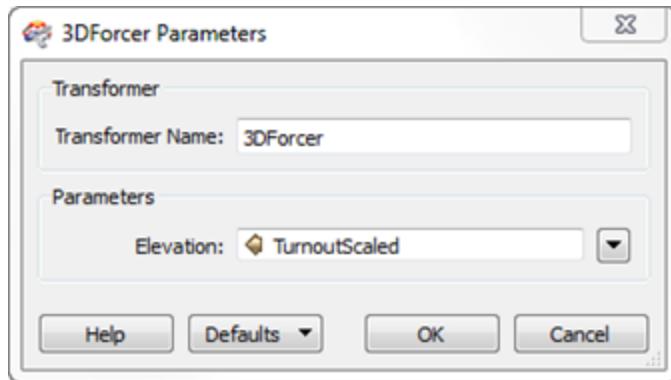
10) Add ExpressionEvaluator

The height of each block should be proportional to the turnout for that division. To exaggerate the scale, place an ExpressionEvaluator and use it to multiple the Turnout attribute by a value of 50. Put it into a new attribute called TurnoutScaled.



11) Add 3DForcer

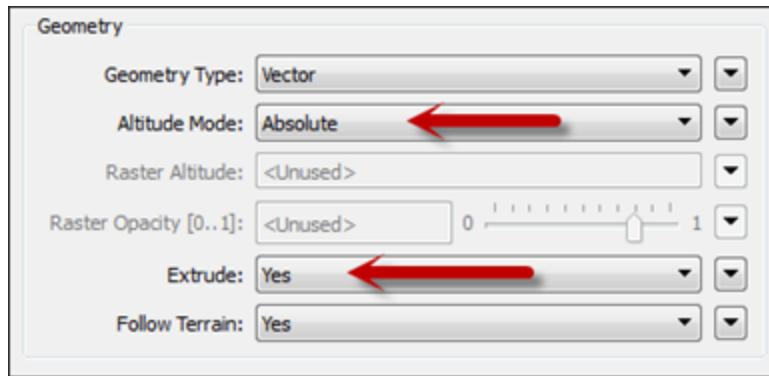
Add a 3DForcer transformer. This will elevate the feature to the required height. Open the parameters dialog and set the elevation to **Attribute Value > TurnoutScaled**.



12) Add KMLPropertySetter

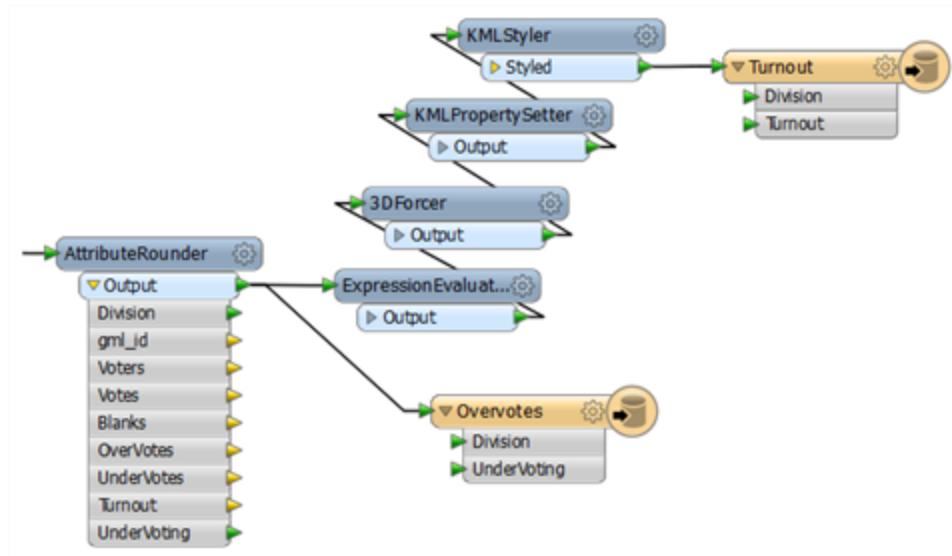
Add a KMLPropertySetter transformer. This will allow us to set up the 3D blocks in the output. Open the parameters dialog. Set:

- Altitude Mode: Absolute
- Extrude: Yes

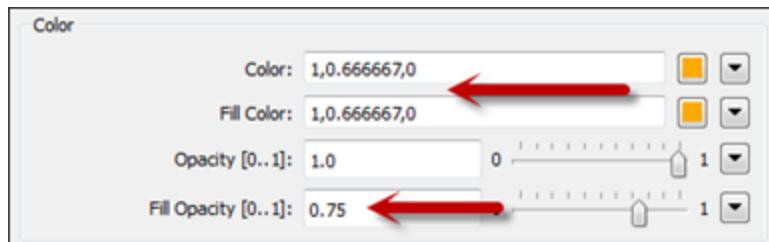


13) Add KMLStyler

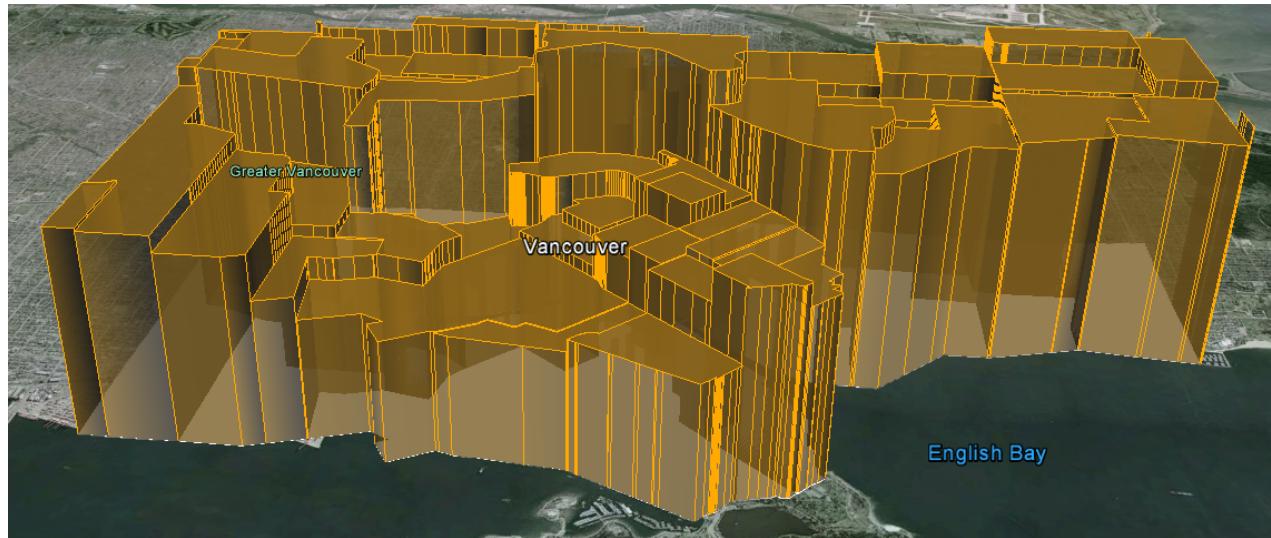
Finally add a KMLStyler transformer. The workspace will now look like this:



Open the parameters dialog. 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:



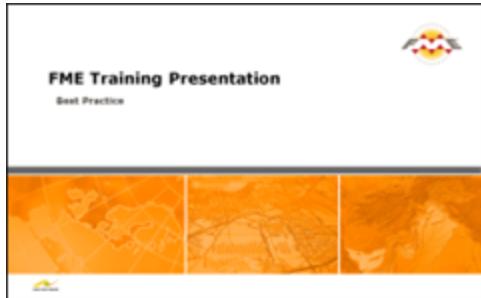
These 3D blocks will show users where the voting turnout is high/low in the city.

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



Congratulations! You have now completed the exercise for this chapter.

Chapter 3 - Best Practice



In a corporate setting, it's not enough just to know FME functionality and terminology. 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"

In this manual we'll cover four different categories of Best Practice.

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.

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.

Debugging

This section covers tools and methods to help identify and fix problems in translations.

Organization

This section covers what functionality exists to organize FME within a workplace. Sharing resources among several FME users is one technique made possible by special FME functionality.



Each section will have a number of recommendations. These are suggested methods that meet the principles of best practice in FME.

Why use Best Practice?

Best Practice in FME can help a user to...

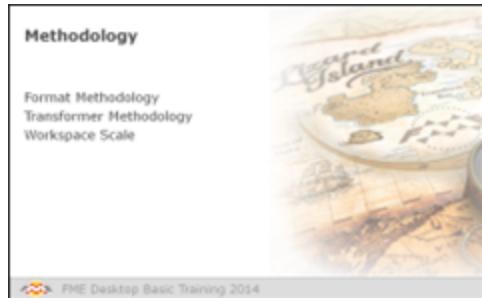
- Create well-styled workspaces that are self-documenting
- Create workspaces that use the correct functionality in the correct place
- Create high-performance workspaces
- Use FME Workbench in a way that's most efficient
- Debug a workspace when it doesn't work the way intended
- Use FME in a project-based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



Dr. Workbench says...

'I 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 taken.'

Methodology



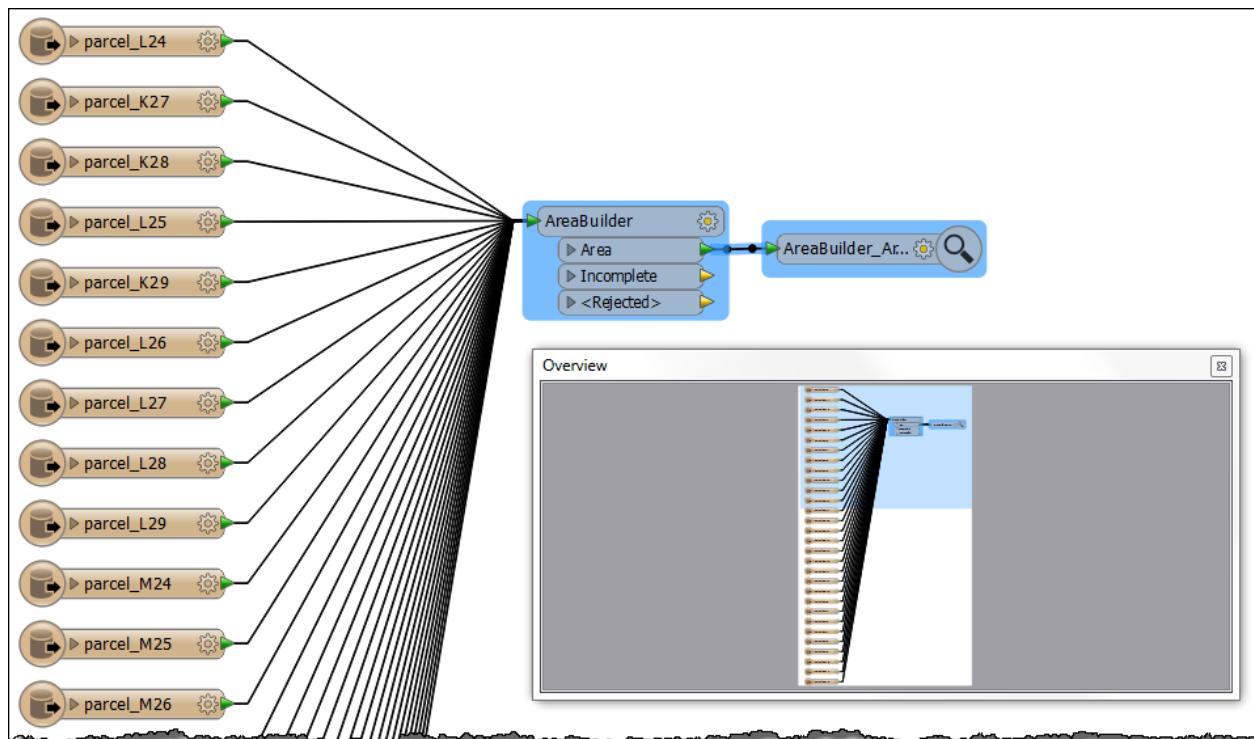
Using Workbench the right way makes for a more productive and efficient experience.

Format Methodology

Format best practices mostly involve setting up readers and writers in a clear and uncluttered manner in the workspace canvas.

Minimizing Feature Type Objects

A schema definition may contain many, many Feature Types. If these are all listed separately then the workspace can become extremely unwieldy and poorly laid out.





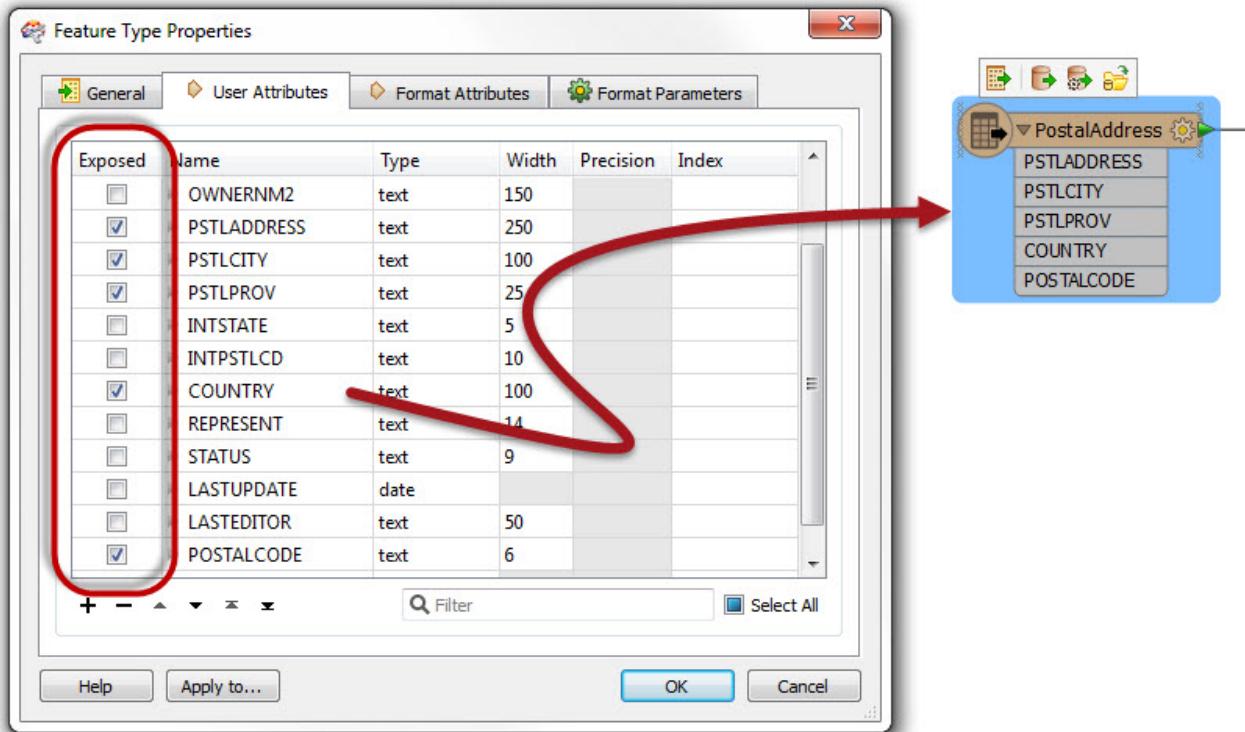
When creating a workspace you'll be prompted which feature types to add to the canvas. Don't add feature types you don't need, as it will clutter the canvas.

Minimizing Source Attributes

Excessive attributes can also cause workspace clutter. This can be relieved using some functionality to hide (unexpose) unwanted attributes.

This functionality can be found in a Reader Feature Type Properties dialog.

For example, here the user has hidden a number of attributes that are unnecessary for their workspace. Only the required attributes remain exposed:



This makes the source schema, and all transformer dialogs, much clearer and tidier than if the entire set of attributes was still exposed.

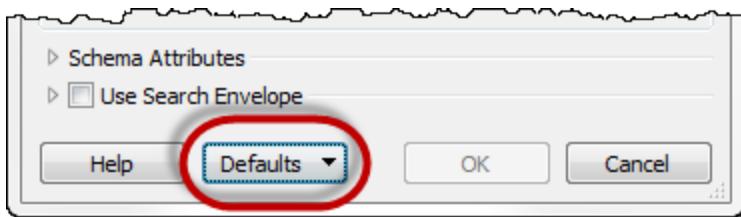


Hide any attributes you don't need to use in the translation, as it will make the workspace tidier. Plus, with fewer connections to track, the Workbench canvas will respond faster to edits.

Format Defaults

Each Reader and Writer parameter has a default value when FME is installed.

However, a user may find they always need a different value than the default whenever a new workspace is created. For example, if they always read MicroStation data in the same way (for example they always want to preserve multi-text as a single feature) it would be useful to permanently set the relevant parameters so that they are used by all new workspaces. This can be achieved using the Defaults button at the foot of each Reader/Writer parameters dialog:



Now whenever a specific format is used, the parameters already have predefined defaults.



If you create the same workspace repeatedly, then use the Defaults options to avoid having to enter the same parameter values each time.



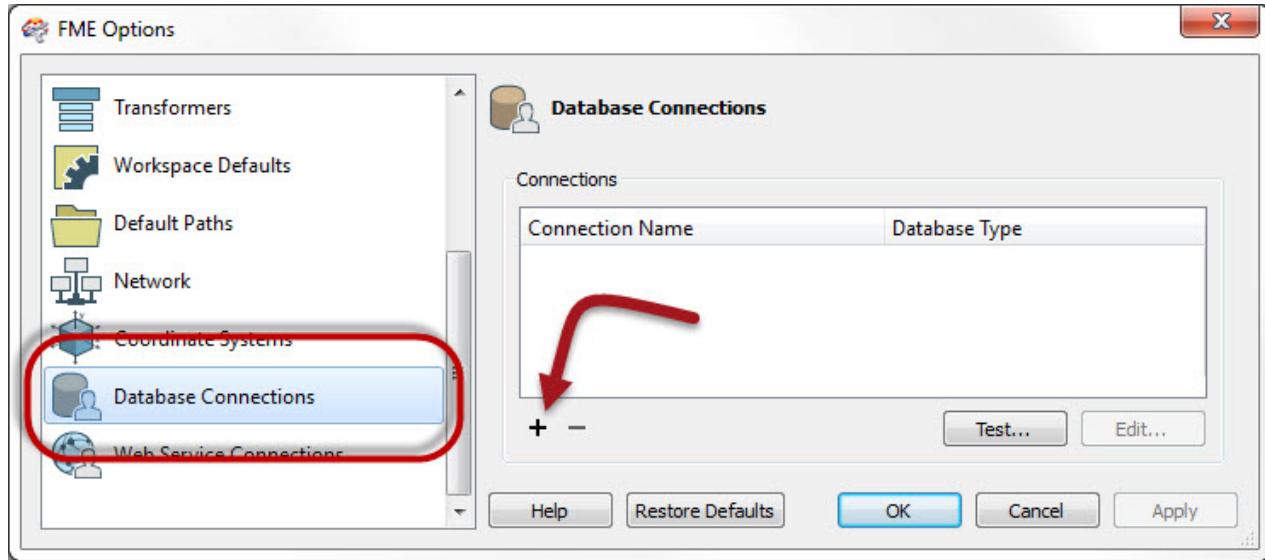
Database Connections

For database formats, the main repetition is entering the same connection and authentication parameters whenever a Reader or Writer is added, or when a database connection is required inside a transformer.

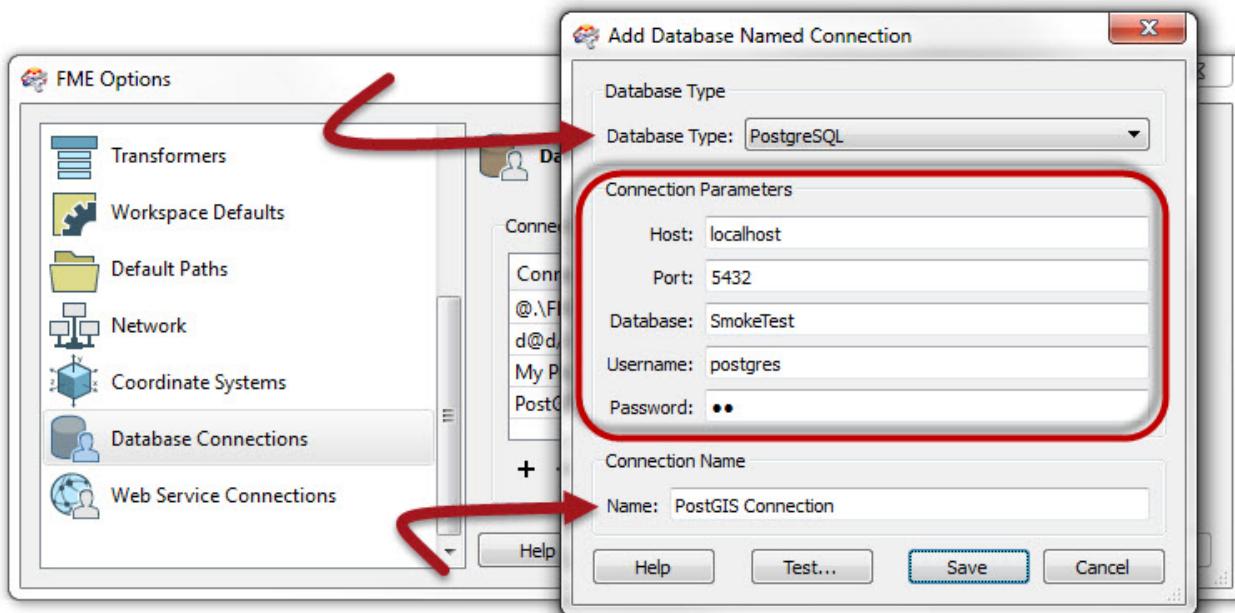
That could be solved by using format defaults, but a better method is to use Database Connections, a piece of functionality new to FME2015.

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.

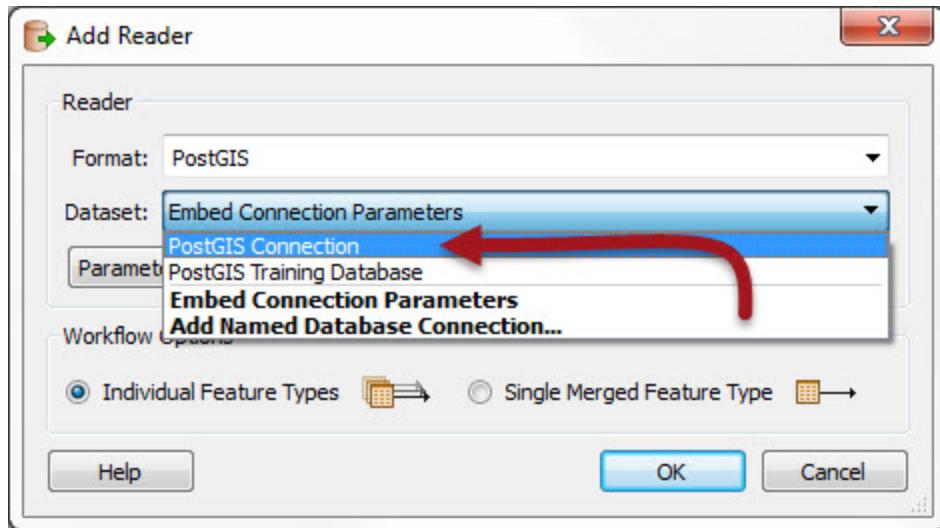
Database Connections are created in a dialog opened by selecting Tools > FME Options from the menubar, and then clicking on a Database Connections icon. A new connection is created by clicking the + button:



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 a quicker process - in any workspace you create - but workspaces that you send to other people will be more secure, because they won't include any database authentication details.

! Obviously workspaces that make use of new functionality like Database Connections are not compatible with previous versions of FME. Therefore don't switch over to Database Connections until you are certain that you are ready to migrate fully to FME2015.

Tip If you read from or write to databases on a regular basis, be sure to create database connections to avoid repeatedly entering the same connection and authentication parameters.

Transformer Methodology

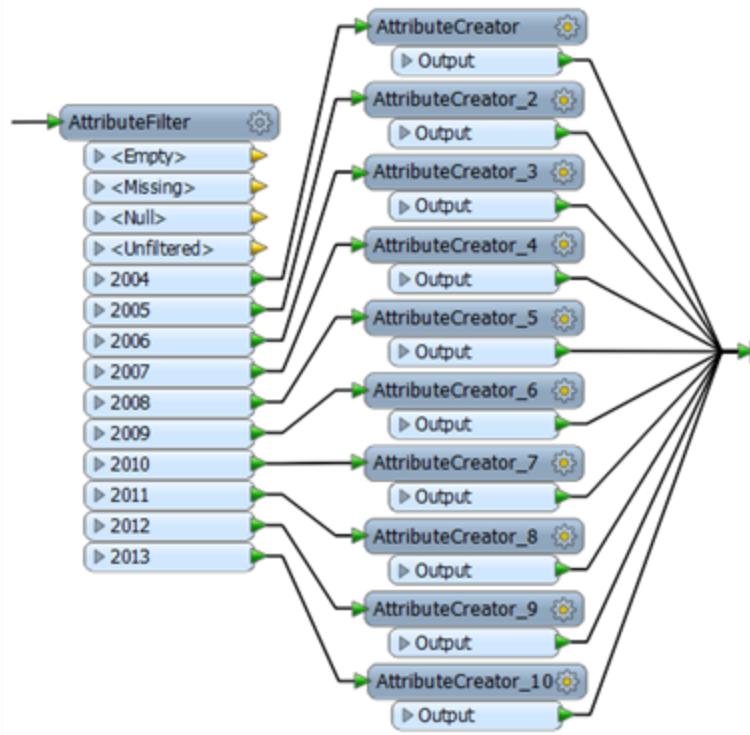
As part of the business of creating translations, there are a number of ways to use FME in an efficient and skillful manner. Some of these apply specifically to the use of transformers.

The Right Transformer

Although there are usually several ways to achieve the same goal with different transformers, the methods are not always equally efficient. It's important not to assume that the first way is the best way.

If a workspace duplicates the same transformer again, and again – or divides data up into multiple streams to process it only slightly differently – then it is probable that the workspace has been designed very inefficiently.

The multiple attribute transformers here indicate there is a problem.



If you're unsure as to what transformer to use, or are worried that your workspace has multiple streams of data for no good reason, then consult other users in the FME community, or the Safe Software support team.

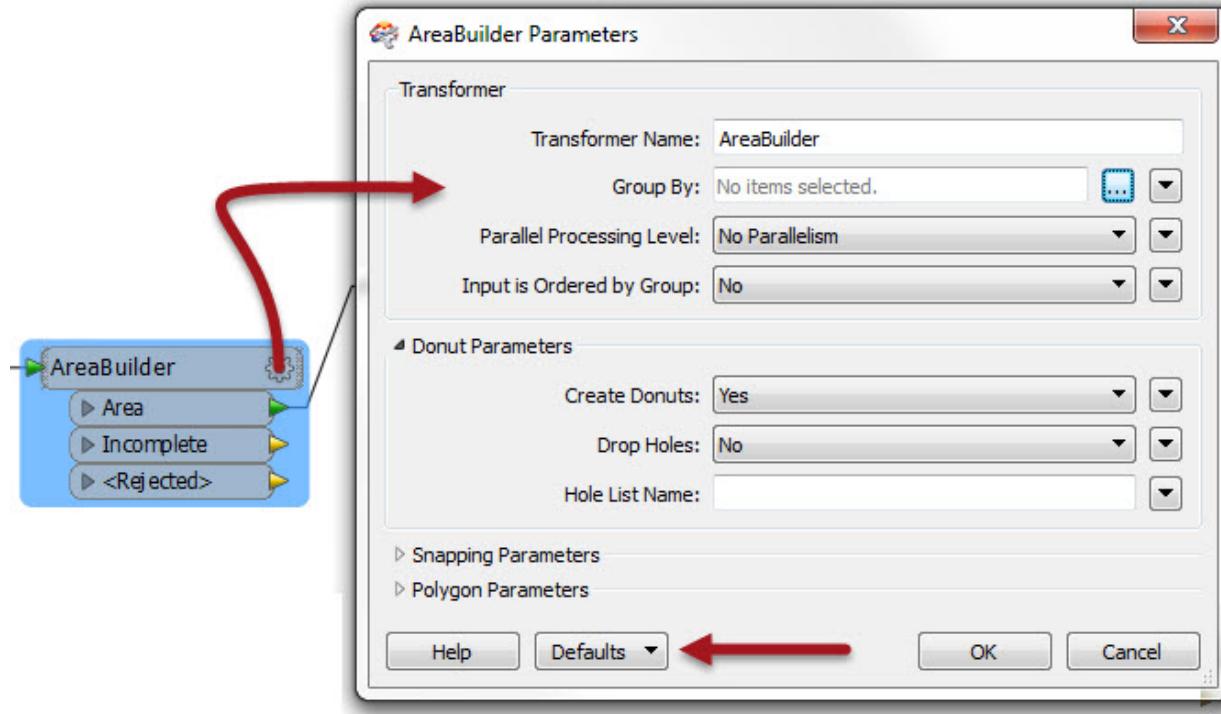
In the above example, a single *AttributeValueMapper* transformer could be used to replace all of the above transformers.

Default Values

As with reader and writer dialogs, transformer parameter dialogs also possess a button to set the current values as future defaults.

In this parameters dialog for the *AreaBuilder* transformer, the Defaults button is highlighted.

By entering a set of values and saving them as the current defaults, each newly placed instance of this transformer inherits the same parameter values.



Multi-Workspace Translations

Although FME has the capability to run one workspace from another – using the *WorkspaceRunner* transformer – this is designed for the specific situation of batch processing.

In most other cases a user should never need to split a translation over two or more workspaces.

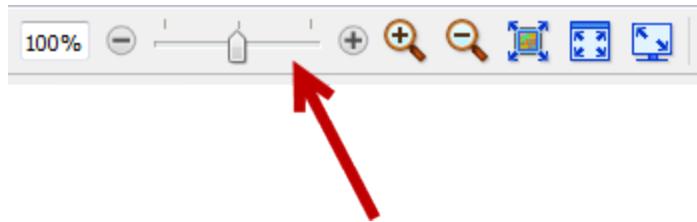


If you're in a situation that requires the use of two workspaces to carry out one translation, then reconsider your decision and consult other FME users for advice.

Workspace Scale

You might have noticed view tools for zooming in and out of a workspace, but did you know that there is a default zoom scale (100%) at which transformer are shown in "actual size"?

To revert to this optimum zoom level you can either drag the scale bar to 100% or double-click anywhere on the scale bar.



Working with a 100% zoom is the optimum scale for viewing FME canvas objects.

Style



A well-styled workspace provides many benefits as it is further developed and edited in the future.

An FME Workspace Style Guide

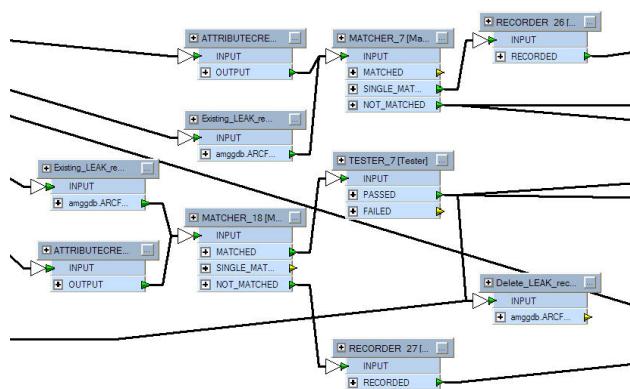
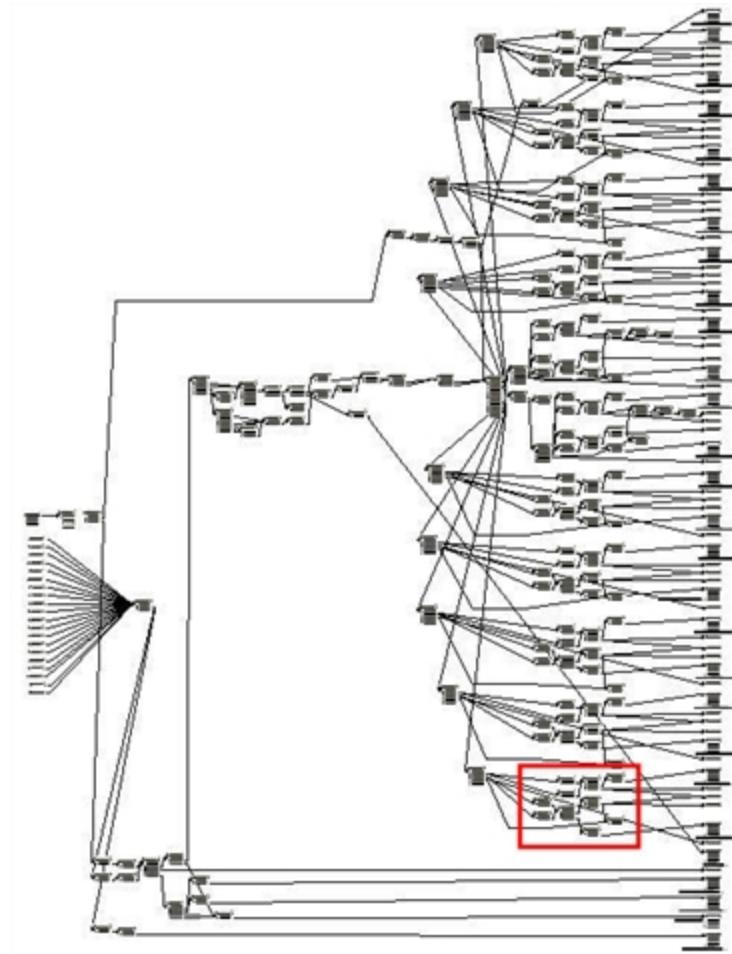
Workspaces are rarely static or used repeatedly without being edited; sooner or later the source or destination schema will change, or edits will be needed to take advantage of new or updated FME functionality.

A good style of design makes it easier to navigate and understand an existing workspace, and thus simplifies the editing process. It's very much like a computer programmer adding comments to explain his 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



Who would envy a user given the task of editing this workspace?

Annotating Workspaces

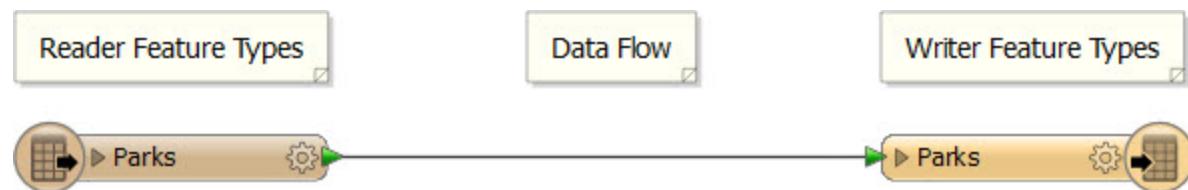
Annotation is a key method for a clear and comprehensible design.

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

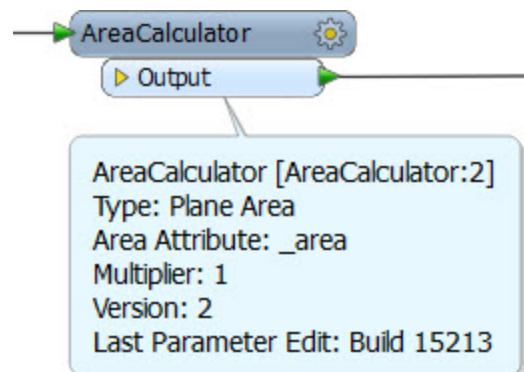
By default FME adds three annotations to a newly-created workspace. The annotations are basic comments that indicate the source data, the transformation, and the destination data. They can be deleted or, optionally, left ungenerated, and therefore may not appear in every workspace.



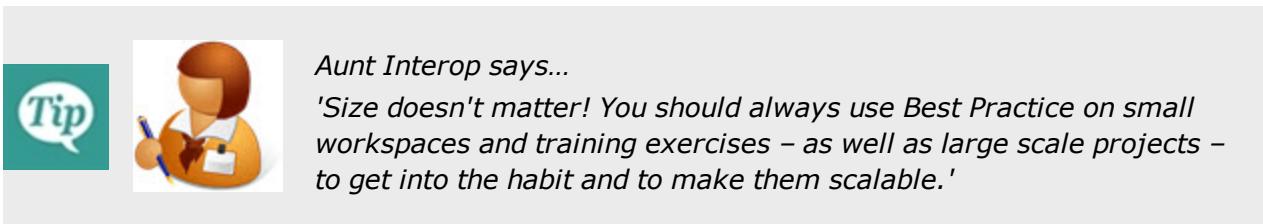
Summary Annotation

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.

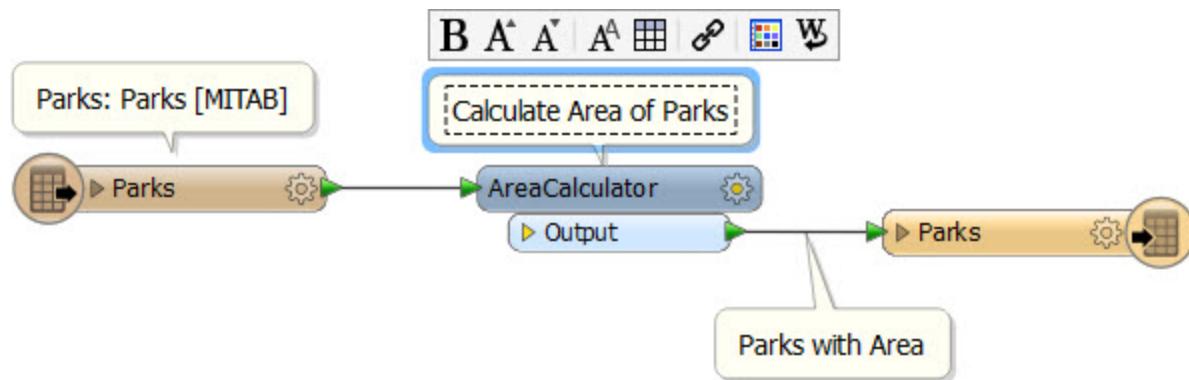


The nice thing about Summary Annotation is that it automatically updates in response to changes. It's also useful in situations where the transformer parameters are set through a wizard (for example, the *SchemaMapper* or *FeatureReader*) and would take longer to check that way.



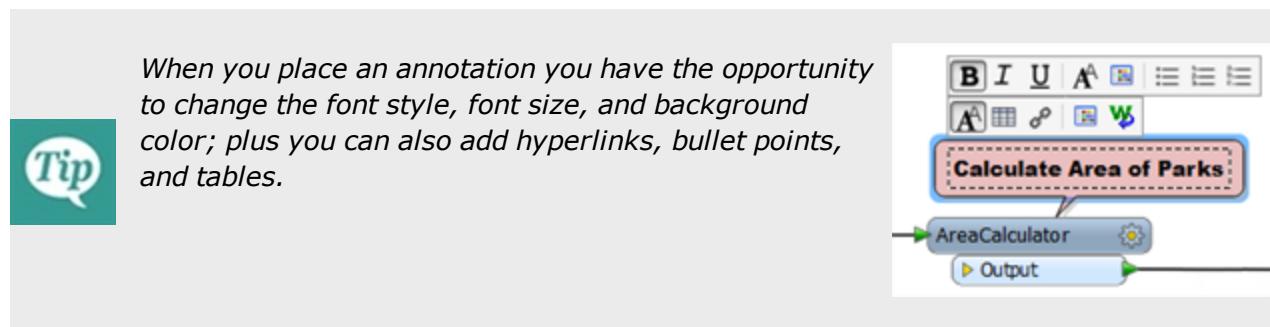
User Annotation

User annotation is a comment created by the user. It can be connected to a particular workspace object or 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.

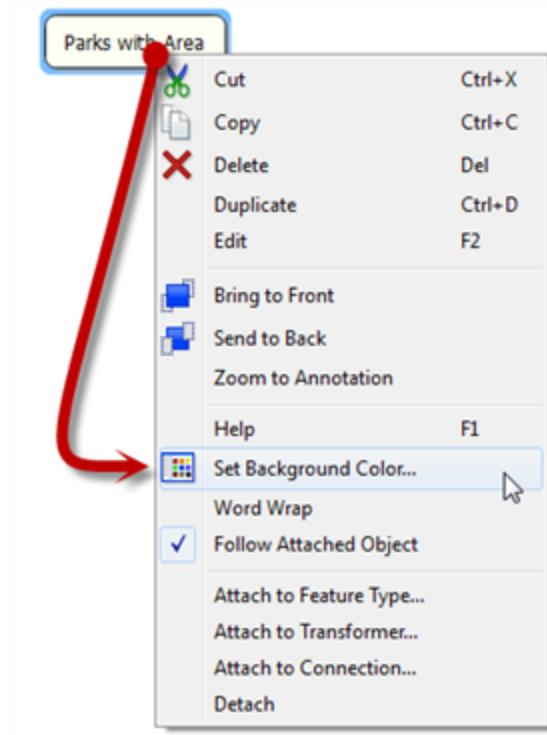




Create user annotation on nearly all transformers, to help future edits and updates go more smoothly.

Annotation Options

Right-click an annotation object to reveal (among other things) the following options:



- **Set Background Color** - change the background color of the annotation
- **Word Wrap** - Toggle word wrapping on and off
- **Follow Attached Object** – make the annotation move when the attached object is moved
- **Attach to Feature Type** – attach to a specific feature type
- **Attach to Transformer** – attach to a specific transformer
- **Attach to Connection** – attach to a specific connection between two objects
- **Detach** – detach annotation from an object

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:** For dividing up a workspace into different - clearly marked - sections.
- **Quick Access:** For placing a marker for quick access to a certain part.
- **Organization:** For moving about groups of transformers at a time.

Adding a Bookmark

To add a bookmark, click the Bookmark icon on the toolbar.



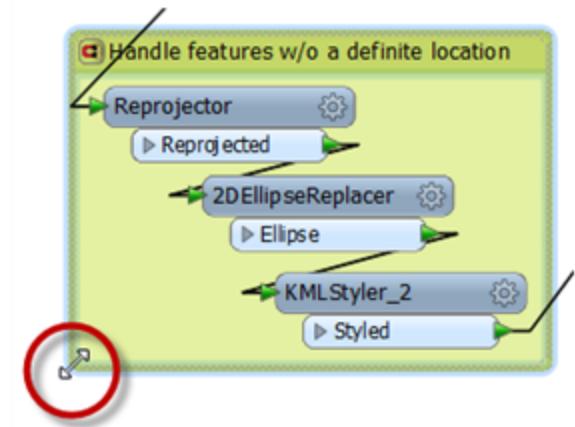
Whereas a traditional bookmark marks just a single page in a book, the FME bookmark can cover a wide area of the canvas. A single workspace can be divided into different sections by applying multiple bookmarks.



If any objects on the workspace canvas are selected when a bookmark is created, the bookmark expands 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.



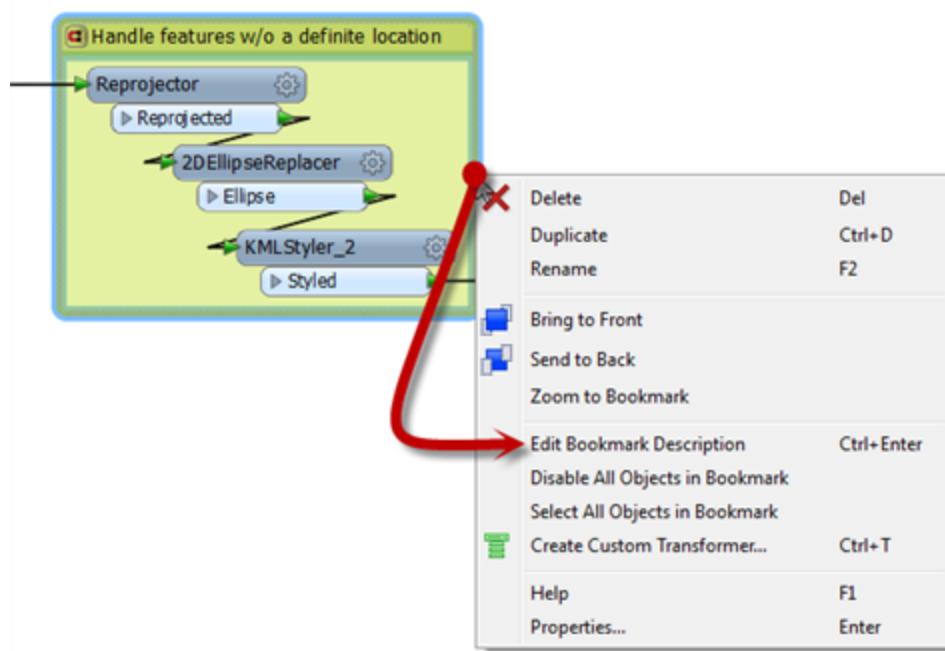
Double-clicking a bookmark title allows the color and title of the bookmark to be edited.



*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.*

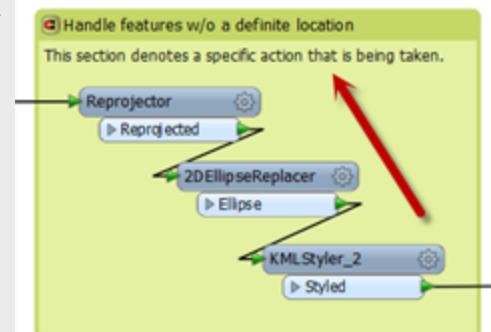
Bookmark Options

Right-click a bookmark to show the following options:



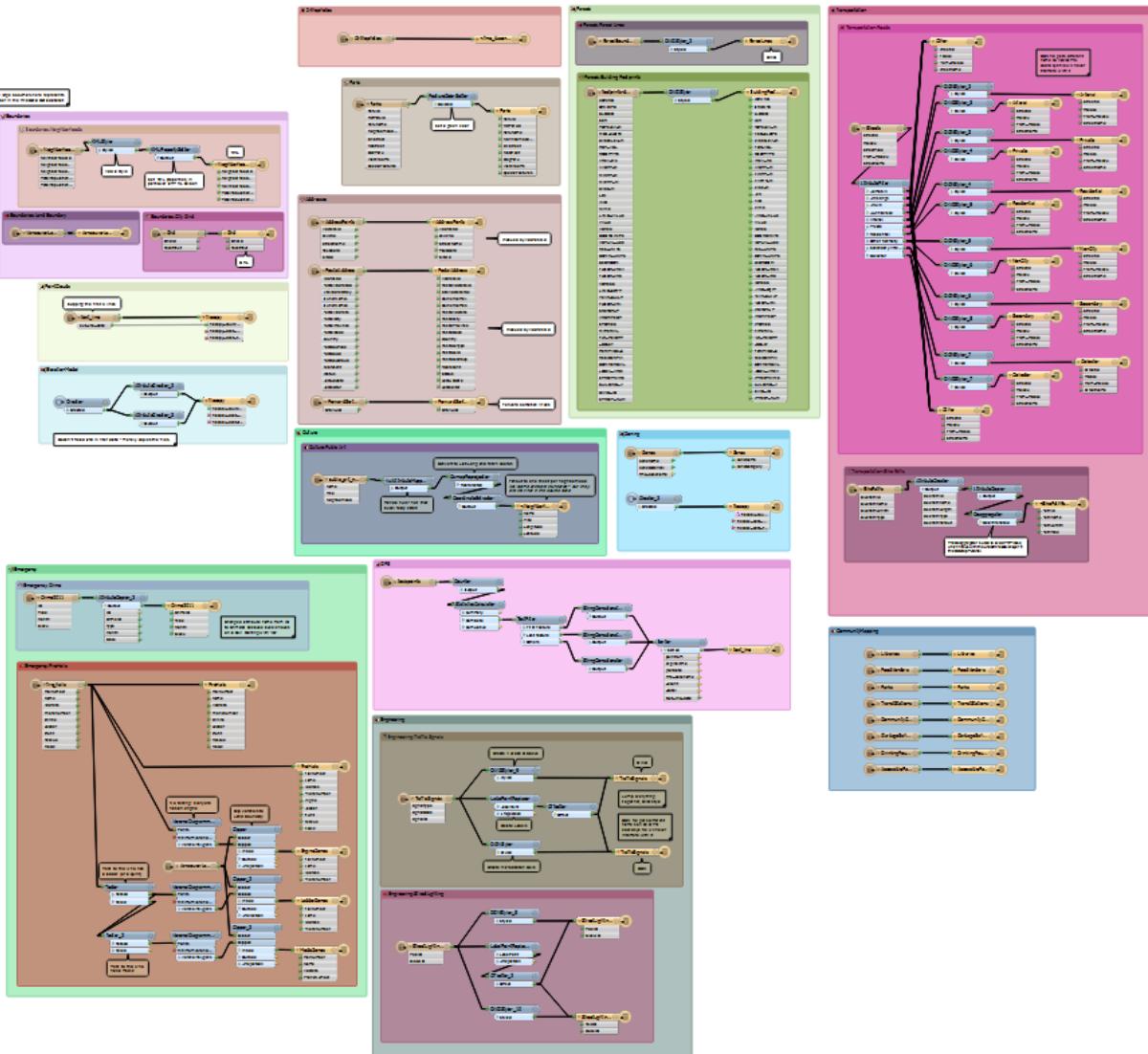
- **Delete:** Delete the bookmark
- **Duplicate:** Create a duplicate bookmark
- **Rename:** Edit the bookmark title.
- **Edit Bookmark Description:** Edit the description of the bookmark.
- **Select All Objects in Bookmark:** All objects (transformers included) in the bookmark are selected.
- **Properties:** Open a dialog to set bookmark name and color.

A bookmark description shows as a block of text inside the bookmark. Like annotation, it can be styled to a specific font or color.



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. Think of bookmarks as being like the chapter headings in a book!



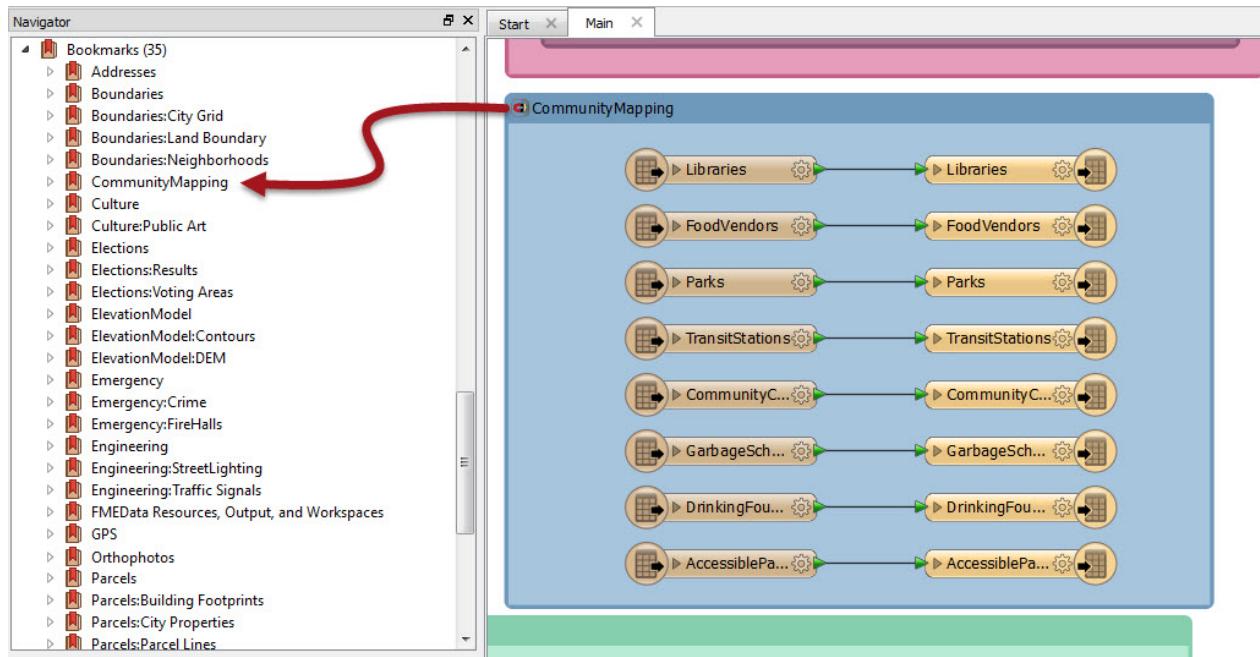
This workspace illustrates nicely how to mark up different sections of a workspace using Bookmarks.



If you use bookmarks for nothing else, use them to divide your workspace into easily identifiable sections. It really makes the canvas more readable.

Bookmarks for Quick Access

If a workspace is sectioned with bookmarks, they can be used to navigate to a particular section.

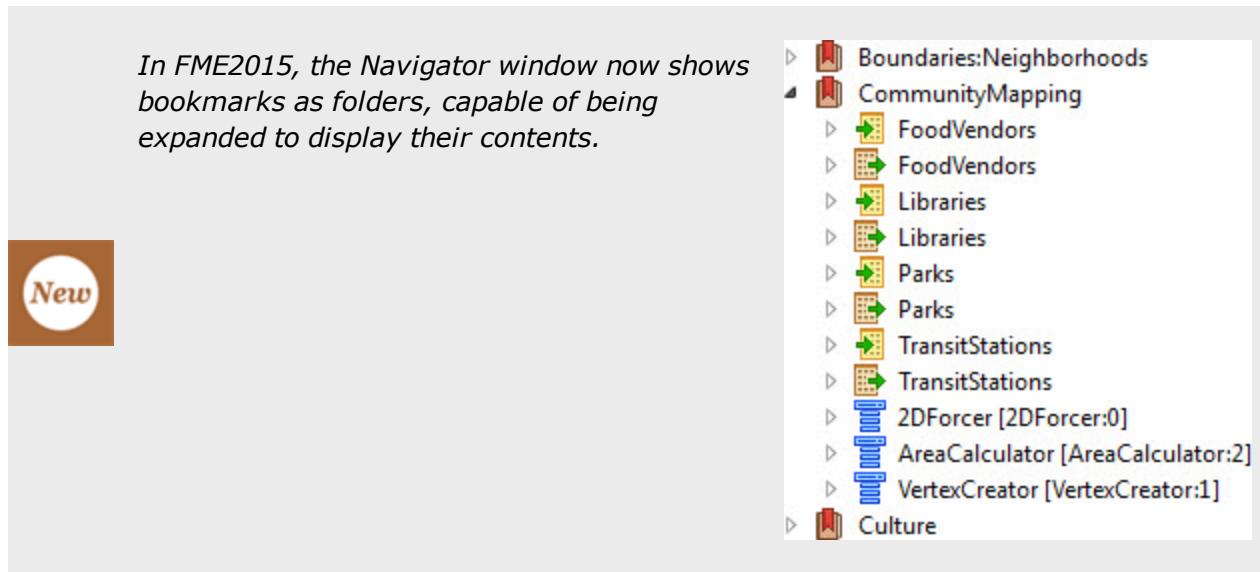


Bookmarks are listed on the Workbench Navigator.

Clicking a bookmark here selects that bookmark and brings it into view.

Double-clicking it selects the bookmark and brings it into the centre of the view.

Right-clicking it shows the same options as right-clicking a bookmark on the canvas.



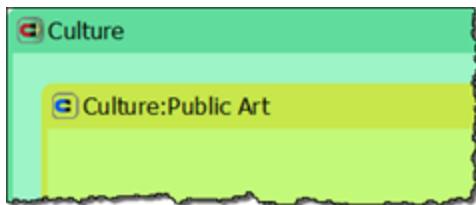
Bookmarks for Organization

A bookmark 'magnet' is a toggle that has a status represented by the icon shown in its top-left corner.

The default status of a bookmark magnet is active. Clicking the icon lets a user de-activate or re-activate the magnet.

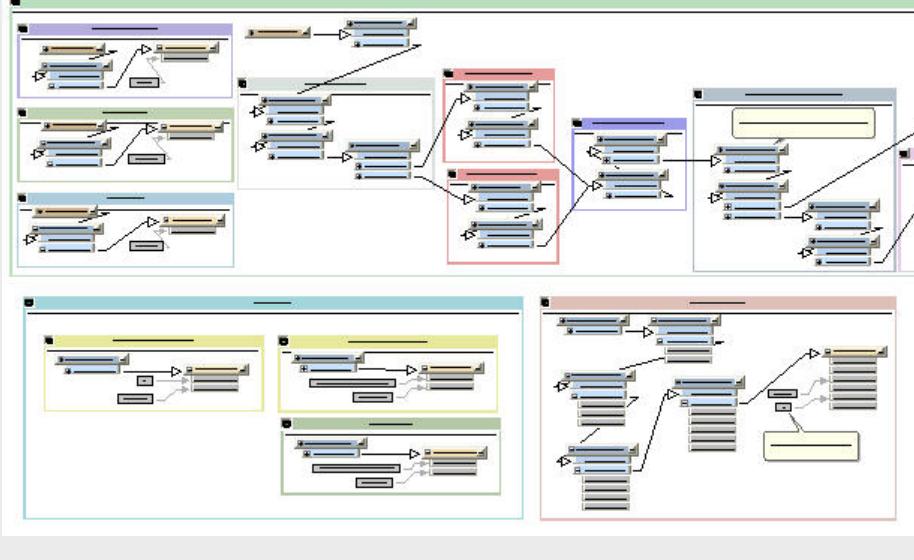
This feature aids in organizing a workspace because an active magnet (red) causes objects on the canvas to move as the bookmark is moved. Therefore large groups of transformers can be moved about the workspace canvas to create a clearer layout.

An inactive magnet (blue) allows the bookmark to be moved without disturbing the contents, which is also useful in some situations.



 *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.'



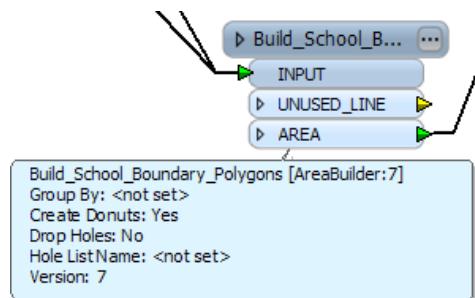


General Style Suggestions

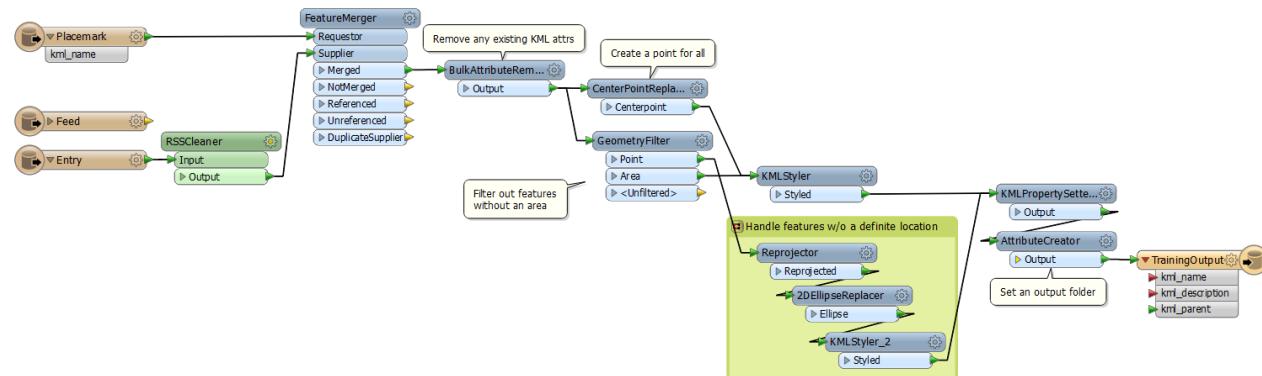
These items are general suggestions for what the style of an FME workspace should look like.

Transformer Layout

The layout of transformers – and, in particular, a consistent method of positioning – can really make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

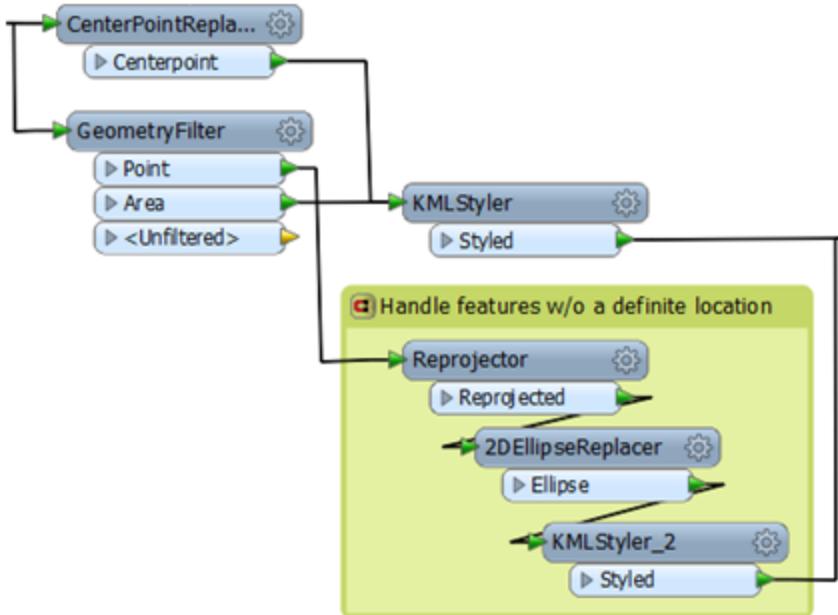


Try to line up objects where possible and use the minimum of backwards-pointing connections (those that go from right to left):

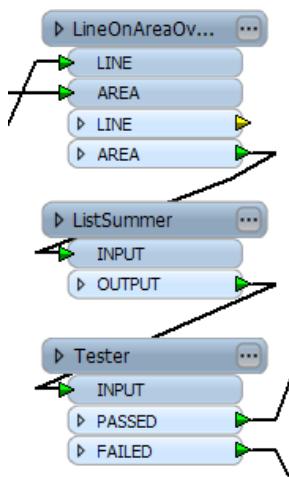


Using straight connection lines between objects – rather than crooked – is one way to enhance the look of any workspace.

Some users prefer to add extra vertices to connections to square them off.



Try to avoid adding extra vertices to a connection line, because their behavior is unpredictable when you move one of the attached transformers.



