



FME Desktop Training Manual

FME Desktop 2013 Edition





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.

Any spatial data included in this material is intended for training use only. It is not meant to accurately reflect any real life situation and may have been altered in some way from the original source data.

Much of the data used here originates from public domain data made available by the City of Austin and Travis County, Texas. The datasets can be found at:

ftp://ftp.ci.austin.tx.us/GIS-Data/Regional/coa_gis.html

Copyright

© 1994–2013 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.
Suite 2017, 7445 – 132nd Street
Surrey, B.C., Canada V3W 1J8
fax: 604-501-9965
e-mail: services@safe.com
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 holders and should be noted as such.

Document Information

Document Name:	FME Desktop Training Manual 2013 v9.0
Version:	FME Desktop 2013
Updated:	January 2013

FME Desktop Training Manual	1
Introduction	8
Training Pathway	8
FME Version	8
Sample Data	8
Course Overview	10
About the Manual	12
Course Resources	14
About Safe Software	16
Introductions	17
Barriers to Interoperability	18
Data Types	18
Data Formats	19
Interoperability	20
Interoperability Solutions	21
Introduction to FME	24
FME Editions and Licensing	25
FME Desktop Components	27
Other FME Products	29
Introduction to FME Workbench	31
Setting up a Translation	37
Introduction to Data Inspection	43
Introduction to FME Universal Viewer	44
Major Components of the FME Universal Viewer	44
Using FME Universal Viewer	46
More FME Universal Viewer Functionality	49
Raster Data and FME Universal Viewer	54
Translation Previews	58
Introduction to FME Quick Translator	60
Setting up a Translation	60
Module Review	63
What You Should Have Learned from this Module	63
Q&A Answers	63
Appendix A: Supported Geometries	66
Data Transformation	70
What is Data Transformation?	70
Data Transformation Types	70
Structural Transformation	72
Transformation Using Transformers	79
Content Transformation	86
Transformers used in Series	88



Transformer Handling	94
Transformers used in Parallel	103
Group-based and Feature-based Transformation	107
Data Inspection Using FME Workbench	112
Coordinate System Transformation	115
Module Review	118
What You Should Have Learned from this Module	118
Translation Components	122
Key Components	123
Managing Components	129
Controlling Translations	142
Workspace Parameters	145
Reader and Writer Parameters	148
Feature Type Parameters	151
Format Attributes	153
Parameter Documentation	164
Published Parameters	168
Module Review	174
What You Should Have Learned from this Module	174
FME and Data Formats	175
Supported Formats	175
Licensing and System Requirements	177
Datasets	180
Managing Reader Datasets	186
Multiple Dataset Translations	198
Advanced Format Controls	208
Semantic Transformations	212
Dynamic Translations	216
Module Review	219
What You Should Have Learned from this Module	219
Finding Transformers	220
Transformer Gallery	220
Transformer Searching	221
Inserting Transformers	223
Integrated Attribute Construction	226
Integrated Parameter Handling	227
Integrated List Handling	230
Integrated Functions	232
Most Valuable Transformers	238
Managing Attributes	240
Conditional Filtering	250



Synchronous Read and Write	259
Basic Custom Transformers	268
Schema Mapping	275
Parallel Processing	280
Module Review	283
What You Should Have Learned from this Module	283
Best Practice	284
What is Best Practice?	284
Why use Best Practice?	285
Methodology	285
Style	298
Performance	311
Debugging	317
Organization	329
Module Review	336
What You Should Have Learned from this Module	336
Chapter 7 - Course Wrap-Up	337
Product Information and Resources	337
Community Information and Resources	339
Course Feedback	339
Certificates	341
Thank You	342
Congratulations!	342

Introduction



This training material is part of the FME Training Pathway system.

Training Pathway

This training material is a key part of every FME Training Pathway.

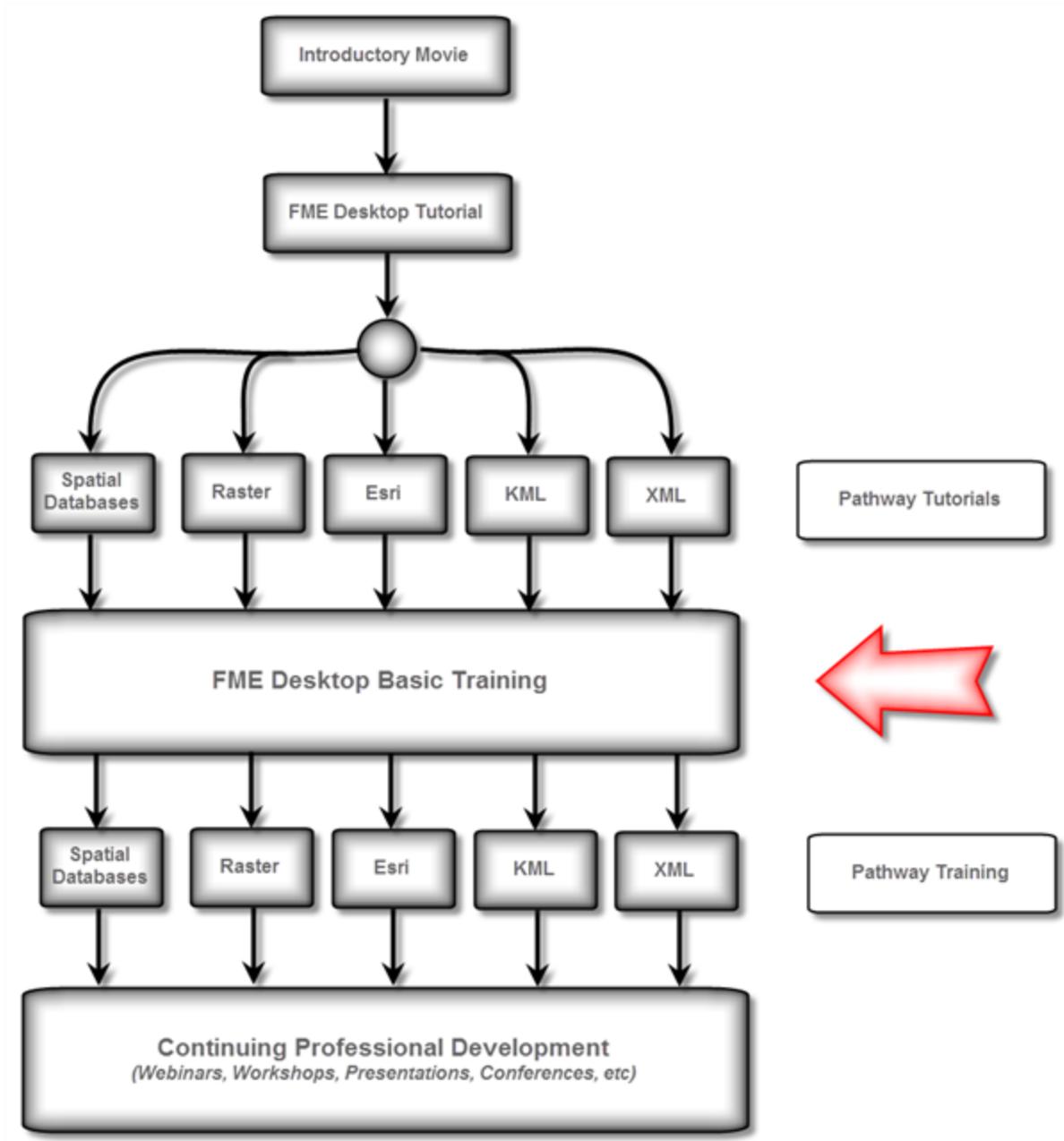
It assumes a basic familiarity with the concepts and practices covered by the FME Desktop Tutorial, and covers material that will be required to take specific Pathway Training Courses.

FME Version

This training material is designed specifically for use with **FME2013**. 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:



Course Overview



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

Who Am I?

This is the instructor’s chance to introduce herself or himself.

Certified FME Trainers

Your FME training instructor may possess FME certified trainer status, indicating Safe Software’s guarantee of quality for your FME training course.

To qualify as a certified FME trainer requires these minimum commitments to trainees:

- To provide a set of official FME training materials to each trainee
- To only advertise a course as FME certified if a certified trainer will be leading the course
- To give the opportunity for each trainee to provide feedback direct to Safe Software



The form for providing training feedback can be found online at: www.safe.com/feedback

If you have comments or concerns on items not covered by the feedback form then please use the contact page on our website (www.safe.com/contact), or else contact our Training Manager directly via train@safe.com

Course Goal

This training course provides a framework for understanding 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 six main sections. These sections are:

- Data Translation Basics
- Data Transformation
- Translation Components
- Datasets and Feature Types
- Practical Transformer Use
- FME Best Practice



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 and exercise workbook are yours to keep. They include detailed material to help you remember your training.

This manual 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.

Both training manual and exercise workbook are structured in the same way as the course, in six sections.

Icons

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



Tip: Additional advice to help apply the knowledge you have learned.

Caution: A warning where misuse of FME could lead to difficulties.



Example: An example or exercise that the instructor will use to demonstrate a particular point or feature of FME. You may work through the example with the instructor or on your own.

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



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

Questions about the course content and how it is applied.





Important hints and tips appear in a box like this, to separate them from ordinary content.

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 datasets from the fictional City of Interopolis.

Most exercises ask you to assume the role of a city planner and solve a particular problem using both the City’s corporate data and data provided by external agencies.

Resources for the examples and exercises in the manual can be found at the following locations:

Location	Resource
C:\FMEData\Data	Datasets belonging to the City of Interopolis and local organizations and companies
C:\FMEData\Resources	Resources used in the training such as datasets that don’t fit into the City of Interopolis
C:\FMEData\Workspaces	Workspaces used in the student exercises. There are subfolders for each session
C:\FMEData\Output	The location in which to write exercise output. There are subfolders for each session
< documents>\My FME Workspaces	The default location to save FME workspaces

On Your Virtual Machine

If you are taking FME Desktop training online, you may be provided with a virtual machine hosted in the cloud. If so, you should find all of the above resources, plus a copy of the FME release used in the training, just as if you were working on a physical computer.

You should also find a digital copy of this manual and the training workbook, either on the computer desktop or in the documents folder.

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.

About Safe Software



Safe Software: Achieve total spatial data mastery!

History of Safe Software

Safe Software Inc. is the maker of FME® and the global leader in spatial data transformation technology that helps GIS professionals and organizations master their data interoperability challenges.

A common question is, “where did the names ‘Safe Software’ and ‘FME’ come from?”

Safe Software was founded way back in 1993 by Don Murray and Dale Lutz.

Back then, there was a format called the Spatial Archive and Interchange Format (SAIF). Since “SAIF” is pronounced “safe,” we decided to take the name Safe Software as a play on the SAIF format name.

FME officially stands for the Feature Manipulation Engine, and reflects its ability to transform data. By 2008, we had shortened it to “FME,” since by then most everyone knew what we were talking about, and to be honest, it was a mouthful.

Another common question is, “what happened to your hair?”, but that’s a story for another time!

Today, FME is used by thousands of customers in over 116 countries in a wide variety of industries. Our channel partner network has more than 100 value-added resellers across six continents.



The Safe Software Vision...

‘Our vision is to be the global leader in spatial data transformation, to knock down every last barrier to free and unrestricted spatial data use.’

The Safe Software Mission Statement...

‘Our mission is to help organizations use spatial data more easily and efficiently.’

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?

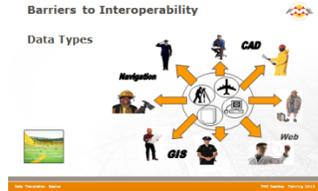
This is your chance to introduce yourself to the rest of the class. You may want to mention the organization you represent, what experience you have using FME, and which formats and functionality you're most interested in.



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!

You can find an online version of this manual at: www.safe.com/onlinelearning

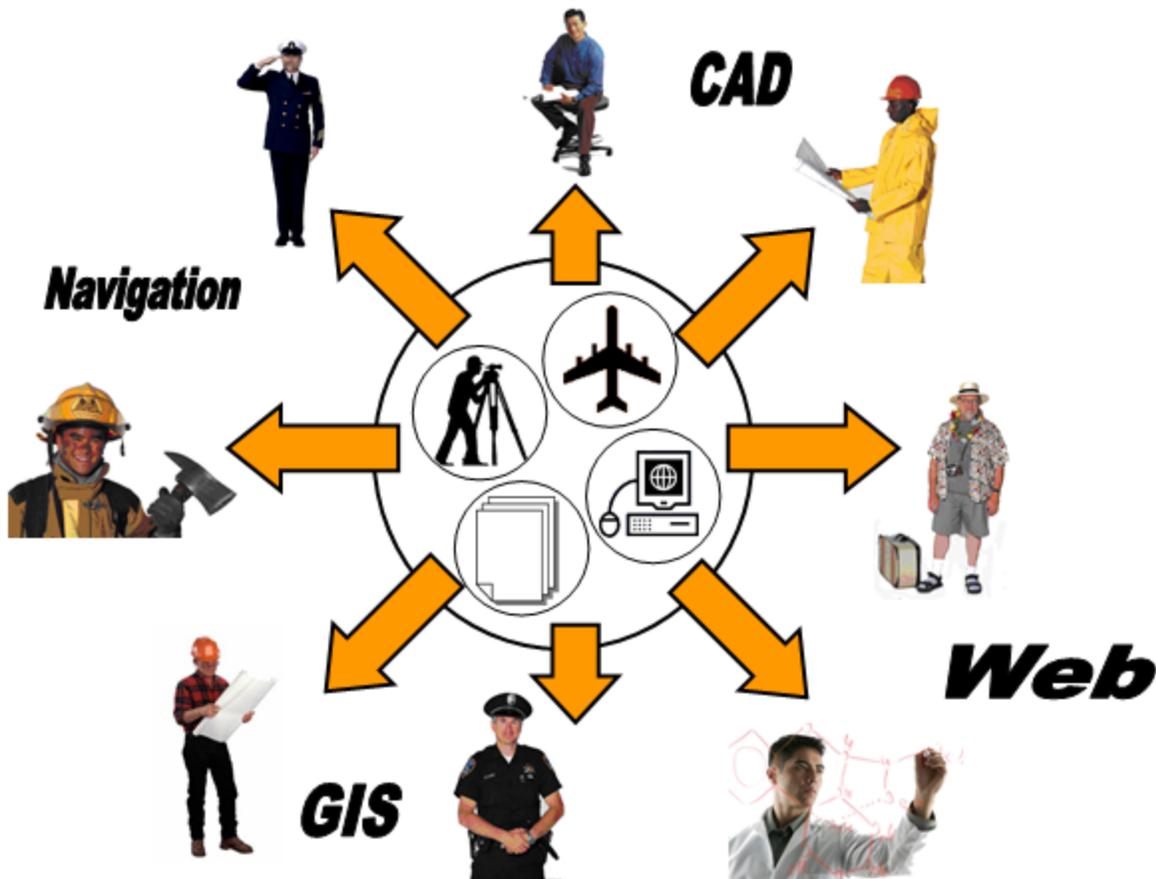
Barriers to Interoperability



Unfortunately, interoperability is not always easy. For spatial data there are two major barriers.

Data Types

In the spatial data world there are many 'types' of data; for example CAD, GIS, BIM (Building Information), Navigational, and so forth.



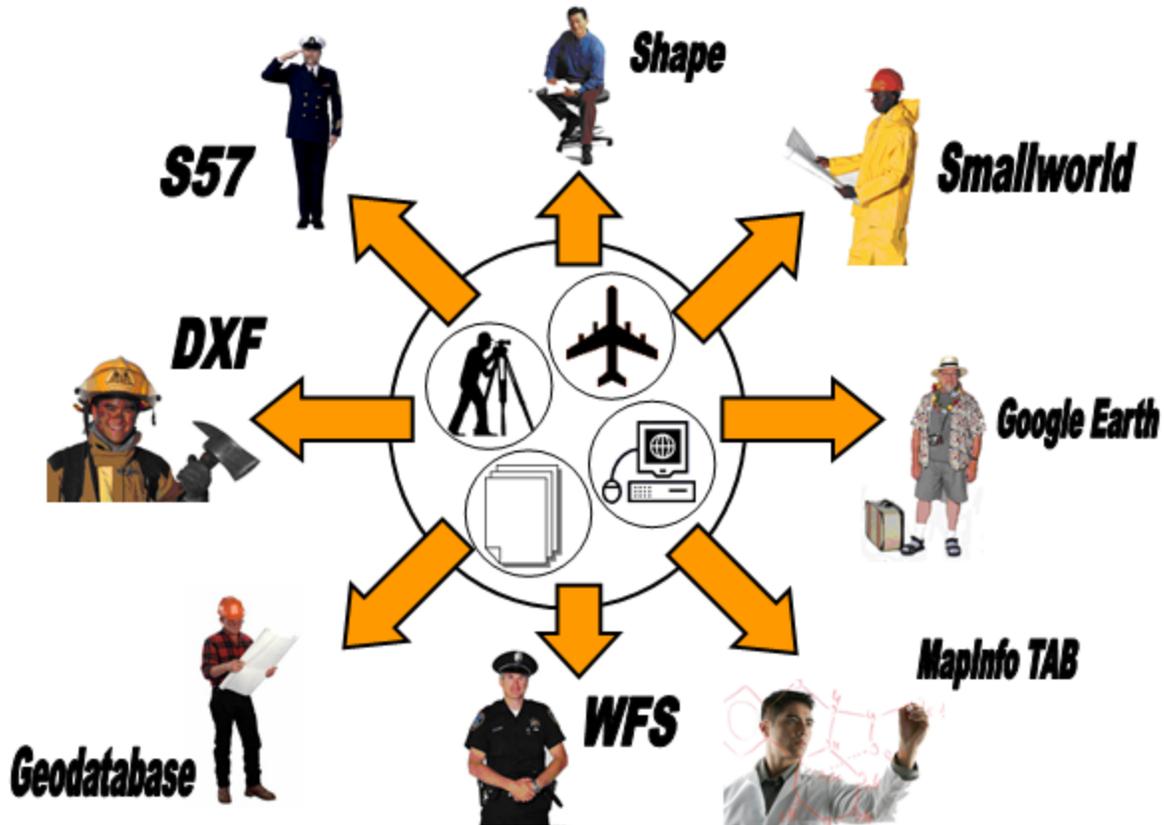
Because the different types of data are designed for different purposes they are not always compatible with one another. We can say they have different *meanings*.

For example, a navigational system such as an in-car GPS and a CAD dataset for a civil engineering project may both deal with the concept of "roads", but their views of the data will be quite different.

The navigational data will include a generalized – but topological - network of data with traffic based attributes. The CAD data will include precise positional geometry and attributes concerning road surface. Neither data is easily interoperable with the other.

Data Formats

Also unique to the world of spatial data is the number of ‘formats’ of data; for example, Geodatabase, DWG, MIF/MID, KML, and so forth.



Because different datasets have different formats, they are not always compatible with the same computer applications, even if they are of the same type.

For example, a system that reads Geodatabase datasets may not also be capable of importing MapInfo TAB data, despite the fact that both of these are GIS type datasets. A GPS unit might be able to read GPX data, but not NMEA.

Interoperability



FME's marketing materials state that "FME is the dominant technology for solving spatial data interoperability headaches".

What is Interoperability?

Interoperability is about communication; in our case communication by the sharing and distribution of data, and the ability to use that data transparently. The key words in this definition are:

Communication: The exchange of information

Data Sharing/Distribution: Supplying either the data itself, or direct access to the source

Transparently: Without the need for background knowledge of the data structure

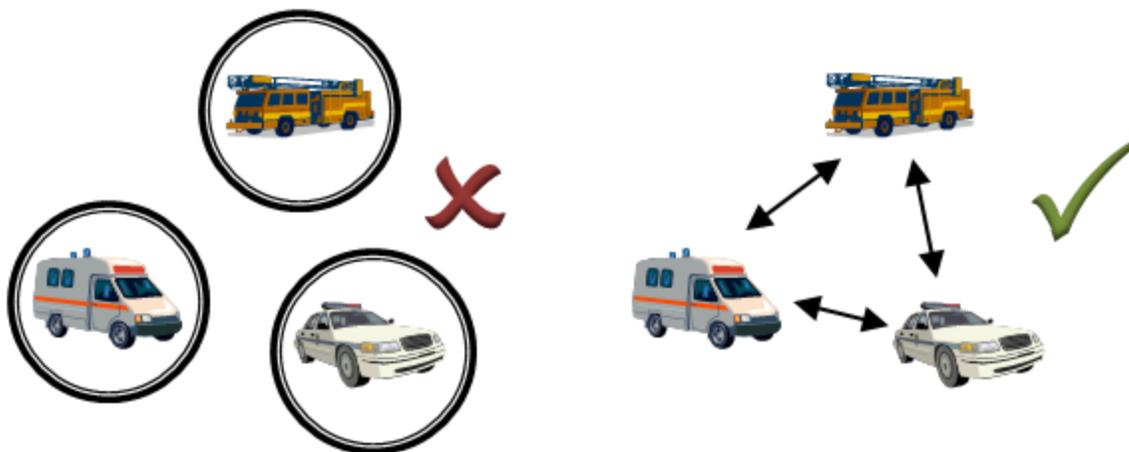
Interoperability is sometimes also known as *Data Harmonization* or *Data Model Transformation*.

Example of Interoperability



Police Chief Webb-Mapp says...

"When emergency services work in isolation - whether it is spatial data, rescue equipment or radio frequencies - then chaos can ensue. Interoperability is important for coordinating emergency response efforts".



Interoperability Solutions



To overcome the barriers to spatial data interoperability, there are a number of solutions.

Data Standards

The most obvious solution to a lack of interoperability is *Data Standards*. If everyone used the same type and format of data then there would be no problems sharing it.

Standards can be divided into *Formats* or *Data Models*, and some standards are both!

Some are used to store data, and some used merely as a vehicle for exchanging datasets.

Common Standards

Many key spatial standards are created and/or maintained by the OGC (*Open Geospatial Consortium*), a non-profit and international standards organization.

ISO (*International Organization for Standardization*) also produces some standards, such as S-57.

Other standards are created either *ad-hoc* or for a particular country only. These may be entirely new or may be an extension or a subset of an OGC or ISO standard.

Some common standards:

OGC:GML, KML, WKT, WKB

National:MasterMap (UK – a data model based on OGC GML)

TIGER/Line (US – both a format and a data model)

KF85 (Sweden) and NAS (Germany)

Others:S-57 (both a format and a data model)

GeoJSON, GeoRSS, LandXML, CIM (Common Information Model)

VPF (format) and VPF products (DNC, VMAP, World Vector Shoreline)

Pros and Cons

On the whole standards work well for simple datasets, but when faced with greater complexity either become complex themselves, or become hived off into new standards.

In the previous example – where a CAD and Navigational dataset had a different view of “roads” – it’s hard to see a single standard that could reconcile both of these points of view.

Also, governments are not always known for close cooperation, which leads to a large number of national standards. Additionally a number of standards start out as non-spatial but later acquire spatial capabilities (GeoJSON and GeoRSS for example).

With all this – and software producers who are understandably reluctant to drop their own proprietary data formats – the number of recognized formats is probably increasing, not decreasing. This can lead to data interoperability being limited to a smaller sphere, such as an industry or particular data type.

Data Translation

A second interoperability solution is a software tool designed for *Data Translation*; the translation of data between different formats.

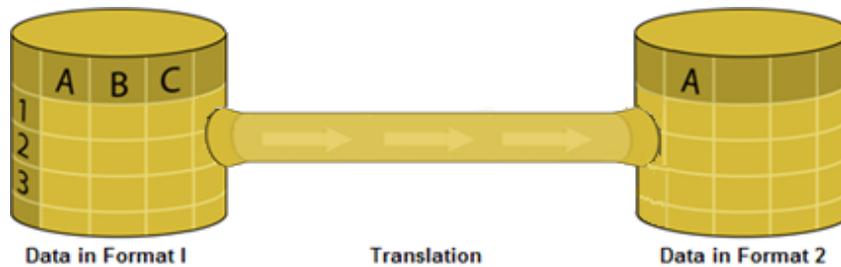
For example, applications exist to specifically convert one format of data to another, and producers of GIS/CAD systems incorporate import and export tools into their products.

However, such tools are limited in their effect because they do not solve the data type barrier. They only solve the data format barrier; and even then there can be major limitations.

The problem is that most tools of this type force data through a limited data model; i.e. the model has a limited understanding of the meaning of the data.

So such a tool might be able to convert the format of spatial data, but often loses the intent.

At Safe we typically illustrate this scenario as a pipeline. Within this limited pipeline, attributes (B and C) are lost from the newly-created dataset in Format 2.



Maybe it's because the attributes are a special type (e.g. a domain or subtype) that the data model does not recognize or support, and so discarded (leaked) them.

Alternatively, maybe it's because Format 2 is a different data type (e.g. for a CAD package instead of a GIS application) that does not support such attributes, and the translation tool has again discarded them, not knowing how best to represent that change of intent.

Also notice the flow arrows that show this is a one-way translation. This is a common problem for simple translation tools. And even if a return translation does exist, there's no guarantee the data will make it back to Format 1 with the exact same content as when it left.



Ms. Surveyor says...

"Working at a land survey company I have to provide data to customers in many formats. Without FME each format requires a custom Data Translation tool to be written. Project overheads are high and limited data models cause output to vary from what the original land survey intended."

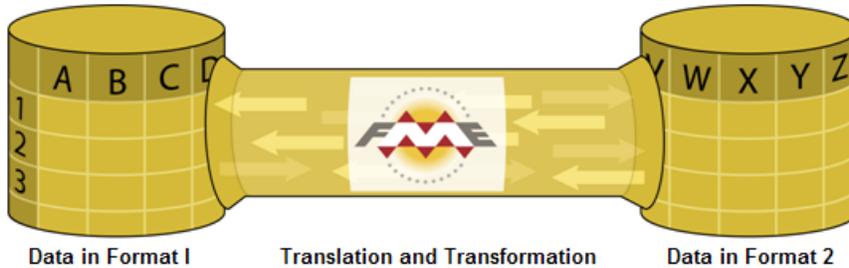
Data Transformation

Much better than simple format conversion, is the ability to carry out *Data Transformation*.

Data Transformation is the ability to restructure data during format translation. Such restructuring allows a user to manually handle the differences between data types, so that no data is lost during translation.

It is also known as *Transformational Translation*.

Data Transformation also implies a rich geometry model, so that, for the most part, a user won't have to use manual techniques. The application will automatically handle the data type differences, as well as the format differences, to keep all data and its meaning during automated translations.



FME is a tool specifically designed for Data Transformation. It has a rich data model and the ability for users to manually define data restructuring in order to best fit the destination format.



Ms. Surveyor says...

“Now that I use FME I can translate spatial data between different formats AND transform the data to the required structure, without any custom coding. Project overheads have plummeted, and the resulting data is high quality and reflects the original land survey.”

Introduction to FME



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

What is FME?

As noted, FME (Feature Manipulation Engine) was designed as a Data Transformation tool. In fact, because it can read, write and transform spatial data, it is often 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.

A SpatialETL tool is the geographic equivalent of ETL.

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.

How Does FME Work?

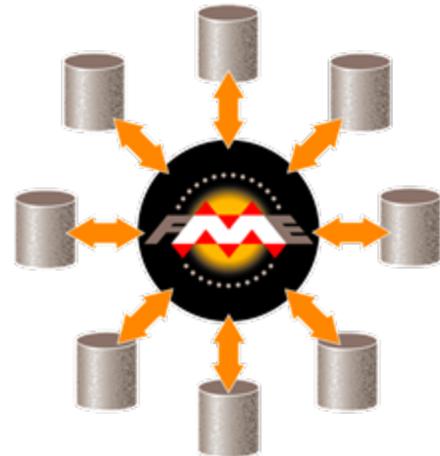
FME has a number of key characteristics:

Centralized

FME is a central engine among an array of supported format. Data can be read from any format and written to any other.

Adding support for a new format is as simple as plugging it into the FME engine.

FME can support both raster and vector formats under the same centralized model.



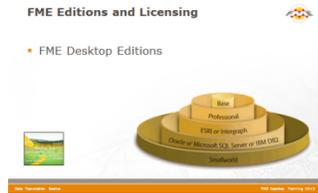
Semantic

FME has 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.

Transformational

The 'T' in ETL is what Format Translation tools lack. FME provides tremendous transformation functionality, resulting in output that can be much greater than the sum of the inputs, and allowing data to be transformed from one type (for example, GIS) to another (for example, CAD).

FME Editions and Licensing



FME comes in a range of different editions that vary according to the needs of the user.

FME Desktop Editions

FME is available in a number of editions.

Each edition has a different set of functions and formats available, but includes all the features from lesser editions.

The editions are:



FME Base edition

This is an entry-level FME edition that supports 40 formats and a set of basic transformation tools.

FME Professional edition

This is a general purpose FME edition with more formats and the full set of transformation tools.

Esri/Intergraph editions

These editions add support for formats tied to a specific application; for example the Esri edition includes support for Geodatabase and the Intergraph edition support for writing GeoMedia.

Oracle/SQL Server/DB2 editions

These editions add support for mostly database formats; for example the Oracle edition includes support for writing to Oracle Spatial databases.

Smallworld

This edition adds support for GE Smallworld reading and writing.

Always check www.safe.com for the latest info on which editions support which formats.

FME Licensing

A common mistake is to think each edition is a different download/installer file. This is not true. All editions of FME Desktop have the same installer, with different functionality being unlocked by different licenses.

FME has two licensing methods:

Node-Locked (Fixed) License

A node-locked license is for use with FME on a specific computer only. The license cannot be transferred to another computer except by a special request to Safe Software.



Floating License

Floating licenses are held on a server and issued to individual users as they start up FME. This is useful for the situation where there are many FME users, but not all using FME at the same time.

FME Desktop Components



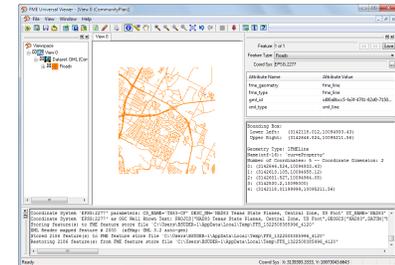
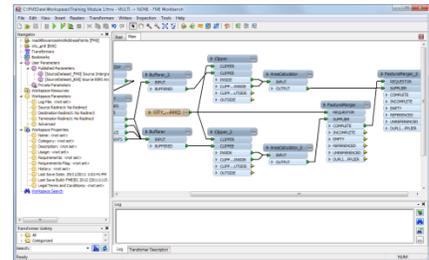
FME comprises a number of spatial data handling components. Everything here is included with every edition of FME Desktop.

FME Applications

There are three key applications within FME; FME Workbench, FME Universal Viewer, and FME Quick Translator.

FME Workbench

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

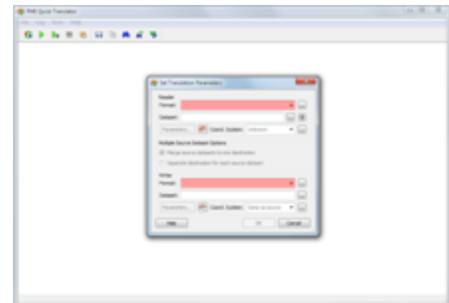


FME Universal Viewer

The FME Universal Viewer utility allows quick viewing of data in any of the FME supported formats. It is used primarily for data validation and quality assurance by allowing the previewing of data before translation or its reviewing after translation.

FME Quick Translator

The FME Quick Translator is a descendent of the FME Universal Translator, which used a scripting language (Mapping Files) rather than a graphic interface. It is used for quick format translations, or for running more sophisticated translations created in FME Workbench.



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 Application Extenders

FME Application Extenders are FME components embedded into other GIS applications. These commonly enable a GIS product to view datasets not native to that application.



FME Objects

FME Objects is a software library for working with spatial data. Application developers use FME Objects to add spatial data reading and writing support into their stand-alone applications.

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

Using the same underlying technology as FME Desktop, FME Server is a networked Data Transformation application. It can operate on a local-area network, or on the Internet.

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

In other words, FME Server facilitates sharing of spatial data; where, when and how it is needed.

FME Server is scalable, and can grow as system demands increase.

FME Server is also a *Model-Driven Architecture* (MDA) using workflows defined in FME Desktop, making it simple to convert a desktop process to one usable by a client anywhere.

See www.safe.com/fmeserver for more product 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.

Example extensions include:

- MRFCleaner transformer for geometry cleaning
- GDF format read/write support

Again, see the Safe Software web site for more information.

FME Store

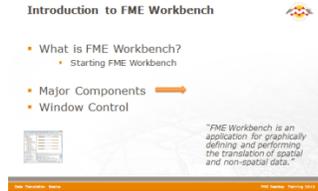
The FME Store is a marketplace for FME components that also add functionality or formats to FME. Offerings include both free and licensed products, provided by either Safe Software or 3rd party partners.

Find more information at:

<http://www.safe.com/support/support-resources/fme-store/>



Introduction to FME Workbench



Workbench is FME's primary tool for data translations. Its intuitive point-and-click graphic interface allows translations to be graphically described as a flow of data.

What is FME Workbench?

FME Workbench is an application for defining data translation and transformation processes.

With Workbench, underlying FME functionality is exposed in an intuitive interface that allows users to graphically define a custom dataflow from source, through transformation, to destination.

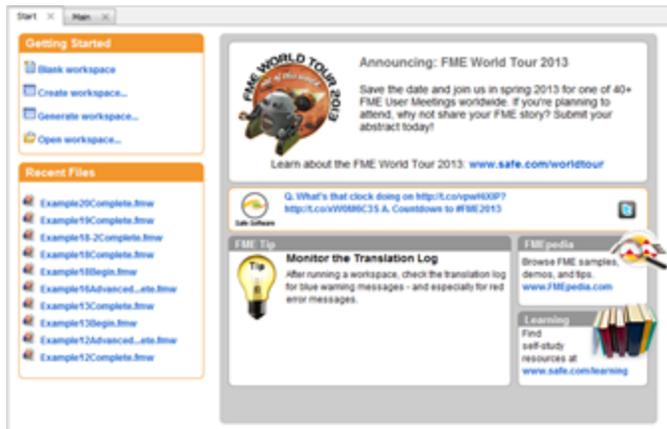
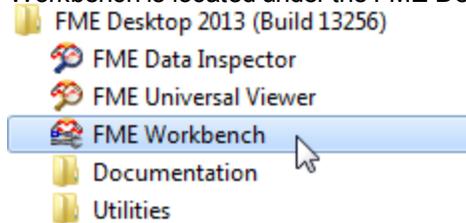
Workbench has tools for defining the source and destination dataset structure (or schema), and also for manipulating the geometry and attributes of spatial data.

Workbench is fully integrated to interact with other FME applications such as the FME Universal Viewer and other products such as FME Server, and is the authoring tool for FME Server models.

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.

Workbench is located under the FME Desktop entry in the Windows start menu.

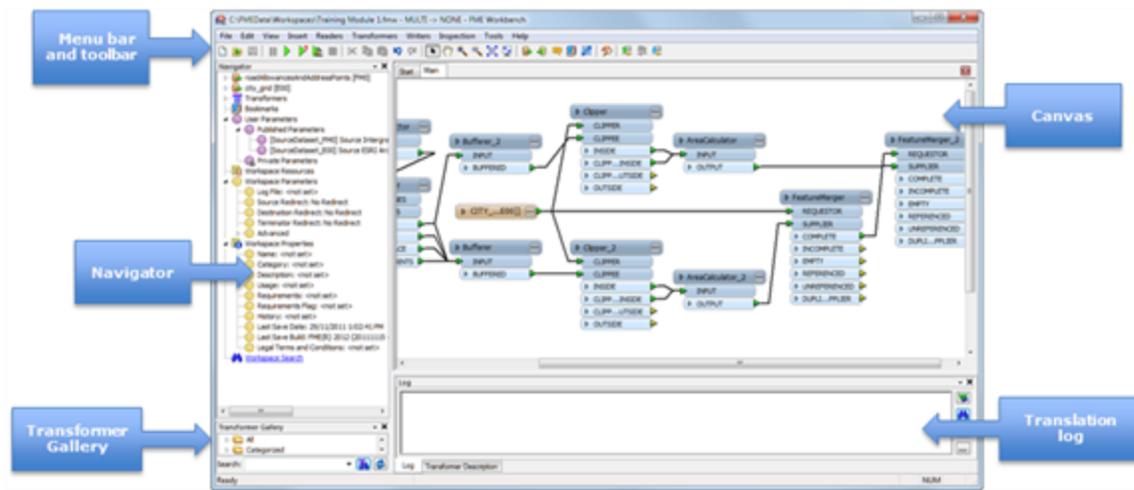


The startup tab links to a live web page and therefore the display will change over time as new information and resources are shown.

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 or even lead to a dead-end if required.



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. The difference is minor so it doesn’t really matter, but listen to certain instructors grind their teeth when you get it wrong!

Navigator

The navigator is an explorer type tool that shows a text definition of source and destination datasets, plus all settings that apply to these datasets.

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.

Overview Window (Not shown above)

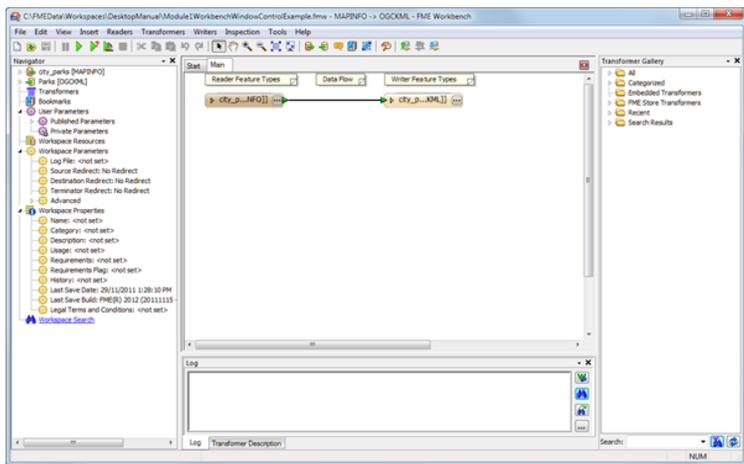
The overview window shows a small-scale view of the current FME workspace, and is therefore most useful when a large workspace is being worked upon.

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.

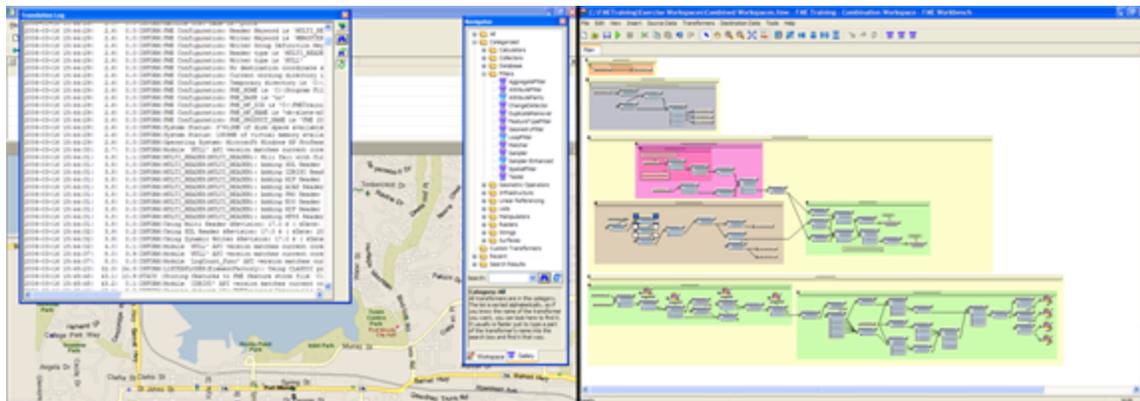
This user has opted to dock the Navigator and Transformer Gallery on the left and right sides of Workbench respectively. The Log window is docked in its traditional position at the bottom of the window.





InteropGeek68 says...

"I use multiple monitors. Then I can float Workbench menus onto a different monitor, and leave the main monitor free for the workspace canvas."





The F11 button toggles the Workbench canvas between full screen mode and back again. Shift+F11 does the same thing, but leaves the menubar and toolbar available for use. The F11 option is also available as a toolbar button:



When two or more windows are docked in the same location there is the option to arrange them either stacked or tabbed.

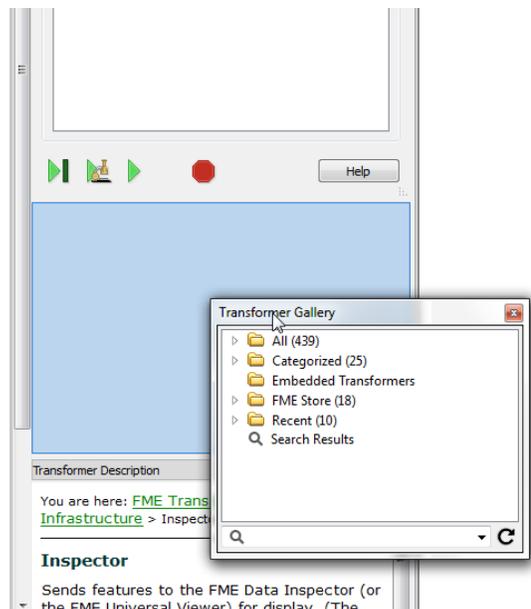
Stacking windows (such as the Navigator and Transformer Gallery) means one appears either vertically above the other (on the left or right hand side of Workbench) or horizontally next to each other (at the top or bottom of Workbench).

The functionality to stack or tab windows is controlled by the way the window is dragged and dropped into a new position.

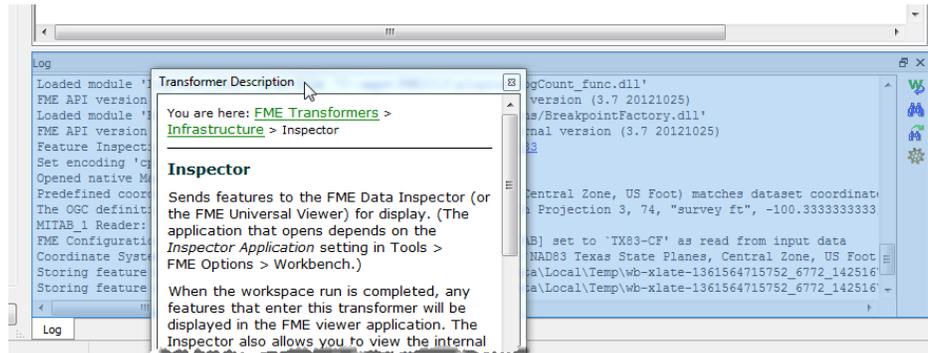
If a window is dragged and dropped on top of an existing window, then the two will become tabbed.

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

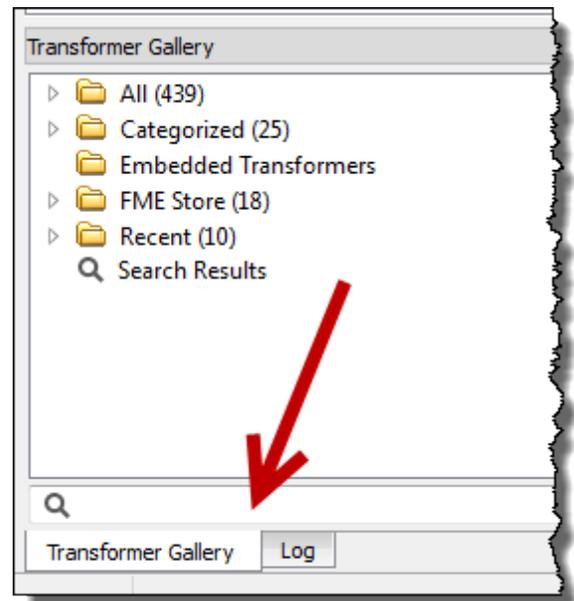
Here a user is stacking the Transformer Gallery between two other windows on the left-hand side of Workbench.



Here a user is choosing to arrange the Transformer Gallery and Log Window in a tabbed configuration.



The result will look like this.



Each user may have a preferred layout so feel free to adjust the windows in whichever way works best for you.



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.'

Which of the following words describe FME Desktop?

- 1) Distributed*
- 2) Semantic*
- 3) Transformational*
- 4) Centralized*

<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

Which of the following applications are parts of FME Desktop?

- 1) FME Workbench*
- 2) FME Server*
- 3) FME Quick Translator*
- 4) FME Universal Viewer*

<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

Which of the following tools is not found in FME Workbench?

- 1) A data viewing tool*
- 2) A source data selection tool*
- 3) A destination data selection tool*
- 4) Data manipulation tools*

<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

Which of the following windows are on the Workbench interface?

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

<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

Setting up a Translation



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

Dialog or Wizard

Although a new translation can be created from scratch within a blank workspace, it's more useful to get a head start by generating a translation with either the Generate Workspace dialog or wizard.

Both let users choose the source format and dataset, the destination format and dataset, and any settings.

Either of these methods can be accessed through the Create Workspace dialog (available on the start window or through File > New on the menubar) or the Generate Workspace dialog available on the start window (or Ctrl+G).



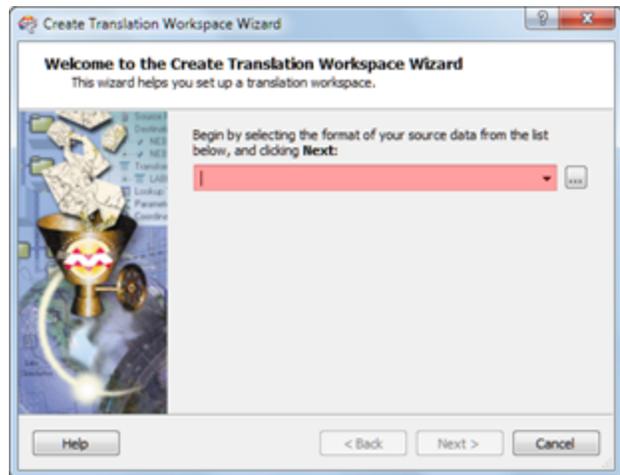
Terminology: In most cases FME uses the terms 'Reader' and 'Writer' instead of 'Source' and 'Destination'. Session 3 explains the why. For now, just be aware that a Reader reads datasets and a Writer writes datasets, and these terms are analogous to source/destination and input/output.

Translation Workspace Wizard

The Translation Workspace Wizard presents a series of choices through which the translation will be defined.

The initial screen of the workspace wizard prompts the user to select the format of the source datasets to be translated.

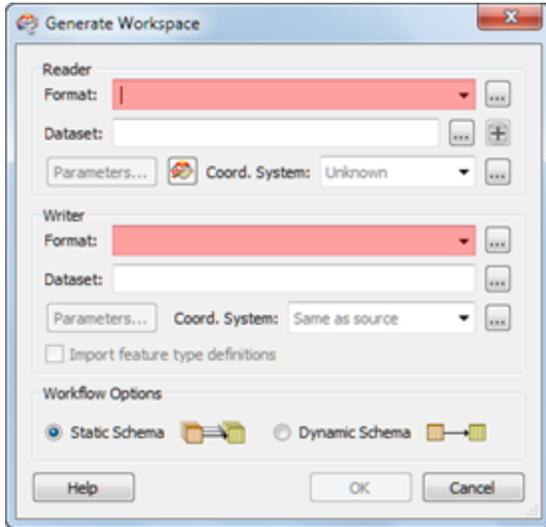
A red color indicates a compulsory field.



All format selection dialogs 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 by browsing through the Formats Gallery.

Generate Workspace Dialog

The Generate Workspace dialog condenses all the choices of the Translation Workspace wizard into a single dialog box. This is the preferred workspace creation tool for experienced users.



The Generate Workspace dialog has fields for the Reader and Writer Formats and Datasets. These prompts have the same drop-down menu and 'Intelli-complete' properties as the Workspace wizard.