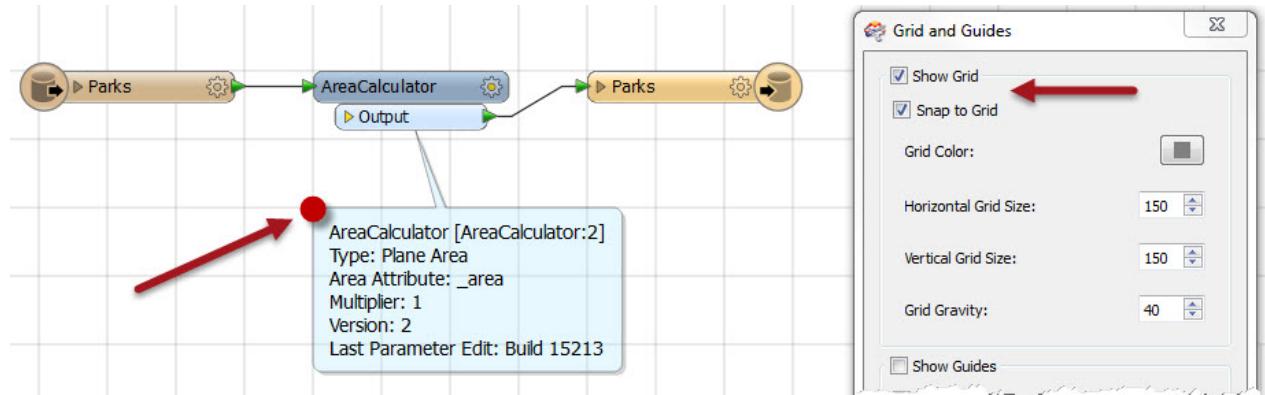
Objects in a stack - particularly feature types - are easier on the eye when vertically aligned.



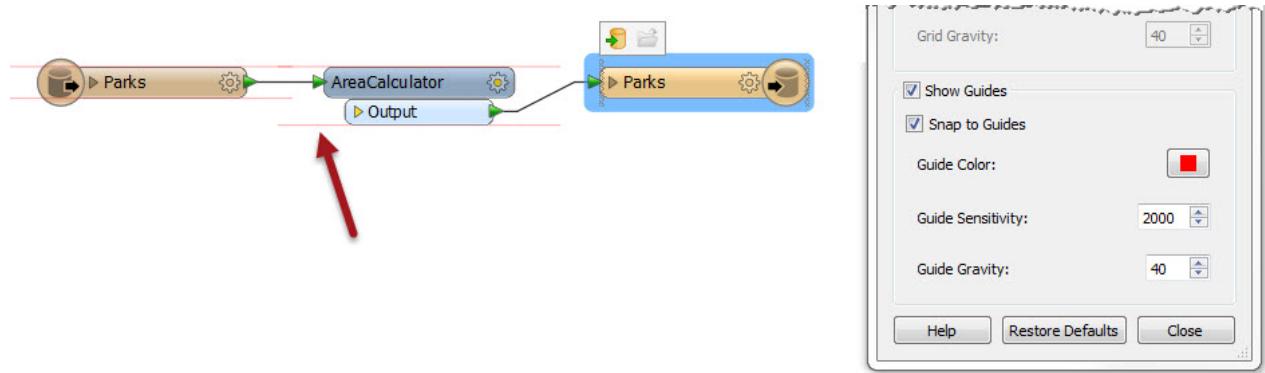
Grid and Guides

FME2015 introduces new functionality to help align workspace objects in a neat and tidy way. This functionality is accessed through View > Grid and Guides on the Workbench menubar.

The Grid and Guides dialog has options for turning on these functions, and for adjusting their exact parameters.



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. In this way objects can be more easily lined up.

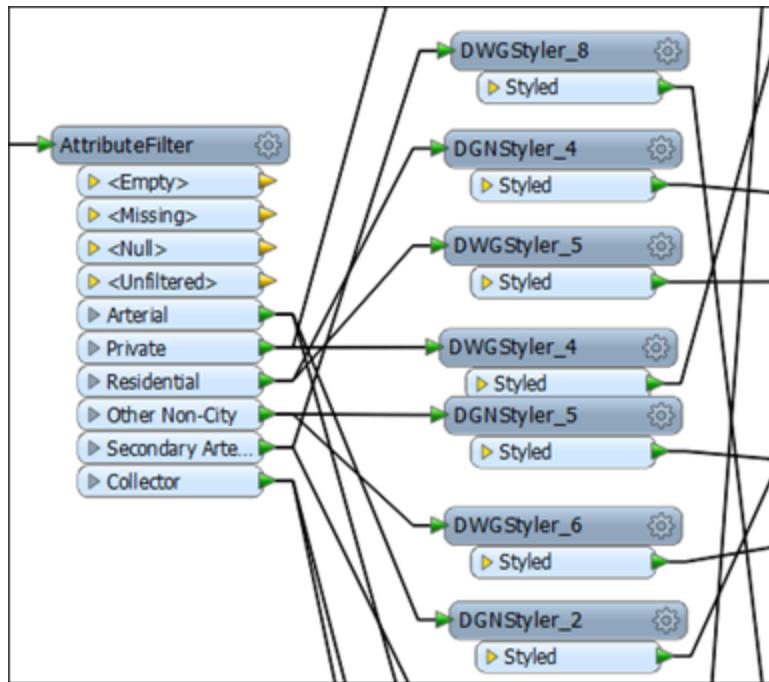


Show Guides causes guidelines to be displayed on the Workbench canvas whenever an object is moved, and lines up approximately to another canvas object. Snap to Guides allows an object to be snapped onto a highlighted guideline.

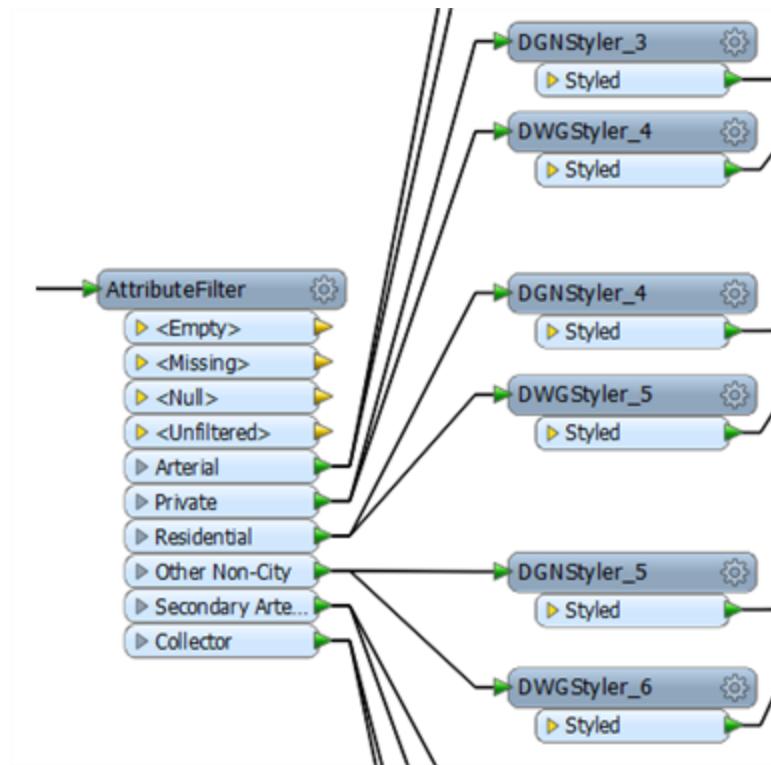
These two tools make it very simple for workspace objects to be aligned in a pleasing style.

Non-Overlapping Connections

Does it really need to be said that overlapping lines are not good for workspace clarity?

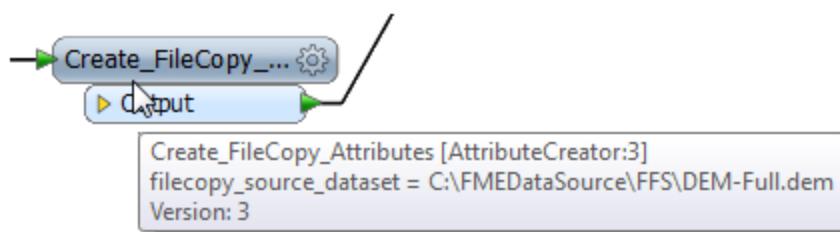
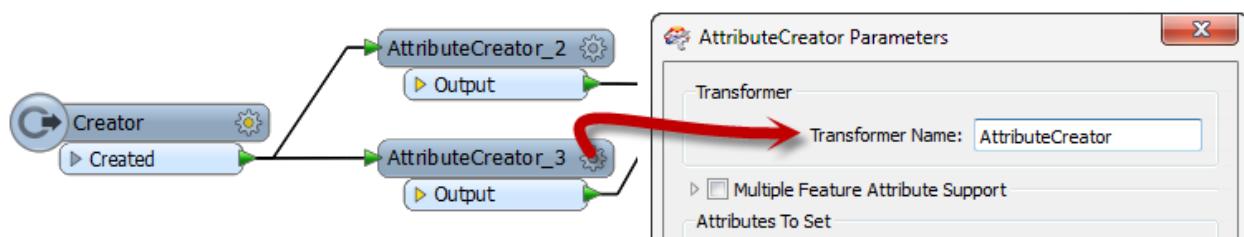


Take the effort to tidy your workspace. It will be so much clearer with just a little reorganization.



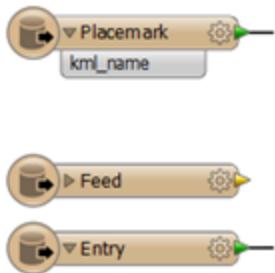
Renamed Transformers

Remember that the Properties dialog for a transformer contains a field where the transformer may be renamed to a more meaningful title.





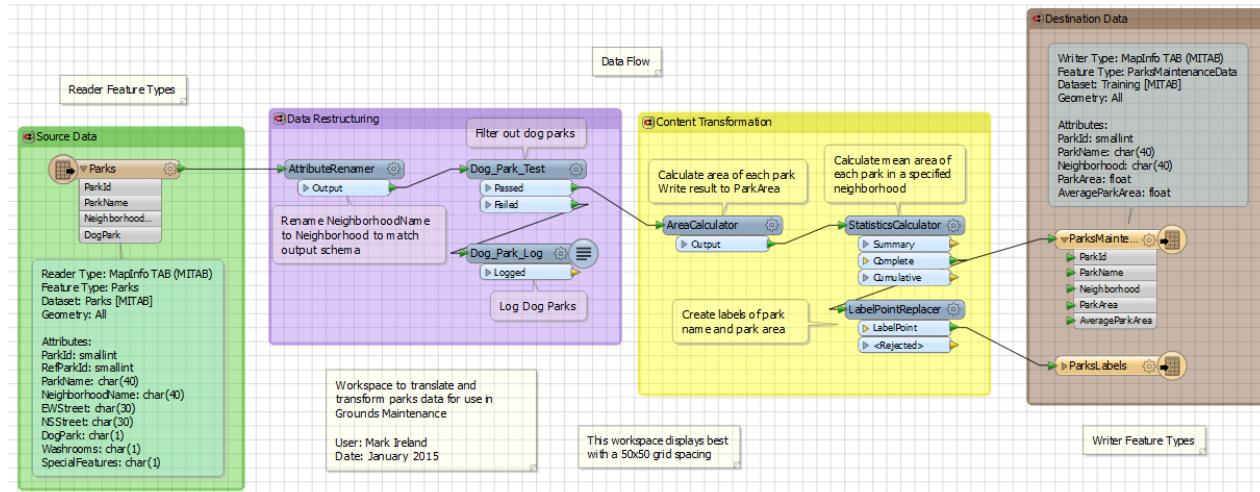
Rename transformers to help identify them both on the canvas and in the navigator window.



Exercise 3a Applying the Style Guide

Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	Clean-up workspace and apply concepts of style guide
Demonstrates	Style Best Practice
Starting Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise3a-Begin.fmw</i>
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise3a-Complete.fmw</i>

Look at the workspace you created for the exercises in Chapter 2. Does it look like this? It should... once you've completed this exercise!



1) Open the Workspace

Open the finished workspace for the exercises in Chapter 2.

Alternatively open C:\FMEData2015\Workspaces\DesktopBasic\Exercise3aBegin.fmw

2) Tidy the Workspace

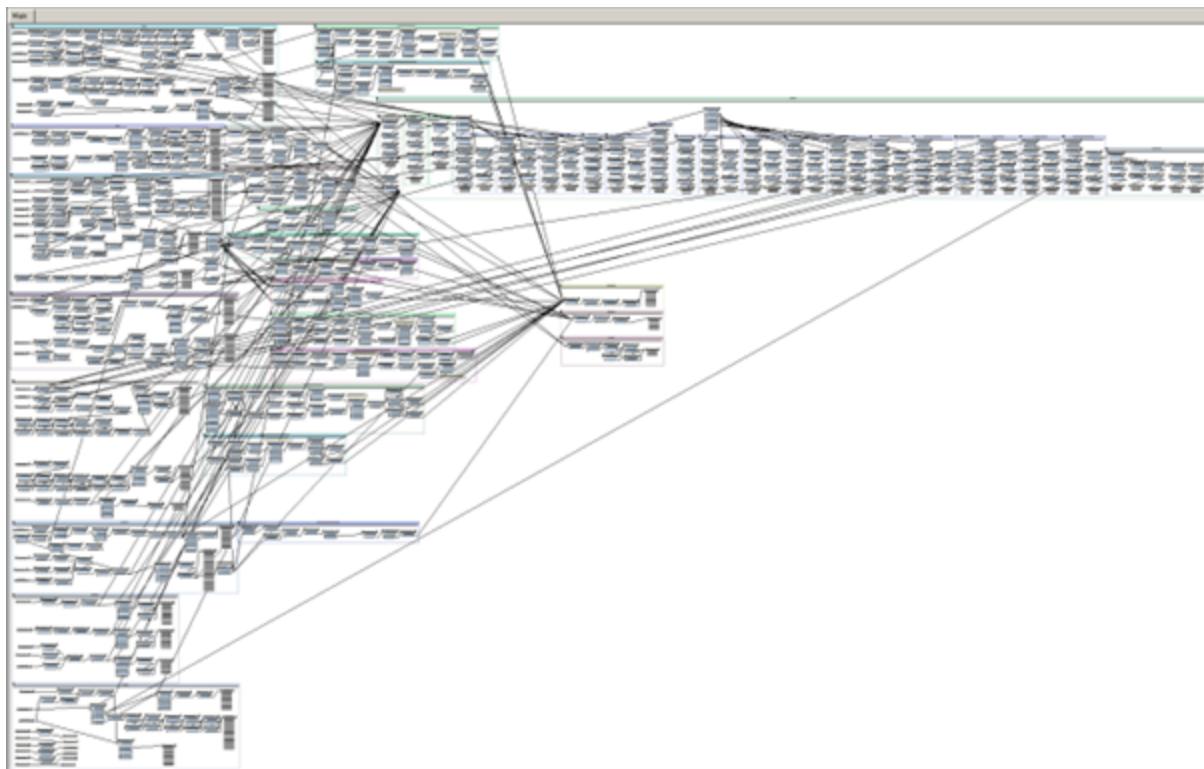
Tidy the workspace layout and setup using the FME Style Guide covered in the previous pages.

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.

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, debugging can be as much an art as a science.

Logging

The log window (and log file) contains 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.

Message Types

There are a number of different message types that show in the log window including:

- An error in the log window, denoted by the term **ERROR**, indicates that a problem has caused FME to terminate processing.

Example: An inability to write the output dataset because of incorrect user permissions.

- A warning in the log window, 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.

Example: An inability to write features with a geometry that's incompatible with the Writer format. FME can filter out these features, but will warn the user of their presence.

- 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.

Example: The number of features processed by a transformer or confirmation of a particular dataset parameter.

- Statistics messages, denoted by the term **STATS**, provide information on the number of features read from the source and written to the destination datasets.

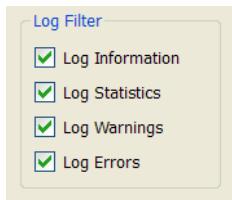
Example: The number of features processed by the workspace as a whole and the time taken to do so.



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**.



Adjusting the view of messages in this way does not affect the output written to the log file.

 *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!*

- Log File Defaults**
- Save log to file
 - Append to log file
 - Log timestamp information
 - Log debugging information

Log Timings

Another useful option under **Tools > FME Options > Translation** 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.



Keep log timings turned on. They do add to the amount of content in the window, but can be very useful when trying to debug a poorly-performing translation.

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.



*If a translation fails, look in the log window **first**.*

Rejected Features

When a transformer rejects a feature in some way – perhaps it has the wrong type of geometry, for example – it usually writes it to a spatial log file.



Identify bad features using the spatial log (.ffs) dataset. It will probably be easier than trying to locate the feature within the full source dataset.

Here are some tips on interpreting the log file:

Check for Warnings

The first thing to do is check for the following comment:

Translation was SUCCESSFUL with X warning(s)

And then (if $X > 0$), use the search option to look for the word WARN

Sequence

Be aware that — because FME processes data feature-by-feature instead of transformer-by-transformer — the log file does not show each transformer in sequence. Don't be fooled into thinking the order of the log file will absolutely match the layout of the workspace.

Output

If the workspace features have been redirected to an output other than a Writer, then this is reported at the very bottom of the log. Watch for such messages. It's easy to miss this and assume that there is a problem with the FME writer.

Timestamps

Literally, FME processing time is the amount of time that FME was actively processing. This is not necessarily the same as the length of time taken to run the workspace!

The absolute start and end times often 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.

Unexpected Input

Unexpected Input is data that is read from a dataset but not used in the workspace. This is reported through a message at the bottom of the log. However, don't automatically assume that there is a definite problem. FME will report this as a warning even if a user deliberately didn't want this data processed.

Filtering Rejected Input

Occasionally a transformer or writer will reject certain features because they do not match the functionality or parameters of that object.

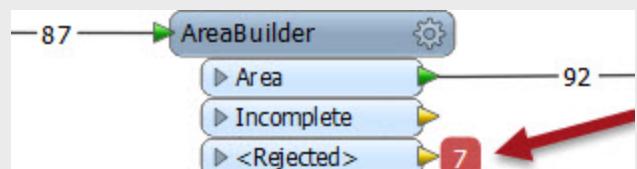
For example, if a mixture of point and line features is sent into the *AreaBuilder* transformer, then the log will report that the point features were discarded.

It will give a tidier log if such features are removed before they get to the point of being a problem. For example, a *GeometryFilter* would be enough to filter out points before they even get to the *AreaBuilder*.

New

In FME2015 or newer, 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.



Logger Transformer

A *Logger* transformer takes features from the workflow and writes information about them to the log window or log file. Limits can be placed on the number of features to be logged in full.

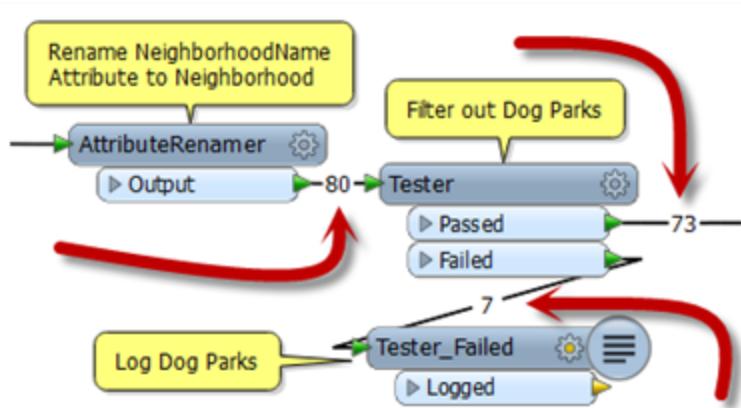
It is extremely useful for writing out information to the log for debugging purposes.



The *Logger* transformer has a setting called 'Log Message:' – when a feature is logged, this message appears with it. Setting a unique message helps you locate logged features by using the log search function to find the message. FME also adds the transformer name to the message, helping you to identify between multiple *Logger* transformers.

Feature Counts

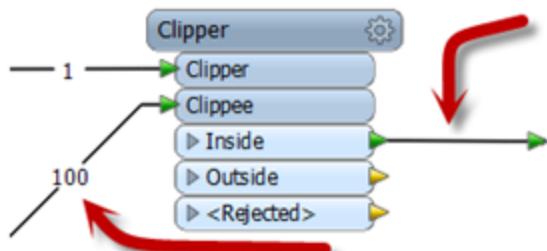
A workspace Feature Count refers to the numbers shown on each connection once a translation is complete:



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.

For example, if all your features entered a transformer, but none emerged, then you can be fairly confident that the transformer is the cause of the problem. You might check that the correct transformer output port is connected, or add a Logger or Inspector transformer before it so that you can examine what data is going in and determine why it is being lost.

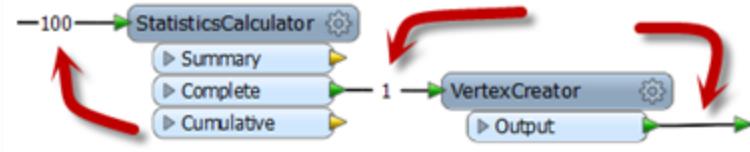
Here, for example, 100 features are entering the Clipper transformer (to be clipped against a single boundary) but none emerge.



Maybe this is because the wrong output port is connected and the data falls outside the clip boundary and not inside. If this itself would be a problem, then inspect the data as it enters the transformer. In this example it's likely that Clipper and Clippees don't occupy the same coordinate system, hence one does not fall inside the other.

Single Feature Counts

When a workspace fails (usually with an **ERROR**) and only a single feature has emerged from a transformer, it is likely to be the following transformer that is the cause of the problem:



Here, for example, 100 features enter the StatisticsCalculator and only 1 emerges. However, the StatisticsCalculator is not the source of the error. The error is triggered when the first feature exits the StatisticsCalculator and reaches the VertexCreator, suggesting the VertexCreator is the problem here. It failed before the other 99 features had the chance to even reach it.



Use Feature counts to identify where a problem occurred.



You should also be clear that these numbers don't work as well for identifying warnings. Most warnings still allow the feature in question to pass, unprocessed, so the counts do not reflect warnings as well as they reflect errors that stop the translation in its failed state.

However, all ports labelled <Rejected> will have a count of features, whether or not they are connected to further transformers.

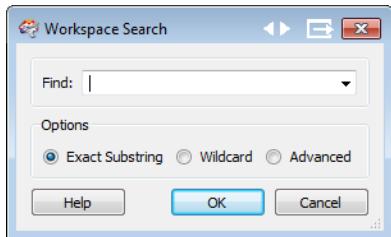
Workspace Searching

The workspace search function is accessed through a hyperlink at the foot of the Navigator window:

- ▷ Transformers
- ▷ Bookmarks
- ▷ User Parameters
- ▷ Workspace Resources
- ▷ Workspace Parameters
- ▷ Workspace Properties
- ▷ [Workspace Search](#) ←

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.

Workspace search is important for debugging because sometimes the transformer that fails does so because of a fault in a completely different part of the workspace. When that happens, the ability to search a workspace for terms found in the error message helps to track down the source of the problem.



Use a Workspace Search to locate items identified as problems in the log. This is especially useful when the workspace is very large.

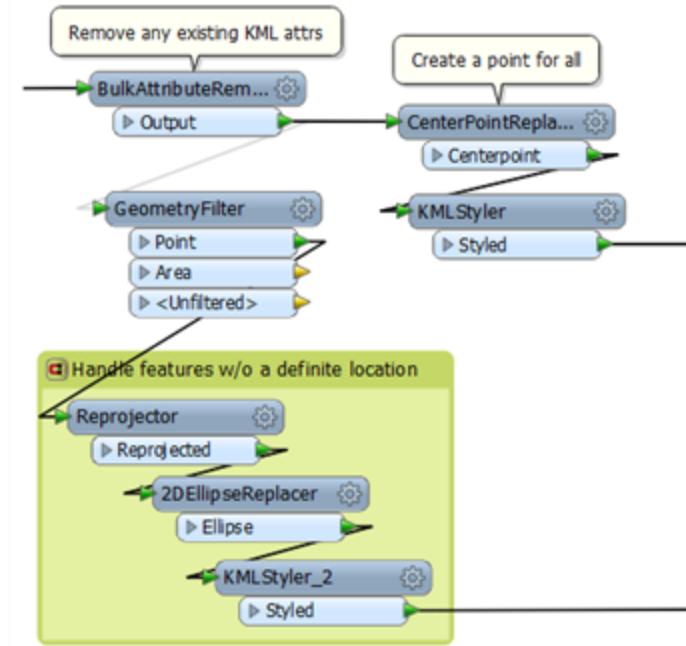
Testing Isolated Sections

Testing and debugging a large workspace is easier when it's possible to isolate sections and test them separately. Of course, it will help this technique if the workspace is already properly divided using bookmarks!

The ability to isolate specific sections is done by disabling connections. This renders a connection inoperative in much the same way as if it had been deleted, and no features will pass through.

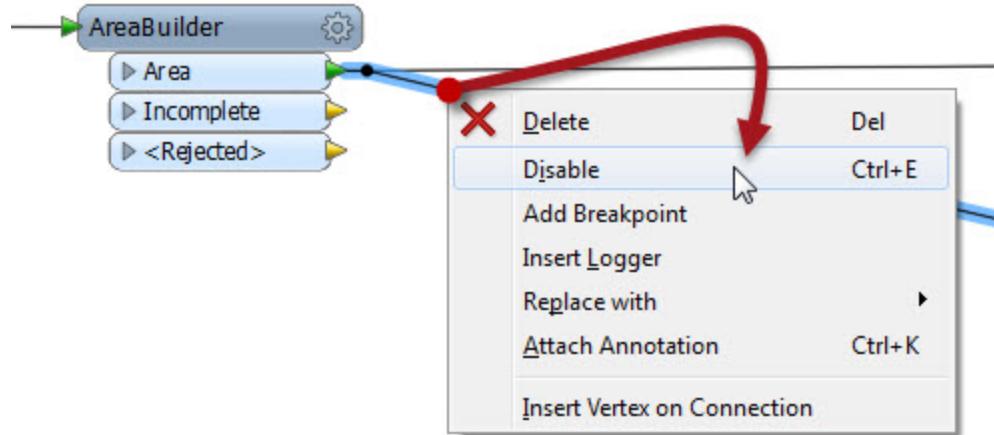
For large workspaces the disable functionality is very useful. It enables the author to isolate sections of a workspace from being executed, so that tests run much more efficiently.

Here the workspace has two outputs from the BulkAttributeRemover , creating two separate parts to the workspace. The user wishes to run the workspace only on one section, so disables the second connection.



Now no data will be sent to the GeometryFilter for processing in that section.

Connections can be disabled by clicking on them and using the shortcut **Ctrl+E**. This keyboard shortcut is a toggle that alternates between enabled and disabled. Alternatively, you can right-click on a connection and use the disable option.



As well as connections, you can disable any object on the canvas; for example a



transformer or a feature type.

Exercise 3b Debugging

Scenario	FME user; City of Interopolis, Planning Department
Data	GPS Road data (JSON)
Overall Goal	Debug workspace
Demonstrates	Debugging Best Practice
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise3b-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise3b-Complete.fmw



On his application for FME Certified Professional status, Shirley U Jest 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 this jester is seriously in error, and has produced a very poor workspace.

1) Start Workbench

Start FME Workbench and open the starting workspace.

Run the workspace and inspect any warning or error messages that occur.

Track down where any warning or error occurs, trace it to its cause, and fix it.

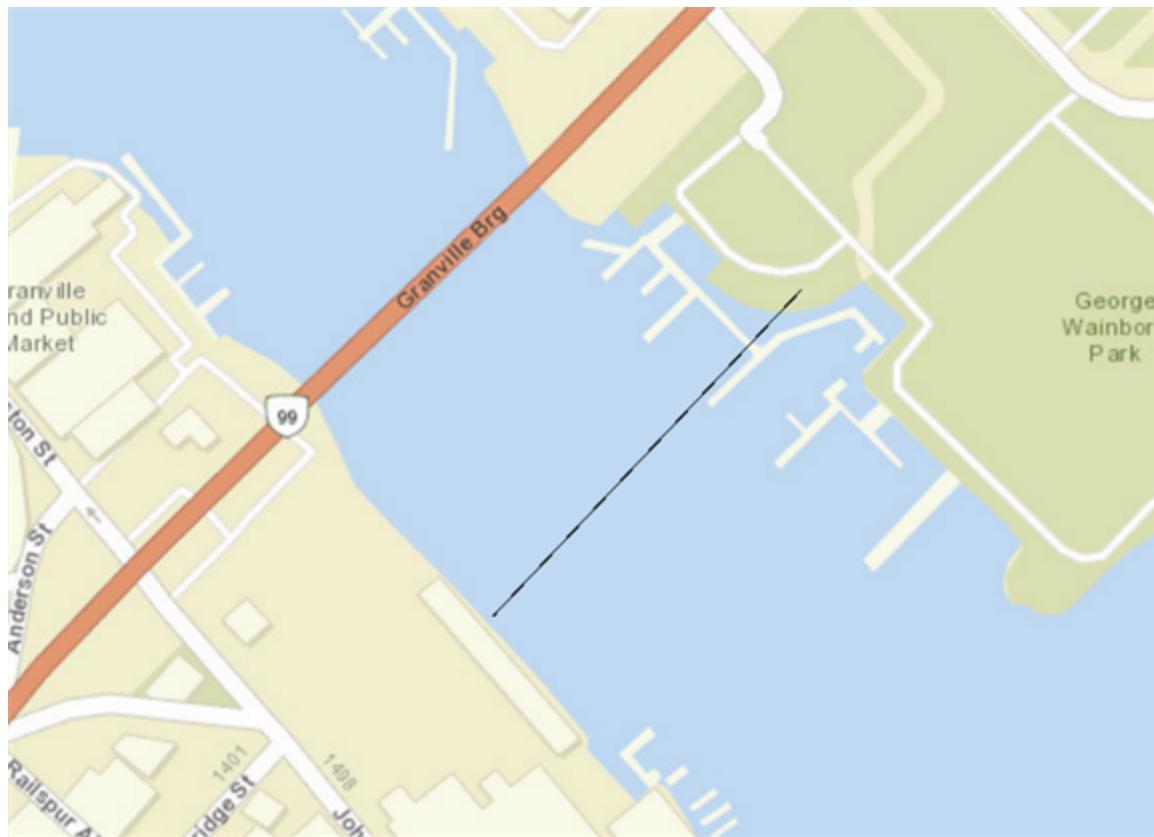
2) Fix Translation

Track down all the other problems in the workspace (I count three or four others) and be sure to fix them.

Remember to make use of:

- The FME Data Inspector
- The Logger and Inspector transformers
- Feature Counts
- Log window WARNings and ERRORs.

Your output should look like this in the FME Data Inspector:



ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

Oh! Looks like I need to get my GPS fixed!

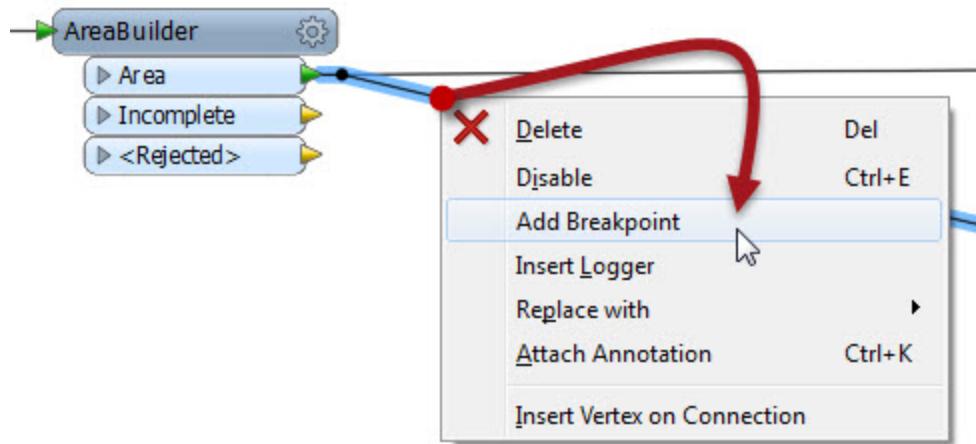
Feature Debugging

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.

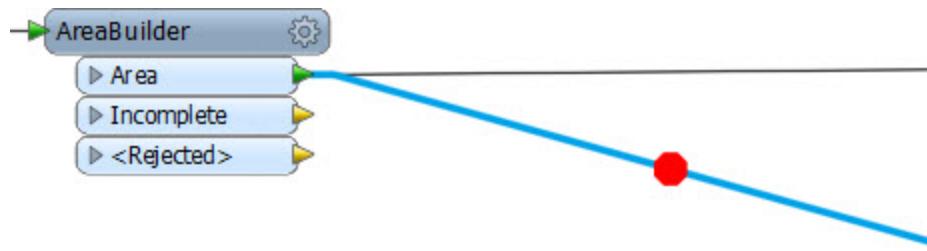
Feature Debugging is triggered by "breakpoints"; workspace connections that are flagged by the user as a location where features should be inspected.

Here a user wishes to inspect data after processing by the *AreaBuilder* transformer.

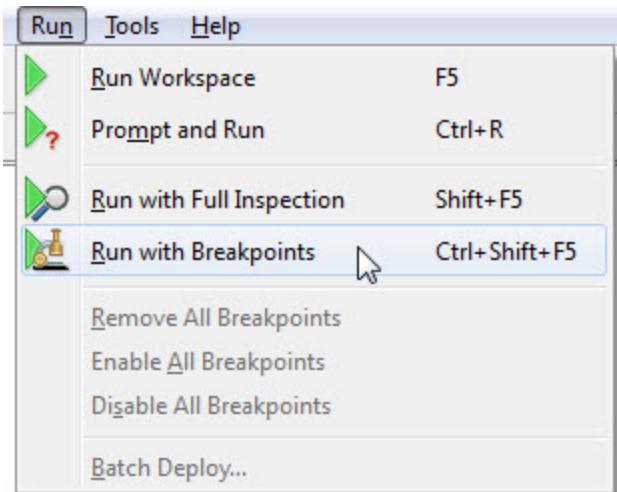
A right-click on the connection and selection of **Add Breakpoint** is used to set it up.



The connection is highlighted blue with a red "stop" sign, to denote its new status.

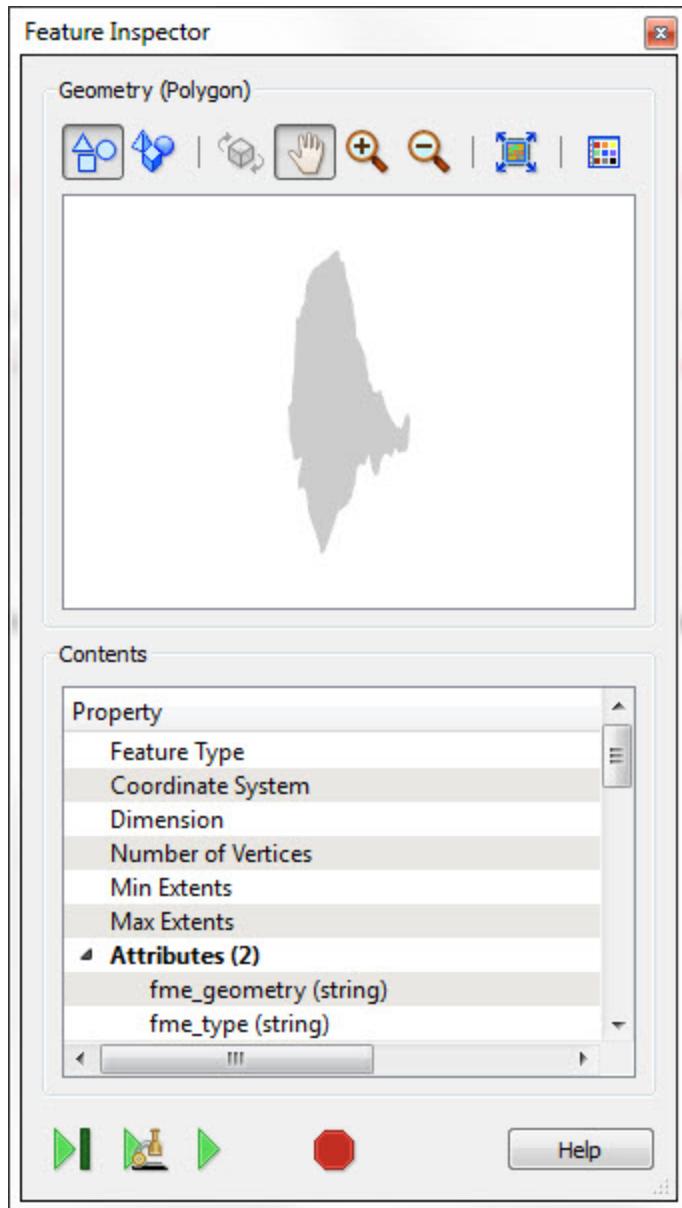


Now the workspace is run using "Run with Breakpoints."



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.



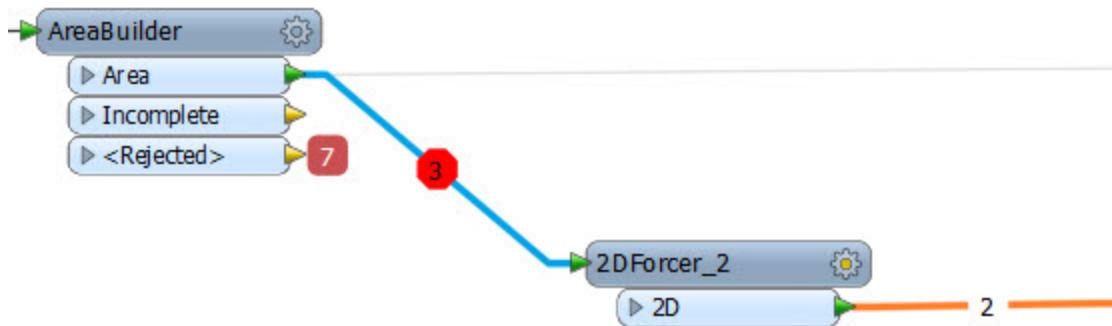
There are four buttons at the foot of the Feature Inspector window:

	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.

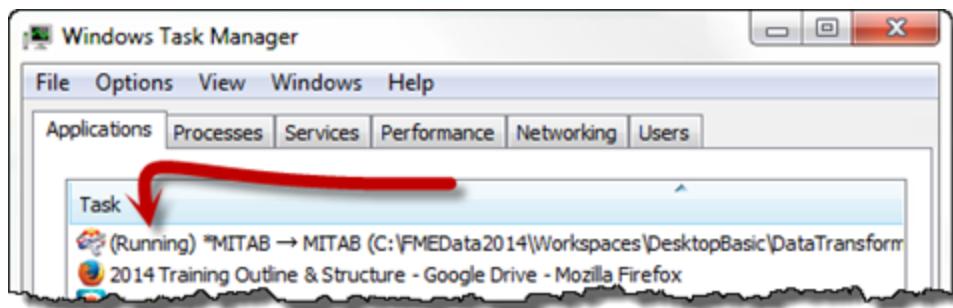
	Continue Translation	This tool re-starts the translation, ignoring all further breakpoints.
	Stop Translation	This tool stops the translation.

The currently active connection is highlighted orange 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.



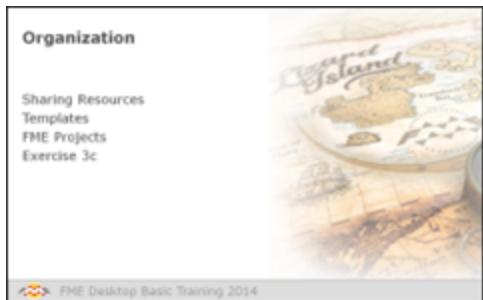
And while feature inspection is going on, the status of the FME process (in the Windows Task Manager for example) shows that the translation is still actually ongoing.



Use Feature Debugging when a transformation is going wrong and you can't tell where, 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.

Now you've learned about Feature Debugging, why not try the previous example again, this time using these techniques to show what happens step, by step?

Organization



A goal without a plan, is just a wish!

A key part to any FME project is to keep organized and combine your efforts with other colleagues by sharing data and resources. Luckily FME has a number of tools that can help you achieve this.

Sharing Resources

Re-using components, and sharing them with fellow FME users, is vital in creating consistent designs among a large group of employees. It also improves efficiency – because a translation author will not have to create every workspace from the beginning – and reliability, because any change to a shared resource is passed on through any workspaces that make use of it.

Resources that may be shared include workspaces, custom transformers, custom formats, custom coordinate systems, and templates.

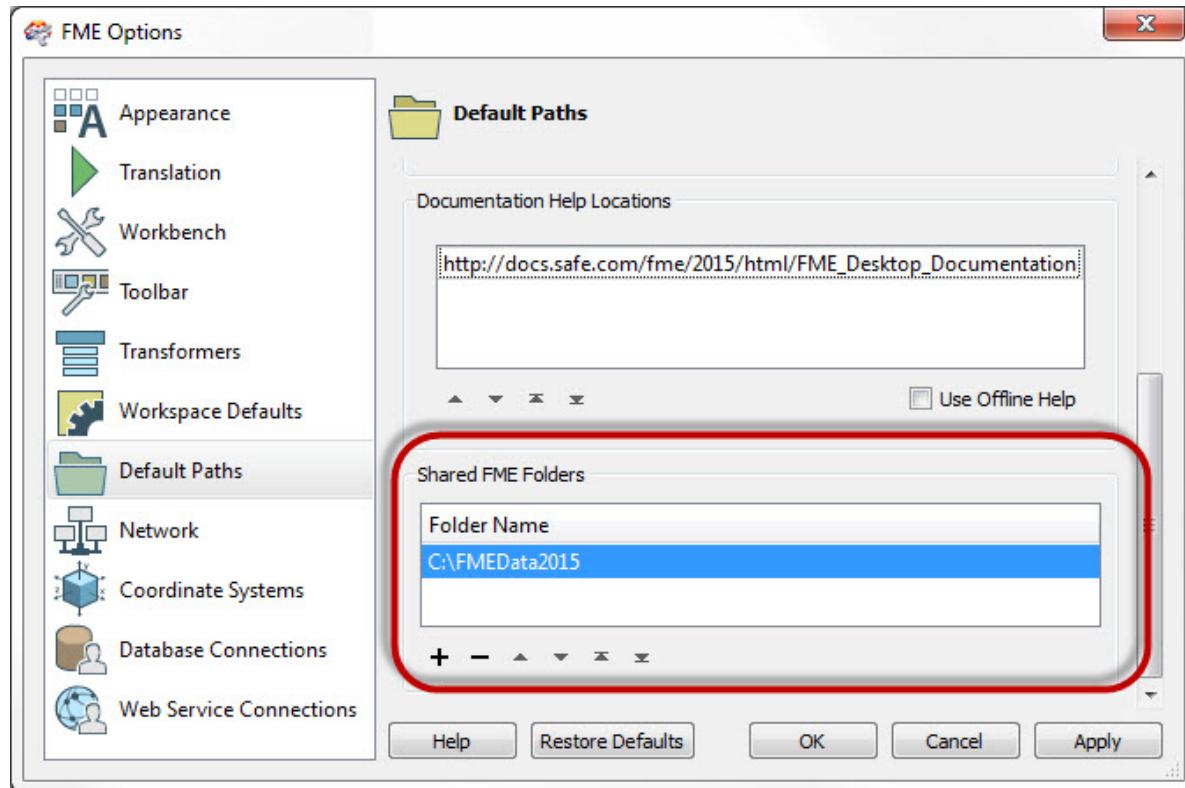
Shared Resource Directories

The basic method of sharing is through a Shared Resource Directory. FME is able to identify resources stored in these directories, and use them directly within a translation.

A shared resource directory can be used by just one person, or many.

Using a shared directory 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.



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.



<user>/<documents>/<FME> is a shared resource directory created and used by default, without having to define it within the options dialog.

Templates

FME Templates are a particularly useful for of shared resource. Like similarly-named items in other software, FME templates are a means of creating a workspace with/from a predesigned 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:

- 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.

- 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.

- 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.

- An Example Translation

A template is also a great way to share demos and examples in a way that another user can run immediately.

FME Projects

FME is often used not just for one-off translations, but for a particular project or series of projects.

Project best practice involves suggestions more than hard and fast rules; but suggestions that are highly recommended!

Directory Structure

Project use of FME is more efficient when a proper project structure is maintained.

For example, assuming a special disk for storing projects (*P:*), a simple project structure could be:

myproject\ Use a project ID if you have one; for example:
P:\P2015-999_PROPERTY_UPDATES

Follow this with subdirectories, such as:

.\FMEworkspaces	which could itself have subdirectories such as
.\includes	for include files
.\Scripts	for Python, SQL, or Tcl scripts
.\SourceData	for storing source dataset files
.\OutputData	for storing destination dataset files
.\NoteDocs&Stuff	project estimates, specifications, proposals, scope of work, and so forth
.\testing	a folder for test data and workspaces



For a large scale FME project, set up a project structure and stick to it. Keep copies of all FME workspaces, maybe in a revision control system like Subversion.

File Naming

Based on past experience the Safe Professional Services team has the following recommendations for file and folder naming within an FME project:

- Use revision numbers on workspaces or directories; for example, myworkspace_rev17.fmwt.

It's tempting to save to the same workspace all the time, but then there is no fallback position from a mistake or corrupt workspace. It only takes one such mistake to regret it.

- When using dates for file names or directories, use international dates such as 2015-03-21 or 20150321 so files are listed in the correct order. Do not use a date such as Mar-21-15, which will be listed alphabetically after April in Windows Explorer.
- Reduce clutter in the project's main directory by saving files inside subdirectories.
- Avoid meaningless directory and files names, such as 'fr_dnt'. From Client? Father Clint?!
- Share ideas with others. Make sure all users conform to the same conventions.

Relative Paths

Although it isn't possible to browse for a relative path, it is possible to manually edit a path to be relative; for example, change `P:\P2015-999\SourceData\myfile.dgn` to `.\SourceData\myfile.dgn`.

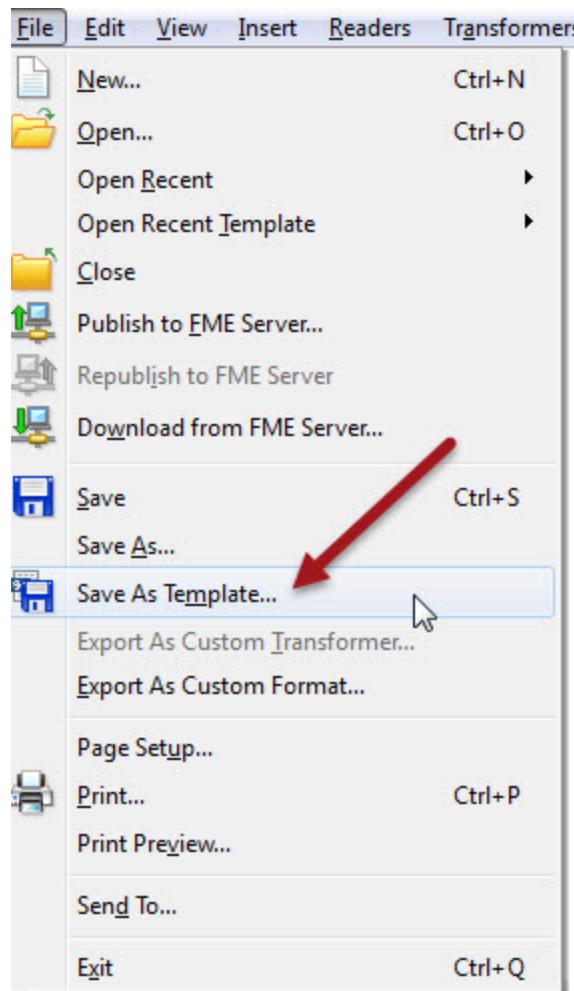
The advantage of doing this is that if the original project is renamed or even moved to a new location, all of the workspaces will still function correctly without further editing.

Exercise 3c Using a Template	
Scenario	FME user; City of Interopolis, Planning Department
Overall Goal	Download and use a template
Demonstrates	Organization Best Practice
Starting Workspace	None
Finished Workspace	<code>C:\FMEData2015\Workspaces\DesktopBasic\Exercise3c-Complete.fmwt</code>

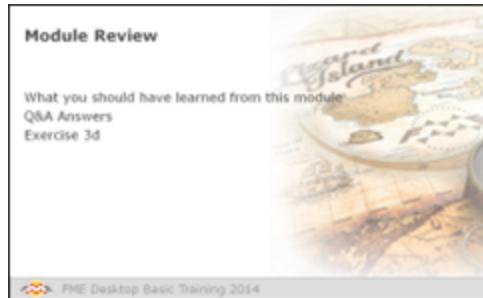
As a little practice, open the Create Workspace dialog and experiment with opening and using a template. It can either be an installed template or one from the FME Store.



Then take a previous workspace and convert it into a template.



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 **Methodology, Style, Performance, Debugging** and **Organization**.

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 interpret an FME log file at a basic level and to use debugging techniques to locate problems in a translation.
- The ability to use FME as part of a larger project, in an organized and efficient manner.

Best Practice

Exercise 3d Best Practice	
Scenario	FME User Conference Attendee
Data	Roads (DWG, Shape, PostGIS)
Overall Goal	Find route from user conference to sports bar
Demonstrates	Data translation, data transformation, best practice
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise3d-Complete.fmw

In this exercise you are an attendee at an FME World Tour event. After a full day of learning about FME at the Vancouver Convention Centre you and some colleagues want to go and watch a soccer game on TV somewhere. The Safe Software staff tell you the best experience would be at an Italian cafe on bustling Commercial Drive, but they neglect to tell you how to get there.

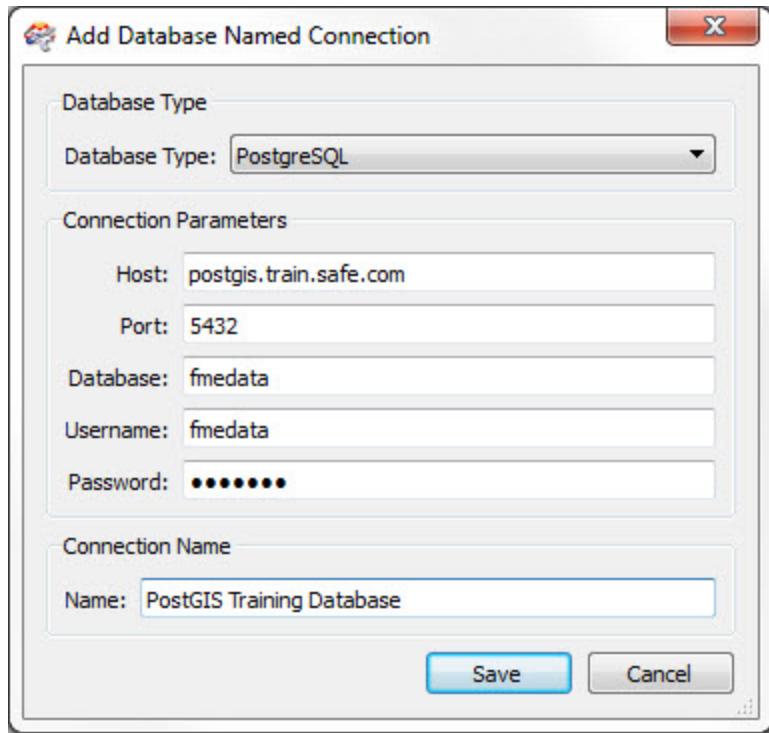
So, your task is to use the data available to you to calculate the best route from the convention centre to Commercial Drive, and to write out that data to GPX format so you can use it in your in-car navigation/GPS.

1) Create Database Connection

As we are going to use a PostGIS database, we should first 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.



Give the connection a name (such as PostGIS Training Database) and click Save. Then click OK to close the FME Options dialog.

2) Inspect Data

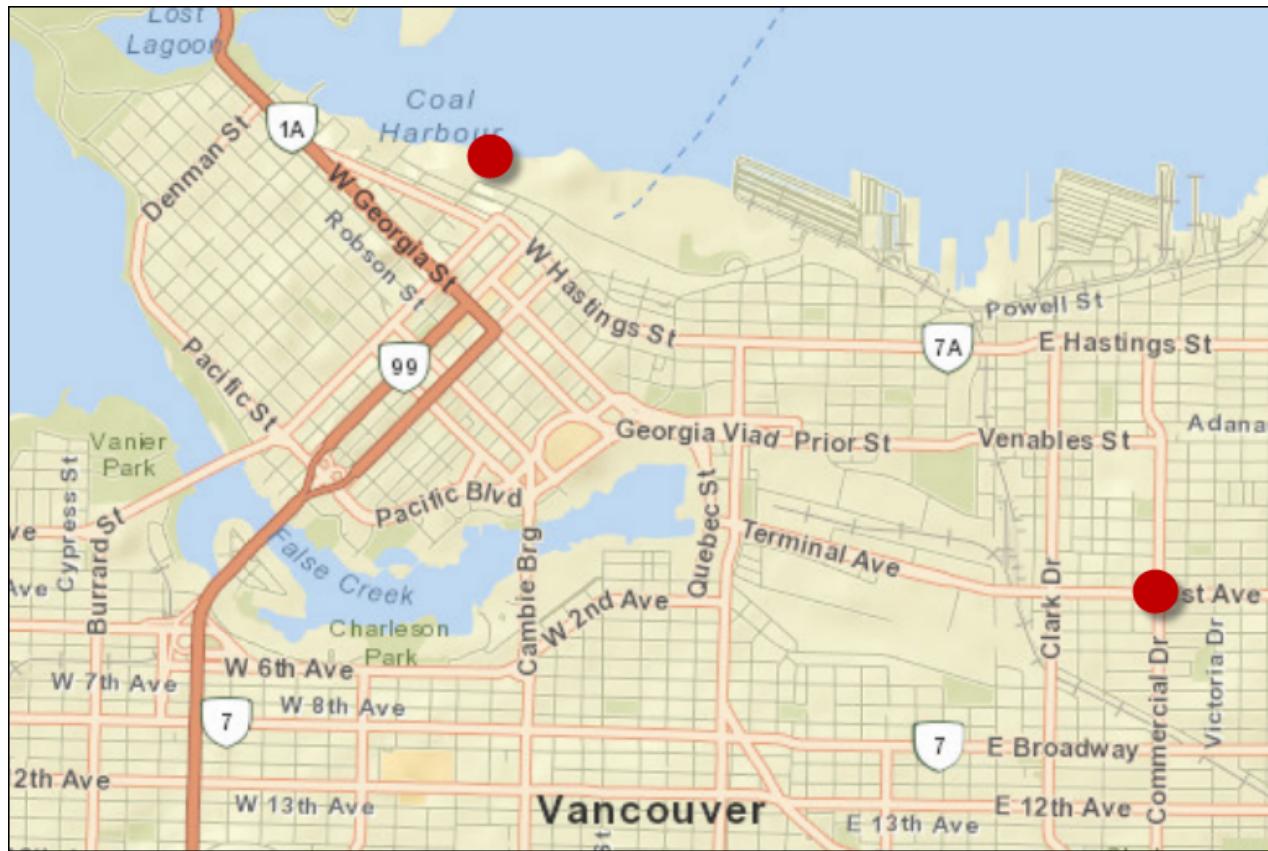
Start the FME Data Inspector to inspect the dataset we will be using.

Select **File > Open Dataset** and, when prompted, enter the following:

Reader Format	PostGIS
Reader Dataset	PostGIS Training Database
	Check the parameters and under Table List, choose the tables Roads and
Parameters	One-Way Streets.

Click OK to close the dialogs and open the data. Change the color of the one-way streets so you can see that they are a different set of features. It's important to know which roads are one-way if we want to calculate a route that is actually legal!

If you have a background map applied, try to locate both the Vancouver Convention Centre and Commercial Drive.



ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

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:\FMEData2015\Data\Transportation\OneWayStreets.dwg

3) Start Workbench

Start Workbench and use the option to Generate a workspace.

Reader Format PostGIS

Reader Dataset PostGIS Training Database

Parameters Again, under table list select Roads and OneWayStreets

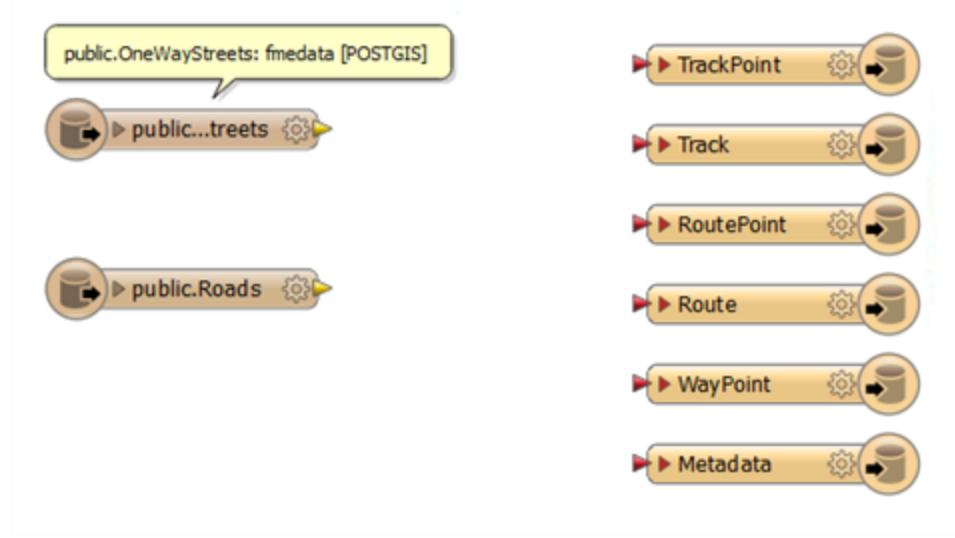
Writer Format

GPS eXchange Format (GPX)

Writer Dataset

C:\FMEData2015\Output\Training\DrivingRoute.gpx

The workspace will look like this:

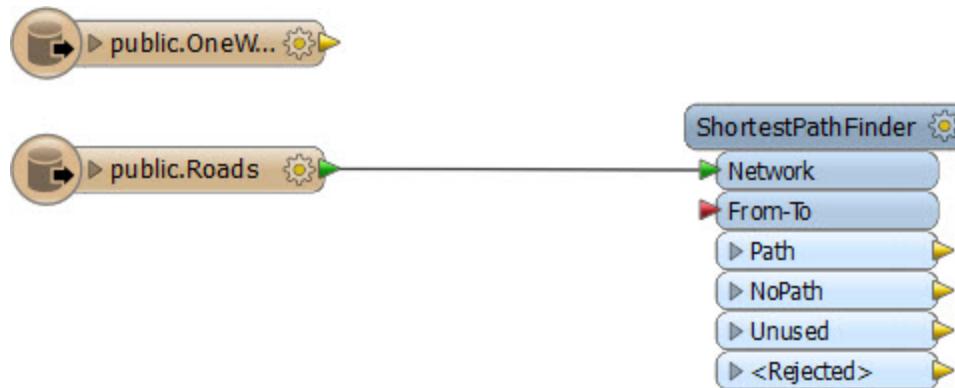


For the sake of Best Practice, you may want to put a bookmark around these features, and maybe annotate which are the one-way streets.

4) Add ShortestPathFinder

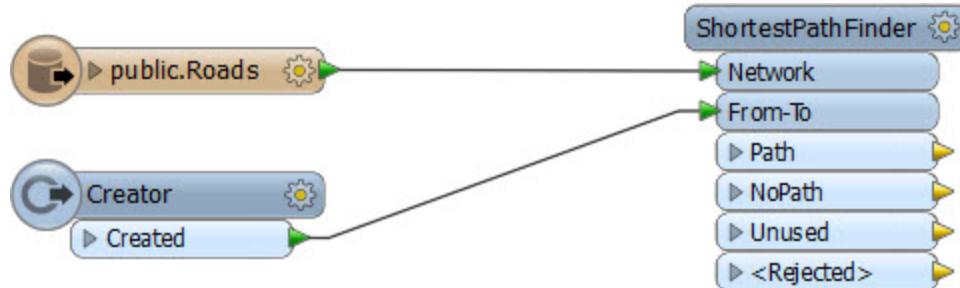
Now we need to start calculating a route. The obvious first step is to add a ShortestPathFinder transformer, which is how we can calculate our route.

So, add a ShortestPathFinder transformer. Connect Roads 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). To add a feature to input here we can manually create it with the Creator transformer. Add a Creator transformer and connect it to the From-To port.



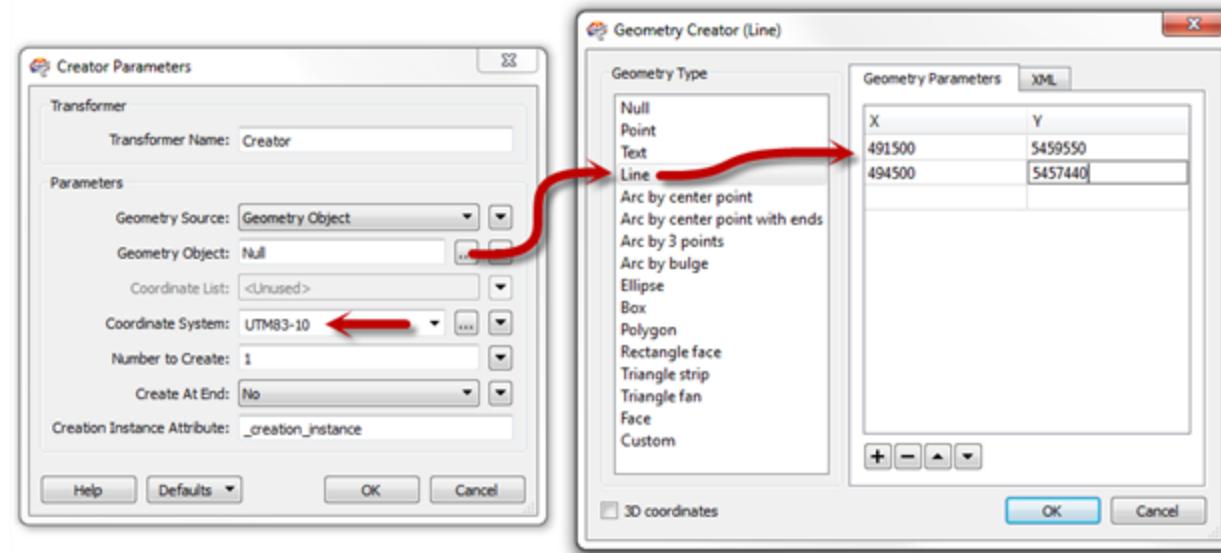
Open the Creator parameters dialog.

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 convention centre and the second is the closest point in our network to Commercial Drive.

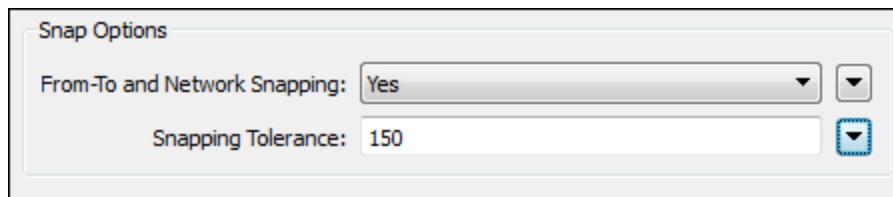


Click the OK button to close the dialog.

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.

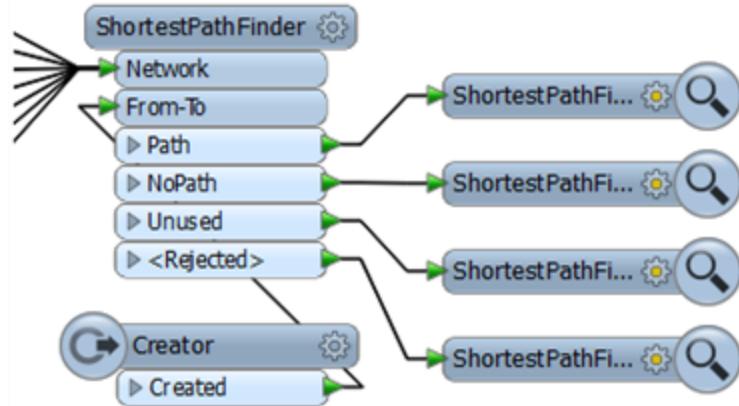
So, open the ShortestPathFinder parameters dialog. Under Snap Options set From-To Snapping to Yes and enter 150 as the Snapping Tolerance:



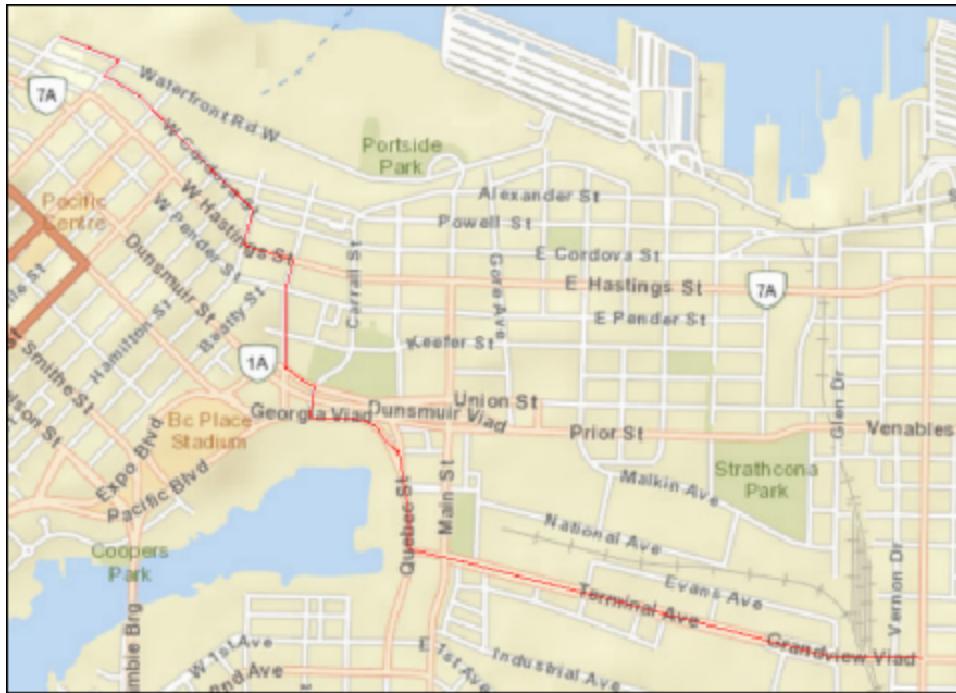
Also notice that there are parameters for network costs - we'll be making use of those later.

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:



ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

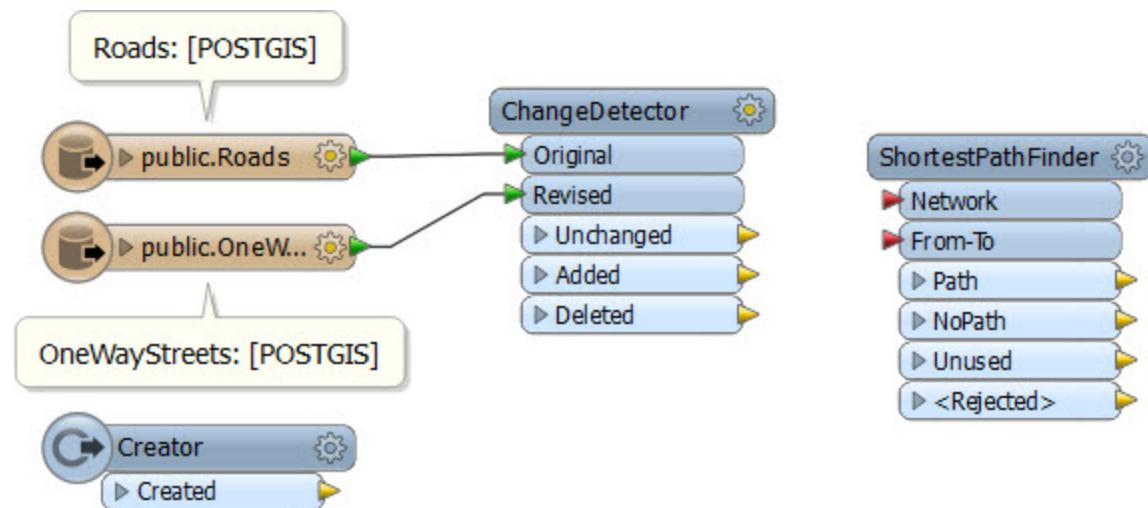
Of course, if all has not gone correctly, you must make use of Inspector and Logger transformers, plus Feature Debugging, to try and locate the error.

8) Add ChangeDetector

The result looks fine, but there are some things we are yet uncertain about: what if the route takes us the wrong way down a one-way street; and what if it uses slower, residential routes?

The first issue we can fix - most quickly - by ignoring one-way streets (i.e. avoiding them). This we can implement in FME by matching up one-way streets so we can remove them from the network.

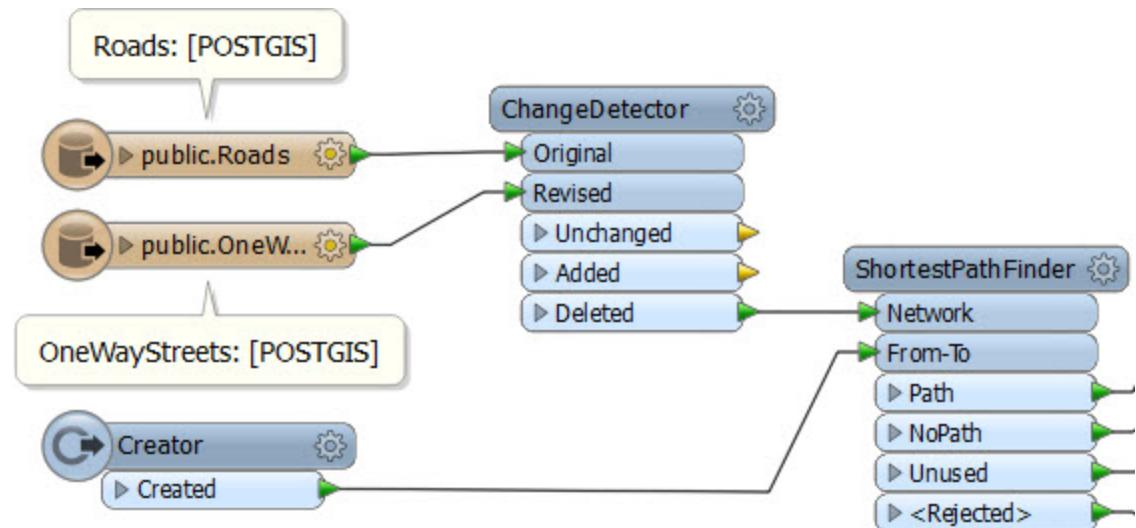
Add a ChangeDetector transformer to the workspace. Connect the Roads feature type to the Original port and the OneWayStreets feature type to the Revised port: i.e the Roads feature type includes one-way streets, and we can match them with the One-Way Streets feature type to filter them out:



Open up the ChangeDetector parameters. Note that we are matching geometry (correct) so don't change any parameters, just click OK to close the dialog.

By using the ChangeDetector road features that match a one-way street will emerge from the Unchanged port, so we should leave this unconnected. Similarly, Added features will be one-way streets that haven't been matched, and we can leave these out too.

So connect the Deleted port to the ShortestPathFinder Network port. The "Deleted" features are those that the transformer thinks have been deleted because they exist as roads, but not in the one-way data.



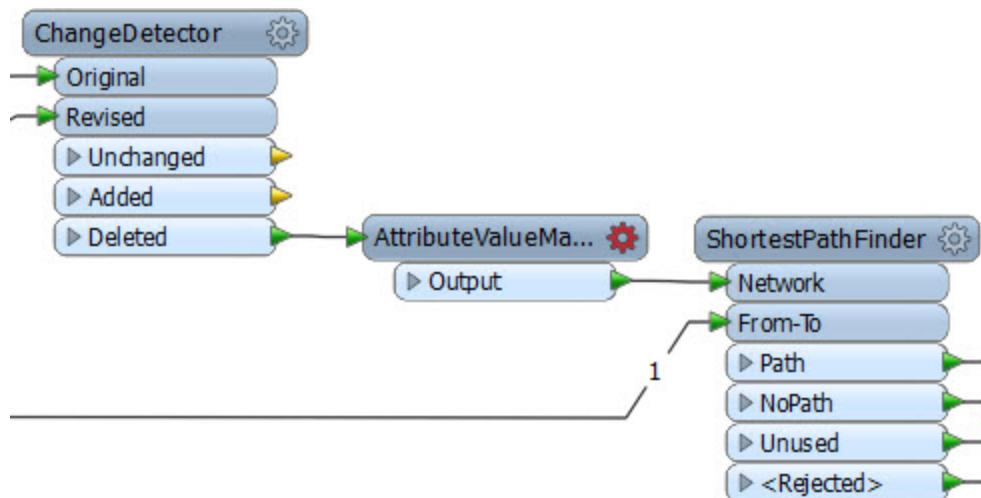
9) Re-Run Workspace

Now re-attach any Inspector transformers (if necessary) and re-run the workspace. Check the new route. If it has changed from before then you know this is because it is now avoiding one-way streets. This might not be the shortest route, but it's one we can confidently follow without heading directly into incoming traffic!

10) Add AttributeValueMapper

To weight our route in favour of arterial roads (not residential) we need to give each type of route a cost.

The best solution is an AttributeValueMapper transformer; so place one of these into the workspace. After the ChangeDetector will be best, as then some features will have already been filtered out and this transformer will have less work to do.



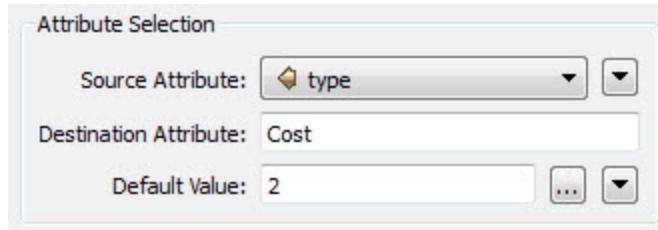
11) Edit AttributeValueMapper

Now we can set up the AttributeValueMapper transformer. Open its parameters dialog:

Select 'type' as the Source Attribute.

Enter Cost as the Destination Attribute.

Enter 2 as the Default Value.



Now, underneath those parameters, we'll map some data.

Enter "Arterial" into the first Source Value field and a value of 1 in the matching Destination Value.

Enter "Residential" into the second Source Value field and a value of 3 in the matching Destination Value.

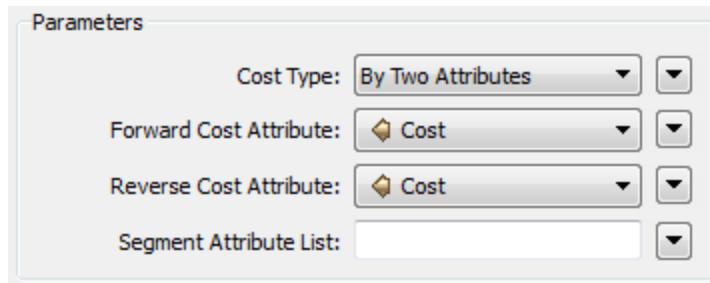
Value Map	
Source Value	Destination Value
Arterial	1
Residential	3

+ - ▲ ▼ ✎ ✏

Import...

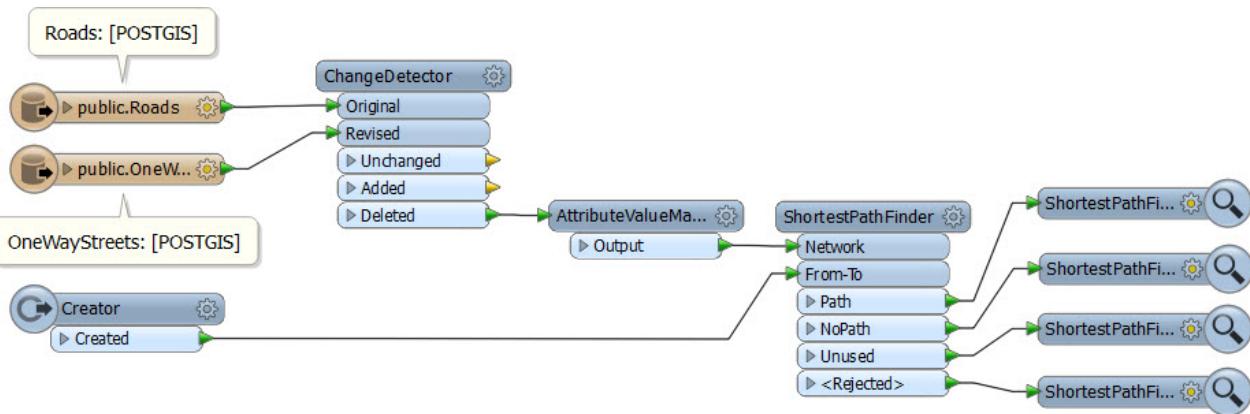
Basically, if the route is arterial (a main road) it will get a cost of 1; residential routes will get a cost of 3 and all other types will get a cost of 2 (because that's the default value). Click OK to close the dialog.

12) Now we'll apply that newly calculated cost in the ShortestPathFinder. Open the ShortestPathFinder parameters dialog. Set the Cost Type parameter to be "By Two Attributes". Select "Cost" as the attribute for both the Forward and Reverse cost parameters (i.e. it will cost the same whichever direction the road feature runs and whichever direction we travel along it).



13) Run Workspace

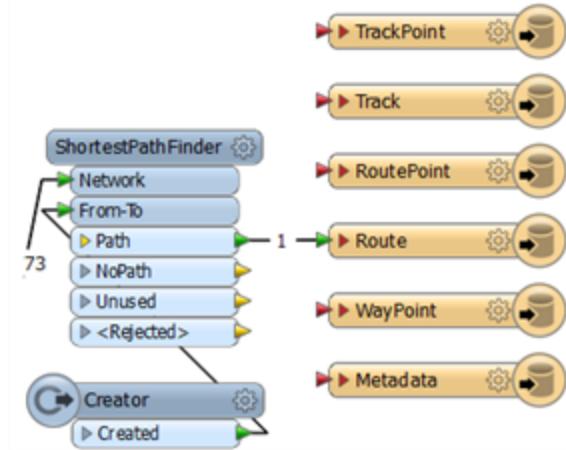
The workspace will now look like this:



Save and then run the workspace. You'll find that your new route directs you from the convention centre to Commercial Drive avoiding one-way streets and taking the slowness of residential routes into account.

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!

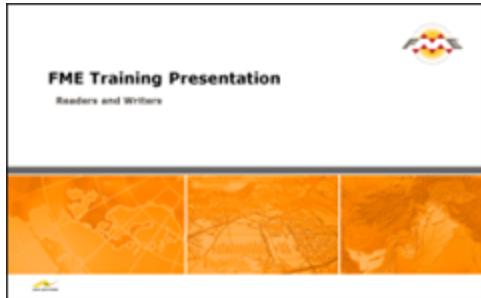


Not really advanced, but you did use Best Practice throughout, right? I mean, you have bookmarks and annotations where needed, and no overlapping connections? If not, well you might want to fix that!



Congratulations! You have now completed the exercise for this chapter.

Chapter 4 - Readers and Writers

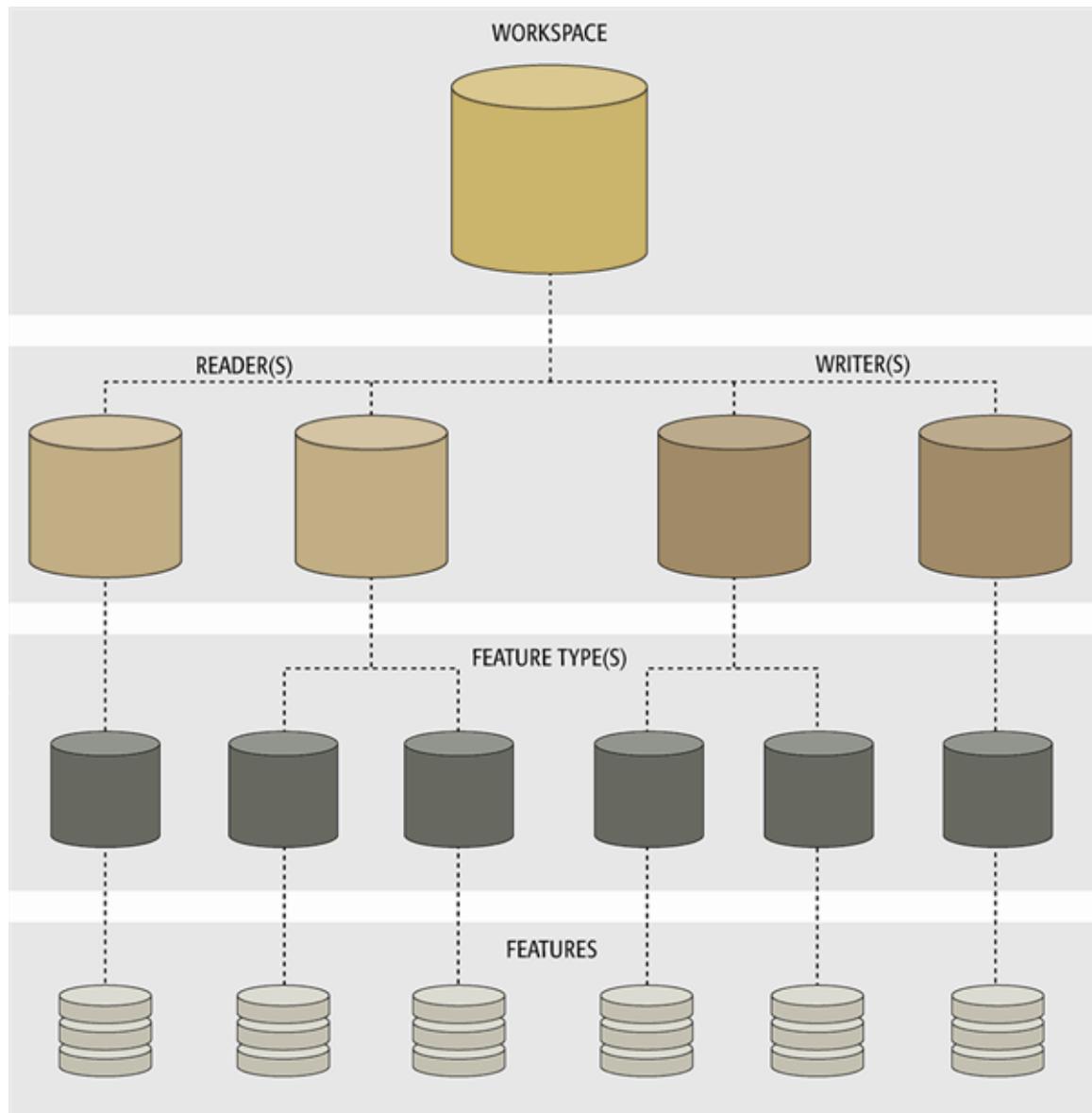


An FME translation is made up of a number of different components.

There are many different components that go to make up a translation.

For each component there are tools within FME to create, add, or remove them; and parameters in Workbench in order to control them.

In particular, each component has very specific terminology applied to it, and it's useful to have a full understanding of this terminology, especially when working with multiple datasets.



Key Components



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

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

- Workspace
- Readers
- Writers
- Feature Types
- Features

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

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



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, it's easy to look at this hierarchy diagram and imagine where such custom components might fit it.

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.



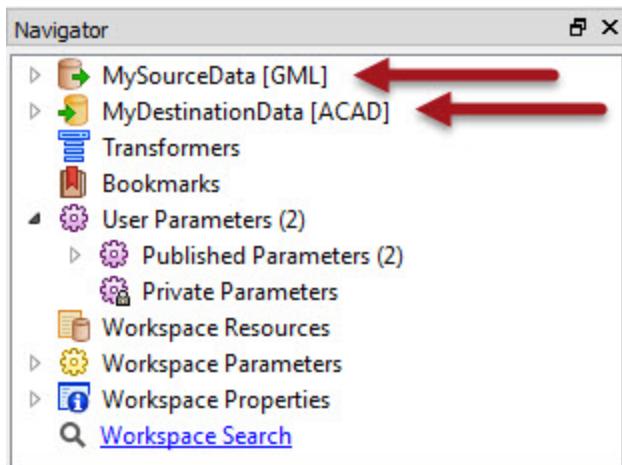
Readers and Writers

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.

Each Reader and Writer in a workspace handles just a single format of data. To read or write multiple formats requires the use of multiple readers and/or writers.

It's important to note that **Readers and Writers don't appear as objects on the Workbench canvas.**

Instead they are represented by entries in the Navigator window. Each reader and writer appears as a separate entry in the list.



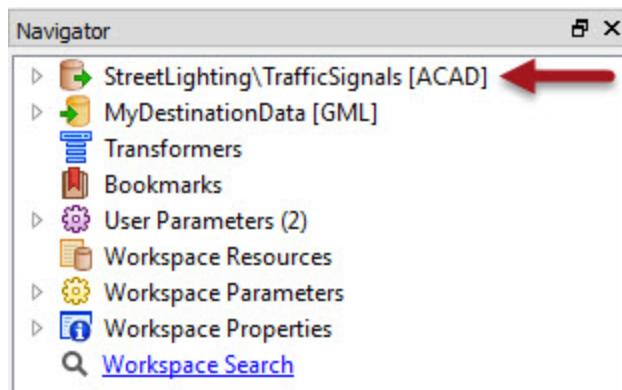
Each item in the Navigator window is represented by an icon. The icons for readers and writers look like this:



The format of each reader and writer is denoted by the format keyword. In this example the reader format is SQL Server Spatial, and the writer format is PostGIS.

The "MySourceData" and "MyDestinationData" parts of the above screenshot are the names of the datasets being read/written.

It's possible for any one Reader to read multiple datasets and, when this is the case, each is listed, like in this AutoCAD reader.



NB: Because there are some important differences between functionality of Readers and Writers, we'll deal with them as separate items throughout this chapter of the training manual.

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.

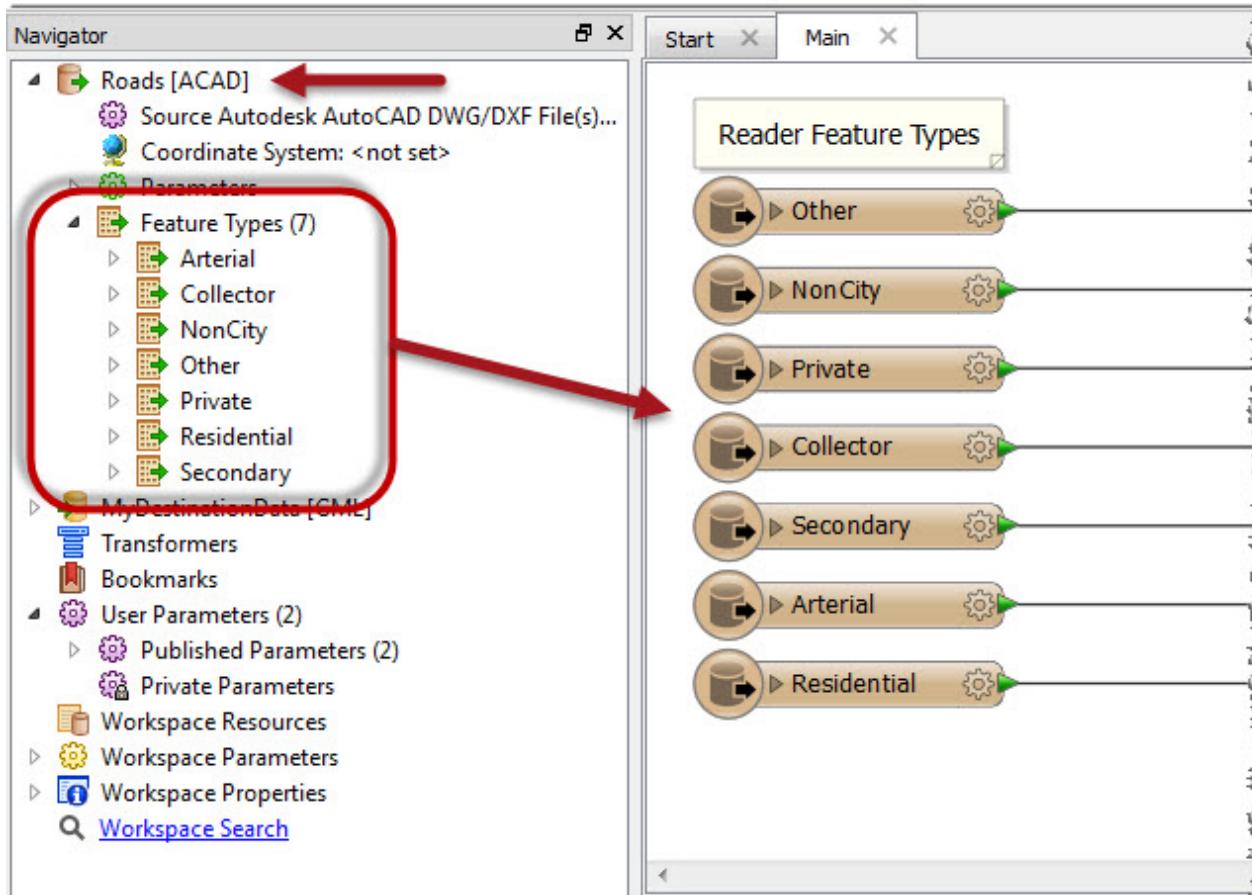
Feature Types are represented by objects in the Workbench canvas. The important part here is that **each feature type belongs to a Reader or Writer defined in the Navigator window.**

Therefore, when reading data, the Navigator window defines what dataset is being read, and the canvas objects define what layers are being read from that data. If a specific layer is not defined by a feature type, then that data will not be either read and/or written.

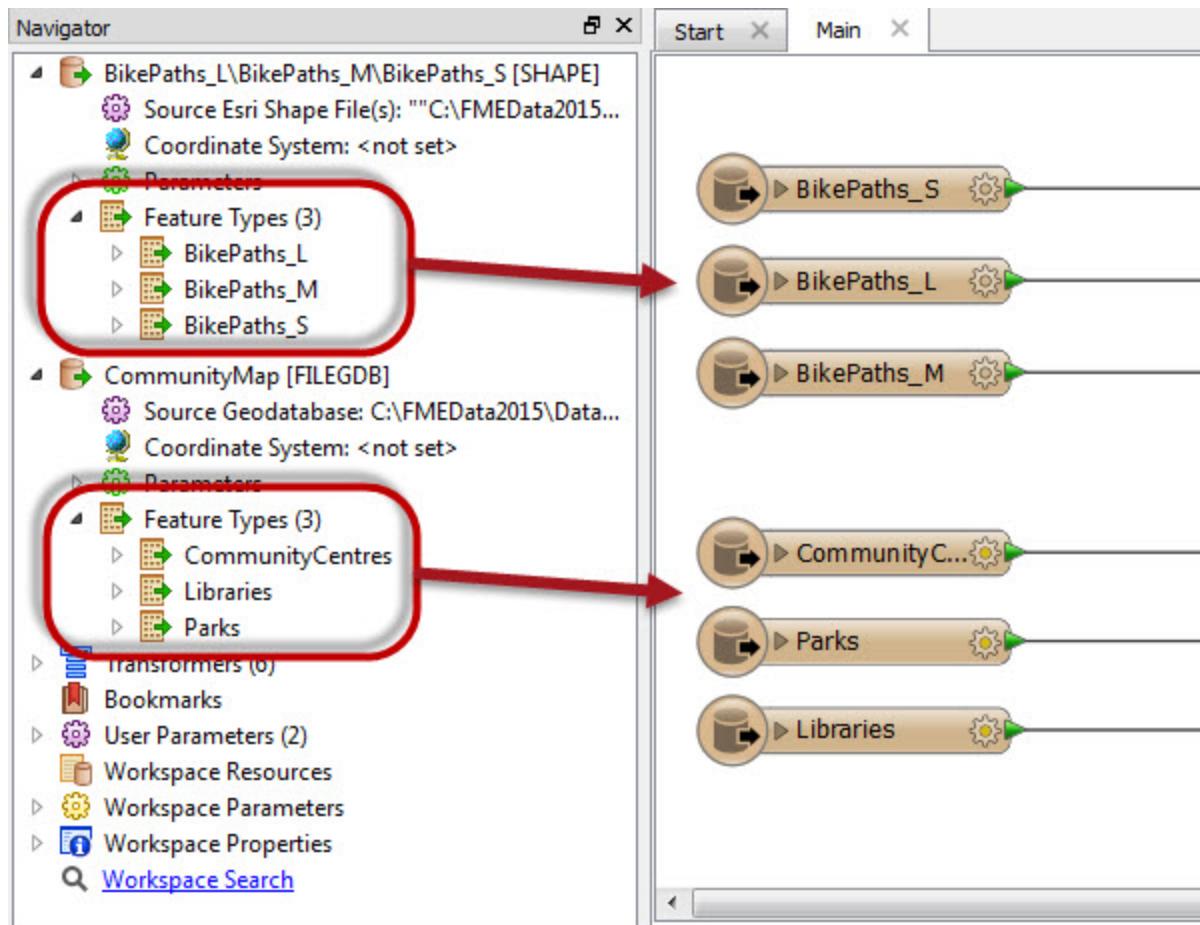


Don't confuse the term 'Feature Type' with 'Geometry Type.'
Feature Type means "layer;" Geometry Type means "lines," "points," "polygons."

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 Shape, one File Geodatabase). Each Reader has three different feature types.



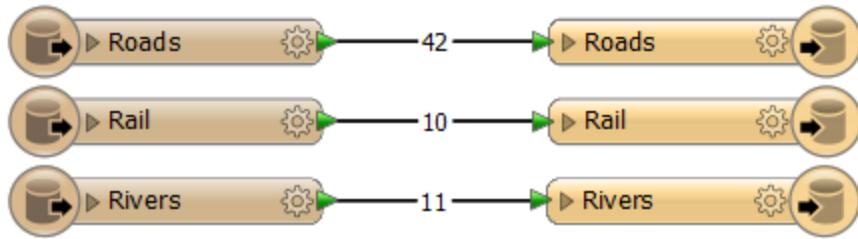
Notice - and this is where it can be confusing - there is no visible connection between the Reader and its feature types, nor is it certain that feature types on the canvas will be organized in this way (though for Best Practice purposes they should be!)

Features

Features are the smallest single components of an FME translation.

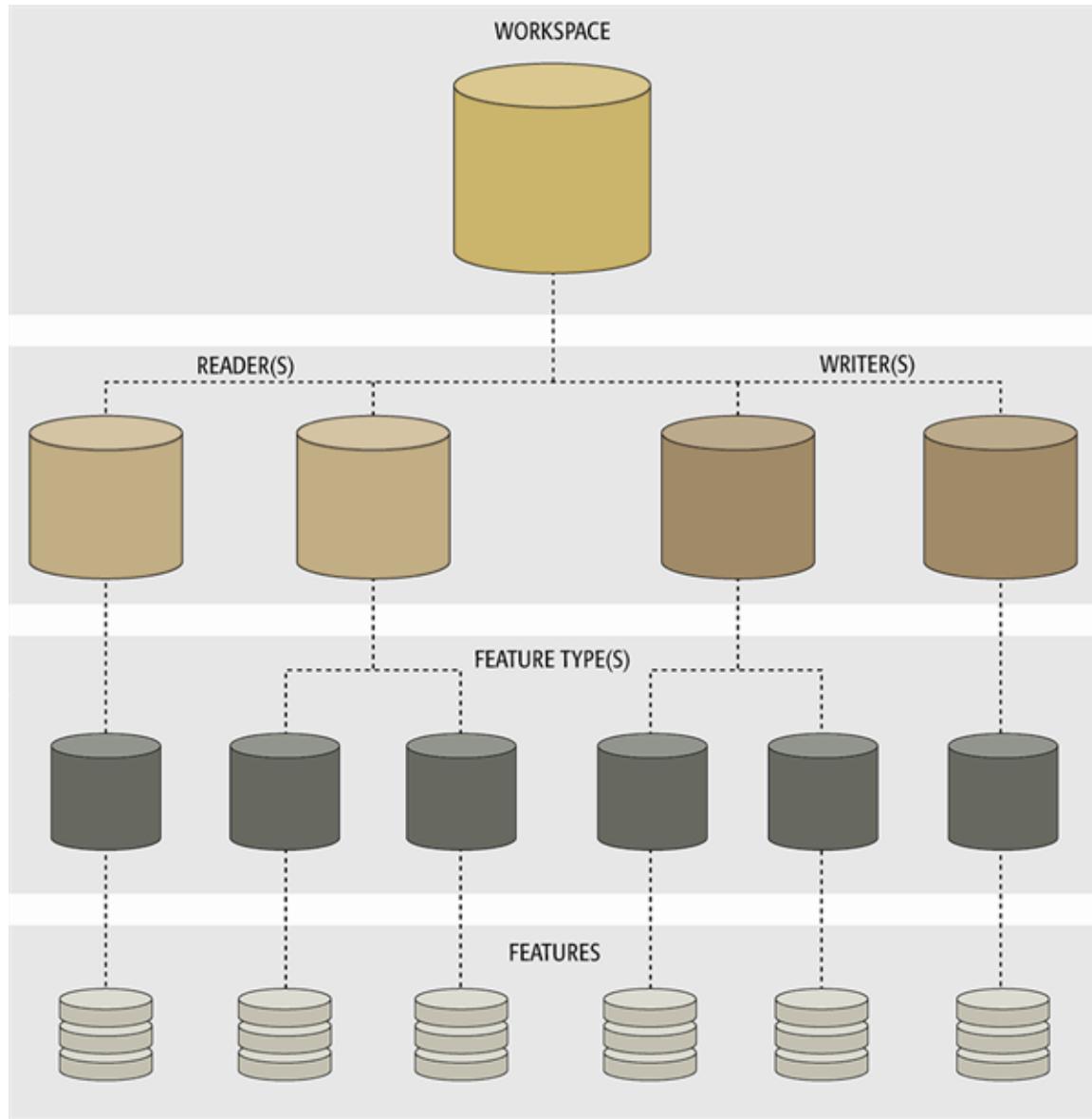
They aren't individually represented within a workspace, except by the feature counts on a completed translation.

Here 42 road features were translated.



One-To-Many Relationships

The hierarchical relationship between workspace, readers, writers, feature types, and features is always one-to-many (1:M) with the level beneath:



Notice how 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 any number of different datasets and data types, each of which can have its own unique set of feature types. Each feature type will, usually, contain multiple features.

Managing Components

FME has a number of tools for managing components in an FME translation:

- Create Workspace
- Add Reader/Writer
- Add Feature Types
- Import Feature Types
- Remove Reader/Writer
- Remove Feature Types
- Update Feature Types
- Move Feature Types

Controlling Components

Each component has a set of settings that control how the component behaves when the translation is run. In FME terms we call these Parameters and, like transformers, each component has its own set of parameters that control its actions.

Now we know what these components are, let's look at each of them individually, in detail.

Workspaces



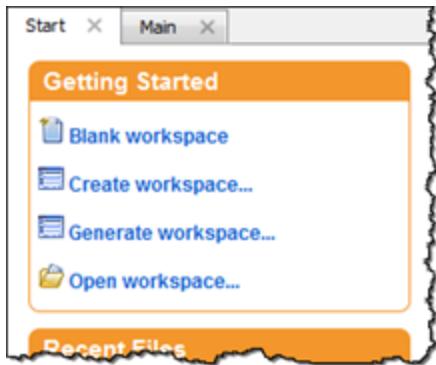
A workspace is a file responsible for storing a translation definition.

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

A workspace can be created empty - i.e. the canvas is blank and each new component is added from scratch - or it can be created so that its initial state contains a number of other components.

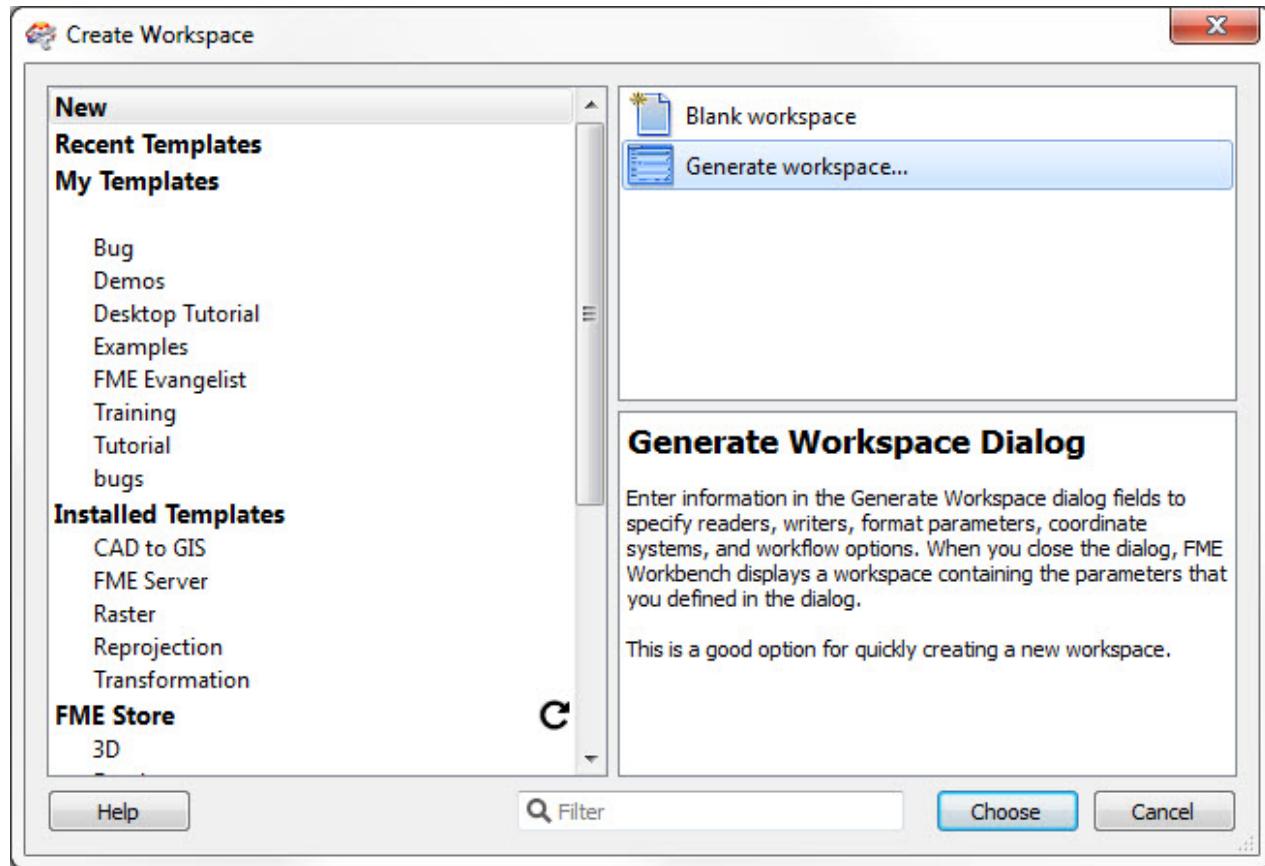
Workspaces can be created using the commands on the File menu, or through shortcuts in the Start tab.



The Blank Workspace option simply empties the canvas and allows the user to construct all components manually. The Create and Generate options are ways to create a workspace containing some components.

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

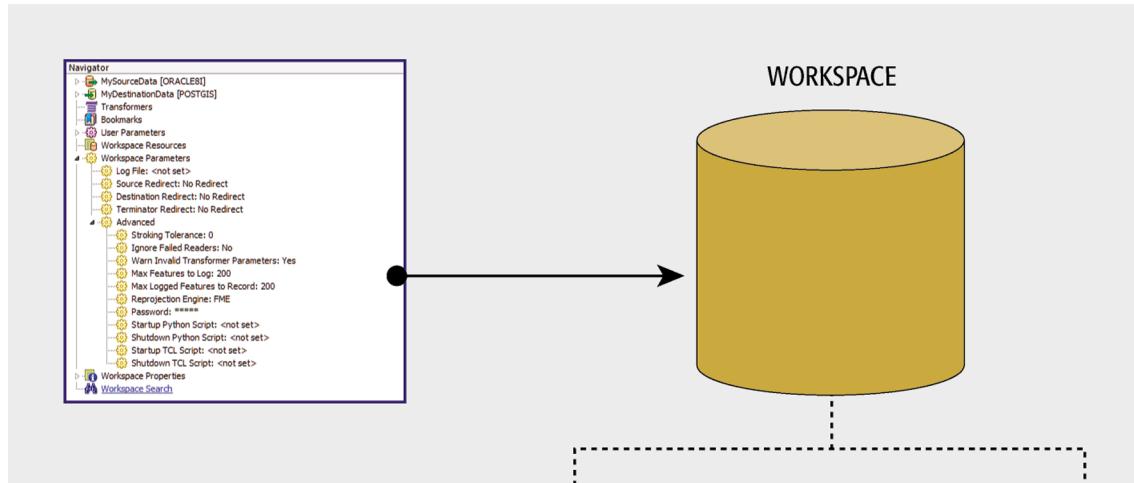
The Create Workspace dialog provides options for generating a workspace, but also for creating one from a template installed either locally or in the FME Store.



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. They apply to the current workspace only and may change between workspaces.

Workspace parameters are shown and set in the Navigator Window.

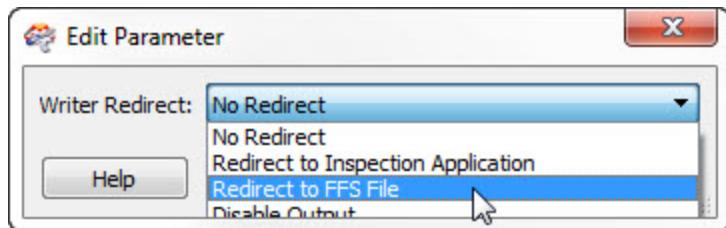


For ease-of-use, workspace parameters are divided into two sections: basic and advanced.

Basic Workspace Parameters

There are a number of basic workspace parameters. The most important one is Writer Redirect.

The Writer Redirect parameter overrides the Writer defined in the workspace. It causes FME to send the translation output elsewhere and no data is written to the destination datasets. To write output again the user must remove the redirect by choosing the No Redirect setting.



The Writer Redirect options are:

- Redirect to Inspection Application: Output is sent directly to the FME Universal Viewer.
- Redirect to FFS File: Output is sent to an FFS (FME Feature Store) file.
- Disable Output: Output is ignored and not used (similar to a NULL format writer).



'Redirect to Inspection Application' can also be found on the menu bar, under the Writers menu.

Password

It's often desirable to pass a workspace to an FME user for them to run, but not to edit. A password-protected workspace cannot be opened for editing in Workbench without the password.

It can, however, still be run within the FME Universal Translator or from the command line.

Also, developers or consultants may want to pass on a workspace to an FME user without revealing the contents. Password protecting a workspace causes it to be encoded so that its contents cannot be read in a standard text editor.

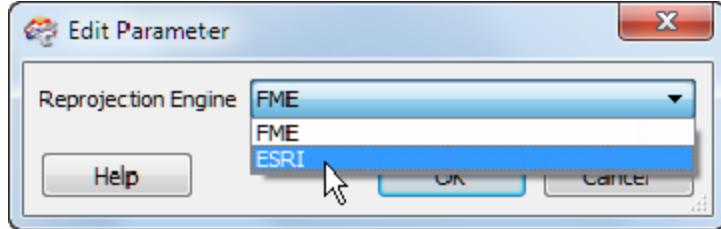
Advanced Workspace Parameters

The advanced workspace parameters are perhaps not as valuable in everyday use, but have great importance in specific scenarios. Some particularly important ones are:

Reprojection Engine

Different GIS applications have slightly different algorithms for reprojecting data between different coordinate systems. To ensure that the data FME writes matches exactly to existing data, this parameter permits a user to use the reprojection engine from a different application.

A user with ArcGIS installed is choosing to use that package's engine for reprojecting the spatial data.



Start-up and Shutdown Scripts

These parameters deliver the ability to run a TCL or Python script before or after an FME translation.

Script parameters in the workspace settings dialog:

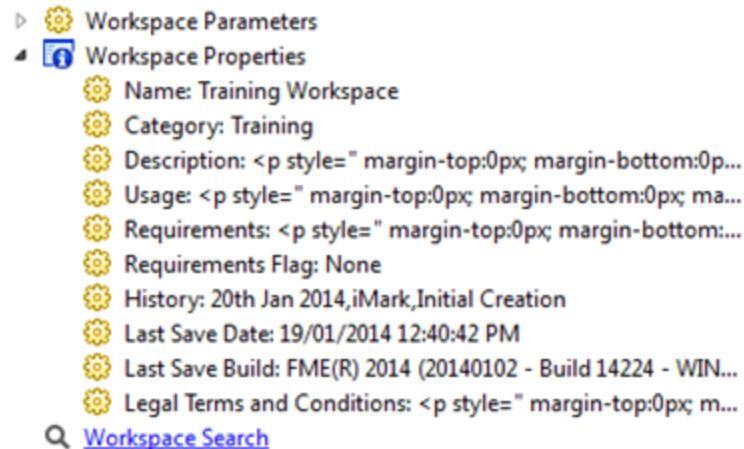
- ⚙ Startup Python Script: <not set>
- ⚙ Shutdown Python Script: <not set>
- ⚙ Startup TCL Script: <not set>
- ⚙ Shutdown TCL Script: <not set>

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

Workspace Properties

Settings that provide information about a workspace, but have no effect on the translation - such as Workspace Name and Workspace Description - are found in the **Workspace Properties** section.



A screenshot showing the 'Workspace Properties' section of a software interface. The section includes fields for Name (Training Workspace), Category (Training), Description (HTML placeholder), Usage (HTML placeholder), Requirements (HTML placeholder), Requirements Flag (None), History (Creation date), Last Save Date (19/01/2014 12:40:42 PM), Last Save Build (FME(R) 2014 (20140102 - Build 14224 - WIN...)), and Legal Terms and Conditions (HTML placeholder). A search bar at the bottom is labeled 'Workspace Search'.



Workspace Properties are basically metadata fields that are useful in investigating the workspace's history. They appear in the Create Workspace dialog when saved as a template.

Readers



Readers are the main components that read data in FME.

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, but can read any number of datasets in that format.

When a workspace is generated with the Generate Workspace dialog then it is created with just 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 25 Readers and 32 Writers of all data types and formats!

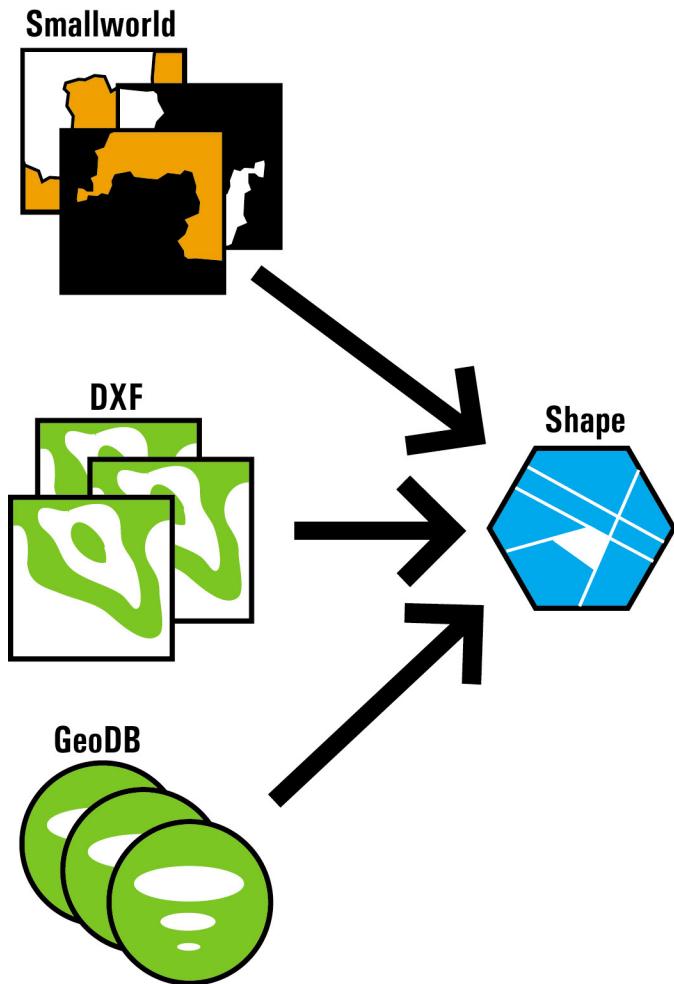


Adding a Reader

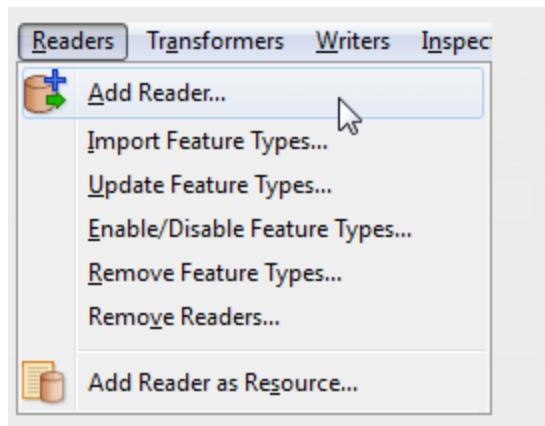
Adding a Reader to a workspace is a common requirement. There are several reasons:

- 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

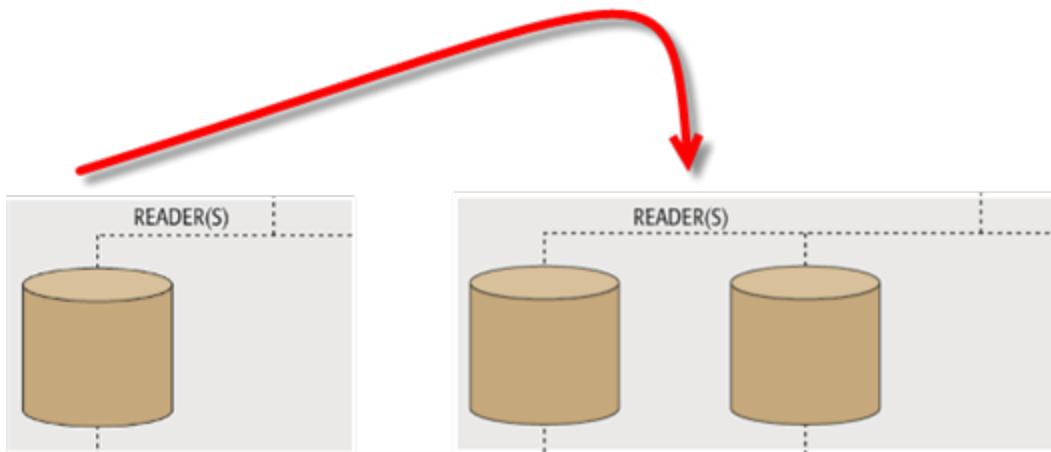
Therefore the need to read multiple formats of data – such as Smallworld, DXF, and Geodatabase – requires multiple Readers.



Additional Readers are added to a translation using **Readers>Add Reader** from the menubar.



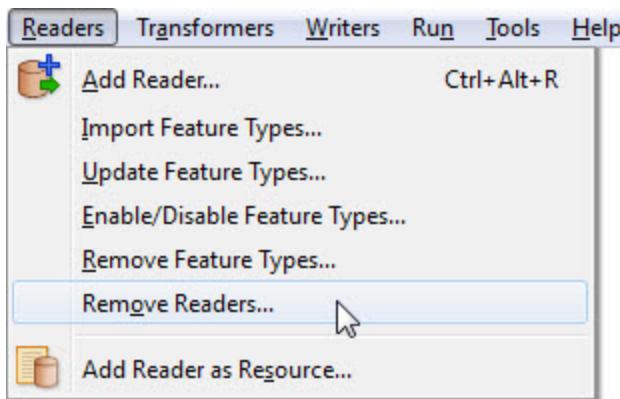
Adding a Reader has this effect on the hierarchy diagram:



Although the usual workflow is to create a new workspace with the Generate dialog and then add extra components as necessary, there's nothing to prevent a user starting with an empty workspace and simply adding Readers (and Writers) one-by-one.

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.



Controlling Readers

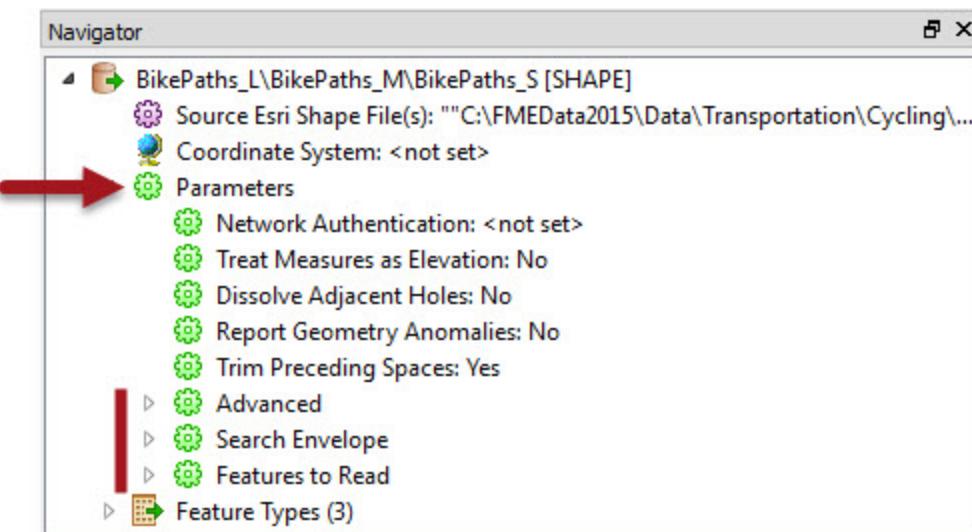
Reader parameters are those that control how data is read.

Because these parameters refer to specific components and characteristics of the related format, no two formats will have the same set of control parameters.

Reader Parameters

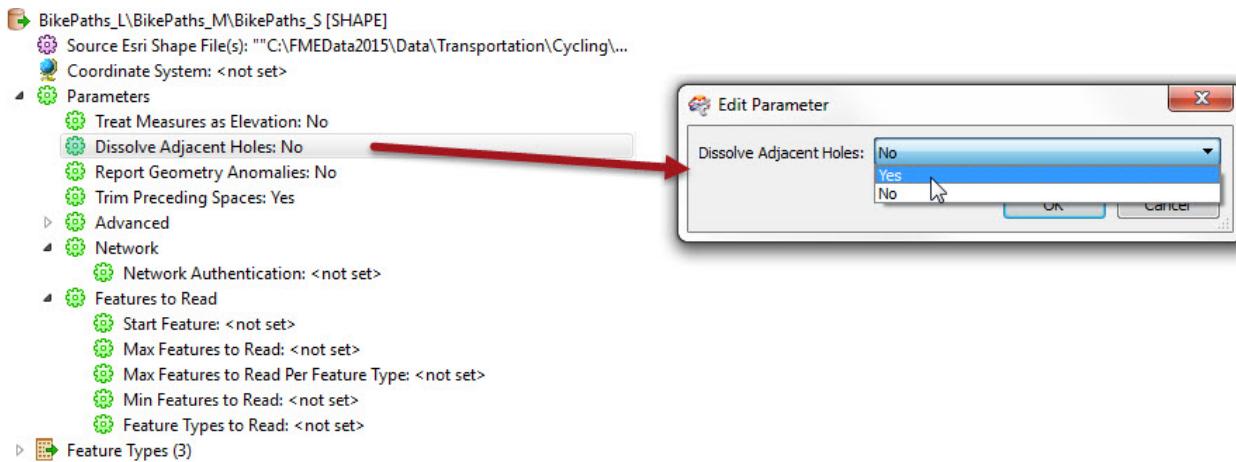
Reader parameters are shown and set in the Navigator Window. You expose the list of parameters by clicking on the expand arrow icons to the left of the Reader.

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.



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.

To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.




Doctor Workbench says...

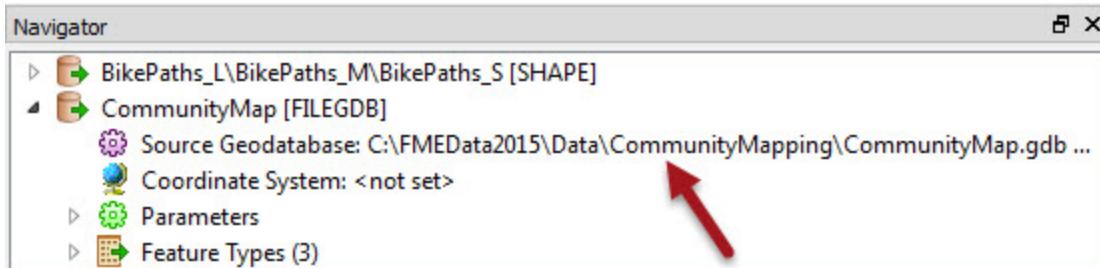
'Some Reader parameters are ONLY accessible through the Parameters button when you initially create a workspace or add the Reader to an existing workspace. That's because they affect how the schema is read and therefore how the workspace is constructed.

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 Parameters

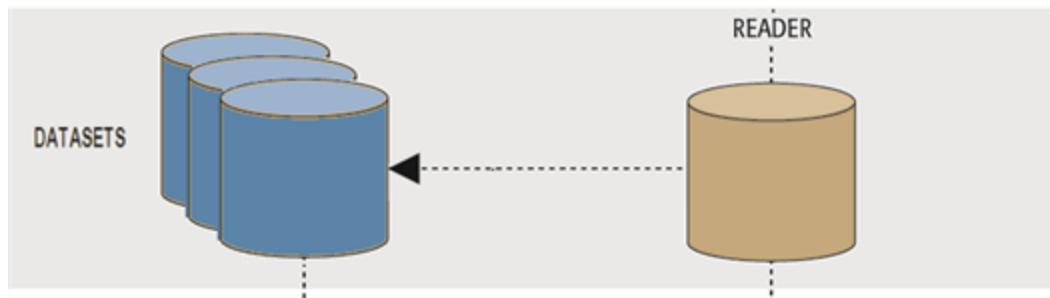
As you may have noticed, among the different parameters for each Reader are ones to define which dataset it is reading.



Therefore, to select a dataset , these are the parameters you use. The dataset being read is NOT defined by a canvas object.

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.

Each Reader is capable of reading multiple source datasets.



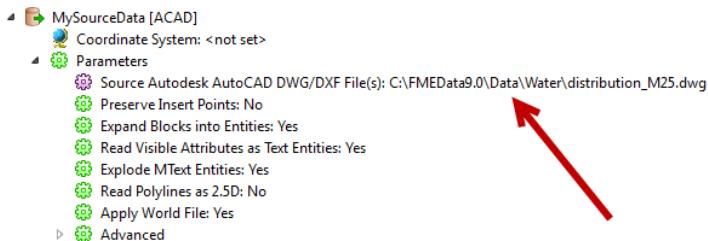
When dealing with files (as opposed to databases) FME recognizes two different types of dataset: File-based and Folder-based.

File-Based Datasets

The key characteristic of a file-based dataset is that all layers (feature types) are stored within a single file. Basically a format in this classification will have a way to assign data to different layers within a single file.

An AutoCAD DWG file is a good example of this: each DWG file is a separate dataset, and each DWG file has its own set of layers. For tabular data, Excel is a good example. A single Excel file can contain multiple sheets of data.

In this scenario the dataset parameter is simply a pointer to the location of the file(s).



Notice here the source dataset is a single .DWG AutoCAD file.

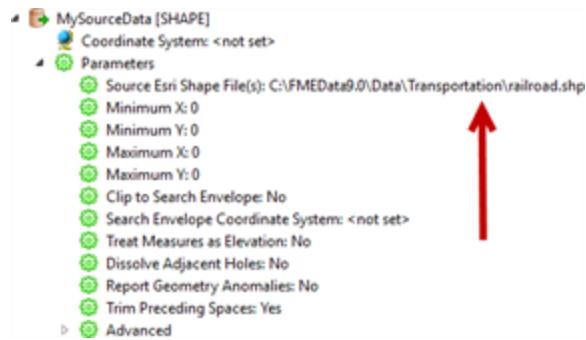
Folder-Based Datasets

In this class of format, layers are stored as separate files. In other words, a format of this type DOES NOT have a way to assign data to different layers within a single file; therefore each 'layer' is a separate file.

An Esri Shape dataset is a good example of this: a single Shape file cannot store different layers (for example, roads and railway). To create that structure you would have two Shape files: roads.shp and railway.shp

For tabular data, comma-separated (CSV) or column-aligned (CAT) files are of this type. It's simply not possible to create separate tables inside a simple text file.

Reading folder-based datasets is not quite as logical as reading a file-based dataset. Rather than select the folder and prompt which feature types (files) to read, the user is prompted to select the **files** (feature types) directly.