The red fields indicate mandatory fields. Users must enter data in these fields to continue. Notice that the OK button is deactivated until the mandatory fields are complete.

There are also buttons for checking and/or altering settings for each dataset, and a button for previewing the data in the FME Universal Viewer.



The Create Workspace button on the Workbench toolbar is a shortcut to the Create Workspace dialog.



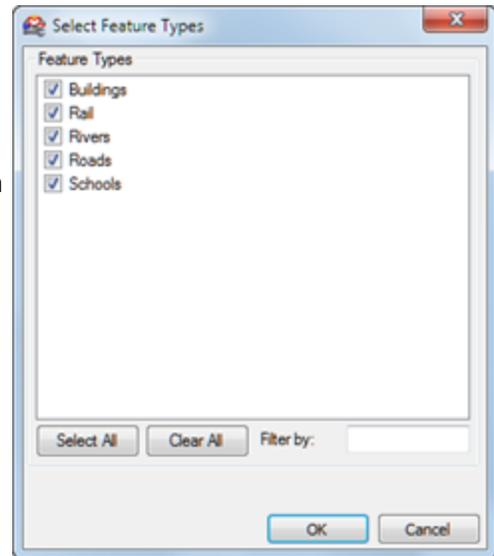
It's always worth checking the settings at this point. Although most settings will be exposed in the Workbench Navigator window, and can be set there, some settings affect how the translation workspace will be created and so need adjusting before you accept this dialog.

Feature Types Dialog

Whichever method of workspace creation is used, whenever a Reader (source) Dataset contains a number of different layers the user is prompted to select which layers they want to translate.

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.

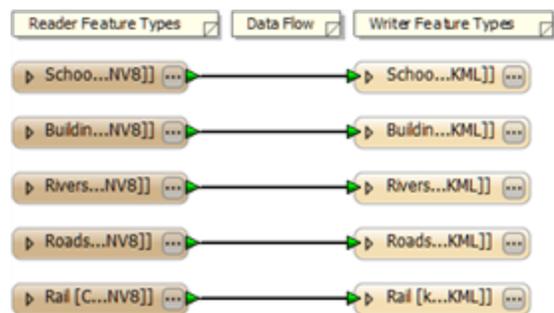
The New Workspace

A new workspace reads from left to right, from a source (Reader), through a dataflow to 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.

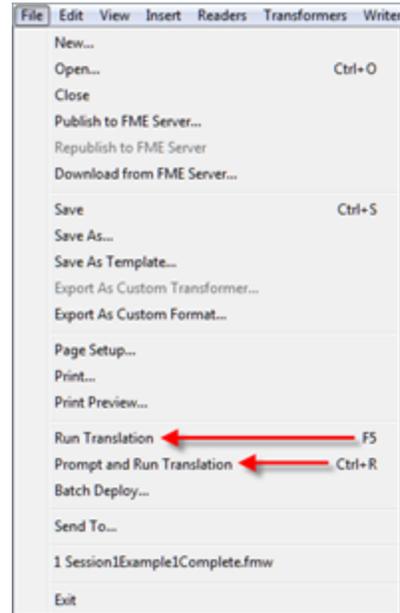
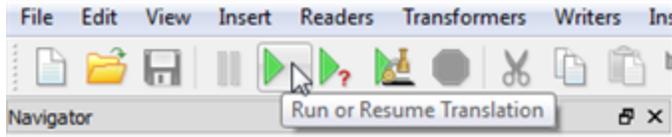
FME places annotation to emphasize the E-T-L structure (Source > Flow > Destination).

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



Running the Translation

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



There are also options under **File** on the menu bar to either 'Run' or 'Prompt and Run' a translation.

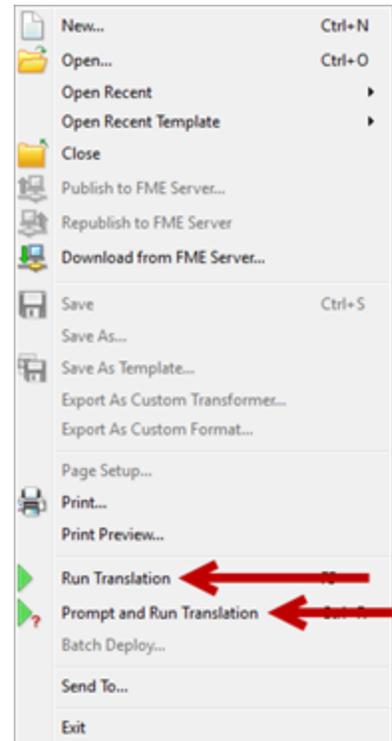
The File menu with run options include shortcut keys that can be used – the F5 key to simply run a translation and Ctrl+R to prompt and run a translation.

Saving the Translation

Workspaces can be saved to a file so that they can be reused at a later date.

Simply use **File > Save** (shortcut = Ctrl+S) or **File > Save As...** to save the translation.

The default file extension is .f_mw. 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.

Translation Results

After running a translation 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.

In this example the log file reveals that 2319 features were read from a MicroStation dgn file.

These features were written to a GML output file.

The overall process was a success (with 1 warning).

The elapsed time for the translation was 4.6 seconds.

```

-----
Features Read Summary
-----
Rail                                     4
Rivers                                  117
Roads                                   2198
-----
Total Features Read                      2319
-----
Features Written Summary
-----
Roads                                   2198
-----
Total Features Written                   2198
-----
SESSION HEADER: Closing DGN VB file
Translation was SUCCESSFUL with 1 warning(s) (2198 feature(s)/15458 coordinate(s) output)
FME Session Duration: 4.6 seconds. (CPU: 3.6s user, 0.5s system)
END - ProcessID: 7996, peak process memory usage: 56888 KB, current process memory usage: 51356 KB.
    
```



In FME2013 the log window has been updated with a few new features. Chief among these is that text representing a URL or Email address now automatically gets a hyperlink added to it. That way you can just click on it in the log window to activate its target.

```

Emptying factory pipeline
Logger: For more information contact http://www.safe.com
Storing feature(s) to FME feature store file `mapping_log.ffe'
    
```





Example 1: Quick Translation	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Grid (ArcInfo E00)
Overall Goal	Translate the city grid data from E00 to GML format
Demonstrates	Quick Translation in FME Workbench
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example1Complete.fmw

Let's see how intuitive FME's interface is by doing an example translation with minimal instruction.

Start FME Workbench and use it to carry out this conversion:

- Reader Format** Esri ArcInfo Export (E00)
- Reader Dataset** C:\FMEData\Data\CityGrid\city_grid.e00

- Writer Format** GML (Geography Markup Language)
- Writer Dataset** C:\FMEData\Output\DesktopTraining\Grid.gml

For now, ignore the Workflow Options and leave the default of 'Static Schema'.

Accept all feature types when prompted to select them.

Run the translation. Locate the destination data in Windows Explorer to prove that it's been written.



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.

Introduction to Data Inspection



Inspecting spatial 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?

One piece of FME marketing material states:

'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.'

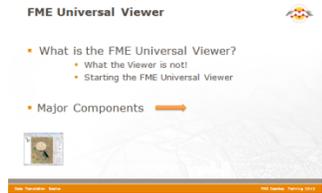
Features in the wrong source layer could need the whole process to be repeated.

Data Inspection saves me that hassle.'



See "Appendix A: Supported Geometries" on page 66 for a list of which geometries the FME data model supports.

Introduction to FME Universal Viewer



FME Universal Viewer is a Data Inspection utility for spatial data.

As you may have noticed, FME Workbench does not include any data viewing functionality. This inspection role is carried out by a complementary application, FME Universal Viewer.

What is FME Universal Viewer?

FME Universal Viewer (commonly called the 'Viewer') 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 Viewer 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.

FME Universal Viewer is closely tied to FME Workbench so that Workbench can send data directly to the Viewer. FME Universal Viewer also has basic tools for editing the display symbology.

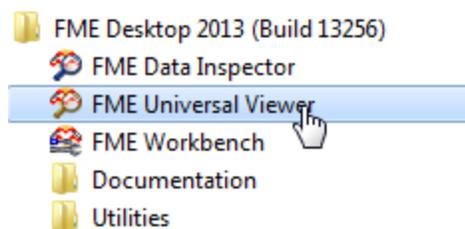
What FME Universal Viewer Is Not!

FME Universal Viewer 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.

Although FME can handle true three-dimensional features, the Viewer shows data only in two dimensions. Full 3D support is handled by the FME Data Inspector application, now in beta and due for final release with FME2014. The Data Inspector also includes a Table View for easier inspection of non-spatial data.

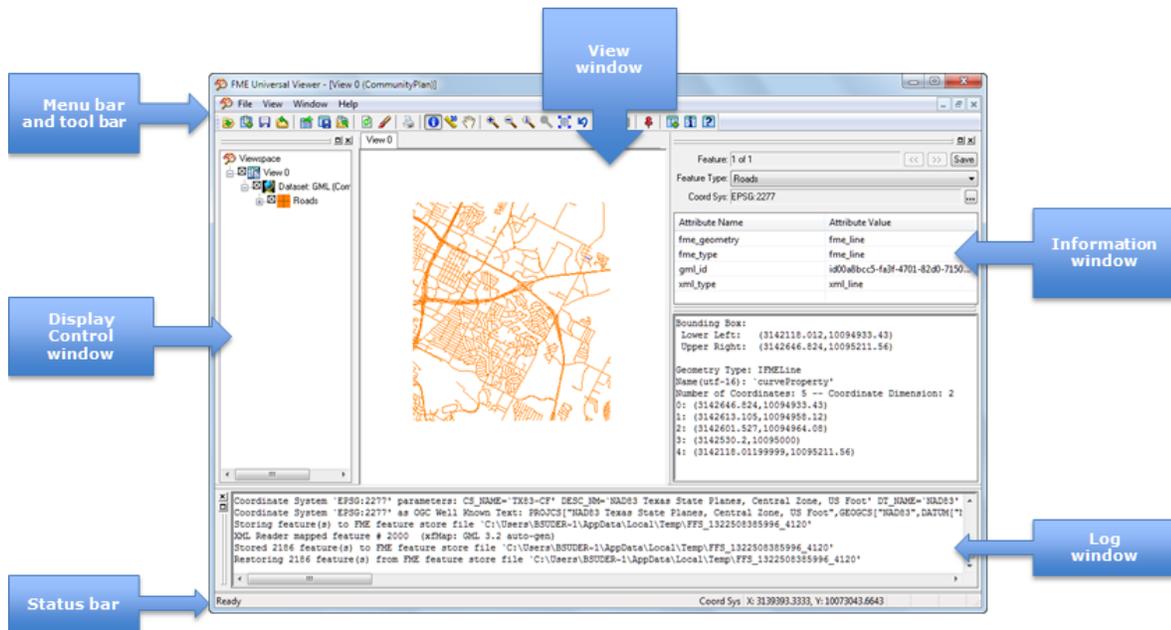
Starting FME Universal Viewer

To start FME Universal Viewer, from the Windows start menu click **FME Desktop**, then on the sub-menu click **FME Universal Viewer**.



Major Components of the FME Universal Viewer

When the FME Universal Viewer 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 Universal Viewer. 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.

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.

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.

Status Bar

The status bar is a general report of the status of the Viewer. It shows which feature is being read when a dataset is opened, indicates when the Viewer is drawing a dataset on screen, and prompts users when the Viewer is ready and waiting for user input.

Using FME Universal Viewer



With FME Universal Viewer it's easy to open and view any number of datasets and to query features within them.

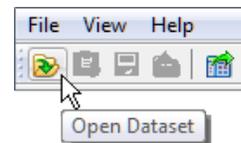
Viewing Data

FME Universal Viewer provides two methods for viewing data: opening or adding.

Opening a Dataset

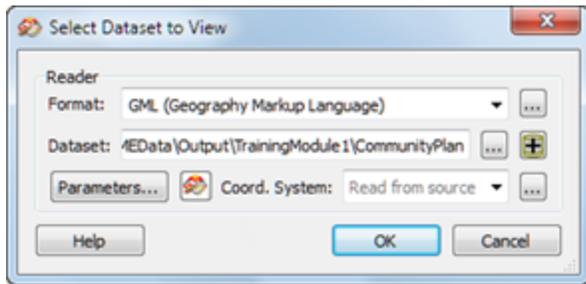
Datasets can be *opened* in the FME Universal Viewer in a number of ways.

- 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 Universal Viewer in which to define the dataset to view. In the case of the Drag-and-Drop and Workbench Inspect methods, the dialog is automatically filled in by FME.



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

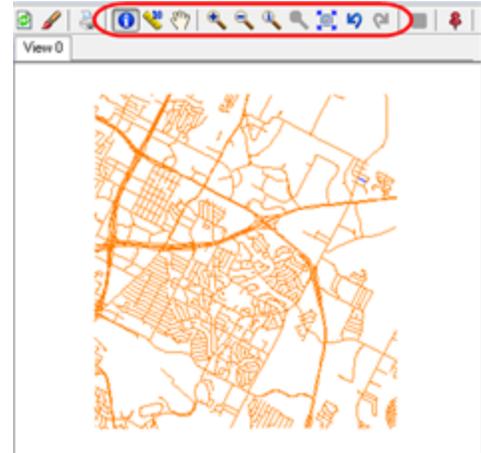
Once data has been opened in the FME Universal Viewer, there are a number of tools available for altering the view or querying features.

Windowing tools are:

- Pan
- Zoom In
- Zoom Out
- Zoom to a selected feature
- Zoom to a marked location
- Zoom to the full extent of the data
- Revert to the previous zoom extent
- Advance to the next zoom extent

Querying tools are

- Query an individual feature
- Measure a distance within a View Window
- Query all non-geometry features



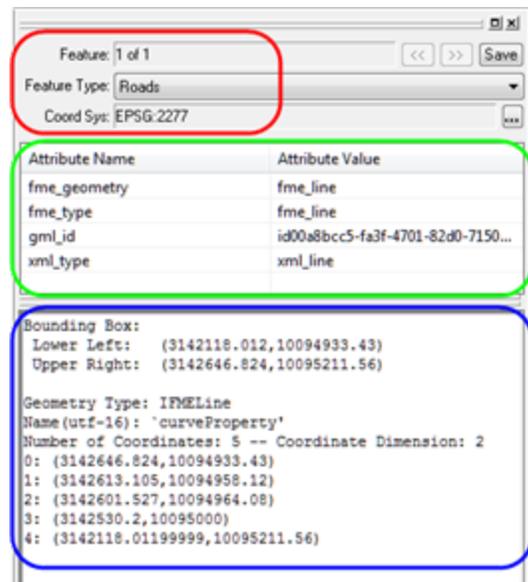
By default the Query tool is active when you start the FME Universal Viewer. Clicking the toolbar Query button at this point only turns the tool off.

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

The upper part reports on general information about the feature; which type (layer) it belongs to and which coordinate system it is in.

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 bounding box of the feature, the geometry type, the feature dimension (2D or 3D), and a list of the coordinates that go to make up the feature.





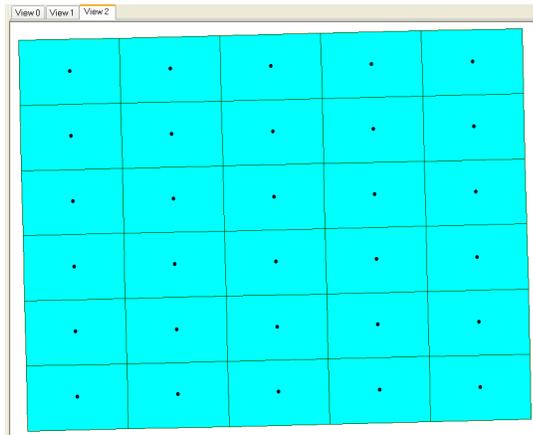
Example 2: Data Inspection	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Grid (ArcInfo E00 and Geography Markup Language)
Overall Goal	Inspect the city grid data from Example 1
Demonstrates	Data Inspection
Starting Workspace	None
Finished Workspace	None

Now let’s see how the FME Universal Viewer interface works by inspecting the input and output datasets from any quick translation you have already carried out.

For example, you should be able to find the output from example 1 at:

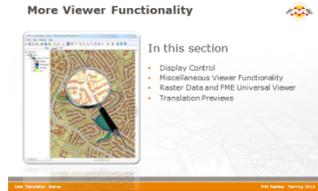
- Format** GML (Geography Markup Language)
- Dataset** C:\FMEData\Output\TrainingModule1\Grid.gml

That dataset should look something like this:



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”.

More FME Universal Viewer Functionality



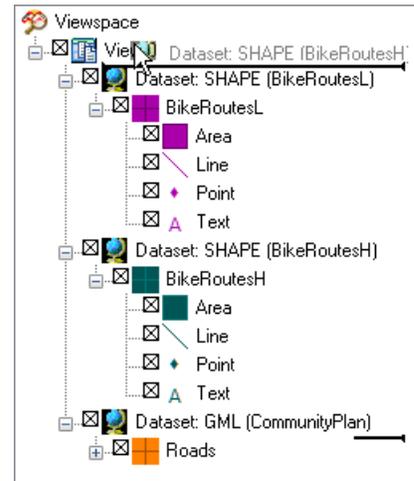
The FME Universal Viewer has a number of controls to assist in showing the data in an orderly manner.

Display Control

Display control is carried out in the Display Control window rather than through options on the toolbar or menu bar.

Datasets and feature types are shown in the same order in the view window as they appear 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.



Layers can only be ordered within their container Dataset.

For example, BikeRoutesH > Area cannot be promoted above BikeRoutesL > Area unless the entire BikeRoutesH Dataset is also promoted above BikeRoutesL.

Display Status

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

Turning off a higher level in the hierarchy turns off everything below it.

For example, clearing the checkbox for View 1 turns off data for the entire view. Clearing a dataset checkbox turns off data in the view for that particular dataset only.

Symbology

Each feature type can be assigned a different color or style that applies to all geometries. At a lower level each separate geometry type can be assigned a different color or style.



Mr. R.G.B. Color says...

'A color defined with a disk symbol  signifies that colors are defined by the source dataset, as opposed to being randomly selected by the FME Universal Viewer, which it does for non-color formats.'

Features in the wrong source layer could need the whole translation to be repeated. Data Inspection saves me that hassle.'

Miscellaneous Viewer Functionality

FME Universal Viewer has a number of miscellaneous functions that help users navigate through data, investigating data, and even translating 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 Universal Viewer has two 'Save' tools on the menubar.

File > Save Data As is used to convert the currently open data to a different format.

Only data shown within the current View window will be output, so a crude envelope can be set by using the windowing tools to show only the area of data required.

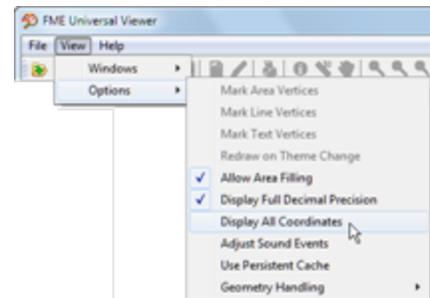
File > Save Viewspace is used to save the current Viewer setup, including details of the open datasets and the symbology/layout used to display them.

Saving a viewspace allows the use to view the same data in the same way, at a later date, without having to open each dataset individually and without having to re-create the same symbology.

Display All Coordinates

The default coordinate list in the Information window is restricted to a maximum of 50 coordinates.

This optional restriction can be turned off by selecting **View > Options > Display All Coordinates**.



Decimal Precision Display

By default, coordinates are displayed in a truncated format. To show coordinates full decimal precision select **View > Options > Display Full Decimal Precision** from the menubar.

Full precision is vital for checking geometric processes; for example, examining a polygon to check it closes correctly.



Example 3: FME Universal Viewer	
Scenario	FME user; City of Interopolis, Planning Department
Data	Schools (GML), Parks (MapInfo TAB), Airport (QLF), Railroad (SDL)
Overall Goal	Examine city data to locate a suitable property
Demonstrates	Viewing and saving data with FME Universal Viewer
Starting Workspace	None
Finished Viewspace	None

The Mayor of Interopolis is planning to purchase a house and wants you to examine the City’s data to locate a good part of the city to live, that fulfills the following requirements.

1. In a middle school catchment area
2. In an area that includes at least one city park
3. In a quiet area (so one that doesn’t include the airport or any railroad)

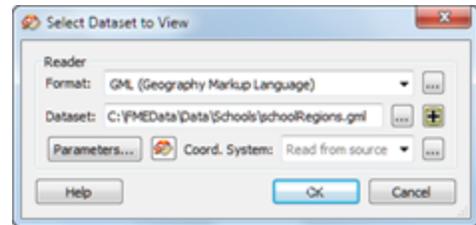
1) Start the FME Universal Viewer.

Start the FME Universal Viewer by selecting its shortcut from the Windows Start menu.

2) Open a Dataset – Schools

The first task is to open the school regions dataset.

Select **File > Open Dataset** from the menu bar in FME Universal Viewer. The standard FME dataset selection dialog will open.

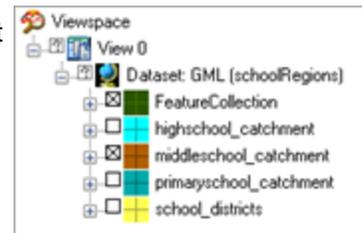


Set the following parameters:

- Format** GML (Geography Markup Language)
- Dataset** C:\FMEData\Data\Schools\schoolRegions.gml

The dataset opened contains information about all types of schools in the city, but you are only interested in middle schools.

In the Display Control window, uncheck the boxes for high schools, primary schools and school districts.



3) Add a Dataset – Parks

Additional datasets may now be added to the view.

Select **File > Add Dataset** from the menu bar. Again the dataset selection dialog will open.



This time choose the following parameters:

Format	MapInfo TAB (MFAL)
Dataset	<u>C:\FMEData\Data\Parks\city_parks.tab</u>

Notice how the data is added to the existing View, rather than opening a new View.

4) Add Datasets – Railroad and Airport

Now add the railroad and airport datasets. Do this by dragging-and-dropping the files from Windows Explorer into the current View window.

The dataset selection dialog will now open again; this time the settings will already be filled in. FME has determined them from the dataset you have chosen.

Format	CITS Data Transfer Format (QLF)
Dataset	<u>C:\FMEData\Data\Airport\airport.qlf</u>

Format	Autodesk MapGuide SDL
Dataset	<u>C:\FMEData\Data\Railroads\railroad.sdl</u>

5) Adjust the View

One potential problem is that datasets added to the View may be hidden beneath the original school dataset. To be visible they need to be promoted above the schools.

If necessary, promote the railroad dataset (for example) click the 'Dataset: SDL (railroad)' item, keeping the mouse button depressed. Drag the dataset to the top of the display, above the school regions dataset. Release the mouse button. Repeat for the parks and airport datasets as required.

6) Adjust Symbology

In the Display Control window click the color icon for a dataset to change the color of features for that dataset. Use the following colors:

School Areas	Gray
Parks	Green
Airport + Railroad	Red

7) Querying

Use the windowing tools on the toolbar to zoom in closer to examine each school area. Visually determine which school areas contain parks, but not railroads or the airport.

By now you should be able to pick a single school area that is the most suitable for the mayor's requirements. Use the Query tool to discover the names of that area.

8) Export Data

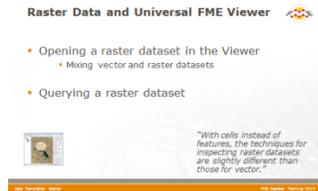
The mayor's office has phoned up and says he would like to see the map data as a visual display. He doesn't have access to FME or the source data, but he does have Google Earth.

Use **File > Save Data As** to save the data as KML so that the mayor can view it in Google Earth.



Open the data yourself to verify that it looks correct (albeit not very well stylized).

Raster Data and FME Universal Viewer

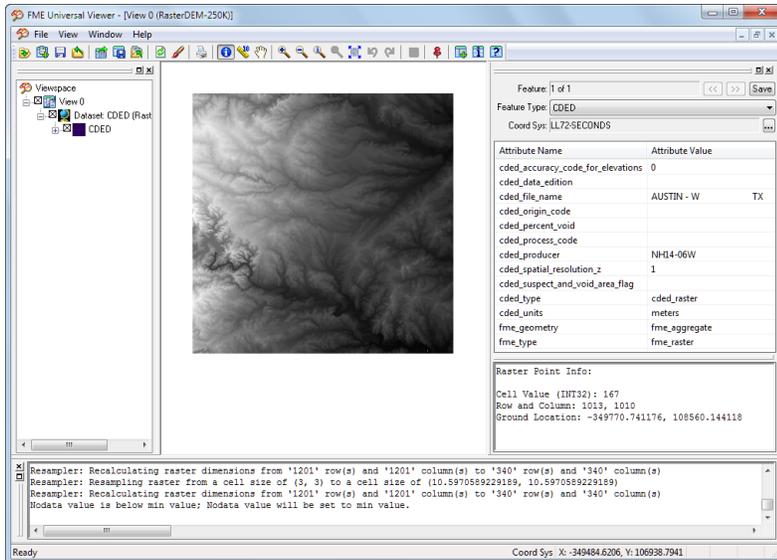


With cells rather than features, the techniques for inspecting raster datasets are slightly different to those for vector.

Opening a Raster Data in the Viewer

The same open and add tools work for opening both raster and vector datasets.

For example, here is the Interopolis CDED format elevation model (DEM) Dataset opened in the FME Universal Viewer. The value of the selected cell is reported in the lower-right corner of the information window.



Mixing Vector and Raster Datasets

The FME Universal Viewer is capable of showing both raster and vector datasets simultaneously.

Here is a MrSID raster dataset overlaid with the Interopolis parks (vector) dataset.

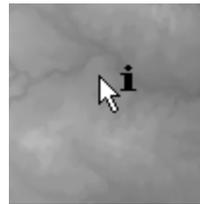


Querying Raster Features

There are two ways to use the query tool on a raster dataset: one to query a cell and one to query the raster dataset itself.

Querying a Cell

A single click onto a raster dataset queries the cell under that click. This gives the output shown.



Attribute Name	Attribute Value
cded_accuracy_code_for_elevations	0
cded_data_edition	
cded_file_name	AUSTIN - W TX
cded_origin_code	
cded_percent_void	
cded_process_code	
cded_producer	NH14-06W
cded_spatial_resolution_z	1
cded_suspect_and_void_area_flag	
cded_type	cded_raster
cded_units	meters
fme_geometry	fme_aggregate
fme_type	fme_raster

Raster Point Info:	
Cell Value (INT32):	167
Row and Column:	917, 956
Ground Location:	-349933.092233, 108849.694175

Attribute Name	Attribute Value
cded_accuracy_code_for_elevations	0
cded_data_edition	
cded_file_name	AUSTIN - W TX
cded_origin_code	
cded_percent_void	
cded_process_code	
cded_producer	NH14-06W
cded_spatial_resolution_z	1
cded_suspect_and_void_area_flag	
cded_type	cded_raster
cded_units	meters
fme_geometry	fme_aggregate
fme_type	fme_raster

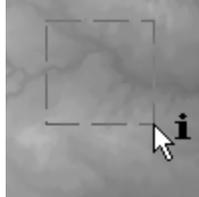

```

Coordinate Listing:
(first and last 25 coords)
-----
Geometry Type: Raster

Number of Rows      : 1201
Number of Columns   : 1201
Cell Origin         : 0.5, 0.5
Cell Spacing        : 3, 3
Origin              : -352801.5, 111601.5
Extents             : (-352801.5, 107998.5),
Rotation            : 0
Number of Bands     : 1
    
```

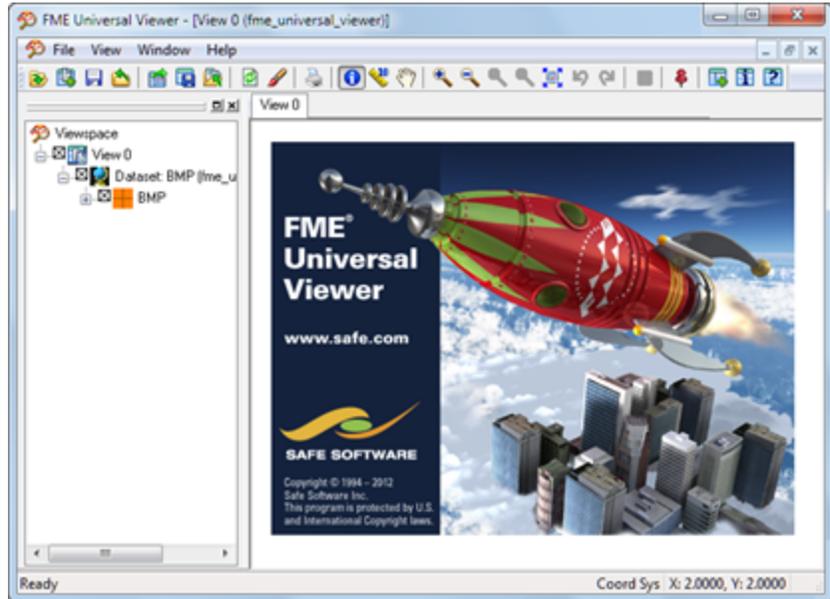
Querying the Dataset

A click and drag motion that selects an area of the raster dataset queries the dataset itself. This gives the output shown.



Here is one example of a bitmap image (an FME splash screen) opened in the FME Universal Viewer.

This is a good example of a color image—each cell value is a triplet that defines the RGB cell color (the green in Safe’s logo is 147, 193, 78).



Example 4: Handling Raster Data	
Scenario	FME user; City of Interopolis, Planning Department
Data	Raster image (LizardTech MrSID)
Overall Goal	Convert data from MrSID format to JPEG format.
Demonstrates	Raster data translation and inspection
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example4Complete.fmw



Judge GIS says...

‘Members of the Jury: Safe Software suggests that raster data is as easy to handle in FME as vector data. Please pass judgment on this claim by doing this example translation!’

1. Start FME Workbench
2. Set up Translation
Set up a translation using the following parameters:

Reader Format LizardTech MrSID

Reader Dataset *C:\FMEData\Data\Raster\130105.sid*

Writer Format JPEG (Joint Photographic Experts Group)

Writer Dataset *C:\FMEData\Output\DesktopTraining*

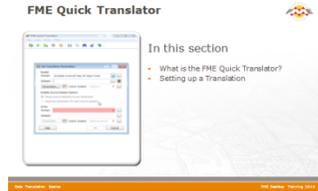
For now, ignore the Workflow Options and leave the default of 'Static Schema'. On this occasion, the writer dataset demands a folder, and not a specific filename.

3. Run Translation
Run the translation.
4. Inspect Output
Inspect the output from this translation using the FME Universal Viewer.



*Once you've finished the exercise, press Ctrl+W in Workbench.
That's a new shortcut in FME2013 to close the workspace.*

Translation Previews



A key ability of FME is communication of data between Workbench and the FME Universal Viewer.

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 Workbench translation that updates a spatial database. That way mistakes can be detected before writing to the database.

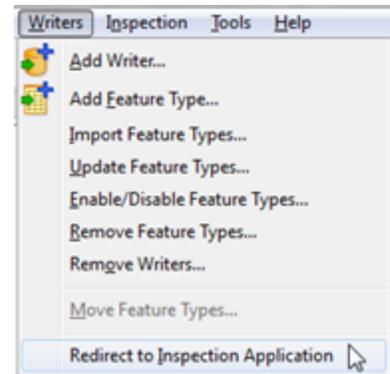
The Workbench tool to do this is the 'Redirect to Inspection Application' setting.

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

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

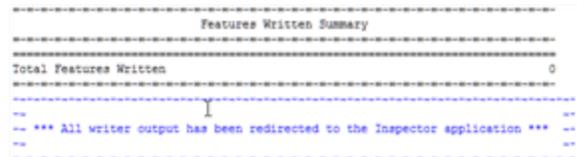
This setting is a toggle, which means that each subsequent selection alternately turns the setting on and off.

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



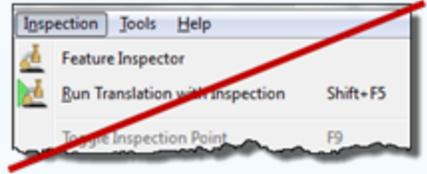
An embarrassing problem occurs when a user forgets to deactivate the setting and does not understand why no output datasets are being written. To help combat this issue, the FME Log window includes a distinctive warning message when data is being redirected.

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

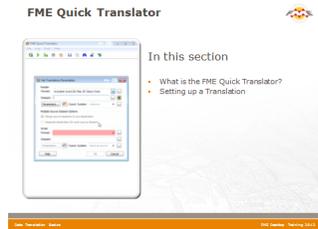




Don't get confused by the Inspection option on the menubar. It is entirely unrelated to this form of Inspection and the Viewer.



Introduction to FME Quick Translator



The FME Quick Translator is a new incarnation of the FME Universal Translator, the original FME tool for Data Translation and Transformation.

What is the FME Quick Translator?

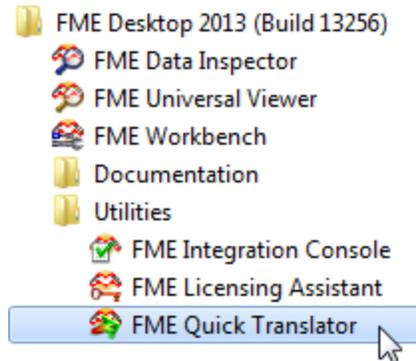
In a previous example FME Workbench was used to carry out what is known as a "quick translation".

The FME Quick Translator is designed to carry out quick translations without having to go through the workspace creation process. This is quicker (hence the name) but cannot be saved, so is not reusable.

It is also designed to run existing workspaces that do not need further editing.

Starting the FME Quick Translator

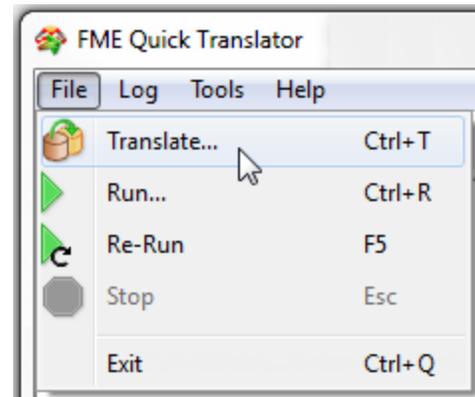
Find FME Quick Translator in the **FME Desktop > Utilities** sub-menu in the Windows start menu. Click on the sub-menu entry to start the FME Quick Translator.

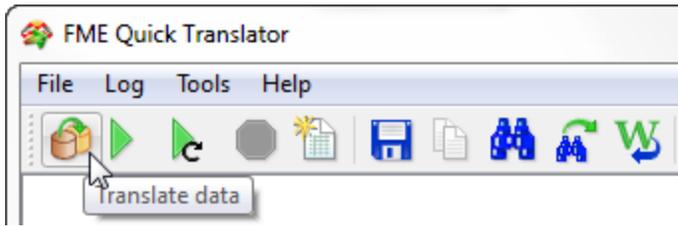


Setting up a Translation

To set up a translation, choose the option **File > Translate** on the menu bar.

Translate is also available as a button on the toolbar.





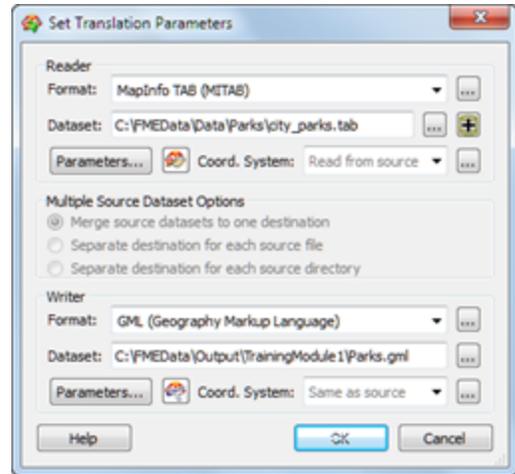
Translation Dialog

Selecting **File > Translate** on the Quick Translator menu bar opens the translation parameters dialog.

When the translation parameters have been filled in and the OK button clicked, the translation is immediately carried out.

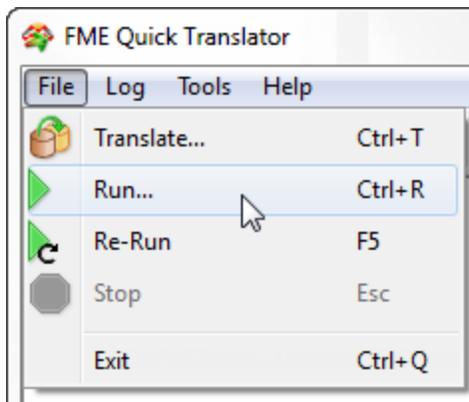
The Quick Translator translation parameters dialog looks like this.

Here the user is converting a MapInfo TAB dataset into GML (Geography Markup Language).



Running an Existing Translation

The FME Quick Translator may also be used to execute an existing translation, whether it's a workspace or a mapping file.



This is of use where a user has been supplied with a mapping file or workspace that does not require further editing (and may even have been password protected to prevent changes from being made).



Example 5: FME Quick Translator	
Scenario	FME user; City of Interopolis, Planning Department
Data	Railroad (Autodesk MapGuide SDL)
Overall Goal	Translate the Engineering department’s railroad dataset from Autodesk MapGuide SDL to the Planning department’s preferred format, MapInfo MIF
Demonstrates	Quick Translation in FME Quick Translator
Starting Workspace	None
Finished Workspace	None

Your objective is to convert the Engineering Department’s dataset of railroad data from the Autodesk MapGuide SDL format to the Planning Department’s preferred format, which is MapInfo MIF/MID

1) Start FME Quick Translator

Start the FME Quick Translator and select **File > Translate** from the menubar.

2) Set up Translation

In the Translation Parameters dialog, set the Reader and Writer parameters as follows:

Reader Format	Autodesk MapGuide SDL
Reader Dataset	C:\FMEData\Data\Railroads\railroad.sdl
Writer Format	MapInfo MIF/MID
Writer Dataset	C:\FMEData\Output\DesktopTraining

Note: When writing MIF/MID format you select a folder and not a specific file name.

3) Run Translation

Click **OK** to run the translation.

4) Inspect Data

Inspect both the source and the newly-created destination datasets, to prove that they are the same and that the translation was successful.

5) Advanced Task

Repeat the same translation in FME Workbench, and then from the command line!

The command syntax is shown in the FME Workbench Log window but – *provided you browse to the folder in which the workspace resides* – you can also attempt the following:

```
fme <workspacename>.fmw
```

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 **Workspace dialog** and the **Workspace wizard** are the two methods 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 Universal Viewer** is a tool for inspecting data. Multiple datasets – including data from different formats – can be opened within the same View window.
- **FME Quick Translator** is an application to carry out Quick Translations, or to run pre-defined workspaces.
- FME has a wide range of **supported geometries**

FME Skills

- The ability to start FME Workbench, set up a Quick Translation, and run it.
- The ability to start FME Universal Viewer, 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 Universal Viewer functionality for debugging data and translations.
- The ability to inspect both raster and vector dataset types.
- The ability to run the Quick Translation within the FME Quick Translator.
- The ability to inspect data by redirecting it from FME Workbench to FME Universal Viewer.

Q&A Answers



Miss Vector says...

Here are the test answers. Anything less than 4 out of 4 and you'll find yourself reviewing the section again during lunch!

Which of the following words describe FME Desktop?

1) Distributed

2) Semantic

3) Transformational

4) Centralized

Which of the following applications are parts of FME Desktop?

1) FME Workbench

2) FME Server

3) FME Universal Translator

4) FME Universal Viewer

Which of the following tools is not found in FME Workbench?

1) A data viewing tool

2) A source data selection tool

3) A destination data selection tool

4) Data manipulation tools

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.

Appendix A: Supported Geometries



To effectively support all the formats that it does, FME has a comprehensive geometry model that includes everything from the simplest geometry to the most complex.

It's important to be familiar with geometry types you will encounter within FME, particularly with how they are reported within the FME Universal Viewer. This section is a comprehensive list of these geometry types and includes items you may not need to be personally familiar with.

There are some geometries that are expected to be supported by most, if not all, data formats. Other geometries are more format-specific: they may not appear in every format and not every user would need to know about them.



See `C:\FMEData\Data\DemoData\GeometryExamples.ffs` for examples of most of the basic geometries.

Non-Geometry

A *non-geometry* feature is a set of attributes without geometry.

The commonest non-geometry feature is a simple database record	ID	1234
	Street_NameCity	Station Rd Cowfold

Points

Point geometries are features represented by a single coordinate.

A simple point feature has just an x/y coordinate; but there are other point geometry subtypes.	
Ellipses are basically circle or oval features; like an arc but closing on themselves, so don't require a start angle or sweep angle.	
Each Point Cloud feature is made up of a large number of unconnected point features.	

Text

Text features represent the position of an annotation.

<p>Text features' definition optionally includes size, rotation, and justification. Although text features are traditionally thought of as a single x/y coordinate, they can have a line or other geometry.</p>	
---	---

Lines

A line is a series of points strung together to form a chain.

<p>The simplest form of line geometry is a <i>two-point line</i>; that is, it has a start coordinate and an end coordinate but no intermediate points.</p>	
<p>A <i>polyline</i> feature is a multi-point line; that is, it has a start coordinate and an end coordinate plus a number of intermediate points.</p>	

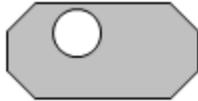
Arcs

Arcs are a bit of a special case. They can be defined in multiple ways with FME, and don't really fall under either point or line features.

<p>Arc features are often defined by an x/y coordinate at their centre point, plus a mathematical definition of arc radius and sweep angles.</p>	
<p>An alternate arc definition is the centre point, plus the two end point X/Y coordinates</p>	
<p>A further arc definition is the two end points, plus the mid-point of the arc 'line'.</p>	

Polygons

A polygon feature is a series of points strung together, whose first and last points coincide to form a closed shape.

<p>A simple polygon forms a single closed shape.</p>	
<p>A donut is made up of an outer boundary to define a perimeter and an inner boundary to represent a hole within it.</p>	

Note: Usually donut boundaries are polygons, but FME can support an ellipse used as either the inner or outer boundary (or both)!

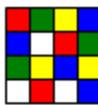
Aggregates

An aggregate is a defined set of any of the above features. It is also sometimes known as a ‘collection’ or ‘group’.

<p>A homogeneous aggregate is made up of features of the same geometry type.</p>	
<p>A non-homogeneous aggregate is made up of features with differing geometry types.</p>	

Raster

A raster geometry feature is a set of pixels or cells in a grid.

<p>A Raster feature is a set of cells, not an individual cell. Cells do not generally have attributes, but may possess a single value or color.</p>	
---	---

Paths

A path (also called chain) is a linear object made up of a number of features connected together.

<p>A path can be made up of features of the same geometry type or, more commonly, of different geometry types; for example, line-arc-line.</p>	
--	---

Surfaces

Surfaces are three dimensional planar features. There are many types of surfaces supported in FME, including faces (see illustration below), meshes, triangle strips, and triangle fans.

<p>A face is one example of a surface feature. It's a planar polygon or donut stored as a true 3D feature.</p>	
<p>A surface may contain holes, in much the same way as a donut polygon.</p>	

Solids

Solids are three-dimensional entities. There are many types of solids supported in FME. These include boxes (see illustration below), extrusions, b-rep solids, and CSGs (see illustration). Although similar, these different types of solids are required for compatibility with the full range of FME-supported formats.

<p>A <i>Box</i> is one example of a solid feature. Like a hole or donut, it can contain a void within it.</p>	
<p>A <i>CSG</i> (Constructive Solid Geometry) is a complex object made up of a set of solids upon which a Boolean operation has been carried out. Boolean operations that can be carried out are <i>Union</i> (right), <i>Difference</i>, and <i>Intersection</i>.</p>	

3D Multiples

Multiples are the 3D equivalent of aggregates. They are always homogeneous, which means that they are made up of the same type.

<p><i>Multi-surfaces</i> are one example of this type of geometry. They are multiple surface features related as a collection. A Composite Surface is a multi-surface where all items are connected topologically.</p>	
--	---

Named Geometries

All features can be given a name to their geometry. This is achieved with the transformers: *GeometryNameSetter*, *GeometryNameExtractor*, and *GeometryNameRemover*.

<p>Naming can be 'recursive'; for example, this feature is an aggregate named 'as-built', whereas its components are named 'roads'.</p>	<pre> Geometry Type: IFMEMultiCurve Name(utf-8): 'AsBuilt' Number of Curves: 2 ----- Curve Number: 0 Geometry Type: IFMELine Name(utf-8): 'Road' </pre>
---	---

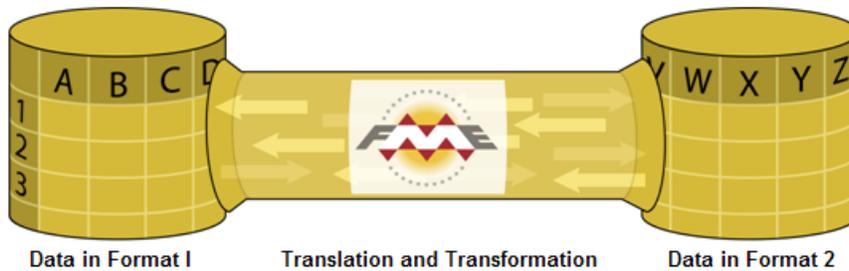
Data Transformation



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; the classic ETL (Extract-Transform-Load) configuration, as shown in the graphic below.



Notice here how the attributes A, B, and C are transformed into a new set of attributes called W, X, Y and Z. The arrows indicate the translation is also bi-directional.

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...

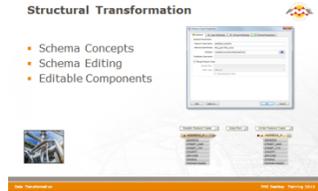
"Before you start on transformation, here's a challenge for you!

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. One could call this a 'physical' schema since it's a physical representation of the data.

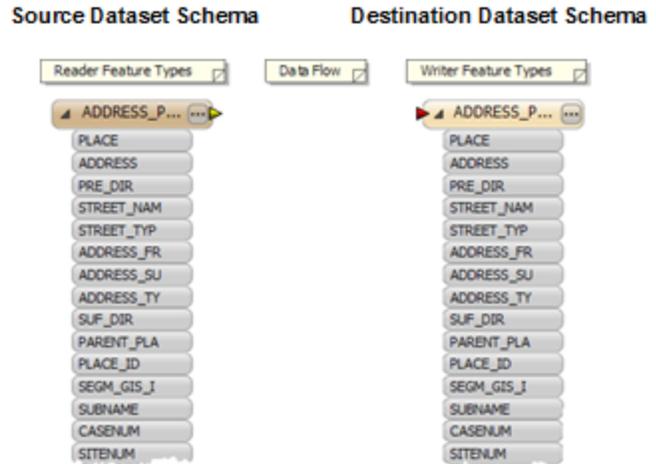
How Does FME Represent Schemas?

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.

To be technical, destination schemas could be called 'logical' schemas because they don't physically exist at this point. Until the workspace is run, they're just one potential output.

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.