In this way, the dataset and feature type selection can be made in a single step. It's more efficient, just a little less logical.

Tip 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.

Reader Feature Types

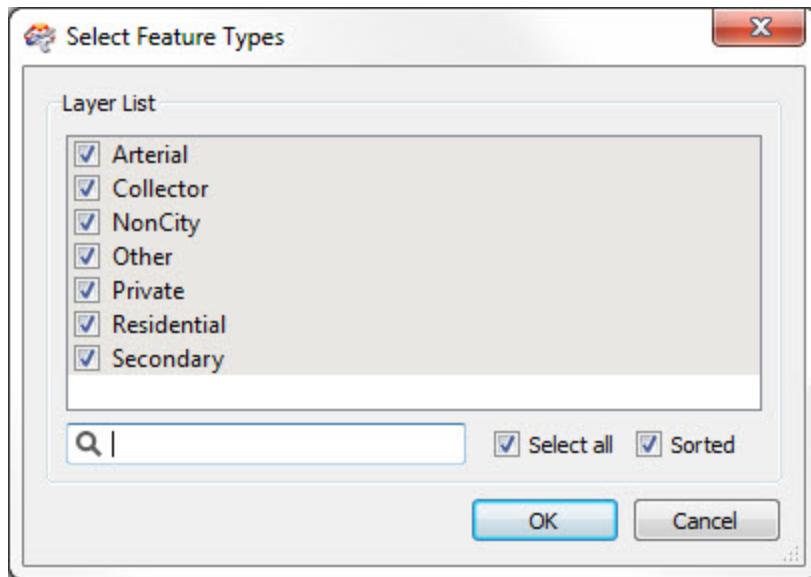
Reader Feature Types are objects on the Workbench canvas that define the schema of the data being read. Most importantly, they define two specific things: they define the layers being read from a source dataset and the attributes that those layers possess.

Adding Reader Feature Types

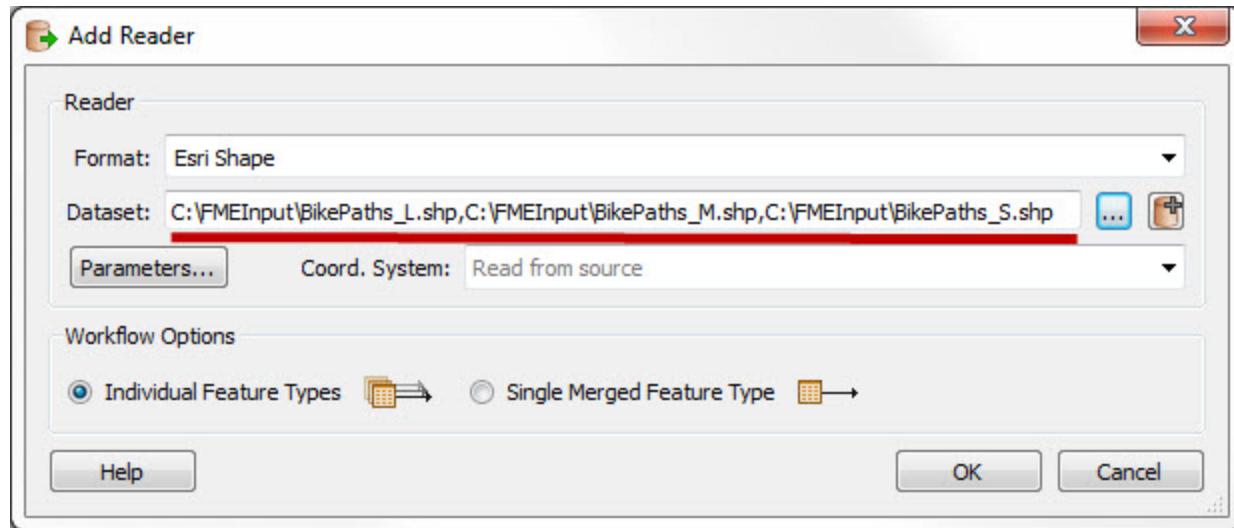
As we have already seen, the Reader schema defines “what we have” as source data. So if you want to read data from a particular layer, in a particular dataset, then it’s not enough to just select that dataset; you also have to ensure that the layer is defined as a feature type in the canvas.

Usually, defining these feature types is done when the workspace is generated or the Reader is being added.

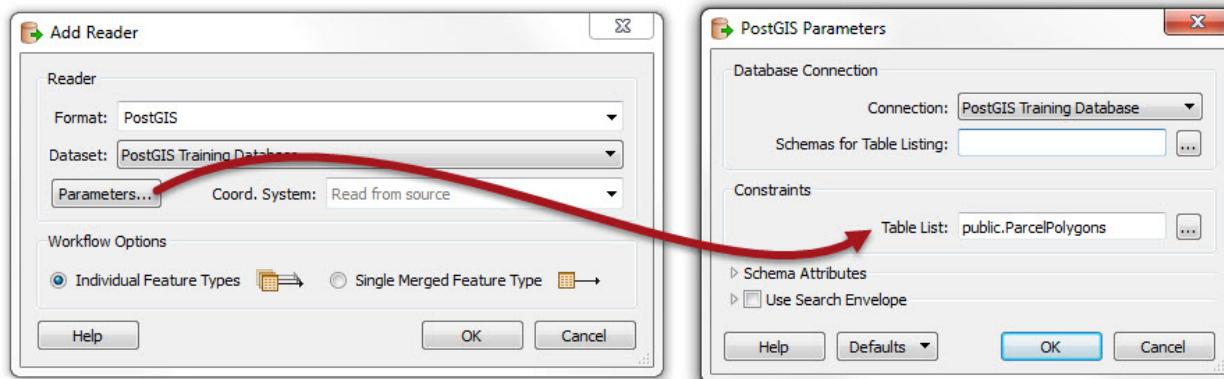
For a file-based dataset, like an AutoCAD file, FME scans the dataset to find what layers exist and then prompts the user which should be added:



For a folder-based dataset, like Esri Shape, the user should just select the files that represent the feature types:



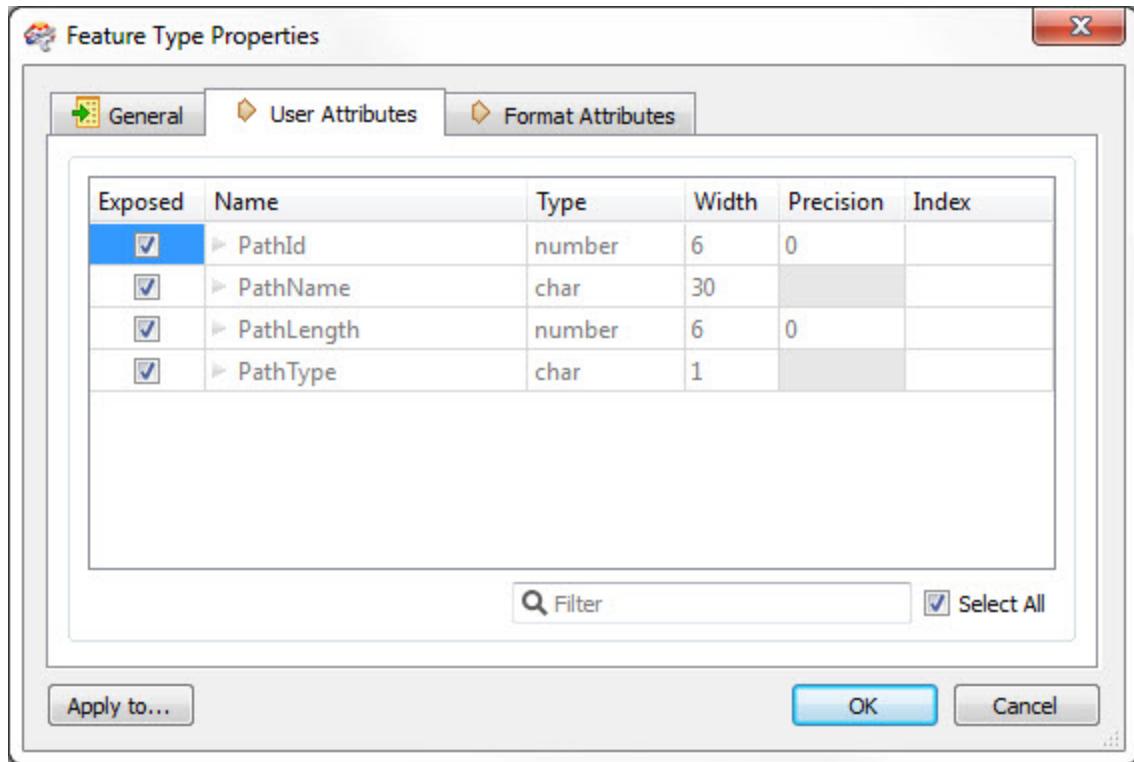
For a database, like PostGIS, the user should click the parameters button and use the Tables to Read parameter:



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.

Feature Type Attributes

Reader feature types contain a definition for the attributes that belong to it:



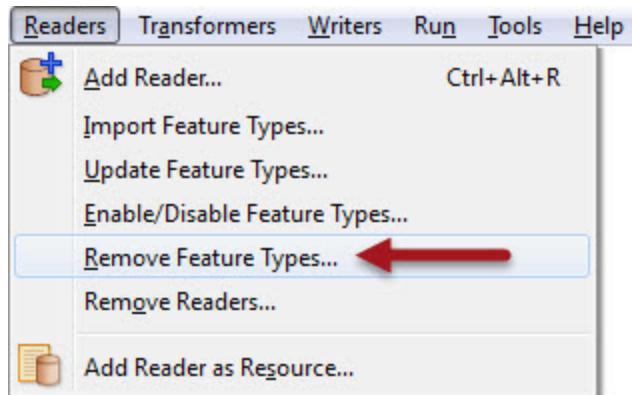
Notice that the attribute types relate to the format being read, and are not a generic FME representation. Also notice that, because the Reader feature types represent "what we have", the attribute definitions are greyed-out and cannot be edited.



*Under **Tools > FME Options > Workbench** there is a setting where you can turn on Reader feature type editing. One scenario where you might want to do so is this: You have a client. His name is Bond. James Bond. He wants you to create a workspace for him that reads a top-secret database. He cannot let you connect to the database, but he will tell you the schema. In that case you can create his workspace manually by turning on this option.*

Removing Feature Types

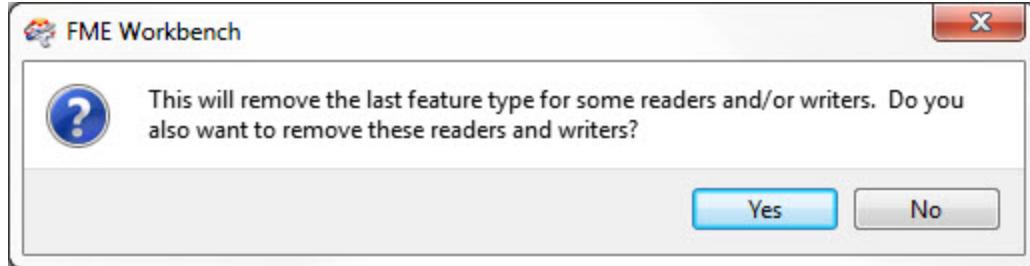
The remove feature types tool is found on the menubar, and simply removes one or more feature types from the Reader.



Another way to do this is to select the feature type on the canvas and press the keyboard delete key.

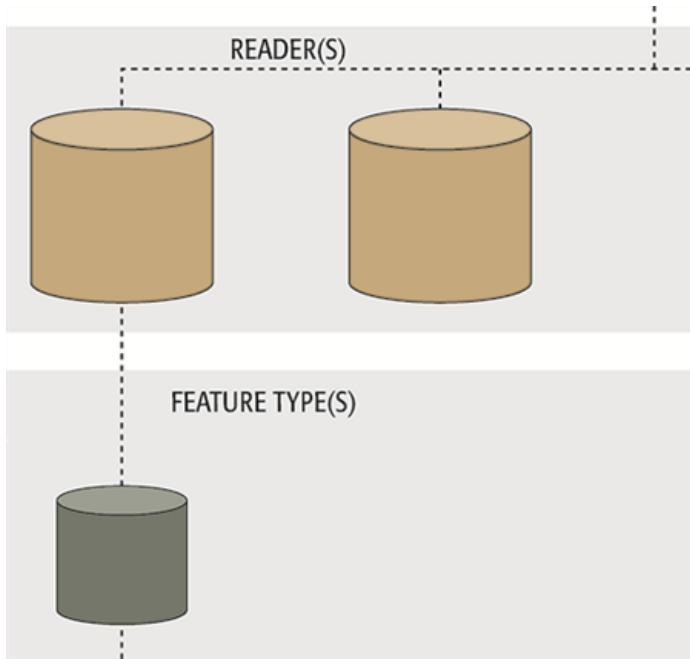
For example, if there is a source dataset layer that the user wishes to no longer read, then they would delete the matching feature type from the workspace.

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.



This makes sense because if there are no feature types you wish to keep, why would you still wish to read the dataset at all?

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.



! A *dangling Reader* isn't a problem provided it's only a temporary situation; i.e. the user intends to now add new feature types.

The workspace should not be run in this condition!

Performance suffers because all the source data is still being read, yet discarded immediately.

Readers

Exercise 4a Readers	
Scenario	FME User
Data	Property Parcels (DWG, PostGIS) Public Art (Excel) City Properties (Esri Shape)
Overall Goal	Create list of public art on city property
Demonstrates	Managing and controlling Readers and Writers
Starting Workspace	None
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise4a-Complete.fmw</i>

In this exercise you are an FME user at the City of Interopolis. The city is planning its maintenance budget for the year and wishes to know how many pieces of public art are erected on city-owned property.

So, your task is to use the data available to identify city properties and find out what pieces of artwork are located on them.

1) Inspect Data

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

Reader Format	PostGIS
Reader Dataset	PostGIS Training Database
Parameters	
Table List	ParcelPolygons
Reader Format	Microsoft Excel
Reader Dataset	<i>C:\FMEData2015\Data\Culture\PublicArt.xlsx</i>
Coordinate System	LL83
Reader Format	Esri Shape
Reader Dataset	<i>C:\FMEData2015\Data\Parcels\CityProperties\CityProperties.shp</i>

If you haven't realized it yet, this is going to be a tough assignment. The city owned properties are point features, not polygons, so we are going to have to use the parcel data to discover the actual area of each property. Then we will have to overlay the public art features, which at the moment are in a non-spatial format of data.

NB: If you don't already have the "PostGIS Training Database" connection, then look up the exercise for Chapter 3 (Best Practice) and carry out the first step in there. Alternatively, 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:\FMEData2015\Data\Parcels\Parcels.dwg

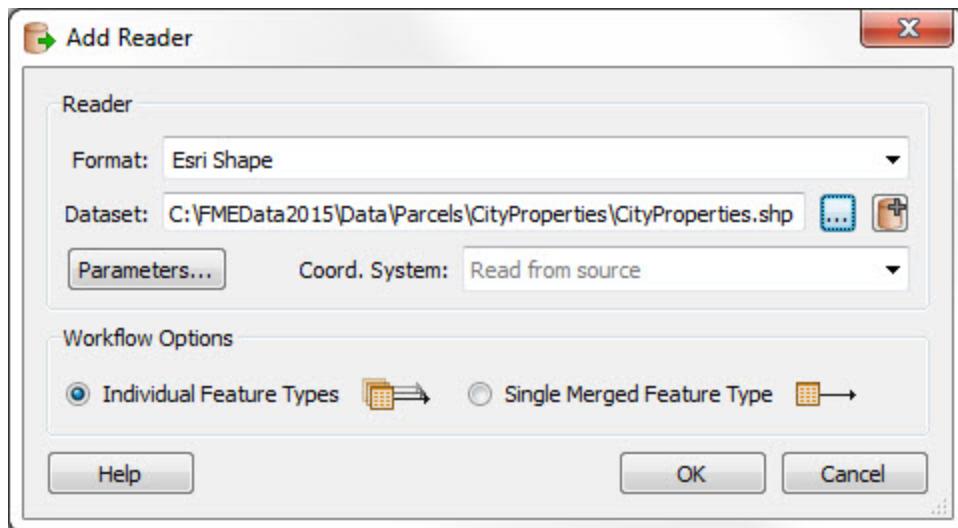
Parameters

GroupEntities By: Attribute Schema

2) Start Workbench

Start Workbench and begin with an empty workspace. Because there are several formats of source data it's going to be easier to add Readers individually, rather than generate a workspace.

Choose **Readers > Add Reader** from the menubar. When prompted enter the format and dataset of the city properties (Shape) dataset we are using. The dialog should be very similar to the one in the Data Inspector.

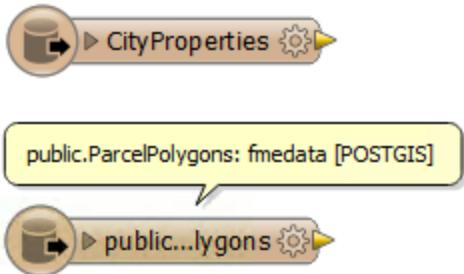


Click OK to close the dialog and add the Reader to the workspace.

3) Add Reader

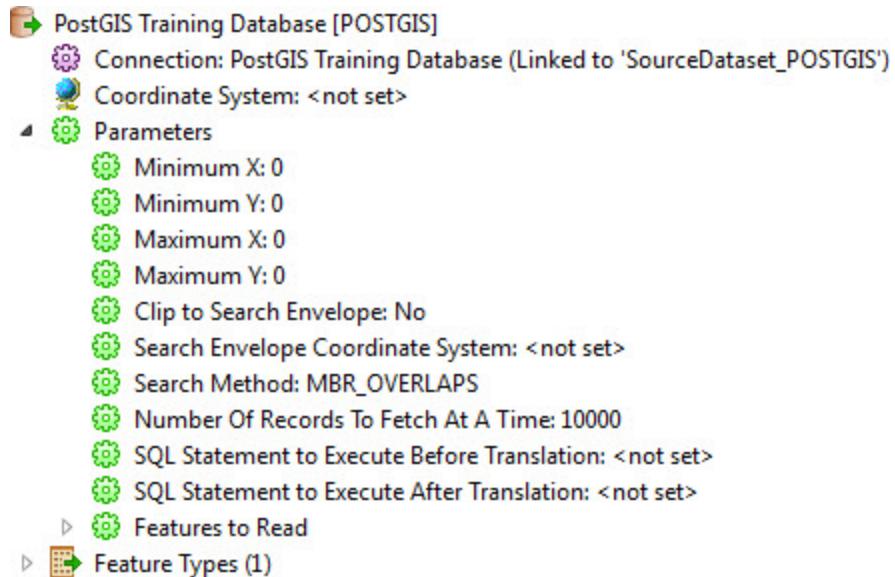
Now we need to add the property boundaries (PostGIS) dataset to the workspace. Use **Readers > Add Reader** as in the previous step and the Reader Format/Dataset shown in step 1.

The workspace will now look like this:



NB: If you are unable to use the PostGIS database, this is where you can substitute the DWG dataset.

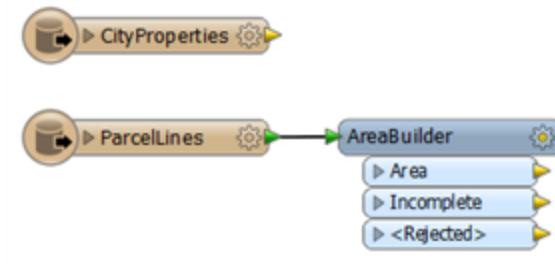
Take a look at the parameters for this Reader in the Navigator window. Notice how there are many options for how we read individual features of the dataset, plus several others relating to database performance:



3b) Add AreaBuilder

If you are using the AutoCAD (not PostGIS) data, you may have noticed that the property boundaries were just lines, and not polygons. To turn them into polygons requires a transformation.

Add an AreaBuilder transformer and connect it to the ParcelLines feature type:

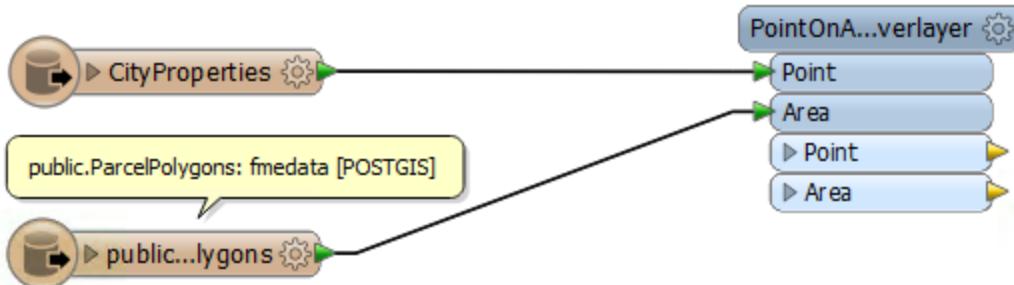


You can check the parameters, but none should need changing.

4) Add PointOnAreaOverlayer

Add a PointOnAreaOverlayer transformer. This will tag the property polygons that belong to the city by carrying out an overlay with the City-Owned dataset.

Connect the CityProperties feature type to the Point input port. Connect the ParcelPolygons feature type (or AreaBuilder:Area port) to the Area input port:

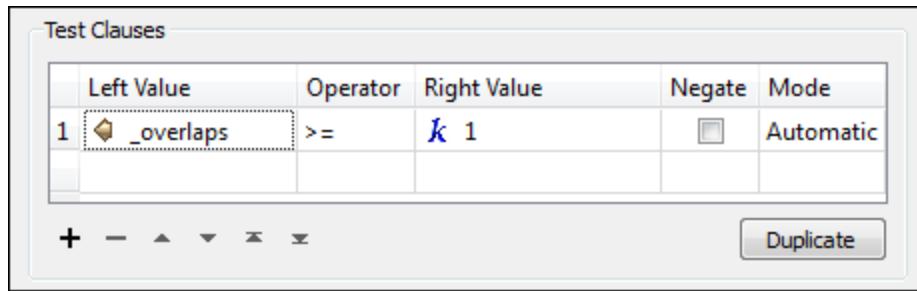


Again, you can check the parameters, but none should need changing. Note that the number of overlaps is denoted by an output attribute called `_overlaps`.

5) Add Tester

Add a Tester transformer. It should be connected to the PointOnAreaOverlayer:Area output port. We'll use this to check the value of `_overlaps`. If the value is greater than 1 then we know this is a city-owned property.

Open the parameters dialog and set up a test for where `_overlaps >= 1`.



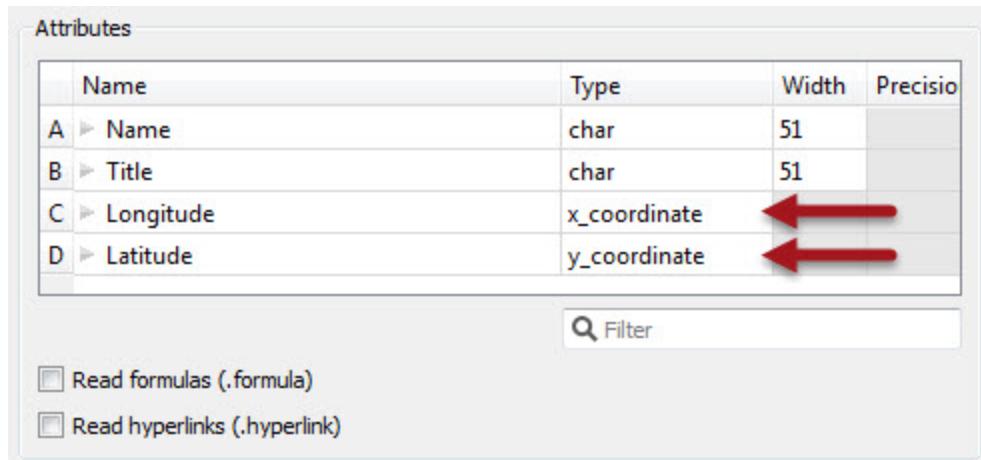
Click OK to close the dialog. You can now add some Inspector transformers to the Tester output and run the workspace to ensure that the results are correct so far. What we should get is a set of property outlines divided into city properties (Passed) and private properties (Failed).

6) Add Reader

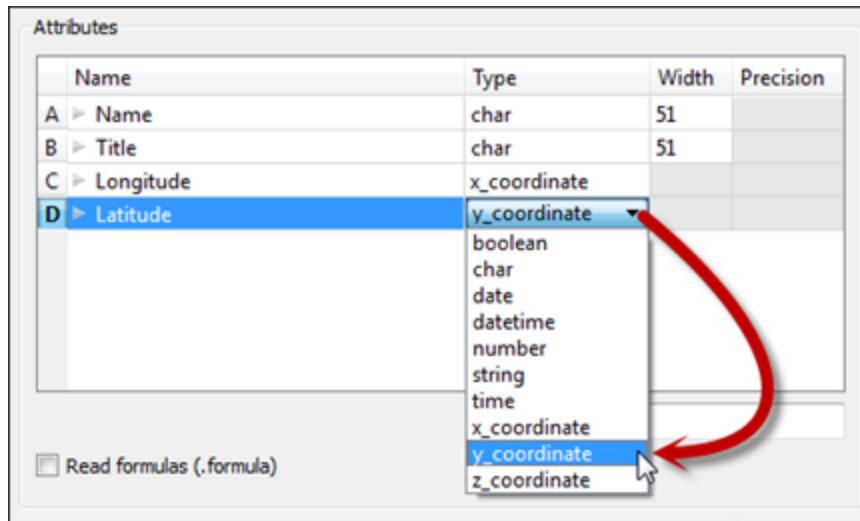
Now we must add a Reader to read the public art data. Choose **Readers > Add Reader** from the menubar. Enter the following info but don't click OK yet:

Reader Format	Microsoft Excel
Reader Dataset	C:\FMEData2015\Data\Culture\PublicArt.xlsx

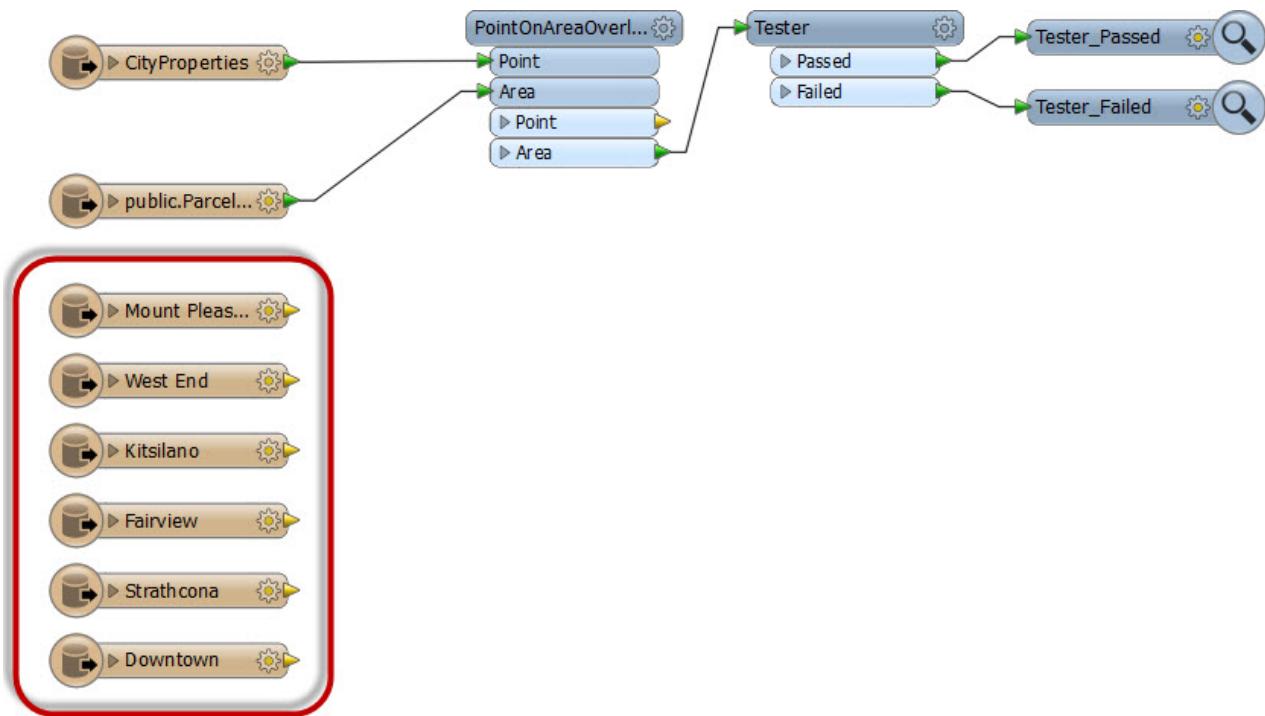
Click the Parameters button. Notice that there are a number of options to preview the data for each sheet. At the foot of the dialog is the attribute schema. Notice how there are attributes for Longitude and Latitude. Because the columns are actually named Longitude and Latitude, FME automatically realized that these can be used to convert this tabular data to true spatial features.



If the names were not as obvious, we could tell FME to automatically convert these to proper spatial features by clicking the drop-down arrow and manually selecting "x_coordinate" and "y_coordinate":



Click OK and OK again to close the dialogs and add the Reader to the workspace. Notice that there are several features types (one for each sheet in the Excel spreadsheet):



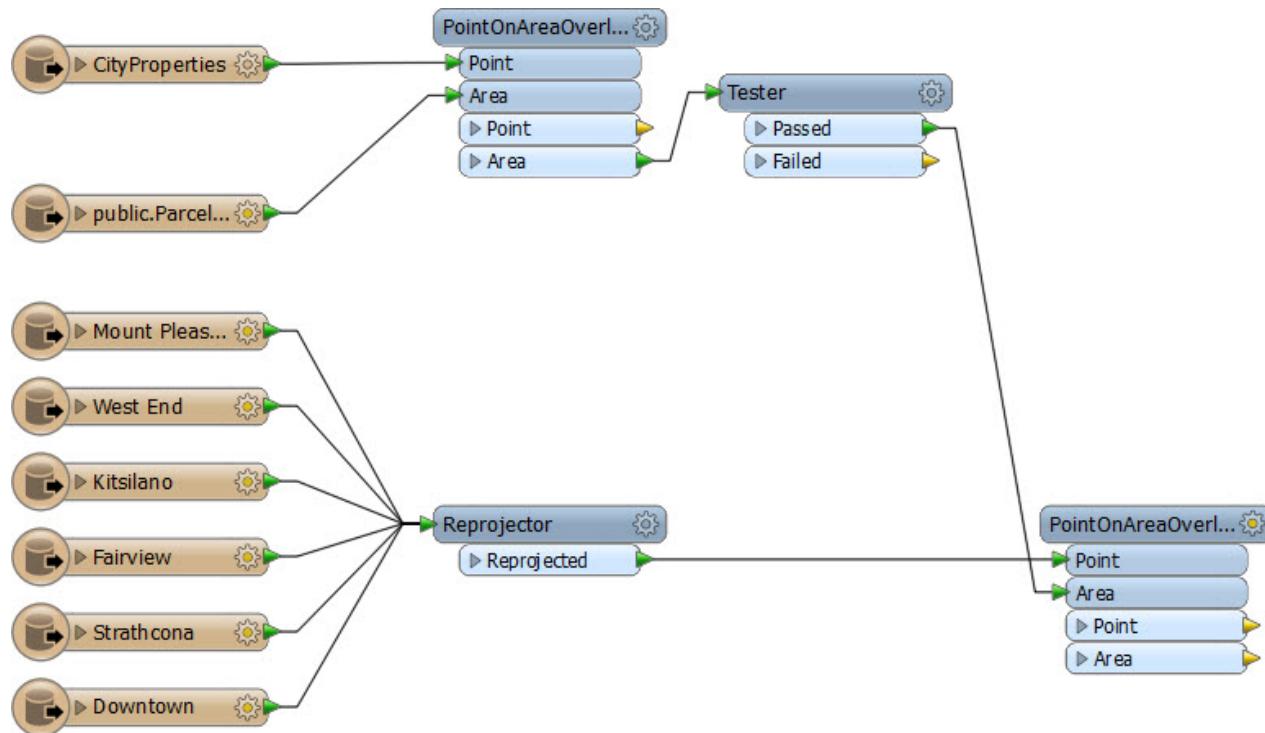
7) Add Reproductor

One issue to still overcome is that the data from the Excel Reader will be in Latitude/Longitude (because that's what the coordinates were saved as) whereas the other datasets are UTM83-10. We'll solve this by reprojecting the data.

Add a Reprojector transformer connected to all of the Excel feature types. Open the parameters dialog and set it to convert from LL83 to UTM83-10:

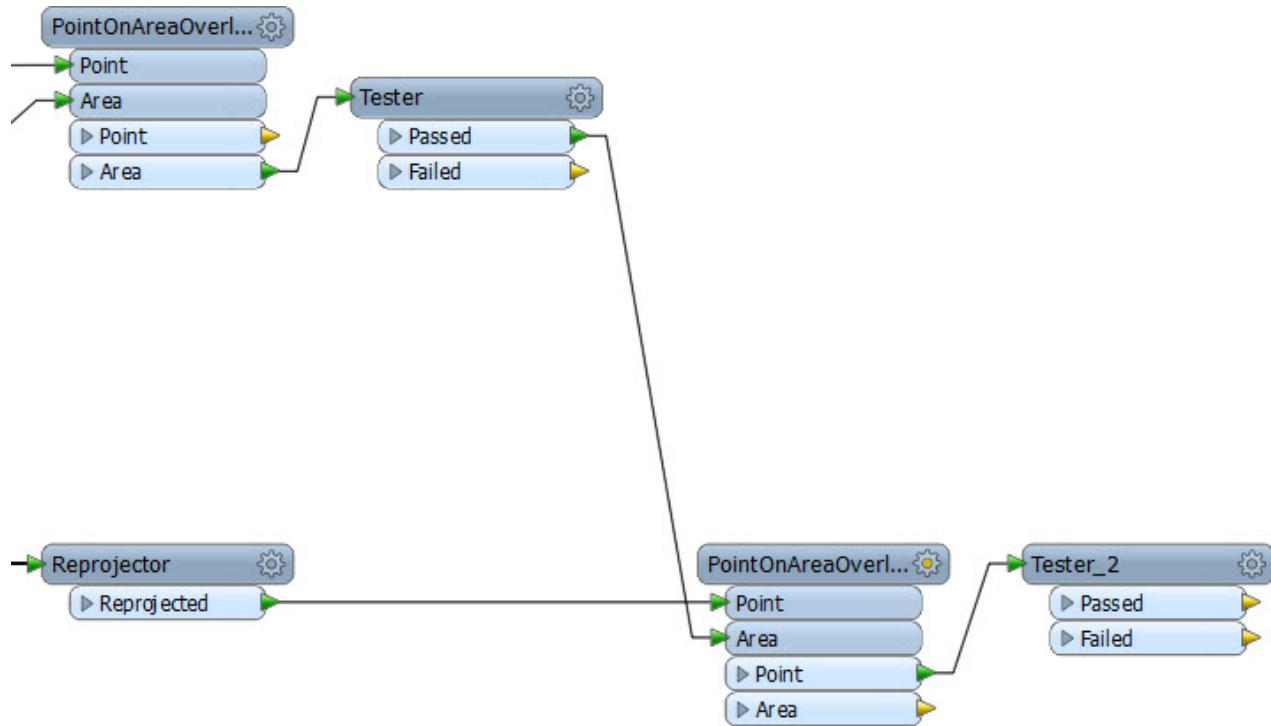
8) Add PointOnAreaOverlayer

Now we just need to determine which of the public art features fall inside one of the city properties. We can do this with another PointOnAreaOverlayer transformer. Add one of these and connect the art features to the Points input port and the Tester:Passed features to the Area port:



9) Add Tester

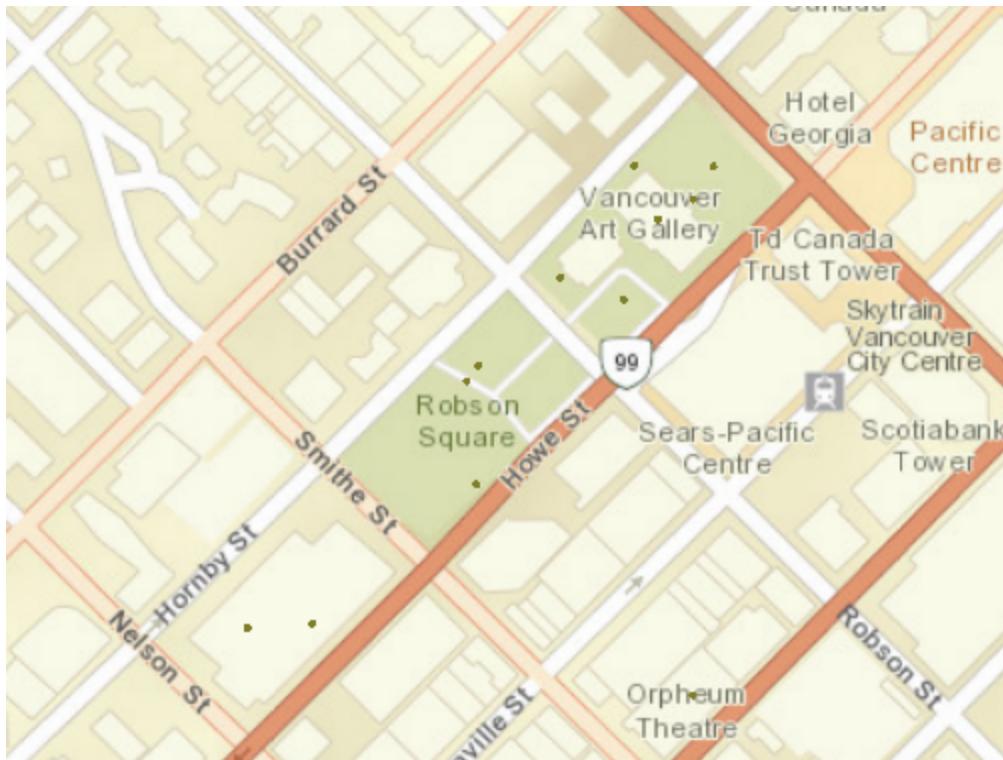
A final Tester will help us filter the art features we need. Add a Tester connected to this new PointOnAreaOverlayer Point output port. Again set it up to test where `_overlaps` is ≥ 1 .



10) Run Workspace

Add an Inspector transformer to the Tester:Passed port. Save and run the workspace.

The output from this will be all public art installations that reside on city property.



ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013



Congratulations! You have now:

- Added Readers to a workspace
- Checked the Reader parameters

Importing Reader Feature Types

A further tool under the Reader menu is labelled Import Feature Types.

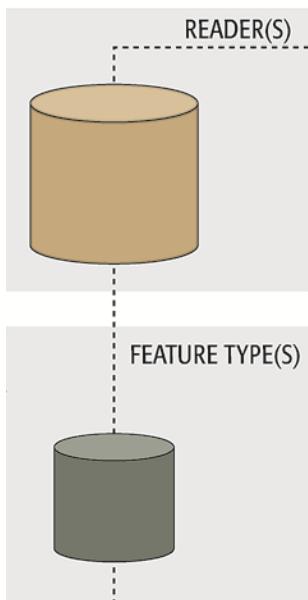
To understand importing feature types, it's important to recognize that a dataset is made up two parts. One part of a dataset is obviously the data itself, but another - separate - part is the dataset schema.

What the import tool does is read the schema of a dataset (not the data) and use it as the template for a set of feature types in a workspace.



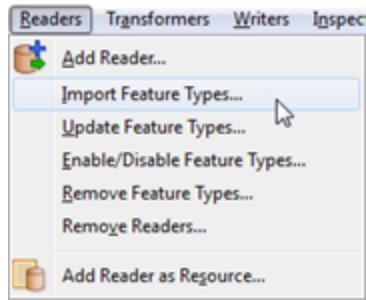
Mr. E. Dict says: "Whenever I create a new legal contract for a client, I copy previous contracts. I don't put the same clauses (data) in there, but they do have the same structure (schema). It's like using a template, and that's exactly what the Import Feature Types tool does."

One reason to import feature types is to read extra tables from a database, or when a source dataset itself is updated. For example, a user has a workspace reading from a spatial database. It only needs to read from a single table (roads), so there is a single Reader representing the database, and a single Feature Type representing the table.

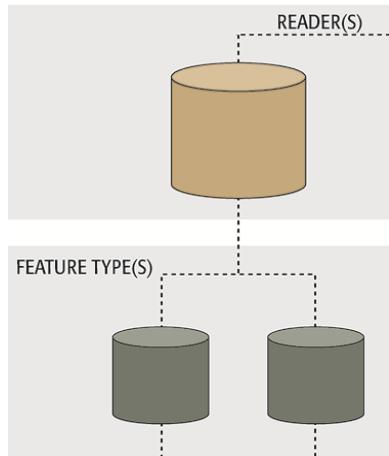


However, at some future point the user decides the workspace also needs to read from a second table ('rail').

The simplest solution is to use the command **Readers>Import Feature Types...**

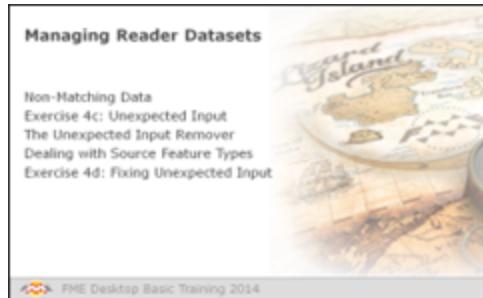


The import tool will take the schema definition of the database table 'rail' and add it to the workspace.



The same process could be used on a file-based dataset when, for example, the dataset was created with just a 'roads' layer, but was later updated to include a 'rail' layer. In that case the FME user would import that layer to ensure the data is read.

Managing Reader Datasets



It's very simple to use a dataset parameter to change the data being read; but care must be taken to avoid certain pitfalls.

Non-Matching Data

As we have seen, each Reader in a workspace contains a parameter with which the source dataset can be selected. Using this parameter it's possible to change a Reader to read a different source dataset (or datasets) than those selected when the workspace was created initially.

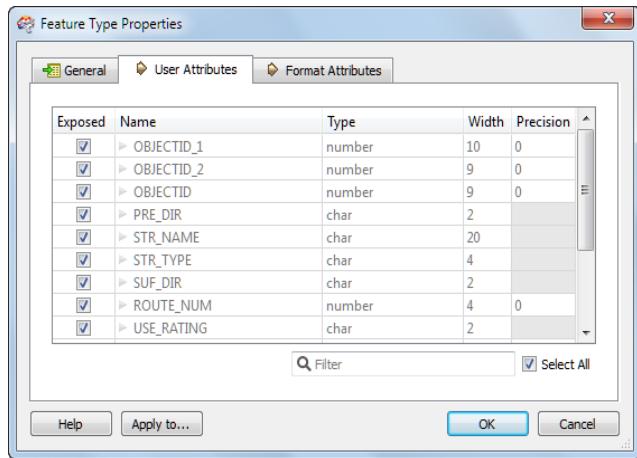
This is useful because it allows a workspace author to create a prototype workspace, test it on a single source dataset, and then adjust the workspace to read all source data.

However, it's important to remember that the original dataset establishes the basis for the Reader schema definition. Problems arise if subsequent datasets do not conform to this original schema.

Datasets with Different Attributes

As already noted, a schema definition includes user attributes. If a Reader is defined with one set of attributes, then made to read datasets with different attributes, there is an obvious conflict.

The problem is that user attributes are fixed in the reader schema, which is why they are grayed-out in the properties dialog.



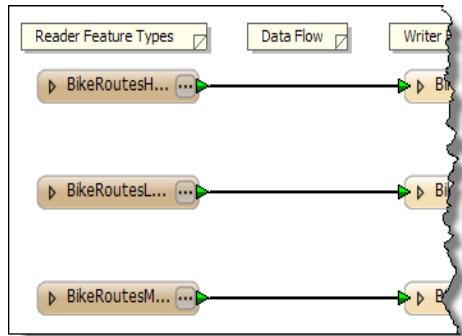
In fact the attributes are read and attached to incoming features, but since they won't appear on the writer schema, they will not get written to any output dataset (that requires a dynamic writer).

Furthermore, since the attributes don't appear in the workspace itself, there is no way for the workspace author to manipulate them with transformers.

Datasets with Different Feature Types

Similarly, a schema includes the definition of a source dataset's feature types.

If the Reader is defined with one set of feature types, then switched to read datasets with different feature types, then that is another source of conflict.



Again, the problem lies in having a fixed source schema, where feature types are not automatically updated when new data is read.

In this scenario the data is automatically discarded; i.e. Reader feature types defined in the workspace act as a type of filter through which incoming data must pass.

Unexpected Input

Exercise 4b Unexpected Input

Scenario	FME User
Data	Contours (Esri Shape)
Overall Goal	Read contour data
Demonstrates	Unexpected Input
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise4b-Complete.fmw

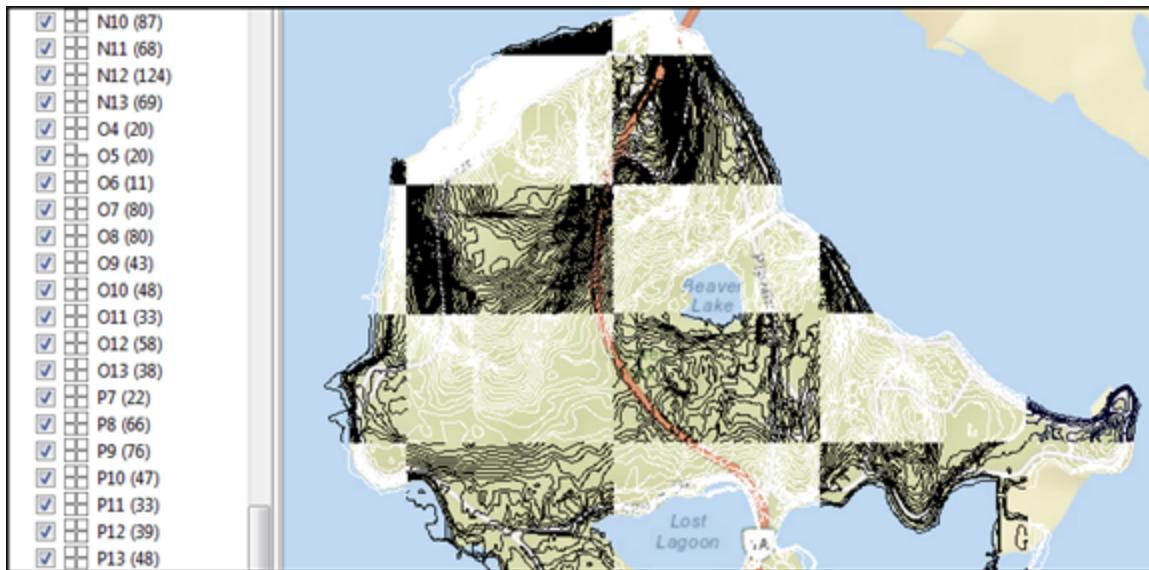
This example merely reads a file of contour data, but demonstrates a potential problem when changing the source dataset parameter.

1) Inspect Data

Start the FME Data Inspector and open the datasets:

Reader Format Esri Shape
Reader Dataset C:\FMEData2015\Data\ElevationModel\Contours*.shp

In other words, select all of the Shape files in that folder.



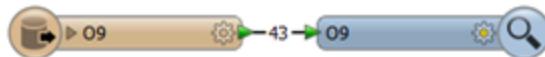
ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

You'll see that the contour data - because of its size - has been split up into a grid of files.

2) Create Workspace

Start FME Workbench and begin with an empty workspace. Select **Readers > Add Reader** from the menubar.

When prompted enter the format and dataset for ONE of the contour files, for example O9.shp.

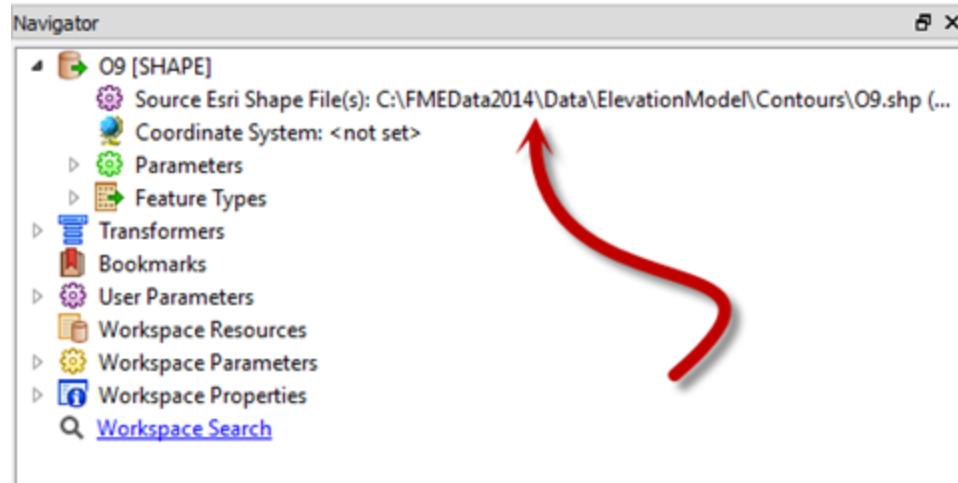


Add an Inspector and run the workspace to ensure it works as expected.

3) Change Dataset

Now we'll assume that someone else has asked you to translate a different file, for example L2.shp.

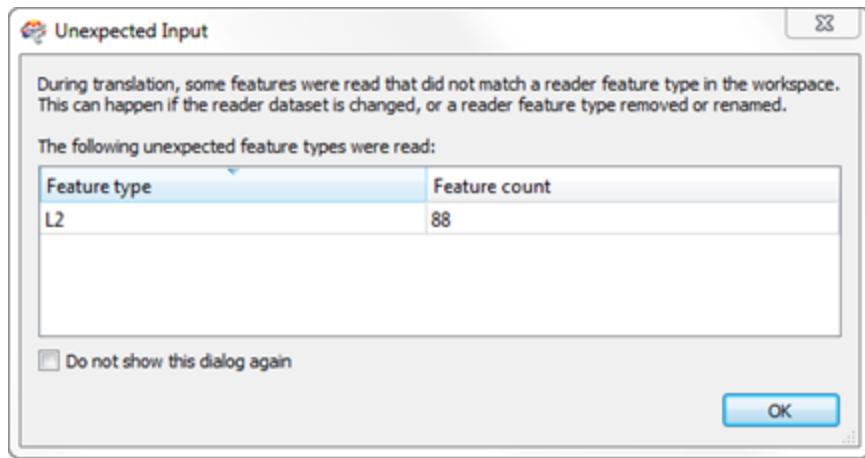
So, locate the source dataset parameter in the Navigator window:



Double-click the parameter and a Source Shape File dialog opens up. Use the browse button to select L2.shp.

4) Run Workspace

Run the workspace. You will get a dialog pop up reporting that there was "Unexpected Input."



What this means is that you've tried to read a layer of data that is not defined in the workspace.



Congratulations! You have now:

- *Modified the Reader dataset parameter*
- *Discovered a potential problem of modifying the Reader dataset parameter!*

This example illustrates how a workspace created to process one dataset may not be suitable for all datasets of the same format, even when that dataset is a simple update to the original.

The Unexpected Input Remover

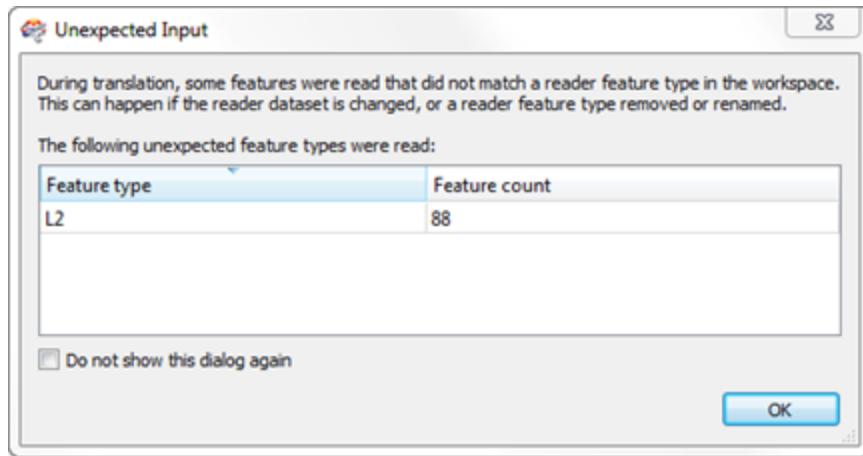
Every time FME reads a dataset, it checks the feature types inside that dataset to ensure that they are all defined within the workspace schema. If there are feature types that exist in the dataset, but do not exist in the workspace, then features are classed as unknown and filtered out by a function called the Unexpected Input Remover.

```
|STATS |Unexpected Input Remover[TestFactory]: Tested 4 input features -- 0 features passed, 4 features failed.
```

The actions of the Unexpected Input Remover are reported in the Log file and through a dialog that opens at the end of a translation.

Recheck the Unexpected Input dialog from the previous example. FME dropped a number of features from the translation because their feature type was unknown in the workspace.

Notice how each feature type mentioned 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.

Remember, the feature types in a dataset do not necessarily match those defined in the workspace. 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.

Therefore this dialog is considered a reminder rather than an error, and is not always an indication something has gone wrong.

Unexpected Input and Dataset Type

Remember, in general terms, there are two different types of dataset: file-based and folder-based. Just as they are read slightly differently, the Unexpected Input Remover operates on them in the same way.

File-Based

File-based datasets are the easiest to understand. If there is a layer in the file that is not represented on the canvas then FME treats the data as "unexpected" and the Unexpected Input Remover will drop it from the translation.

For example, if I set up my translation to read the layers Roads, Rail, and Rivers, then a new layer of data is added to the source dataset (or I switch the dataset to a different one) then new layers - like Vegetation, Buildings, etc - will be dropped.

Folder-Based

Folder-based datasets are when the layers are stored as separate files in a folder; for example a Shape dataset with multiple shp files.

In this case, if I have set up a translation to read one Shape file, then change it to read a different one, then the data is unexpected - because it is a different layer, one not represented by a feature type. This is the exact scenario in the previous example.

In fact, this most often becomes 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.

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

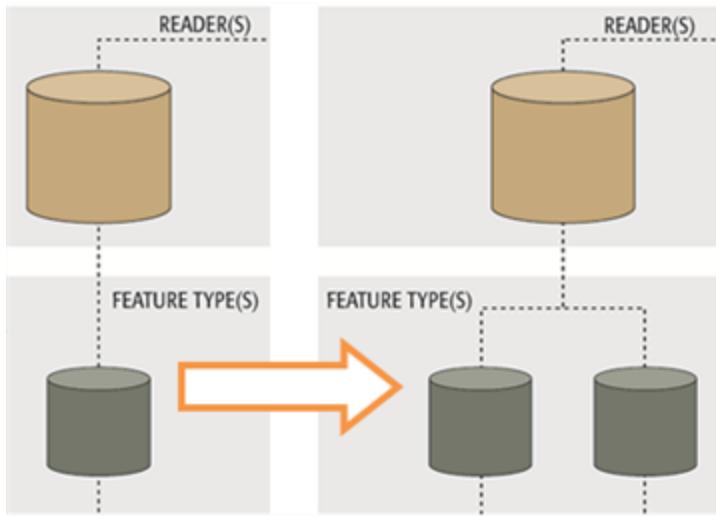
In other words, allow unexpected feature types to pass through an existing one.

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.



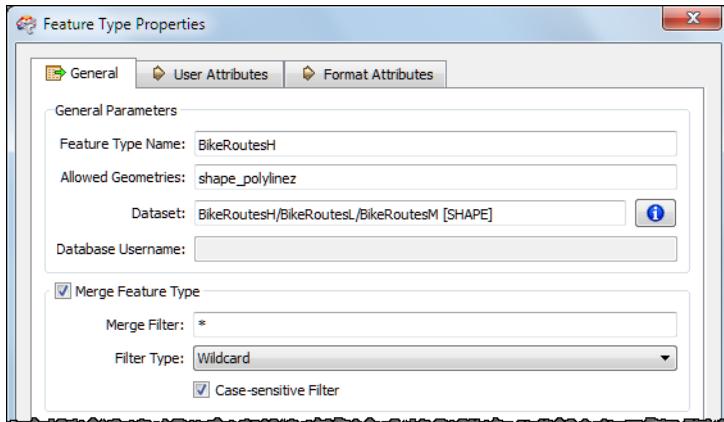
Merge Parameters

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.

Merge Parameters are available in each reader feature type for this purpose.

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.

The first action is to select a feature type through which to merge the data and open its Feature Type Properties dialog.

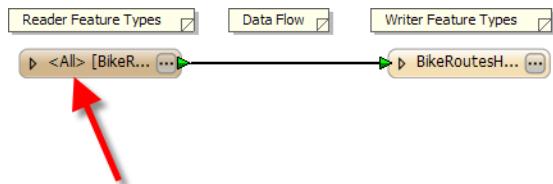


Once the Merge Feature Type option is checked a Merge Filter can be set. This is used to define exactly what incoming feature types are allowed to merge.

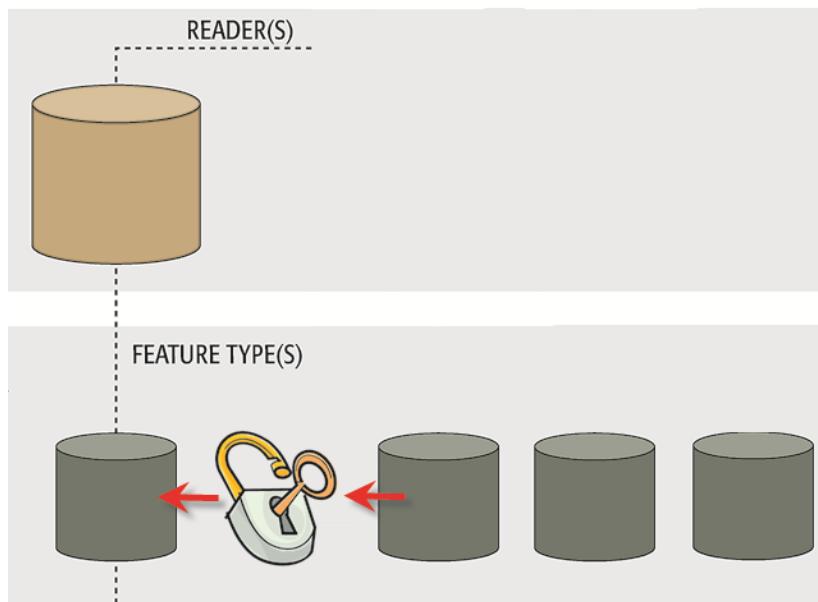
The filter type can be a wildcard or a regular expression.

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.



A Feature Type merge unlocks a workspace and allows to pass any feature types that are not currently defined.



Fixing Unexpected Input

Exercise 4c Fixing Unexpected Input

Scenario	FME User
----------	----------

Data	Contours (Esri Shape)
Overall Goal	Read contour data
Demonstrates	Unexpected Input
Starting Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise4c-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise4c-Complete.fmw

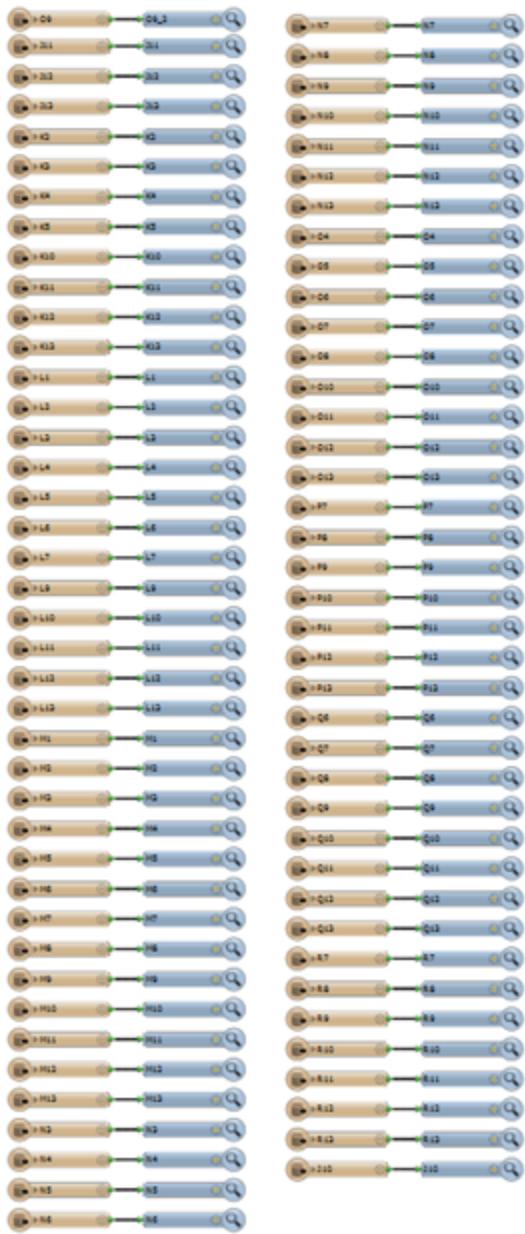
This example fixes, in a simple way, the problem of reading multiple files in a workspace.

1) Start Workbench

Start Workbench and open the workspace from the previous example (if it is not already open).

Remember, when this is run the Unexpected Input Remover pops up. That's because you are trying to read a layer/file called L2, when no L2 feature type exists in the workspace.

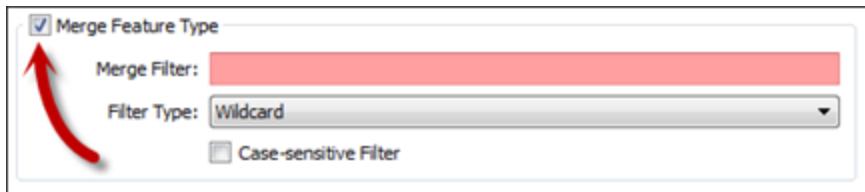
We could resolve this by using the Import Feature Type tool, but to handle all of the files in this way would result in a translation that looked like this:



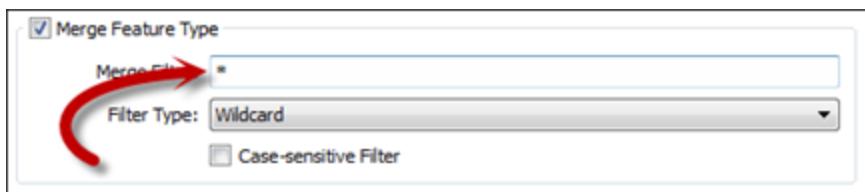
So, we'll fix this translation a better way, by unlocking the single feature type we have to let any of these files past.

2) Set Merge Filter

Open the properties dialog for the Feature Type. In the middle of the dialog select the option Merge Feature Type:



In the Merge Filter field we can define which files we will allow to pass. Because there isn't really a common theme to the file names (or not one that we can easily define without using a regular expression) simply enter an asterisk in that field:

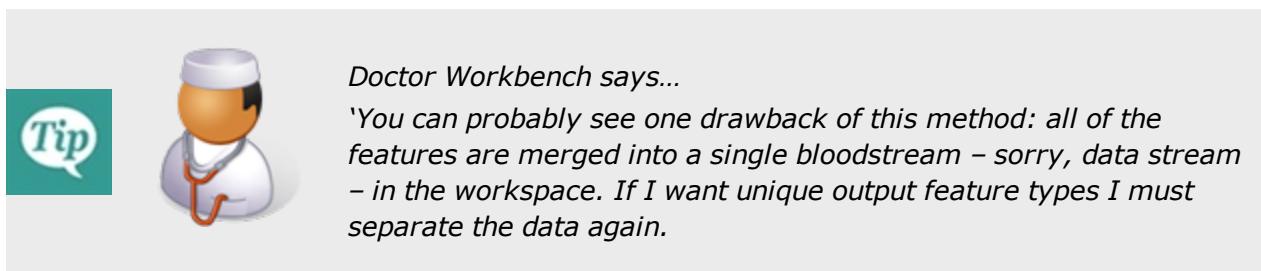


This means we'll allow any and all files to pass. In fact, when you press OK, the feature type object is updated to reflect that fact:



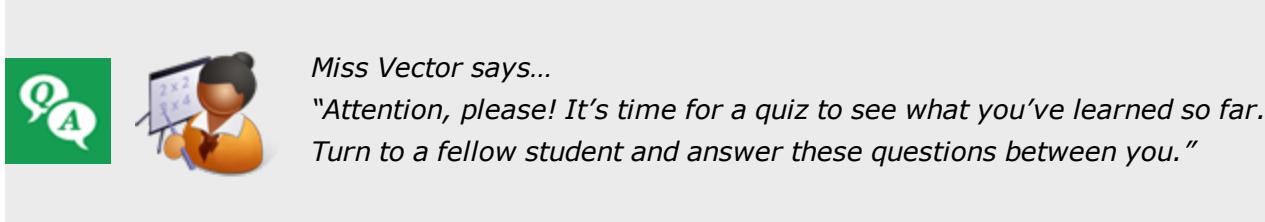
3) Run Workspace

Now when the workspace is run the chosen data will be allowed to pass. In fact, you should be able to select any - or several - of the contour Shape files and they will all pass.





In this case it's not too much of a problem, as the data is all the same type just tiled separately. But you may want to reflect on when the Import Feature Type tool is a better option.



What problem does the Import Feature Types and the Merge Filters options solve?

- 1) Reading datasets with different attributes
- 2) Reading datasets with different Feature Types
- 3) Reading datasets with different formats
- 4) Reading datasets with different geometry types

What does the Import Feature Types option do?

- 1) Adds missing Feature Types
- 2) Lets any Feature Type pass
- 3) Turns off the Unexpected Input Remover
- 4) Updates the attributes on an existing Feature Type

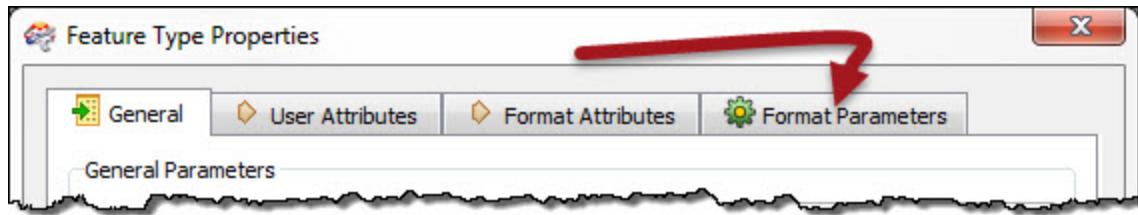
What does merging filters do?

- 1) Adds missing Feature Types
- 2) Lets any Feature Type pass
- 3) Turns off the Unexpected Input Remover
- 4) Updates the attributes on an existing Feature Type

What do you think are the relative benefits to the Import Feature Types and Merge Filters functionality? That is, what advantages and disadvantages does each have?

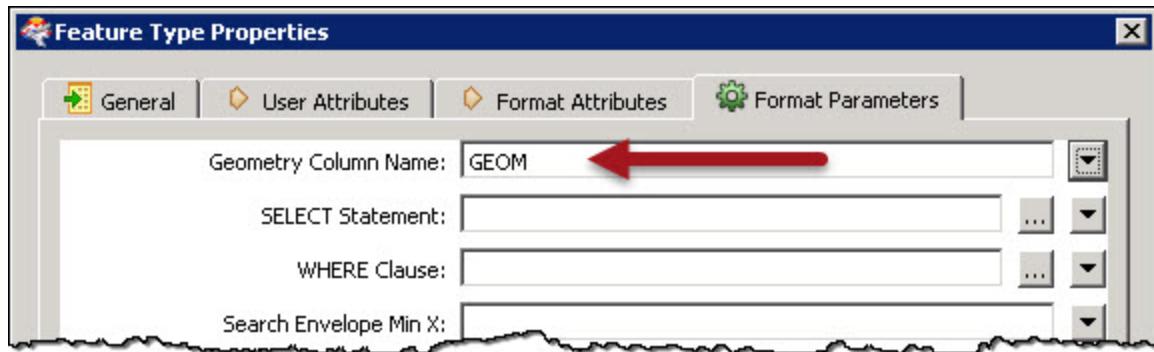
Reader Feature Type Parameters

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 under the Format Parameters tab in the Feature Type Properties dialog:



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, when reading an Oracle database (here on a Windows Server OS) there is a Feature Type parameter that defines the name of the geometry column to read:

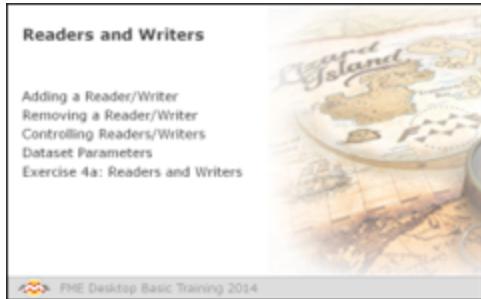


This is a Feature Type parameter because each table might have a different geometry column name, therefore each feature type needs a way to define it. Conversely, there is no password parameter at the feature type level because authentication applies to the entire database, not the individual tables. Those would be Reader parameters.

So feature type parameters provide a degree of individual control over reading different layers or tables.

NB: Not all feature types need a set of parameters, so the Format Parameters tab is not always present.

Writers



Writers are the main components that write data in FME.

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).

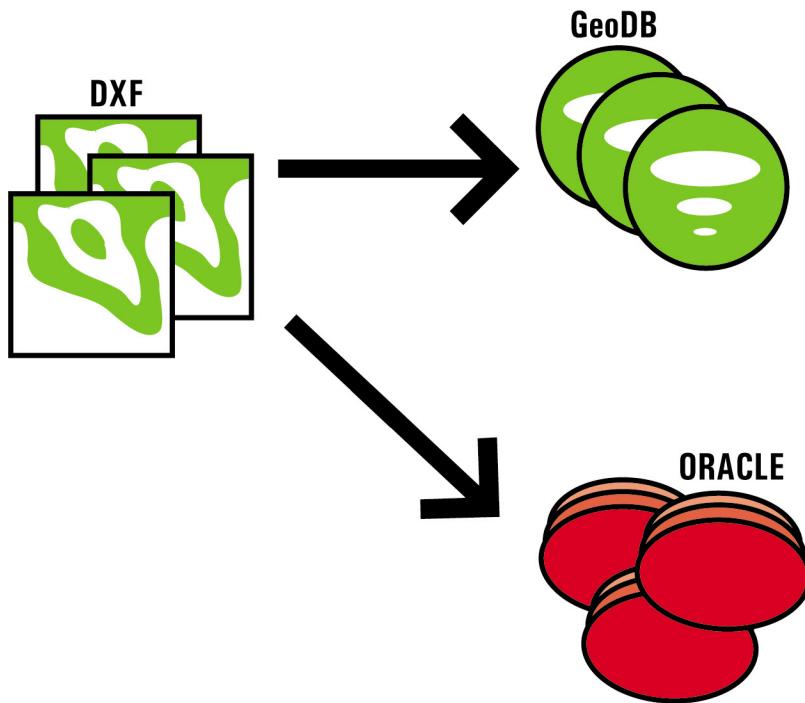
When a workspace is generated with the Generate Workspace dialog then it is created with just a single Writer. However, this does not mean the workspace is forever limited to this. Additional Writers can be added to a workspace at any time, any number of formats can be used, and there does not need to be an equal number of Readers and Writers.

Adding a Writer

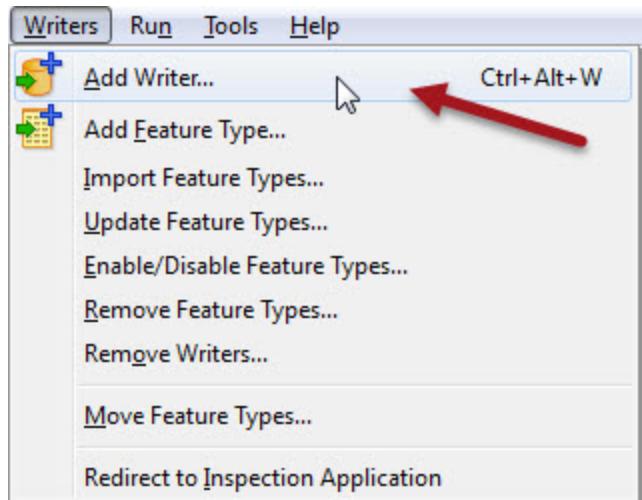
Adding a Writer to a workspace is a common requirement. There are several reasons:

- 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

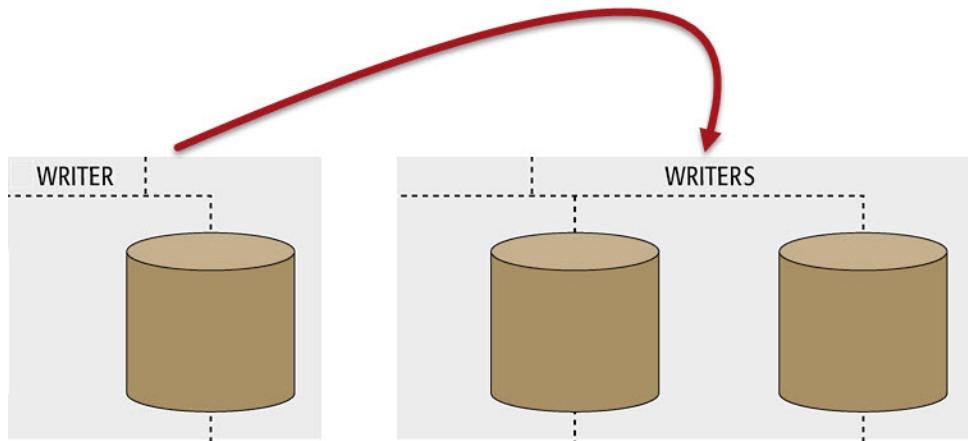
Therefore the need to write multiple formats of data – such as Geodatabase and Oracle – requires multiple Writers.



Additional Writers are added to a translation using **Writers > Add Writer** from the menubar.

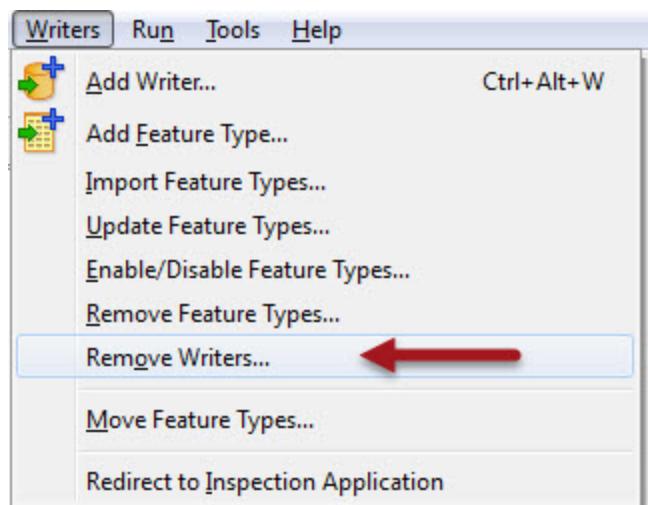


Adding a Writer has this effect on the hierarchy diagram:



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.



Controlling Writers

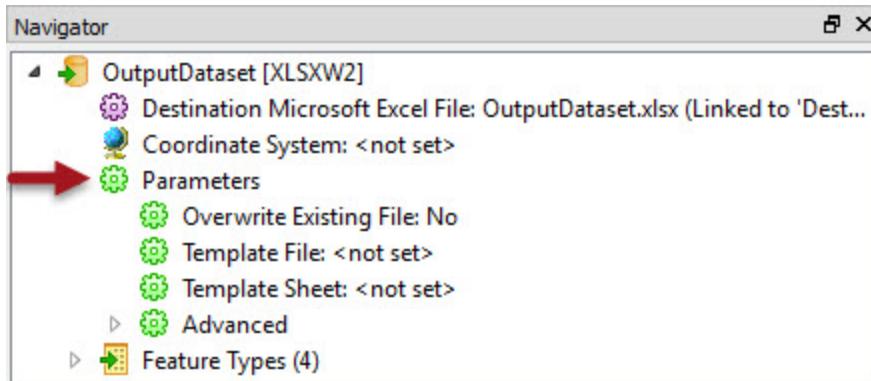
Writer parameters are those that control how data is written.

Because these parameters refer to specific components and characteristics of the related format, no two formats will have the same set of control parameters.

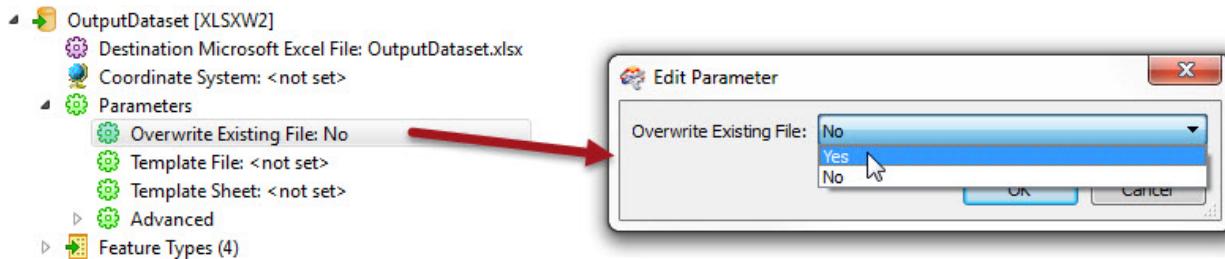
Writer Parameters

Writer parameters are shown and set in the Navigator Window. You expose the list of parameters by clicking on the expand arrow icons to the left of the Writer.

For ease-of-use, basic parameters are listed first, and then advanced parameters are hidden under a separate section.



In general, basic parameters are the ones used most often; advanced parameters a little less so. To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.



Writer Feature Types



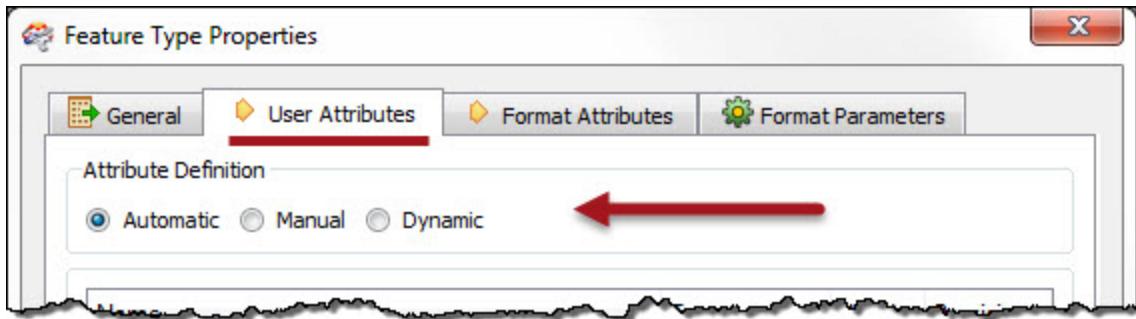
Writer Feature Types are objects on the Workbench canvas that define the schema of the data being written. Most importantly they define two specific things: they define the layers being written to a destination dataset and the attributes that those layers are to possess.

Let's first look at attribute definitions.

Writer Feature Type Attributes

Each Writer feature type has a tab labelled User Attributes. This tab allows you to define the attributes that will be associated with this feature type. Attributes can be defined by either editing an existing feature type (as we saw in the chapter on Data Transformations) or can be defined when a new feature type is added to a workspace (as we will see shortly).

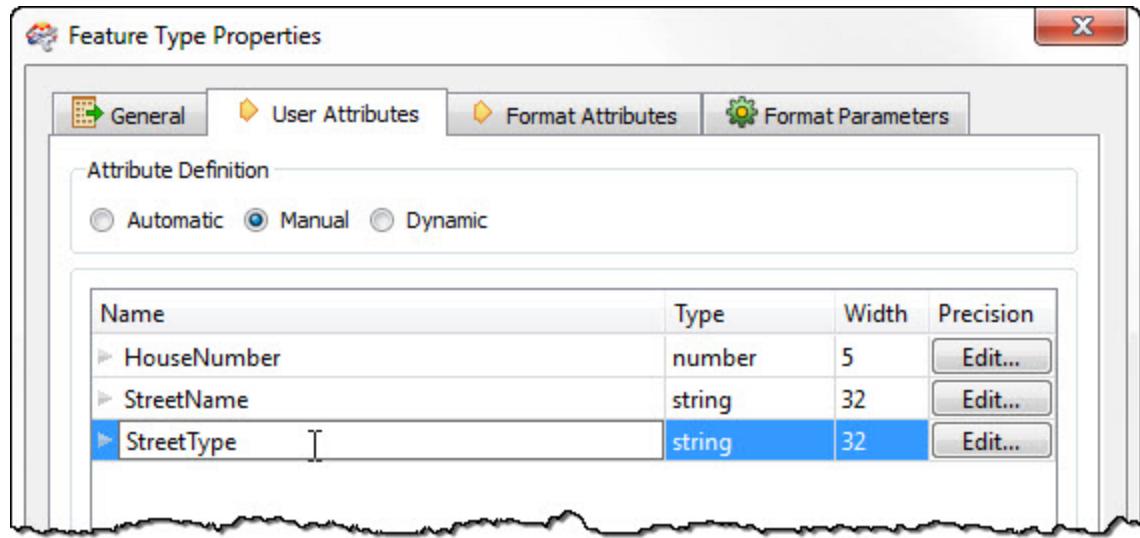
There are three options as to how attributes can be defined:



These options are Automatic, Manual, or Dynamic. We'll look at Manual definitions first, since they are the easiest to understand.

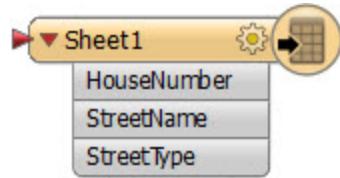
Manual Attribute Definitions

A manual attribute definition is simply when the workspace author manually enters attribute names, and manually selects the attribute types, like so:



Here, for example, the author is defining an address table. So far they have entered the definitions for two attributes, and are just entering a definition for the third.

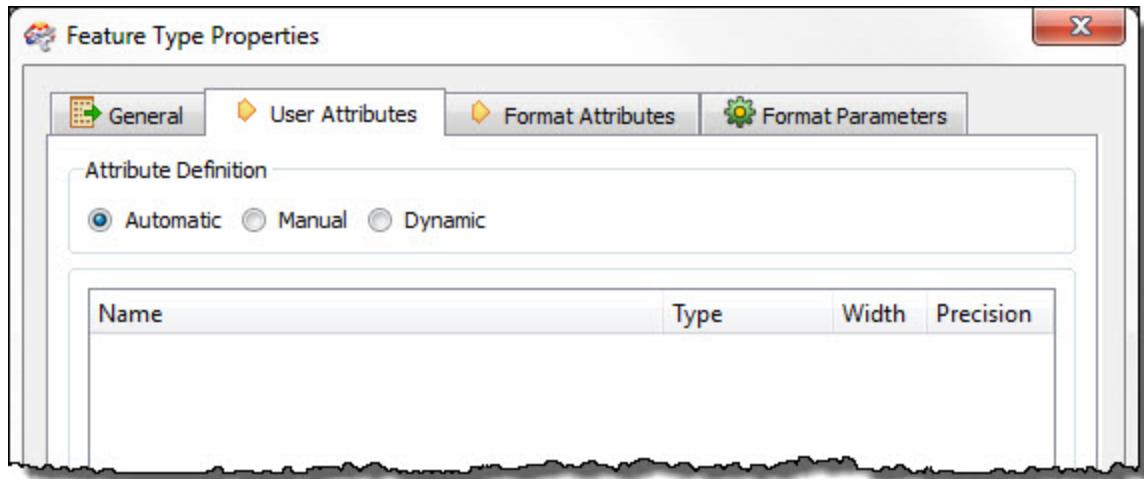
In the Workbench canvas, the feature type would look like this:



Automatic Attribute Definitions

An automatic attribute definition is when Workbench itself automatically defines the list of attributes, depending upon what Reader feature types are connected.

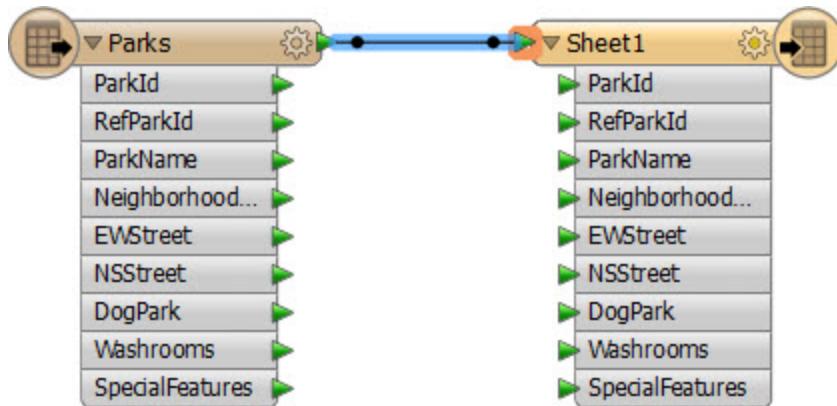
Here – when the user chooses the Automatic option - the attribute definition fields are left empty and uneditable, like so:



Similarly, on the canvas the feature type is also empty initially:

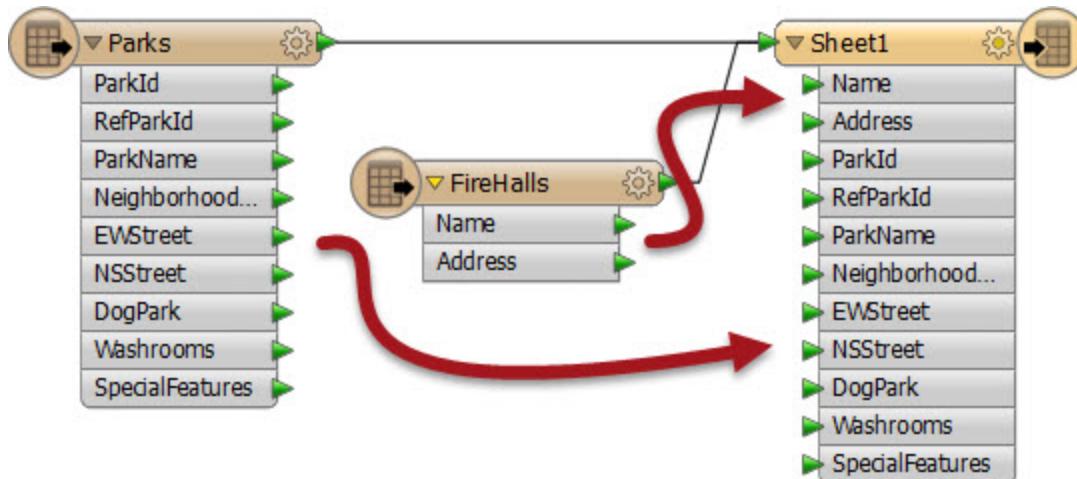


Now, what makes the definitions automatic is the action that occurs when a Reader feature type is connected:



Notice that all of the attributes defined on the Reader feature type are now copied over to the Writer feature type automatically.

Because this operates automatically, if the Reader feature type is removed, and another one connected, the list of attributes on the Writer feature type will be immediately updated to match. Or, if a second Reader feature type is connected, then the list of attributes updates to become a union of the two types:



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 the Reader dataset parameter to read data 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

Specifying dynamic attributes also causes the Dynamic Properties option (in the General tab) to be selected:



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! 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.



To clarify, the "automatic" setting takes its attribute definition from whatever is connected to it in the workspace. If the Source Dataset parameter is changed, it will have no effect. The "dynamic" setting is different. If the Source Dataset parameter



is changed, a dynamic schema will (by default) take its attribute definition from whatever source data is being read.

Adding Feature Types

Having looked at attribute definitions, now let's look at how to manage the feature type objects themselves.

As we have already seen, the Writer schema defines "what we want" as a destination dataset. So if you want to write data to a particular layer, in a particular dataset, then you have to ensure that the layer is defined as a feature type in the canvas.

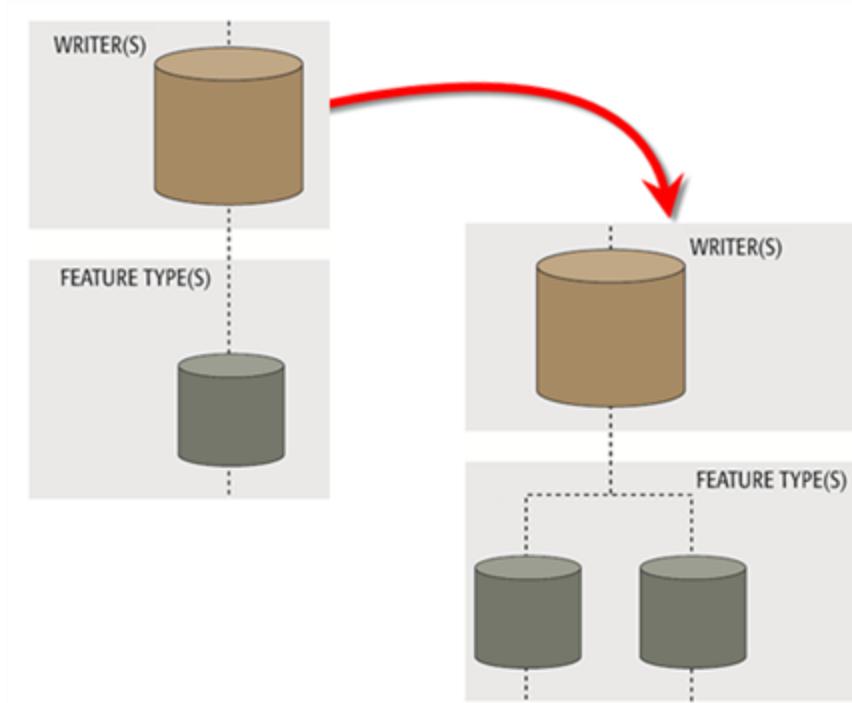
In some cases this happens automatically. When you generate a workspace the source dataset (s) you choose are scanned and all of the layers within get matching feature type objects in the workspace - both on the Reader and the Writer.



When FME adds Writer feature types as part of a new workspace, those feature types will get a fixed (manual) attribute definition that is taken from the selected source dataset. These attributes can then be manually edited, as demonstrated in the chapter on Data Transformation.

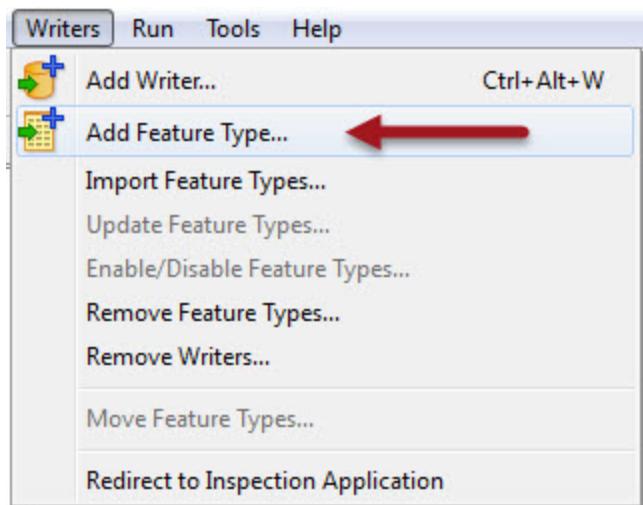
However, in most cases you need to manage Writer feature types separately, because "what you want" is not the same as "what you have". For example, perhaps you want to divide an incoming layer of data into two layers in the output dataset (in which case you will need to add a new Writer feature type).

Adding a Writer feature type manually has this effect on the hierarchy diagram. Your output was designed to have only a single layer; now it will have two.



Adding Feature Types to an Existing Writer

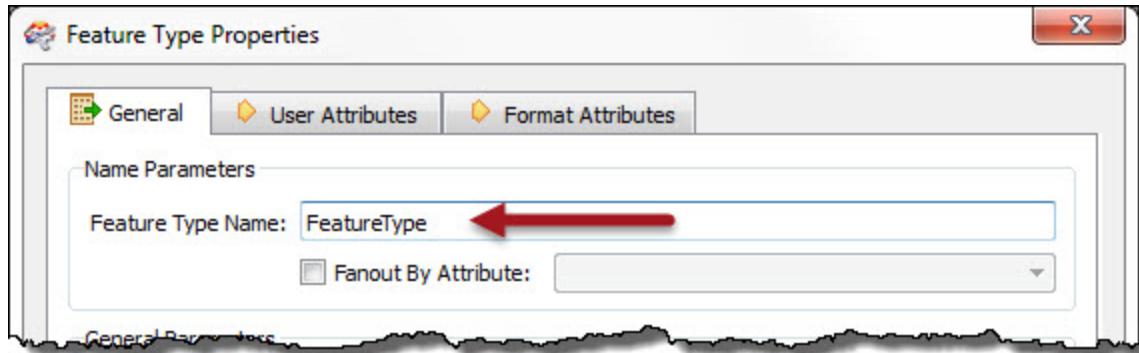
Feature Types can be added manually to a writer using **Writers>Add Feature Type** on the menubar.



At least one writer must exist in the translation hierarchy; else this option will be greyed out.

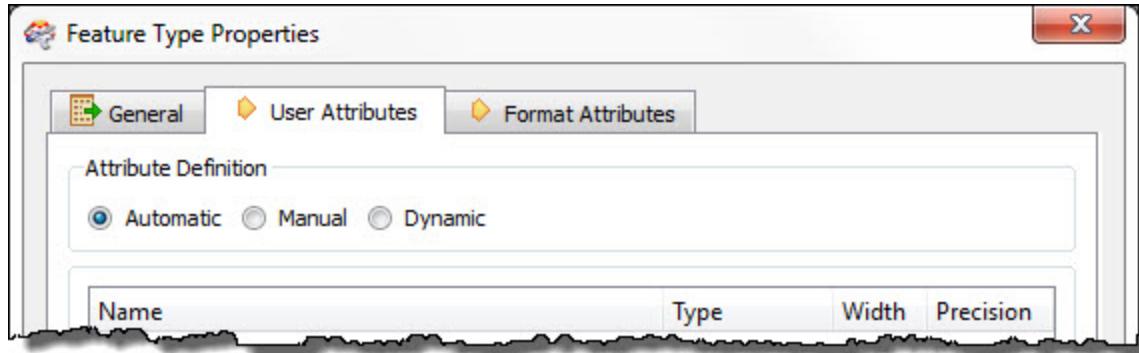
Choosing to add a feature type adds one to the translation, and then causes the Feature Type Properties dialog to appear in order to edit the feature type properties.

As when editing the schema, the General tab can be used to define the new feature type's name.



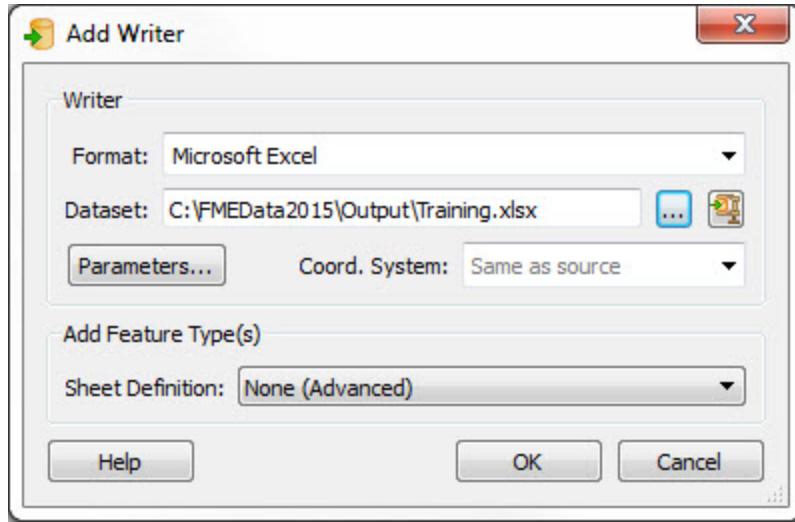
Remember, with format-specific terminology, instead of "Feature Type" this dialog's labels may state "Feature Class," "Layer," "Sheet," or whatever terminology is specific to the format of data you are writing.

As usual, the User Attributes tab can be used to define the new feature type's attribute schema.

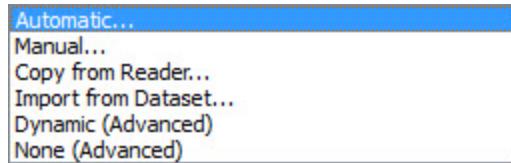


Adding Feature Types to a New Writer

When a new Writer is being created the option is provided to add new feature types:



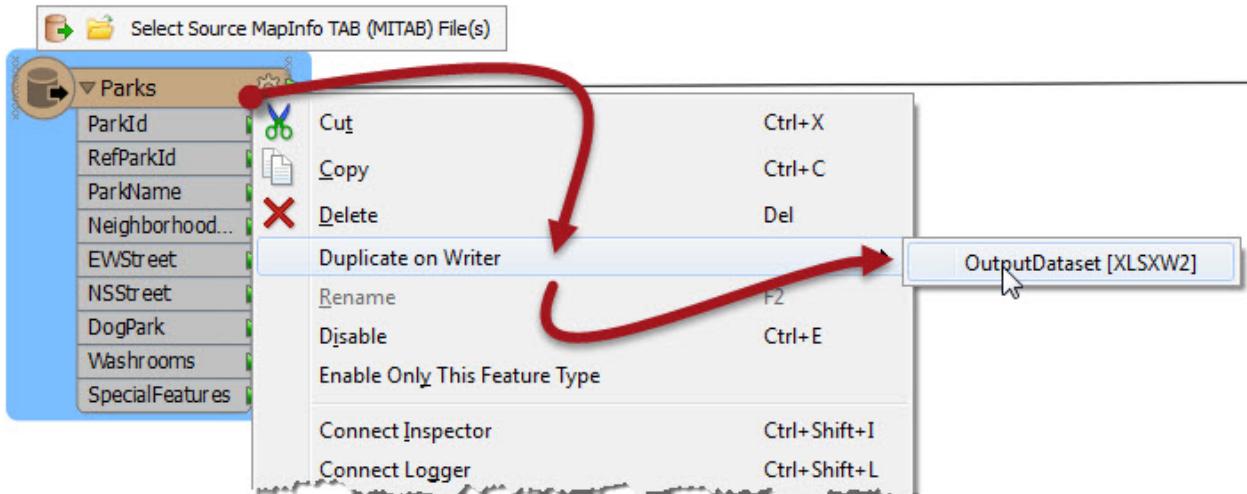
Addition of feature types can be completed in a number of ways:



- 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
- Copy from Reader: Adds a new feature type with the attribute definition parameter set to manual, but pre-defined to the schema of an existing Reader feature type
- Import from Dataset: Adds a new feature type with the attribute definition parameter set to manual, but pre-defined to the schema of an external dataset (just like using the Import Feature Types tool)
- Dynamic: Adds a new feature type with the attribute definition parameter set to dynamic
- None: Does not add any new feature type

Copying Feature Types to a Writer

When adding a Writer it's possible to create a new feature type whose attribute schema is pre-defined to that of an existing Reader. However, if you already have a Writer, you can still carry out the same procedure of copying a Reader schema. This is simply done by selecting the required Reader feature types, right-clicking them, and using the option **Duplicate (on Writer)**.

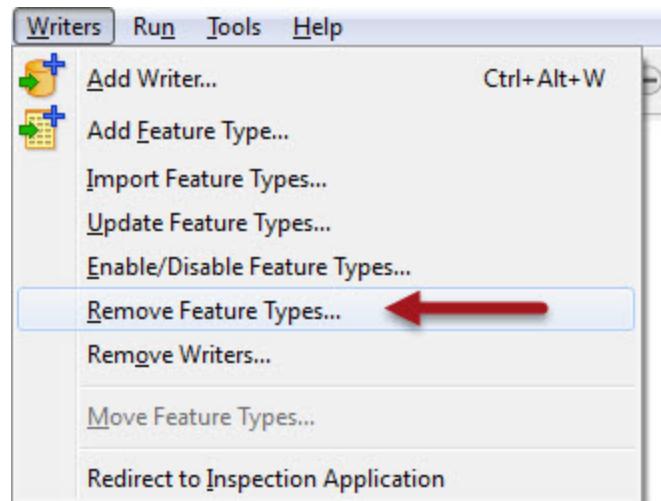


The command causes duplicates of the Reader feature types to be added to the writer, and source/destination feature types to be automatically connected.

Again, at least one writer must exist in the translation hierarchy; else this option will be greyed out.

Removing Feature Types

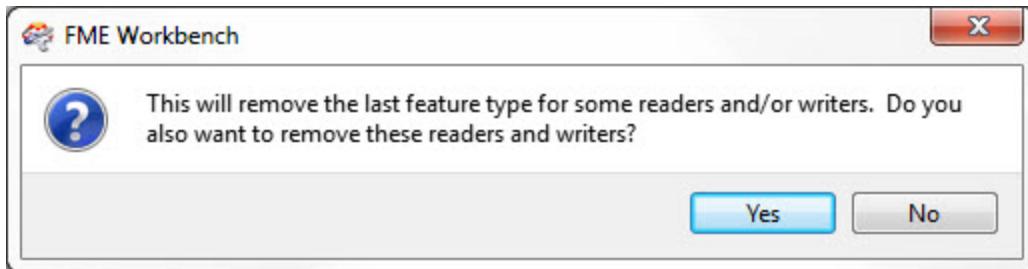
The remove feature types tool is found on the menubar, and simply removes one or more feature types from the Writer.



Another way to do this is to select the feature type on the canvas and press the keyboard delete key.

For example, if there is a destination dataset layer that the user wishes to no longer write, then they would delete the matching feature type from the workspace.

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.



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.

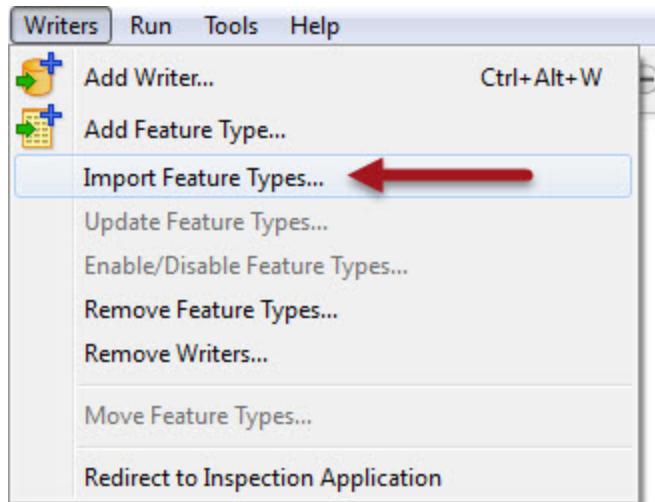
As with Readers, to understand importing feature types, it's important to recognize that a dataset is made up two parts. One part of a dataset is obviously the data itself, but another - separate - part is the dataset schema.

What the import tool does is read the schema of a dataset (not the data) and use it as the template for a set of feature types in a workspace.

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 Reader representing the database, and a single feature type representing the table.

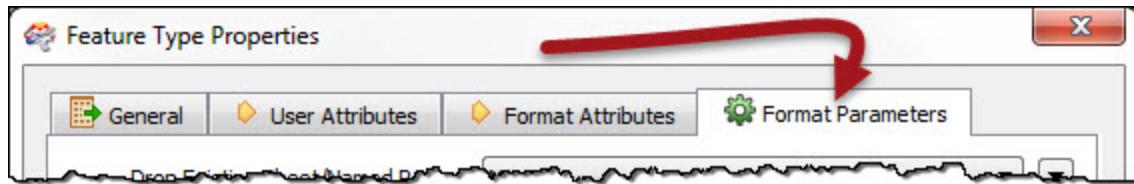
However, at some future point the user decides the workspace also needs to write to a second existing table ('rail'). The simplest solution is to use the command Writers>Import Feature Types...



The import tool will take the schema definition of the database table 'rail' and add it to the workspace.

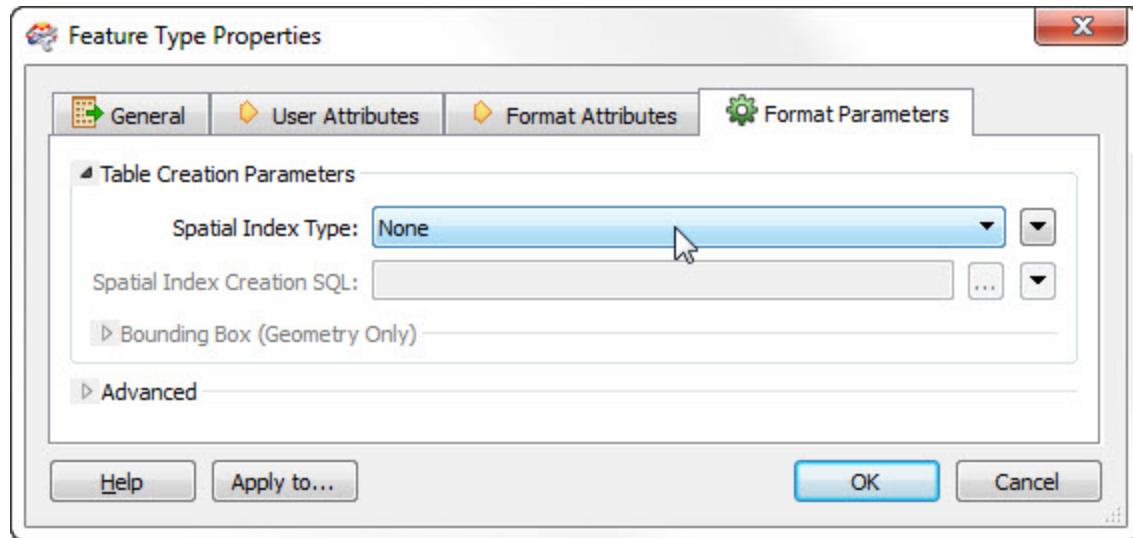
Writer Feature Type Parameters

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 under the Format Parameters tab in the Feature Type Properties dialog:



The important thing is that these (feature type) parameters only apply to a single feature type, whereas a Writer parameter would apply to all feature types.

For example, when writing to a SQL Server database there is a feature type parameter that defines the spatial index to create.



This is a Feature Type parameter because not every table might require a spatial index; therefore each feature type needs a way to define it.

Conversely, there is no password parameter at the feature type level because authentication applies to the entire database, not the individual tables. Those would be Reader parameters.

So feature type parameters provide a degree of individual control over writing different layers or tables.

NB: Not all feature types need a set of parameters, so the Format Parameters tab is not always present.

Feature Types

Exercise 4d Feature Types	
Scenario	FME User
Data	Property Parcels (DWG) Public Art (Excel) City Properties (Esri Shape) Postcodes (Esri Shape)
Overall Goal	Create list of public art on city property
Demonstrates	Managing and controlling Feature Types
Starting Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise4d-Begin.fmw</i>
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise4d-Complete.fmw</i>

This example continues where a previous one left off. In that example you created a workspace to determine what pieces of public artwork are located on city property.

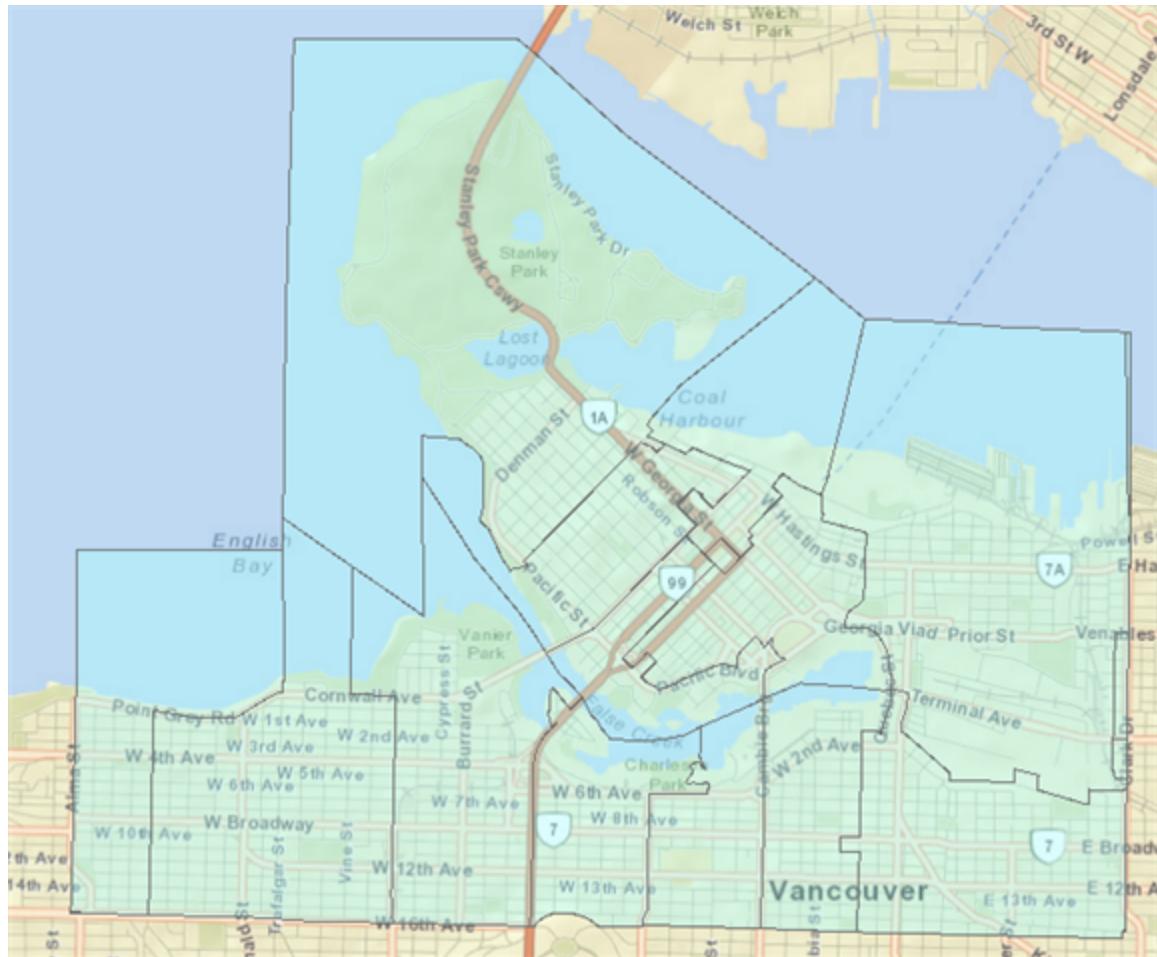
Now the data must be written out so that it can be used by the finance team. However, before that, they have asked that we also divide the data up into separate postcodes.

1) Inspect Data

Start the FME Data Inspector and open the datasets:

Reader Format Esri Shape
Reader Dataset *C:\FMEData2015\Data\Addresses\ForwardSortationAreas.shp*

This dataset contains a set of polygons marking the postcode boundaries for the city.

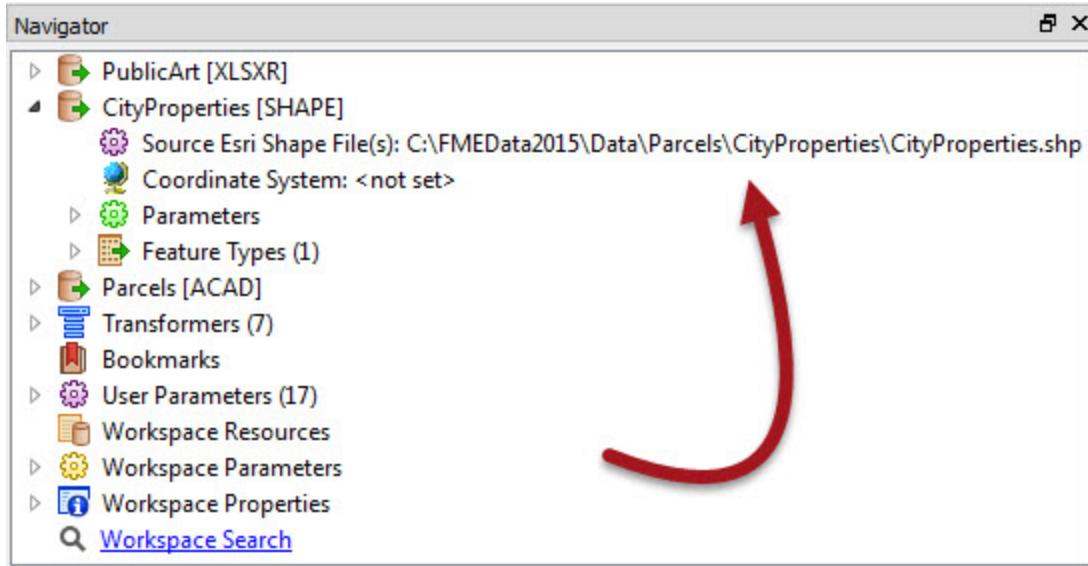


ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

2) Update Dataset Parameter

Start Workbench and open the beginning workspace. To get this data into the workspace we could simply add a new Reader. However, since we already have a Reader to handle Shape data, we can use that, update the source dataset parameter, and simply import the feature type definition.

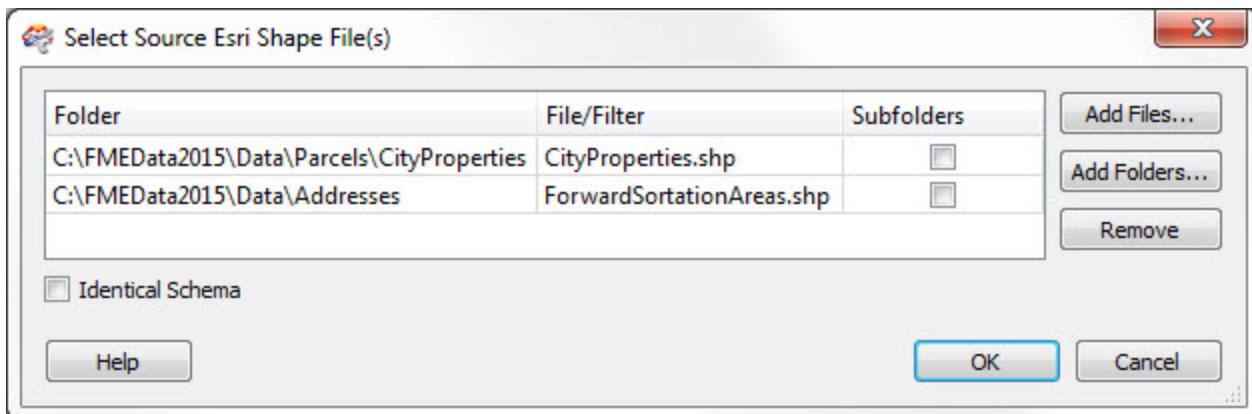
So, firstly locate the source dataset parameter for the Shape Reader:



Double-click the parameter to edit it. If the new postcode Shape file was in the same folder as the CityProperties Shape file, we could simply select both of them with the regular browser. However, since they are not, we must do this a little differently.

Click the drop-down arrow and select Open Advanced Browser.

In the advanced browser, click Add Files, browse to the postal code data (ForwardSortationAreas.shp) and select it:



Click OK and OK again to close these dialogs.

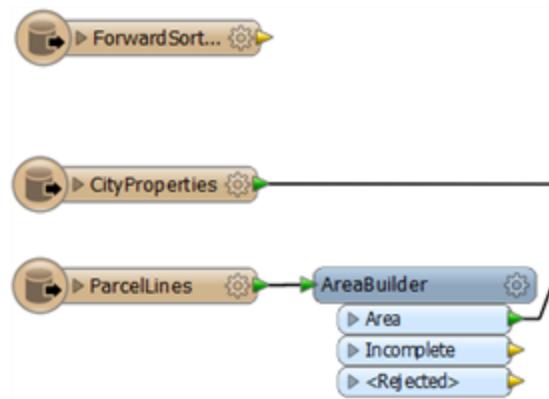
3) Import Feature Type

Now both Shape files will be read when the workspace is run. However, the postcode data will be discarded because there is no matching feature type.

So, select **Readers > Import Feature Type** from the menubar.

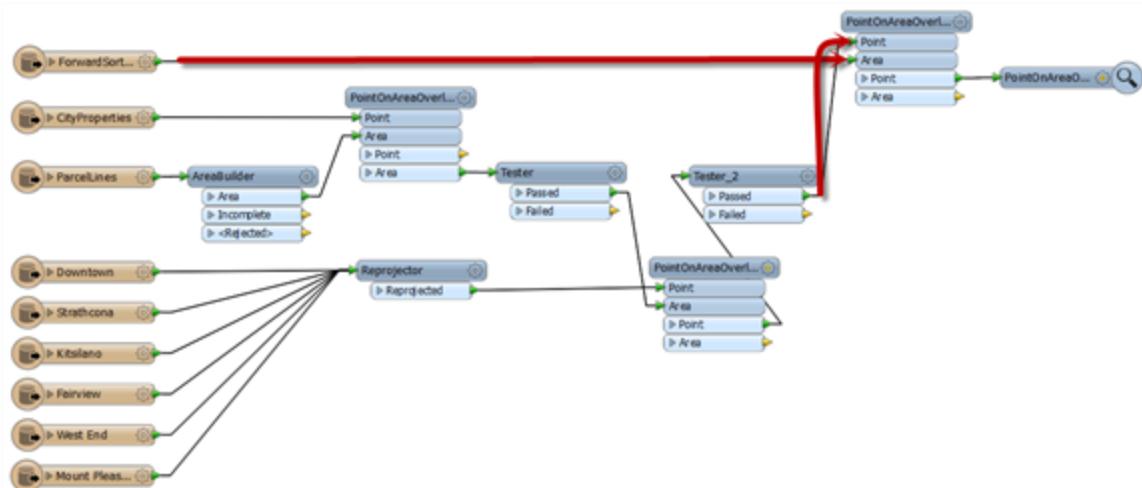
When prompted, select the Shape Reader to import the feature type to. Then, in the next dialog, set the format and dataset parameters, the same way as they usually are.

Now click OK. When prompted to select the types to import, you can click OK again. CityProperties won't be added again because FME recognizes it exists already. However, a feature type for ForwardSortationAreas will be created:



4) Add PointOnAreaOverlayer

Now add a further PointOnAreaOverlayer transformer to copy the postcode info onto matching art features. The ForwardSortationAreas will be the Areas, the output from the final Tester will be the Points:

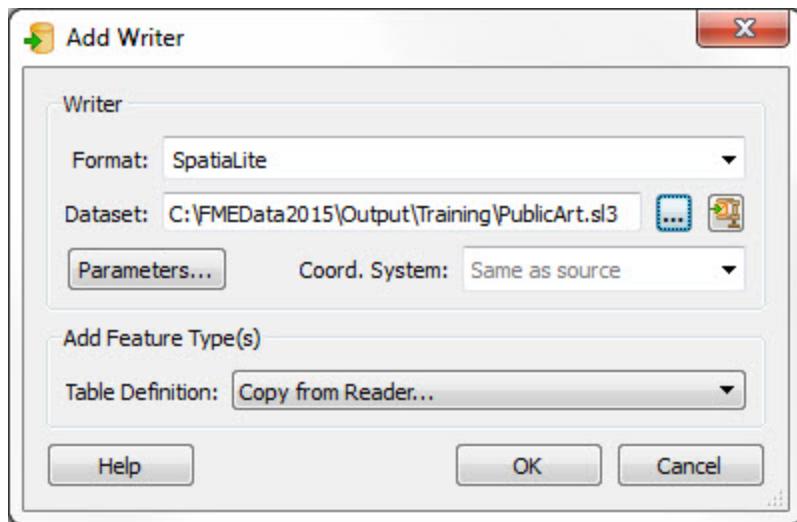


If you add an Inspector and run the translation now, you'll find that there is a postcode (CFSAUOID) attribute added to the public art features.

5) Add Writer

The finance team (who requested this data) would like it as a SpatiaLite database. So add a Writer with the following parameters:

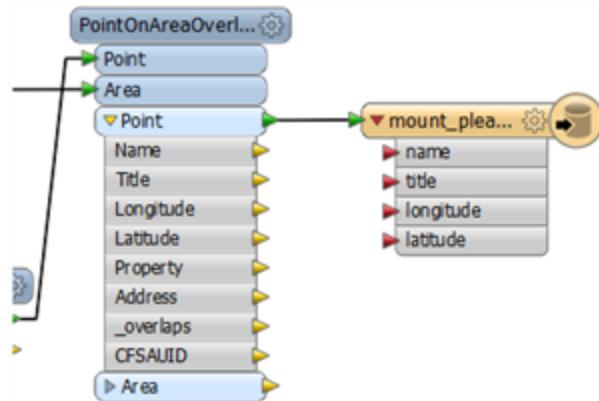
Writer Format SpatiaLite
Writer Dataset C:\FMEData2015\Output\Training\PublicArt.sl3



Add Feature Type(s)" section of the dialog, choose the option to "Copy from Reader."

When prompted, select one of the existing Public Art Reader feature types- for example Mount Pleasant - and click OK. A new Writer feature type will be added that is a copy of the Reader equivalent.

Connect it to the final transformer output:



6) Edit Schema

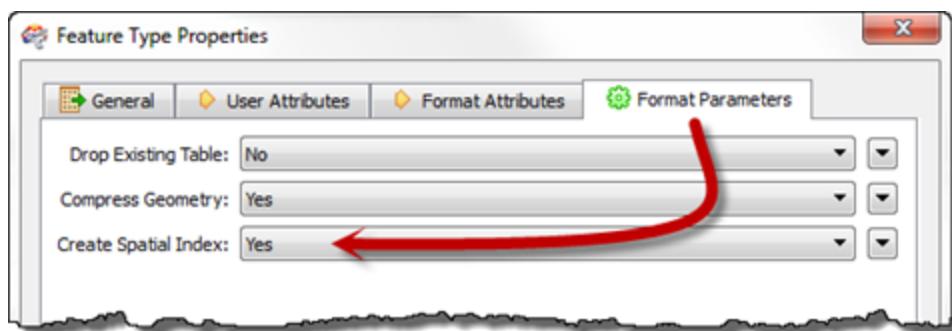
The schema we created is not exactly what we want, so it should be edited. What we need is:

- name - string(46)
- title - string(32)
- address - string(30)
- postcode - string(3)

So open the properties dialog for the new Feature Type. Click on the User Attributes tab and edit the schema to what is required, but don't click OK yet. Notice that we can't use upper-case characters in a field name for SpatiaLite.

7) Set Feature Type Parameters

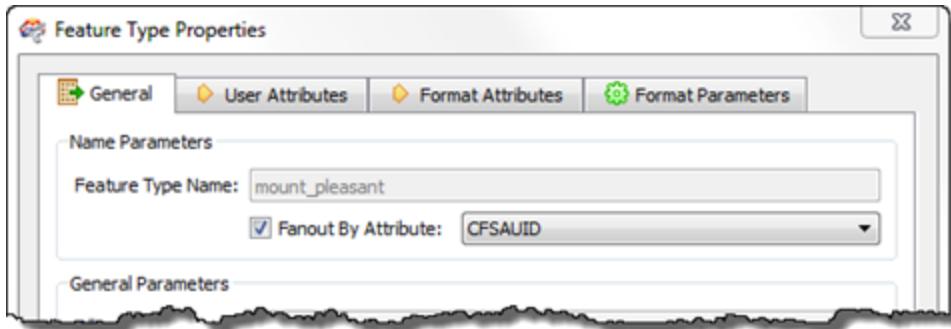
Now click on the Format Parameters tab. These are parameters that apply to this table. We're only creating one table, but if there were more each could have different values for these parameters. Ensure Create Spatial Index is set to yes, so that the output dataset can be queried more efficiently:



Again, don't click OK yet.

8) Set "Fanout"

Click on the General tab. Check the box marked Fanout by Attribute and select the attribute CSFAUID:



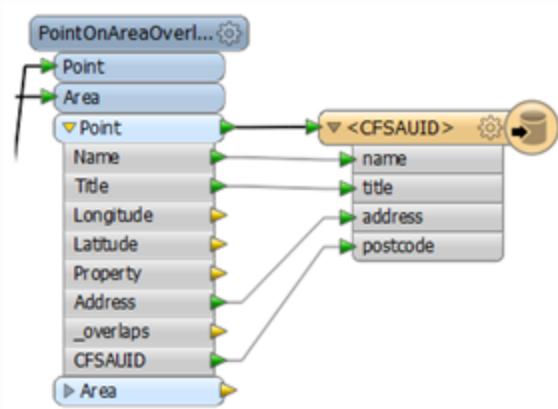
A fanout is an advanced function that will be useful here. By checking this box, and selecting the postcode attribute, we will get a separate table of features for each different postcode.

Click OK to close the dialog.