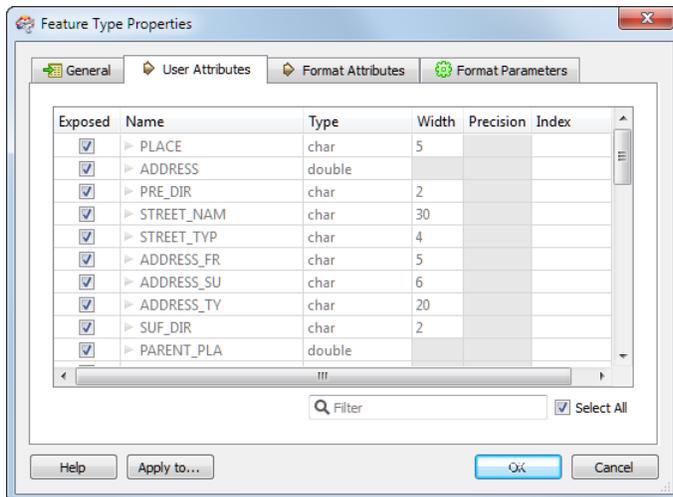
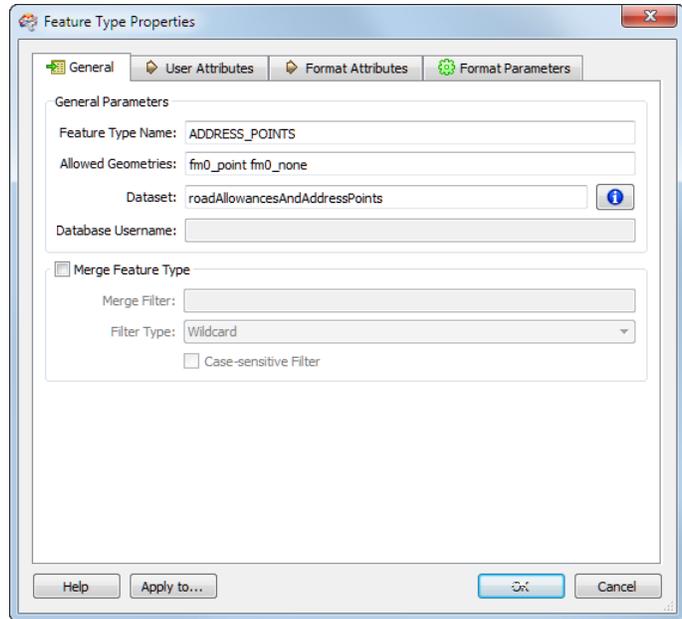
Some parts of the schema relate specifically to a single feature type only. Attributes are one such component. These components are shown in the Properties dialog of a feature type.

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 ADDRESS_POINTS.

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



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. Source attributes cannot be edited (by default) 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'.

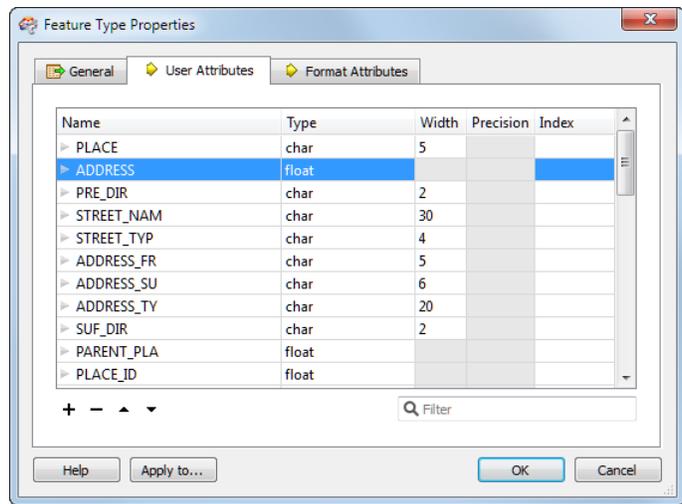
Editable Components

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 ADDRESS to BUILDING_NUMBER.

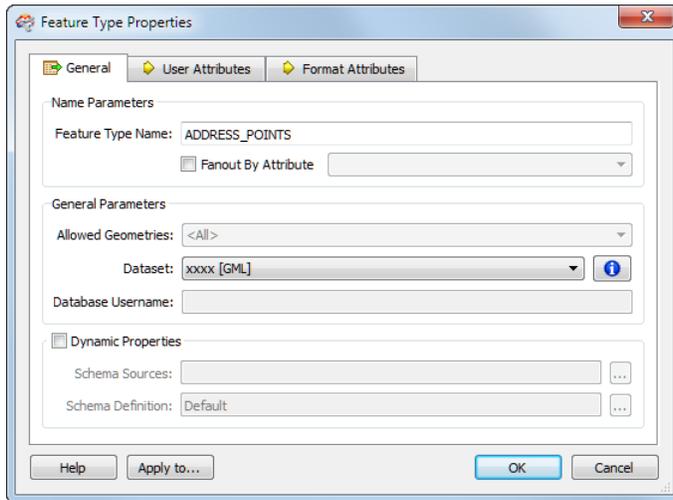
To rename an attribute open the Feature Type Properties dialog and click the User Attributes tab. Click the attribute to be renamed and enter the new name.



Attribute Type Changes

Any attribute on the writer schema can have a change of type; for example, changing ID from an integer to a float.

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 ADDRESS_POINTS 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.

Note: 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.

Domain Mapping

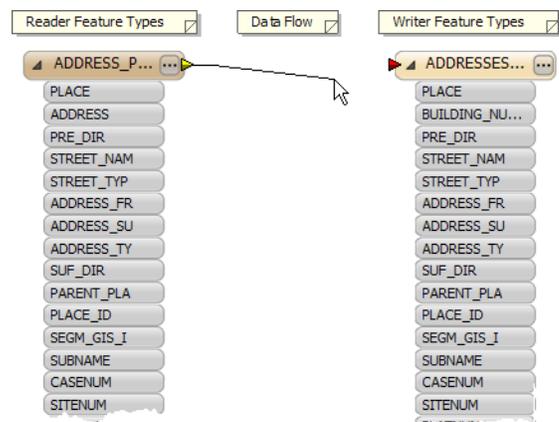
Some formats support Domains, which are a way of defining what values are allowed in a particular attribute. With Coded Domains (where each value is given a Code Number), mapping is sometimes required to convert values from the Code number to the descriptive value, or vice versa.

Domain mapping is also required when attributes need to change from one domain to another.

Feature Mapping in FME Workbench

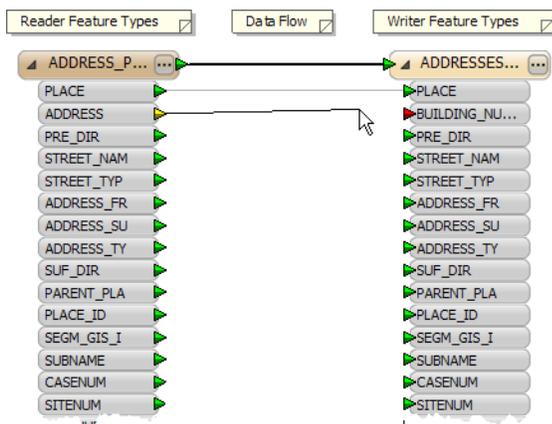
Feature Mapping is carried out by clicking the output port of a source feature type, dragging the arrowhead across to the input port of a destination feature type, and releasing the mouse button.

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



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.

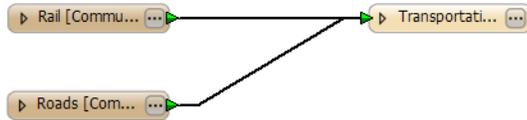
Feature mapping connections (or links) are shown with a thick, black arrow.

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.



Here a user needs a single layer called Transportation in their output, and so is merging two input Feature Types (Rail and Roads) into one output Feature Type (Transportation).



Example 6: 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:\FMEData\Workspaces\DesktopManual\Example6Complete.fmw

Here, 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 (MFAL)
Reader Dataset C:\FMEData\Data\Parks\city_parks.tab

Writer Format MapInfo TAB (MFAL)(yes – here we write back to the same format!)
Writer Dataset C:\FMEData\Output\DesktopTraining

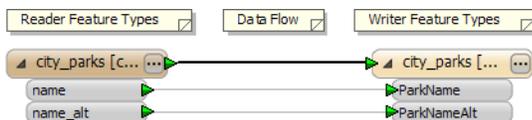
2) Rename Attributes

FME creates a workspace where the destination schema matches the source.

However, the end user of the data has requested the attributes get more meaningful names.

Make sure the list of attributes is expanded on the writer feature type, right-click the *name* attribute and choose the option to **Rename Attribute**. In the field provided change the attribute to *ParkName*.

Now repeat the process for *name_alt*, this time renaming it to *ParkNameAlt*.



3) Add Attributes

Open the Feature Type Properties dialog for the writer feature type by clicking the Properties button or right-clicking the feature type and choosing **Properties**.

Click the User Attributes tab to open the list of destination attributes.

Add a new attribute	<i>ParkArea</i>	[make sure it's attribute type of FLOAT]
Add a new attribute	<i>AverageParkArea</i>	[make sure it's attribute type of FLOAT]

4) Rename Feature Type

Now click back on the General tab.

Click in the field labelled Feature Type Name and change the name from *city_parks* to *ParksWithArea*

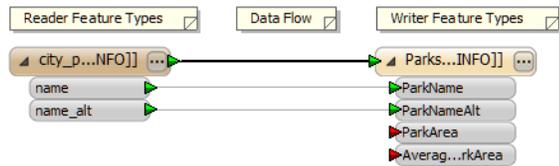
Click **OK** to accept these changes.

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

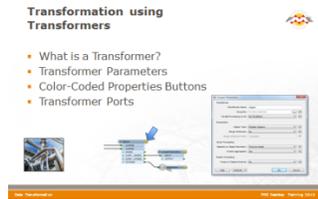
5) Save the Workspace

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

Note: *There is nothing yet to map to ParkArea or AverageParkArea because values for them do not yet exist, 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.

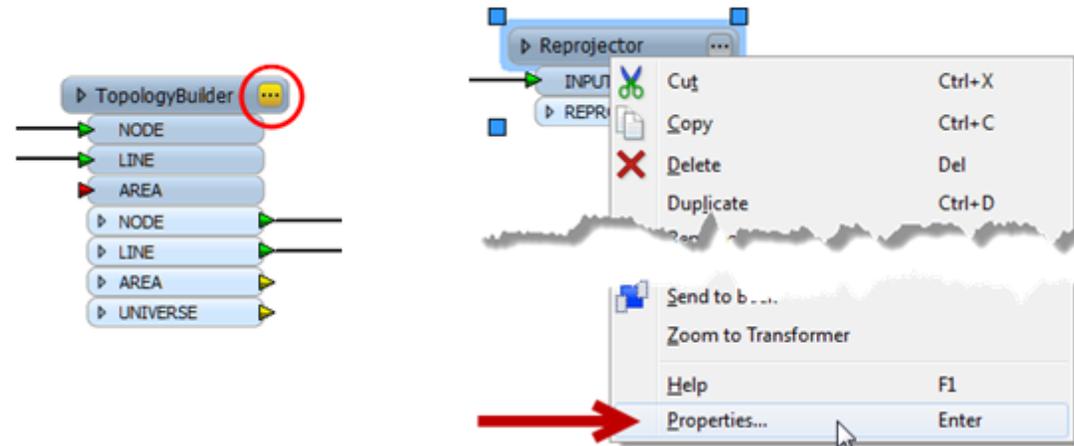
What is a Transformer?

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 colours.

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



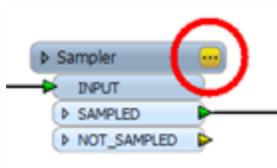
Each transformer may have a number of parameters – also known as settings. Access the settings by either clicking the Properties button at the top right of each transformer or by right-clicking a transformer and selecting the Properties option. Both options are shown here.

Color-Coded Properties Buttons

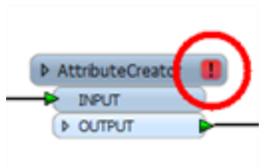
The Properties button on a transformer is colour-coded to reflect the status of the settings, described **below**.



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.

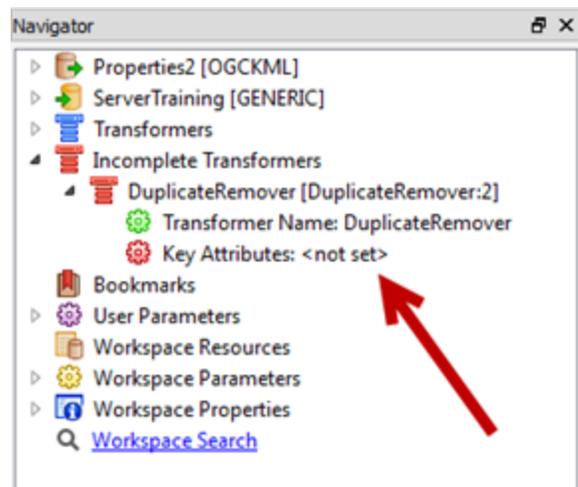


A **yellow** Properties button indicates that the default settings have not yet been checked. The transformer can be used in this state, but the results may be unpredictable.



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.

Incomplete transformers (those with a red Properties button) are also highlighted in red on the Navigator.

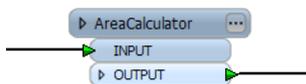




First-Officer Transformer says...
'When you have a red-flagged transformer, your translation just won't fly. You need to fix the settings before you can get off the ground.'

Transformer Colors

Transformers are all color-coded.



All regular transformers are colored blue like this *AreaCalculator* transformer.

Custom transformers have two different states: embedded or linked.



Embedded custom transformers are colored green whereas linked custom transformers are colored cyan. Don't worry if you don't know what a custom transformer is yet; they're covered later.

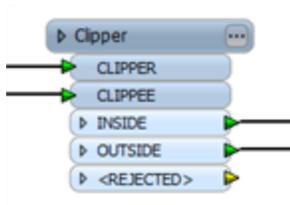
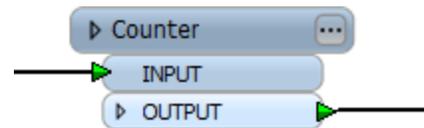


Some transformers, such as the *Inspector*, have their own distinctive icon.

Transformer Ports

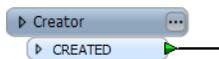
Far from having just a single input and/or output, a transformer can have multiple input ports, multiple output ports, or both.

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 *Logger* has just a single input port...



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



Example 7: Data Restructuring 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	Structural Transformation
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example7Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example7Complete.fmw

We'll now continue the previous example by filtering out golf courses from the source data (as these have a different scale of maintenance costs) and write them to the log window.

1) Start Workbench

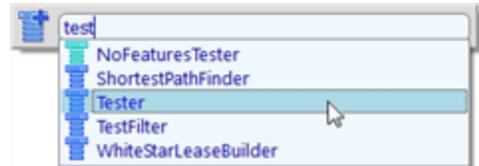
Start Workbench (if necessary) and open the workspace from Example 6.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example7Begin.fmw*

2) Select Connection

The first step is to filter out golf courses. This can be done with a *Tester* transformer.

Click on the connection between reader and writer feature types. Start typing the word “test”. A list of matching transformers appears.



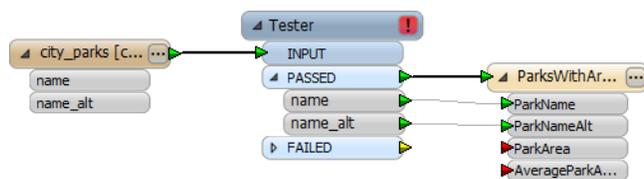
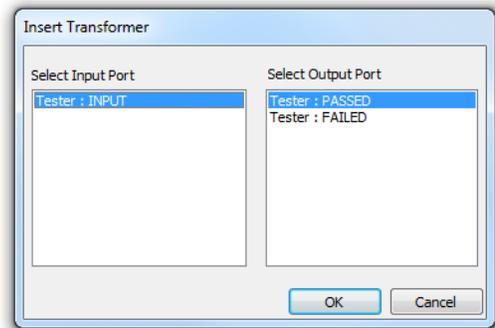
3) Select Transformer

Select the transformer named *Tester*. The transformer is dropped onto the Workbench canvas window. A dialog opens to ask which *Tester* output port to connect.

To make the final workspace more intelligible it’s best to set up the *Tester* so that any attribute NOT containing the word “Golf” will PASS the test.

So ensure the *Tester:PASSED* port is selected (it should be the default) and click **OK** to accept it.

The transformer will be dropped into position. With a little tidying the workspace will now look like this:



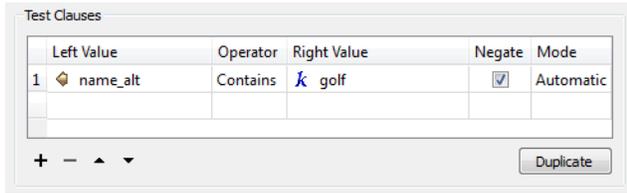
4) Set up test

The next task is to set up the test to be carried out by the *Tester*. The test is for where the alternate park name contains the word “golf”.

Click on the properties button for the *Tester* (it should be colored red). A properties dialog will open.

In the Test Clauses dialog, set:

- Left Value** Select ‘Attribute Value’ > name_alt
- Operator** Select the ‘Contains’ option
- Right Value** Type the word ‘golf’
- Negate** Tick the checkbox
- Mode** Automatic



Negate is important to set because we want to pass features where *name_alt* does NOT contain the word Golf. Click **OK** to accept the test.



Because this operator (Contains) mimics the similar SQL operation, this test is not case sensitive and you can use either “golf” or “Golf” or “GOLF”. However, this is the “exception that proves the rule”: most FME operations ARE case sensitive and do not work the same way.

Use the StringSearcher transformer for more complex string matching operations.

5) Handle Golf Courses

There are various ways to keep track of the features tagged as Golf Courses. The simplest is to record the data in the FME log file/window.

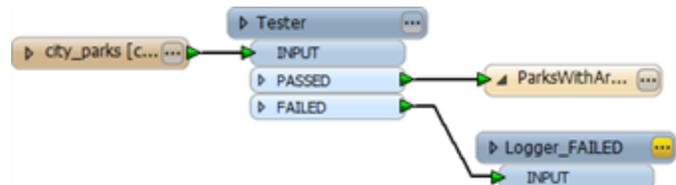
Right-click the Tester:FAILED port. Choose the option **Connect Logger**.

6) Run the Workspace

Run the workspace and check the Workbench log window to prove that there is a single golf course feature among the city parks. The *Logger* transformer will have recorded it in there under the title “Feature is:”

7) Save the Workspace

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



*In FME2013 Loggers inserted by this method are named after – and reported in the log with – the output port they are connected to, here *Logger_FAILED*.*



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*



FME allows a user to re-design the look and feel of the Workbench interface. However, the one thing these FME Themes (or 'skins') can't do is change the connection arrows.

Content Transformation

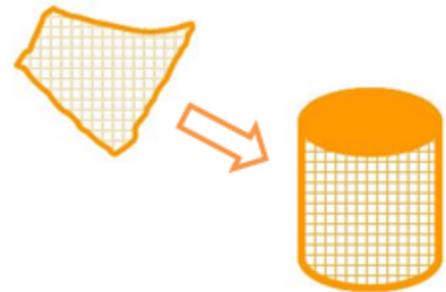


Content transformations are those that operate on the geometry or attribute content of a dataset.

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"

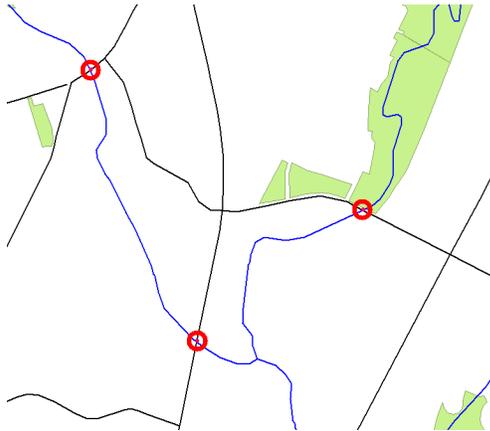
Geometric Transformation

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:

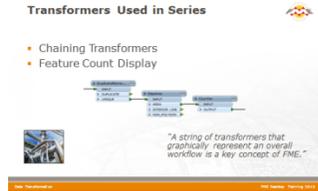
- *Concatenation* – joining together of two or more attributes
- *Splitting* – splitting one attribute into many, which is the opposite of Concatenation
- *Measurement* – measuring a feature's length or area to create a new attribute
- *ID Creation* – creating a unique ID number for a particular feature

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



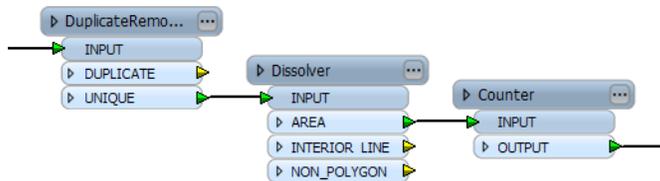
Much like a set of components in an electrical circuit, a series of Workbench Transformers can be connected together to have a cumulative effect on a set of data.

Chaining Transformers

Despite the large number of transformers available in FME, users frequently find that a single transformer does not meet their requirements. In this situation, a combination or chain of transformers must be used.

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.



Example 8: 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:\FMEData\Workspaces\DesktopManual\Example8Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example8Complete.fmw

Now we'll continue this project 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 Example 7.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example8Begin.fmw*

Examples 6 and 7 set up the translation for the park measurement task.

Now FME transformers can be used to measure the park area and calculate the average value.

2) Add an AreaCalculator Transformer

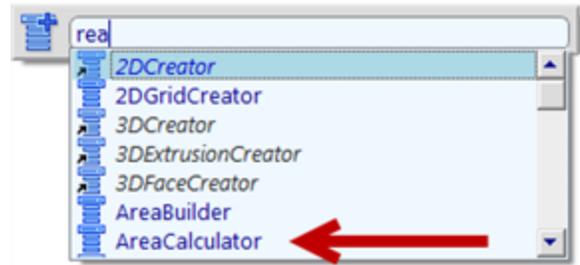
To measure the area of each park feature, an *AreaCalculator* transformer must be used.

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

There are many ways to add transformers to a workspace. One is to add the transformer onto the canvas using Quick Add and then drag-and-drop it into position.

Click onto a blank area of canvas and start typing the letters “areac”. You will see the Quick Add list of matching transformers appear beneath.

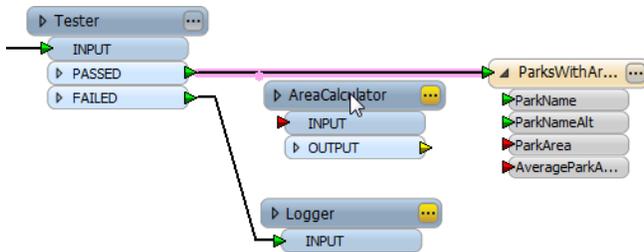
Select the transformer named *AreaCalculator*



3) Position Transformer

Now the transformer must be set into the correct position. Click on the *AreaCalculator* and start to drag it. A pink colored dot will appear in the top-left corner of the transformer.

Drag the transformer until the pink dot hovers over (and highlights) the connection between Tester:PASSED and the writer feature type MAPINFO:ParksWithArea.

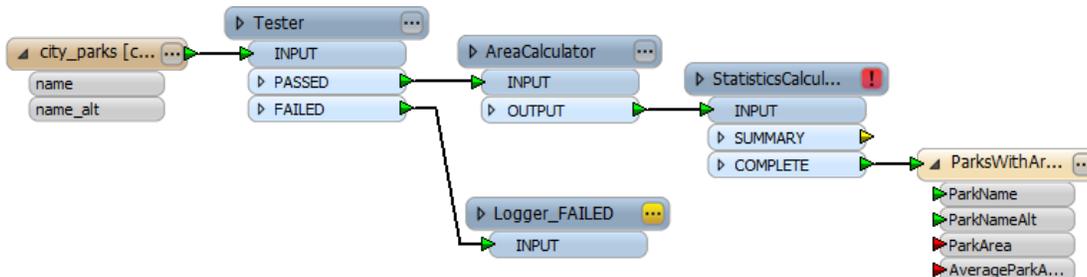


Release the mouse button to drop the transformer into position.

4) Add a StatisticsCalculator Transformer

Using the same method, place a *StatisticsCalculator* transformer between the *AreaCalculator*:OUTPUT port and the *MAPINFO:ParksWithArea* feature type. The output port labelled *COMPLETE* is the one we need to connect.

With a little tidying up the workspace should now look something like this:



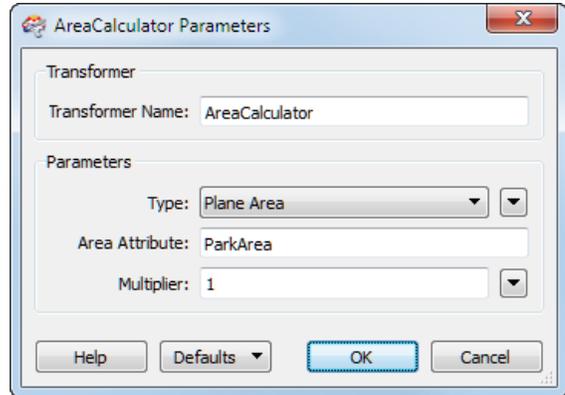
5) Check AreaCalculator Settings

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 *ParksWithArea* schema requires an attribute called `ParkArea`, so change this parameter to create the correct attribute.

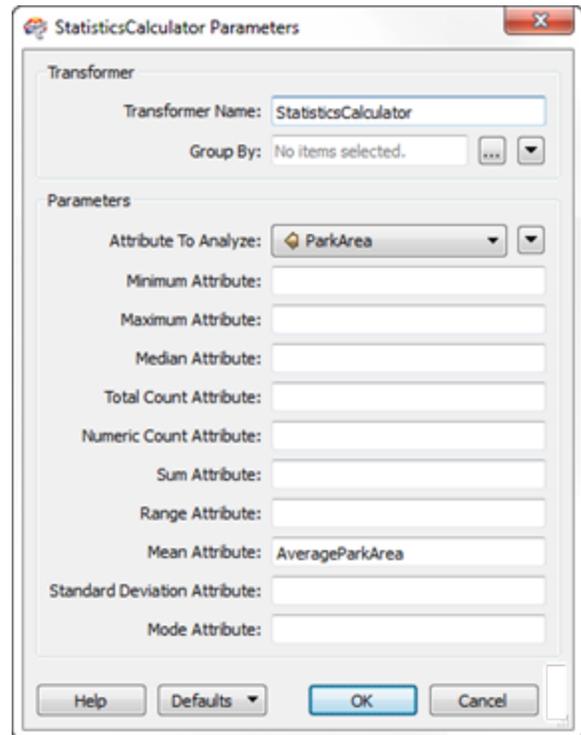


6) Check StatisticsCalculator Settings

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*.



In FME2013, the "Pass Through Features" parameter is replaced by separate output ports.

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

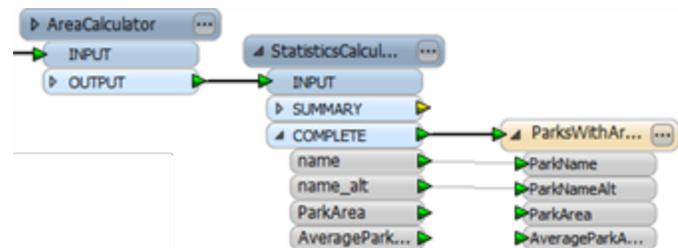
Change the attribute from `_mean` to `AverageParkArea`

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.

7) Examine Schema Mapping

The latter part of the workspace now looks like this:



Notice how the attributes `ParkArea` and `AverageParkArea` are mapped automatically to the writer. This is because their attribute names match those on the writer's schema.

This is the best type of schema mapping, because these connections cannot easily be broken.

Contrast that with the `name` and `name_alt` attributes. These are not connected automatically because the name differs from those on the schema. Such connections are more fragile.

Click on the Feature connection between the *StatisticsCalculator* and *ParksWithArea* object, and then press the delete key to delete it. Now replace it with a new connection. Notice how the attribute mapping is only repaired where the attribute names already matched.

8) Remap and Preserve Schema

Remap the attribute schema by dragging a connection from the `name` attribute port (yellow) to the `ParkName` attribute port (red). Repeat the process to create the `name_alt`/`ParkNameAlt` link.

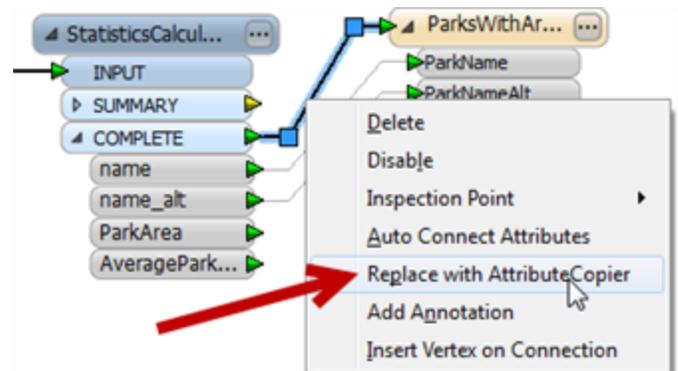
The new schema mapping should now be preserved, so that it cannot be broken again in the future. To do this requires attributes to be renamed within the workspace.

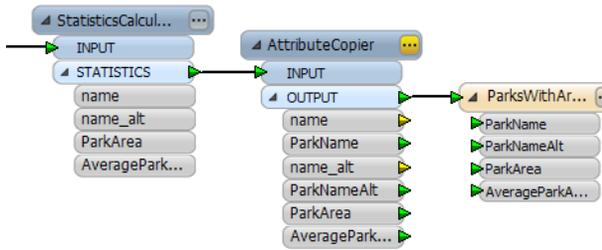
The *AttributeRenamer* or *AttributeCopier* transformers can be used to do this.

It is possible to just drop one of these transformers into position and set it up, but there is a short cut that can be used to replace the manual mapping automatically.

Right-click on the connection between the *StatisticsCalculator*:`COMPLETE` port and the *MAPINFO*:`ParksWithArea` feature type.

Choose the option to "Replace with *AttributeCopier*".





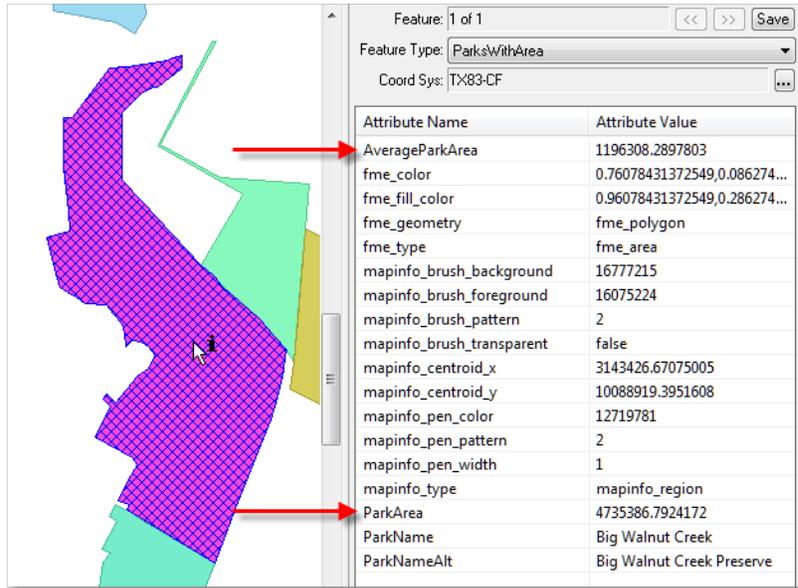
This will replace the manual schema mapping with automatic connections that cannot be interfered with by further editing.

9) Run the Workspace

Run the workspace.

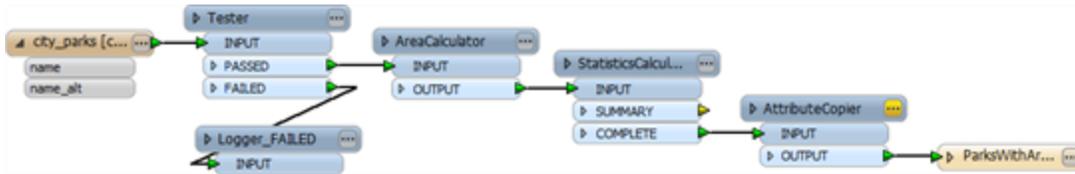
Inspect the result of the translation using the FME Universal Viewer.

Query a feature to find the average size of all the parks.



10) Save the Workspace

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

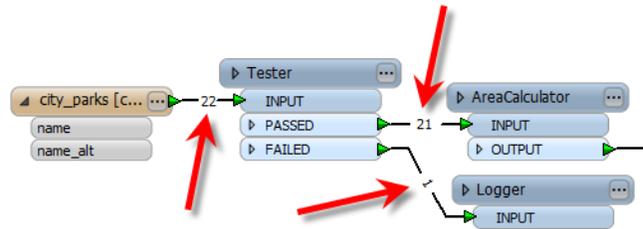


Feature Count Display – Serial Transformers

Look back at the previous example’s translation. Did you notice that after the translation was complete, each connection was marked with the number of features that had passed along it?

The feature counts show that 22 features were read, 21 passed the *Tester* while 1 failed (Golf Course) and went on to the *Logger*.

The Log window confirms the number of features written.



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

Transformer Handling



It's important to know all the ways to locate, place, and connect transformers.

At this point it's worth reviewing the different ways of handling transformers within Workbench.

Transformer Gallery

The transformer gallery is a window through which to locate and place transformers.

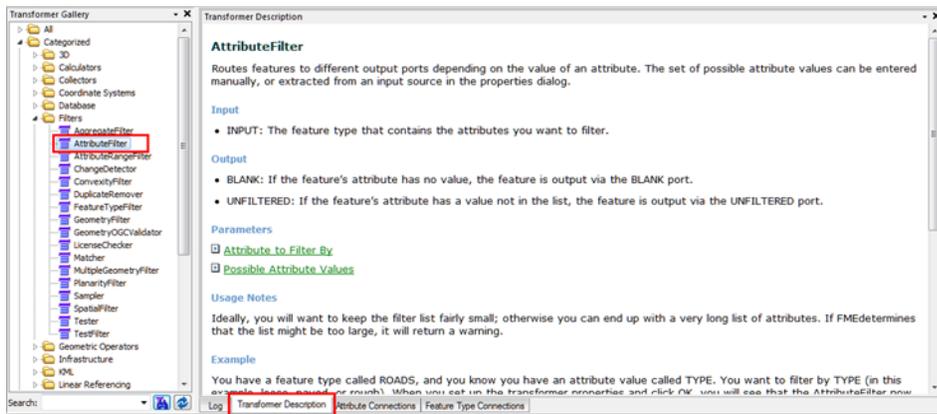
Along with the Transformer Description window it plays a key part in basic transformer placement.

The Transformer Gallery is divided into different sections and categories to help users find transformers. Expanding the Categorized folder displays the various categories. Expanding the All folder displays FME's full list of transformers (400+).

Another way to locate a transformer in the gallery is to type a keyword into the search field.

The transformer description window provides help for the currently selected transformer.

Here the Filters category of transformers is open and the *AttributeFilter* transformer is selected.



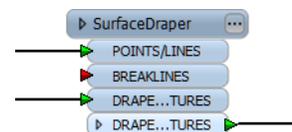
How to use the Transformer Gallery

After a transformer is located in the gallery, it can be double-clicked to place it on the canvas.

Alternatively, it can be placed by dragging and dropping it onto the workspace canvas. Many users drag and drop transformers because that gives them greater control over where the transformer is positioned initially.

When placed, users can add transformers into a workflow by dragging connections onto the transformer's input ports and dragging connections from the transformer's output ports.

Here a *SurfaceDraper* transformer has been placed and connected.



Quick Add

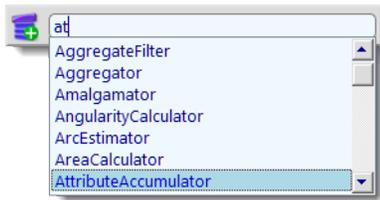
Quick Add is a way to search for and add transformers to the workspace. This makes it easily more convenient than the Transformer Gallery for most uses.

How to Use Quick Add

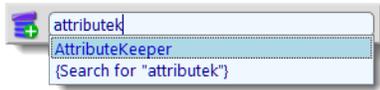
Quick Add is a dialog on the workspace canvas that is used to search for a transformer by name. To use it, click an area of blank space on the workspace canvas and type a letter on the keyboard; for example, a. At this point a small dialog opens that looks like this.



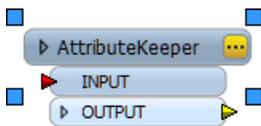
Continue to type (for example the letter t), and a list of transformers with names that match the typed content starts to appear.



The more letters typed, the smaller the list gets until the required transformer can be located.

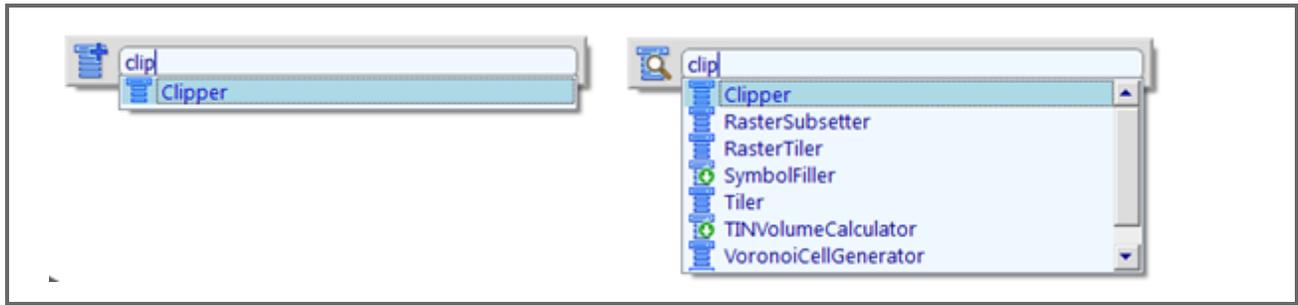


Pressing Enter (the return key) or clicking the transformer name causes it to be placed on the workspace canvas. Notice that it is pre-selected, so that pressing the return key again will now open the parameters dialog for the transformer.

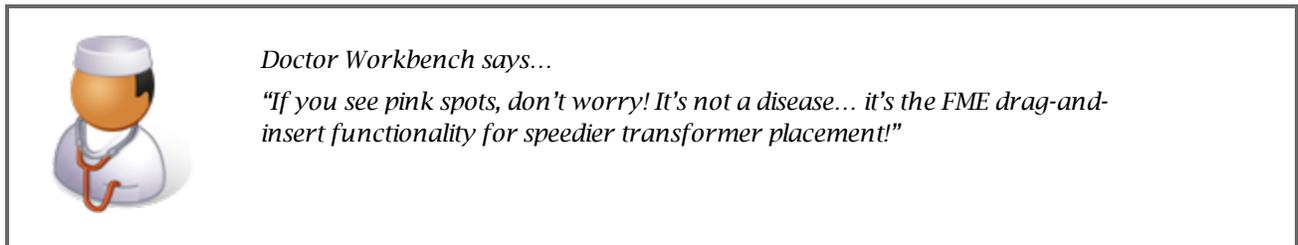


Sister Intuitive says...

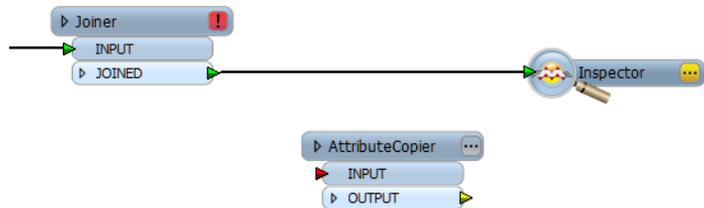
"Quick Add is one of my favorite things about FME. Now, in 2013 if you press the Tab key in Quick Add mode, you switch from a transformer name search to a transformer description search. That's truly heavenly!"



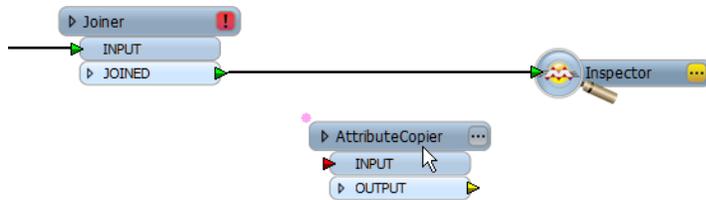
Drag-and-Insert



Here a user wants to insert this *AttributeCopier* transformer into the pipeline.

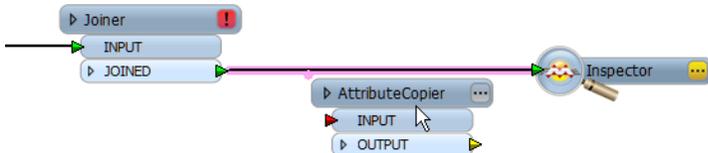


This functionality is activated automatically when a transformer is dragged across the canvas and shows as a small pink colored spot on the transformer's left-hand side.



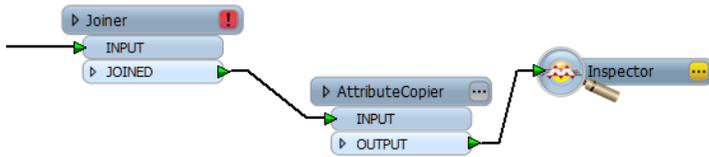
Now the user starts to drag the transformer and the pink spot appears.

Drag the transformer until the pink spot covers an existing connection. That connection becomes highlighted to indicate it is where the transformer is to be inserted.



Now the user drags the pink spot over the existing connection, which becomes highlighted.

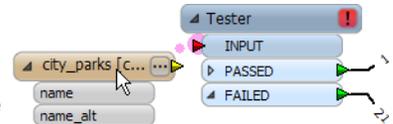
Release the transformer to drop it into position and connect it automatically.



The user releases the transformer and it is inserted into the pipeline.

Don't forget drag-and-insert (or even drag-connect) works on any workspace object, including feature types.

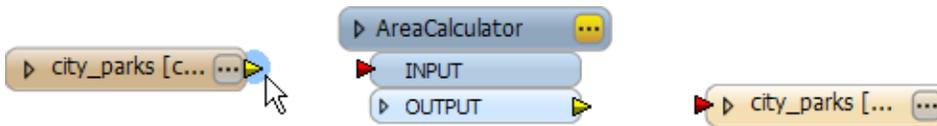
With this ability a feature type can be attached onto a transformer instead of the other way around.



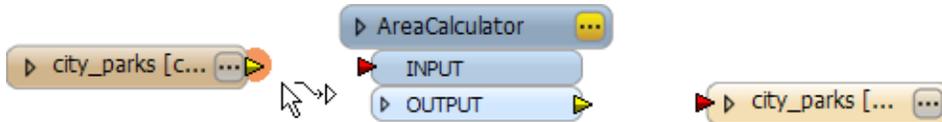
Quick Connect

Quick Connect is a way of making connections between transformers and feature types that does not involve dragging links between the two. All the user needs to do is click the appropriate input and output ports, and a connection is made.

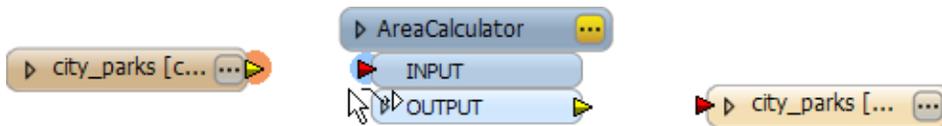
Hover over an output port and it is highlighted in blue.



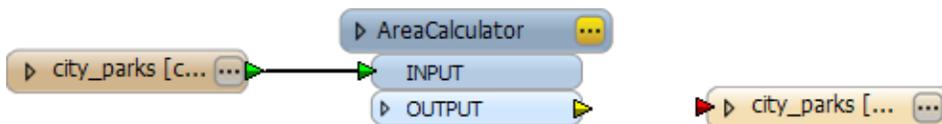
Select the port and it is highlighted orange. The cursor also changes shape.



Hover over an input port and it now highlights blue.



Select the port and the connection is complete.



When to use Quick Connect

Quick Connect is the best method to use when connecting two objects at far ends of a large workspace. Without Quick Connect the user would need to zoom out to a level at which both objects could be seen. This makes it difficult to create the connection.

With Quick Connect the two objects do not need to be on screen simultaneously. A user can click on the first port (at any zoom level) and use window panning tools to move to the next port.

Combining Quick Add, Drag-Insert and Quick Connect

Perhaps the best thing about these transformer shortcuts is that they are all fully integrated, so that Quick Add can be used in combination with either of the Drag-and-Insert and Quick Connect methods of placing transformers.

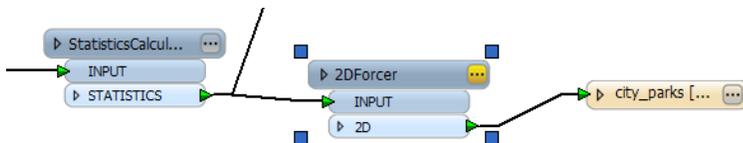
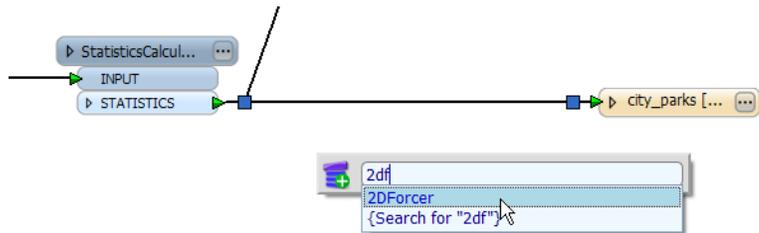
This integration operates in a number of ways.

If a connection is selected on the canvas already, when a transformer is placed using Quick Add the transformer is inserted automatically into that connection.



For example, the user first selects an existing link.

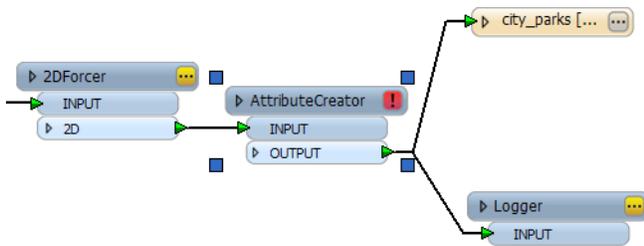
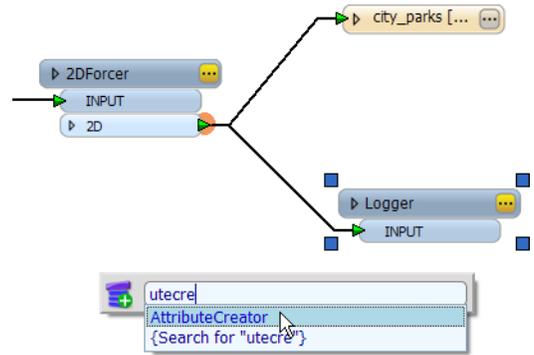
Now they select a transformer with Quick Add.



Then the new transformer is inserted directly into the chosen connection.

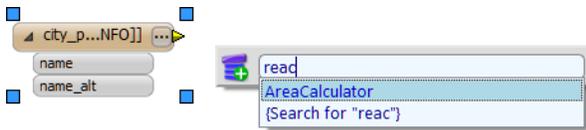
If a port is already selected on the canvas, the new transformer is inserted automatically or attached onto all connections on that port.

Here the user has selected an existing port:

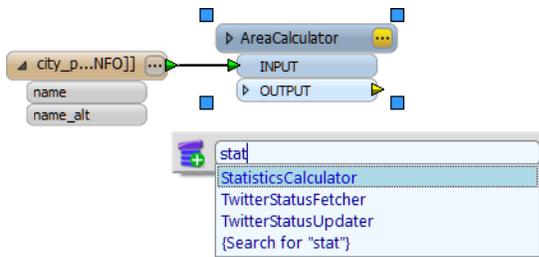


Quick Add adds the new transformer into both links emerging from the chosen port.

If a transformer or source Feature Type is already selected on the canvas, then the new transformer is connected automatically after the existing transformer.