9) Map Schema

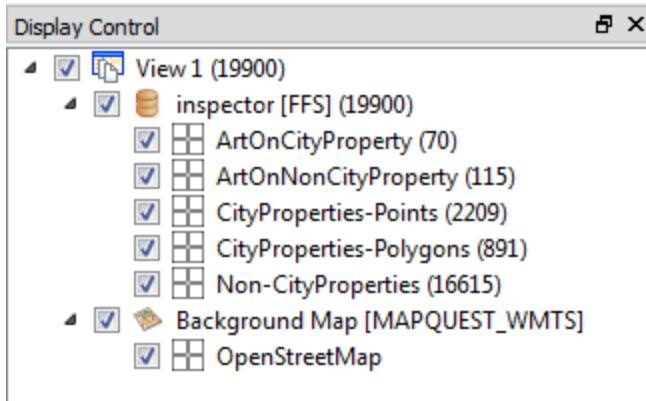
The final step is to map the schema correctly, as nothing is connecting up as yet. Draw connections between:

- Name - name
- Title - title
- Address - address
- CSFAUID - postcode



10) Run Workspace

Save and run the workspace. Inspect the output in the FME Data Inspector. You should find there are 70 features on a number of layers, each defined by a postal code:



Congratulations! You have now:

- Imported a Feature Type Definition
- Added a Writer
- Added a Writer Feature Type
- Modified Feature Type parameters
- Used a fanout

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.

Exercise

Now let's prove what you have learned by carrying out an exercise on basic data translation and data inspection.

Readers and Writers

Exercise 4e Readers and Writers	
Scenario	FME User
Data	Contours (Shape) Raster Orthophotos (GeoTIFF)
Overall Goal	Generate a new surface model in 3D PDF
Demonstrates	Handling Readers, Writers, and Feature Types
Starting Workspace	None
Finished Workspace	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise4e-Complete.fmw</i>
	<i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise4e-CompleteAdvanced.fmw</i>

In this exercise you are an FME user at the City of Interopolis who has just received a dataset for a recently completed property development. Because some earthwork took place, you must apply this development to the city's elevation model and create new output in a 3D PDF file.

1) Add Reader

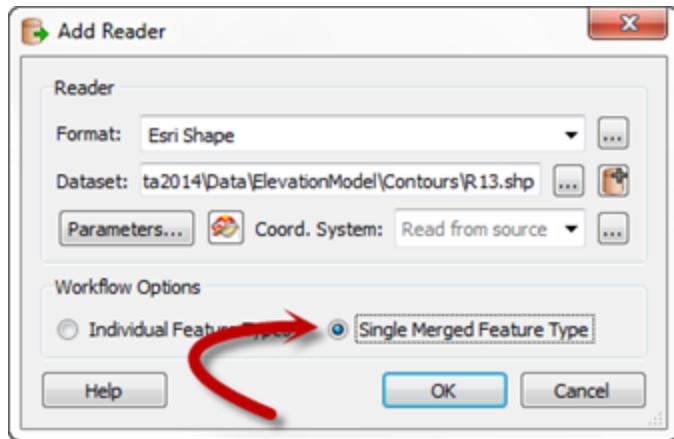
Start FME Workbench and begin with a blank workspace. Select **Readers > Add Reader** to read the city's existing contour files:

Reader Format Esri Shape

Reader Dataset: *C:\FMEData2015\Data\ElevationModel\Contours*.shp*

In other words, 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. It saves you from having to manually set this and change the dataset parameter.

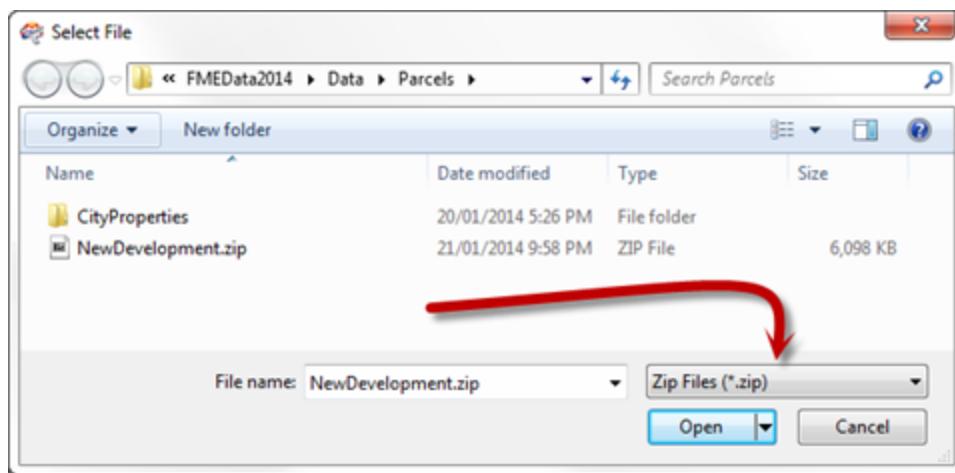
2) Add Reader

Now let's add the developer's data. Again select **Readers > Add Reader** from the menubar.

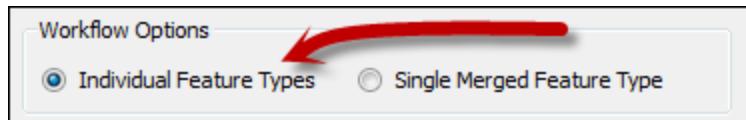
Reader Format Esri Shape

Reader Datasets C:\FMEData2015\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:



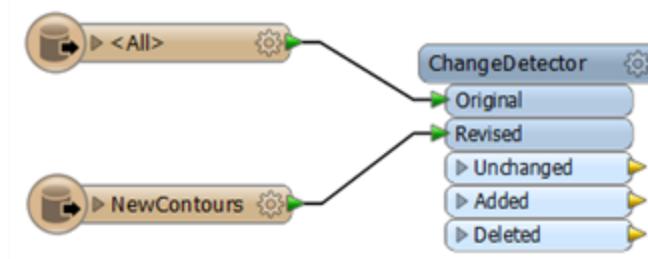
The other thing we should do is set the workflow option back to Individual Feature Types, in case there is data other than contours in there.



Now click OK. When prompted for which feature types you want from the zip file, select NewContours only.

3) 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, please add 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.

4) Add Inspectors

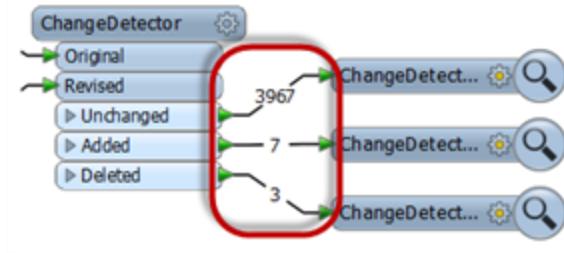
Add Inspector transformers to the output ports of the ChangeDetector and run the workspace to see what we have got so far.

NB: *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

7 contours have been added; i.e. they appear in the new data but not the original version

3 contours have been deleted; i.e. they appear in the original data but not in the new data.



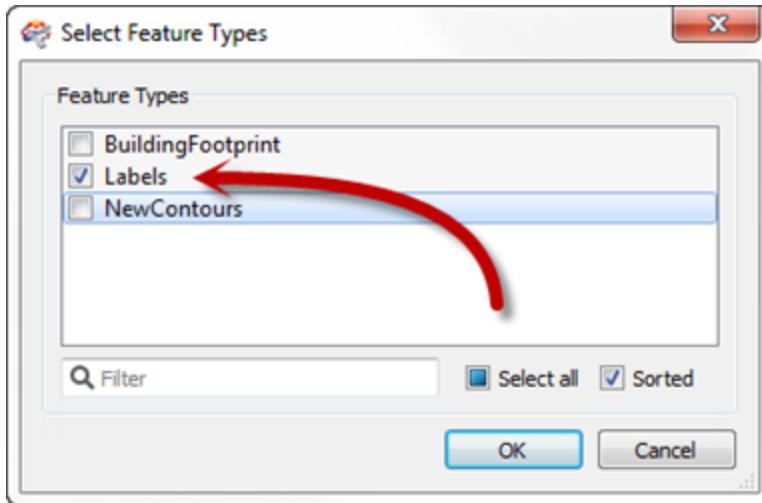
We'll want to generate the new elevation model from the Unchanged and the Added features; deleted ones we can drop. But first, there is a more pressing problem...

5) 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.

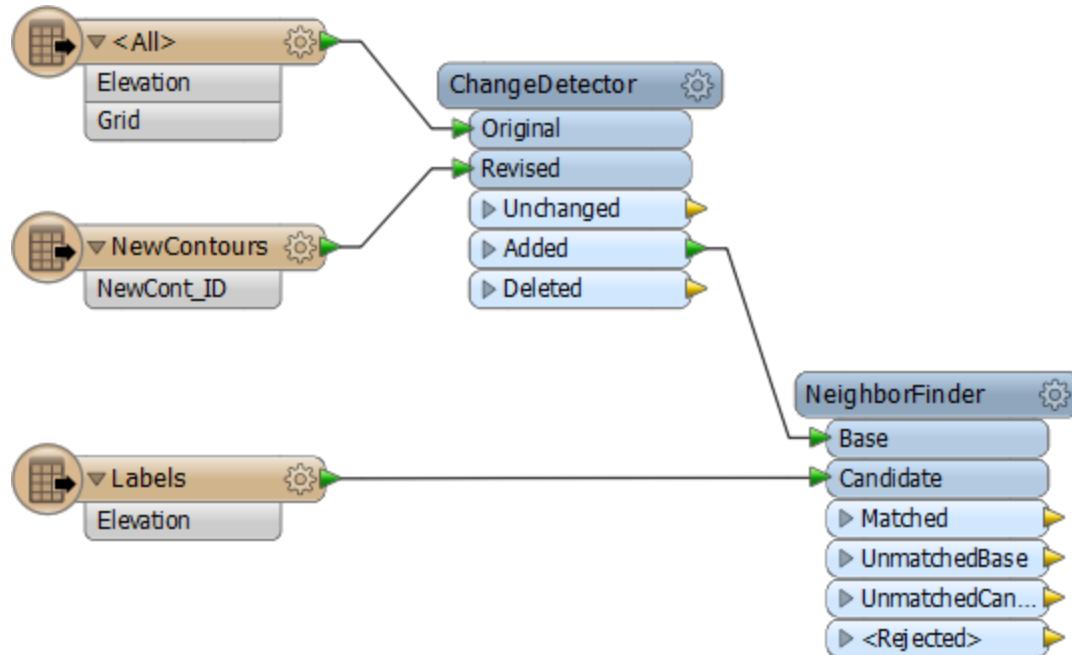
So choose **Readers > Import Feature Type** and select the NewDevelopment Reader to import to. This time, when the list appears, be sure to select the Labels feature type:



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.

6) 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:

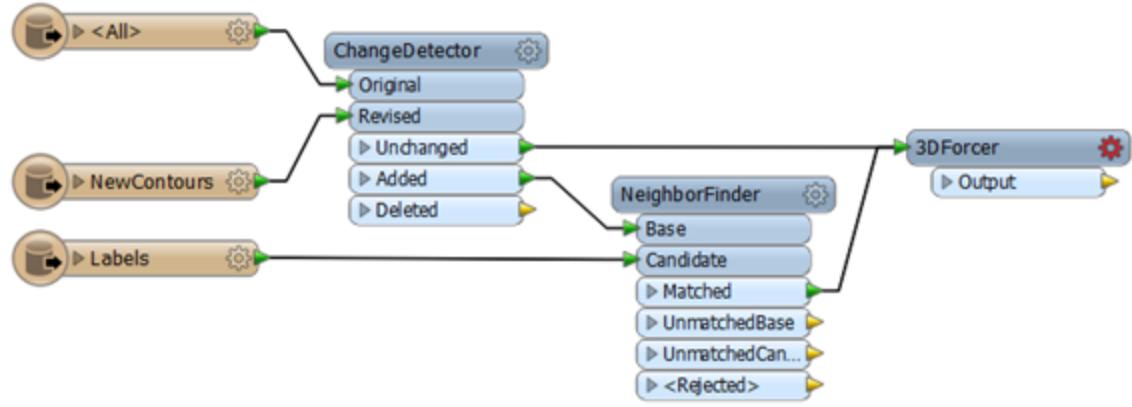


Open the parameters dialog and set the Maximum Distance parameter. This defines the maximum distance from the contour to the label. Because the labels appear to be almost on top of the contour you can set this to a low number, for example 5.

Now add some more Inspectors and rerun the workspace to ensure all seven contour features have an elevation value now.

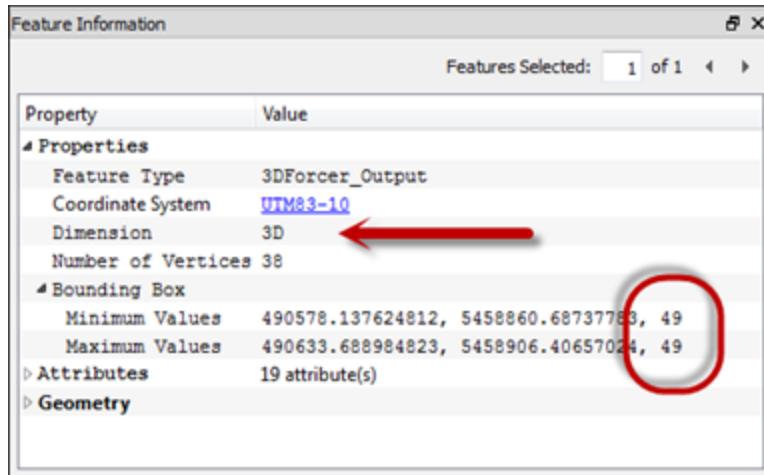
7) Add 3DForcer

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.



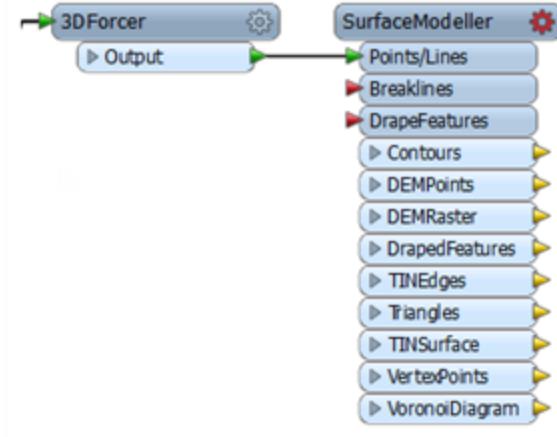
Now open the parameters dialog. For Elevation, click the drop-down arrow and choose **Attribute Value > Elevation**.

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:



8) Add SurfaceModeller

Now we can create a surface model. Add a SurfaceModeller transformer connected to the 3DForcer output. The input port to connect is Points/Lines.

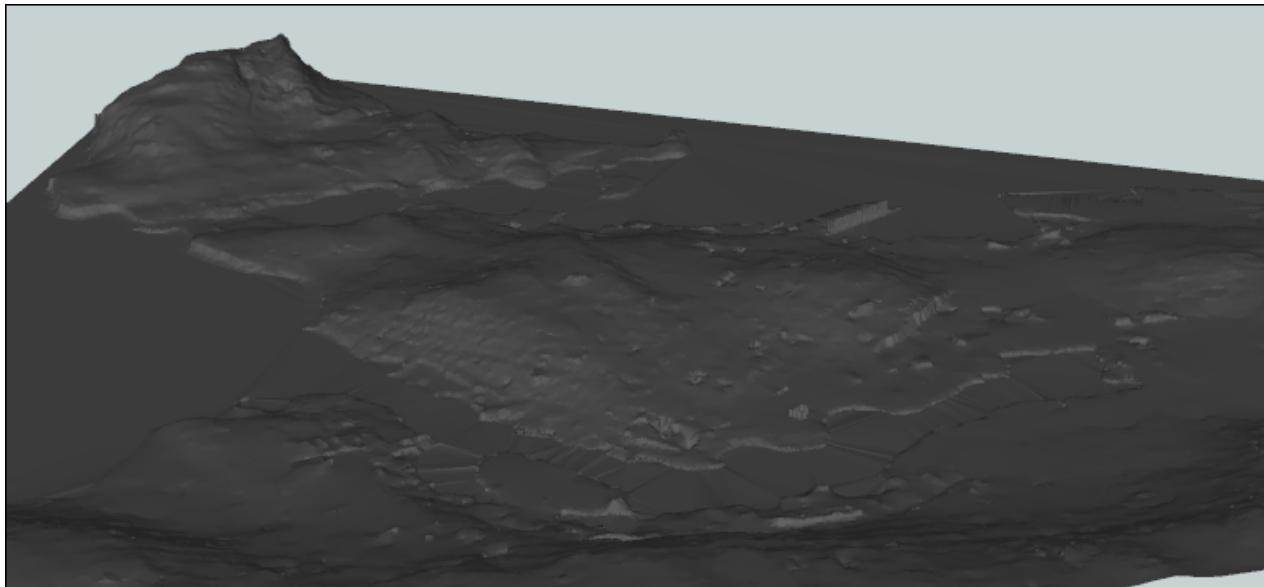


Open the parameters dialog. Set a Surface Tolerance of 1, and turn off the checkbox for Output Contours. If you now connect an Inspector to the TINSurface output port and run the workspace, you will be able to inspect the data in 3D (click the orbit icon on the toolbar)

9) 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 to option to apply the result to an existing attribute. Now Vancouver will have a more interesting profile:



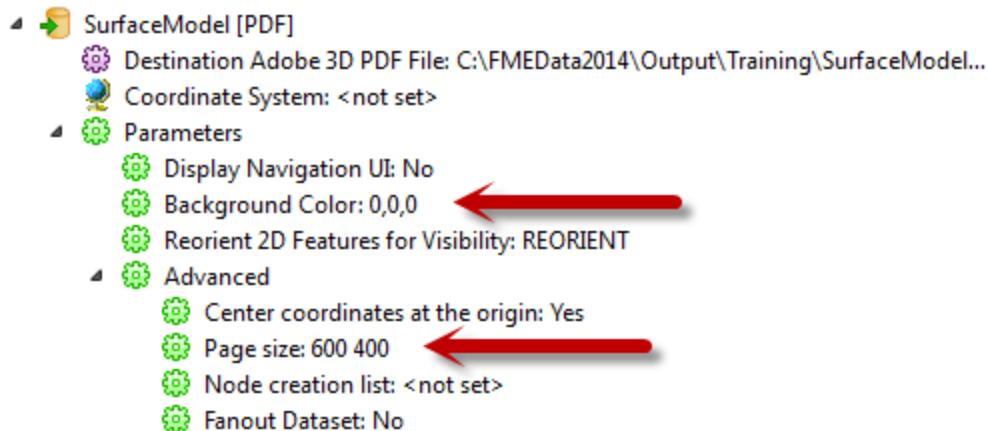
10) Add Writer

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:\FMEData2015\Output\Training\SurfaceModel.pdf

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.

Congratulations: You've now used Readers and Writers (and transformers) to update an elevation model and create a 3D rendering of it



As an advanced task we can drape a raster image onto the surface model to make it look prettier.

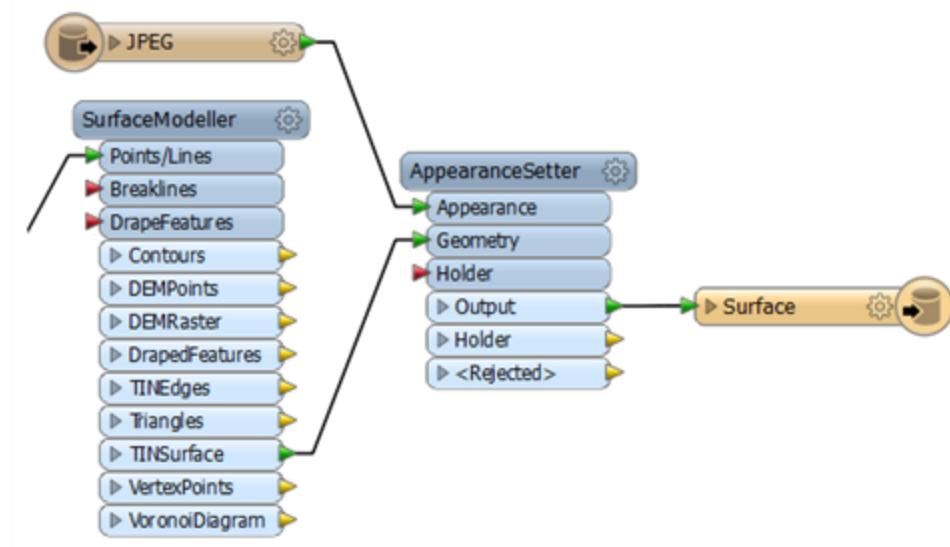
11) Add Reader

Add a Reader to read the raster data. The details are:

Reader Format	JPEG (Joint Photographic Experts Group)
Reader Datasets	C:\FMEData2015\Resources\OverlayImage.jpg

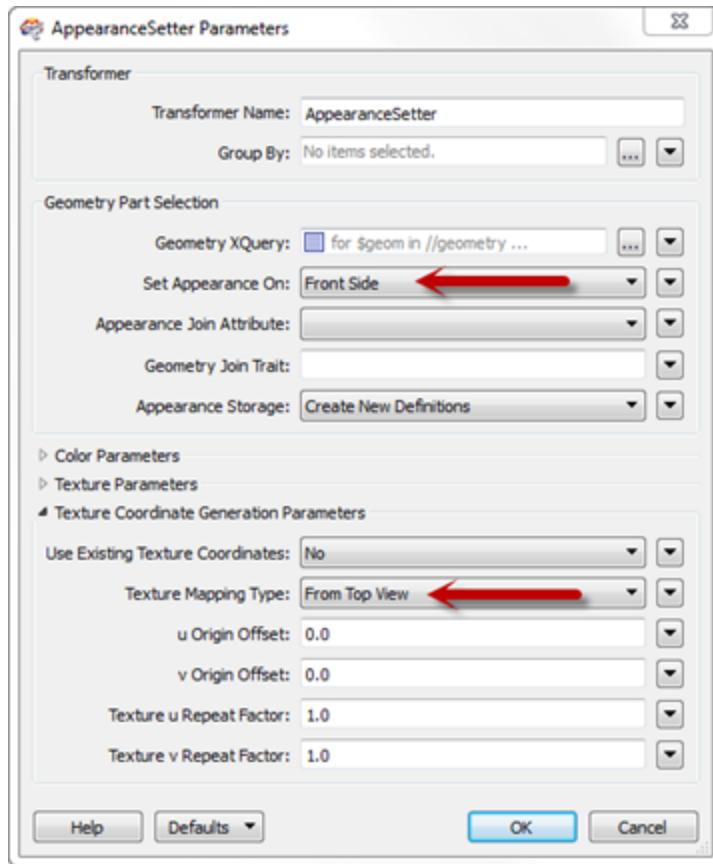
12) Add AppearanceSetter

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:



Open the parameters dialog. 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:



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.

NB: By default the PDF output will now be 87mb in size. If this is too large for your computer to handle, add a RasterResampler transformer after 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.

13) Overlay Building Footprint

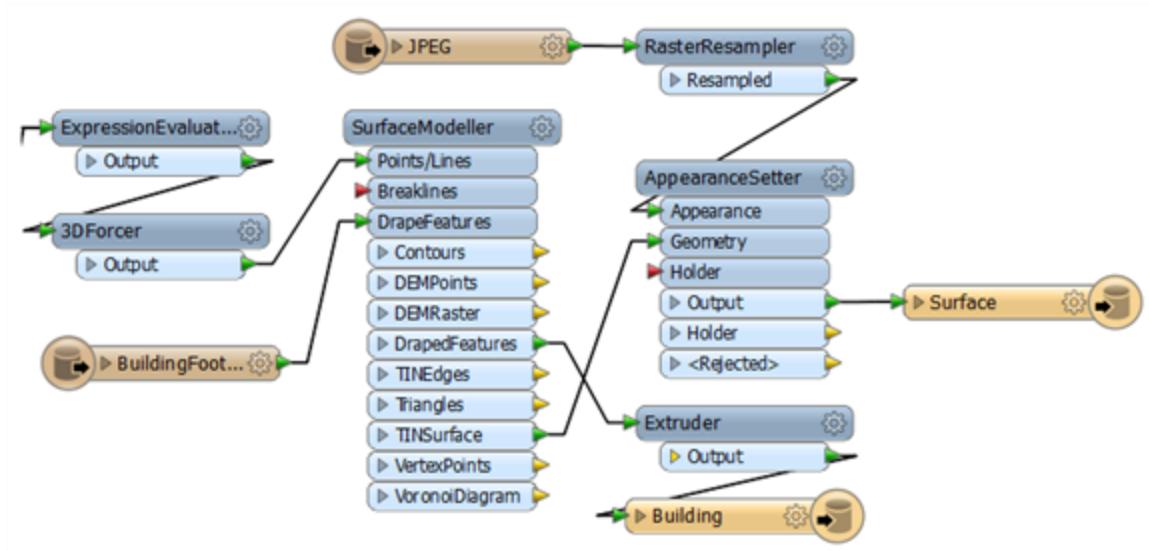
In the previous output, did you spot the lump in the surface where the new property is going to be developed? The one last thing we could do is add the building outline so that we can see where it will go.

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 Shape Reader.

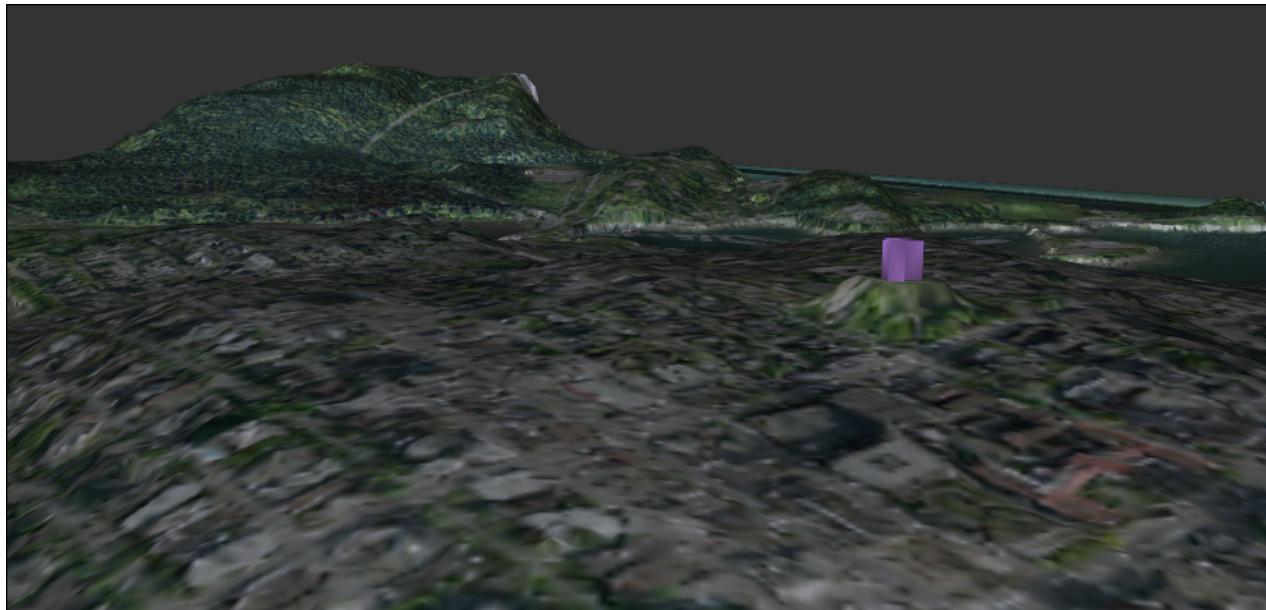
Connect that up to the SurfaceModeller:DrapeFeatures input port.

Add an Extruder transformer, connected to the SurfaceModeller:DrapedFeatures output port. Set the 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! You have now completed the exercise for this chapter.

Chapter 5 - 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.



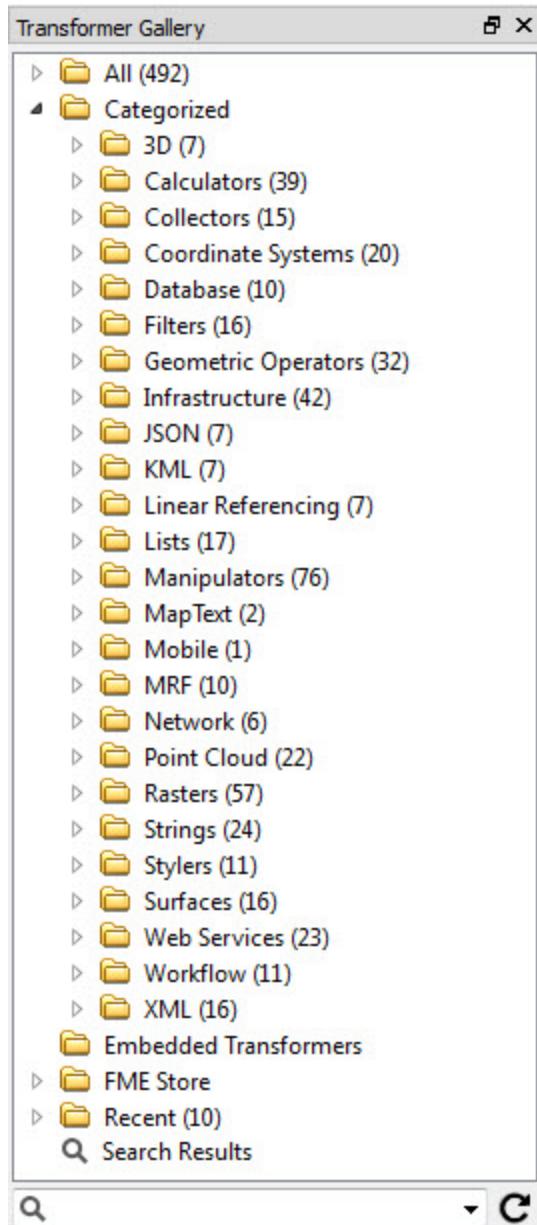
Although the transformer list can look a bit overwhelming, don't panic! The reality is that most users focus on 20-30 transformers that are relevant to their day-to-day workflow. You don't need to know every single transformer to use FME effectively.

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.



Notice that in FME 2015 each category now lists the number of transformers that it contains!

Important categories include:

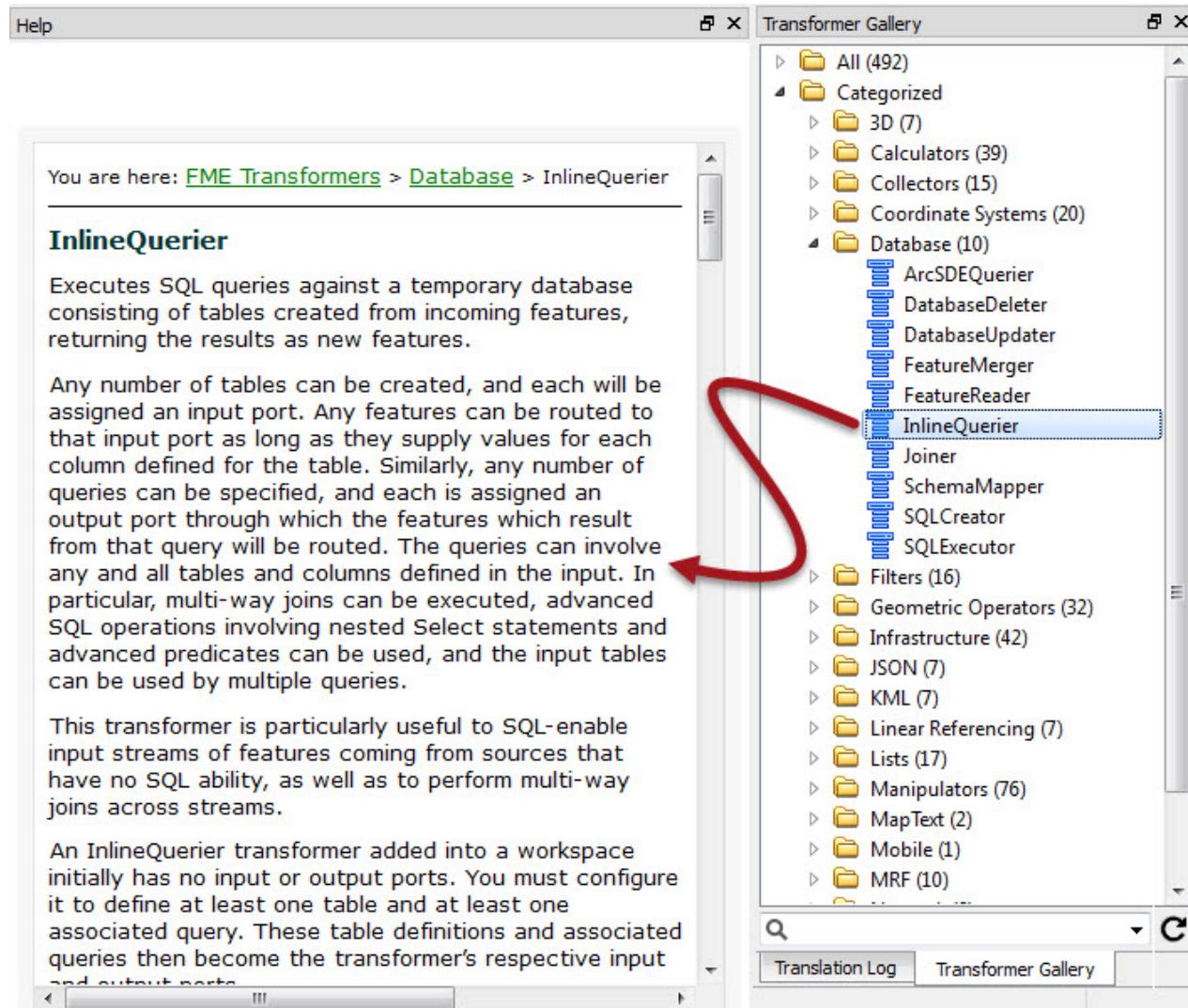
- **Calculators:** Calculate a value and supply it as a new attribute.
- **Database:** Interact with external databases.
- **Filters:** Split and re-route data
- **Geometric Operators:** Process feature geometry.
- **Infrastructure:** Structural transformation and scripting with Tcl/Python
- **Lists:** Work with list attributes.
- **Rasters:** Work with raster datasets.
- **Strings:** Create, modify and delete string (character) attributes.
- **Stylers:** Prepare features for output to particular formats.
- **Surfaces:** Work with surfaces; for example, create contours.
- **Web Services:** Communicate with web services using HTTP.
- **Workflow:** Run workspaces either locally or on an FME Server.
- **XML:** Transformers that deal with bringing XML data into FME.

Simply click on the expand button to show all transformers within a particular category.

Transformer Help

The Help window in Workbench displays information about transformers from the FME documentation.

The Transformer Gallery is directly linked to the Help window - that is, a transformer selected in the gallery triggers help contents to display in the Help window.



A screenshot of the FME Desktop software interface. On the left, a help window titled 'InlineQuerier' provides documentation. The right side shows the 'Transformer Gallery' with a tree view of transformer categories. A red arrow points from the 'InlineQuerier' section in the help window to its corresponding entry in the gallery.

You are here: [FME Transformers](#) > [Database](#) > [InlineQuerier](#)

InlineQuerier

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

Any number of tables can be created, and each will be assigned an input port. Any features can be routed to that input port as long as they supply values for each column defined for the table. Similarly, any number of queries can be specified, and each is assigned an output port through which the features which result from that query will be routed. The queries can involve any and all tables and columns defined in the input. In particular, multi-way joins can be executed, advanced SQL operations involving nested Select statements and advanced predicates can be used, and the input tables can be used by multiple queries.

This transformer is particularly useful to SQL-enable input streams of features coming from sources that have no SQL ability, as well as to perform multi-way joins across streams.

An InlineQuerier transformer added into a workspace initially has no input or output ports. You must configure it to define at least one table and at least one associated query. These table definitions and associated queries then become the transformer's respective input and output ports.



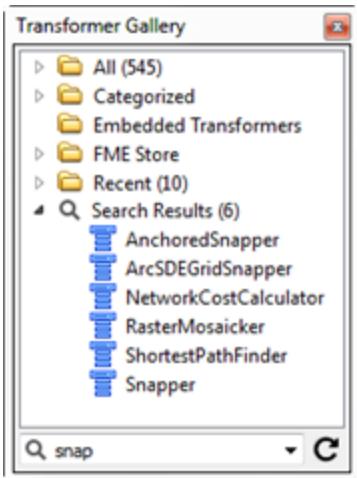
Another useful piece of documentation is the FME Transformer Reference Guide, which can be found on our [web site](#).

Transformer Searching

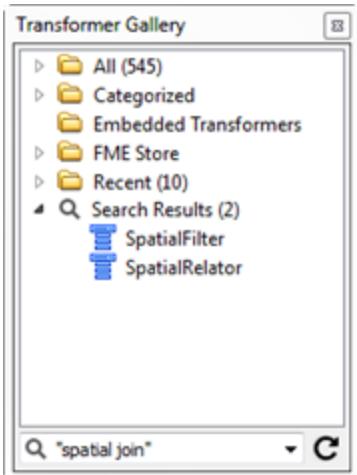
There are search functions in both the transformer gallery and Quick Add dialog.

To perform a search in the transformer gallery, simply enter the search terms and either press the <enter> 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.



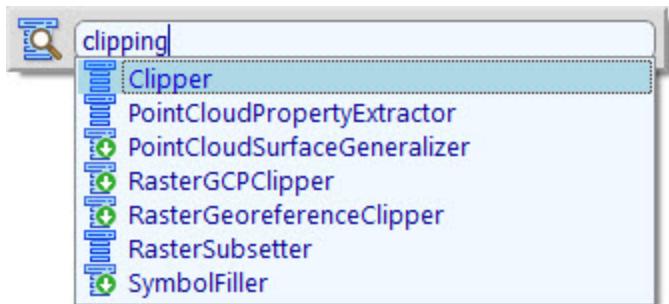
Quick Add search-terms can also be full or partial words:



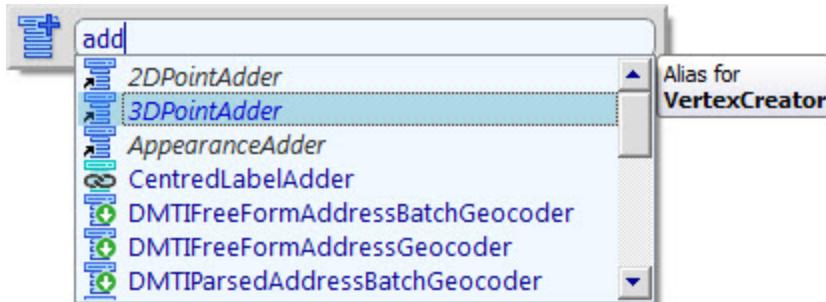
By default, Quick Add does not look in transformer descriptions, so the search term must be the actual name of a transformer.



However, Quick Add will search in the transformer descriptions if you press the <TAB> key.



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 Store.



The FME Store is a facility for sharing (and selling) FME functionality such as custom transformers and formats.

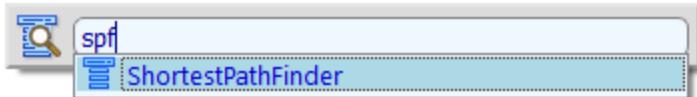
Such transformers are flagged in the search tools with a small, downwards-pointing arrow icon.

The full set of available transformers can be browsed by visiting fimestore.safe.com

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).

For example, Quick Add will return the *ShortestPathFinder* when the search term is the initials "spf":



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 a proficient in FME using only a handful of transformers; if they are the right ones!

The Top 25

At Safe Software, we find it useful to keep a list of the most-used transformers. It tells us where to direct our development efforts in making improvements. And no doubt it will give users a head-start on knowing which of the (400+) FME transformers they're most likely to need in their work.

The following table provides the list of the most commonly used 25 transformers in FME 2014, calculated from user feedback. The Tester transformer is consistently number one in the list every year, highlighting its importance.

Rank	Transformer	Rank	Transformer
1	Tester	14	Bufferer
2	AttributeCreator	15	Aggregator
3	Inspector	16	GeometryFilter
4	FeatureMerger	17	AttributeExposer
5	Creator	18	ExpressionEvaluator
6	AttributeRenamer	19	Joiner
7	AttributeKeeper	20	StringConcatenator
8	Reprojector	21	DuplicateRemover
9	Clipper	22	StringReplacer
10	AttributeFilter	23	AttributeCopier
11	AttributeRemover	24	Dissolver
12	TestFilter	25	Sorter
13	Counter		

Besides the obvious transformers for transforming geometry (Clipper, Bufferer, Dissolver) and the obvious transformers for transforming attribute values (StringReplacer, StringConcatenator, Counter) there are some other distinct groups of transformers.

Managing Attributes

These transformers 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.

Filtering

Filtering is the technique of subdividing 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



The top 25 transformer list showed transformers for managing attributes are very popular.

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 transformers for managing attributes are these:

- AttributeCreator
- AttributeRenamer (and BulkAttributeRenamer)
- AttributeCopier
- AttributeRemover (and BulkAttributeRemover)
- AttributeKeeper
- AttributeExposer

Many of these transformers can carry out similar operations. In fact, with the *AttributeCreator* and *AttributeRenamer* transformers having the widest range of functionality, some users may use these two attribute-related transformers to the exclusion of most others.

Lists

A List in FME is a mechanism that allows multiple values per attribute.

For example, the attribute *myAttribute* can only contain a single value.

However, the list attribute *myList{ }.myAttribute* can contain multiple values as:

- *myList{0}.myAttribute*
- *myList{1}.myAttribute*
- *myList{2}.myAttribute*
- etc.

There are various transformers designed to operate specifically on lists - they can be found in the Lists category of the transformer gallery - and list functionality is built into other transformers (such as the AttributeCreator) too.

AttributeCreator

With functionality for string concatenation and expression evaluation the *AttributeCreator* is the all-singing, all-dancing transformer of choice for handling attributes in workspaces.

There is an ever-lengthening list of capabilities for this transformer, but the main ones are:

- Manually Create an Attribute
- Set an Attribute Value
- Copy an Attribute

Manually Create an Attribute

The fundamental purpose of this transformer is to manually enter an attribute name and value for it. It is essentially an author-defined constant since every feature gets the same value.

Set an Attribute Value

Because the Attribute Name field allows an attribute to be selected from a list, it can be used to set the value of an existing attribute, rather than manually entering a new attribute name. Again, this is an author-defined constant, since every feature gets the same value for that attribute.

Copy an Attribute (Can replace the AttributeCopier)

Since the Value field allows an attribute to be selected from a list, this transformer can be used in place of an *AttributeCopier*, to copy the value from one attribute into another (either a new attribute, or an existing one).

Attributes To Set	
Attribute Name	Value
◆ NewAttribute1	<i>k</i> A value for a new attribute
◆ ExistingAttribute1	<i>k</i> A new value for an existing attribute
◆ NewAttribute2	◆ ExistingAttribute2
<input style="font-size: small; width: 15px; height: 15px; border: 1px solid #ccc; border-radius: 50%; margin-right: 5px; vertical-align: middle;" type="button" value="+"/> <input style="font-size: small; width: 15px; height: 15px; border: 1px solid #ccc; border-radius: 50%; margin-right: 5px; vertical-align: middle;" type="button" value="-"/> <input style="font-size: small; width: 15px; height: 15px; border: 1px solid #ccc; border-radius: 50%; margin-right: 5px; vertical-align: middle;" type="button" value="▲"/> <input style="font-size: small; width: 15px; height: 15px; border: 1px solid #ccc; border-radius: 50%; margin-right: 5px; vertical-align: middle;" type="button" value="▼"/> <input style="font-size: small; width: 15px; height: 15px; border: 1px solid #ccc; border-radius: 50%; margin-right: 5px; vertical-align: middle;" type="button" value="✖"/>	<input style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px; margin-top: 5px;" type="button" value="Duplicate"/>

Here the user is:

- Manually creating a new attribute and setting a value for it
- Setting a value for an existing attribute
- Copying an existing attribute to a new one

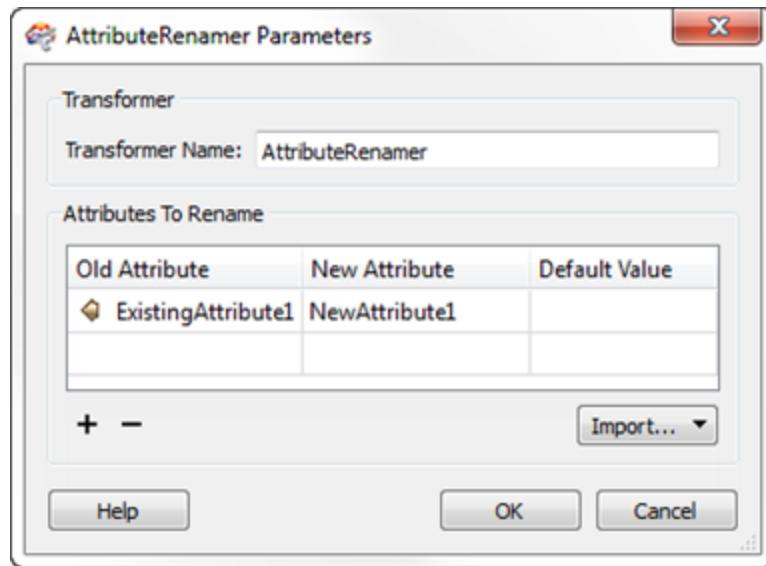
AttributeRenamer

This is a comprehensive transformer that can be used to do many tasks, including:

- Renaming an existing attribute
- Renaming an existing attribute and setting a new value for it
- Manually creating a new Attribute (and setting a value)
- Deleting an attribute

Rename an Attribute

The fundamental purpose of this transformer is to select an attribute and manually enter a new name for it. The old attribute is removed and replaced with the newly named one.



Provided the Default Value field is left empty, the new attribute will take the value of the old one.

This is obviously of most use when wanting to read data that has one schema (set of attribute names) and write it to a dataset that needs a different schema.

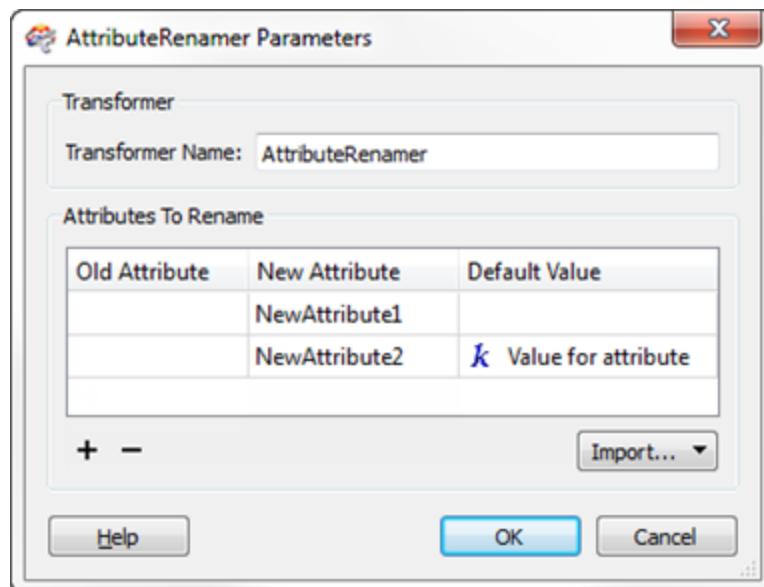
Rename and Set an Attribute

A field in the parameters dialog allows the user to set a default value. This way an attribute can be renamed, and a new value defined for it, simultaneously.

Manually Create (and Set) an Attribute

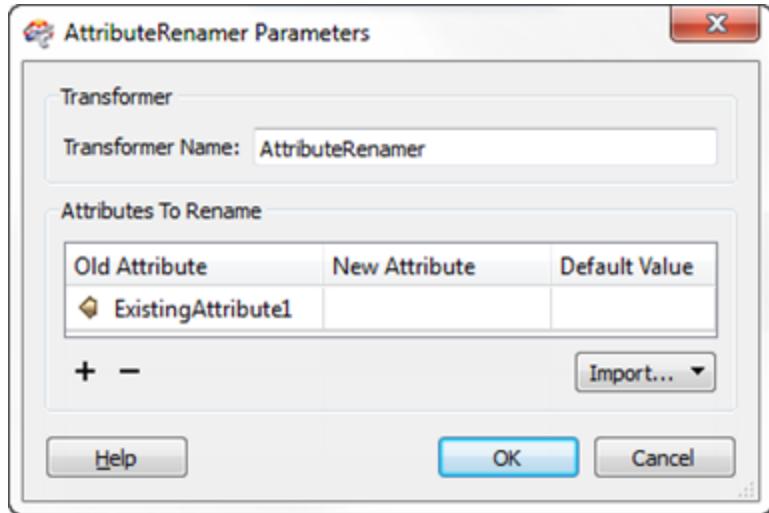
Entering an attribute name into the New Attribute field, without selecting an Old Attribute, means a new attribute is created; substituting for the *AttributeCreator*.

Entering an attribute name into the Default Value field when a new attribute is created effectively creates the attribute and sets a value for it in one operation. Every feature gets the same attribute with the same value. Again, this can be used as a substitute for the *AttributeCreator*.



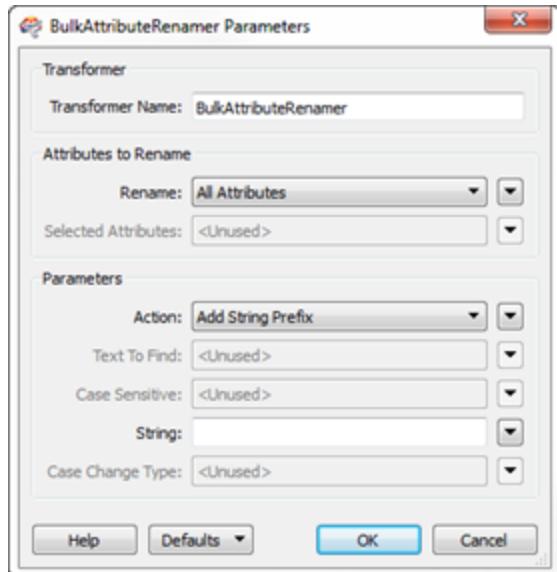
Delete an Attribute

By selecting an Old Attribute, but leaving the New Attribute field empty, the old attribute will be deleted.



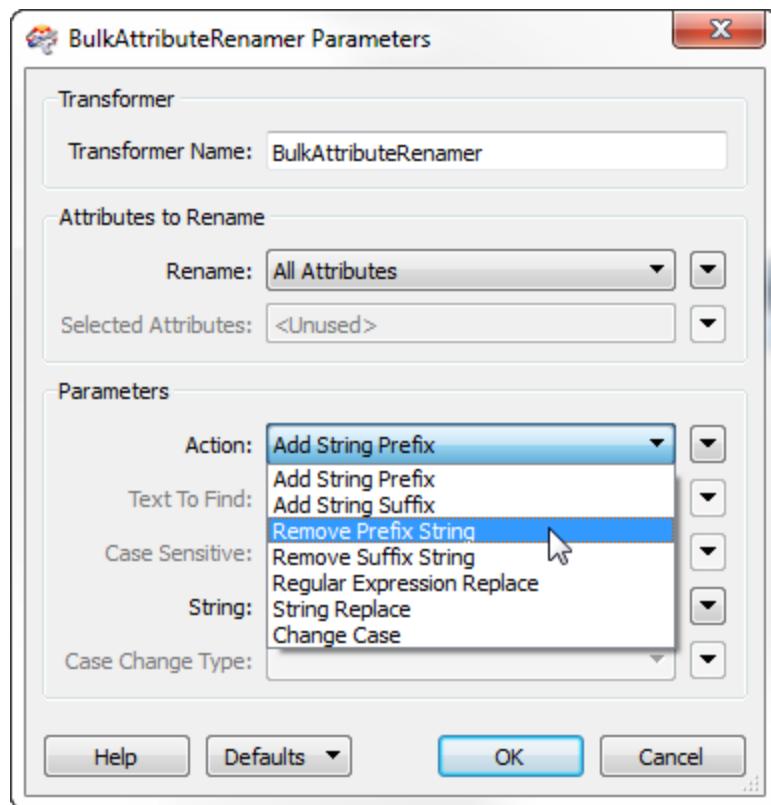
BulkAttributeRenamer

This transformer carries out the core function of the *AttributeRenamer*; renaming attributes. But instead of manually specifying each attribute, this transformer lets the user enter a string-matching expression in order to define which attributes to rename.

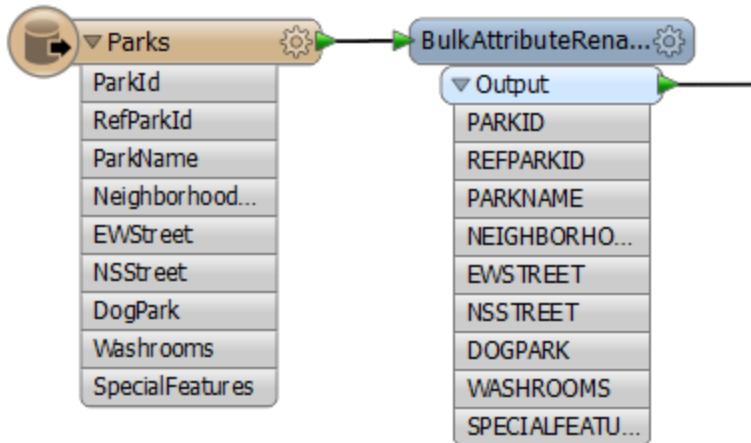


This transformer is very powerful as it can carry out a number of renaming actions. These are:

- Add String Prefix
- Add String Suffix
- Remove Prefix String
- Remove Suffix String
- Regular Expression Replace
- String Replace
- Case Change



The power of the transformer is also in its ability to manipulate multiple attributes at once, without having to individually select them all.

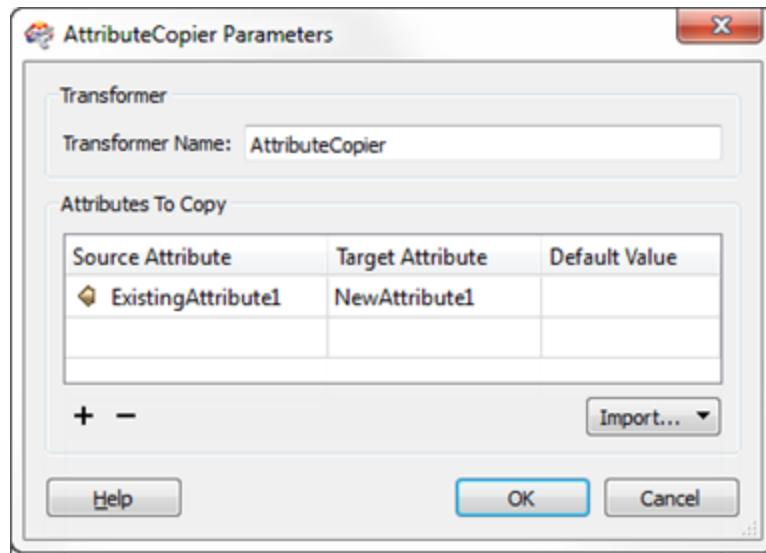


It can be particularly useful when the incoming data has a prefix or suffix on the attribute names that needs to be removed, or (like here) when all the incoming attributes are mixed/lowercase and need to be UPPER case on the output.

AttributeCopier

The *AttributeCopier* transformer is similar in structure to the *AttributeRenamer* – obviously retaining the old attribute as well as creating the new – but in a much simpler dialog.

The source attribute is simply a pick list, and the target can either be an existing attribute or a newly defined one.



The dialog also includes a Default Value field so that the newly copied attribute can (optionally) be given a new value.

Again this is of use when wanting to read data that has one schema (set of attribute names) and write it to a dataset that needs a different schema.

AttributeRemover

AttributeKeeper

These two transformers carry out the same function, but approach it from opposite directions.

Both remove attributes (and list attributes) from features. However the *AttributeRemover* lets the user specify which ones are to be removed, whereas the *AttributeKeeper* lets them specify which ones are not to be removed; in other words, this transformer lets the user specify which ones to keep.

Basically, use:

- The *AttributeRemover* when one or two attributes are to be removed, but the majority of them kept.
- The *AttributeKeeper* when the majority of attributes are to be removed, and only one or two of them kept.

Perhaps a more important question is what benefit removing attributes brings.

- Removing attributes that aren't required tidies up a workspace and makes it easier to understand what is happening.
- Removing attributes helps speed up the Workbench interface. Whenever a connection is changed or a transformer added, Workbench has to check to see how this affects the workspace. The fewer attributes, the less checking is required; therefore, Workbench performs faster.
- Removing attributes also helps speed up the translation. FME may cache data to memory and disk at various stages of a translation. The fewer attributes, the less data needs to be cached, improving both speed and use of memory resources.



*Attributes **don't** need to be removed to avoid writing them to a destination dataset, provided the attribute schema is manual. If the attribute doesn't exist on a manually defined schema, then it won't get written. However, if the attribute schema is automatic, then all attributes present on incoming features will be written to the output. Any attributes not required should therefore be removed using these transformers.*

BulkAttributeRemover

This transformer lets the user enter a string-matching expression in order to define which attributes to remove. As its name implies, it is very similar in operation to the *BulkAttributeRenamer*.

It is useful for removing a whole series of similarly named attributes.

Attribute Management

Exercise 5a Attribute Management	
Scenario	Converting an address database to a format suitable for web streaming/delivery
Data	Addresses (File Geodatabase)
Overall Goal	Convert File Geodatabase to GML and map schema
Demonstrates	Attribute Management for Schema Mapping
Start Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\DesktopBasic\Exercise5a-Complete.fmw

The city has an important project. As part of a push for open government, the corporate address database is to be made available online. However, it is currently held in Esri File Geodatabase format and this is not thought suitable to deliver in its raw form. Instead the data will be translated to a GML/XML format and transformed into a new schema.

1) Inspect Source Data

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:\FMEData2015\Data\Addresses\Addresses.gdb

The table that is to be translated is called "PostalAddresses." 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						
	OBJECTID	GlobalID	OWNERNM1	OWNERNM2	PSTLADDRESS	PSTLCITY
1	1	{1C55C207-5A3E-4565-88D5-994AEDF5F445}	Russel Reding	<null>	1188 W Pender St	Vancouver
2	2	{8AFE5C37-E0A7-49FF-A281-186BD1AC6112}	Nicolas Hammel	<null>	1661 Ontario St	Vancouver
3	3	{6963B7DB-653A-4035-B031-2E4CCFDD14A6}	Brandon Scanlan	<null>	53 Smithe St	Vancouver
4	4	{67133025-D2F6-4C67-A8A9-8C24796E3281}	Tyisha Hullett	<null>	181 W 1st Av	Vancouver
5	5	{3E5D9415-BDD8-4F2B-A9FE-21C1B28D622C}	Epifania Sorrentino	<null>	141 W 1st Av	Vancouver
6	6	{E191FAAD-295B-49D7-AEC5-D564558B5D77}	Gwyn Blacker	<null>	808 Gore Av	Vancouver
7	7	{9E7BDFEB-2288-48AA-9AFD-D9436204A1E8}	Armand Augustyn	<null>	266 E 15th Av	Vancouver
!!!						
<input type="text"/> in any column		13597 row(s)				
PostalAddress						

Optionally you may wish to locate the file `C:\FMEData2015\Resources\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 GML dataset.

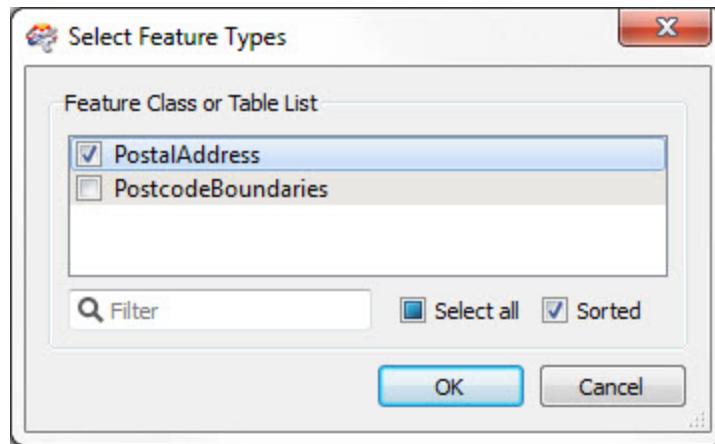
2) Create Workspace

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:\FMEData2015\Data\Addresses\Addresses.gdb`

When prompted, select only the `PostalAddresses` table, and no others:

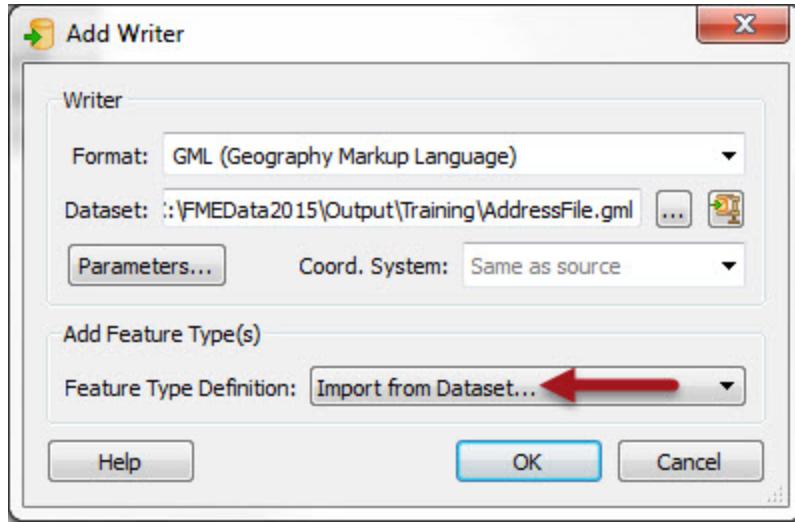


3) Add Writer

Now let's add a Writer to write the output data. Select **Writers > Add Writer** from the menubar and use the following:

Writer Format GML (Geography Markup Language)
Writer Dataset `C:\FMEData2015\Output\Training\AddressFile.gml`

Select 'Import from Dataset' in the 'Add Feature Types' section of the dialog. Then Click OK to add the Writer.

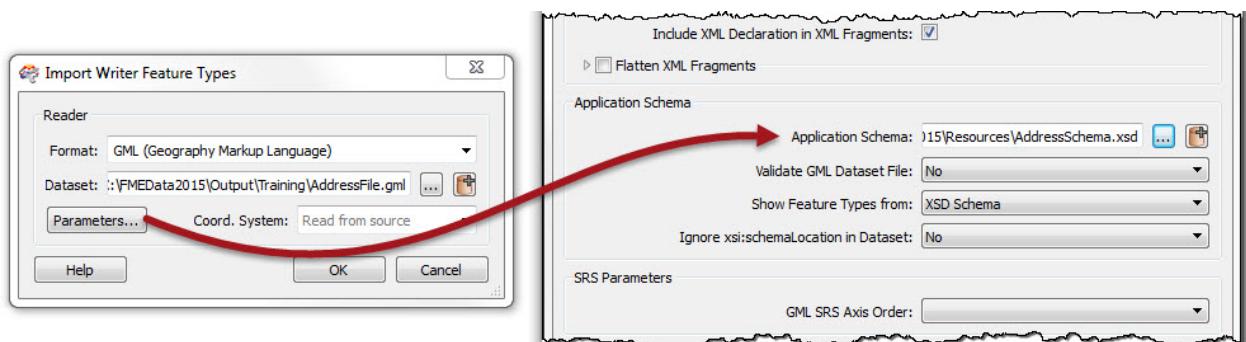


Import Feature Types

When prompted set the following parameters. The Dataset can be left as it is or emptied (left blank). It is not the important part:

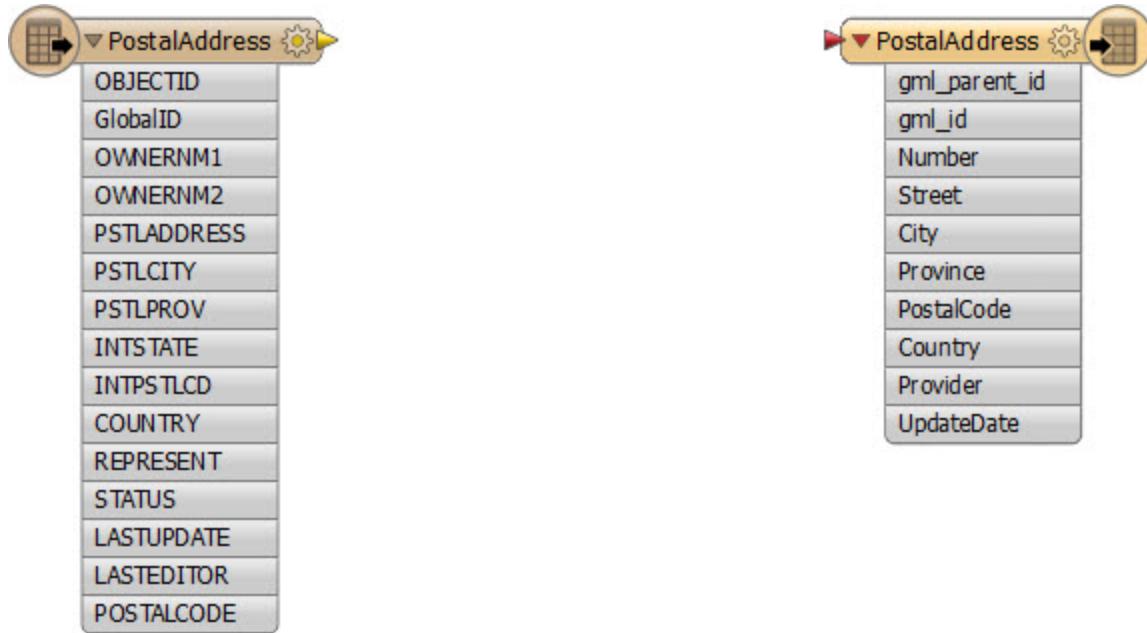
Reader Format GML (Geography Markup Language)
Reader Dataset

Parameters
Application Schema C:\FMEData2015\Resources\AddressSchema.xsd



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:



4) Map Schema

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.

Firstly let's clear up the Reader schema by hiding some of the unwanted attributes. Open the properties dialog for the Reader feature type. Click on the User Attributes tab to show the attributes available. Many of these are not required in the output. The simplest thing to do is hide them from view in Workbench.

Uncheck the buttons against these attributes to hide them:

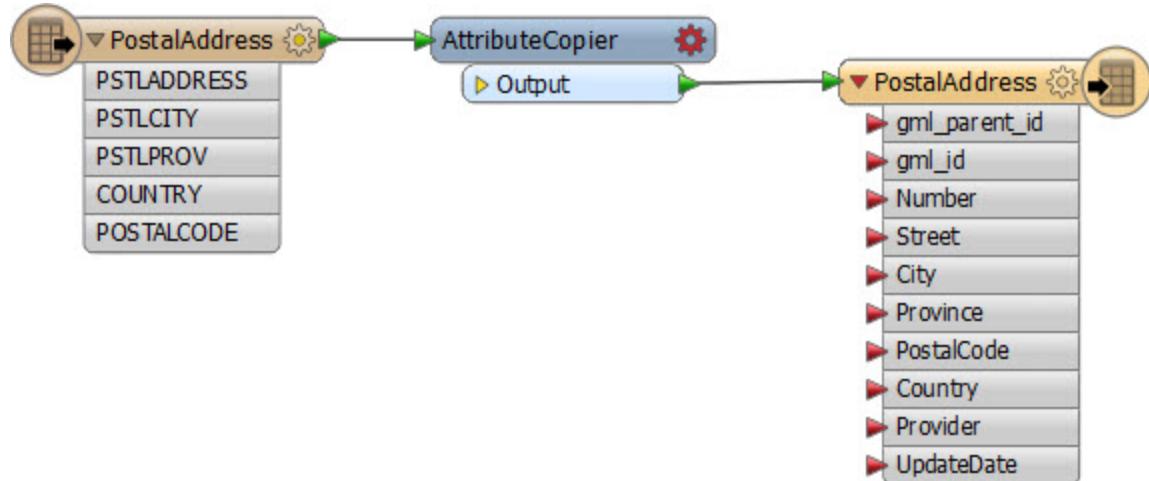
- OBJECTID
- GlobalID
- OWNERNM1
- OWNERNM2
- INTSTATE
- INTPSTLCD
- REPRESENT
- STATUS
- LASTUPDATE
- LASTEDITOR

Click OK to close the dialog.

5) Add AttributeCopier

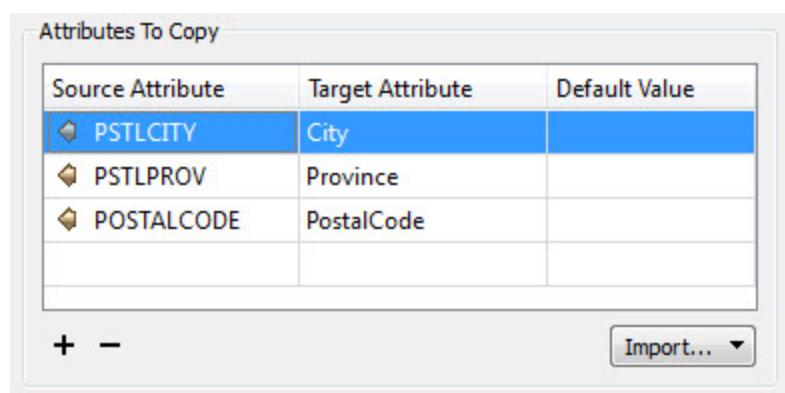
Several source attributes can be written to the output as they are, but do need renaming first.

Add an AttributeCopier transformer and connect it in between Reader and Writer feature types.



Open the parameters dialog. Set up the parameters to copy:

- PSTLCITY to City
- PSTLPROV to Province
- POSTALCODE to PostalCode

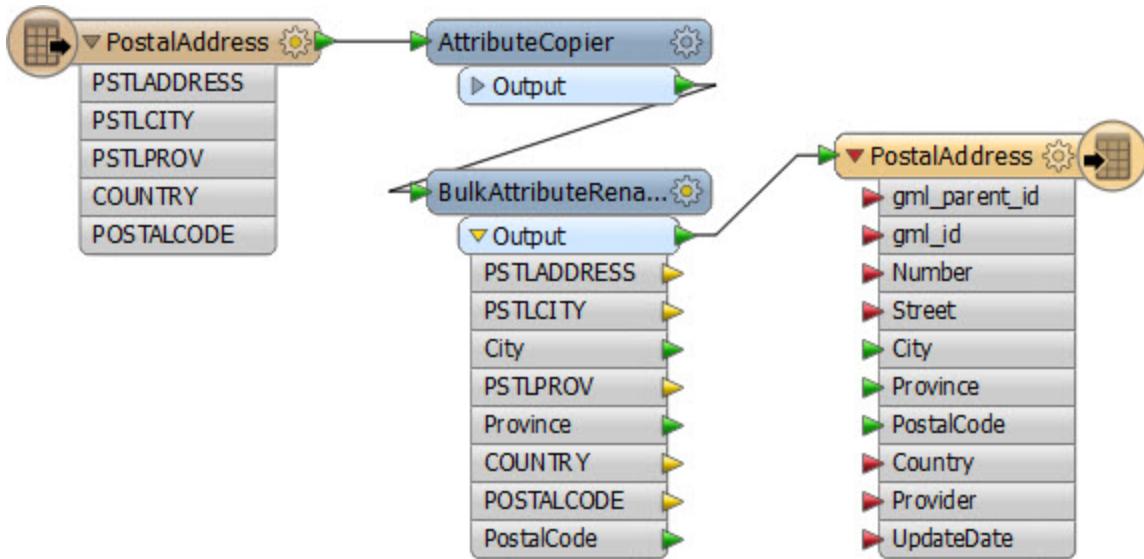


Notice how the source attributes remain in the data, as opposed to an AttributeRenamer transformer where they would be removed completely.

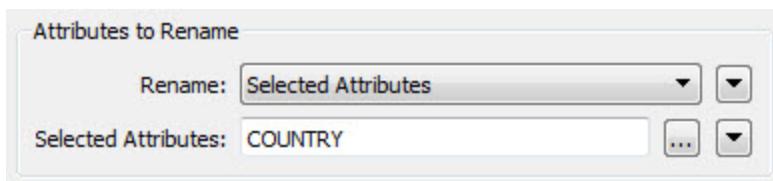
6) Add BulkAttributeRenamer

Any attributes whose name matches, except for the case of the name (i.e. UPPER or lower or Title case) we can fix with the BulkAttributeRenamer. In our exercise there is only one attribute that falls into that scenario, but we'll use the BulkAttributeRenamer anyway for practice.

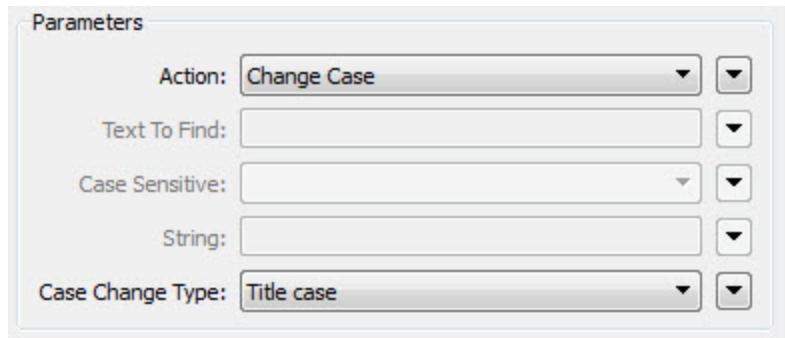
So, place a BulkAttributeRenamer into the workspace, after the AttributeCopier:



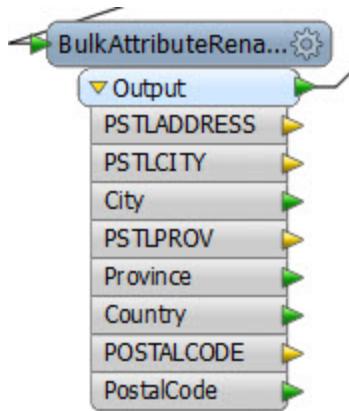
Open the parameters dialog. The only attribute we need to rename is COUNTRY. So change the Rename option to Selected Attributes and select COUNTRY:



Now set the Action parameter to Change Case, and the Case Change Type to "Title case":



Click OK to close the dialog. Notice how the attribute is renamed in the workspace:



NB: Yes, we could have just carried out this action in the AttributeCopier transformer, and for a single attribute we probably should. But this is a good way to familiarize you with the BulkAttributeRenamer in the safety of a training exercise!

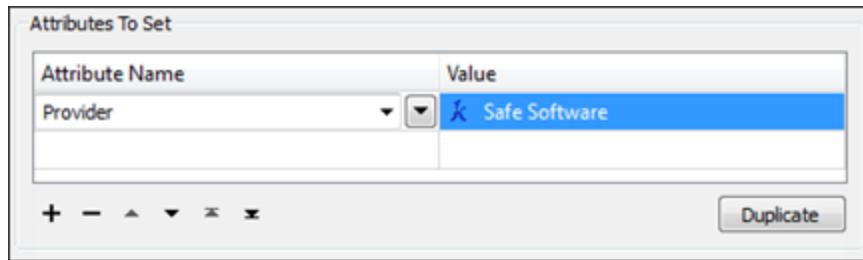
Question: Why do you think we can't use the BulkAttributeReplacer transformer on the attribute POSTALCODE? Why not try it and see what happens?

7) Add AttributeCreator

Two attributes on the output (Provider and UpdateDate) are new and cannot be copied from the source data. They must be created. Provider can be created with an AttributeCreator.

Place an AttributeCreator transformer after the BulkAttributeRenamer. Open the parameters dialog. Set up the parameters to create the attribute Provider. Because the attribute does not already exist you'll need to type the new name, rather than select it from a drop-down list.

The value can be your own organization name, or "Safe Software" or "City of Interopolis."

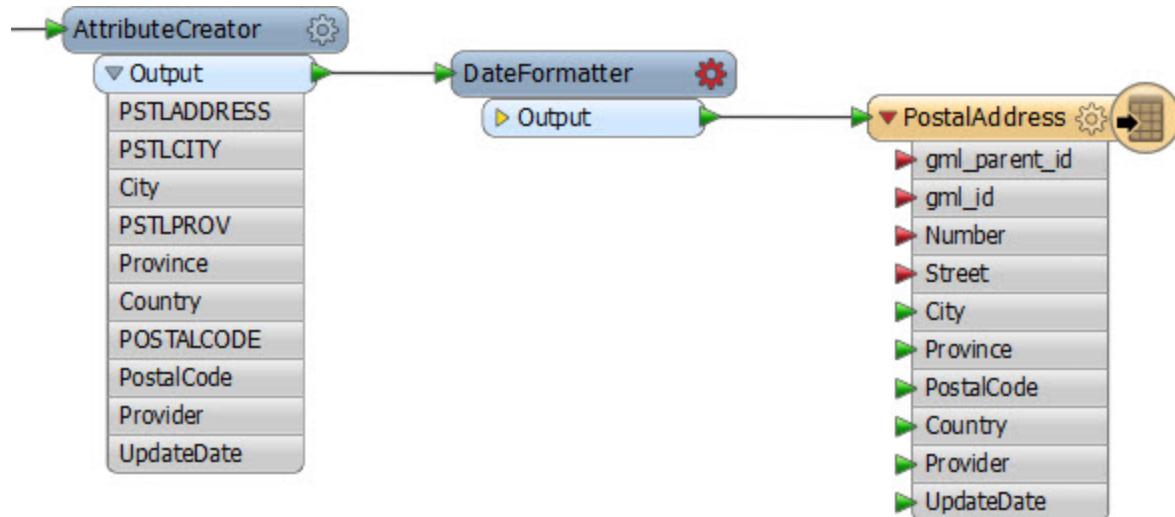


On the next line create the attribute called UpdateDate. Rather than hard-coding a value we'll set it to "TODAY" and deal with that using the DateFormatter:

Attribute Name	Value
UpdateDate	TODAY
Provider	Safe Software

8) Add DateFormatter

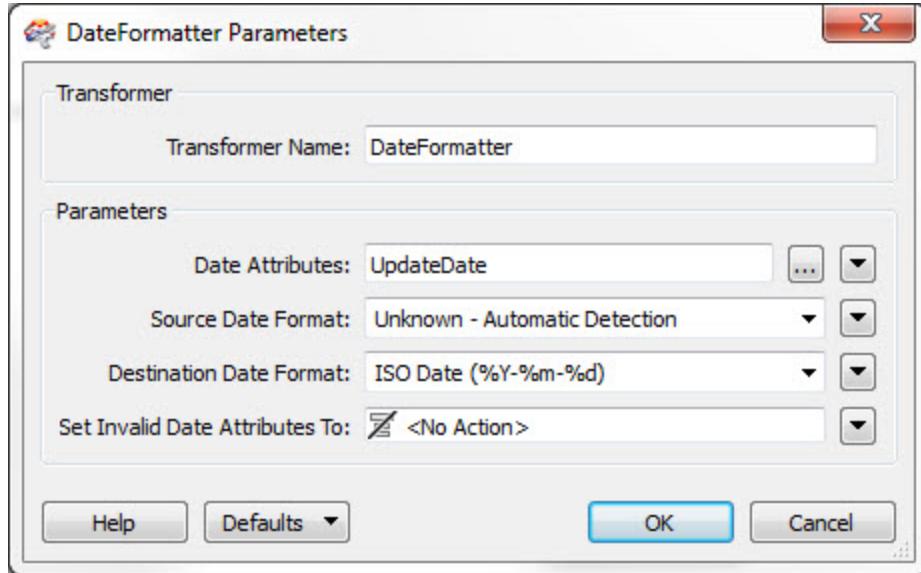
Add a DateFormatter transformer after the AttributeCreator. The DateFormatter can handle relative values - such as "yesterday" or "3 weeks ago" (see the documentation for a full list) - and so here we can use it to convert "TODAY" into a proper date value.



Open the parameters dialog for the DateFormatter.

Select UpdateDate as the date attribute to process. Select "Unknown - Automatic Detection" as the source date format.

Select ISO Date as the destination date format as ISO is the format used in XML/GML for times and dates.



Click OK to close the dialog.

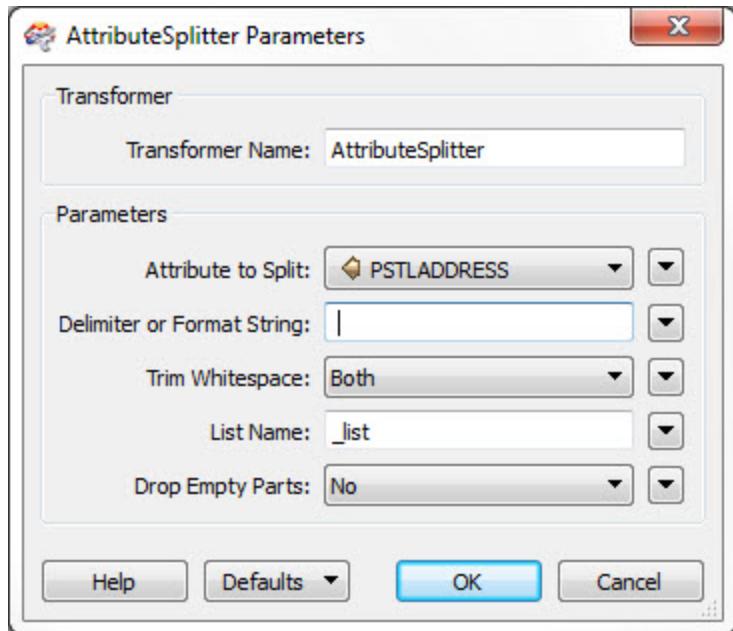


Why not run the translation now, with Redirect to Inspection Application turned on, to see what the result of our efforts so far is?

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. Open the parameters dialog. 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.



Click OK to close the dialog. If you run the workspace now you'll see the address as a list:

_list{0} (encoded: utf-8) 3305

_list{1} (encoded: utf-8) W

_list{2} (encoded: utf-8) 10th

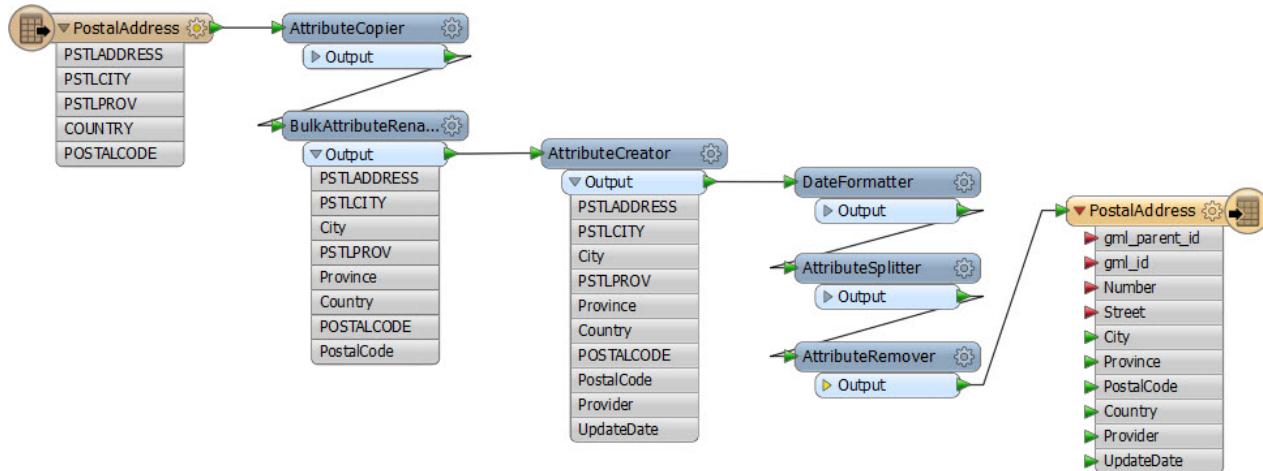
_list{3} (encoded: utf-8) Av

10) Add AttributeRemover

Now the PSTLADDRESS attribute has been split up into a list of items, we no longer need it. To clean up the data let's remove that attribute.

Add an AttributeRemover transformer and select PSTLADDRESS as the attribute to be removed.

The workspace will now look like this:

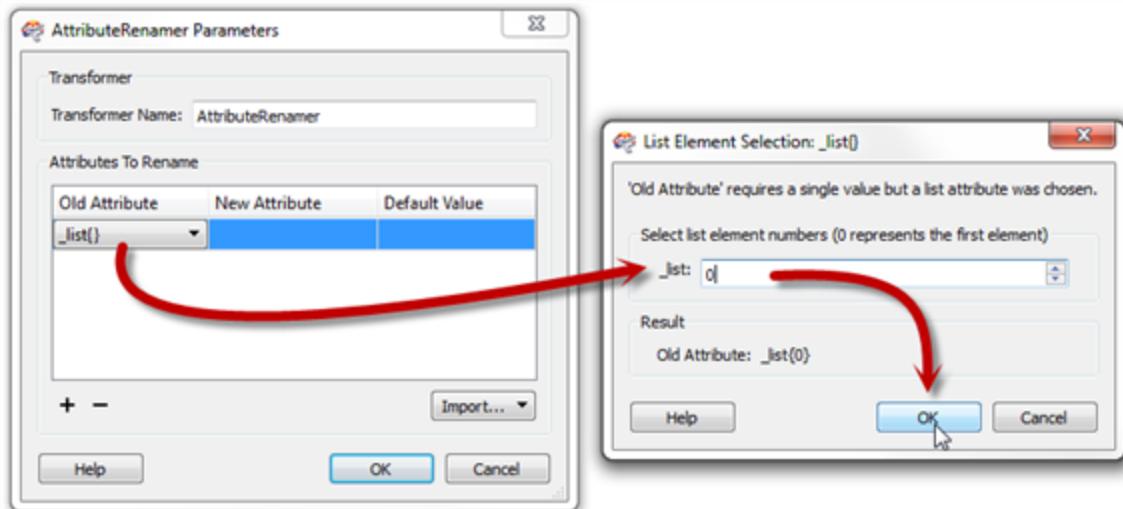


11) Add AttributeRenamer

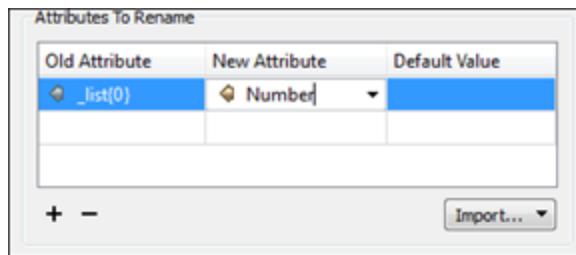
Now let's add an AttributeRenamer to handle the Number field in the output.

Place an AttributeRenamer transformer and open the parameters dialog.

For the "Old Attribute" field select `_list{}` (which is the list of separated values created by the AttributeSplitter). You will be prompted which item in the list is to be used. It will be the first part - entry number 0 - so simply click OK to accept this.



Now for the "New Attribute" field select Number from the drop down list



Click OK to close the dialog.

12) Add StringConcatenator

The final step is to recreate the Street attribute, without it being prefixed by the address number.

Place a StringConcatenator transformer and open the parameters dialog.

For the Destination Attribute field enter the name Street

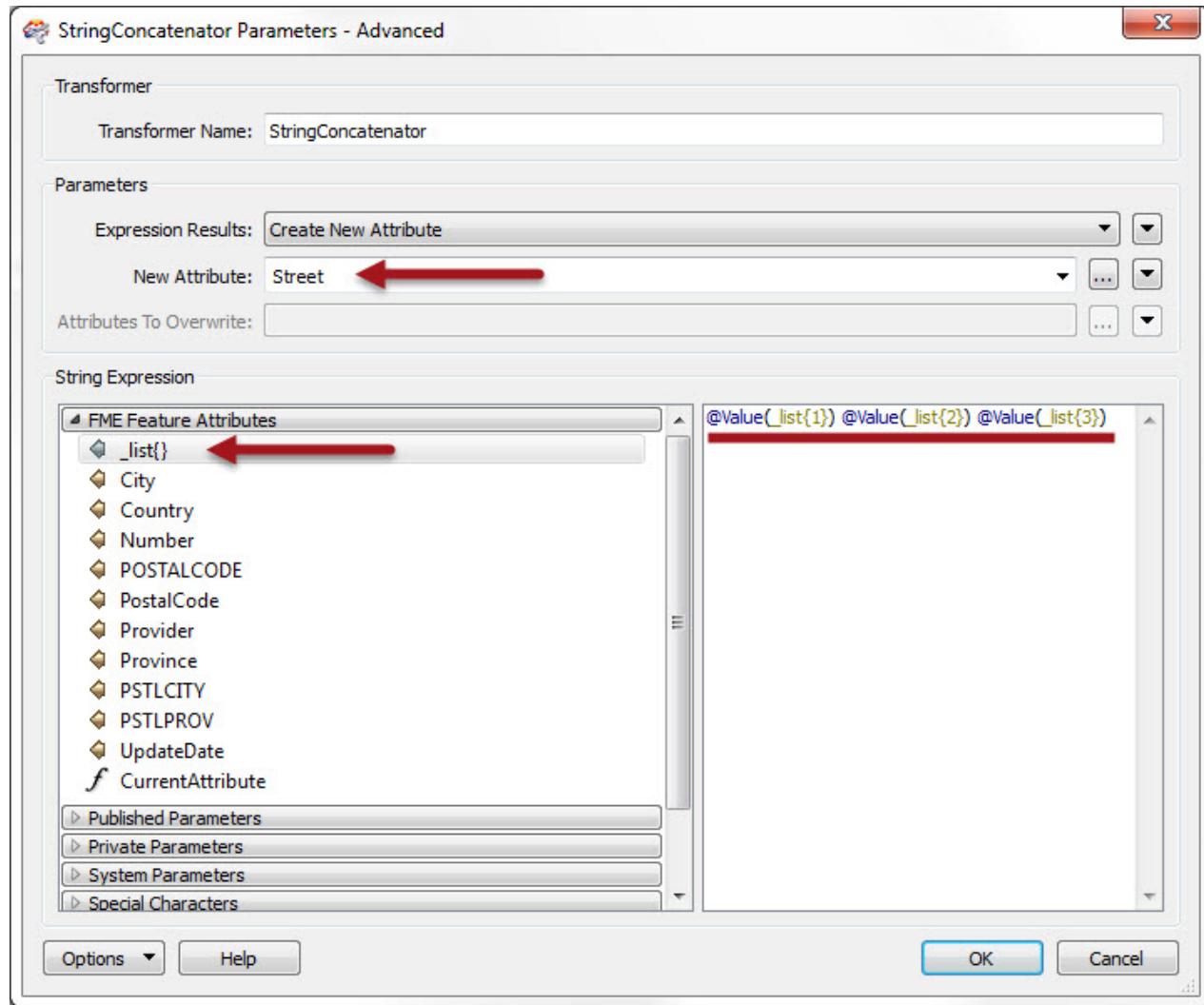
In the Expression field double-click `_list{}` to select a list item. When prompted select list item 1.

Press the spacebar to enter a <space> character.

Double-click `_list{}` again and this time select list item 2.

Again press the spacebar and double-click `_list{}` again, this time selecting item 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:

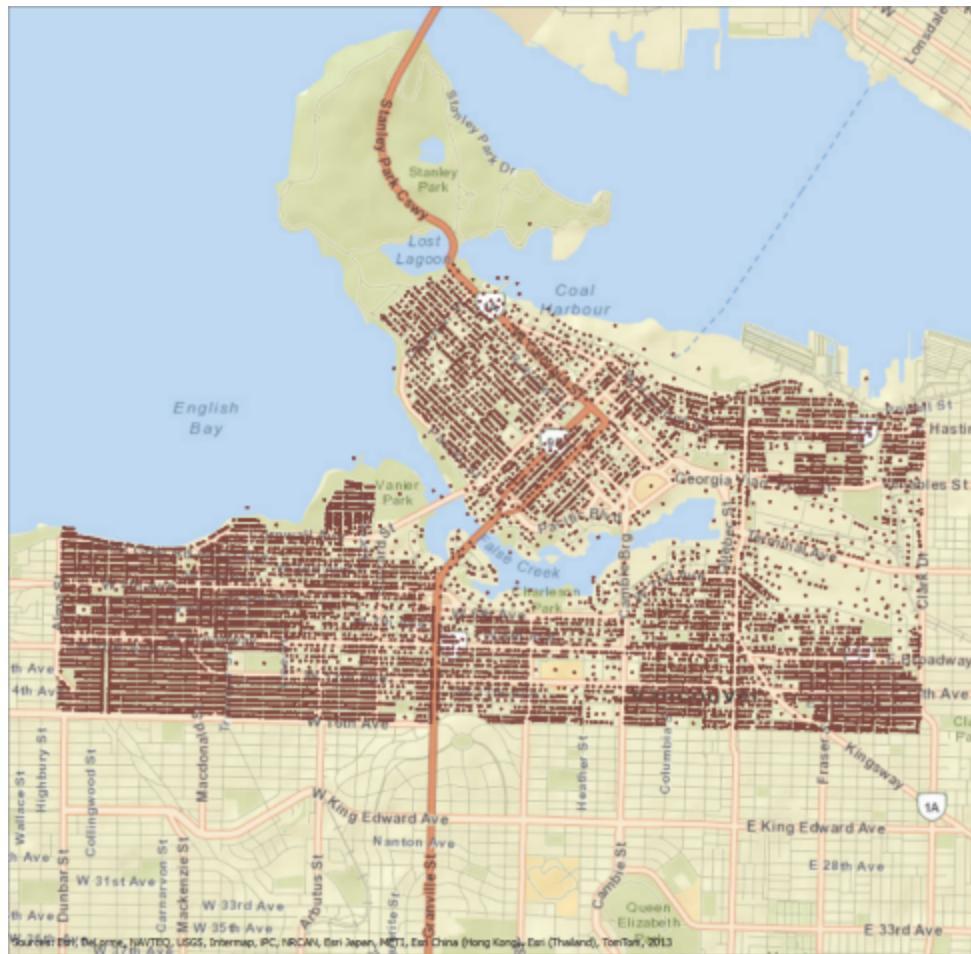
"W"+"17th"+"St" becomes "W 17th St"

We're assuming that no street name has more than three parts to it, but that's reasonable for our example.

13) Run Workspace

Save and then run the workspace.

Open the containing folder to check the output has been written. There should be both an output file and schema document. Inspect the data in the FME Data Inspector. The output (with a background map turned on) should look like this:



ArcGIS Online Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, META, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

Querying a feature will return all of the attributes we've just put together:

Feature Information	
Features Selected: 1 of 1	
Property	Value
Feature Type	PostalAddress
Coordinate System	EPSG:26910
Dimension	2D
Number of Vertices	1
Min Extents	490239.36229999998, 5458809.3531999998
Max Extents	490239.36229999998, 5458809.3531999998
▲ Attributes (13)	
City (encoded: utf-16)	Vancouver
Country (encoded: utf-16)	Canada
fme_geometry (string)	fme_point
fme_type (string)	fme_point
gml_id (encoded: utf-16)	id03a57038-789b-4e6d-aac0-a4a774bd6de3
gml_original_coordinate_system (encoded: utf-16)	EPSG:26910
Number (encoded: utf-16)	1224
PostalCode (encoded: utf-16)	V6E9J0
Provider (encoded: utf-16)	Safe Software
Province (encoded: utf-16)	British Columbia
Street (encoded: utf-16)	Davie St
UpdateDate (encoded: utf-16)	2015-01-28
xml_type (string)	xml_point
▲ ● IFMPoint	490239.36229999998, 5458809.3531999998
Name (encoded: utf-16)	pointProperty



Congratulations! You have now:

- Used various transformers to manage attributes and map a schema

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.

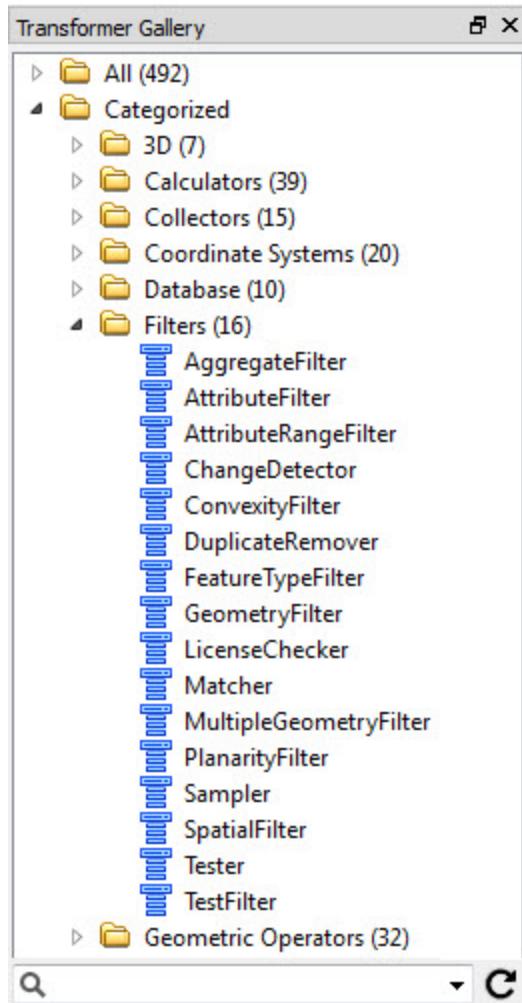
A filtering transformer may be part of the Filter category (such as the *ChangeDetector* transformer) or the definition can be stretched to include any transformer with multiple output ports, such as the *Sampler* transformer, which creates a sample selection of data and separates it out through Sampled and NotSampled output ports.

However, *Conditional* filtering is where the decision about which features are output to which connection is decided by some form of 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

The transformers in the Filter category are the main group that 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*.

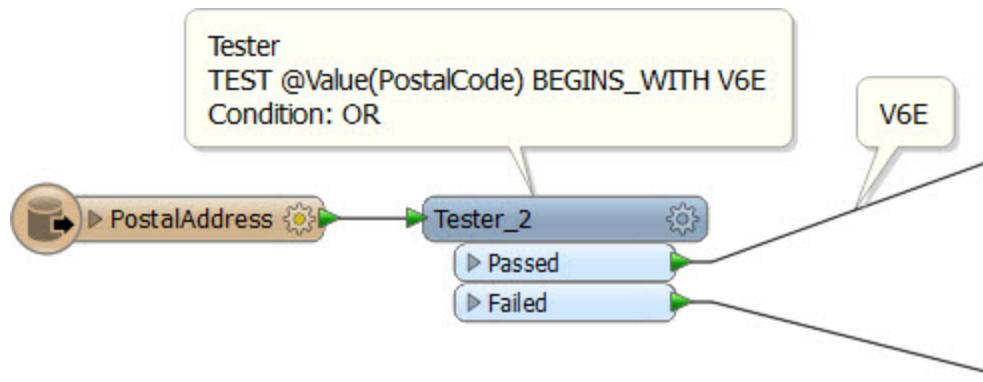


The *Tester* and *TestFilter* Transformers

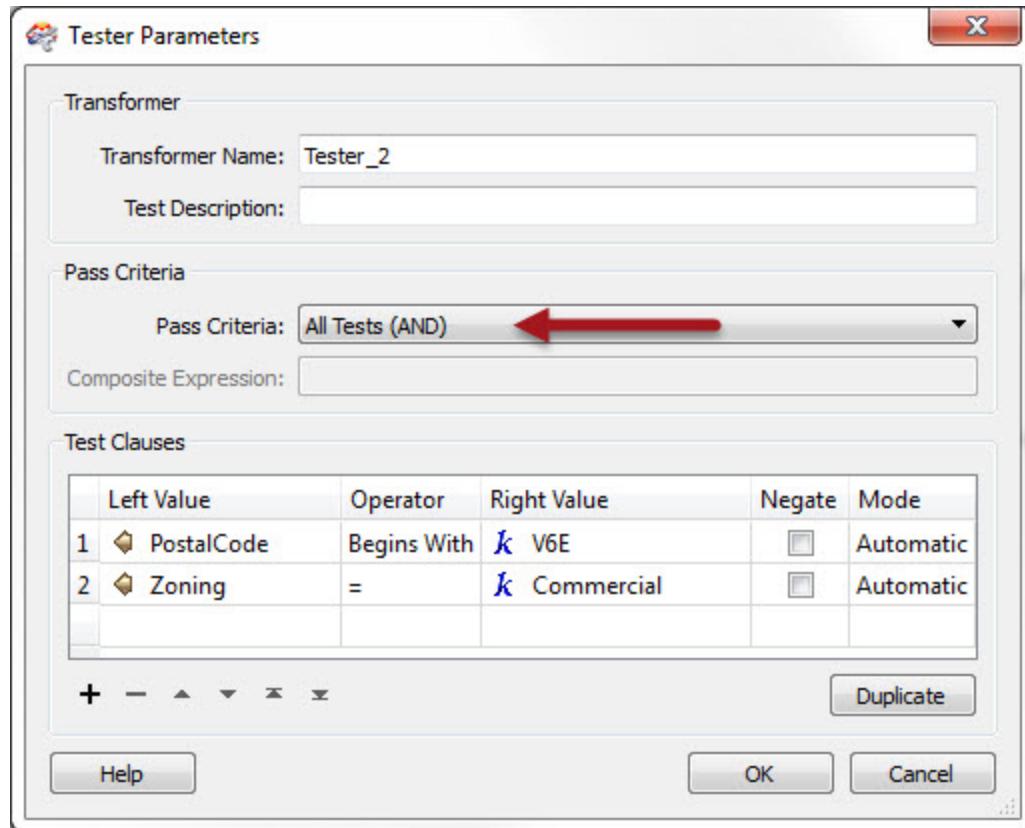
The *Tester* transformer is the key transformer for conditional filtering of data.

Tester

Not only can the *Tester* (number 1 in the top 25) carry out single tests, it allows the combination of multiple tests, where a user can combine any number of clauses using an AND and OR statement.



This *Tester* simply filters address features, depending on whether they are part of the V6E postal code. It also demonstrates the usefulness of annotating a workspace!



This Tester is using the AND pass criteria. Here all clauses must apply for a feature to pass the test. In this case an address must be in postal code V6E **and** also be a commercial property.

Non-commercial properties in V6E, or commercial properties in a different postal code, will fail this test.

Besides the usual operators, there are also some based on a SQL where clause. These include:

- *In*
- *Between*
- *Like*
- *Matches Regex*
- *Contains*
- *Begins With*
- *Ends With*

...plus there are other tests that test for the existence of attributes and values:

- *Attribute is Null*
- *Attribute is Empty String*
- *Attribute is Missing*

Operator	Right Value
=	
!=	
<	
>	
<=	
>=	
In	
Between	
Like	
Matches Regex	
Contains	
Begins With	
Ends With	
Attribute Is Null	
Attribute Is Empty String	
Attribute Is Missing	

TestFilter

The *TestFilter* (#12 in the top 25) is essentially a way to combine multiple *Tester* transformers into one.

Because of this, it's equivalent to a whole string of *Testers*, and similar to the CASE or SWITCH command in programming or scripting languages. Instead of the *Tester*'s single Passed and Failed ports, you can create an output port for each condition (it does not need to be called "Passed") and an output port for features that fail all of the test conditions.

The *TestFilter* has the full set of operands available with the *Tester* such as equals, greater than, less than, and so forth. Each condition is tested in turn.

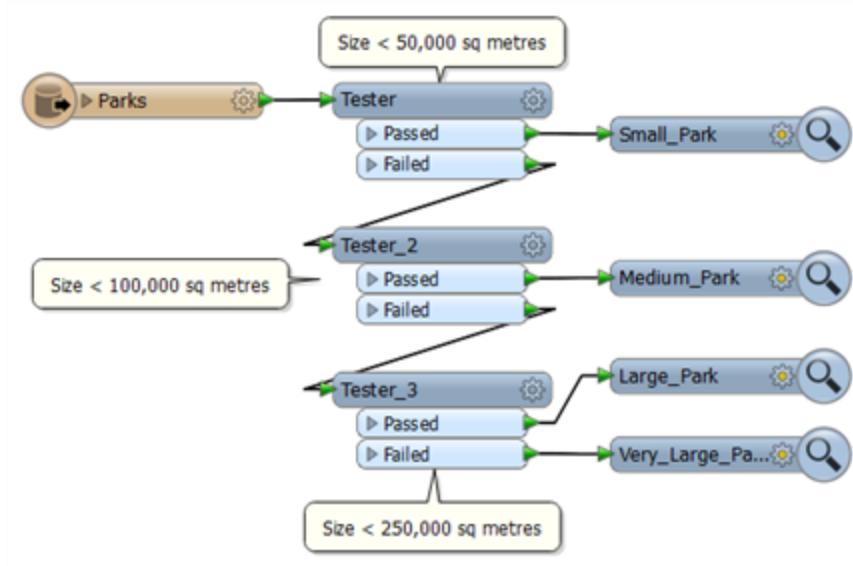
Features that pass are output through the matching output port. Features that fail are sent on to the next condition in the list. Therefore it's very important to get the conditions in the correct order.

TestFilter also lets the output ports be individually named, which is very convenient.

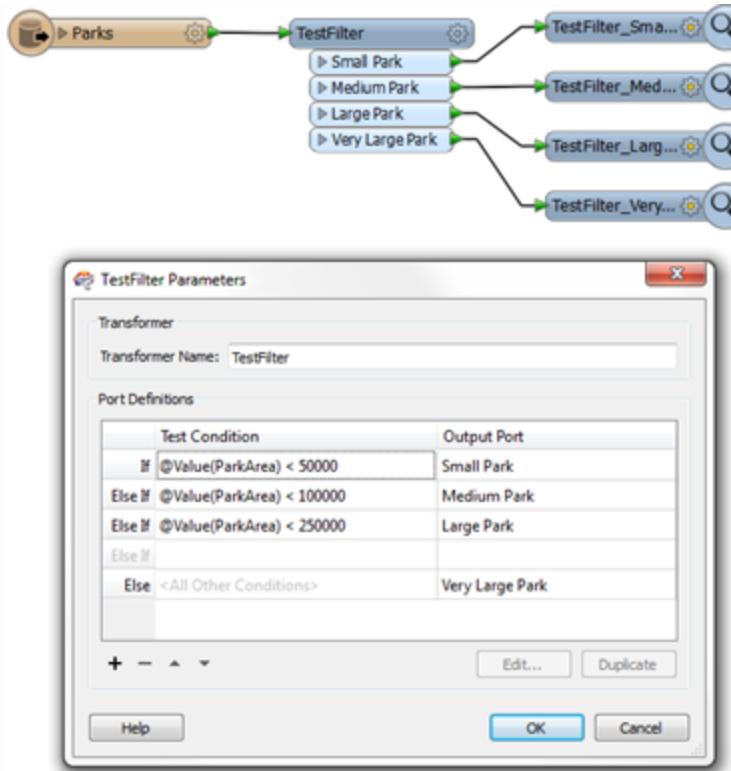


Because the TestFilter can operate as a single Tester transformer would, it's possible to replace all instances of the Tester in a workspace with a TestFilter.

In this example the user has used three *Tester* transformers (and six connections) to filter out the different parts of this data.



With the *TestFilter*, the three Testers are now replaced with one single transformer and there are only four connections.



Also notice how the *TestFilter* output ports have custom naming. This is another advantage to this transformer.

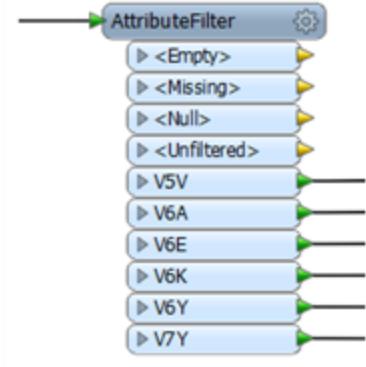
Other Key Filter Transformers

The *Tester* and *TestFilter* are not the only useful filter transformers.

AttributeFilter

The *AttributeFilter* transformer (#5 in the top 25) directs features on the basis of values in a chosen attribute.

In this example features are divided into different postal codes depending on the value of a *PostalCode* attribute.



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?

Confused from Interopolis

Dear Confused,

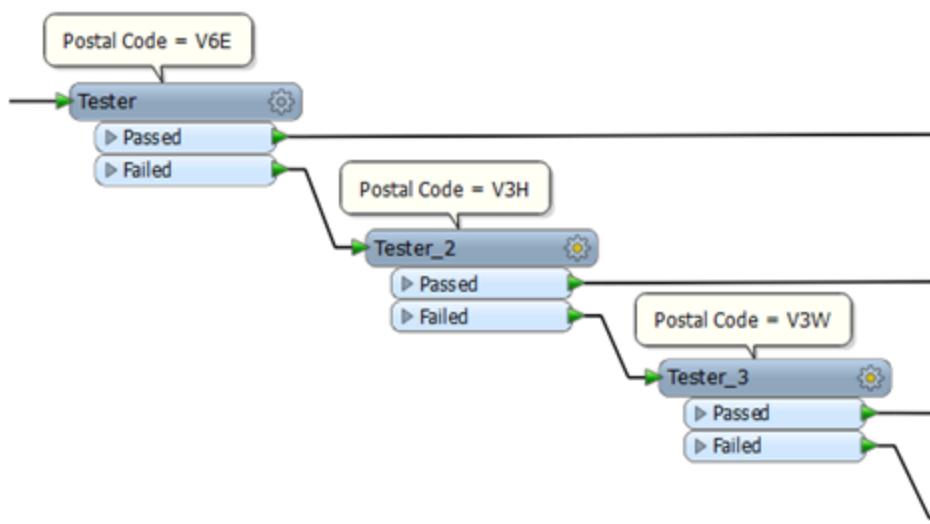


The Tester is best for simple yes/no decisions, such as "Heads or Tails" or "Are you 21 years or older".

The AttributeFilter is best for testing many values for a single attribute; for example "Is your hair color Black, Brown, Red, Grey, or Another Color."

The TestFilter is best for filtering a feature by a set of cascading conditions; for example to play in the FME golf tournament you must pass one of these conditions: "Are you a previous champion," "Have you won another tournament in the last 5 years," "Were you in the top 16 in last year's tournament," "Are you in the top-30 player rankings," etc.

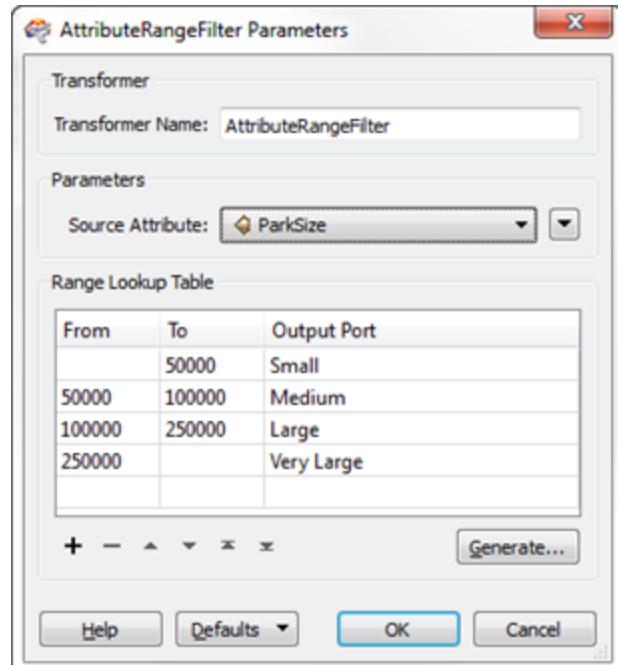
If your workspace looks like this then an *AttributeFilter* transformer might be a better option.



If the conditions were mathematical (for example, size > 100), then the *TestFilter* would be the best choice.

AttributeRangeFilter

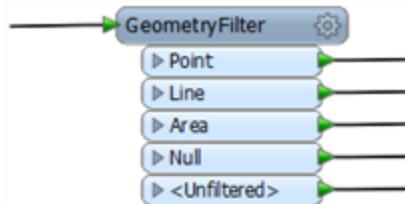
The *AttributeRangeFilter* carries out the same operation as the *AttributeFilter*, except that it can handle a range of numeric values instead of just a simple one-to-one match.



The *AttributeRangeFilter* parameters dialog has the option to generate ranges automatically from a set of user-defined extents.

GeometryFilter

The *GeometryFilter* (#11 in the top 25) directs features on the basis of geometry type; for example, point, line, area, ellipse.



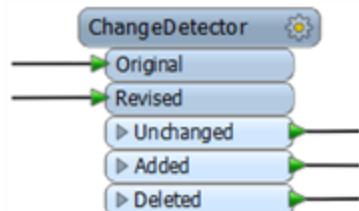
The *GeometryFilter* is useful for...

- Filtering out unwanted geometry types; for example removing text features before using an *AreaBuilder* transformer
- Dividing up geometry types to write to separate destination Feature Types; for example, when writing to a geometry-restricted format such as Esri Shape

ChangeDetector

The *ChangeDetector* detects changes between two sets of input features and routes data based on the results of this comparison.

'UNCHANGED' features are those present in both the original and revised streams. 'ADDED' features only exist in the revised stream and 'DELETED' features only exist in the original stream.



Exercise 5b Conditional Filtering

Scenario	FME user; City of Interopolis, Planning Department
Data	Addresses (File Geodatabase), Zoning (MapInfo TAB) Roads (AutoCAD DWG)

Overall Goal	To find all residential addresses within 1000 feet of an arterial highway.
Demonstrates	Conditional Filtering
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Exercise5b-Complete.fmw

Because noise control laws are being amended in the City of Interopolis, residents living in a single-family residence within 50 metres of a major highway must be contacted to inform them of these changes.

It is your task to find all affected residential addresses.

1) Start Workbench

Start Workbench and create a new empty workspace.

2) Add Reader (Address data)

Use **Readers > Add Reader** to add a reader for address data.

Reader Format Esri Geodatabase (File Geodb API)
Reader Dataset C:\FMEData2015\Data\Addresses\Addresses.gdb

Reader Parameters

Table List PostalAddress

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:\FMEData2015\Data\Zoning\Zones.tab

4) 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:\FMEData2015\Data\Transportation\Roads.dwg

When prompted, select only the feature type (layer) called Arterial. The workspace will now look like this:



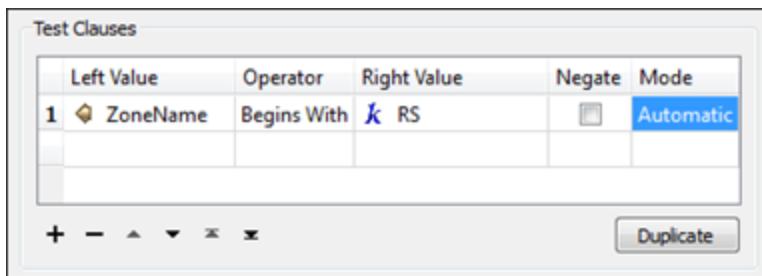
Don't forget to inspect all of the source data to familiarize yourself with the contents.

5) 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:



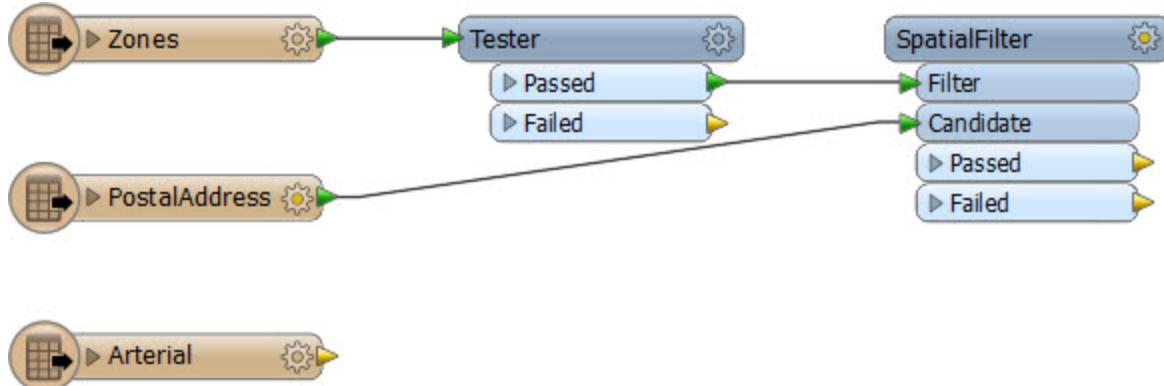
The important thing is to set up the tests using the "Begins With" predicate.

6) 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 feature type PostalAddress to the SpatialFilter:Candidate port

Connect the Tester:Passed output to the SpatialFilter:Filterport.

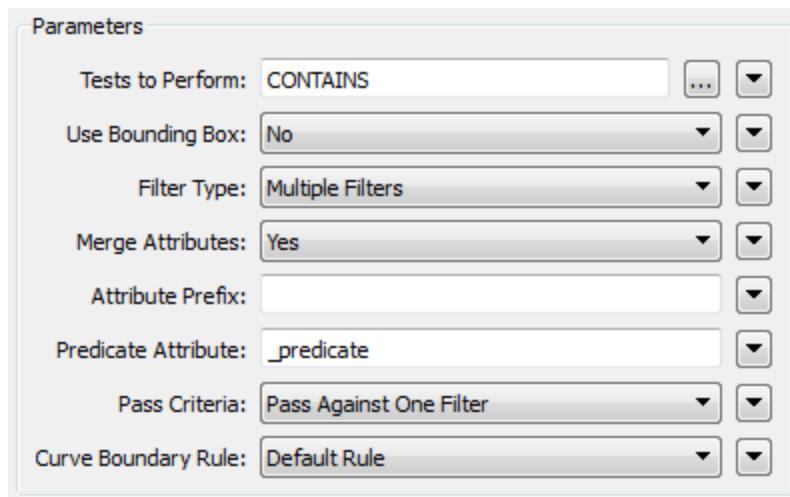


7) Set up SpatialFilter

Open the *SpatialFilter* parameters dialog. Set up the parameters.

Tests to Perform should be set to CONTAINS (i.e. find addresses that the zones contain). The Filter Type should be set to Multiple Filters (as there are multiple zoning areas). However, the Pass Criteria parameter should be “Pass Against One Filter.”

This parameter is very important. The test will fail if it is not set correctly, as a single address cannot be in ALL zones.



8) Add Inspectors

Now let's test what we have so far. Add *Inspectors* to both *SpatialFilter* output ports, and the Tester:Passed port.

9) Save and Run Workspace

Save the workspace. 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 features will fall inside these areas.

10) 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:



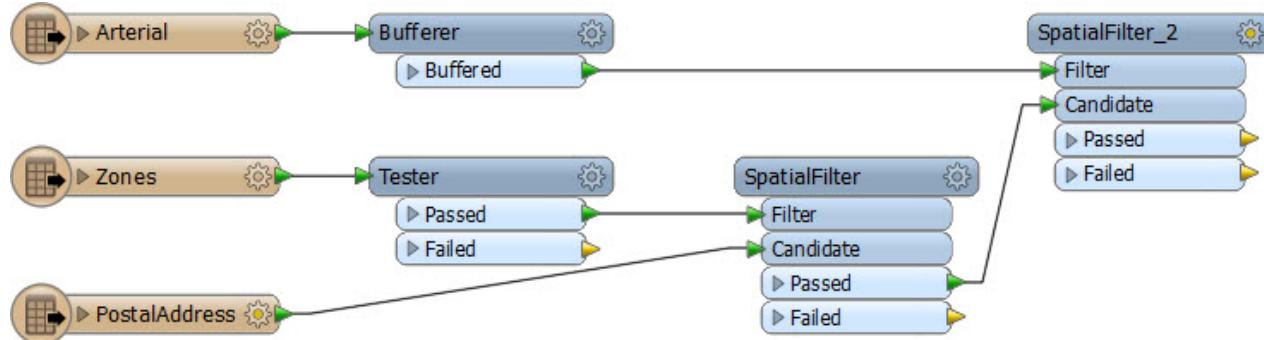
Now open the parameters dialog for the Bufferer. Set the buffer amount to be 50.



Optionally you can add a Dissolver transformer after the Bufferer, to merge all the buffer features together. The output will be the same (in terms of addresses selected) but it will look nicer in the FME Data Inspector.

11) Add SpatialFilter

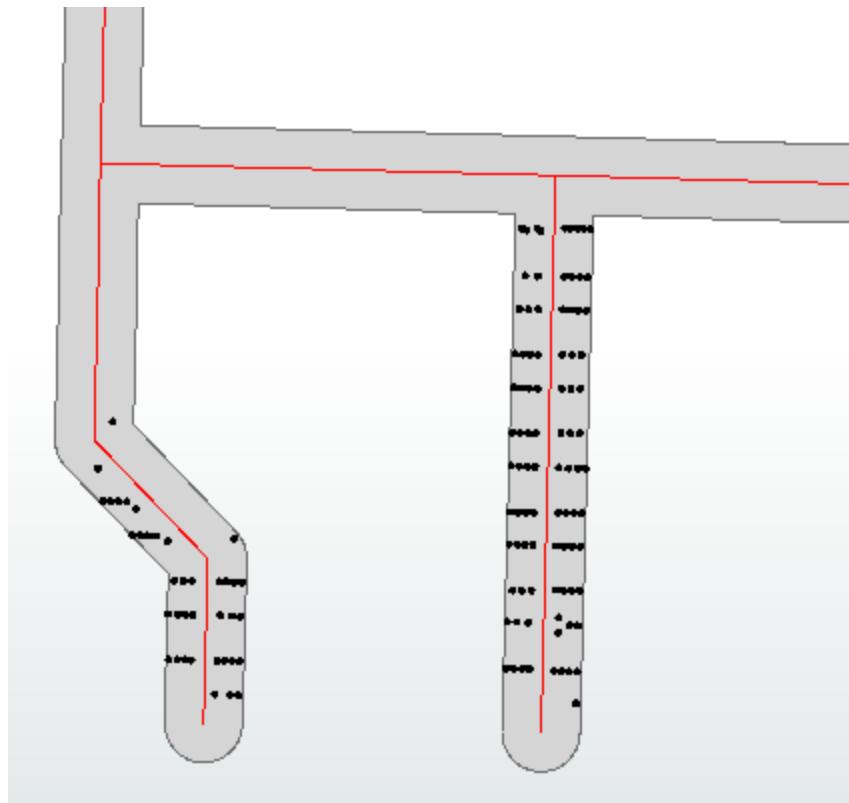
Add a second SpatialFilter transformer. The buffered arterial routes are the Filter. The pre-filtered addresses are the Candidates:



As before, open the parameters dialog and change the settings to be an overlap against multiple filters.

12) Run Workspace

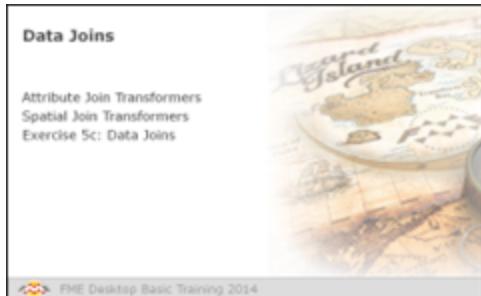
Attach some Inspector transformers to show you the output from various transformers. Save and run the workspace. The output (zoomed in) should look like this:



Congratulations!

You have now used filter transformers to filter data depending on different sets of conditions.

Data Joins



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.

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.

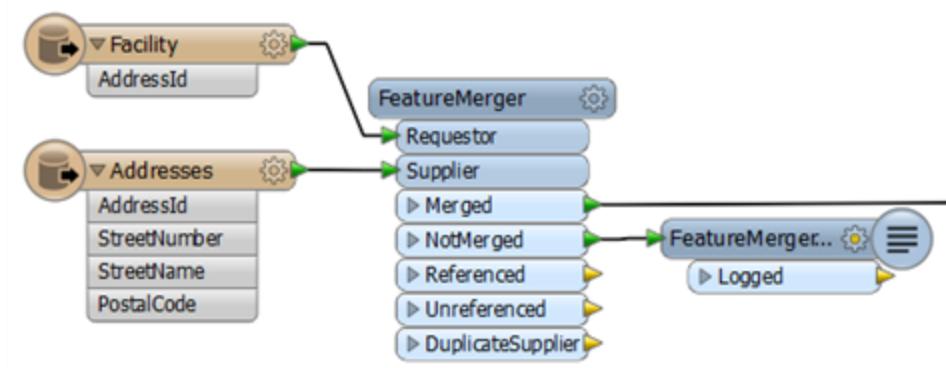
Joins in FME can either be based on two matching attribute values, or they can be based on a spatial relationship such as an overlap between features or proximity from one feature to another.

Attribute Join Transformers

These are transformers that join data on the basis of a matching attribute value.