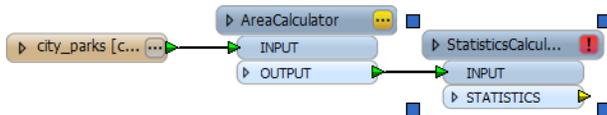


Using Quick Add a source Feature Type is selected while a transformer is added.



The new transformer is placed and connected to the selected Feature Type.

Because the new transformer is pre-selected automatically, the user can now continue adding transformers.



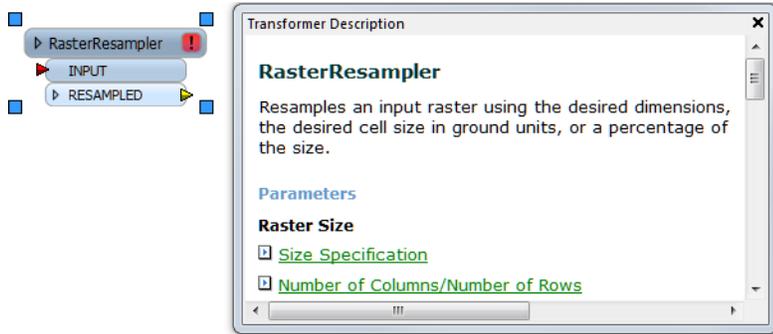
The second transformer is added after the first, and so on.

Transformer Description Window

As mentioned, it's possible now to keep the Transformer Gallery closed for most of the time, and to use Quick Add functionality instead. What really makes this feasible is that the Transformer Description is split off as a separate window.

Then the Transformer Gallery may be closed and the Transformer Description window retained.

Additionally, the Transformer Description window shows HTML-based help for whatever transformer is selected on the workspace canvas.



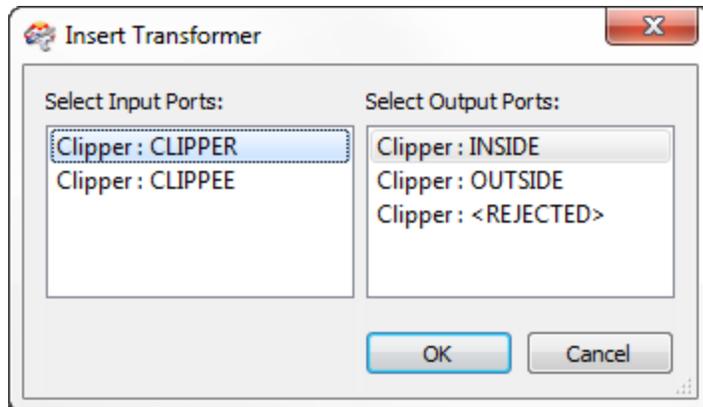
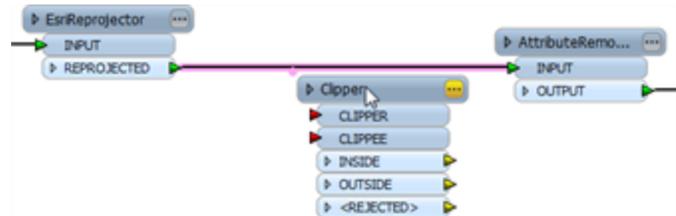
Multi-Port Transformers

When a transformer has only a single input and output port, FME is able to determine immediately the drag-n-insert connections to make.

However, many transformers possess more than a single input or output port and, therefore, an automatic connection cannot be made. In these cases, the user is prompted to define what connections are required through a dialog.

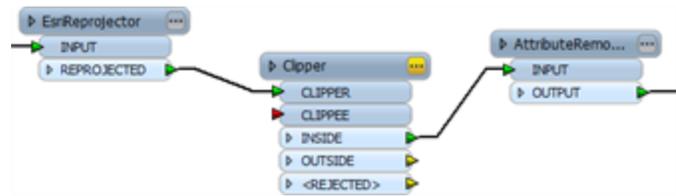
Here the transformer to be inserted is a *Clipper*.

The *Clipper* has two input ports and three output ports.



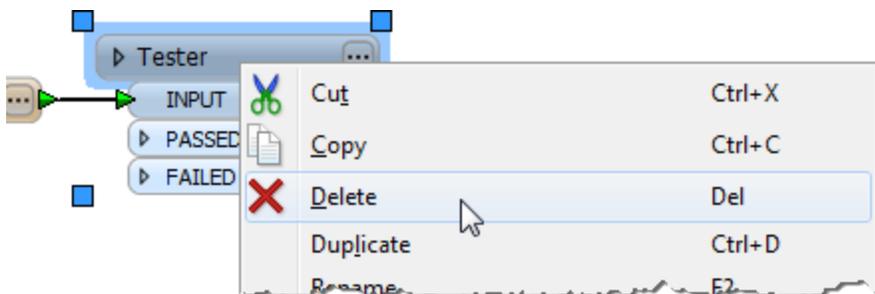
When the *Clipper* is dropped into position, FME opens a dialog and prompts the user to select which input and output ports are to be connected.

When the user clicks OK, Workbench makes the connection as defined by the selected ports.



Transformer Deletions

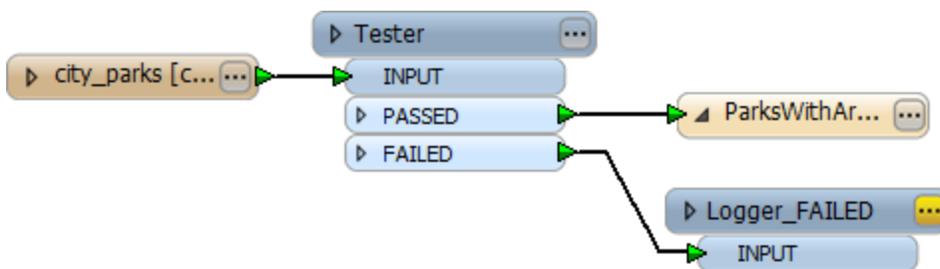
FME doesn't only provide tools for adding transformers efficiently, it also has powerful tools for removing them too. Transformers can be deleted either by selecting the transformer and pressing the <delete> key, or by right-clicking the transformer and choosing Delete.



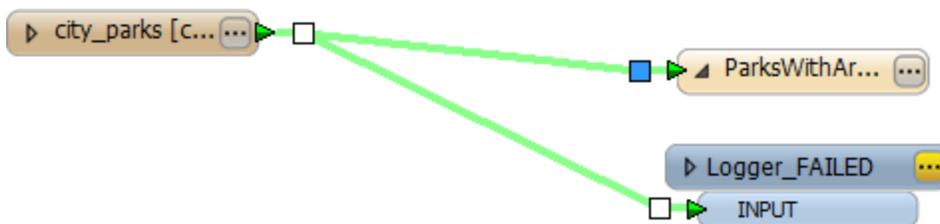
Smart Delete

When a transformer is removed, FME doesn't just leave a gap, but instead tries to reconnect the surrounding objects. This technology is called Smart Delete. When objects are automatically reconnected in this way, the connections are colored green so the user is aware of what has happened. They are also automatically selected.

Here a user wishes to remove the Tester transformer:



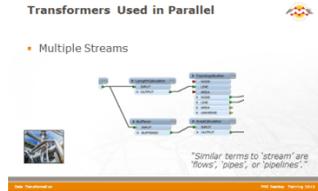
They select the Tester, press the delete key and Smart Delete removes the transformer and reconnects the surrounding objects. If the user does not require the new connections they can simply press the delete key again.





To bypass the Smart Delete function, press the <shift> key while deleting an object.

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.



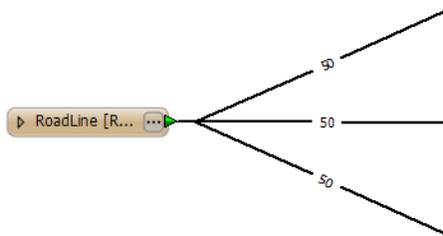
Similar terms to 'stream' are 'flows', 'pipes', or 'pipelines'.

Creating Multiple Streams

Splitting data occurs with any transformer with multiple output ports (example 2 is a good illustration of this point) but can also be achieved by simply making multiple connections out of a single output port.

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.

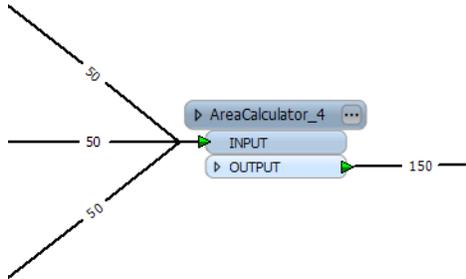


Here 50 road features are being read but, because there are multiple connections from the feature type, multiple copies of the data are being created.

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*.



Example 9: 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:\FMEData\Workspaces\DesktopManual\Example9Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example9Complete.fmw

Now we'll create a label for each park and write it to a new output layer.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 8.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example9Begin.fmw*

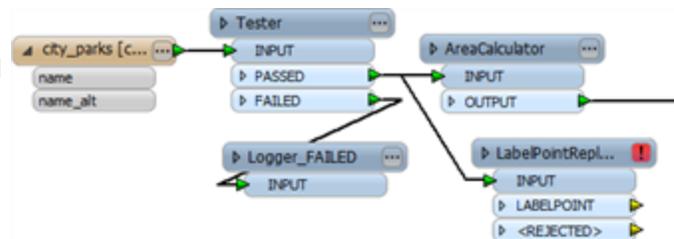
Example 8 measured park areas with the *AreaCalculator*. Now the FME user has been asked to add park names as labels to the output dataset.

This can be achieved using the *LabelPointReplacer* transformer.

2) Place a LabelPointReplacer Transformer

Using the method of your choice, locate and place a *LabelPointReplacer* transformer.

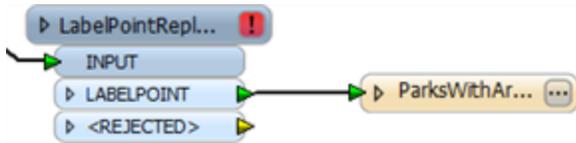
It should appear after the Tester:PASSED port (we don't want to label golf courses that have been filtered out), but on a separate (parallel) stream.



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.



Use Quick Connect to make a new connection from the *LabelPointReplacer* to the new feature type.

4) Check Transformer Settings

Open the settings dialog for the *LabelPointReplacer* transformer.

Choose the attribute **name** to be used for the Label Attribute.

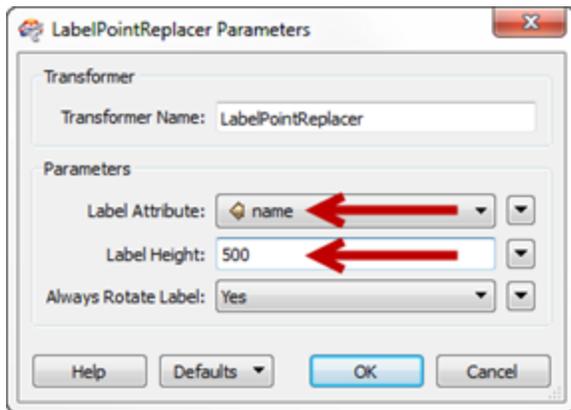
Click in the Label Height field and type 500 (that's 500 working units, which in this case is feet).

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.



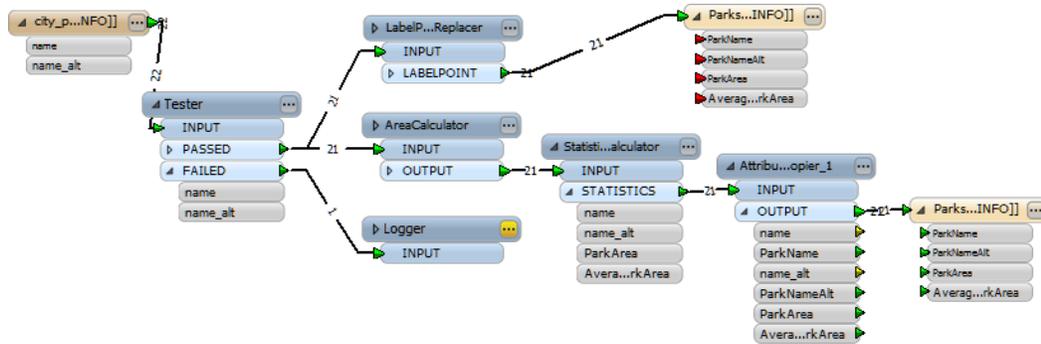
5) Run the Translation

Run the translation and inspect the output.

Notice that the output is in two layers in two files. Use FME Universal Viewer to open both output files in the same view. Or you can choose **Writers > Redirect to Inspection Application** to inspect the output automatically.

6) Save the Workspace

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



Group-based and Feature-based Transformation



FME transformers carry out transformations using one of two basic methods—feature-based or group-based.

As mentioned previously, transformers may operate on either one feature at a time, or on a whole set of features at once.

These two methods are called *Feature-based* and *Group-based* Transformation.

Feature-based Transformation

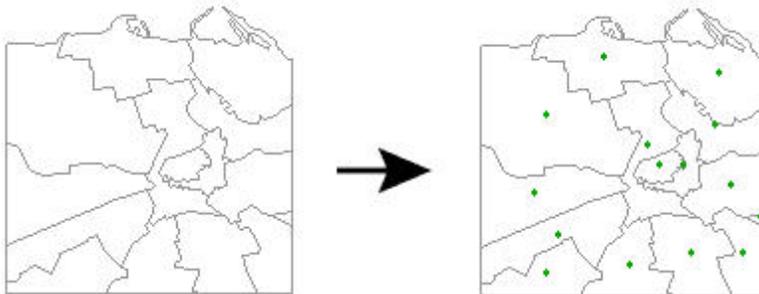
In feature-based transformation (or restructuring) a transformer performs an operation on a feature-by-feature basis.

A single feature at a time is processed and the results of processing one feature have no bearing on the processing of other features.

Generally the same number of features emerges from the transformer as entered it.

Some examples of feature-based restructuring include:

- *Measurements* – length and area calculations are performed on only one feature at a time, and the area of one feature has no impact on the area of another.
- *Centre of Gravity Calculations* – FME can calculate the ‘center of gravity’ (the geographic centre) of an area feature. Each calculation is unique and independent of other features.



Group-based Transformation

In group-based transformation a transformer performs an operation on a group of features, rather than just one.

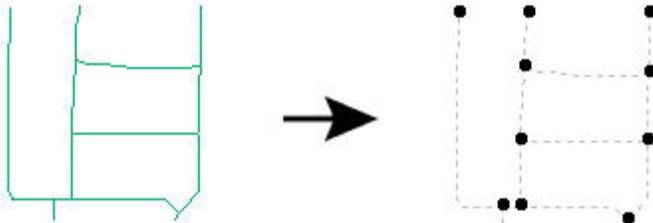
All features are grouped together before processing starts and each feature in the group may have a direct effect on how other features are processed.

With group-based transformations the number of features that emerge from the transformer is more likely (but not certain) to be different from the number of features that entered it.

Some examples of Group-Based restructuring include:

- *Polygon Creation* – FME can create polygon features from a set of line features when they form a closed shape. The line features are grouped together to produce a single area feature.

- *Statistics Calculation* – the *StatisticsCalculator* transformer calculates the mean, maximum, and minimum values of an attribute for a whole group of features.
- *Intersections* – calculating the intersection points of a set of lines falls under group-based restructuring because each input feature in the group affects the output.



By combining the concepts of Feature-based and Group-based Restructuring with Geometric and Attribute Restructuring, a matrix showing examples of each type of process can be created.

	Feature-based	Group-based
Geometric	Generalizer	AreaBuilder
Attribute	AreaCalculator	StatisticsCalculator

For example, anAreaBuilder is a transformer for group-based,geometric restructuring.

Browse through the workbench Transformer Gallery to find your own set of examples.

	Feature-based	Group-based
Geometric		
Attribute		

Group-By Processing

A group-based transformer operates on a whole group of features; but this begs the question how that group is selected.

By default a transformer creates a group from **ALL** features that pass through. So any incoming feature is classed as part of the group.

However, many cases require that the group be made up more selectively. To meet this need many transformers have an option called Group-By.

A Group-By option allows the selection of an attribute held by the data. Before processing the transformer first arranges the features into groups that share this attribute value.



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.

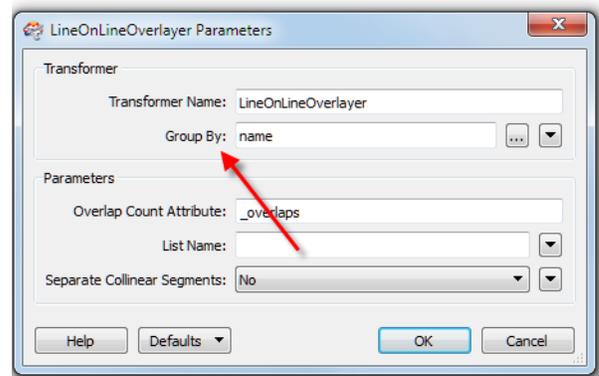
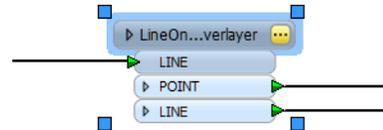
This is the same as a group-by in FME”

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.



Example 10: 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, Group-By Processing
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example10Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example10Complete.fmw

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 9.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example10Begin.fmw*

Example 8 measured park areas with the *AreaCalculator* transformer.

Unfortunately, someone has pointed out that many of the polygons in the data belong to the same park. Big Walnut Creek is a multi-part geometry (or disjoint polygon if you prefer). It ought to be treated as a single park, but at the moment it's treated as several. Let's fix that problem.

2) Place a Transformer – Aggregator

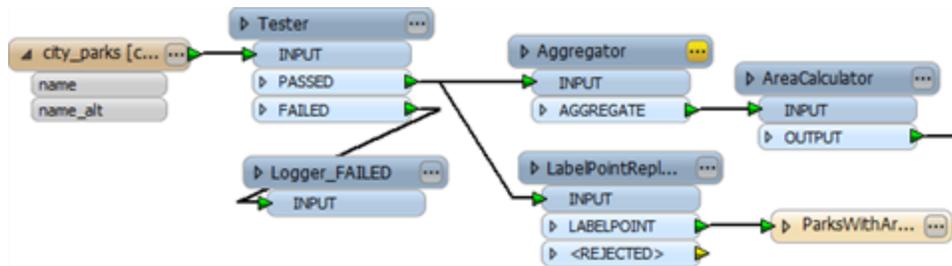
An *Aggregator* transformer will create a collection of geometry.

With this transformer all of the polygons in Big Walnut Creek Park can be collected into one for the purpose of calculating average area.

How will the *Aggregator* know which polygons to aggregate? Simple; a group-by parameter can be used.

Locate and place an *Aggregator* transformer onto the canvas. It should be connected before the *AreaCalculator*; between *Tester:PASSED* and *AreaCalculator:INPUT* would be ideal.

This section of workspace will now look like this:

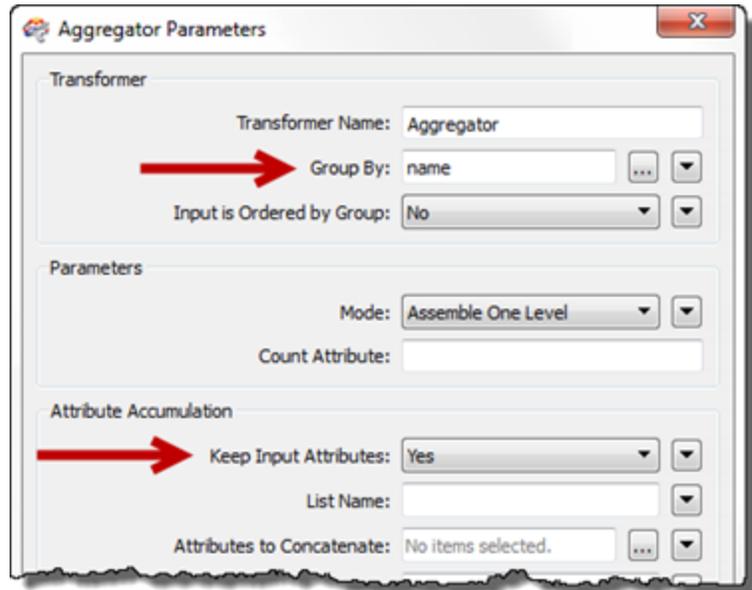


3) Check Transformer Parameters

Open the *Aggregator* Parameters dialog.

Change the Group By setting to group by the attribute *name*.

Set **Keep Input Attributes to Yes**



4) Run the Workspace

Save and then run the workspace.

With a smaller number of parks, but the same overall area, the `AverageParkArea` value should be larger.

The previous value was 1196308ft^2 .

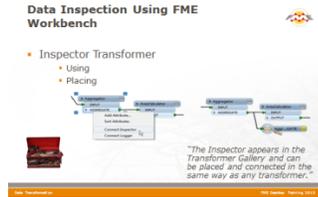
What is it now?

5) Follow-Up Question

OK – here's a question that tests how smart you are!

Why is the *Aggregator* transformer used instead of the group-by option on the *StatisticsCalculator*?

Data Inspection Using FME Workbench



Besides ‘Redirect to Inspection Application’, Workbench can route data to FME Universal Viewer from individual transformers.

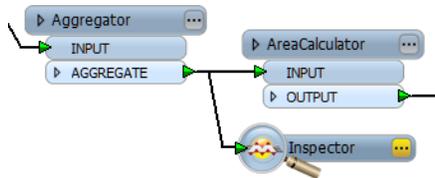
Using an Inspector Transformer

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



Students familiar with older versions of FME will notice that this transformer used to be called the *Visualizer*. It was renamed the *Inspector* in FME2012.

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 away to the *Inspector* after the *Aggregator*, but before the *AreaCalculator*.

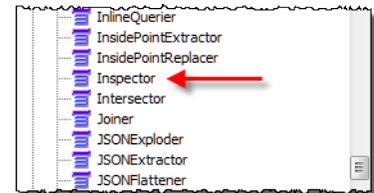
Note: An *Inspector* transformer can even be connected directly to a source Feature Type, in order to view the data immediately after it has been read by FME.

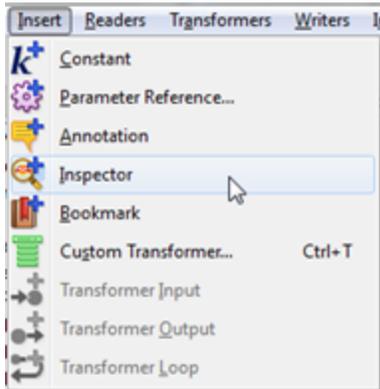
Placing an Inspector Transformer

There are a number of ways to place an *Inspector* transformer.

From the Transformer Gallery

The *Inspector* – like any other transformer – appears in the Transformer Gallery and can be placed and connected in a workspace in the same way.



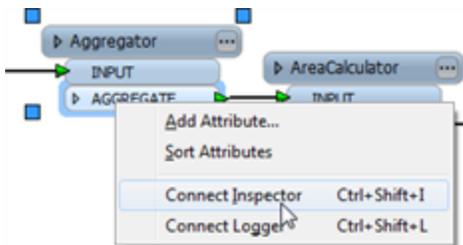


From the Menu bar

An *Inspector* can also be placed using **Insert > Inspector** on the Workbench menu bar.

On the Canvas

Probably 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.



Copying the previous example, the user wants to output data from the *Aggregator* to FME Universal Viewer.

The user right-clicks where it says “AGGREGATE” and chooses the option **Connect Inspector**.

Notice how the *Inspector* is named automatically using the transformer and output port names. This is a big advantage of using this method.

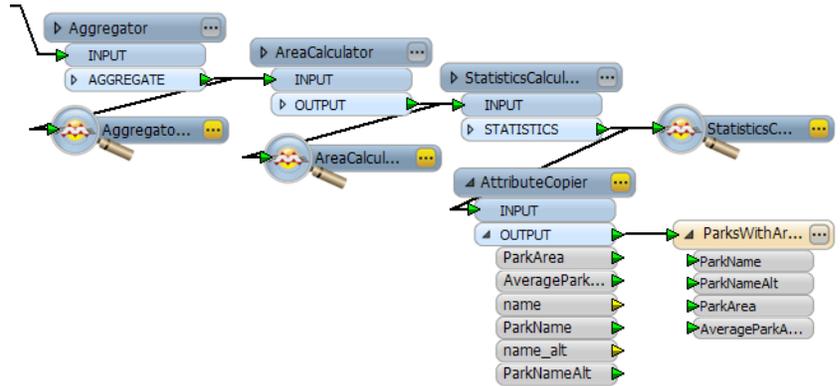


Notice that in the FME Universal Viewer > Display Control window, the feature type is named the same as the *Inspector* within the workspace. This helps the user to identify data when multiple *Inspectors* are applied.



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.”



Note that the Inspector transformer only opens the FME Universal Viewer when there are features to view. If there are zero features, then the Viewer 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

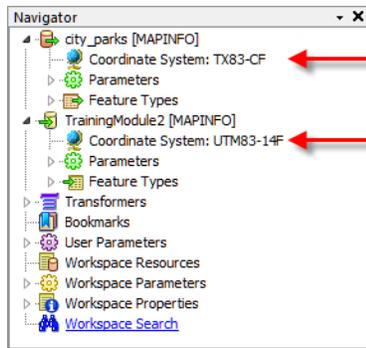
Each feature that is processed by FME is coordinate system aware; that is, it knows what coordinate system it belongs to at all times. This helps prevent confusion when reading multiple datasets that belong to different coordinate systems.



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 TX83-CF and the destination as UTM83-13F.

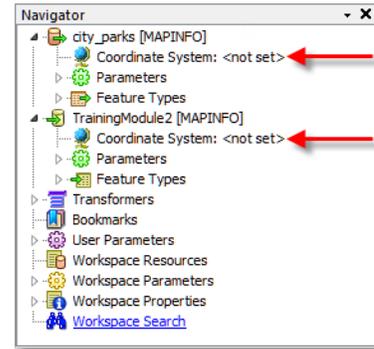
When a translation is carried out where the source and the destination coordinate systems differ in this way, FME automatically restructures the data, at the end of the translation, so that the output is in the correct location.

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.



Example 11: 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:\FMEData\Workspaces\DesktopManual\Example11Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example11Complete.fmw

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 10.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example11Begin.fmw*

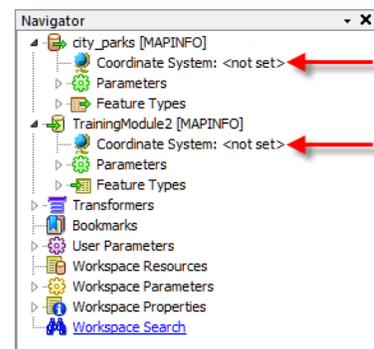
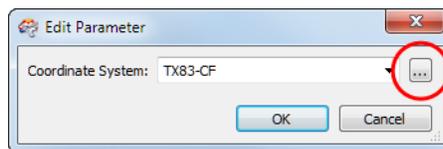
The Grounds Maintenance team has requested that you reproject their data into a UTM coordinate system (Universal Transverse Mercator, Zone 14, Feet).

2) Edit Reader Coordinate System

On the Navigator locate the city_parks 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 TX83-CF or select it from the Coordinate System Gallery using the Browse button.



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 '<not set>'.

Double-click the setting. Enter the coordinate system name UTM83-14F or select it from the coordinate system gallery using the Browse button.