FeatureMerger

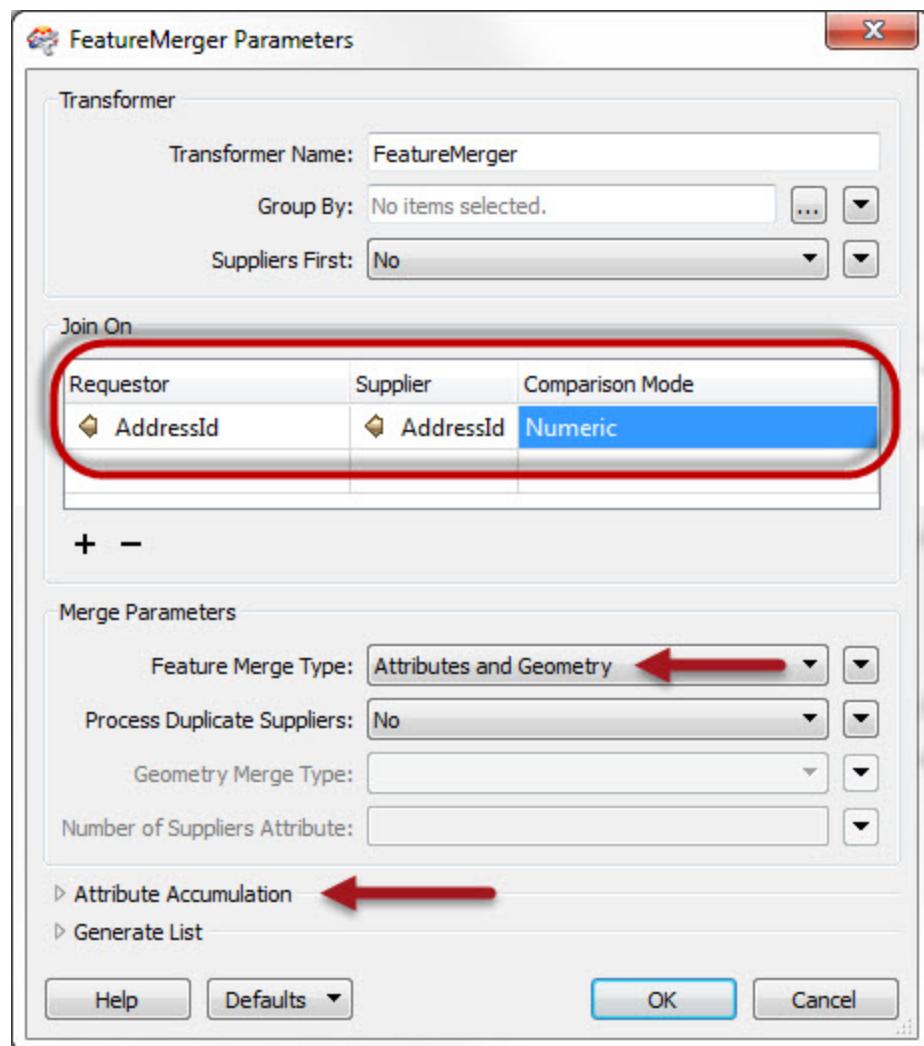
The FeatureMerger is the primary transformer for joining two streams of data within a workspace. This is achieved on the basis of one or more matching attribute values (keys).



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 NotMerged port. Here facilities that did not have a matching AddressId are sent to a Logger 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:



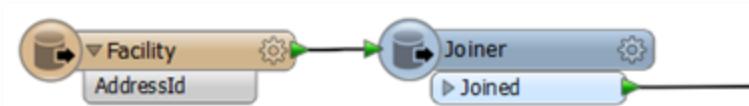
Notice how the AddressId attribute from each set of data is being used to merge the features together. Also notice that the merge can (and in this case does) include both attributes and geometry; i.e. the "requestor" can be updated with the geometry of the "supplier."



Notice that there is a new "Number of Suppliers Attribute" parameter that lets you record the number of suppliers that were matched to any one requestor.

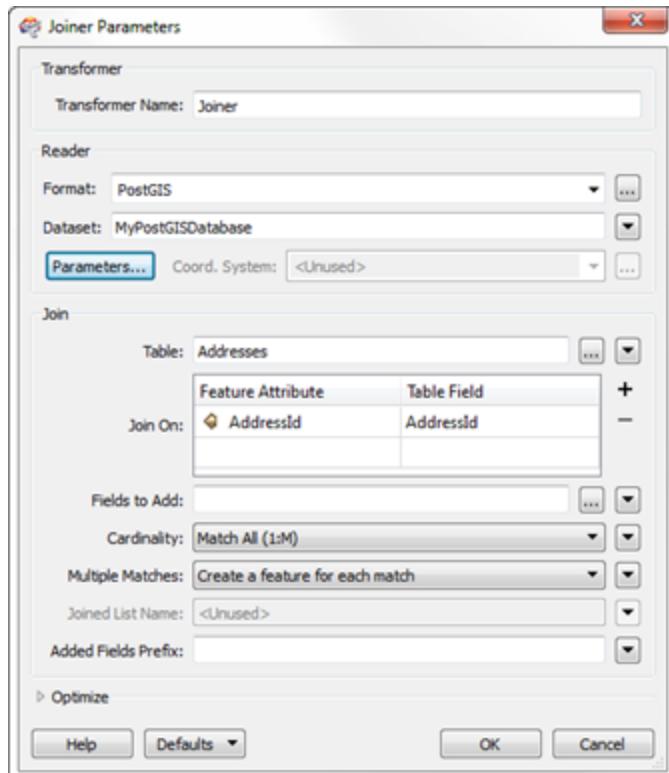
Joiner

The Joiner 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.



Here is the same example as for the FeatureMerger above. In this case the facilities features are obtaining address data directly from an address database. Therefore the database does not need to be read as a stream of features in the FME workspace.

The parameters dialog for the Joiner looks like this:



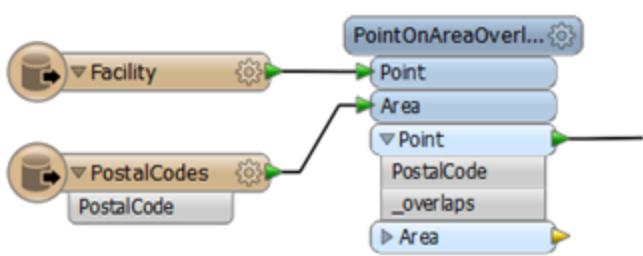
Again, AddressId is being used to facilitate a merge between the two sets of data. However, this transformer is a little more sophisticated and has parameters to control how multiple matches are handled, as well as parameters for optimizing the database query.

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.

Overlays

There are a number of different "overlay" transformers, each handling a different form of overlay. For example the PointOnAreaOverlay 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.*"



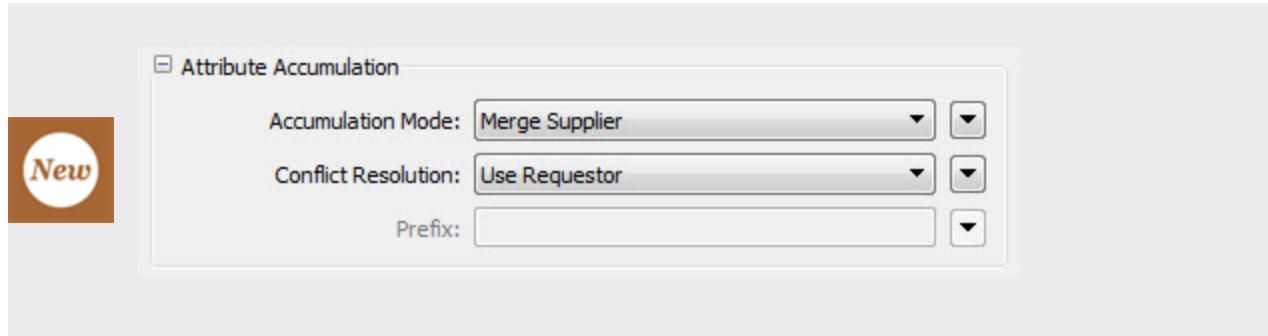
Here the facility features are being provided with a postal code depending on which postal code polygon feature they fall inside.

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.

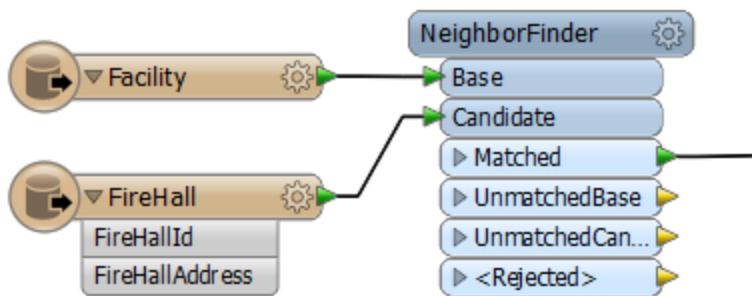


In FME 2014 transformers that merge data together have been given a new set of standardized parameters, to make it simpler to decide how to merge attributes. You can choose for the Supplier to overwrite the Requestor, or vice versa, and can also decide what to do when there is a clash of attribute names.



NeighborFinder

The NeighborFinder transformer carries out a spatial join based on a proximity relationship.

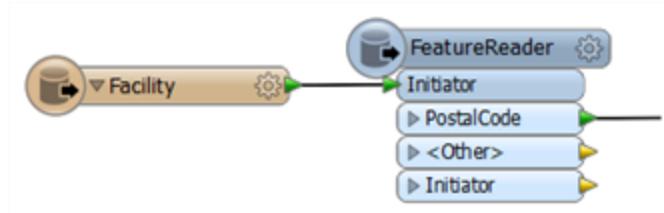


Here the NeighborFinder is being used to identify the closest fire hall to each facility. The FireHallId and FireHallAddress attributes will be merged onto each Facility feature along with a number of useful attributes recording the X/Y coordinate, direction, and distance of the closest fire hall.

The parameters of the NeighborFinder includes the ability to specify a maximum distance for the relationship.

FeatureReader

The FeatureReader is the spatial equivalent of the Joiner transformer. It reads from an external database and forms a match based on a spatial relationship between the initiating feature and features in the database.



Here the FeatureReader is being used to carry out the same overlay of facility and postal code, but the postal code information is being stored in a separate database and queried to retrieve data based on the spatial relationship with the facility.

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.

Data Joins

Exercise 5c Data Joins	
Scenario	Mapping Crime Statistics
Data	Roads (AutoCAD DWG) Crime Statistics (CSV)
Overall Goal	Carry out a join between Crime Statistics and City Blocks
Demonstrates	Attribute-Based Data Joins
Start Workspace	None
Finish Workspace	<p>C:\FMEData2015\Workspaces\DesktopBasic\Exercise5c-Complete.fmw</p> <p>C:\FMEData2015\Workspaces\DesktopBasic\Exercise5c-Complete-Advanced.fmw</p>

You have been handed a set of crime statistics in CSV (Comma Separated) format and asked to create a map from them. In order to map the data you will need to merge it onto a set of spatial features, such as city blocks.

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:\FMEData2015\Data\Emergency\Crime2011.csv

Parameters

File Has Field Names

Yes (Checked)

Lines to Skip: Header

1

The data will look like this in the Data Inspector Table View window.

Table View

CrimeID	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

CSV

Notice how there is a block number, but no other spatial information.

2) Inspect Source Data (Roads)

Now let's inspect some spatial data that we can merge the crime statistics onto. Open the following dataset in the Data Inspector

Reader Format

Autodesk AutoCAD DWG/DXF

Reader Dataset

C:\FMEData2015\Data\Transportation\Roads.dwg

Parameters

Group Entities By *Attribute Schema*

You'll see that the roads dataset has a block number that is an almost identical match to the crime figures; the only difference is that the road block attribute is (for example) "2400" whereas the crime data is "24XX."

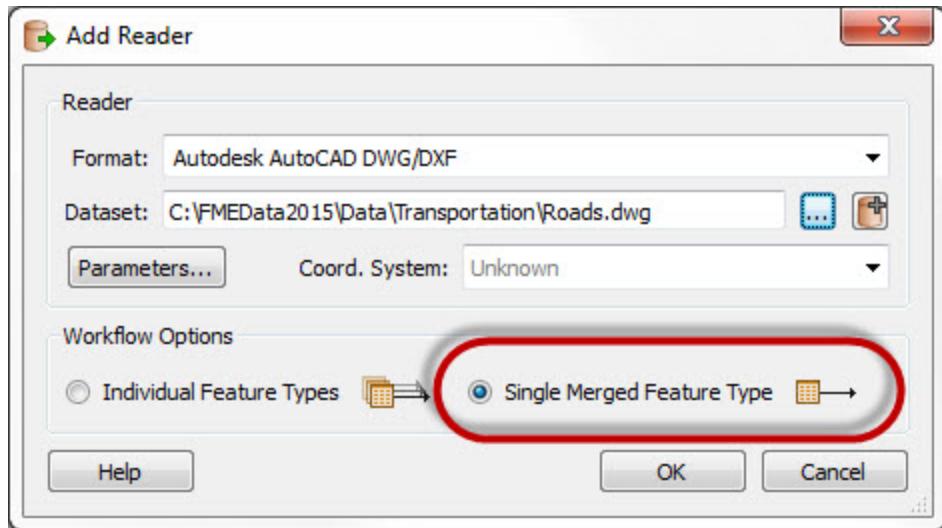
The other difference is that the road data is stored in Title case ("Bute St") in the roads dataset, whereas the crime dataset is upper case ("BUTE ST").

3) Add Readers

Now let's start working with this data. Start FME Workbench and begin with a blank canvas.

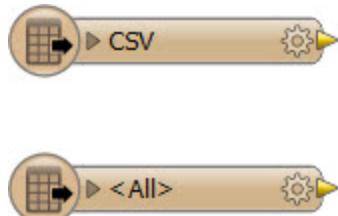
Add a Readers to the workspace using **Readers > Add Reader** from the menubar. This Reader should be used to read the crime (CSV) data. Be sure to use the same parameters as specified for the Data Inspector.

Now add a second Reader to read the roads (DWG) data. However, from inspecting the data, I see that the roads are stored on a set of seven different layers. We don't need that, so set the "Single Merged Feature Type" parameter when adding the Reader, like so:



Again, be sure to use the same Group Entities By parameter as specified for the Data Inspector.

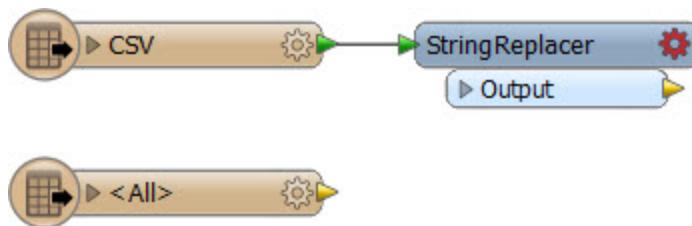
The workspace will now look like this:



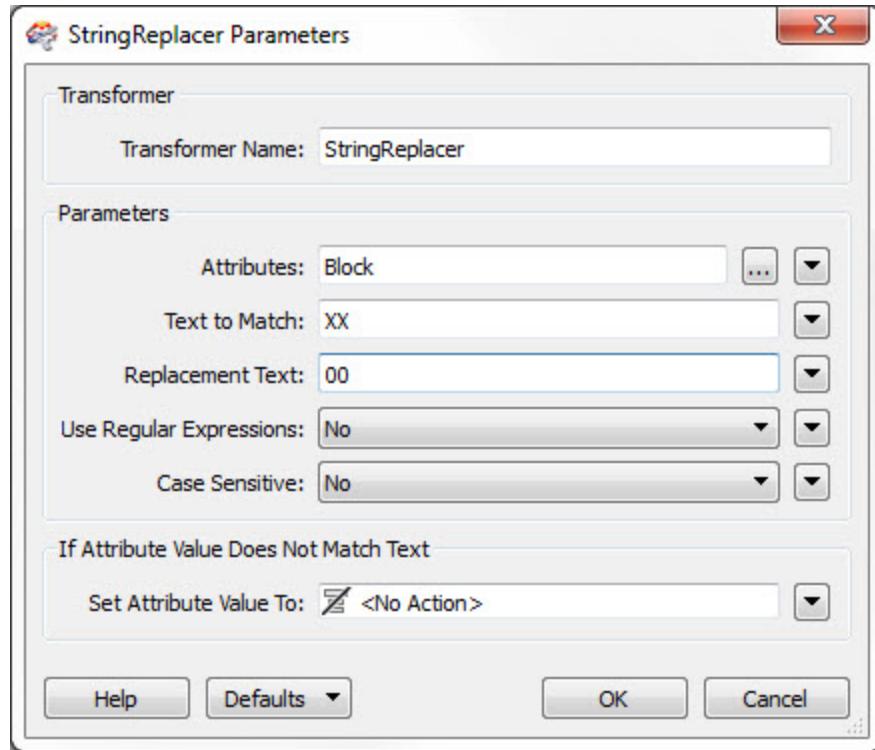
4) Add StringReplacer

To merge the data we need a common block attribute. The current difference in how the block is structured ("00" vs "XX") can be fixed very simply with a StringReplacer transformer.

Add a StringReplacer transformer and connect it to the Crime dataset feature type.



Open the parameters dialog for the StringReplacer. Set the proper parameters. The Attribute to process will be "Block", the Text to Match will be "XX" and the Replacement Text will be "00", like so:

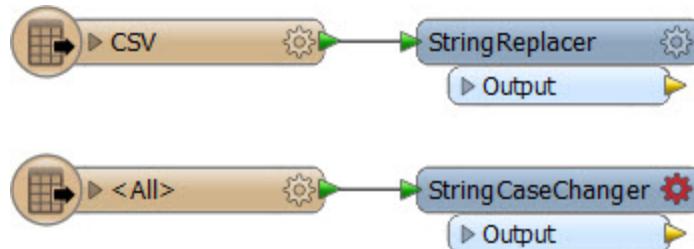


Click OK to accept the values. If you wish, attach Inspector transformers and run the workspace to ensure the transformer is working as expected.

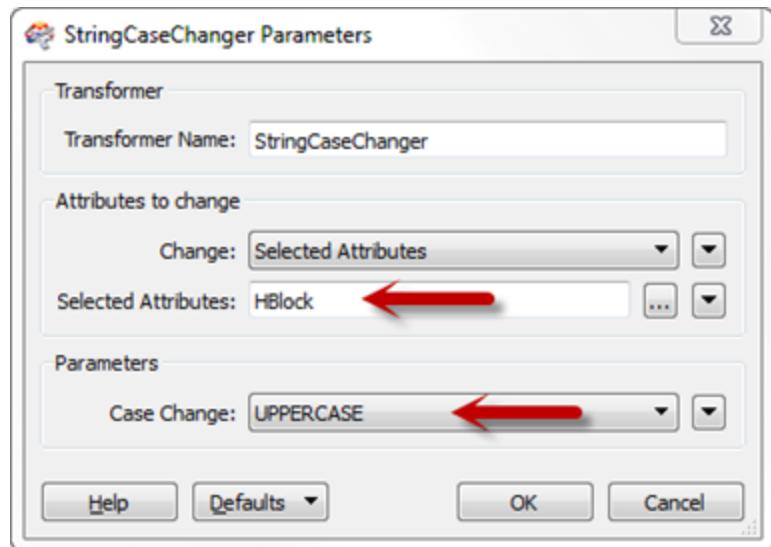
5) Add StringCaseChanger

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 and connect it to the roads dataset feature type:



Open the parameters dialog. Set the parameters to change the values of HBlock to upper case:

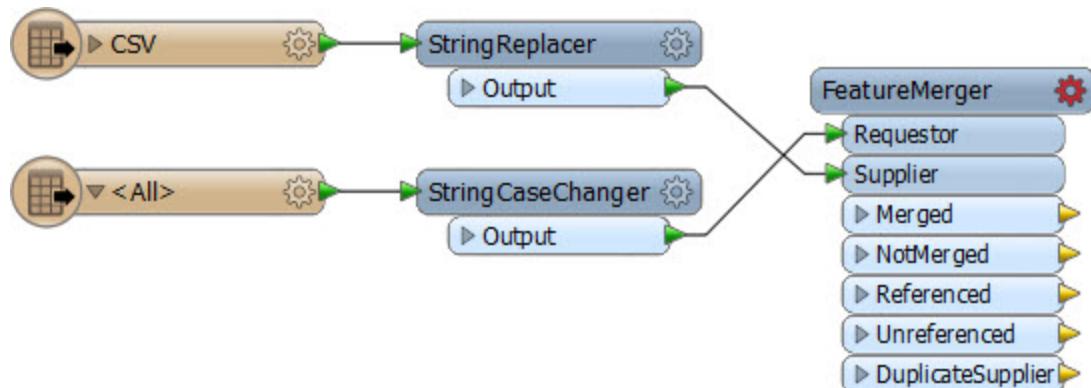


Why are we using the StringCaseChanger transformer here and not the BulkAttributeRenamer (which also transforms items to upper case)? There's an obvious difference between the two - but do you know what it is?

6) Add FeatureMerger

Now we've sorted out the structure of our join keys we can merge the data together with a FeatureMerger.

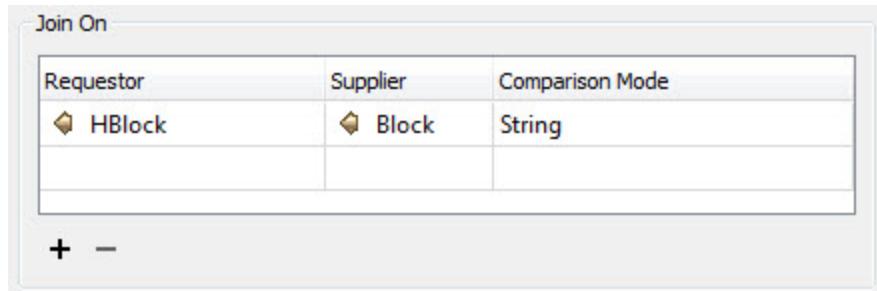
Add a FeatureMerger to the canvas. Connect the roads data as the Requestor and the crime data as the Supplier (we wish to supply the roads data with crime statistics):



If, like me, you don't like connections that cross over, feel free to swap the position of the objects around to untangle it.

Open the parameters dialog for the FeatureMerger.

For the "Join On" set of parameters, the Requestor join attribute will be HBlock, the Supplier join attribute will be Block, and the Comparison Mode will be String:



If you are sharp, you may have noticed that the Requestor and Supplier parameters can be defined within a text edit window.

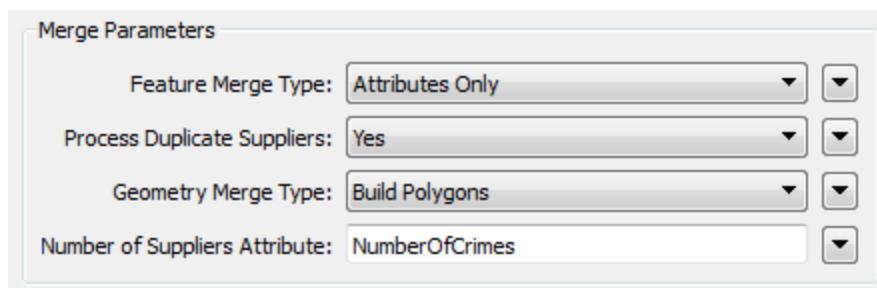
That means, instead of using the transformers we just added, we could replicate their functionality using functions inside that text editor, like so:

`@ReplaceString(@Value(Block),XX,00)
@UpperCase(@Value(HBlock))`

We'll learn more about constructing values like this in the next section of this chapter!



For the "Merge Parameters" 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.



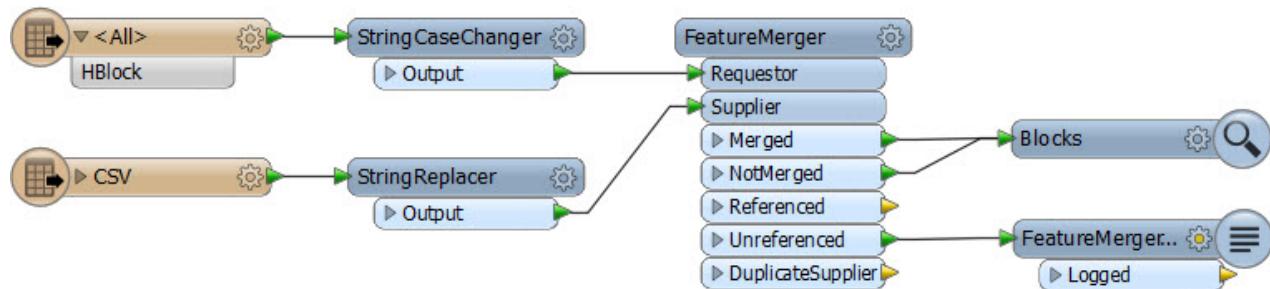
Finally, the attribute accumulation needs to be "Merge Supplier."

Click OK to close the dialog.

7) Add Inspectors

Add Inspector transformers to the Merged and NotMerged output ports (or connect both outputs to one Inspector). This will give us the roads data with crime info attached. The NotMerged data is important because there may be blocks without any crime.

The Unreferenced output port will show us crimes that didn't match to a known address, so connect a Logger transformer here to record any features that fail in this respect.



8) Save and Run Translation

Save the workspace and then run the translation. By querying a block you should be able to see (in the Feature Information window) a list of all the crimes that occurred in that block.



We've now successfully merged crime statistics onto a set of spatial data, which is what the aim of the example was. However, it does leave us in a fairly unsatisfactory state, not having output or processed those figures in any way.

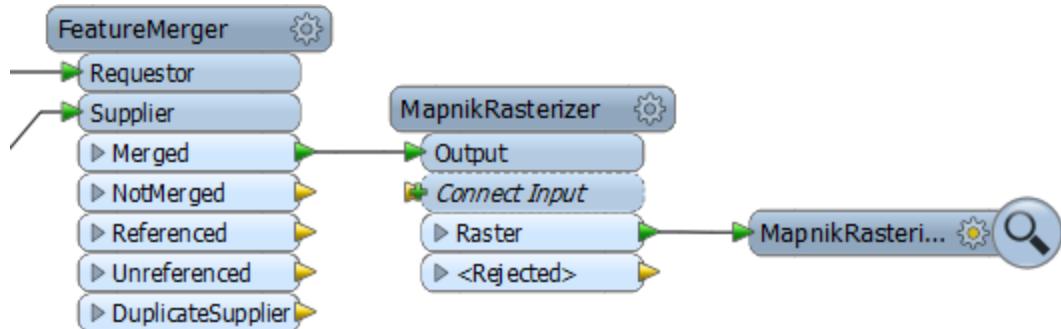
As an advanced task, if you have time, let's try handling the data we've just merged.

9) Add MapnikRasterizer



The MapnikRasterizer is a way of creating nicely formatted raster map output from a set of vector data.

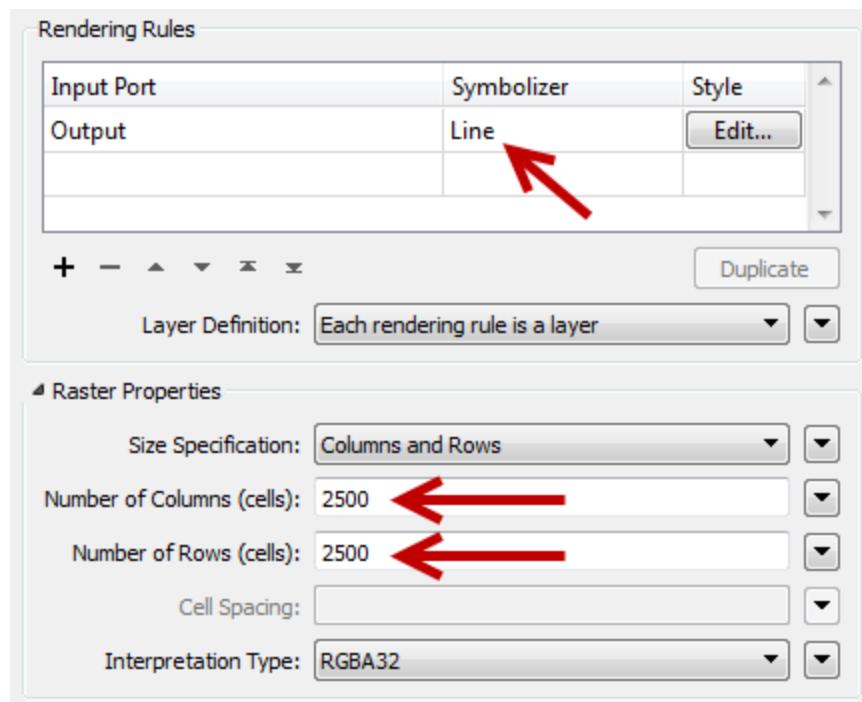
Place a MapnikRasterizer transformer connected to the Merged output port of the FeatureMerger:



Open the parameters dialog. Set the output layer to be a Line symbolizer. Then click the Edit button to the right.

For the Line Width (pixels) parameter click the drop down arrow and select **Attribute Value > Number of Crimes**. Click OK to close the dialog.

Set the output raster size to be 2500 cells for both Columns and Rows.



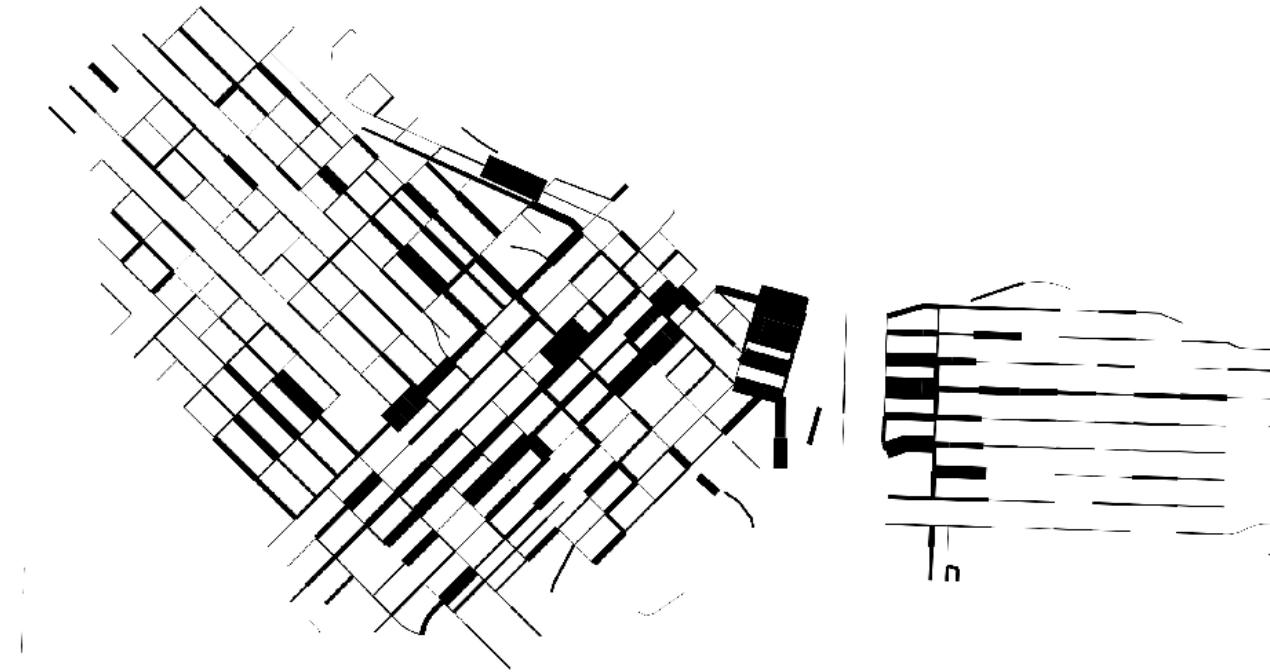
10) Run Workspace

Now you can add an Inspector and run the workspace.



It's a good step to turn off background maps in the Data Inspector first - else there will be a delay while the raster output is reprojected to match.

The output will look like this:



Basically we've created a map where road width reflects the amount of crime on that block.

The image is fairly basic at the moment, but you can enhance it by adding more layers to the MapnikRasterizer (merge in some orthophoto data perhaps) and/or adjusting some of the other parameters like background image.



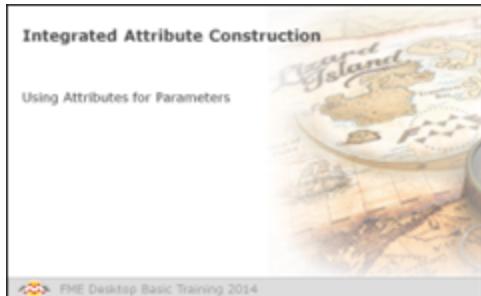
Congratulations! You have now:

- Pre-processed data to get join keys with a matching structure
- Joined non-spatial data onto spatial features using a join key
- Advanced: Created a map output of merged attribute data

Q+A Solution:

Why are we using the StringCaseChanger transformer here and not the BulkAttributeRenamer? Because the BulkAttributeRenamer changes the case of the attribute NAMES! We want to change the case of the attribute VALUES - and that is done with the StringCaseChanger.

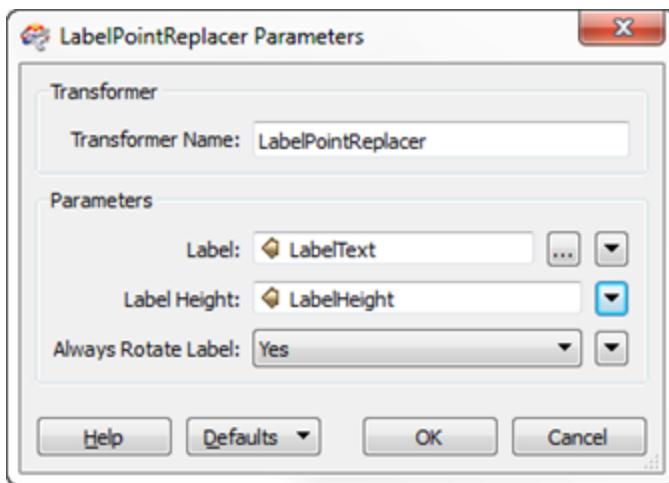
Constructing Transformer Parameters



Besides direct use of attributes FME also allows you to integrate string construction and arithmetic calculations within transformer parameters.

Using Attributes for Parameters

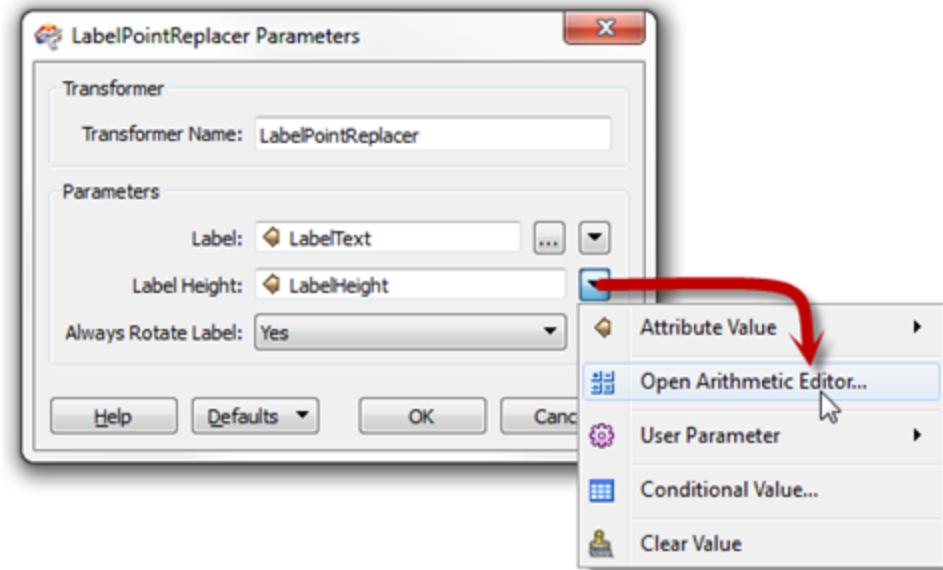
Many transformer parameters allow the user to select an attribute value instead of manually entering a fixed value. For example, the LabelPointReplacer can create a label whose contents and height are specified by attribute values:



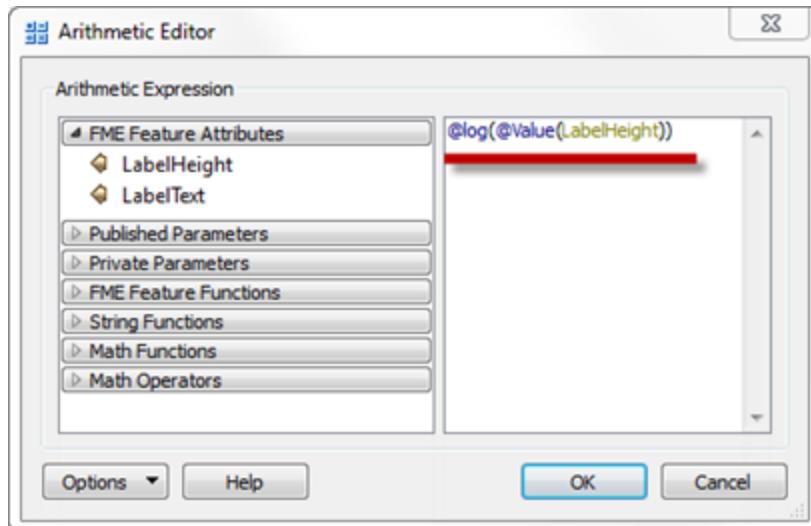
This is very useful because it allows the parameter (here label size) to get a different value for each feature. One feature could create a label 10 units in height, another feature could create one 15 units high, and so on. It is no longer a fixed value.

However, FME takes the concept a step further by integrating string and numeric editors into parameter dialogs, so that values can be not just taken from an attribute, but can be constructed from a mix of attributes and other user inputs.

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 use a Log() function to calculate the height of their labels:



Reducing Workspace Congestion

Integrated text and arithmetic editors provide a great benefit for workspace creation.

Workspaces will be more compact and well-defined when as many peripheral operations as possible are directly integrated into transformers; In other words, when a single transformer can be used instead of a number of transformers used in series.

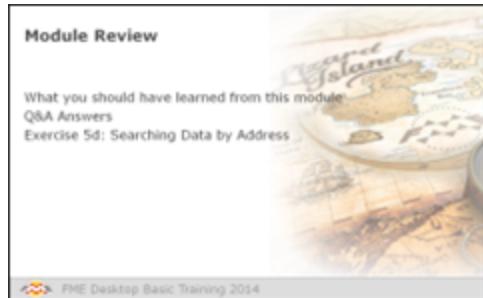
For example, the LabelPointReplacer can calculate label heights without the need for a prior ExpressionEvaluator transformer.

In this way the number of transformers required to carry out a single task is reduced.



It's important to note one particular drawback of the integrated method: you don't get the information as an attribute to use elsewhere. For example, you can't create and use a string AND also have it as an output attribute. For that you would need the AttributeCreator as usual.

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.
- 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
- **Integrated** functionality allows the author to replace support transformers with tools built-into operational transformers.

FME Skills

- The ability to locate a transformer to carry out a particular task, without knowing about that transformer in advance.
- 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
- The ability to build strings and calculate arithmetic values using integrated tools.

Exercise

Now let's prove what you have learned by carrying out an exercise on basic data translation and data inspection.

Practical Transformer Use

Exercise 5d Practical Transformer Use	
Scenario	FME Workspace Author
Data	Various
Overall Goal	Identify information about a given address
Demonstrates	Advanced transformer use
Start Workspace	None
Finish Workspace	<p><i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise5d-Complete.fmw</i></p> <p><i>C:\FMEData2015\Workspaces\DesktopBasic\Exercise5d-Complete-Advanced.fmw</i></p>

In this exercise, you are a GIS technician working for a city planning department.

Today's project is for city residents. You have been asked to set up a system where they can enter their address and find out local information; for example what neighborhood is the address in, what public trees grow nearby, where is the nearest library, and when is the garbage collection day.

You realize this can be done relatively easy using transformers in FME for attribute handling, filtering, and data joins.

1) Inspect Source Data

Use the FME Data Inspector to inspect all of the datasets listed below, in order to familiarize yourself with the data.

2) Start Workbench

Start FME Workbench and begin with an empty canvas.

Now let's first consider our strategy here. The first part of the project will be to extract the address we want. In lieu of using a "proper" database, we'll read all of the address features and filter out the one we want with a Tester transformer.

3) Add Reader

Add a Reader to read the following dataset:

Reader Format

Esri Geodatabase (File Geodb API)

Reader Dataset

C:\FMEData2015\Data\Addresses\Addresses.gdb

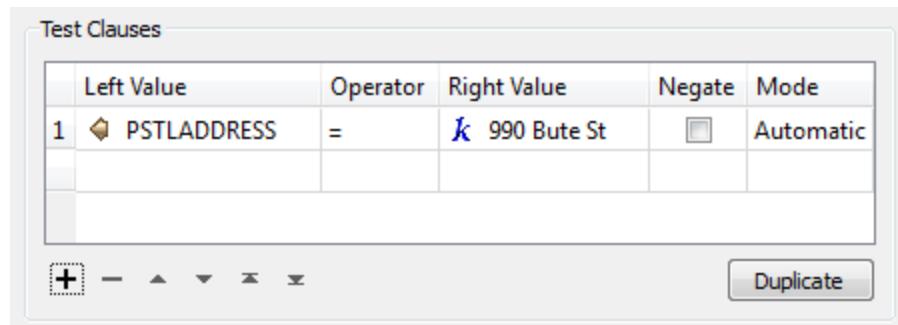


Be sure to check the Workflow Options parameter. In this exercise it needs to be "Individual Feature Types". However, if you've previously set this parameter to "Single Merged Feature Type" it may default to that without you being aware!

When prompted the feature type to add is PostalAddress.

4) Add Tester

Add a Tester transformer to the workspace. Set it up to test for a specific address. For this exercise we'll take the easy way and hard code that address. You can pick one from the Data Inspector (for example, 990 Bute St).



Connect Inspector transformers and try the workspace to make sure it works correctly. Only one address should pass.

5) Add Readers

Now let's start to gather information about this address. To do this let's add two new Readers, the first of which is:

Reader Format	Esri Geodatabase (File Geodb API)
Reader Dataset	C:\FMEData2015\Data\CommunityMapping\CommunityMap.gdb

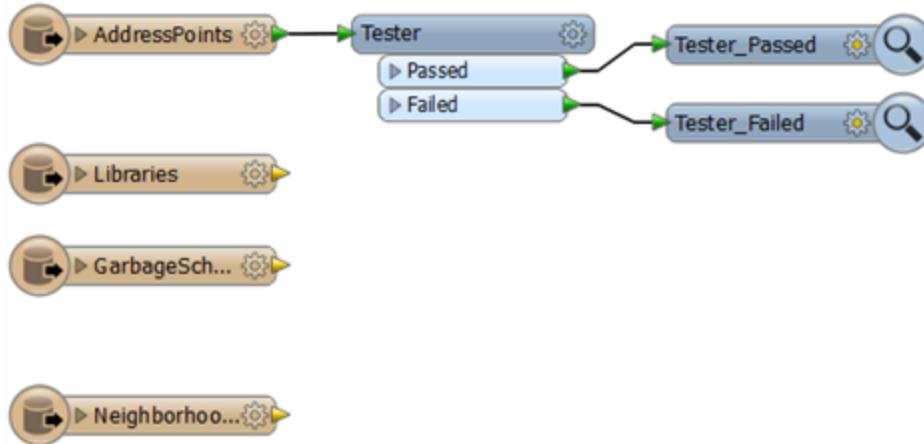
When prompted deselect everything except GarbageSchedule from the list of feature types (we'll use this now) and also Libraries, which we'll use later.

Now add the second new Reader:

Reader Format	Google Earth KML
Reader Dataset	C:\FMEData2015\Data\Boundaries\VancouverNeighborhoods.kml

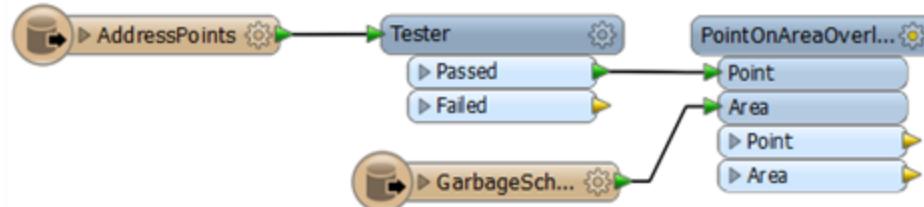
The feature type you need here is Neighborhoods. De-select everything else.

The workspace will look like this:



6) Add PointOnAreaOverlayer

Add a PointOnAreaOverlayer transformer to the workspace. Connect the Tester Passed port to the Point input port, and the GarbageSchedule feature type to the Area.



Check the transformer parameters, but you do not need to change anything; none will have any effect here.

7) Clean Schema

Attach an Inspector and run the workspace. You'll see that there are several attributes added from the garbage data:

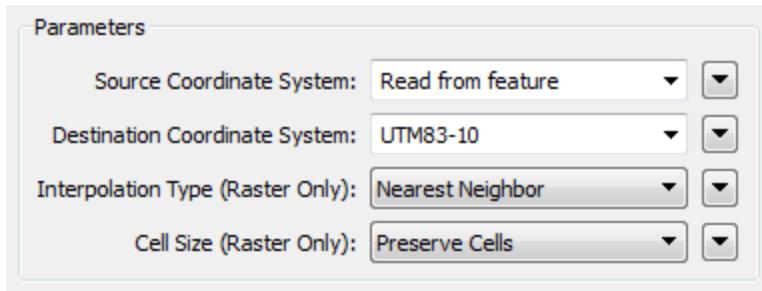
- Zone
- Subzone
- Schedule
- NumAddresses

The only one of these we really need is **Schedule**. So use an AttributeRenamer to rename it to GarbageSchedule, and use either an AttributeKeeper, AttributeRemover, or even an AttributeRenamer to delete the other garbage attributes.

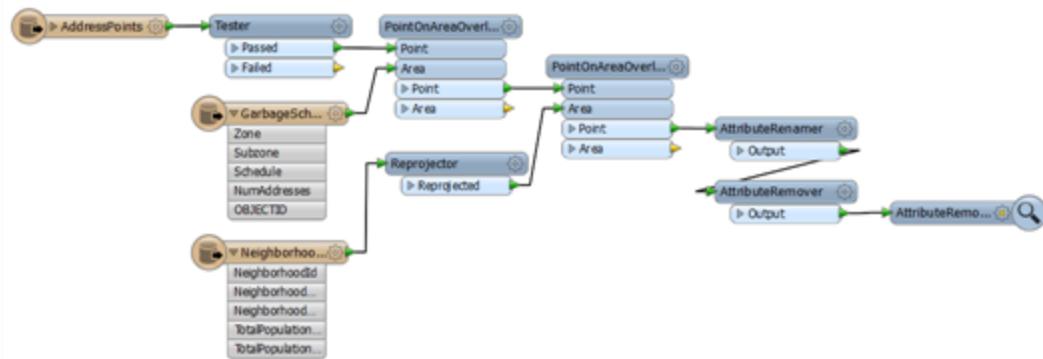
8) Add Reprojector

Now we can start to use the Neighborhood data. However, because it is in a Lat/Long coordinate system (the address and other data is UTM83-10) we'll need to reproject it first.

Add a Reprojector transformer after the Neighborhood feature type. Open the parameters dialog and set it to reproject from the source feature coordinate system to UTM83-10:

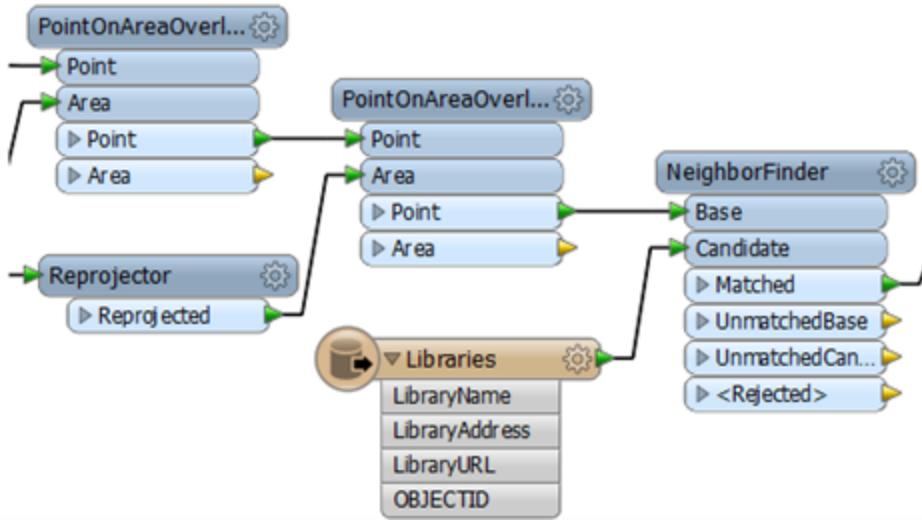


Now this is done you can repeat the previous two steps (PointOnAreaOverlayer/Clean Schema) to determine which Neighborhood the address is in; the neighborhood name is the attribute we require. If you add the PointOnAreaOverlayer before the AttributeRenamer then you can re-use these two attribute-handling transformer for this step:



9) Add NeighborFinder

The NeighborFinder is a tool for joining data on proximity. Add a NeighborFinder with the address (PointOnAreaOverlayer) output connected to the Base port and the Libraries dataset connected to the Candidate.



Open the parameters dialog. Set the Maximum Distance to be 1000 (in this case it is metres).

10) Run Workspace

Again, update your transformers to get rid of what you consider excess attributes, add an Inspector, and run the workspace.

The Table View in the Data Inspector should show the newly merged attributes:

PSTLADDRESS	PSTLCITY	PSTLPROV	INTSTATE	GarbageSchedule	NeighborhoodName	TotalPopulation2001	TotalPopulation2011	LibraryName
1 990 Bute St	Vancouver	British Columbia	CA	Tuesday	West End	42120	44540	Joe Fortes

AttributeRenamer_Output

In this case, the garbage schedule is Tuesday, the neighborhood West End, and the nearest library is called Joe Fortes.

11) Add Joiner

Now to add some information about what trees exist in the area. Add a Joiner transformer to retrieve this information from a CSV dataset. Open the Parameters dialog.

Notice that the first task is to define the dataset to read from. It is:

Reader Format
Reader Dataset

Comma Separated Value (CSV)
C:\FMEData2015\Data\Parks\PublicTrees.csv

Parameters

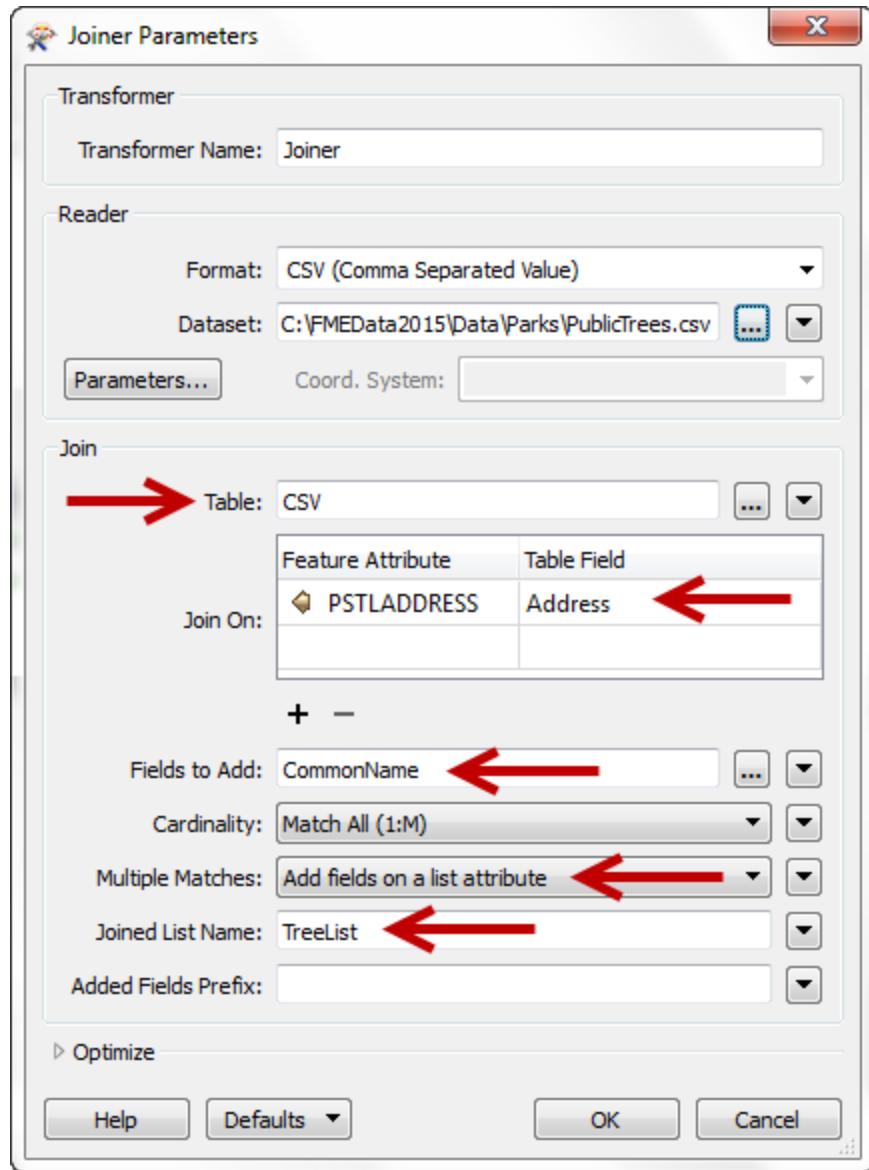
File Has Field Names Yes (Checked)
Lines to Skip 1

In the Join parameters first select the table. There will only be a single table (either CSV or the file name itself).

Then set the join attributes. We need to join PSTLADDRESS to Address.

In Fields to Add select CommonName. The ability to select the attributes we want to merge means we won't have to remove them later.

Finally, because there may be more than one tree per address, under Multiple Matches select "Add Fields on a List Attribute" and then enter a list name (e.g. TreeList) underneath.



Click OK to close the dialog.

12) Run Workspace

Run the workspace again. By querying the feature you should see that it now also has a list of trees (it won't appear in the Table View, only the Feature Information window):

```
NeighborhoodName (utf-8) West End
StreetName (utf-16) BUTE
TreeList{0}.CommonName PISSARD PLUM
TreeList{1}.CommonName KWANZAN FLOWERING CHERRY
TreeList{2}.CommonName WHITE ASH
TreeList{3}.CommonName YELLOWWOOD
```



We've now successfully filtered, managed, and merge some data to identify key properties for a given address; which is what the aim of the example was. However, it does leave us in a fairly unsatisfactory state, not having output or processed the information in any way.

As an advanced task, if you have time, let's try handling the data we've just merged, by creating a report in HTML.

13) Inspect HTML Template

Fortunately we already have a template for the HTML at:

C:\FMEData2015\Resources\AddressReportTemplate.html which you can open in a text editor to familiarize yourself with. Items marked like so...

```
{fme:get-attribute("AttributeName")}
```

...will be replaced with the attribute value when we process it in FME.

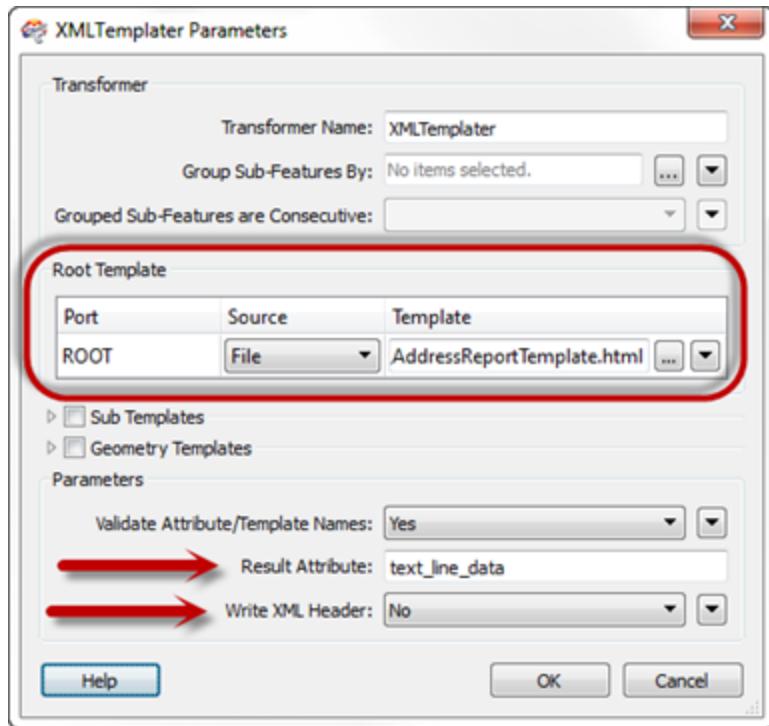
14) Add XMLTemplater

Add an XMLTemplater transformer to the end of the workspace. This transformer will process the pre-defined HTML template with the information we've just extracted.

Open the parameters dialog. Set the Root Template source to be a file and browse to the file at:
C:\FMEData2015\Resources\AddressReportTemplate.html

Set the Result Attribute to be called: text_line_data (this will match the output attribute in the Writer below).

Set the Write XML Header parameter to be No.



15) Add Text File Writer

Now let's add a writer to write the HTML to. We can do this is a plain text writer:

Writer Format:

Text File

Writer Dataset:

C:\FMEData2015\Output\Training\AddressReport.html

Parameters

MIME Type

text/html

Connect the new feature type to the XMLTemplater output. Run the workspace. Open the output file in a web browser. The output should look something like this:

Property Report

Address: 990 BUTE

Neighborhood

Garbage Day

Local Library

West End

Tuesday

Joe Fortes

Trees

- PISSARD PLUM
- KWANZAN FLOWERING CHERRY
- WHITE ASH
- YELLOWWOOD
-



Powered by



Congratulations! You have now completed the exercise for this chapter.

Chapter 6 - 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



Our 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 and services philosophy built on the principle of knowledge transfer.

Safe Software Blog



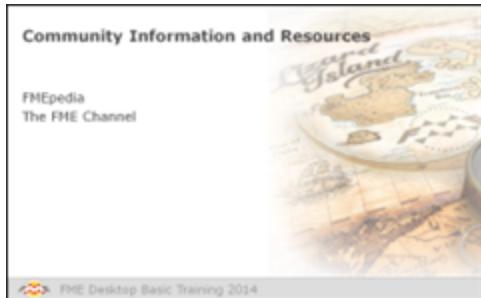
The Safe Software blog provides technical information about FME, articles about customers' use cases, and general thoughts on spatial data interoperability.

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 Downloads section.

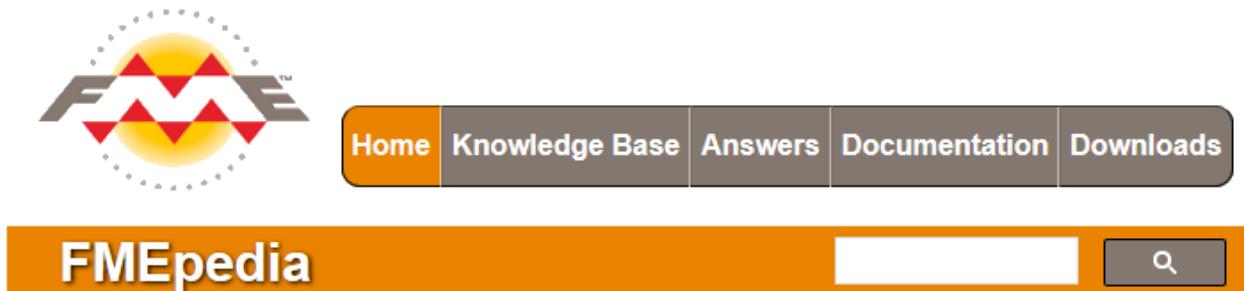
Community Information and Resources



Safe Software actively promotes users of FME to become part of the FME Community.

FMEpedia

FMEpedia is our community web site - a one-stop shop for all community resources, plus tools for browsing documentation and downloads.



FME 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.

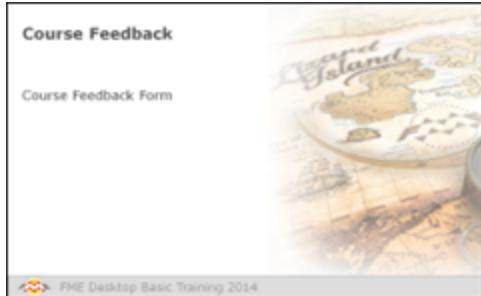
FME Community Answers

FME community members post FME related messages and questions and share in answering other users' questions. Top users are known as "heroes". Come and see how they can help with your FME projects!

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.

Course Feedback



The format of this training course undergoes regular changes prompted by comments and feedback from previous courses.



There's one final Q+A to go – and this time you'll be telling us the answers!

Safe Software Inc. greatly values feedback from training course attendees. This is your chance to tell us what you really think about how well we're meeting your training goals.

The current course structure has been determined by attendee comments and we appreciate your feedback more than ever.

You'll be sent a link to a feedback form after your course.



FME Desktop Training

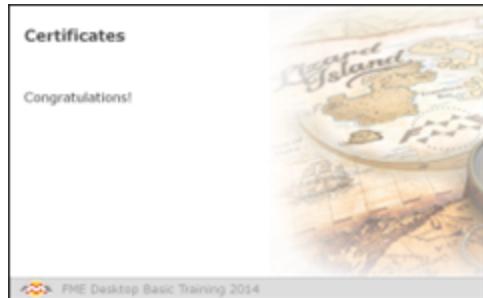
*1. Please rate the following statements

	Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
I can apply the course content to my job	<input type="radio"/>				
The course length was adequate to cover the content	<input type="radio"/>				
There was an appropriate balance between lecture and lab	<input type="radio"/>				
The exercises were directly applicable to the material covered	<input type="radio"/>				
The graphics and instructions were clear and concise	<input type="radio"/>				

*2. Please rate the following

	Excellent	Good	Fair	Poor
Overall course rating	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overall Instructor rating	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overall learning environment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overall usability of FME	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Certificates



All FME training course attendees may now receive a certificate.

Congratulations!

With the presentation of your certificate of achievement, you have now officially completed the FME Desktop 2015 Training Course.



Thank You



Thank you for attending this FME training course.

all of us at Safe
Software wish you
good luck and lots
of fun as you build
exciting new
solutions using
fme

Congratulations!



"As a reward for reading this far, here's a small puzzle for you to try out.

*How well do you know the FME splashscreens? Here are six past
splashscreens. Can you tell which version of FME each one belongs to?"*