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 `TX83-CF`
```

```
FME Configuration: Destination coordinate system set to `UTM83-14F`
```



5) Inspect the Output

Inspect the output data. Query a feature. The Coordinate System ought to be shown with the new coordinate system.

Note: The coordinate system name is BLM-14N because that's the MapInfo equivalent of UTM83-14F.



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.

In FME2013, there are multiple Reprojection transformers, including a new one called BMGReprojector. The BMGReprojector uses the Blue Marble Reprojection engine to carry out its operations, which some users may prefer to the standard FME Reprojection engine.

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**.
- Processing can be **feature-based** or **group-based**. Groups can be created using the **Group-By** option in a transformer’s settings 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!'



Q&A given on page 2-38:

By combining the concepts of Feature-based and Group-based Restructuring with Geometric/Attribute Restructuring, a matrix showing examples of each type of process can be created...

	Feature-based	Group-based
Geometric	Generalizer	AreaBuilder
Attribute	AreaCalculator	StatisticsCalculator

For example, anAreaBuilder is a transformer for group-based Geometric restructuring.

Browse through the Workbench transformer gallery to find your own set of examples.



	Feature-based	Group-based
Geometric		
Attribute		

Answer – There are many combinations of transformers that you could add to your table. Ask your instructor if you are unsure about a particular transformer. Remember, not all transformers are used for the purpose of restructuring data.

Translation Components

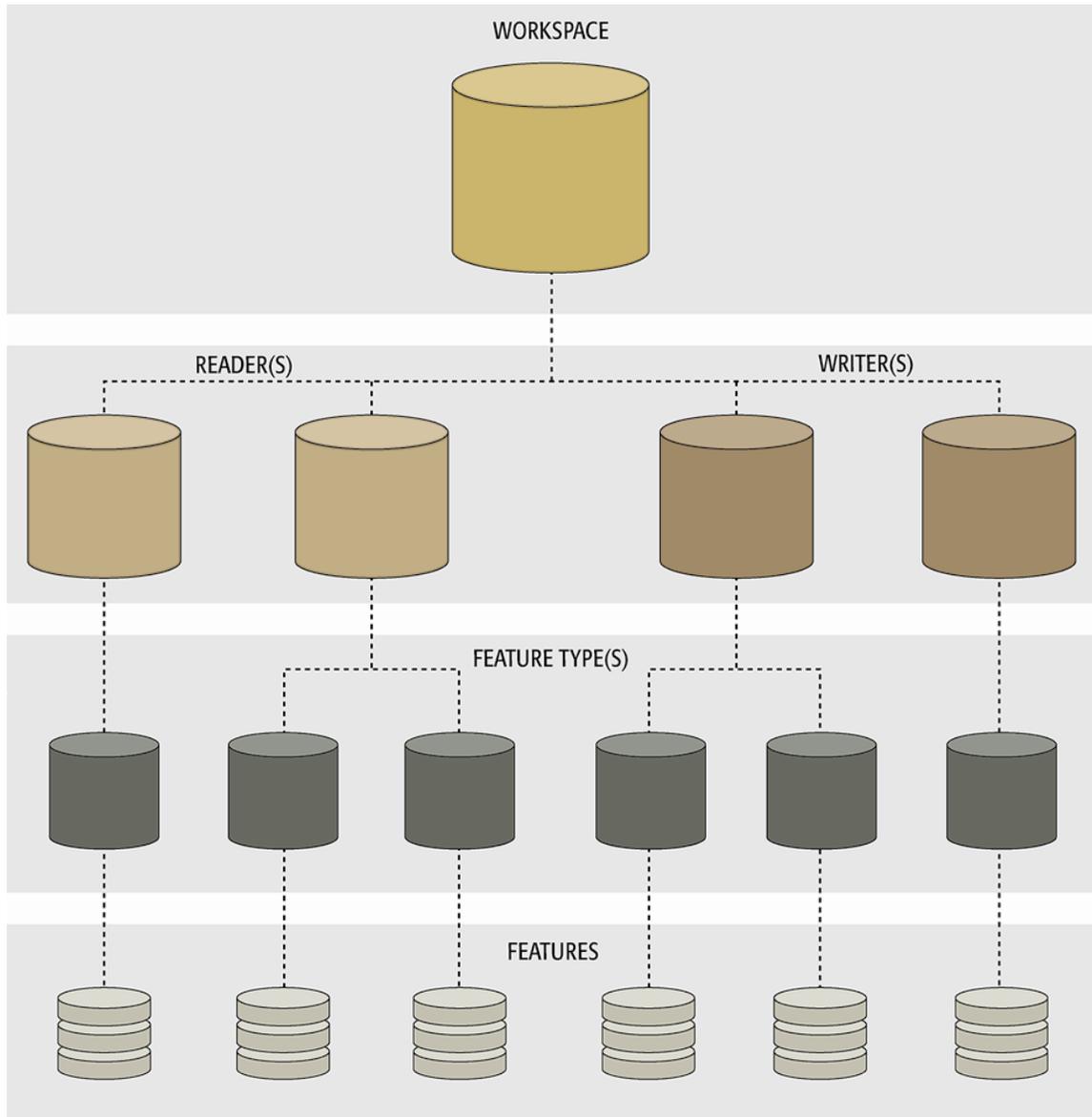


An FME translation is made up of a number of different components.

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.



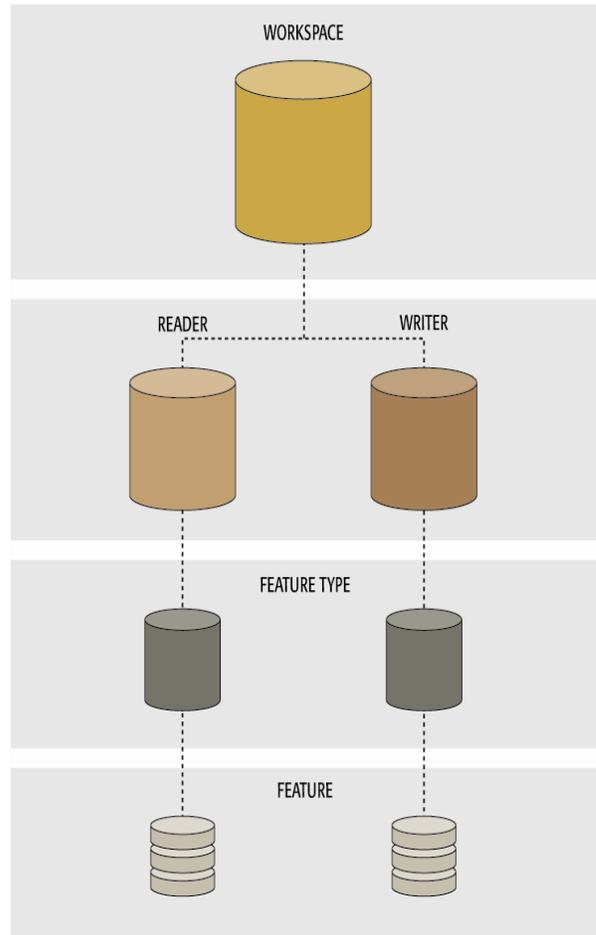
Key Components

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

- Workspace
- Readers and 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 - parameters at one level of the hierarchy can affect components at a different level.



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. Hierarchy is an important concept because it affects how components are added to a translation, and - more importantly - parameters at one level of the hierarchy can affect components at a different level.

Workspace

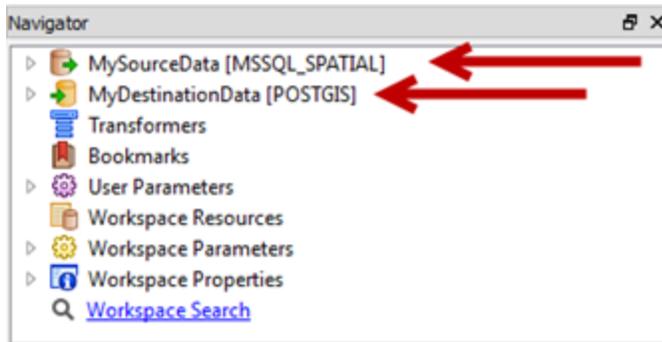
A workspace is the primary element in an FME translation and is responsible for storing a translation definition. A workspace is held as a file with an .fmw file extension. It can be run in either the Quick Translator or FME Workbench, but can only be opened for editing in Workbench.

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.



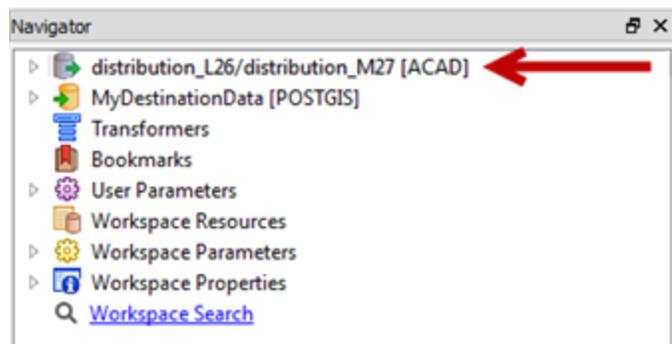
Readers and writers don't appear as objects on the Workbench canvas, but are represented by entries in the Navigator window. Each reader and writer appears as a separate entry in the list.

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 screenshot are the names of the datasets being read/written. When multiple datasets are read they are all listed, like in this AutoCAD reader.



Feature Types

Feature Type is the FME term that describes a subset of records. Common alternatives for this term are 'layer', 'table', 'feature class', and 'object class'.

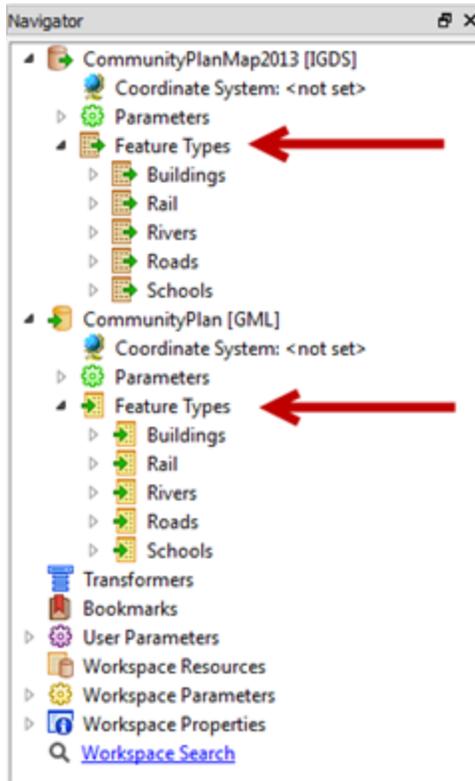
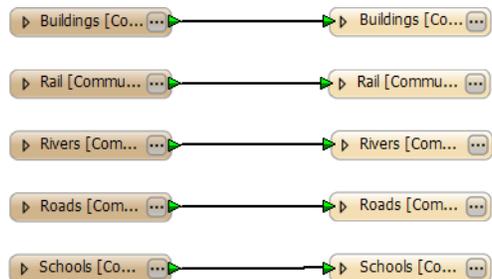
A Feature Type as a translation component is simply a defined layer in the process. If a specific layer is not defined by a feature type, then that data will not be either read and/or written.

Feature Types are represented by objects in the Workbench canvas, so it is easy to see at a glance which layers are represented, and where they are connected to in the translation.



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

In this workspace, feature types represent layers of source data called Buildings, Rail, Rivers, Roads, and Schools. The workspace canvas contains a different object for each feature type in both reader and writer.

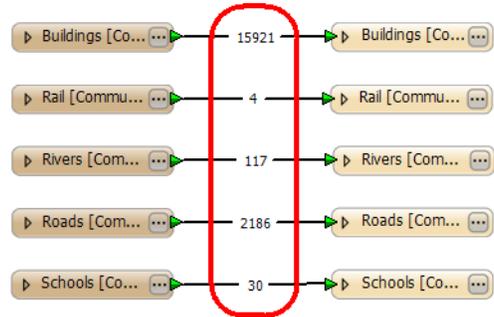


Beside canvas objects, feature types can be found listed in the Navigator window. They are the only component to be listed in two places in this way. A writer feature type icon looks like this:



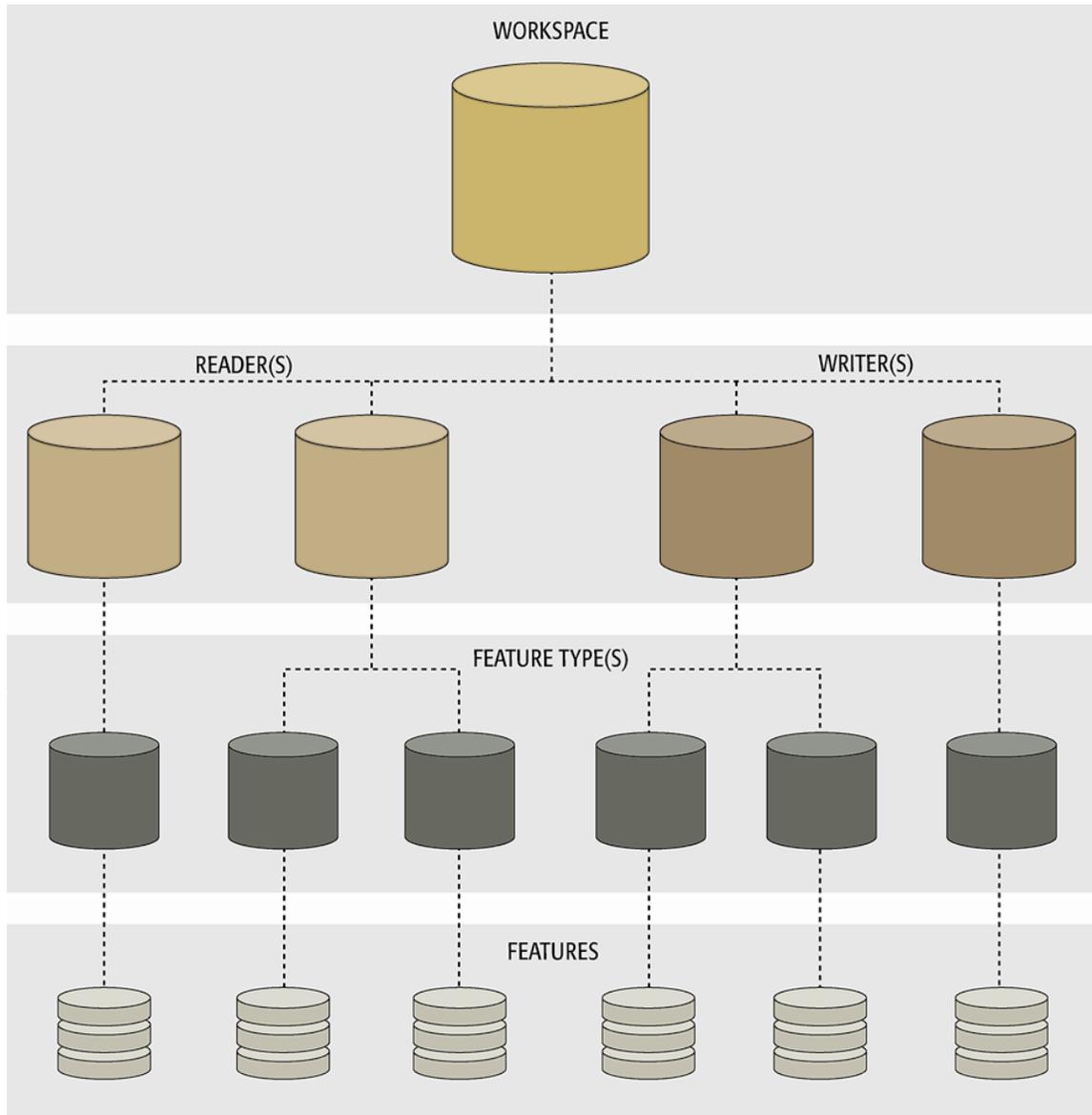
Features

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 2,186 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.

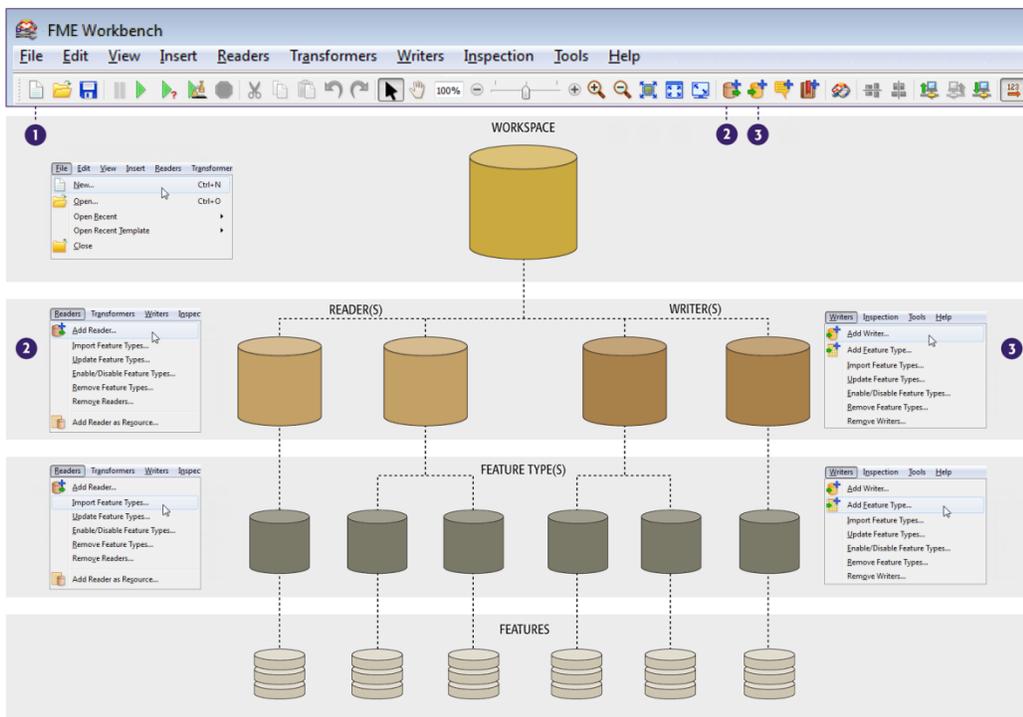
Managing Components



There are a number of tools for creating or adding components to a translation.

Component Management Tools

All of the tools for managing components (creating, adding, or deleting) in a translation can be found on the FME Workbench menubar, as well as in some context-menus.



The list of tools for managing components in an FME translation is as follows:

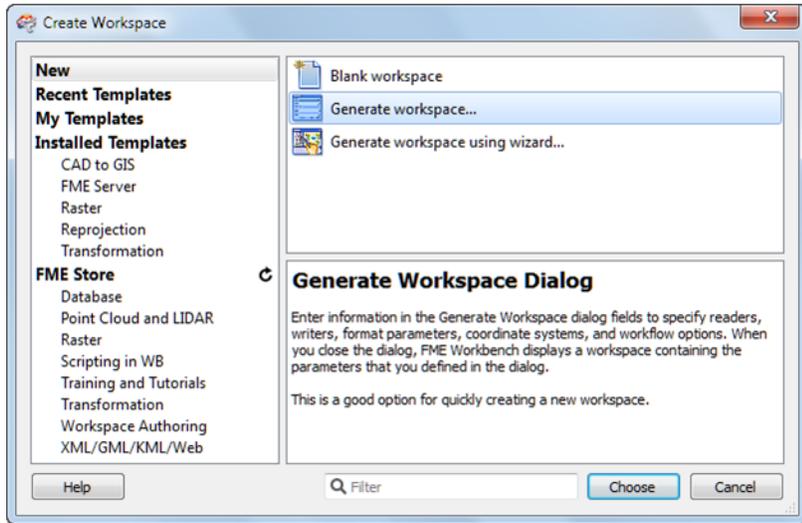
- Create Workspace
- Add Reader/Writer
- Add Feature Types
- Import Feature Types
- Remove Reader/Writer
- Remove Feature Types
- Update Feature Types
- Move Feature Types

Create Workspace

The most common way to start creating a translation definition is to create a workspace through one of the tools in FME Workbench.

File > New on the menubar opens up the Create Workspace dialog and provides a list of methods for creating a new workspace, or even starting out with a template workspace from an online source.

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.

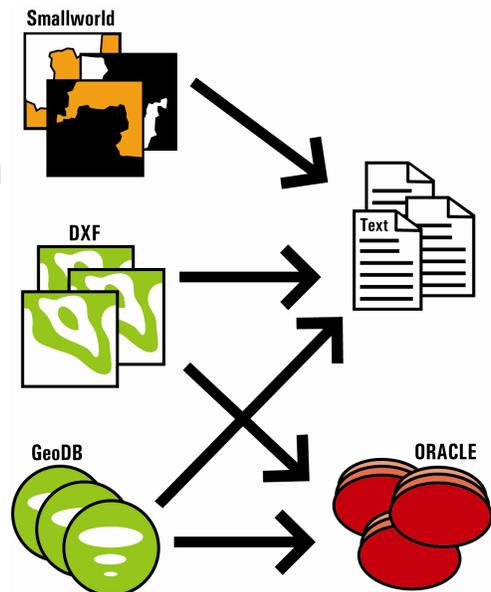


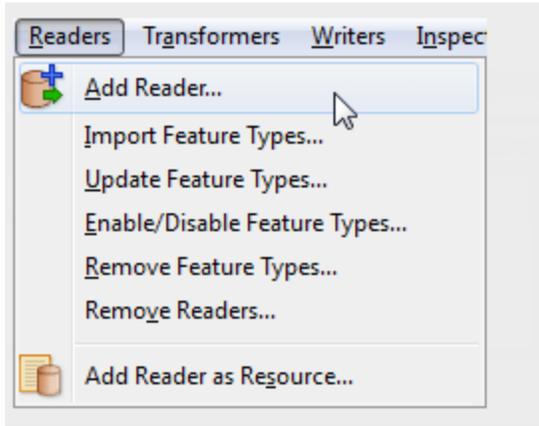
Add Reader/Writer

Adding a reader or writer to a workspace is a common requirement. There are several reasons:

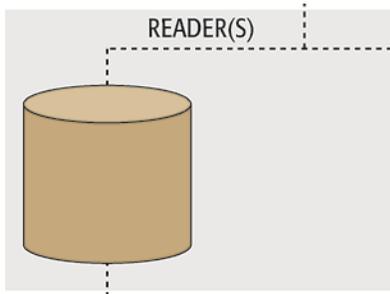
- The Generate Workspace dialog only adds a single Reader and Writer
- Each Reader and Writer handles only one format of data.
- Different datasets (of the same format) may require handling with different parameters

Therefore handling multiple formats of data – such as a workspace that reads Smallworld, DXF, and Geodatabase; and writes to both Oracle and a text file – requires multiple readers/writers.





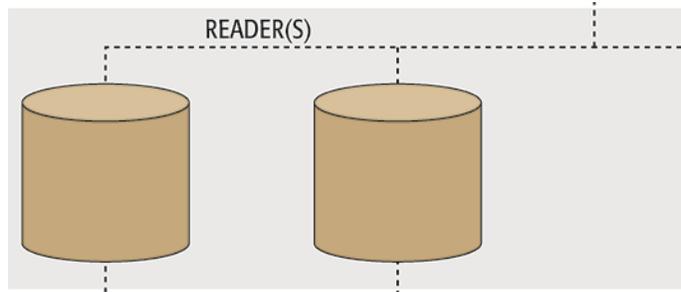
Additional readers are added to a translation using **Readers > Add Reader** from the menubar. Similarly, additional writers are added using **Writers > Add Writer**



Adding a reader has this effect on the hierarchy diagram:



Be aware that adding Readers and Writers can also add Feature Types to the workspace too, as is explained in the next section.



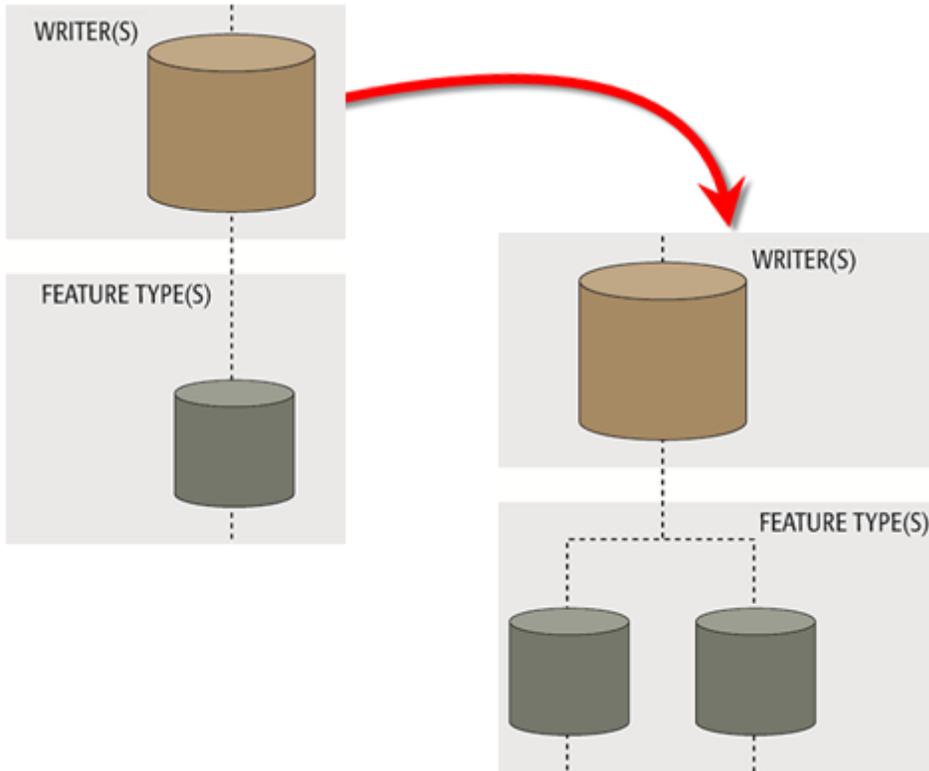
Although the usual workflow is to create a new workspace with the Generate dialog and then add extra components as necessary, there's nothing to prevent a user starting with an empty workspace and simply adding readers and writers one-by-one.

Add Feature Types

Adding feature types can take place either when adding a reader/writer, or by a manual process.

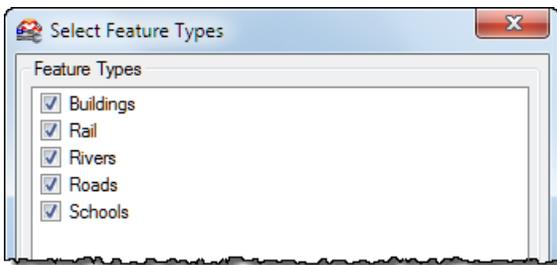
In general, manually adding a feature type is only permitted on the writer side of a translation. That’s because the writer side is “What We Want” and is therefore open to manual editing.

Adding a feature type manually has this effect on the hierarchy diagram.

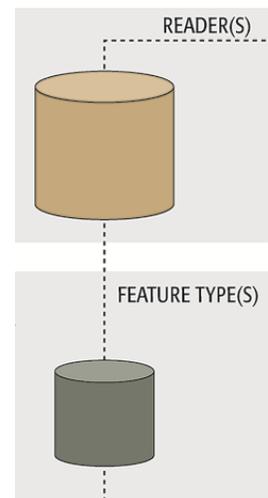


Adding Feature Types with a Reader

When adding a reader, FME will scan the chosen source data and prompt the user as to which source feature types should be added to the workspace.

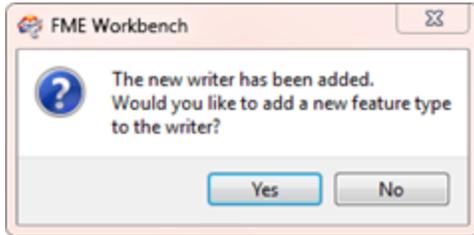


Each chosen type – and not all have to be selected – will be represented by a feature type object below the new Reader.



Adding Feature Types with a Writer

When adding a writer, FME will offer a chance to manually add a new feature type.



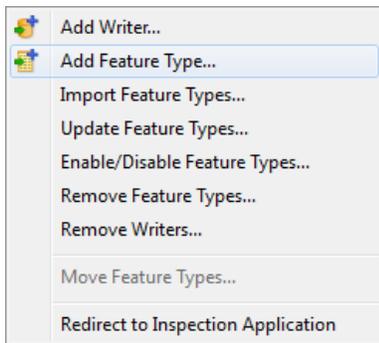
Responding yes to this question opens up the dialog for manually defining a new feature type.

Only one feature type can be added at this time. Additional ones must be added manually.

No is the correct response when feature types are to be copied from a reader (see below) or imported from a different dataset (see next section).

Adding Manually

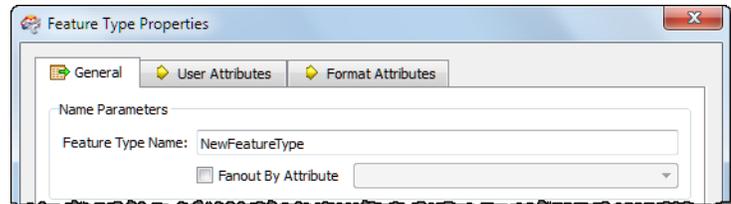
Feature Types can be added manually to a writer using **Writers > Add Feature Type** on the menubar.



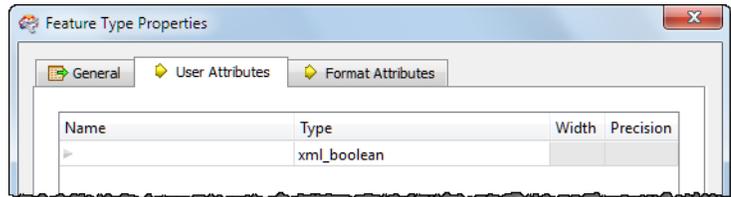
Note: At least one writer must exist in the translation hierarchy; else this option will be greyed out.

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.

The General tab can be used to define the new feature type's name

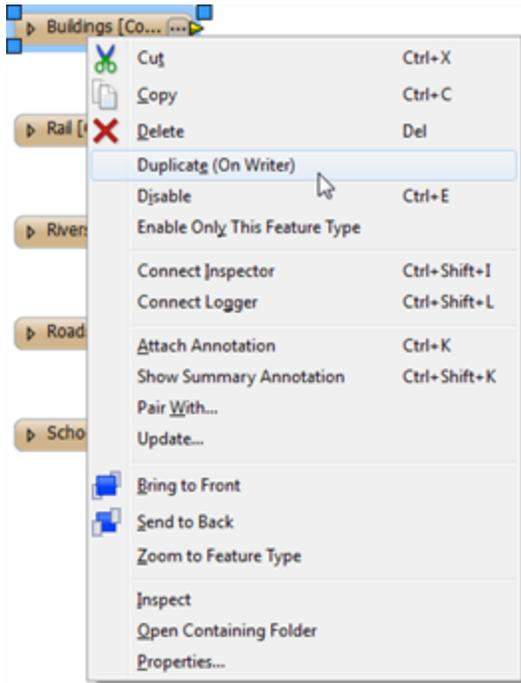


The User Attributes tab can be used to define the new feature type's attribute schema



Copying from a Reader

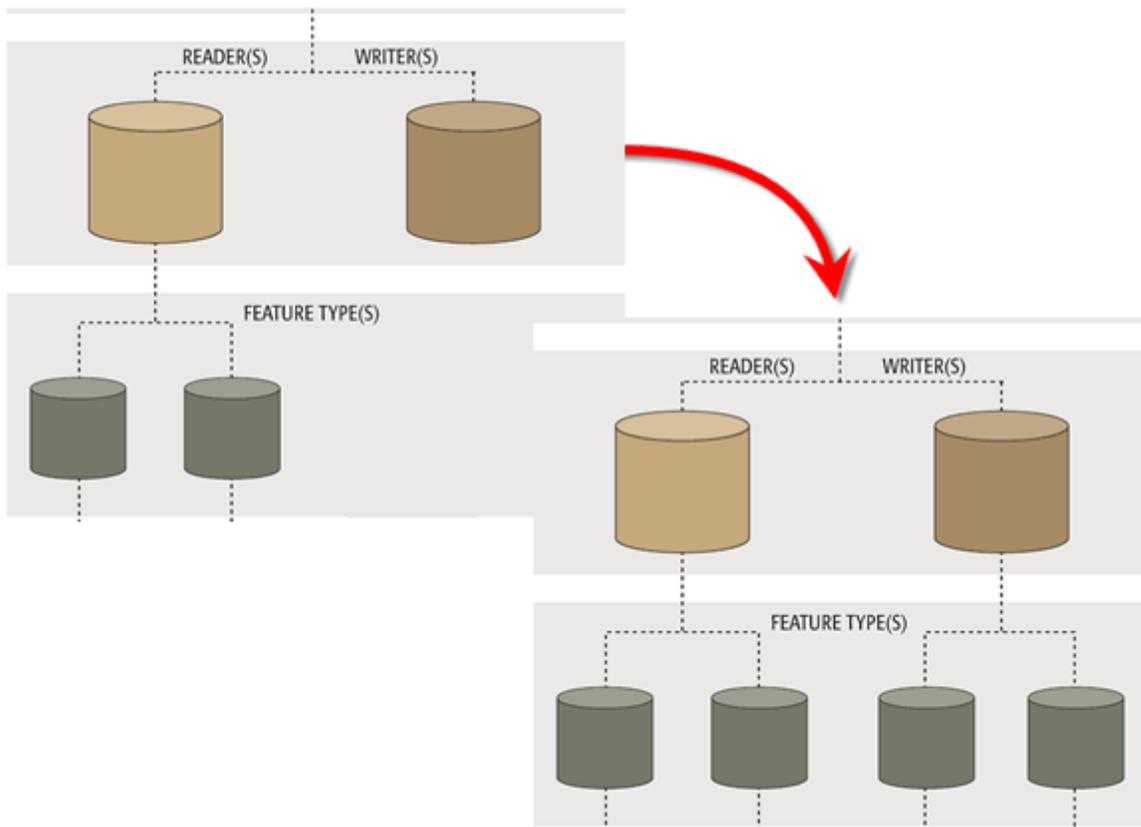
In some cases a user will have a reader with multiple feature types, and wish to add the same ones to a writer. This is simply done by selecting the source feature types, right-clicking them, and using the option **Duplicate (on Writer)**.



The command causes duplicates of the 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.

In the hierarchy diagram, copying feature types looks like this:





Example 2: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads (MicroStation Design v8, Esri File Geodatabase)
Overall Goal	Add CAD data from the engineering department to a public GIS dataset
Demonstrates	Adding readers, adding writers
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example12Complete.fmw

Since Geodatabase format can be treated like a database, the translation can simply add data to it, rather than overwriting the whole dataset. That simplifies the workspace because there's no need to read the Geodatabase every time just to add it back to a new version.



There are two readers/writers for Esri File Geodatabase format datasets. One requires ArcGIS to be installed and has greater functionality; the other offers less functionality but operates without any application software requirements.

1) Start Workbench

Start Workbench and click on the Start tab if necessary.
Choose the option to start with a blank workspace.

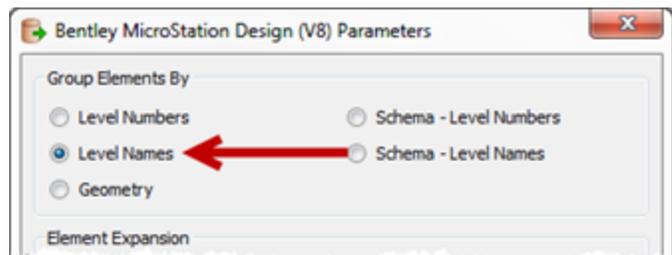


2) Add a Reader

Now start adding components by selecting **Readers > Add Reader** from the menubar. Fill in the Add Reader dialog as follows:

Reader Format Bentley MicroStation Design (V8)
Reader Dataset C:\FMEData\Data\Roads\MajorRoads.dgn

Parameters'Group Elements By' is set to 'Level Names'



3) Choose Feature Types

When prompted, select both Roads and Labels as the feature types to add to the workspace.



4) Add a Writer

Now add a writer by selecting **Writers > Add Writer** from the menubar.

If you do have ArcGIS installed and licensed, then fill in the Add Writer dialog as follows:

Writer Format Esri Geodatabase (File Geodatabase ArcObjects)
Writer Dataset C:\FMEData\Data\Transit\Transit.gdb

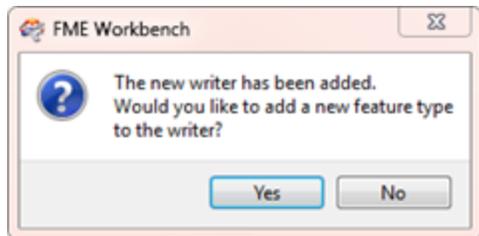
If you do not have ArcGIS installed, then fill in the dialog like this:

Writer Format Esri Geodatabase (File Geodatabase API)
Writer Dataset C:\FMEData\Data\Transit\Transit.gdb

You should find that the transit dataset already exists, and so be sure to select the file rather than entering a name manually.

When prompted to add a new feature type to the writer, click **No**.

As shown in the next section, there are better ways to create writer feature types than to manually add them.



5) Copy Feature Type

Now a new feature type can be added to store the roads data.

Right-click the Roads feature type and choose **Duplicate (on Writer)**.

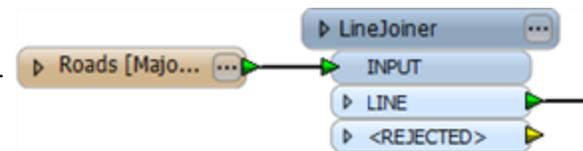
The Roads feature type will be duplicated on the writer and automatically connected.

6) Clean Up Linework

When you inspected the source data before starting this example (did you inspect it?) you may have noticed that all road features are split up at intersection points.

The specification for the destination data requires that these are joined into single features.

Connect a *LineJoiner* transformer between the reader and writer feature types. The Insert Transformer dialog will appear. Click **OK** to accept the line-to-line port selections as given.



Open the properties dialog and accept the default transformer parameters, which are sufficient for this example.

7) Check Feature Type Parameters

Open the properties dialog for the Geodatabase Roads feature type.

Ensure the Allowed Geometries parameter is set to geodb_polyline

In the user attributes tab, remove any attributes (such as igds_class, igds_color) that have been copied from MicroStation format attributes.

8) Save the Workspace

Save the workspace, but **DO NOT** run the workspace yet!

Import Feature Types

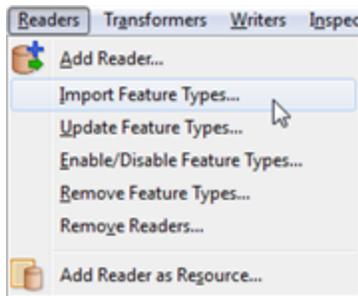
To understand importing feature types, it's important to recognize that "schema" is a separate entity, capable of being used and copied independently of any actions on the data itself.

What the import tool does is essentially take a dataset's schema, and add it to a workspace.

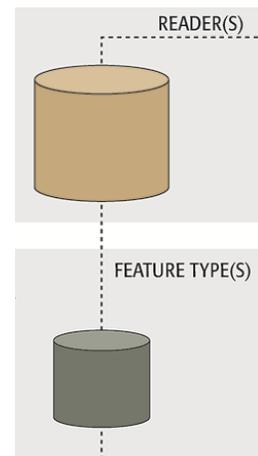
For example, a user has a workspace reading from a spatial database.

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...**

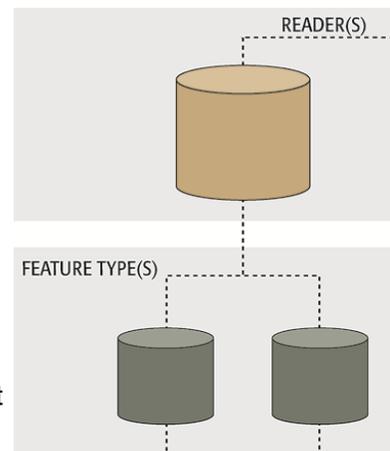


The import tool will take the schema definition of the database table 'rail' and add it to the workspace.

This is particularly important for a reader, because there is no "Add Feature Type" tool for a reader; the reason being that a source schema represents "What We Have" and adding user-defined definitions doesn't reflect that reality.

Interestingly, the import tool can import schemas from a dataset without that dataset being part of the actual translation. Even a different format is no impediment to using a schema this way.

For example, a user may wish to store table definitions in XML, importing feature types from the XML schema for use in writing to another format. A Spatial Data Infrastructure (SDI) with a rigidly defined schema, but open format specification, might be one use of this scenario.

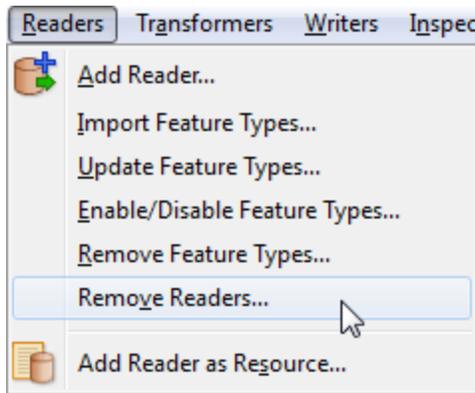




It's always preferable to import feature types, or copy them from an existing reader, rather than manually add them. That's because a manual process is slower and more prone to user error; especially when case sensitivity is an issue.

Remove Reader/Writer

Tools exist to remove a reader or writer from a workspace, both on the menubar and in context menus in the Navigator window.



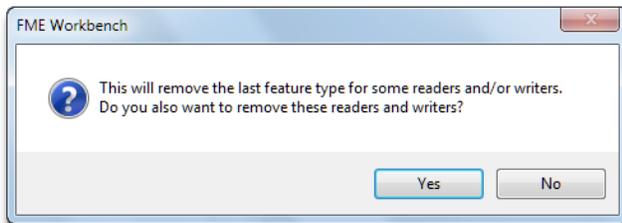
Note: Whenever a reader or writer is removed, then all the related feature types will also be removed.

Remove Feature Types

This remove feature types function works at a lower level of the hierarchy than reader/writer removal, and just removes one or more feature types from the translation.

There is a menubar tool, but the easier method is to select the feature type(s) in the canvas and simply press the delete key on the keyboard.

Whenever all feature types are deleted from a reader or writer then FME will prompt the user to decide whether to remove the reader/writer as well.



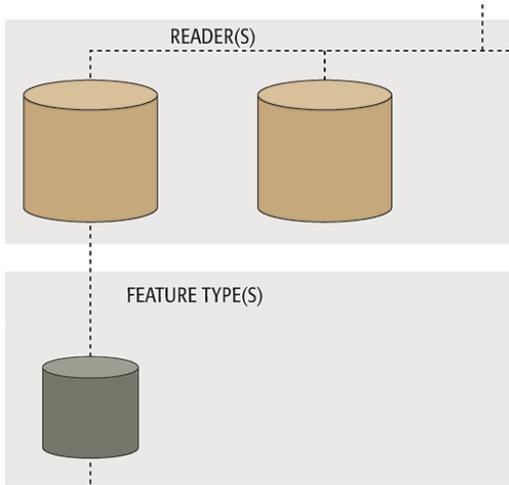
If the feature types are all removed, and yet the reader or writer is left in the translation, then the hierarchy diagram has a “dangling” reader/writer.



A dangling reader/writer isn't a problem provided it's only a temporary situation; i.e. the user intends to now import or add new feature types.

The workspace should not be run in this condition!

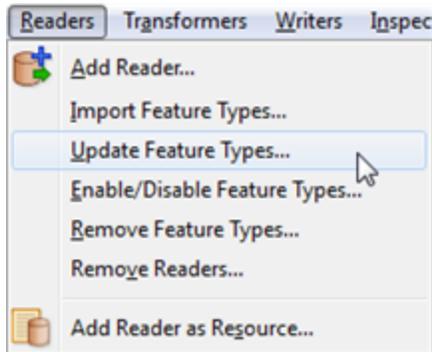
Performance suffers because all the source data is still being read. With a dangling reader it is discarded immediately, but with a dangling writer it is read and transformed too.



Update Feature Types

A frequent problem for users of spatial data occurs when – after setting up a data processing task – the structure of the source data changes; for example, a new attribute is added or the type of an existing attribute is changed from an integer to a floating point.

Because feature type definitions in Workbench are prone to the same problem, FME provides a way to update a workspace based on a new data structure.



Readers > Update Feature Types and **Writers > Update Feature Types** are the tools available to do this.

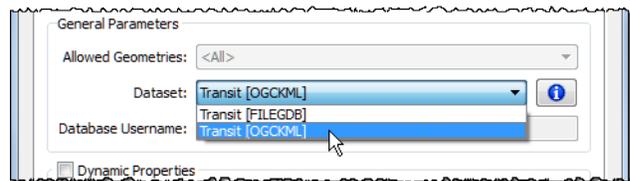
Both are a little like the Import Feature Type tool, except that the external schema is used to update a translation component of the same name, rather than to add a new one.

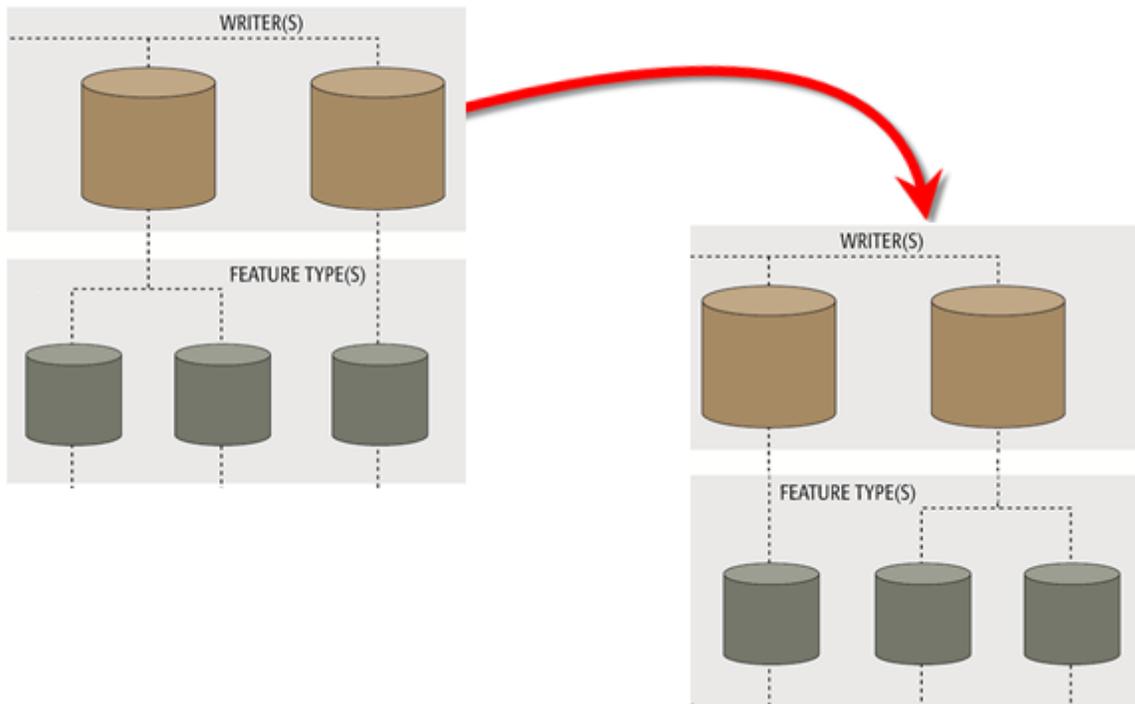
Move Feature Types

As previously noted, a schema can be considered an independent object capable of being used for any reader/writer. This means there is no reason why feature types can't be moved from one writer to another, as and when required.

Whenever there are two or more writers, the Dataset setting in the writer Feature Type properties becomes active.

By changing this, the feature type is moved from one writer to another.





Moving a feature type is not a common procedure, but it's a very useful time-saver when necessary.



Example 13: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Airport (CITS Data Transfer Format (QLF), Esri File Geodatabase)
Overall Goal	Add CAD data from the engineering department to a public GIS dataset
Demonstrates	Adding readers, importing feature types
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example13Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example13Complete.fmw

The next step in this CAD to GIS task is to convert some QLF data representing an airport, to the Esri File Geodatabase. This time, because a table already exists in the output dataset, Import Feature Type will be used.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from the previous example.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example13Begin.fmw*

2) Add a Reader

The Airport data will come from a dataset in a different format to the Roads, so an additional reader is required.

Select **Readers > Add Reader** from the menubar. Fill in the Add Reader dialog as follows:

Reader Format CITS Data Transfer Format (QLF)
Reader Dataset C:\FMEData\Data\Airport\airport.qlf

A new reader and feature type is added to the workspace.

3) Import a Writer Feature Type

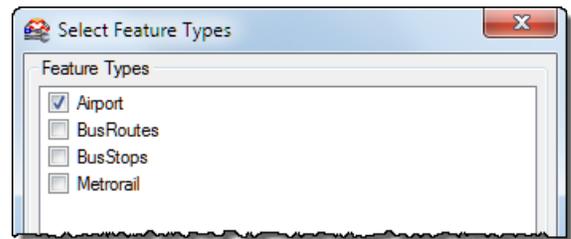
Select **Writers > Import Feature Type** from the menubar.

The Import Feature Types Dialog should be filled in automatically with the existing reader definition. If not, use the following parameters:

WriterFormat Esri Geodatabase (File Geodatabase API)
 or Esri Geodatabase (File Geodatabase ArcObjects)

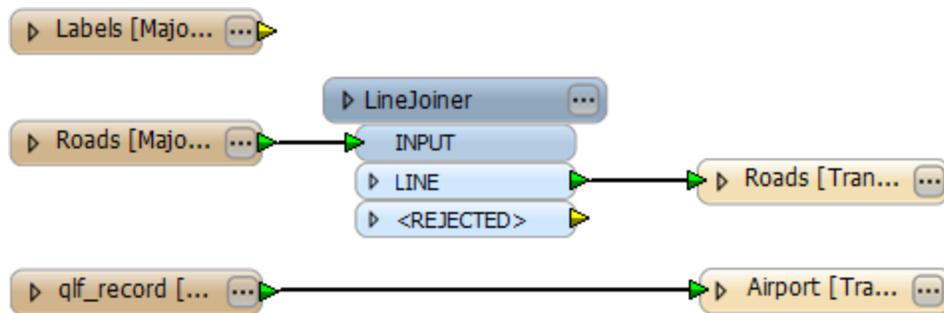
Writer Dataset C:\FMEData\Data\Transit\Transit.gdb

Since the only table required is Airport, when prompted uncheck all of the feature types except for Airport.



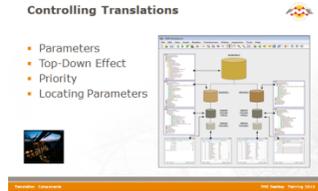
4) Map Schema

Map the reader feature type for Airport (it should be named *qlf_record*) to the writer feature type for Airport.



Save the workspace, but again, **DO NOT** run it yet!

Controlling Translations



Successfully translating from one format to another requires a firm grasp on all the parameters that control the translation.

Parameters are what control a translation.



Chef Bimm says...

"Parameters in a translation are like the options when you order a coffee.

You get to choose what ingredients you start with, how they are combined, and what the end result of the process will be like.

And I think both coffee and data are better when you add a whipped cream topping!"

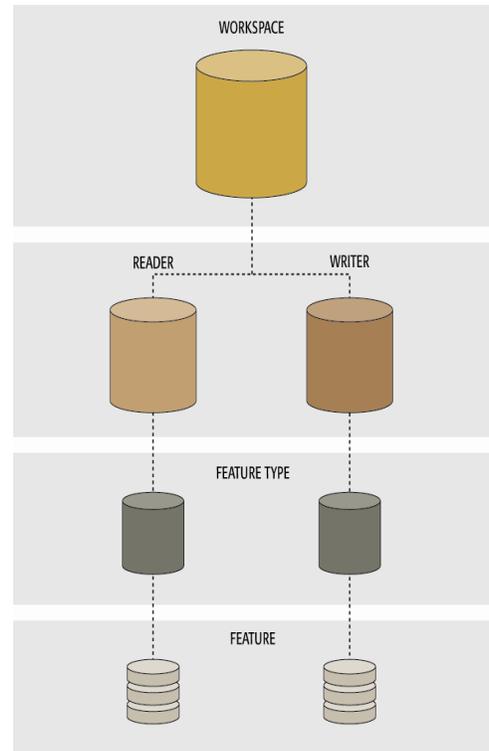


In the hierarchy of different translation components, each different level of the hierarchy has a set of parameters that belong to it.

So there are:

- Workspace Parameters
- Reader Parameters
- Writer Parameters
- Feature Type Parameters
- Format Attributes (Feature Parameters)

Having parameters right down to the feature level provides a huge degree of control over every aspect of a translation.



Top-Down Effect

The basic rule is that any higher-level parameter affects every component below it.

For example, a Workspace Parameter affects all Readers and Writers, all Feature Types that belong to those Readers/Writers, and all features that belong to the Feature Types.

A Reader Parameter affects all Feature Types that belong to that particular reader, but not Feature Types belonging to another Reader.



*To carry on Chef Bimm’s analogy, if you load a coffee machine with Decaffeinated coffee, then all of the drinks will be decaffeinated (Workspace Parameter).
But, each drink can still separately include cream and sugar (Feature Type Parameter)*

Priority

Priority is important because, in some cases, the same parameter exists at different levels.

Perhaps the best example of this is database writing mode.

Database writing mode (Insert, Update, or Delete) can be set firstly at the Writer level, in which case it applies to all tables and features. For example, if the writer level is set to INSERT then ALL features are written to tables as an insert.

But database writing mode can also be set at the Feature Type level, in which case it applies only to features written to that table. This allows different tables to have different modes.

Finally, database writing mode can be set on individual features. Different features can be used to insert, update, or delete records – simultaneously – in the same table.

Interestingly, the higher-up parameter only applies when the lower-down parameters are not set. When the same parameter is set at different levels, then the lower-level parameter wins out.

For example, a Writer might be set as INSERT mode; but a table is set to UPDATE mode. In that case the feature type level parameter wins out, and features are written to that table as an update.



*Again it may help to return to the coffee maker analogy.
The coffee machine may have a temperature option to set the temperature of drinks. This parameter (being at the top level) will apply to **all** drinks.*

However, there may also be a temperature override button as a drink is prepared.

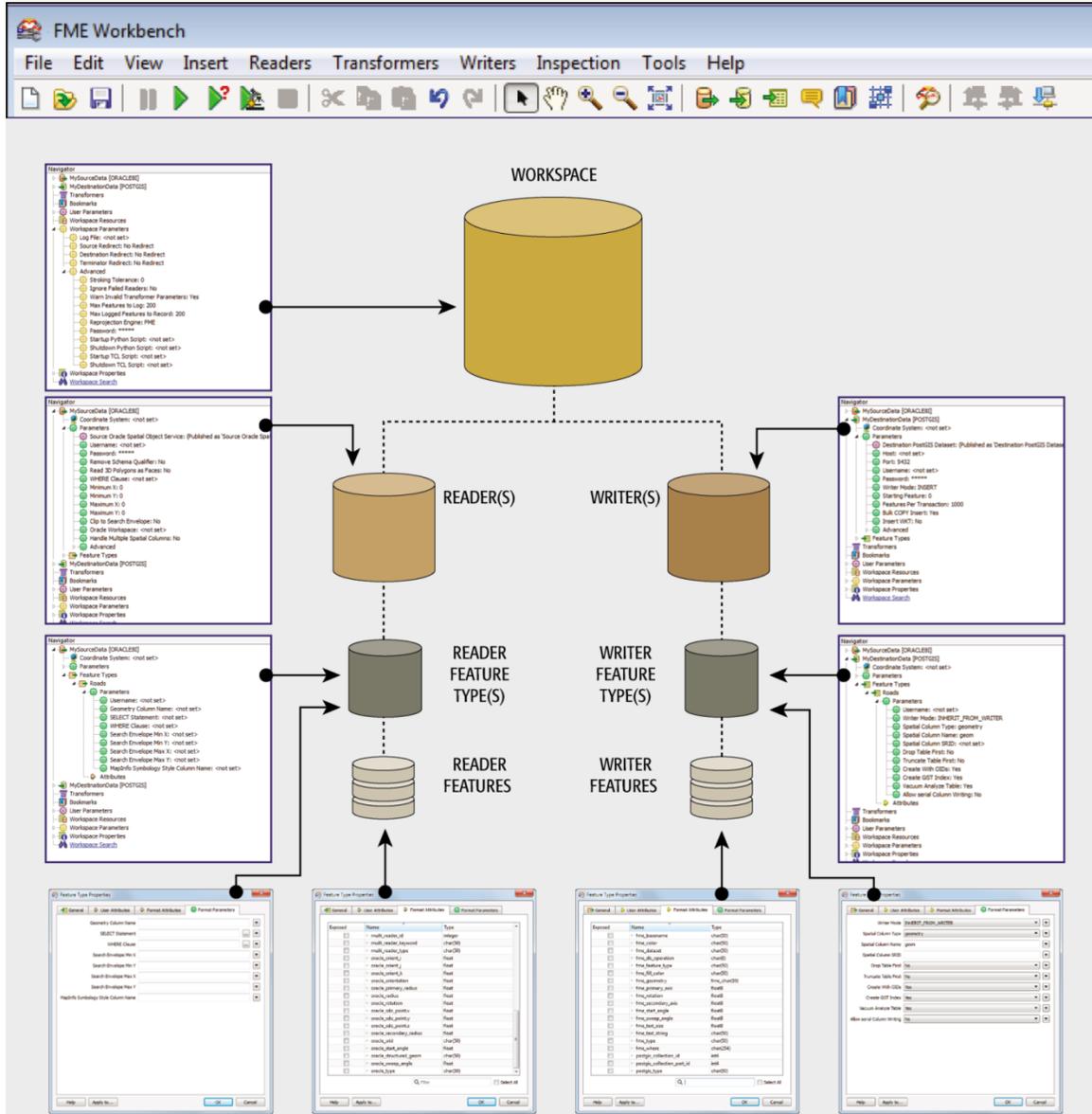
Therefore, although the same temperature option occurs at each level, the lower level parameter takes priority (i.e. it overrides the higher-up parameter).

Locating Parameters

This diagram shows where the parameters are located for each translation component.

Feature Types are interesting because their parameters are found in both the Navigator Window and the Feature

Type Properties dialogs.



Parameter

Location

- Workspace Parameters Navigator Window
- Reader and Writer Parameters Navigator Window
- Feature Type Parameters Navigator Window *and* Feature Type Properties dialog
- Format Attributes Feature Type Properties dialog

As will be shown, although parameters to control features are *exposed* in the Feature Type Properties dialog, they are usually *set* using a transformer.

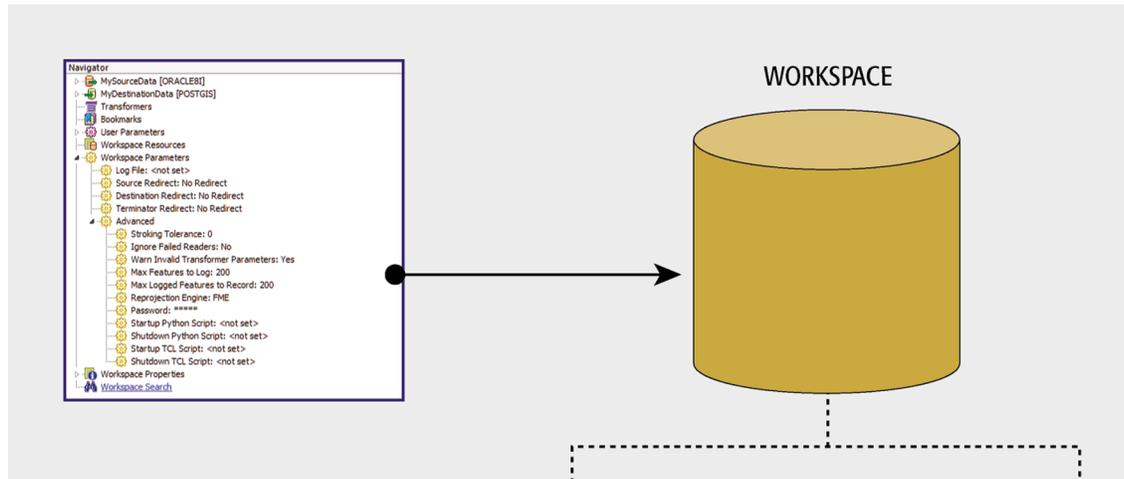
Workspace Parameters



Workspace parameters relate to the workspace as a whole.

Workspace parameters (settings) are all of the parameters that relate to a workspace as a whole. They apply to the current workspace only and may change between workspaces.

Workspace parameters are shown and set in the Navigator Window.



The Workspace Parameters section contains settings that have an effect on how the translation is performed. Settings that provide information about a workspace such as Workspace Name and Workspace Description, but have no effect on the translation, are found in the Workspace Properties section.

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



Don't confuse Workspace Parameters with another set of Navigator items called Workspace Properties.

Workspace Properties are a set of metadata fields, including Workspace Name and Workspace Description, and have no direct effect on how a translation is carried out.

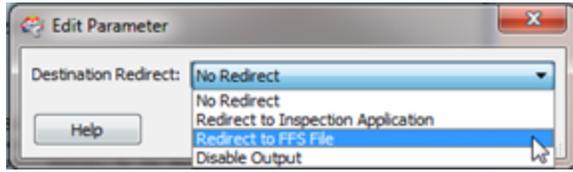
See the session on Best Practice for more information on Workspace Properties.

Basic Workspace Parameters

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

Destination Redirect

The Destination 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 destination 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.

Advanced Workspace Parameters

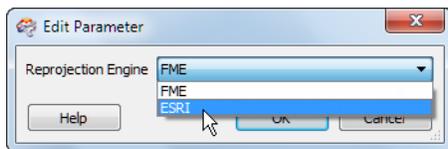
The advanced workspace parameters are perhaps not as valuable in everyday use, but have great importance in specific scenarios. Some particularly important ones are:

Ignore Failed Readers

This YES/NO parameter tells FME whether to continue a translation when reading a dataset fails. For example, if the wrong password is entered so that FME cannot read from a database, should the translation continue with any other datasets that FME was able to read from?

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.

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.

Check for Missing Attribute References

This parameter determines what happens when an attribute used to supply a value is deleted or otherwise removed, so that it becomes unavailable. In most cases the workspace user will wish to be alerted to this problem, so the default for this parameter is “Yes”.

However, in some cases – for example as can occur in an older workspace – the attribute is not really “missing” but just invisible to FME. In this scenario, where the author is certain that such attributes do exist, then he/she may set this parameter to “No” to avoid unwanted error messages.

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

Reader and Writer Parameters



Each reader or writer in a workspace is controlled by a separate set of parameters in Workbench.

Reader and Writer parameters are those that control how data is read and 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.

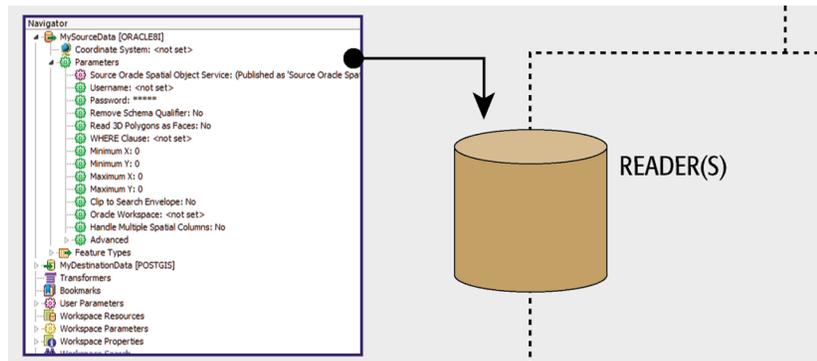
Also, because different actions may be required, even the reader and writer of a single format will have dissimilar sets of parameters.

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

Reader Parameters

Reader parameters are shown and set in the Navigator Window.

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



Doctor Workbench says...

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

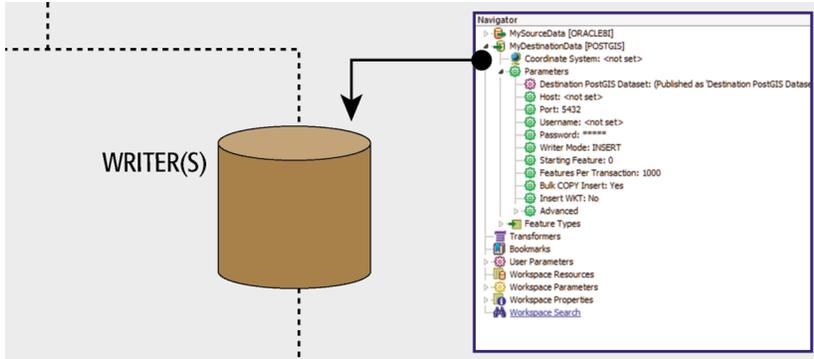
It's like preparing a patient for surgery. Once the workspace (patient) is created (prepped) those parameters aren't available because you're past the point where they would have any effect.

Of course, sometimes you get such a parameter wrong, in which case you simply recreate the workspace. Or find yourself a new patient!

Writer Parameters

Writer parameters are shown and set in the Navigator Window.

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



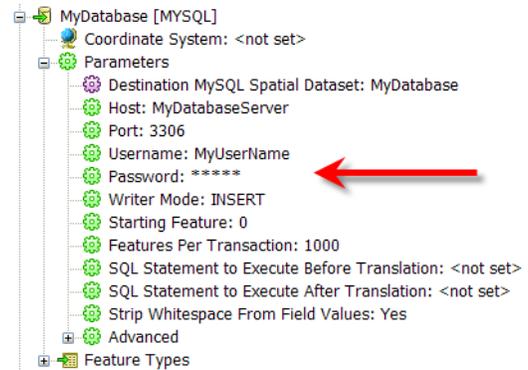
Parameter Priority

It's worth emphasizing that, because Readers and Writers are at a relatively high level in the translation hierarchy, their parameters apply to everything beneath them; that is, ALL feature types and ALL features.

Database Password is a good example of a Reader/Writer-level parameter.

A database user password is something that applies to ALL tables being read.

There isn't a different password per table! Therefore it is a Reader/Writer level parameter.



Example 14: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	AirportCITS Data Transfer Format (QLF), Esri File Geodatabase
Overall Goal	Add CAD data from the engineering department to a GIS dataset
Demonstrates	Workspace Parameters, Reader/Writer Parameters
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example14Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example14Complete.fmw

The next step in this CAD to GIS task is to run the workspace.

Because it's the first run, a user might not be sure the translation will succeed. If this is the case, you don't want to overwrite or append data to the existing database.

What needs to happen is to run the translation without actually writing any data.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from the previous example.

Alternatively you can open `C:\FMEData\Workspaces\DesktopManual\Example14Begin.fmw`

In the Navigator window, check the workspace parameter 'Destination Redirect'.

At the moment it will be set to 'No Redirect'.

Changing this setting to 'Redirect to Inspection Application' is the equivalent to using **Writers > Redirect to Inspection Application**. In fact, you can try this to show how changing one automatically affects the other.

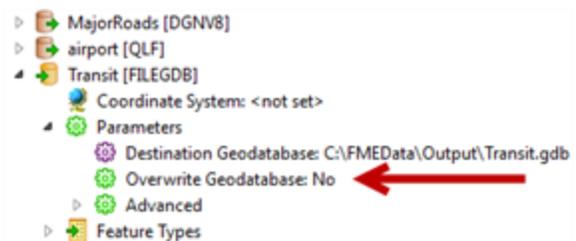
Changing the Destination Redirect setting to 'Disable Output' prevents writing any data to the destination dataset or to FME Universal Viewer. In effect, you're running the workspace up until the writers take effect, testing the reading and transformation steps of the translation.

Experiment with setting the Destination Redirect option to either Redirect to Inspection Application or Disable Output, and running the workspace.

2) Change a Writer Parameter

Once you are happy the translation works, it needs to be run for real. However, because the Geodatabase already contains data that should not get erased, the writer needs to be used in an append mode.

In the Navigator window locate the writer parameter **Overwrite Geodatabase**. Make sure it's set to **No**.

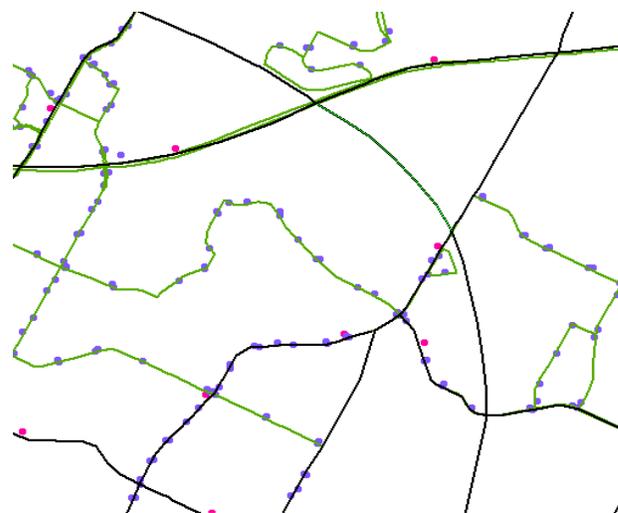


3) Run the Workspace

Turn off the Destination Redirect and run the workspace.

Only run it once else there will be multiple copies of the Road data.

Inspect the Geodatabase contents to make sure the original data still exists and that the roads have been successfully added to it.



Notice how the roads are broken at intersection points, so that one road (for example, US HWY 290E) is made up of many sections. This is something that needs adjusting in upcoming examples.

Feature Type Parameters



Each Reader or Writer Feature Type (layer) is also controlled by a number of settings and parameters.

Feature Type Parameters

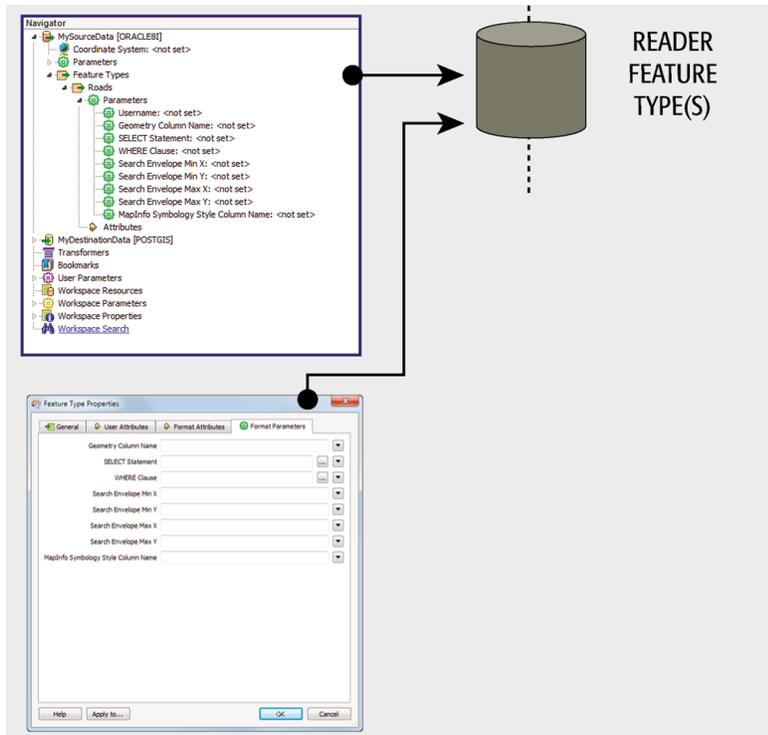
Feature Types are at a lower level in the hierarchy than readers and writers.

Therefore, Feature Type parameters don't apply to datasets as a whole, but only to individual feature types within a dataset. They provide a degree of individual control over reading and writing different layers or tables.

Reader Feature Type Parameters

Reader feature type parameters apply to *reading* of specific layers/tables.

A general rule is that database formats have reader feature type parameters, but few file-based formats do.



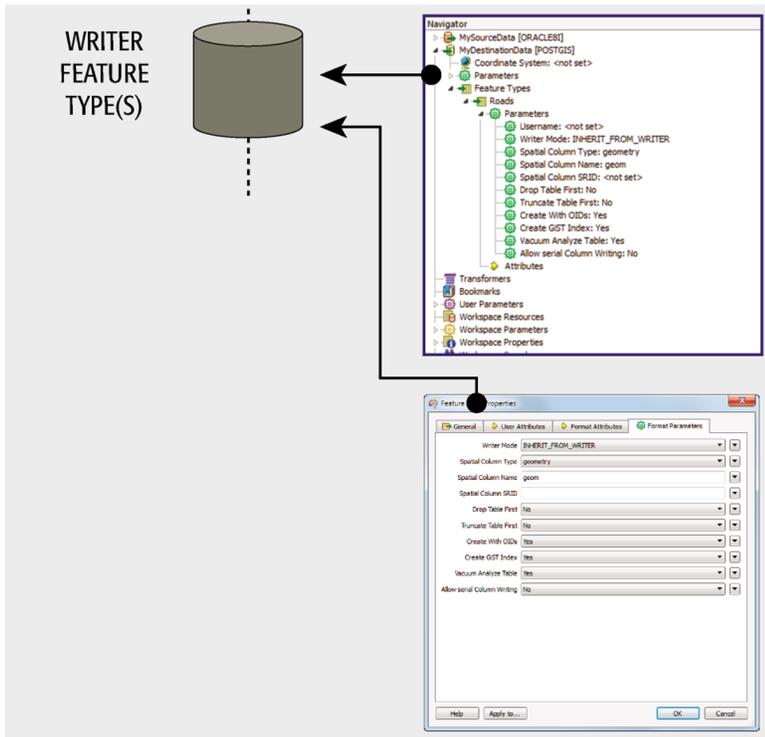
*Feature Type Parameters can also be accessed through the Feature Type Properties dialog. Notice the tab named **FormatParameters**.*

Not all feature types have parameters, so this tab is not always present.

Writer Feature Type Parameters

Writer feature type parameters apply to *writing* of specific layers/tables.

Again, most database formats have writer feature type parameters, but a high proportion of file-based formats also have these.



Create Spatial Index is a good example of a feature type parameter.

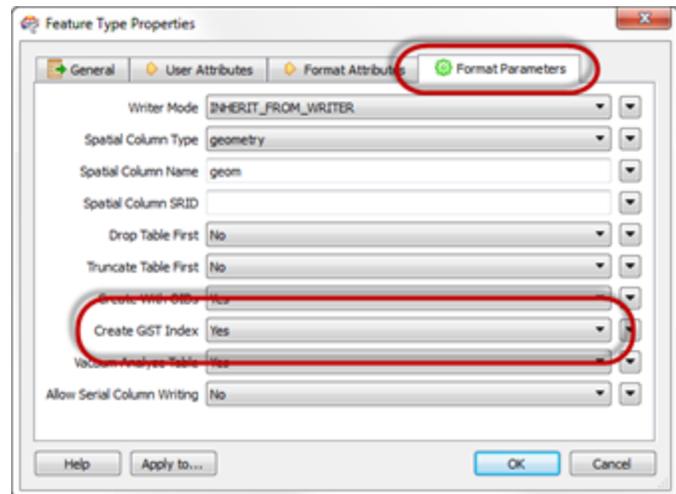
The decision about whether or not to apply an index is made on a table-by-table basis.

Not all tables may require an index. There can be a different index per table.

Therefore this is a feature type parameter.

If it were a writer-level parameter, then ALL tables would get an index; not necessarily what the user wants.

Conversely, no password parameter is listed because it applies to the entire database, not the individual tables.



Format Attributes



Besides ‘user attributes’ there is a whole range of attributes created by FME: Format Attributes

Although features are the lowest level in the hierarchy of translation components, it’s very useful to be able to control and manipulate individual features.

However, control of features isn’t done using parameters, but instead with a constituent part of features called Format Attributes.

Format Attributes

A format attribute is a built-in, FME-generated attribute. It represents part of the structure of a feature for any given format; i.e. information that isn’t generally carried as part of the geometry or as a user attribute. The color of a feature is one example of information held by format attributes.

FME uses these format attributes to keep track of such information and make sure it is passed on correctly to a destination dataset.

Format attributes are most obvious when viewing a dataset with FME Universal Viewer. Querying a feature causes both user attributes and format attributes to be reported.

For example, inspecting AutoCAD Map3D Object data in the FME Universal Viewer will show many format attributes, such as:

- autocad_color: Color of the feature
- autocad_entity: Type of geometry
- autocad_od_entity_key: Object Data ID
- autocad_source_filename: Source file

Other features, such as a point geometry, might have a format attribute to record rotation, while an arc feature would have format attributes to record arc length and angle.

Notice how the attribute name starts with a format keyword, to differentiate the same format attributes for different formats of data.

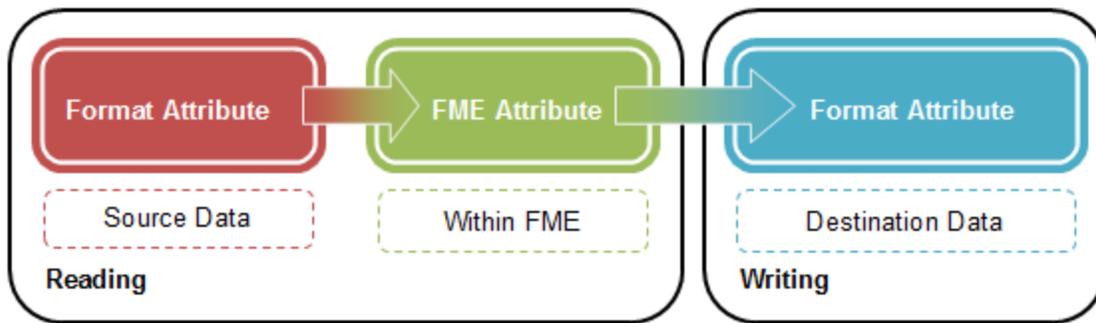
Feature: 1 of 1	
Feature Type: StateData	
Coord Sys: Unknown	
Attribute Name	Attribute Value
autocad_color	10
autocad_elevation	0
autocad_entity	autocad_line
autocad_entity_handle	F31
autocad_entity_visibility	visible
autocad_layer	Boundaries
autocad_layer_frozen	no
autocad_layer_hidden	no
autocad_layer_locked	no
autocad_linetype	ByLayer
autocad_linetype_generation	0
autocad_linetype_scale	1
autocad_lineweight	-1
autocad_map_odtable{0}	CountyData
autocad_map_odtable{1}	StateData
autocad_od_entity_key	F31
autocad_original_color	ByLayer
autocad_original_entity_type	autocad_lwpolyline
autocad_resolved_linetype	Continuous
autocad_source_filename	C:\FMEData\Data\Gov
autocad_space	model_space
autocad_thickness	0
autocad_width	0
fme_color	1,0,0
fme_geometry	fme_line
fme_type	fme_line
State	Texas

FME Attributes

A particular set of Format Attributes has the prefix **fme_**. These attributes represent the data as it is perceived by FME and are sometimes known as FME Attributes or Generic FME Attributes.

When a translation is carried out, the following occurs:

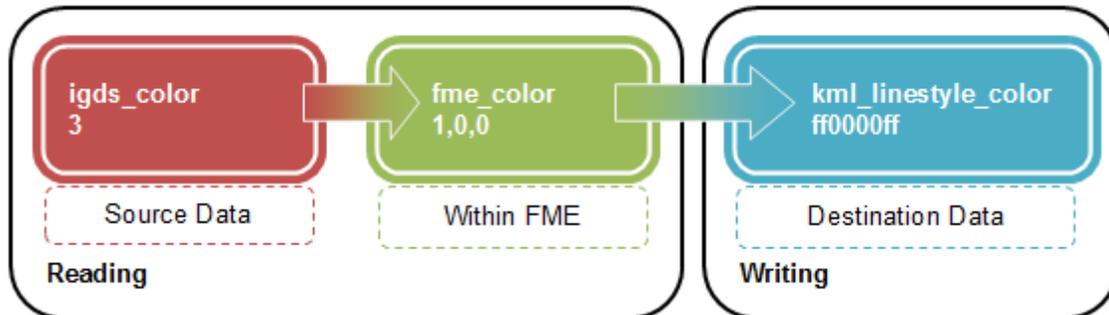
- FME reads the source data and stores information about its features as format attributes. These **format attributes** reflect the data that is stored in the original source data.
- FME converts the source data’s format attributes into **FME Attributes**. These FME attributes reflect the source data as it is perceived within FME.
- FME writes the destination data by creating a second set of format attributes. These format attributes reflect the information as it will be stored in the destination data.



This is why, when a user inspects data, there are two sets of attributes. In the previous screenshot were both *autocad_color* and *fme_color*; the latter is the FME representation of the former.

Using this method, FME can convert from one format to another, without having to separately map the source format attributes to the destination format attributes for every format.

FME merely converts everything to an FME standard and then from there to the Writer Format.



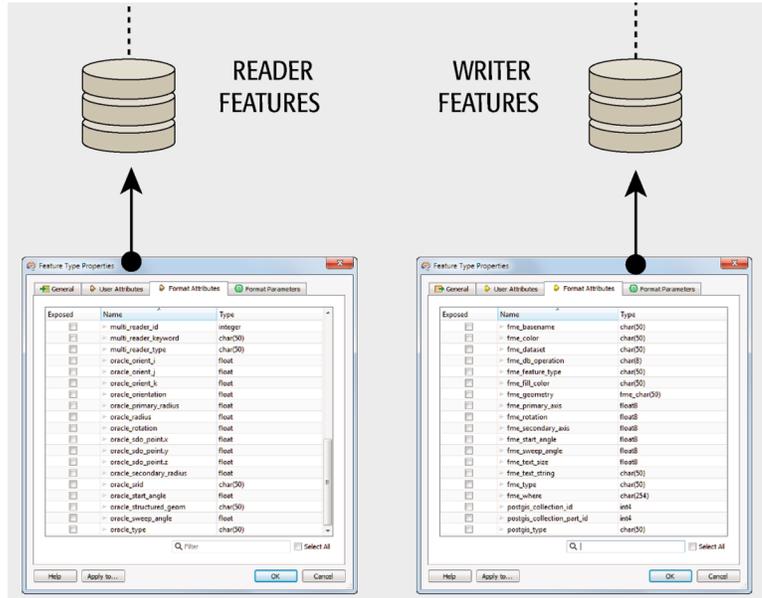
Controlling Features with Format Attributes

A user can make use of these attributes to carry out certain tasks by making the attributes part of the workspace.

As mentioned, a user doesn't have direct control over features through parameters, but instead uses format attributes.

However, to avoid cluttering the workspace these attributes are not all visible by default.

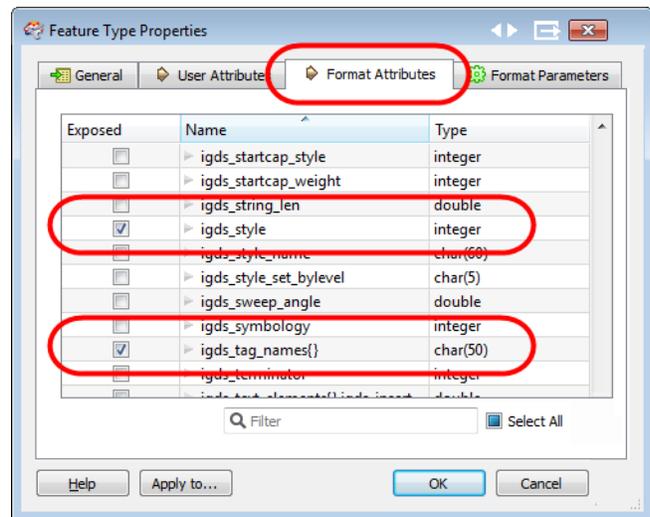
To make them visible is known as "exposing" them, and involves the Feature Types Properties dialog.



Exposing Format Attributes

To expose a format attribute, open the Feature Type Properties dialog, and click the 'Format Attributes' tab. Locate the Format Attribute to expose and check the box provided. Click OK to make the format attribute available for use within Workbench.

Format attributes might be a single attribute (for example, `igds_style`) or they might be a list-based format attribute, for example (`igds_tag_names{}`)



Firefighter Mapp says...

'An alternate method is to use the AttributeExposer transformer.'

Filtering with Format Attributes

One primary use of format attributes is as a means of filtering and directing source data within a workspace.

For example, suppose features in a source AutoCAD dataset are not divided into different layers as they should be. Because the user is able to determine the proper layer from maybe the color of the feature or the size of a text entity, they can expose the format attributes *autocad_color* or *autocad_text_size*, and use them to interpret the correct layer.

Most “filter” transformers (described in more detail in Chapter 6) can be used to process data in this way.

List Format Attributes

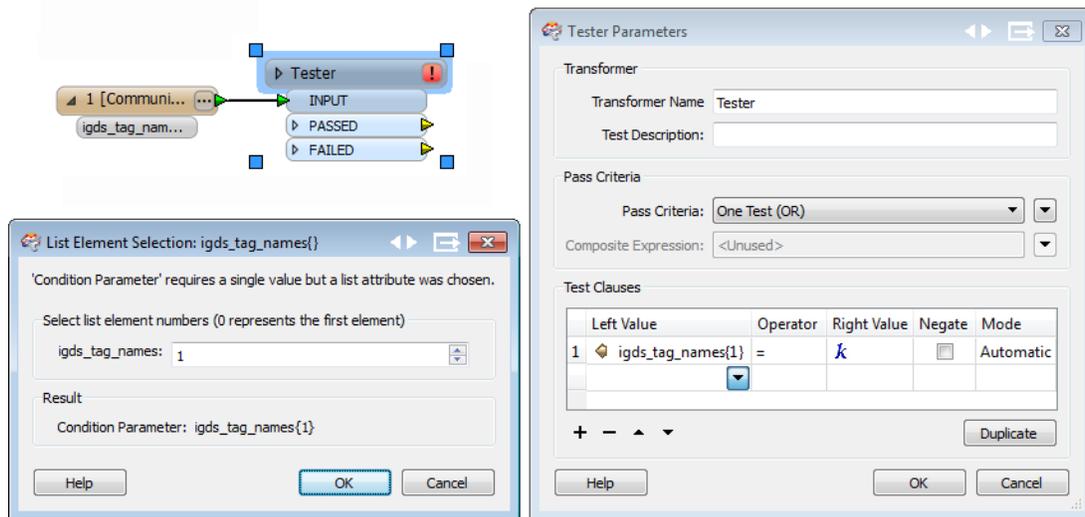
A List attribute is an FME structure that allows multiple values for each attribute. For example, an area of forestry might have a list of tree types (Pine, Oak, Cedar) in which case a list attribute in FME might be something like:

`parcelList.treeType{0} = Pine, parcelList.treeType{1} = Oak, parcelList.treeType{2} = Cedar`

This is important here because some format attributes can also be a list type of attribute.

In this example, the user has exposed the list format attribute `igds_tag_names{}`

When they attempt to use that attribute in a Tester transformer, they are prompted to choose which element in the list is to be tested. Because transformer dialogs ask which element in a list is to be used, it’s not necessary to have to expose multiple elements solely to get access to a single one.

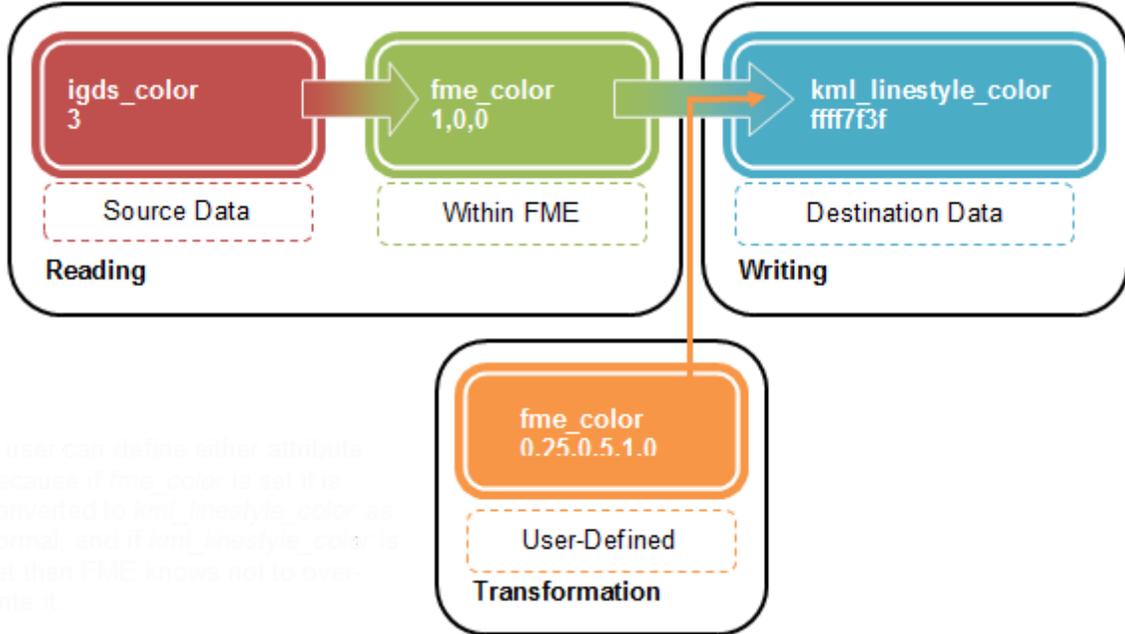


Transforming with Format Attributes

The other primary use of format attributes is to transform the data itself.

When writing data, FME attributes are turned into format attributes that reflect the data as it’s supposed to be written. However, a user can override this process by predefining the value of these attributes before they are sent to the writer.

In other words, setting a format attribute may cause a transformation in the data to take place. Either the writer format attribute (here *kml_linestyle_color*) or the equivalent FME attribute (here *fme_color*) may be set to achieve the same end.



A user can define either attribute because if *fme_color* is set it is converted to *kml_linestyle_color* as normal, and if *kml_linestyle_color* is set then FME knows not to overwrite it.

If a user manages to define both a format attribute and its FME equivalent, then the format attribute takes precedence and is used. For example, set *fme_color* and *kml_linestyle_color*, and the *kml* attribute gets priority.

This only really becomes a problem when reading and writing the same format, and the same format attributes exists on both reader and writer. In that case use the format attribute; it's not safe to use the fme equivalents.

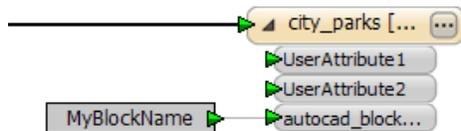
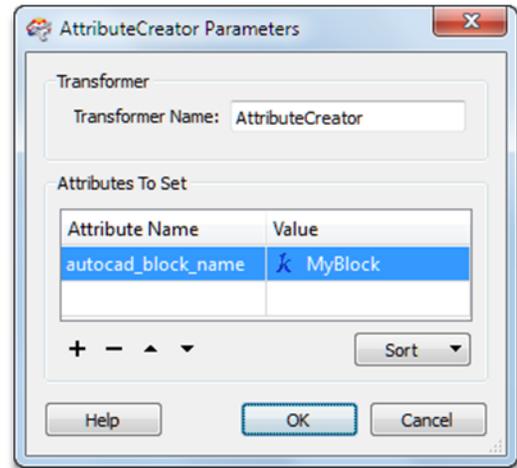
Transformers for Setting Format Attributes

Format Attributes can actually be a little tricky to set.

Once a format attribute is exposed on a reader feature type, then an *AttributeCreator* transformer can be used to change its value. But this won't have any effect unless the translation is reading and writing from the same format.

On the other hand, exposing a format attribute on a writer feature type doesn't make that attribute available in the workspace in the same way (i.e. it isn't exposed back upstream).

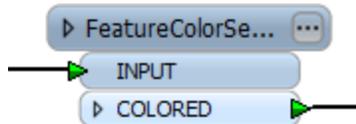
So, the most common method is to use the *AttributeCreator* transformer, but actually create the writer attribute and set it; for example create `autocad_block_name` and set a value for it.



The other common method is to use a constant, where format attributes exposed on a destination can be set by right-clicking the attribute and choosing 'Set to Constant Value'.

Used this way, the action is more like a Feature Type parameter; that is, it applies to all features written to that feature type.

Besides setting format attributes manually like this, there are a number of FME transformers that are designed to be merely a more user-friendly front end to setting a format attribute.



The *FeatureColorSetter* (for example) "Assigns color to incoming features", but in reality all it does is set a new value for the format attributes `fme_color`, `fme_fill_color`, etc.



The *DGNStyler* helps a user to define the symbology of features to be written to a MicroStation Design File. It is really just a more pleasant way of using format attributes.

Similar transformers are:

- *DWGStyler*
- *KMLPropertySetter*
- *KMLStyler*
- *MapInfoStyler*
- *PDFStyler*



Example 15: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Airport CITS Data Transfer Format (QLF), Esri File Geodatabase
Overall Goal	Add CAD data from the engineering department to a GIS dataset
Demonstrates	Feature Type Parameters, Format Attributes
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example15Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example15aComplete.fmw

On consideration, it's a good idea to empty a table before writing data to it, in case there were any features in the airport table that we no longer need. The simplest method is to re-run the workspace, dropping the table before re-writing to it.

The same technique should also be applied to the Roads table, since that would get a double set of data if the workspace were to be simply re-run.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 14. Alternatively you can open:

C:\FMEData\Workspaces\DesktopManual\Example15Begin.fmw

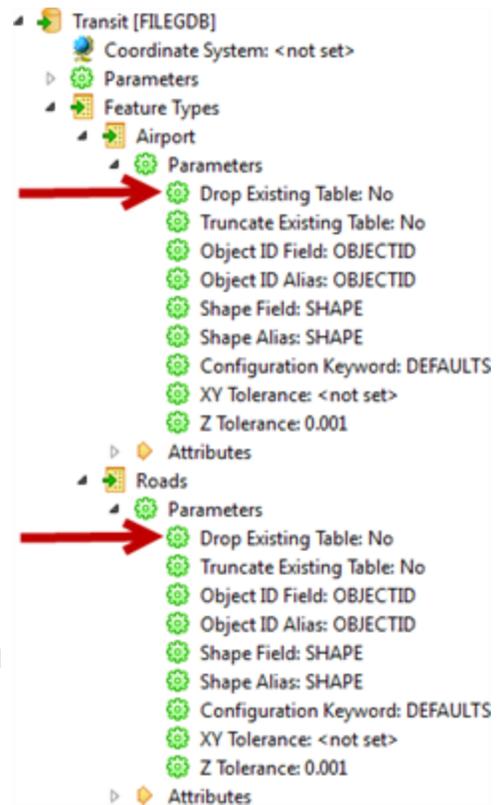
2) Set Feature Type Parameter

In the Navigator window, locate the Geodatabase writer feature type parameter 'Drop Existing Table' ('Drop Table First' in ArcObjects writer) for each of the Airport and Roads feature types.

Set them both to Yes. This will ensure the tables are emptied out before any data is written to them.

3) Run the Workspace

At this point you can re-run the workspace to prove that the Airport and Roads tables only contain a single set of geometry.



When inspecting the data you may notice that the *LineJoiner* transformer did not do a perfect job joining line features. That's because the *LineJoiner* works on geometry and lines won't be joined where there is more than one possible connection that can be made; for example, where two roads cross.

This can be solved by using a Group-By; that is, joining only lines with the same road ID should remove unwanted connections and let FME determine where the proper join lies.

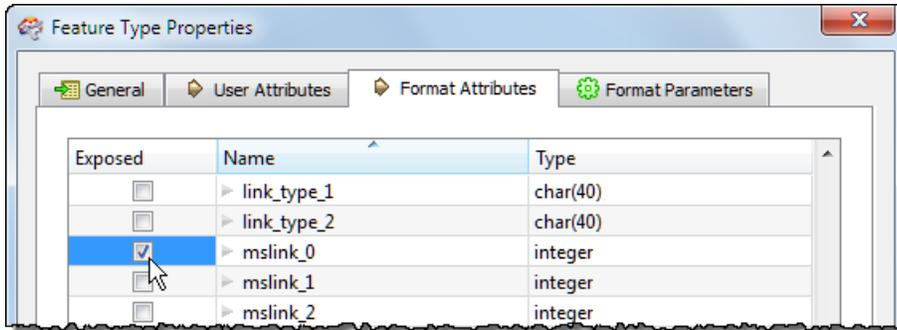
4) Expose a Format Attribute

The road ID to group by is not available as a user attribute. That's because MicroStation datasets do not permit attributes. However, MicroStation does permit basic feature IDs called 'mslinks'.

In FME this ID is available as a format attribute that must be exposed before it can be used by Workbench.

Open the properties dialog for the DGNV8:Roads reader feature type.

Click the Format Attributes tab. Place a check mark next to the entry for mslink_0 and click OK.



5) Fix LineJoiner Transformer

In the *LineJoiner* transformer set a group-by to group features by mslink_0.

6) Run Workspace

Run the workspace again and you should now find that all roads with the same ID are properly joined together.



Example 15a: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Esri File Geodatabase, Updated Bus Stops (Esri Shape)
Overall Goal	Add CAD data from the engineering department to a GIS dataset

Example 15a: CAD to GIS	
Demonstrates	Format Attributes, Database Updates
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example15bBegin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example15bComplete.fmw

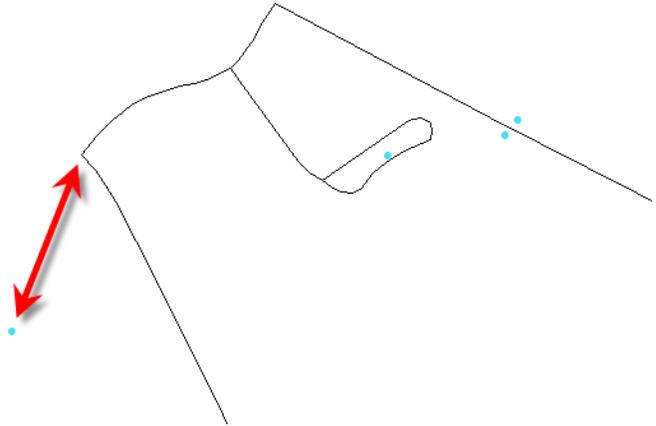
This example requires use of the ArcObjects File Geodatabase reader and so is only for users who have ArcGIS installed

The team responsible for managing transit data have released updates to the Bus Stops table.

One bus stop in particular was in the wrong position and has been moved.

However, the new data has been released as an “updates only” dataset and must be applied as an update to the existing bus stops table.

Since only individual features are being updated (not the whole table) this can be achieved using format attributes.



7) Add a Reader

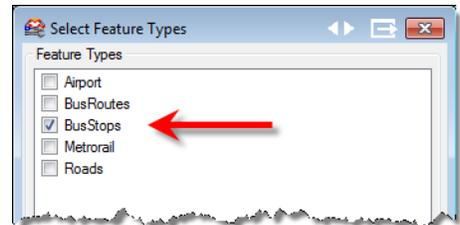
Select **Readers > Add Reader** from the menubar. Fill in the Add Reader dialog as follows:

Reader Format Esri Shape
Reader Dataset C:\FMEData\Data\Transit\BusStopUpdates\BusStops.shp

8) Import Feature Type

As the workspace doesn't yet have a WRITER feature type for BusStops, import it from the existing dataset Transit.gdb using **Writers > Import Feature Types**.

When prompted, only BusStops needs to be selected.

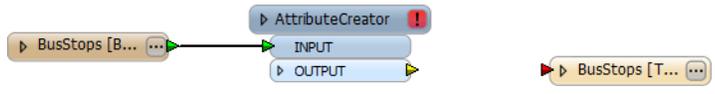


9) Place AttributeCreator Transformer

Format attributes for use on a writer are most easily defined using an *AttributeCreator* transformer.

This is because exposing them on a writer does not let you set them within the workspace.

Place an *AttributeCreator* transformer connected to the reader feature type for BusStops.



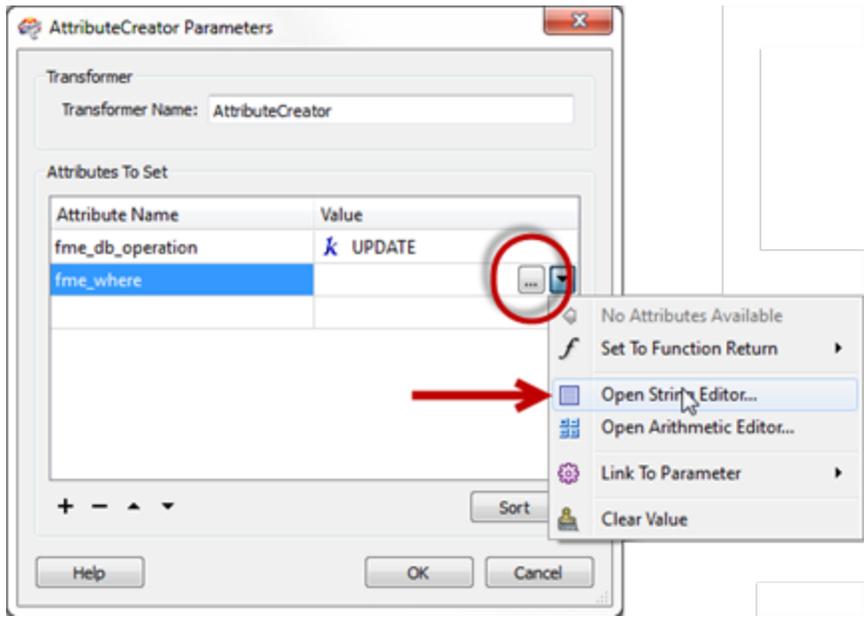
10) Set AttributeCreator Parameters

Open the *AttributeCreator* parameters.

In the Attribute Name field, create two new attributes: *fme_db_operation* and *fme_where*

Click in the value field for *fme_db_operation* and enter the value: UPDATE

Click on the menu for *fme_where* and choose the option: Open String Editor



11) Set String Editor Parameters

The string editor dialog will be used to construct a 'where' clause for the update.

It requires three parts to be concatenated.

Firstly, set a string type of Constant.

In the value field enter: STOPABBR='

Secondly, set a string type of Attribute Value.

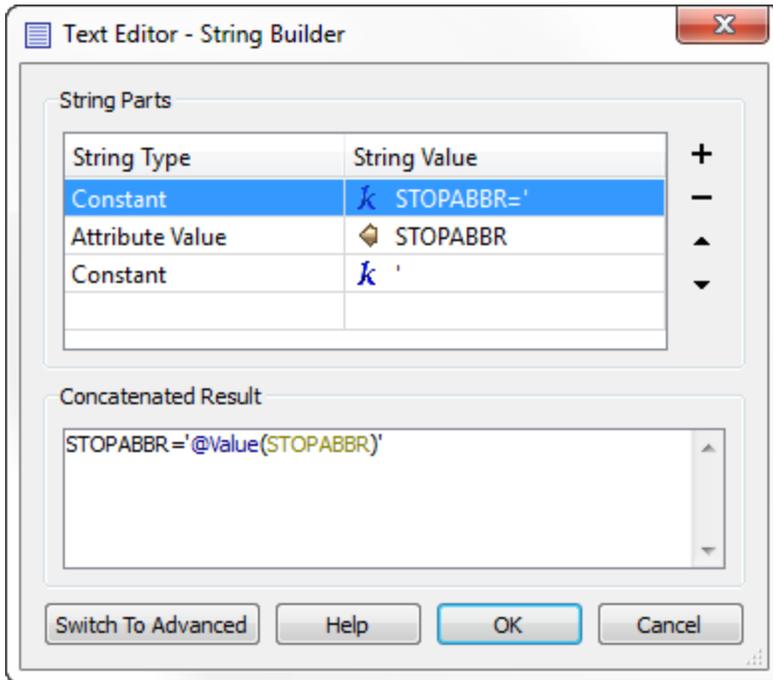
In the value field, select the attribute STOPABBR

Finally, set a string type of Constant again.

This time enter just a single quote character: '

The Concatenated Result field should now show: STOPABBR='[STOPABBR]'

...this is basically a 'where' clause that means "where the field STOPABBR matches the value of the incoming attribute STOPABBR". The quotes are required around the value because it is a text field (not just numeric).



12) Run Workspace

Connect the AttributeCreator:OUTPUT port to the Geodatabase BusStops feature type.

Run the workspace again and inspect the transit dataset.

You should now find that the erroneous bus stop has been moved into the correct position.

If you can't find the new bus stop, it should be located close to:

X: 3129150

Y: 10097400

Parameter Documentation



The FME Readers and Writers Reference Manual documents all of the parameters available for each format, from the reader and writer level down to format attributes for controlling features.

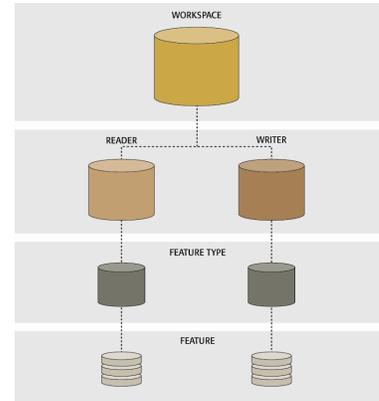
FME Readers and Writers Reference Manual

The FME Readers and Writers Reference Manual is part of the help system included with FME Workbench. It is where the parameters for each level of the translation hierarchy are documented.

This manual lists – format by format – what parameters exist for the reader, writer, feature types and features.

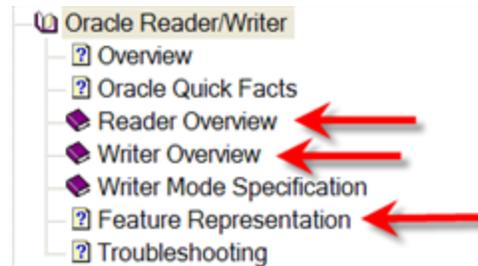
Importantly, each parameter includes a description of what it does, and in some cases what values are acceptable.

However, because the terminology is oriented to mapping files, it differs slightly from what is used in this manual.



In general: Reader/Writer parameters are called Directives, and feature type parameters are documented under a particular directive called a DEF line. Format Attributes are listed in a section called Feature Representation.

Translation Component	Reader and Writer Terminology
Reader Parameters	Reader Directives
Writer Parameters	Writer Directives
Reader Feature Type Parameters	Reader DEF Line Directive
Writer Feature Type Parameters	Writer DEF Line Directive
Format Attributes	Feature Representation Attributes



In the contents page Reader Directives (including the Reader DEF line) are accessed through Reader Overview.

Writer Directives (including the Writer DEF line) are accessed through Writer Overview

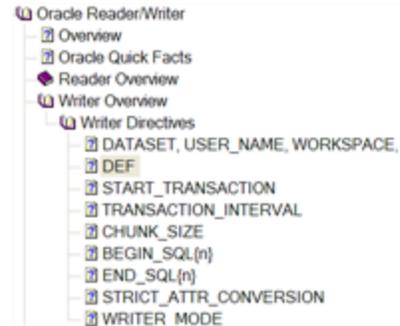
Feature Representation is where the list of format attributes is accessed.

Reader/Writer Directives

Taking the format Oracle Spatial Object as an example, the writer parameters are nicely mirrored in the Writer Directives section of the Manual.

For example, there are directives for transaction interval, chunk size, and writer mode.

Username and password are documented under the Oracle Reader Directives, and so aren't repeated here.



Feature Type Directives

Notice one of the writer directives is called DEF. This is where all of the writer feature type parameters are documented; for example Drop Table, Truncate Table, and Update Key.

DEF

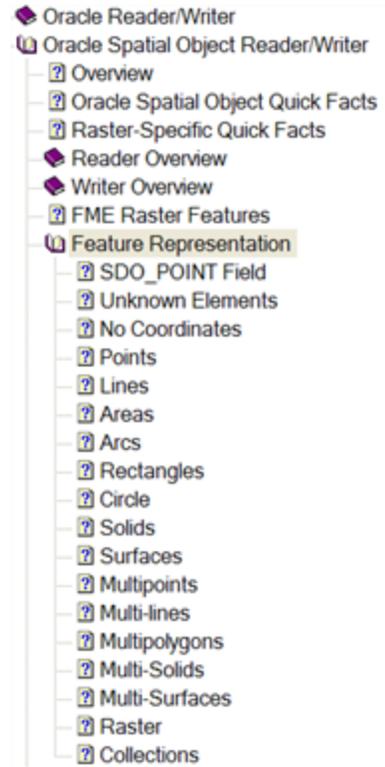
Required/Optional: *Required*

Each Oracle Database table must be defined before it can be written. The general form of an Oracle Database definition statement is:

```
ORACLESI_DB_DEF <tableName> \
  [oracle_sql_encoded <sqlQuery>] \
  [oracle_update_key_columns <column>[,<column>]... \
  [oracle_delete_key_columns <column>[,<column>]... \
  [oracle_drop_table (yes|no)] \
  [oracle_truncate_table (yes|no)] \
  [oracle_params <creationParams>] \
  [oracle_sequenced_cols column[:seqname][:column[:seqname]]...] \
  [<fieldName> <fieldType>]*
```

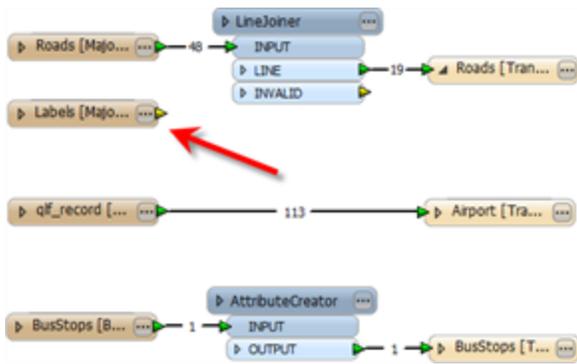
Feature Representation

The Feature Representation section is divided up into different geometry types. Each geometry has a list of applicable format attributes; for example a geometry type of oracle_circle possesses format attributes that describe circle radius (oracle_radius) and circle rotation (oracle_rotation).



Example16: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	File Geodatabase
Overall Goal	Add CAD data from the engineering department to a public GIS dataset
Demonstrates	Use of Readers and Writers Manual
Starting Workspace	None
Finished Workspace	None

Notice that the Labels feature type from the source MicroStation dataset has not been connected up to any table in the Geodatabase output.



As a very quick example, check the Quick Facts section of the Readers and Writers Manual for the two Esri Geodatabase reader/writers.

Can you tell if it is even possible to write text features to a Geodatabase?

Published Parameters



Publishing parameters is a method by which FME prompts to change read and write control parameters at runtime.

As shown, each different level of translation hierarchy has a related set of control parameters.

However, there are two potential users of FME; a workspace author and the end-user.

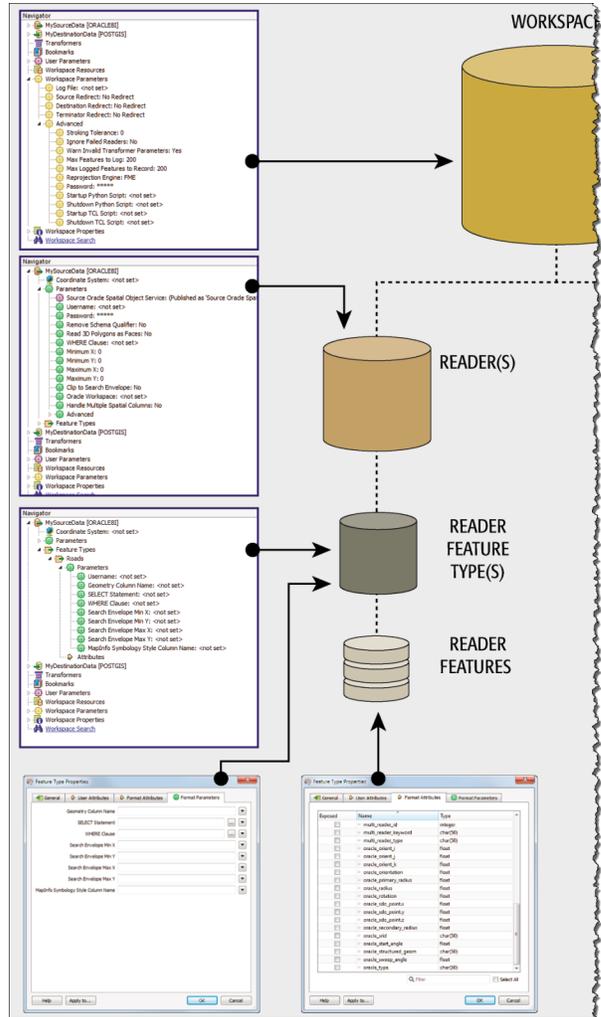
Ideally, if the end-user wanted to change one or more of these parameters, they should not need to edit the workspace in the same way that an author would.

Published Parameters provide this functionality.

Publishing parameters is a way to prompt the end user to enter a value, in much the same way that FME will prompt for any undefined mandatory parameters.

Prompting the user for values thus avoids the need to make manual edits in the workspace.

Any setting that can be defined through the Navigator window is capable of being set as a published parameter. This includes most workspace parameters, all reader and writer parameters, and all feature type parameters.





Chef Bimm says...

"Think of Published Parameters as your coffee options printed on the side of a paper cup.

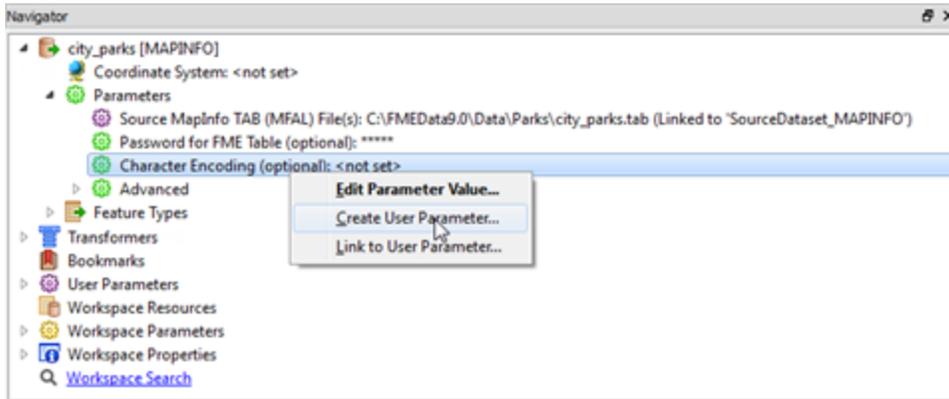
They are a way for a customer to have their wishes defined without setting the machine themselves".



Publishing Parameters

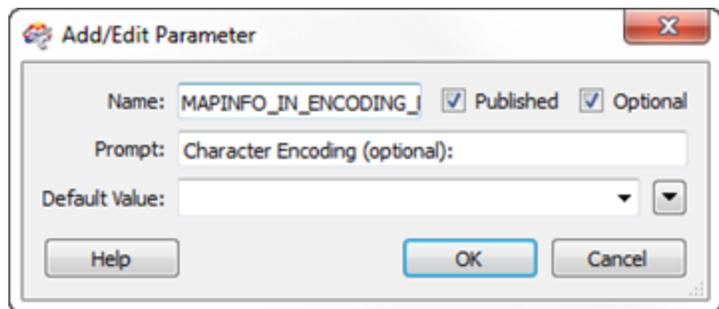
To publish a parameter, simply locate it in the Navigator window, right-click it, and choose the **Create User Parameter** option. This opens a dialog for defining the publishing settings.

Here the user is choosing to publish a reader parameter that sets what character encoding the source data is in. The purple color of the Source Dataset parameter shows that this has already been published.



Notice how the parameter definition includes:

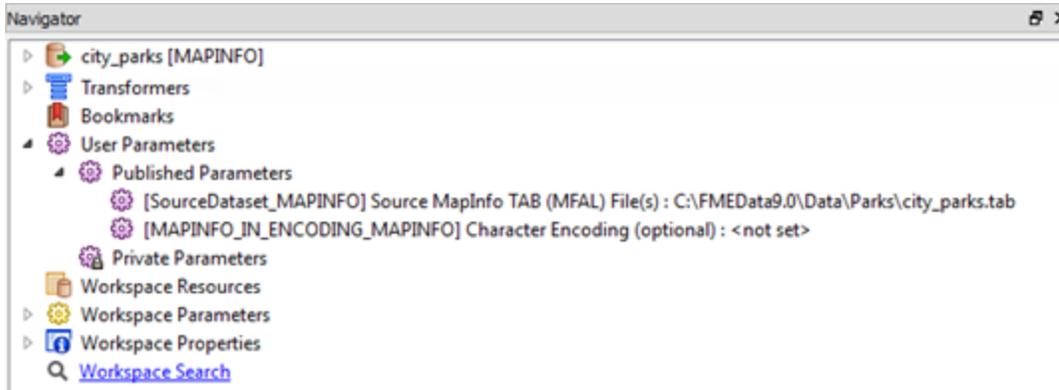
- Parameter name
- Parameter prompt
- Default value
- Optional Flag
- Published Flag



The 'published' flag exists because there are two different parameter states: public and private.

- Public means the parameter is published for an end-user to set.
- Private means it can only be used inside the workspace, but shareable in several places.

For ease of editing, all published parameters also appear in a separate section of the Navigator.

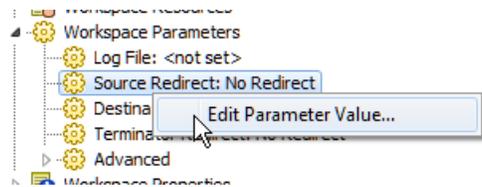


Publishing Workspace Parameters

To publish a workspace parameter simply locate the parameter in the Navigator, right-click, and choose Create User Parameter.

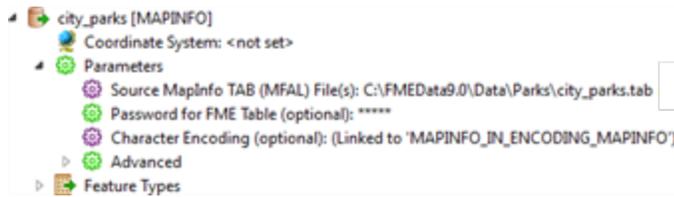
The limitation here is that many basic parameters (such as Destination Redirect or Workspace Password) can't be published because it doesn't make sense to do so.

In such cases that option is simply left off the menu:



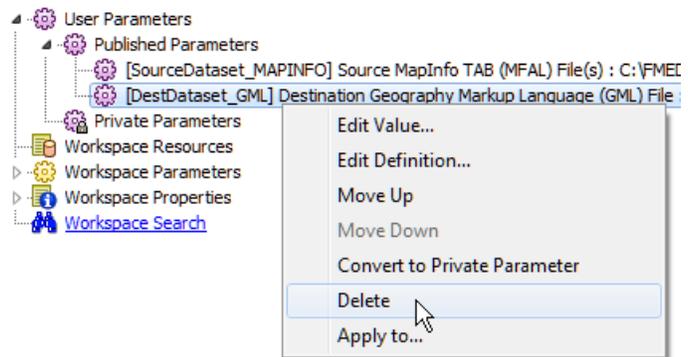
Publishing Reader/Writer Parameters

Publishing reader and writer parameters is the most common use of this functionality. It is achieved by using the same right-click context menu in the Navigator window.



Notice that some parameters are automatically published by FME. That's because these are common parameters the user will often need to set; for example reader input file and writer output file locations.

If it's not appropriate for a user to select these, then the published parameters can be simply removed.



Publishing Feature Type Parameters

For Feature Type parameters, it's important to notice that these can only be published in the Navigator window. There are NO right-click > publish options in the Feature Type Properties dialog.

Publishing Format Attributes

For publishing Format Attributes, or any other attribute or transformer values, see Chapter 5.



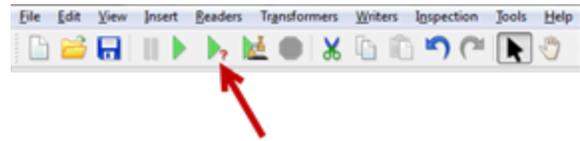
First-Officer Transformer says...

'Besides translation components for reading and writing, you can also publish parameters for transformers; again by locating the parameter to be published in the Navigator window, right-clicking, and choosing Create User Parameter.'

Using Published Parameters

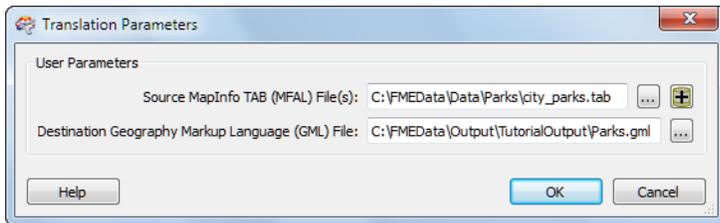
Published parameters are activated whenever a workspace is run using the option **File > Prompt and Run Translation** from the menu bar in FME Workbench.

The shortcut for this command is *Ctrl+R* and there is a related Prompt and Run button on the toolbar.



Note: *The simpler File > Run (shortcut F5) will not prompt for values, but re-use existing ones.*

When a workspace is run in prompt mode, the user receives a dialog prompting them to enter new values.



Running a workspace in the FME Quick Translator always prompts for parameters; which is why it's so suitable for non-FME-authoring end-users to run a workspace.

Published Parameters on the Command Line

In the same way that FME translations can be run from the command line, they can be passed values to published parameters on the command line using the syntax:

```
--<Parameter Name> <Parameter Value>
```

Notice that the parameter name matches that supplied in the published parameter definition.

Windows command-line to run this workspace:

```
fme.exe Ex5-StructuralTransformation-begin.fmw
--SourceDataset_MAPINFO C:\FMEData\Data\Parks\city_parks.tab
--DestDataset_GML C:\FMEData\Output\TutorialOutput\Parks.gml
```

As usual, the log window reveals the command line used in the above translation.

The command line in the Log window specifically states 'Windows command-line'.

The biggest difference between Windows and UNIX shells is how parameters with spaces in them are quoted.



Example 17: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Esri File Geodatabase
Overall Goal	Add CAD data from the engineering department to a public GIS dataset
Demonstrates	Feature Type Parameters, Format Attributes
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example17Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example17Complete.fmw

As a final step in this project, let's make sure the user has access to no more parameters than are necessary.

1) Start Workbench

Start Workbench (if necessary) and open the completed workspace from Example 15.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example17Begin.fmw*

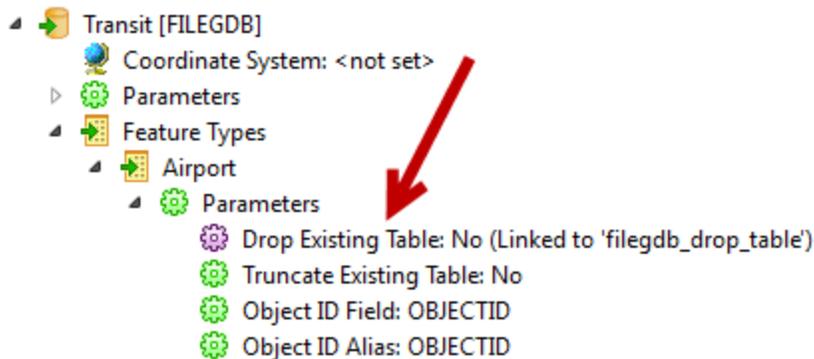
2) Delete Parameters

Because it's not necessary for end-users to change the source or destination datasets, locate the published parameters for all reader and writer dataset parameters.

Either delete them from the Published Parameters section or un-publish them from the Reader/Writer sections.

3) Publish Parameter

Publish the 'Overwrite Table' parameters for both Geodatabase Roads and Airport tables so that users are prompted at runtime whether to drop the contents or not. Make sure you alter the prompts so that the user can tell which parameter refers to which table.



4) Run the Workspace Again

Use **File > Prompt and Run** to run the workspace again to prove that the published parameters now prompt for a value.



Module Review



This session was designed to increase your knowledge of how FME handles different spatial data formats.

What You Should Have Learned from this Module

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

Theory

- A Workspace has a hierarchy of ***Readers*** and ***Writers***, ***Feature Types***, and ***Features***
- Translations are controlled by ***Read and Write Parameters*** and ***Format Attributes***, which have a similar hierarchy to the structure of Reader-Dataset-Feature Type.
- ***Format Attributes*** store information related to the structure and symbology of a feature. Format attributes can be used to ***filter*** source features, or to ***re-symbolize*** or ***re-structure destination*** features.
- ***Published Parameters*** are a way to prompt users to set values at run time.

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 apply Format Attributes on either source or destination features.
- The ability to find out how features will be affected when converting between formats.
- The ability to publish parameters from readers and writers.

FME and Data Formats



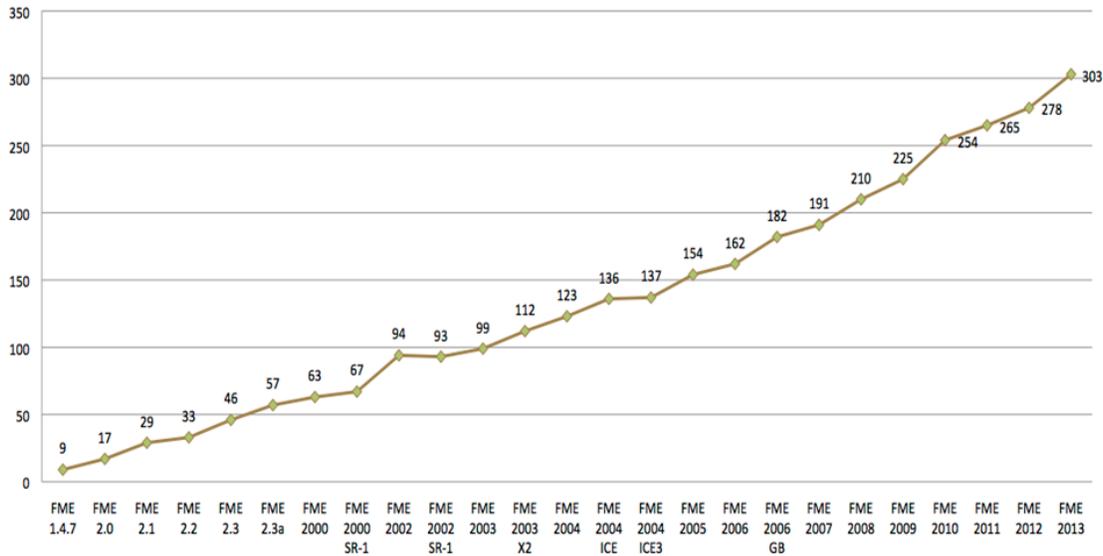
Interoperability is the ability to exchange information between systems and applications. FME aids interoperability by supporting translations between numerous spatial and non-spatial data formats.

Supported Formats

FME supports over 300 data formats.

Surprisingly the rate of increase has been steady over the last fifteen years and shows little sign of levelling out.

FME Supported Formats by Release



Course Overview



- Who am I?
- FME Certified Trainers
- Course Goal
- Course Structure



Mr. Saif-Investor says...

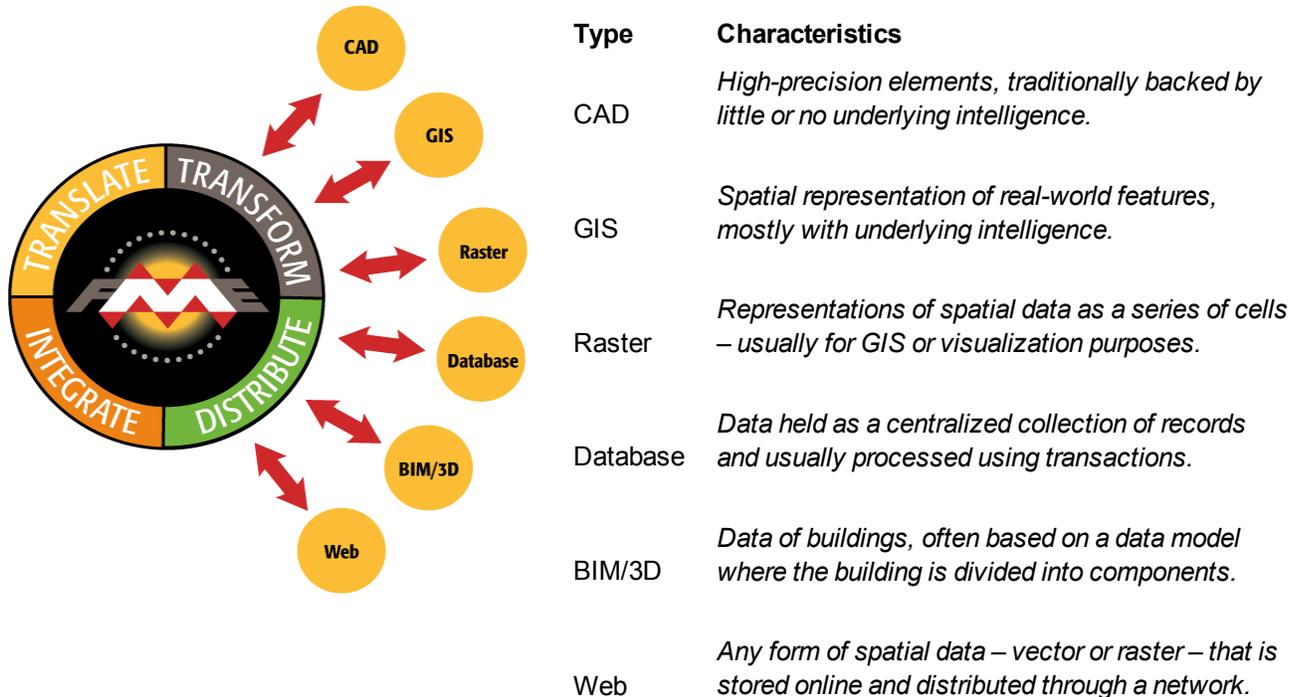
'Although past performance is no guarantee of future results, this is one graph I don't see turning downwards any time soon.

FME's ability to work simultaneously with multiple formats -in a totally semantic manner - lets me bring together widely disparate datasets in the form I want and with minimal effort on my part.'

Welcome to FME Desktop Training
FME Desktop Training 2013

Why So Many Formats?

The large number of formats arises because there are so many different fields that use spatial data. At Safe we sometimes call these *families* of data and each of these families has data with a set of characteristics that differentiates it from the structures of other data types.



To the preceding families, the following items could also be added:

Cartographic	<i>Data optimized to visually highlight certain spatial characteristics or concepts.</i>
Transfer	<i>Formats specifically designed as a means to standardize the supply of data.</i>
Point Cloud	<i>Formats specialized for storing and transferring large quantities of point data</i>
XML	<i>Formats based on XML or GML</i>

Of course, the big challenge is to preserve meaning and content when working with different types of data; for example, combining cartographic and database data into a GIS ready output. This is where in-depth knowledge of FME’s readers and writers is a great benefit.

Non Spatial

FME also supports a number of non-spatial formats. Therefore, not only can FME work with the non-spatial attributes of spatial features, it’s also able to work on a totally non-spatial basis.

Reading or Writing

It’s worth noting that not every FME supported format permits both reading and writing. Some formats only support reading (for example, Esri ArcGIS mxd files), whereas others only support writing; for example, SVG. However the majority do have support for both.

Licensing and System Requirements

FME is available in various editions. The FME edition the user gets is set by their license, rather than by the product that is installed; the FME installation download/file is the same for all editions.

Most FME editions differ only in the formats available. Only the FME Desktop Base Edition has a different (more limited) set of functionality. Editions are named for the formats they support; for example, support for GE Smallworld format datasets is provided by the FME Smallworld Edition.

Some formats are only supported when suitable application software is also installed on the user’s system. Some Esri Geodatabase formats are an example of this. Because some FME readers and writers use ArcObjects to communicate with a Geodatabase, it’s necessary that ArcGIS is installed and licensed to enable such writers in FME.

Similar limitations restrict the availability of formats on a particular platform; for example some formats aren’t available on Linux or Windows 64-bit operating systems, because the required technology (or application) just isn’t available for that platform.

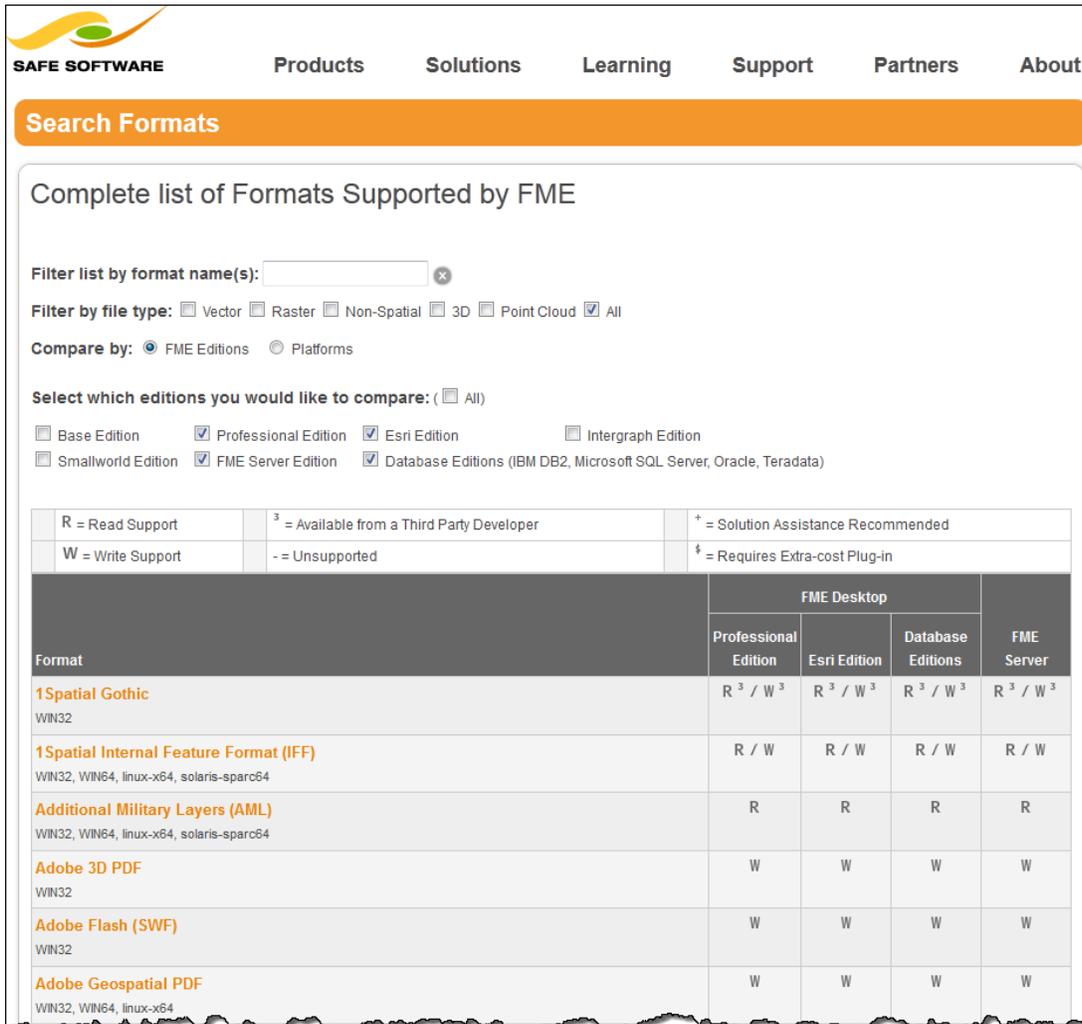
Format Plug-Ins

Some FME supported formats relate to very specialist data types. In these cases support is provided by an extra-cost plug-in. In some cases these plug-ins are created by third-party suppliers using the FME Plug-In SDK.



Ms. Surveyor says...

‘The full list of supported formats - including the edition they are supported by - is available on Safe Software’s web site at www.safe.com/formats.’



SAFE SOFTWARE Products Solutions Learning Support Partners About

Search Formats

Complete list of Formats Supported by FME

Filter list by format name(s):

Filter by file type: Vector Raster Non-Spatial 3D Point Cloud All

Compare by: FME Editions Platforms

Select which editions you would like to compare: (All)

Base Edition Professional Edition Esri Edition Intergraph Edition
 Smallworld Edition FME Server Edition Database Editions (IBM DB2, Microsoft SQL Server, Oracle, Teradata)

R = Read Support	³ = Available from a Third Party Developer	⁺ = Solution Assistance Recommended
W = Write Support	- = Unsupported	[‡] = Requires Extra-cost Plug-in

Format	FME Desktop			
	Professional Edition	Esri Edition	Database Editions	FME Server
1Spatial Gothic WIN32	R ³ / W ³			
1Spatial Internal Feature Format (IFF) WIN32, WIN64, linux-x64, solaris-sparc64	R / W	R / W	R / W	R / W
Additional Military Layers (AML) WIN32, WIN64, linux-x64, solaris-sparc64	R	R	R	R
Adobe 3D PDF WIN32	W	W	W	W
Adobe Flash (SWF) WIN32	W	W	W	W
Adobe Geospatial PDF WIN32, WIN64, linux-x64	W	W	W	W



Knowing the FME Version, FME edition, license type, and platform is key to determining whether a required format is supported by a certain FME installation.

*This information can be obtained in FME Workbench by selecting **Help** > **About FME Workbench** from the menubar, and then clicking the **More Info...** button.*



Example 18: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Various
Overall Goal	Check pre-requisites for translations
Demonstrates	Supported Formats
Starting Workspace	None
Finished Workspace	None

Imagine you have been asked to translate data from one format to another. The first step in this task is to check for any format pre-requisites.

1) Start Workbench

Start FME Workbench and select **Help > About FME Workbench**.

Click the ‘More Info...’ button to determine details about the FME build and license.

2) Check Format Requirements

Visit the Safe Software web site and visit the “Search formats” page to check the requirements for various translations.

Reader

Bentley MicroStation v8

Oracle Spatial Object

AutoCAD Map 3D

GeoTIFF

Writer

Esri File Geodatabase

S-57

Adobe Geospatial PDF

ER Mapper ECW

Products
Solutions
Learning

FME Technology

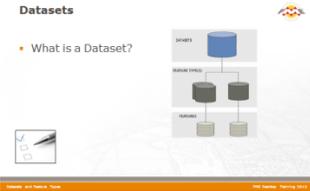
- FME Desktop
- FME Server
- Search Formats

Key Capabilities

- Data Transformation
- Data Conversion
- Data Distribution / Sharing
- Data Integration
- Data Migration / Loading
- Data Validation & QA
- Coordinate Reprojection
- Real Time Data
- Spatial ETL

In particular check for any required application software, the level of FME licensing required, and the supported platforms.

Datasets



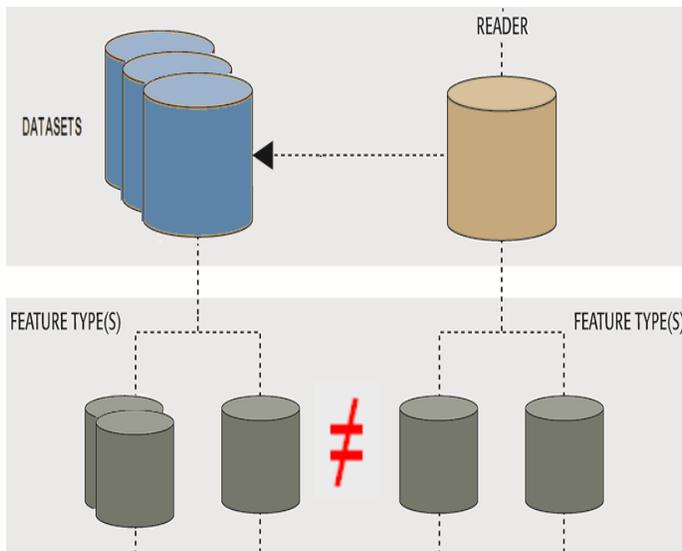
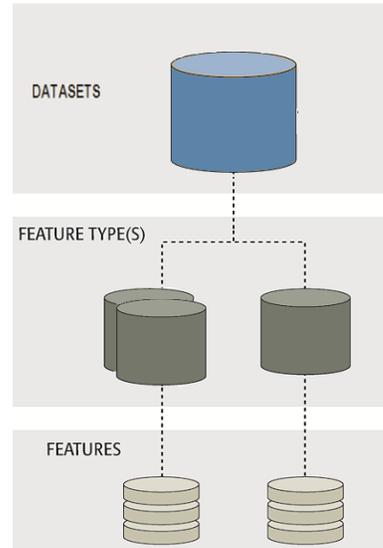
Understanding of the term “Dataset” is one of the keys to unlocking full understanding of FME functionality.

What is a Dataset?

The Association for Geographic Information defines a dataset as "an organized collection of data with a common theme", and this is a good description of what the term means within FME.

Datasets can be represented in a hierarchical sense. Generally each dataset can be described as a number of feature types (or layers), each of which has a number of features within it.

This definition of a dataset is very similar to the definition of a translation. This is helpful, but there are complications caused by different types of dataset.



Types of Dataset

All the FME-supported formats can be placed into one of four classifications of dataset:

- File
- Folder
- Database
- Web

For FME's purposes, the key differentiator is how the feature types are stored.

File-Based Datasets

The key point here is that all feature types (layers) are stored within a single file.

A format in this classification will have a way to assign data to different layers within a single file.

When read into FME these layers become the different feature types.

The name of the dataset is therefore taken directly from the name of the file.

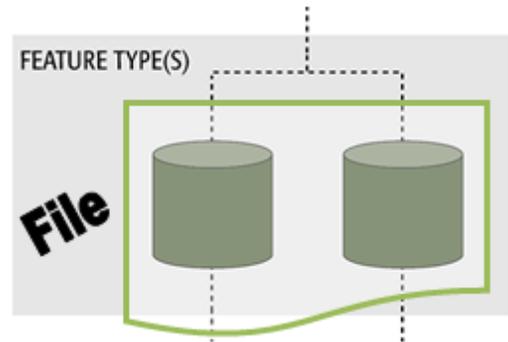
The names of the feature types are taken from the layers stored within that file.

To repeat: what is important is that *the feature types are all stored within the same 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.

Sometimes these formats include several subsidiary files; for example an AutoCAD dataset might include a PRJ file to record the coordinate system, but that’s not really relevant for our definition.

Dataset Type	<i>File</i>
Dataset Name	<i>Name of file</i>
Feature Type	<i>Layer within file</i>
Feature Type Name	<i>Name of 'layer'</i>
Example Format	<i>AutoCAD DWG</i>



Folder-Based Datasets

In this class of format, feature types (layers) are stored as separate files.

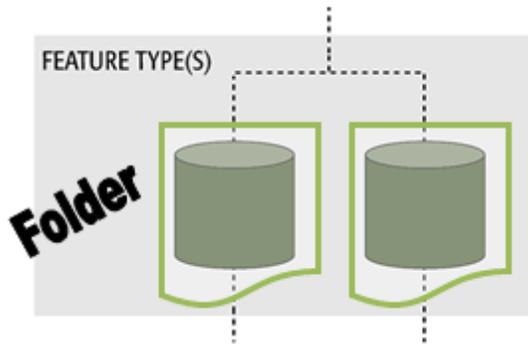
A format of this type DOES NOT have a way to assign data to different layers within a single file; therefore each 'layer' is a separate file.

When read into FME, each of the files becomes a feature type in in Workbench.

The name of the dataset is therefore taken from the folder or directory holding the data.

The names of the feature types are taken from the files themselves.

Dataset Type	<i>Folder</i>
Dataset Name	<i>Name of folder</i>
Feature Type	<i>File within folder</i>
Feature Type Name	<i>Name of file</i>
Example Format	<i>Esri Shape</i>



To repeat: what is important is that *each feature type is stored within a separate file.*

A Shape dataset is a good example: each layer of data is held in a separate *shp* file and no file can have multiple layers.

For example, take the following Esri Shape data structure:

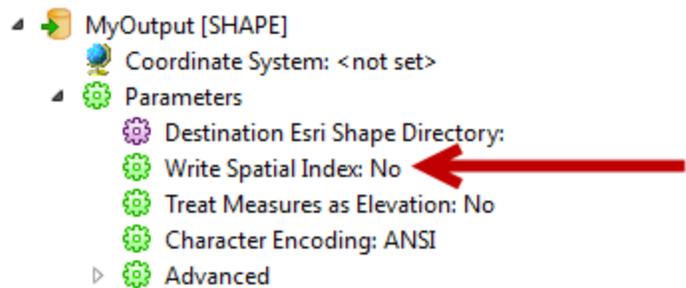
```
C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesL.shp
C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesM.shp
C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesH.shp
```

The dataset is called 'BikeRoutes', from the folder of the same name.

The feature types are "BikeRoutesL", "BikeRoutesM", and "BikeRoutesH", from the files of those names.



For all you aficionados of Shape format, note that FME 2013 includes the ability to write Shape indexes – either a spatial or an attribute index. To do so simply requires the existence of ArcGIS (or ArcObjects) on the same computer.



Database Datasets

A database dataset, as the name implies, is a set of data stored within a database.

A database divides data into tables; therefore each feature type is stored as a separate table within that database.

When read into FME, each table becomes a feature type in Workbench.

Dataset Type	<i>Database</i>
Dataset Name	<i>Name of Database</i>
Feature Type	<i>Table within Database</i>
Feature Type Name	<i>Name of Table</i>
Example Format	<i>PostGIS</i>

The name of the dataset is therefore taken from the name of the database holding the data (to be strictly accurate, each user/schema within a database would be a different dataset). The names of the feature types are taken from the table names themselves.

To apply the previous example in this scenario, a PostGIS database called 'BikeRoutes' (the dataset) could have tables called 'BikeRoutesL', 'BikeRoutesM' and 'BikeRoutesH' (the feature types).

Web Datasets

A web dataset is a collection of data stored on an Internet site. A Web Feature Service (WFS) server is an example of this.

In this case the name of the dataset is the same as the name of the URL (Universal Resource Locator) or service.

Some formats can be both a file/folder format and a web format; for example FME can read a GeorSS dataset locally from a file or remotely as a web feed.

Dataset Type	Web
Dataset Name	Name of URL/Service
Feature Type	Layer within web site
Feature Type Name	Name of Layer
Example Format	WFS

Web datasets are commonly returned in a format such as XML, GML, GeoRSS or GeoJSON.

Each of these will have a schema that defines the layers. Each layer represents a different feature type. WFS format is one example of this type of dataset.



The Readers and Writers manual lists the type of dataset for each format.

Esri Geodatabase can be a true database (as in ArcSDE) but also a File format (Personal Geodatabase) and a Folder format (File Geodatabase)!

FME Readers and Writers – FME 2011

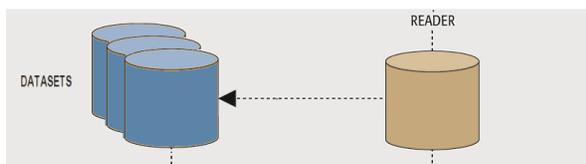
You are here: [Esri Geodatabase Reader/Writer](#) > Geodatabase Quick Facts

Geodatabase Quick Facts

Dataset Type	Database (ArcSDE) File (Access) Directory (File-based)
--------------	--

Dataset Parameters

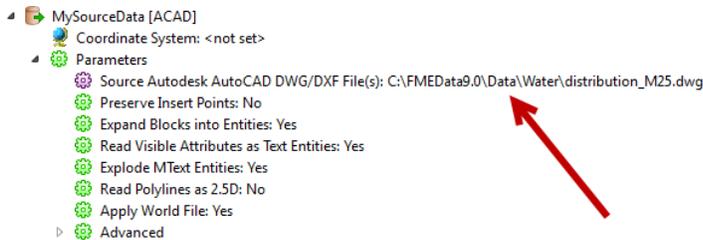
As you may have noticed, a Reader is tied to one or more datasets.



Each reader and writer has one or more parameters in which to define the location of the related datasets. As usual, double-clicking the parameter opens a dialog for it to be changed.

File Dataset Parameters

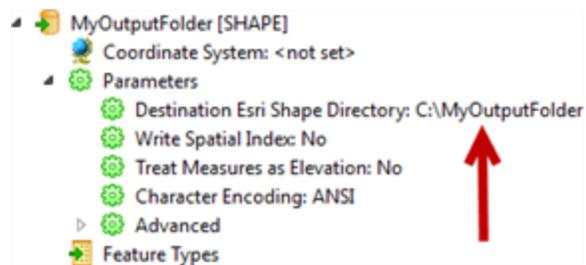
Remember, a file dataset is usually defined by a single file, so this parameter is simply the location of the file(s).



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

Folder Dataset Parameters

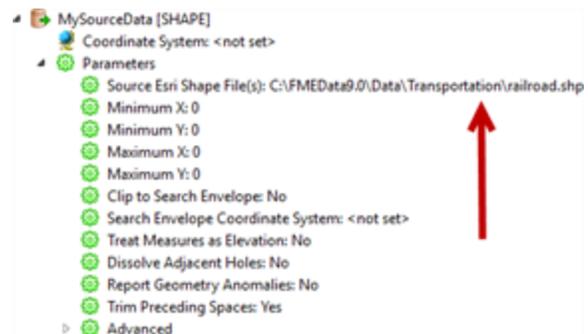
Remember, folder-based datasets are defined by several files, one for each feature type.



The writing side is easy to understand. Selecting a dataset to write to is achieved by selecting a **folder** into which to write the data. This is logical: select a folder to define the folder-based dataset.

Reading is not quite as logical. 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.

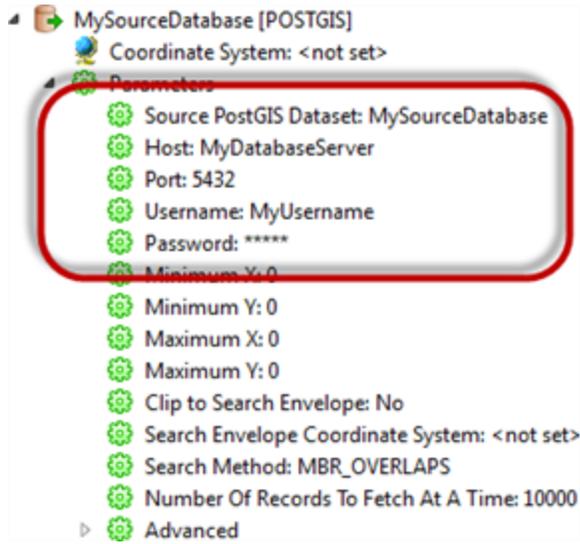


In FME2013 a File or Folder dataset can be read directly from a Zip file. Simply select the zip file in the source parameter. FME will extract the data when it is being read.

Similarly, FME2013 can read a File or Folder dataset directly from a URL. Simply enter the URL into the source parameter. For Folder datasets the URL must point to a zip file.

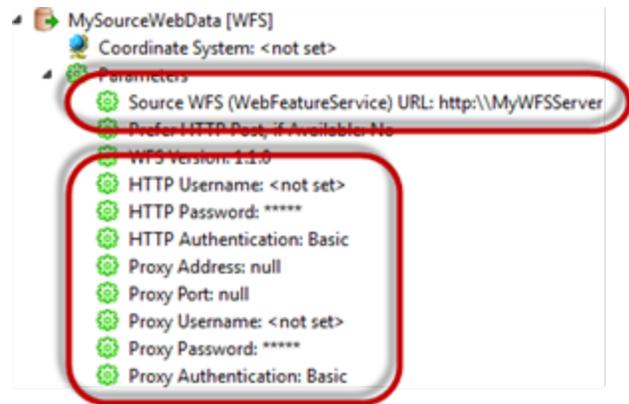
Database Dataset Parameters

A database dataset will usually have a series of connection parameters, including database name, server, port number, username and password.



Web Dataset Parameters

A web dataset will often have a parameter for defining the source URL, and others for specifying authentication and proxy connectivity.



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

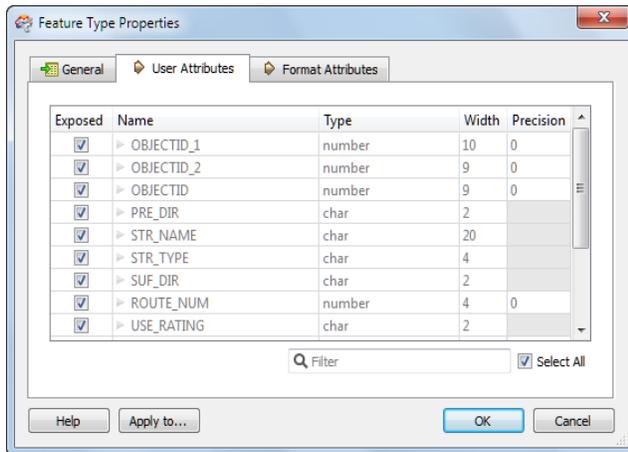
When using dataset parameters it's possible to change a Reader to read a different source dataset (or datasets) than those selected when the workspace was created initially.

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.

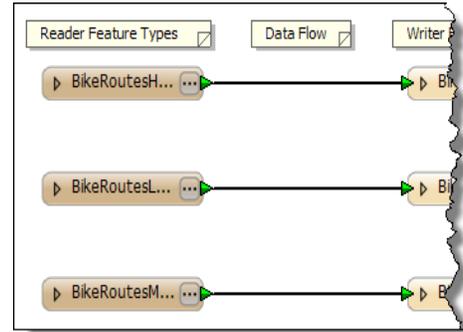
Datasets with Different Feature Types

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.



Example 19: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Community Map (MicroStation DGN, File Geodatabase)
Overall Goal	Convert the community map data to Esri File Geodatabase
Demonstrates	Reading datasets with different feature types
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example19Complete.fmw

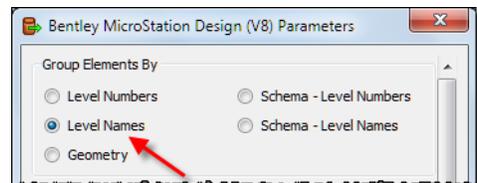
The City of Interopolis has base maps that are used as a background to the Official Community Plan. CommunityPlanBasemap2012.dgn is the dataset used as background to the 2013 plan.

The planning department has requested that you convert the Community Plan dataset to Esri File Geodatabase format for them.

1) Create a Workspace

Start Workbench if necessary. Create a workspace with the following parameters:

- Reader Format** Bentley MicroStation Design (V8)
- Reader Dataset** C:\FMEData\Data\CommunityPlan\CommunityPlanMap2012.dgn
- Reader Parameters** 'Group Elements By': 'Level Names'



- Writer Format** Esri Geodatabase (File Geodatabase API)
- Writer Dataset** C:\FMEData\Output\DesktopTraining\PlanMap.gdb

2) Choose Feature Types

When prompted, select Rail, Rivers, and Roads as the feature types to add to the workspace.



The new workspace then has reader feature types for Roads, Rivers and Rail, although there will be extra feature types on the writer and *GeometryFilter* transformers to avoid the situation of multiple geometry types per output table.

3) Run Workspace

Save and then run the workspace. It should run to completion without error.

Now change the Writer parameter “Overwrite Geodatabase” to Yes, so that future runs will not just keep adding extra copies of the data.

4) Change the Reader Dataset

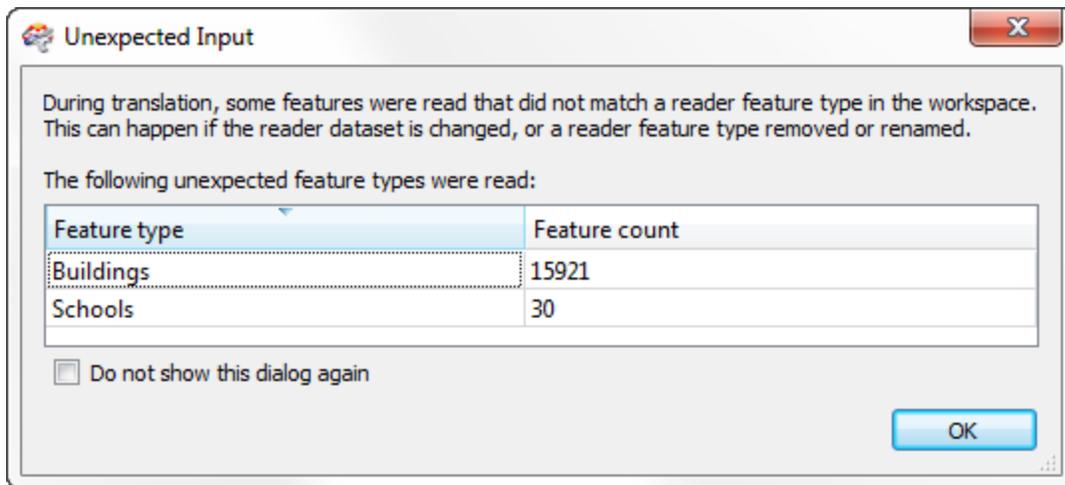
The Planning Department has called about a mistake in the translation. Apparently they have a new base map for 2013 and want to use this instead of the 2012 version.

New Reader Dataset: *C:\FMEData\Data\CommunityPlan\CommunityPlanMap2013.dgn*

In the Navigator window locate the MicroStation reader dataset parameter. Double-click it to edit it and change it to point to the new dataset *CommunityPlanBasemap2013.dgn*.

Save the workspace and run the translation.

The translation is again successful with features read by the (DGN) Reader and written by the (Geodatabase) Writer to the destination.



However... a popup dialog and the log window both report that certain feature types were read but filtered out.

This message must mean that the 2013 dataset has features types unknown in the 2012 data.

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.

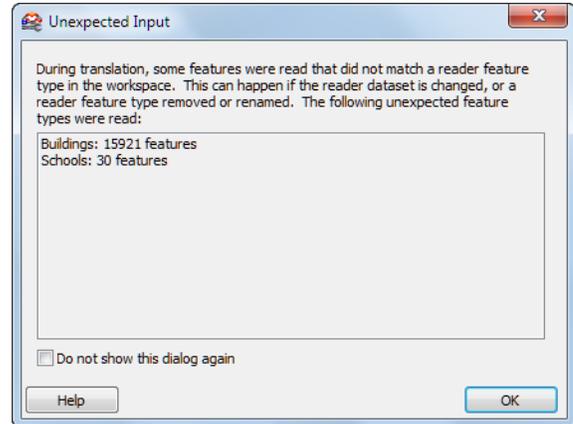
```
|STATS |Unexpected Input Remover(TestFactory): Tested 4 input features -- 0 features passed, 4 features failed.
```

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 example 2. 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 individually, along with a count of the affected features.

Also notice the checkbox that allows this report to be turned off in future translations.



It's important to understand that the feature types in a dataset do not necessarily have to be the same feature types as defined in the workspace.

For example, a user may purposely leave a feature type out of a workspace if that layer of data is not required in the translation.

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 Folder-Based Datasets

It's easy to understand what happens with file based datasets like in the previous example.

The original file had a set of layers, and now the newer file has a different set of layers.

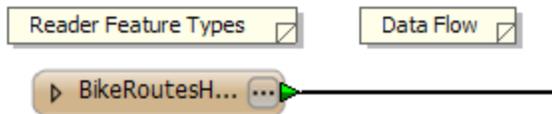
Folder-based datasets can be a little more confusing, but replace "file" with "folder" and "layer" with "file" in the sentence above and it might become clearer:

The original folder had a set of files, and now the newer folder has a different set of files.

To go into a little more detail, the key to understanding folder-based datasets is found in this table shown earlier in this chapter.

Dataset Type	<i>Folder</i>
Dataset Name	<i>Name of folder</i>
Feature Type	<i>File within folder</i>
Feature Type Name	<i>Name of file</i>
Example Format	<i>Esri Shape</i>

In a folder dataset, the name of a feature type added to a workspace is taken from the name of the matching file.



For example, if a user chooses to read BikeRoutesH.shp as a source file, FME creates a feature type called BikeRoutesH.

Therefore changing the source to a differently named file (e.g. BikeRoutesM) guarantees the Unexpected Input Remove will be invoked, because the feature type defined in the

workspace (BikeRoutesH) no longer matches the data being read.

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

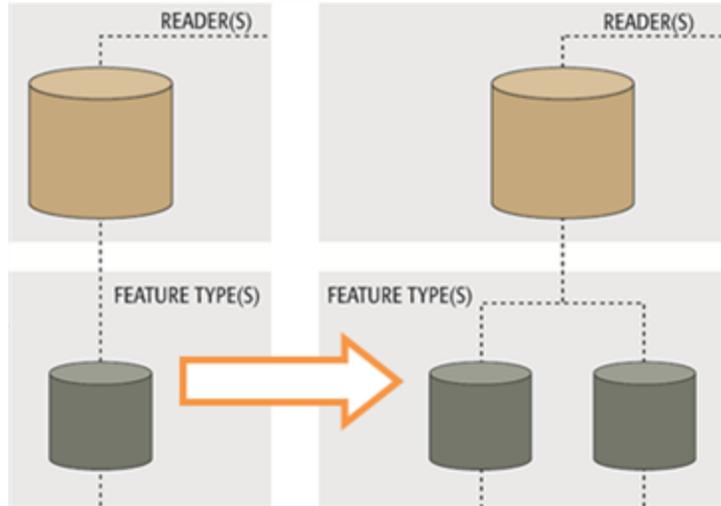
The Import Feature Type tool will take the schema definition of a selected dataset and add it to the workspace. By adding in the missing feature types to the schema, features of that type will be allowed to pass into the workspace.

In the previous example the types Buildings and Schools were missing from the workspace.

To add them simply requires their introduction from the new source dataset.

Once in the workspace the data will now be read and accepted as having a matching feature type.

Of course, reading is only one part of a translation. To actually write the data also requires that the same import function takes place on the writer.



Professor Spatial says...

'It really helps to think of a schema as an object in its own right, related to, but not restricted to, a particular format. Therefore it's possible to import a schema from one dataset, to use in a reader or writer of an entirely different format.'



Example 20: CAD to GIS	
Scenario	FME user; City of Interopolis, Planning Department
Data	Community Map (MicroStation DGN, Esri File Geodatabase)
Overall Goal	Convert the community map data to File Geodatabase
Demonstrates	Reading datasets with different feature types
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example20Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example20Complete.fmw

The previous example showed that a workspace created to convert the 2012 basemap for the Community Plan was not suitable to convert the 2013 basemap data. The workspace fails because it lacks new feature types that have been added for the 2013 update.

This example fixes that problem by importing those feature types.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 19.

Alternatively you can open the 'begin' workspace listed above.

For the sake of clarity, the "begin" workspace has been cleaned up, with excess feature types and GeometryFilter transformers removed.

2) Add Reader Feature Types

Select **Readers > Import Feature Types** on the menu bar. In the dialog set the dataset as follows (it should already be set to this):

Reader Format Bentley MicroStation Design (V8)
Reader Dataset C:\FMEData\Data\CommunityPlan\CommunityPlanMap2013.dgn
Parameters 'Group Elements By': 'Level Names'

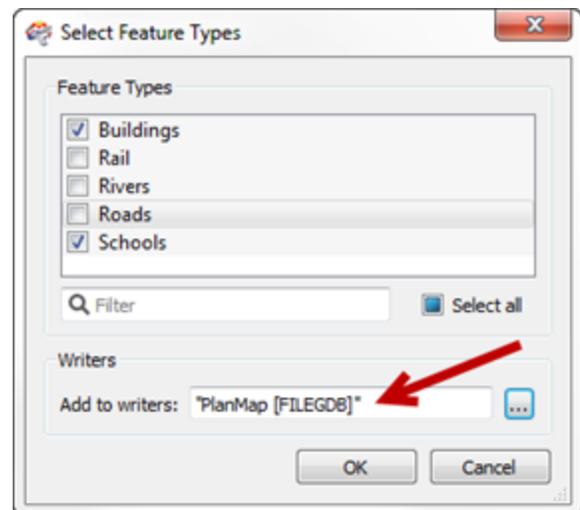
A dialog now opens to choose feature types to import.

Deselect Rail, Rivers, and Roads.

Select the File Geodatabase Writer for the imported types to be added to that as well.

Click **OK**.

FME will read the schema of the remaining Feature Types (Buildings and Schools) and add them to both the reader and the writer.



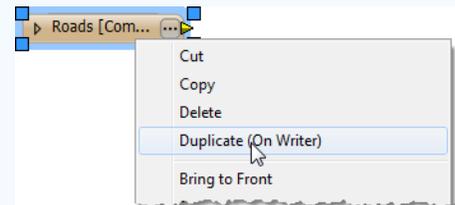


This method (using “Also add to writer”) is the simplest way to import the same set of feature types to both the reader and the writer.

Of course, sometimes you won’t want to import the same on each side of the translation.

To handle that scenario, there are two other ways to import writer feature types.

1. Use Writers > Import Feature Types on the menubar. This has the same effect on the writer as importing types on a reader.
2. Right-click a reader feature type on the canvas, and choose the option “Duplicate (on Writer). This will copy a feature type from reader to writer, and automatically connect it.



3) Set Permitted Geometry Types

Now open up the Feature Type Properties dialogs for the new Schools and Buildings objects, and ensure both are set to allow geodb_polygon (they will probably default to geodb_point)

4) Run the Translation Again

Save the workspace again and rerun the translation.

Check the Log to prove that all reader features have been converted.

Once you are happy, it’s time to send the data to the Planning Department.

This example illustrates one method of how a workspace that was created to process one dataset can be changed to process other datasets of the same format.



Q) Professor Spatial, I have two completely different datasets (City Parks and Polluted Sites) unrelated except for being the same format.

If I want to read them into the same workspace, does each need a separate reader?



A) They don’t necessarily need separate readers, because you can add all the feature types from the two datasets to the same reader.

However, look at the Reader Parameters for that format of data.

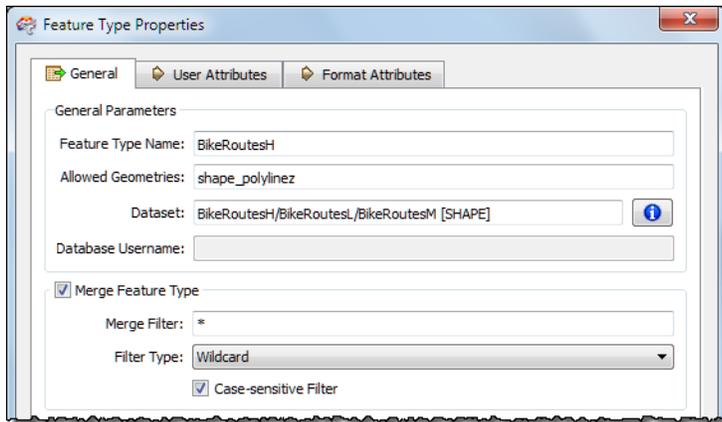
If there is a reader parameter that needs to be different for each dataset, then you will need to create two readers, one for each dataset, and change the parameter on each as necessary.

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.



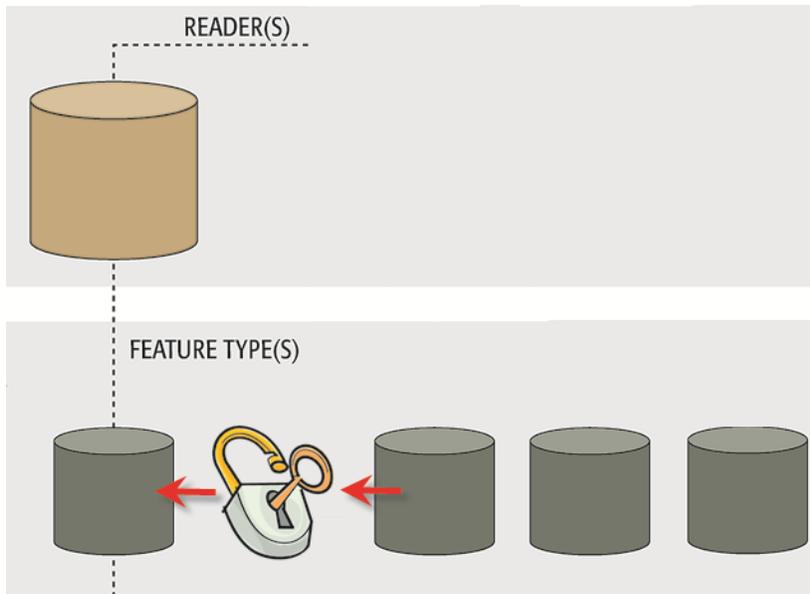
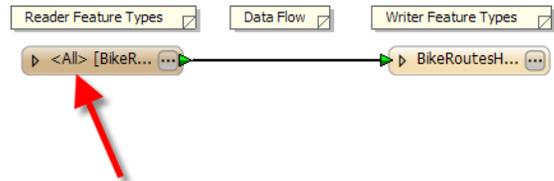
The first action is to select a feature type through which to merge the data and open its Feature Type Properties dialog.

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.

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.



Example 21: GIS Translation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Bike Routes (Esri Shape, Adobe Geospatial PDF)
Overall Goal	Read bike route map data
Demonstrates	Merge Parameters
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example21Complete.fmw

The city’s Traffic Safety Committee would like some data in a Geospatial PDF file. It’s a simple translation from the source Shape dataset. Can you oblige?

1) Create a Workspace

Start Workbench if necessary. Create a workspace with the following parameters:

Reader Format	Esri Shape
Reader Dataset	<u>C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesH.shp</u>
Writer Format	Adobe Geospatial PDF
Writer Dataset	<u>C:\FMEData\Output\DesktopTraining\BikeRoutes.pdf</u>

You might have noticed that the bike routes folder contained three different files. Since Esri Shape is a folder-based dataset, these will represent different layers in the data.

Which do you think is the best method to read each of these files?

- Three separate workspaces
- Three Shape readers in the same workspace
- Import Feature Type definitions to the same reader
- Set a Merge Filter on the same feature type

Let’s examine the options one by one:

- Three separate workspaces
This is more than a little ridiculous! You should never need to do this.
- Three Shape readers in the same workspace
This is the best method only when you need to apply different reader parameters to each file. If the reader parameters can all be the same, then you can use the same reader.
- Import Feature Type definitions to the same reader
This is the best method when each feature type has a different attribute schema. If they all use the same set of attributes, there is no need to separate them. How can you check?
- Set a Merge Filter on the same feature type

This is the best method when all feature types have the same attribute schema, can be read with the same set of parameters, and can be merged into one stream of data.

2) Choose Files

In the Navigator window, locate the Source Files parameter for the Shape reader.

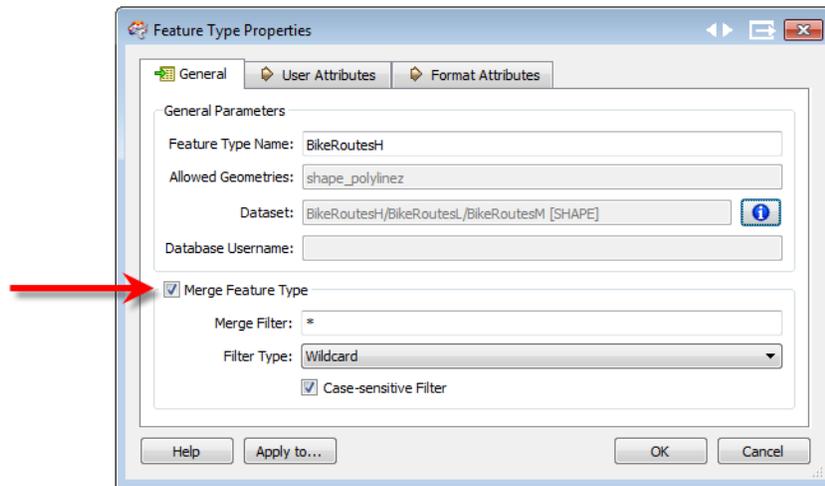
Double-click it to edit the parameter, click the selection button, and choose all three Shape files in the bike routes folder/dataset.

If you run the translation now then you should get the Unexpected Input warning, because the workspace is not yet set up to handle all three incoming feature types.

3) Set Merge Filter

In this case a Merge Filter is a perfectly acceptable option.

Open the single reader Feature Type Properties dialog. Set a merge filter to allow the other cycle routes to pass. Click OK to accept the changes.



4) Run Workspace

Run the workspace. You should find all three Esri Shape files are read and written.



Doctor Workbench says...

'You can probably see one drawback of this method: all of the features are merged into a single bloodstream – sorry, data stream – in the workspace. If I want unique output feature types I must separate the data again.'

In this case it's not too much of a problem. The "H", "M", and "L" parts of the file names are reflected in the value of the USE_RATING attribute.'



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.”

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?

Discuss this issue with a fellow student. Examine the workspace for example 21 to see if there is anything you can do to resolve any merge filter disadvantages.

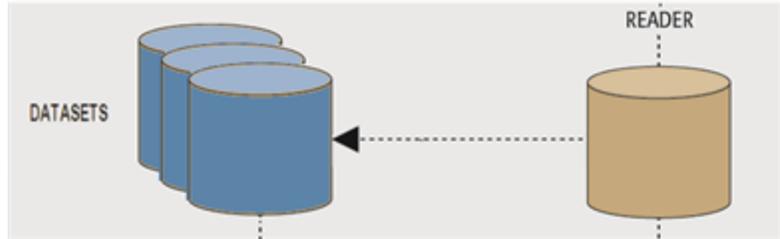
Multiple Dataset Translations



A translation that reads multiple datasets is no more difficult to set up than a workspace reading a single dataset.

Did you notice in the previous diagram that any Reader in Workbench may read multiple datasets?

This is an important piece of functionality for many users.

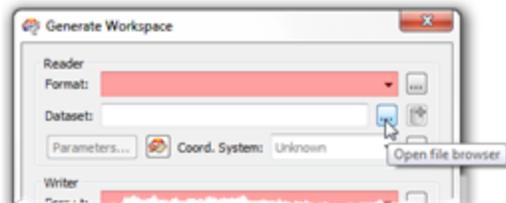


Multiple Reader Datasets

Several tools in FME allow a user to select multiple source datasets, either during the creation of a workspace or by editing an already existing workspace.

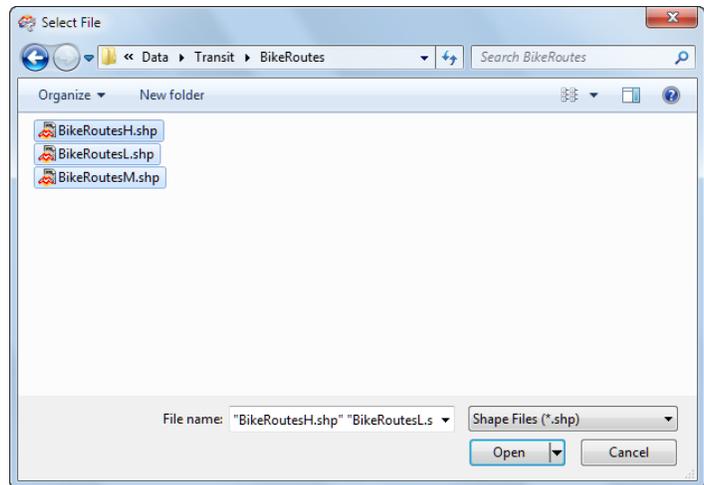
File Browser

Although the file browser is commonly used to select single datasets, it can also be used to select multiple datasets when they are all stored in the same location.



Access the file browser tool from the Generate Workspace dialog by clicking the Browse button.

Select any or all of the datasets required for the translation.





OK, you got me!

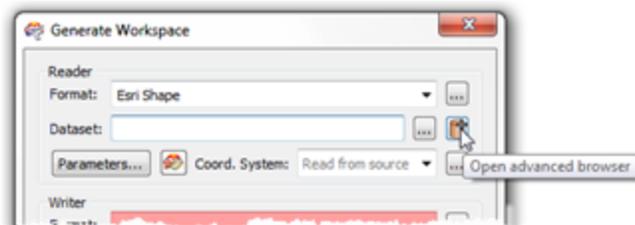
If you select Esri Shape files like this user you are not selecting datasets, but really selecting feature types in a folder dataset.

It's a minor point though; don't get too hung up on the difference in this case.

Advanced Browser

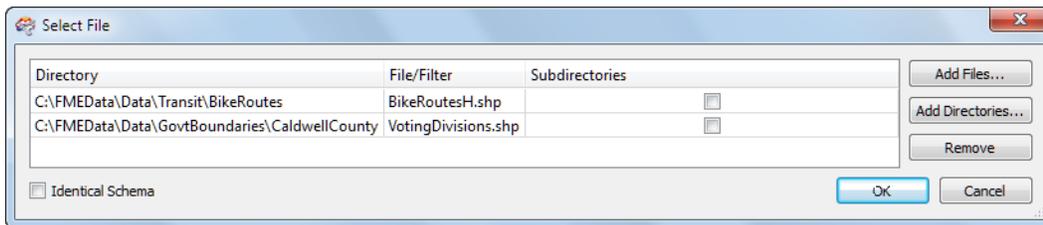
The Advanced Browser – sometimes referred to as ‘the Swizzler’ – is particularly useful when all of the source datasets don't reside in the same folder.

All datasets to be selected must be the same format, and that format must be defined before the advanced browser can be used. The button will not activate until a format is set.

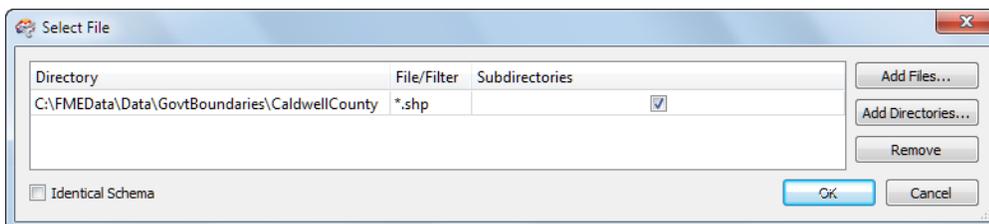


Access the advanced file browser tool from the Generate Workspace dialog by clicking the Advanced Browser button.

The Advanced Dataset Manager dialog allows the user to select datasets that are stored in different directories. Again, for folder-based datasets the user is selecting ‘feature types’ and not datasets; for example here two Shape files (feature types in different datasets) are selected.



Alternatively, a user might select a folder full of datasets, and additionally choose all subfolders too; for example here all the Shape files relating to government boundaries.





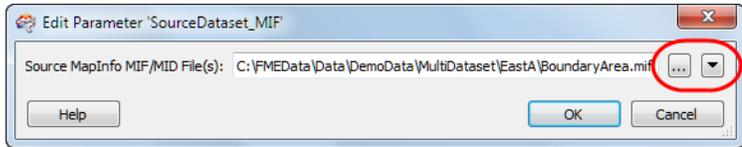
Defining a folder of data is actually a dynamic process, repeated when the workspace is run. i.e. when executed, the workspace will re-scan the selected folder for data to translate.

Therefore a user could run a translation on a folder that has data added and removed on a regular basis. Of course, that user would want to be careful not to trip on “Unexpected Input”.

Workspace Navigator

Multiple files can also be selected when adjusting the dataset parameter in an existing workspace.

Source dataset parameters always have the option of either the standard or advanced file browser.



Of course, if a parameter is changed in this way, it’s important to consider the possible effects of the Unexpected Input Remover!



Example 22: Multi-Feature Type Reading	
Scenario	FME user; City of Interopolis, Planning Department
Data	Properties (MapInfo MIF/MID)
Overall Goal	Translate a set of MapInfo MIF files to KML
Demonstrates	Multi-source reading
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example22Complete.fmw

A set of properties data needs to be translated to KML. The data is tiled into separate files.

1) Inspect the Source Data

In Windows Explorer, browser to C:\FMEData\Data\Properties and inspect the data.

It is a series of files in MapInfo MIF format stored within a single folder.

Can you tell what type of dataset this is?

File	
Folder	
Database	
Web	

Can you tell what object each file is?

Dataset	
Feature Type	
Feature	

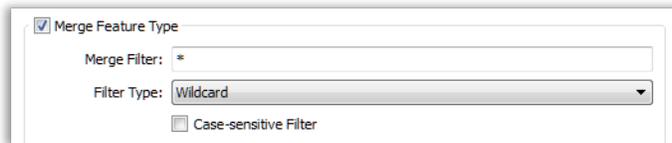
2) Create Workspace

Start Workbench if necessary. Create a workspace with the following parameters:

Reader Format	MapInfo MIF/MID
Reader Dataset	<u>C:\FMEData\Data\Properties\parcel_K24.mif</u>
Writer Format	Google Earth KML
Writer Dataset	C:\FMEData\Output\DesktopTraining\Properties.kml

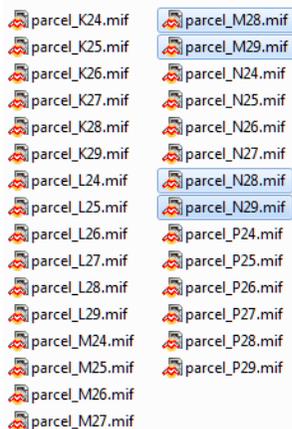
3) Set Merge Parameters

Open the reader feature type properties dialog and set the merge parameters to allow all feature types to pass.



4) Select Multiple Files

Locate the Source MapInfo parameter in the Navigator window.



Double-click it to edit it and use the File Browser tool to select four or five of the MapInfo MIF files in the properties folder.

Note: *Selecting all the files will produce a KML dataset that is very slow to load in Google Earth.*

5) Run Workspace

Save and run the workspace. Inspect the output KML using Google Earth (if available).

The workspace has run correctly, but there are some important questions to consider.

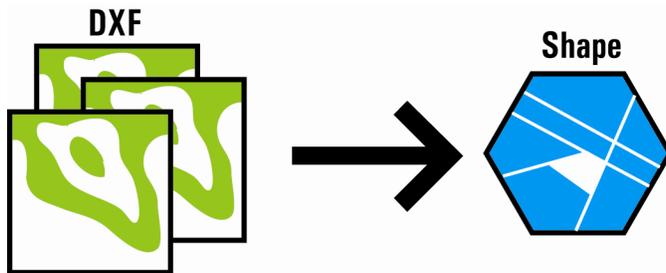
- The workspace was initially created with only a single file selected. Why?
- The output was a single KML file (rather than multiple files). Why?

The technique of selecting a single file, setting a merge filter, and then selecting further files is something that will be improved upon in the Dynamic Workspaces section of this chapter.

Multiple Writer Datasets

Although a reader can read any number of datasets, a writer dataset parameter defines a single dataset, so the default action is a merging of multiple sources into a single output.

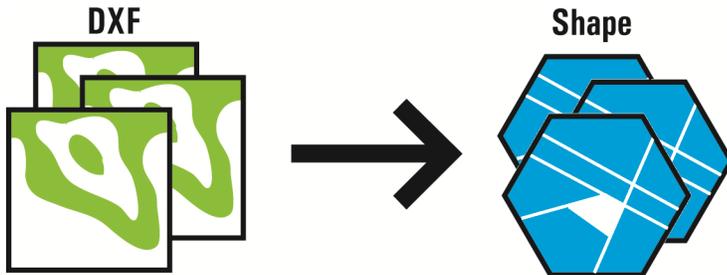
For example, reading a set of three DXF datasets and writing them to Shape will, by default, create a single Shape dataset.



To get multiple output datasets – in this case one for each input – requires either a Dataset Fanout or a batch processing technique such as *Batch Deploy*.

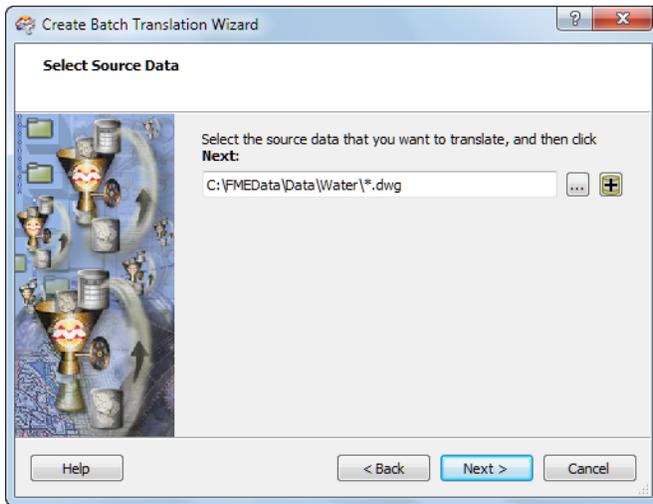
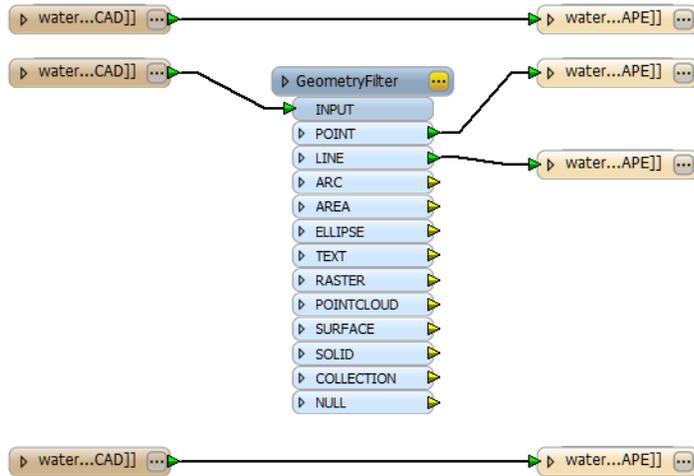
Batch Deploy

Batch Deploy is a tool in FME Workbench for reading multiple source datasets in a way that creates an equivalent destination dataset for each.



Batch Deploy is set up using a wizard accessed by clicking **File > Batch Deploy** on the menubar.

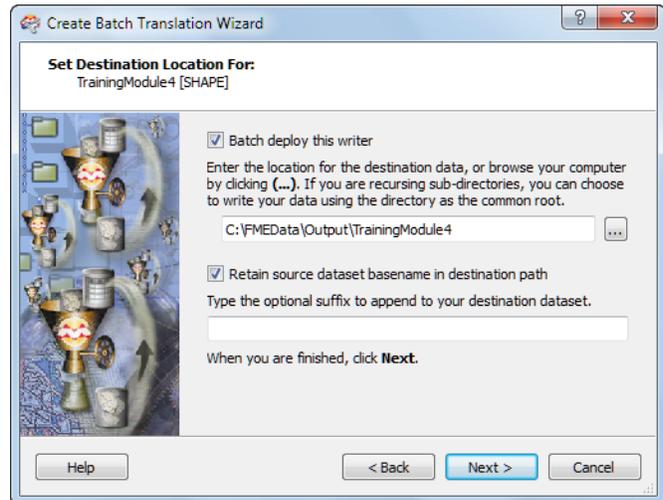
To use Batch Deploy, firstly the user sets up a translation as per normal; here from an AutoCAD DWG file to an Esri Shape dataset:

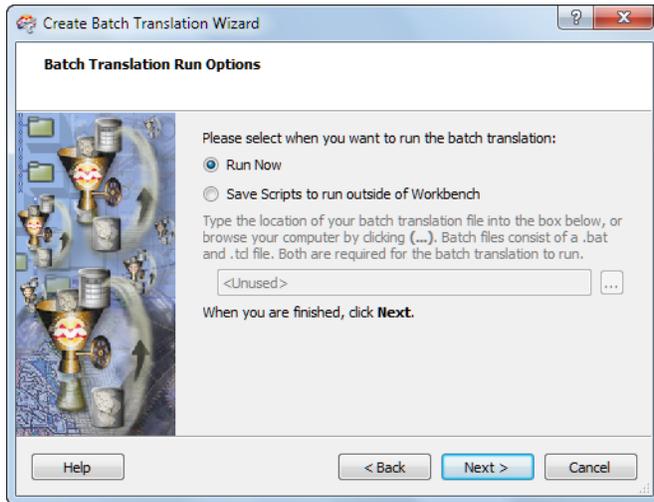


After selecting **File > Batch Deploy**, the user selects all of the AutoCAD DWG files in the Water folder.

Now they decide whether to apply a batching process to the Shape writer.

In this case the answer is obviously yes, but this parameter is helpful where a workspace has several writers, and not all of them need a batched output.

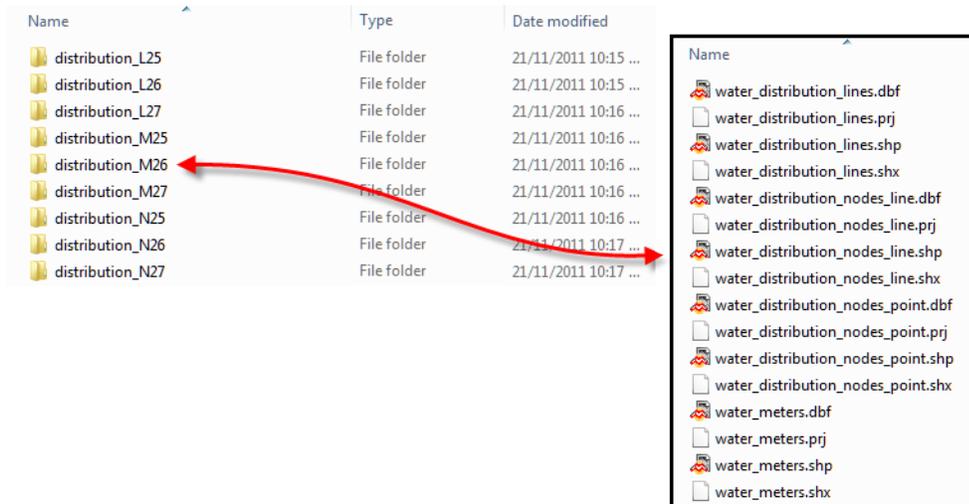




The final decision is whether to run the translation now, or save the batch process as a script to run outside of Workbench.

In this case the user selects “Run Now”.

The result is a new Shape dataset (folder) for every input DWG dataset (file), each of which contains a feature type for the different DWG layers.



Example 23: GIS Multi-Dataset Translation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Parcel Data (AutoCAD DWG)
Overall Goal	Batch Translate a set of parcel data from DWG to KML
Demonstrates	Batch Deploy
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example23Complete.fmw

Another set of boundaries now needs to be translated to KML. This data is similarly tiled.

1) Inspect the Source Data

In Windows Explorer, browse to C:\FMEData\Data\DemoData\ParcelData and inspect the data.

The files of interest are all DWG (AutoCAD Drawing) files.

Can you tell what type of dataset this is?

File	
Folder	
Database	
Web	

Can you tell what object each file is?

Dataset	
Feature Type	
Feature	

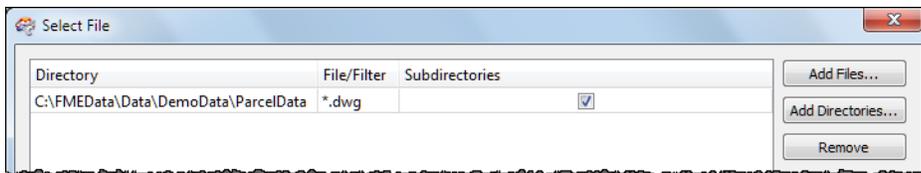
2) Create Workspace

Start Workbench if necessary. Create a workspace with the following parameters:

Reader Format Autodesk AutoCAD DWG/DXF
Reader Dataset C:\FMEData\Data\DemoData\ParcelData***.dwg

This is created by setting the source format first, then clicking the button to open the Advanced File Browser. In the advanced dataset manager dialog choose Add Directories and select C:\FMEData\Data\DemoData\ParcelData. Then click the check box marked 'Subdirectories'.

Remove the entry for *.dxf files. This won't be needed.

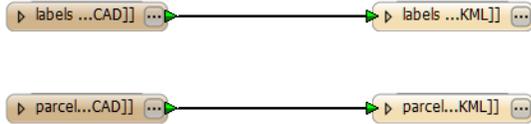


Parameters Group Entities By – Attribute Schema

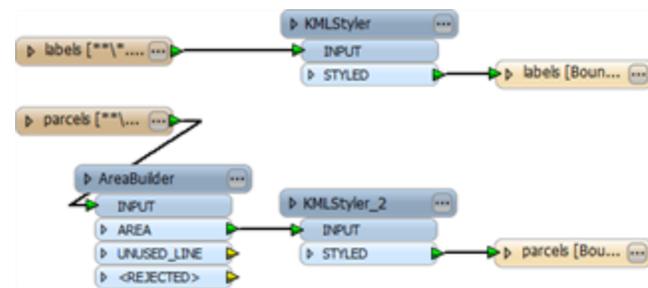


Writer Format Google Earth KML
Writer Dataset C:\FMEData\Data\DemoData\ParcelData***.dwg

Yes, this time we'll create KMZ files, which you can do by simply using a kmz extension to the output dataset. Once you click **OK** and the workspace is created it will look like this:



Notice how, even though each file (dataset) contains a layer called labels, and another called parcels, there is only a single feature type to represent each of them.



3) Add Transformers

Add an *AreaBuilder* transformer to turn the parcel data from lines into areas.

Add some *KMLStyler* transformers (one per stream) to give each feature type a distinct symbology in Google Earth.

Hint: Add an icon for the label features.

4) Run Workspace

Save and run the workspace. It will take about 30 seconds.

Can you tell how many source datasets were read?

Notice that there is just a single KML dataset produced by the translation. Is this what you expected as the output from this workspace?

Open the output and inspect it in Google Earth.



Also check the FME log for “peak process memory”.

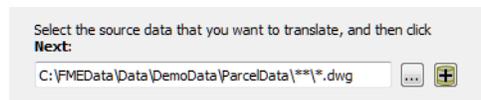
This figure tells us how much memory was used in the FME translation, and it will be interesting to compare it to the number obtained when Batch Deploy is applied to the workspace.

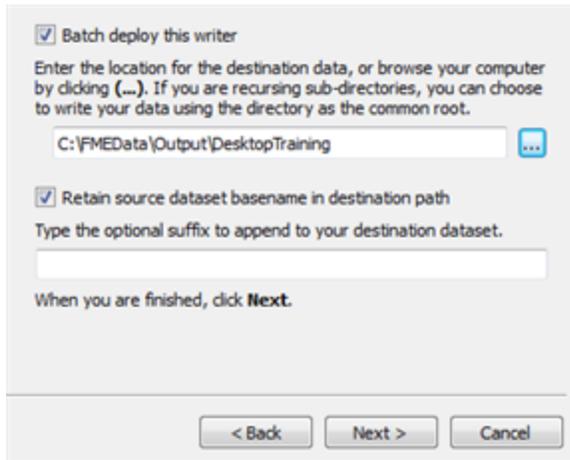
```
Translation was SUCCESSFUL with 0 warning(s) (23643 feature(s) output)
FME Session Duration: 27.3 seconds. (CPU: 22.2s user, 1.6s system)
END - ProcessID: 2992, peak process memory usage: 230684 kB, current process memory usage: 174656 kB.
Translation was SUCCESSFUL
```

5) Batch Deploy Workspace

Select **File > Batch Deploy** from the menubar.

Click **Next** on the first dialog of the wizard. The source parameter should already be correct so again click **Next**.





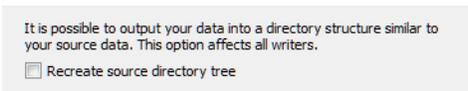
The destination parameters are not quite right.

The output should be a folder, not a file. So change the default .. \DesktopTraining\Boundaries.kmz to simply ..\DesktopTraining

Then just click **Next**.

NB: If the output path has spaces in it, then this parameter will have quotation marks around it. Don't remove them!

Also leave the 'Recreate Source' option unchecked (and just click **Next**)



6) Run Batch Deploy

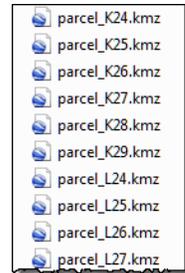
Choose the option to 'Run Now', click **Next** and then complete the batch deploy wizard by clicking **Finish**.

The translation will be carried out.

The results of the translation are a series of KMZ files.

Is this what you would have expected Batch Deploy to produce?

Also check the log file again. Can you see what is different about the number of translations, and also the peak memory usage?



```
Translation was SUCCESSFUL with 1 warning(s) (29 feature(s) output)
FME Session Duration: 0.6 seconds. (CPU: 0.2s user, 0.1s system)
END - ProcessID: 4756, peak process memory usage: 90052 kB, current process memory usage: 81448 kB.
```



7) Advanced Q+A

Check closely the polygons created in both translations, and consider the nature of a batch process. Are there any differences between the two outputs? Are the results surprising and, if so, can you determine why?

Also consider which of the outputs is more suitable for KML format and use in Google Earth.

Advanced Format Controls



A number of advanced control methods are available to help manage different datasets and feature types within a workspace.

Feature Types to Read

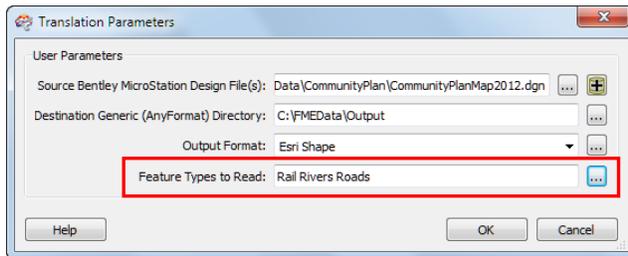
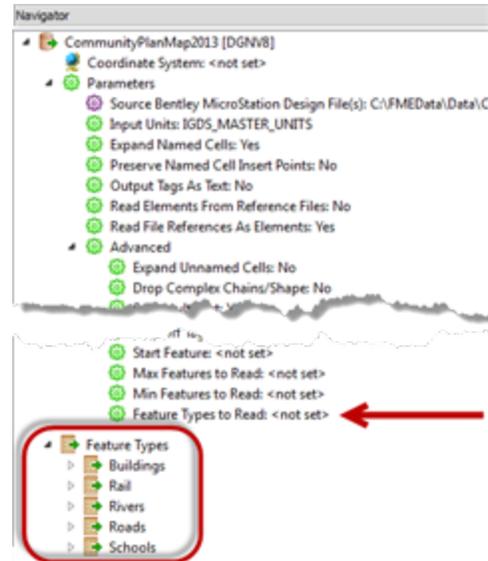
Feature Types to Read is a parameter common to all readers. It acts as a way to give user control to the filtering of feature types read into the workspace.

Of course it is actually possible to delete or disable superfluous feature types before running a workspace; but Feature Types to Read is better because it may be published so that the end-user can select which feature types are required, without editing the workspace.

Here a workspace contains five different feature types:

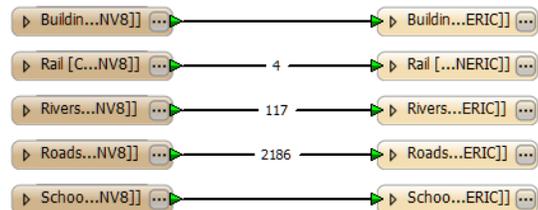
Buildings, Rail, Rivers, Roads, and Schools

...from which the user may choose.

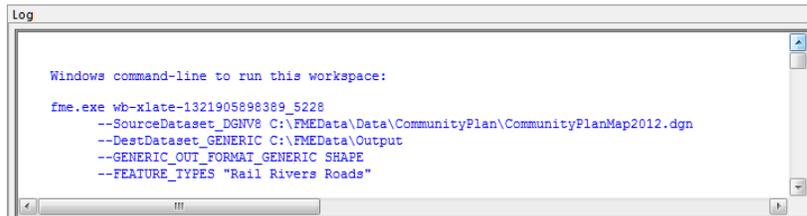


By publishing the Feature Types To Read parameter the user can be prompted for it at run time.

The feature counts show that the workspace has only read features from the selected feature types.



Again, as a publishable parameter, this can also be set on the command line.



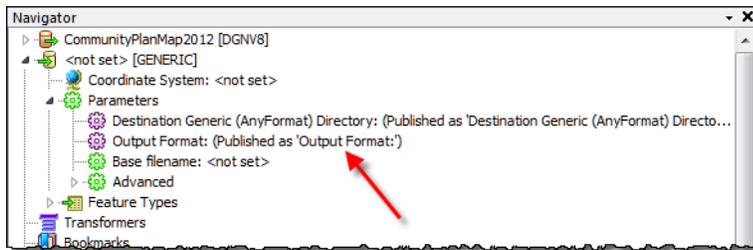
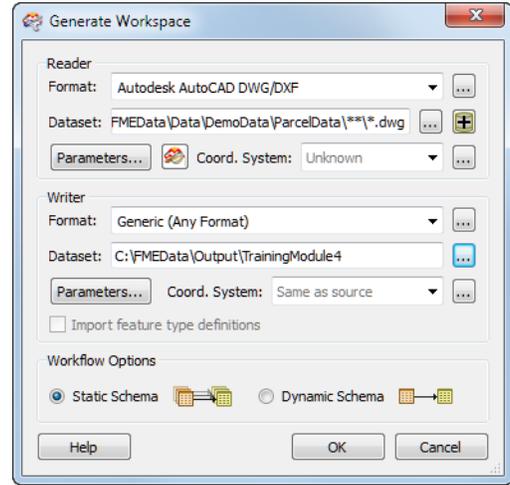
```
Log
Windows command-line to run this workspace:
fme.exe wb-xlate-1321905898389_5228
--SourceDataset_DGNV8 C:\FMEData\Data\CommunityPlan\CommunityPlanMap2012.dgn
--DestDataset_GENERIC C:\FMEData\Output
--GENERIC_OUT_FORMAT_GENERIC SHAPE
--FEATURE_TYPES "Rail Rivers Roads"
```

The Generic Writer

Whereas all other writers are tied to a particular format, the Generic Writer is a component of FME without a specific format.

When a workspace with a generic writer is run, the format of data written is determined by a parameter that can be set on the Navigator.

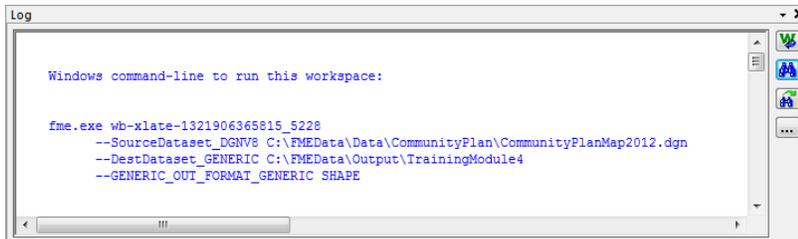
Because the format parameter is publishable, it permits an end-user to choose at runtime which format to write to.



The format parameter is simply one of the standard writer parameters. And can be published in the same way as the destination location.

Like dataset parameters, the Output Format parameter is published automatically, so FME always prompts for the output format.

Note: The destination for this writer is always a directory, even when the selected format is file-based.



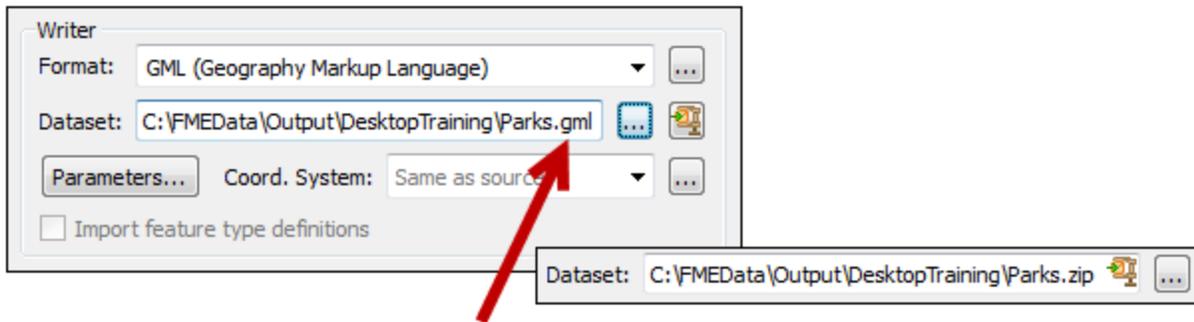
Being publishable, the Generic Writer format can even be set from the command line.



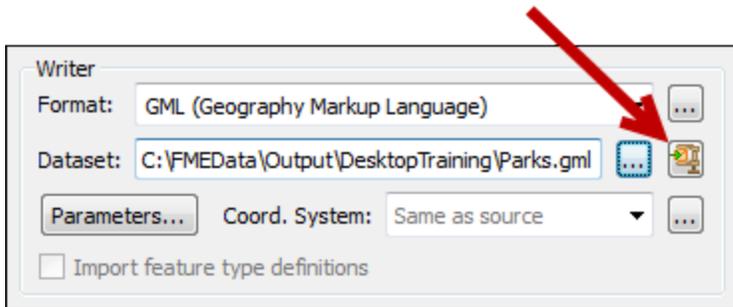
Zip File Writing

As part of the same functionality for reading data from zip files, FME can also write data TO zip files. This is easily done by giving the output dataset a .zip file extension.

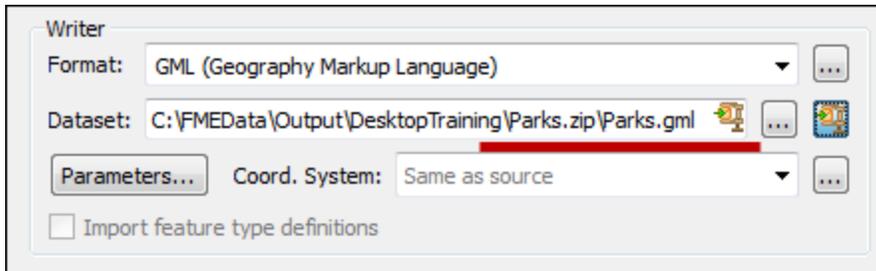
One way to achieve this is to simply change the extension to .zip in the output dataset field:



Another method is to simply click the new “Zip Output” button in the writer dialog:



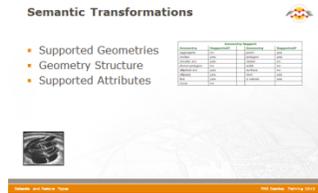
Doing so automatically sets up the output to be written with the given name, but inside of a similarly named zip file:



Although the above images show how to achieve a zipped output in the Generate Workspace dialog, the same functionality is available regardless of where and how you set the output file.



Semantic Transformations



FME is a semantic translator, therefore, it makes decisions regarding the transformations of features according to the format being used.

As a semantic translator, FME will carry out transformations on data to fit the definition and rules of the destination format.

In other words, transformation occurs that is hidden to the user: it's all done with the best of intentions to maintain as much meaning as possible in the data. However, users who aren't always aware of the subtleties of various formats can be surprised by such restructuring.

Supported Geometries

Not all of the geometries that FME supports are actually supported by every format. For example, Esri Shape format does not support arc features and Oracle Spatial does not support text features.

When translating unsupported geometries, FME always attempts to convert the data to a geometry that is supported.

For example, FME will 'stroke' arcs (that is, turn them into line features) and replace text features with a point feature that has an attribute holding the text string content.



Ms. Analyst writes...

'Dear Aunt Interop, I'm dating a format I'm not familiar with. How can I ensure my geometry types are supported?'



Aunt Interop replies...

'Dear Ms. A,

Concerns of this type are perfectly natural. Supported geometries are listed – for all formats – in the FME Readers and Writers Manual under the Quick Facts section.

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	yes
circles	yes	polygon	yes
circular arc	yes	raster	no
donut polygon	no	solid	no
elliptical arc	yes	surface	no
ellipses	yes	text	yes
line	yes	z values	yes
none	no		

Geometry Structure

Sometimes FME is required to transform spatial data because of faults with the geometry or because of structural rules that are required by a particular format.

Some geometry faults are fixed when the data is read (to give a good generic representation for FME use) and some are fixed when the data is written (to give a representation that conforms to the destination format).

For example, there's a rule for Oracle databases that no two vertices on a line or polygon are the same, and FME validates data to make sure it adheres to this rule.

Restrictions

It should be emphasized that the FME processes will not change the overall shape, area, length, or size of a feature. The sorts of faults fixed are not ones that would have this effect.

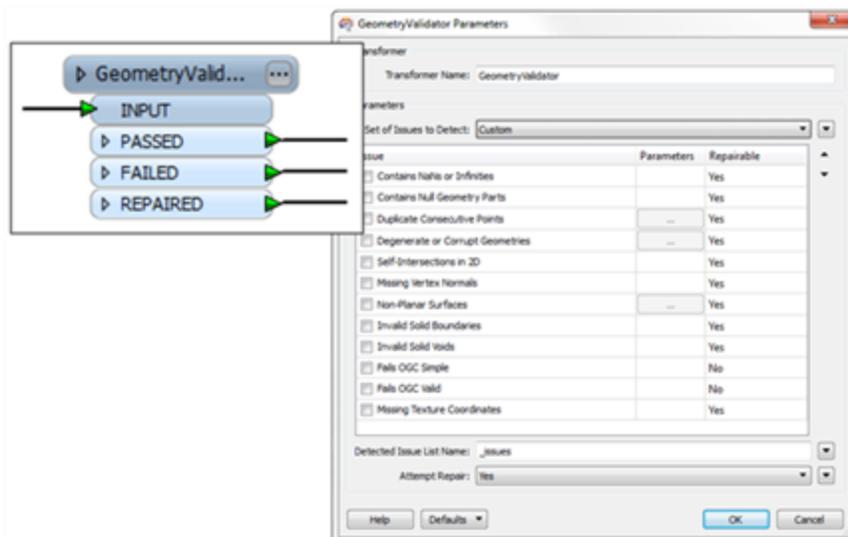
Some example faults that FME might fix automatically include the following:

- Duplicate points on a line feature
- Duplicate line segments on a polygon feature
- Polygon orientation—clockwise/counter-clockwise
- Twisted or figure-of-eight polygons
- Donut polygons where the inner boundary touches the outer boundary at more than a single point

In these cases the only noticeable difference in the data is a change in the number of vertices.



Some faults are just too much for FME to fix automatically, in which case the user needs to apply transformers. The GeometryValidator transformer (new for FME2013) is an excellent all-round transformer for validating and repairing spatial data.

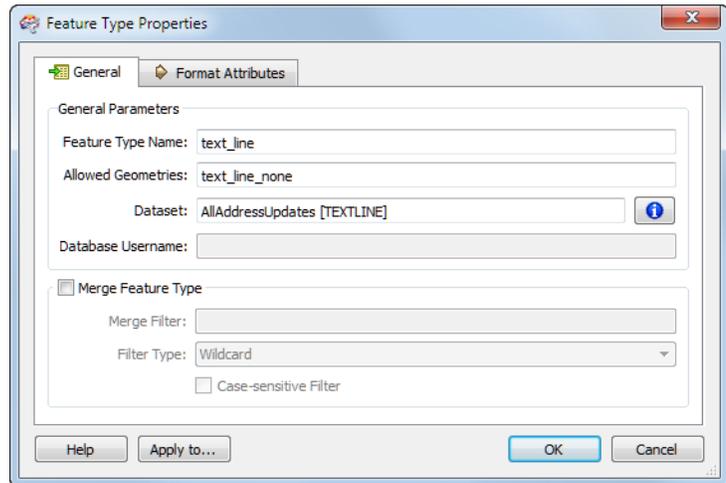


Supported Attributes

Some formats either do not allow attributes or store them outside the dataset.

In these situations, the FME schema does not have user attributes. Therefore, there will be no User Attributes tab in the Feature Type Properties dialog.

TXT (Text) is one example of such as format. Note there is no User Attributes tab in this feature type properties dialog.



Mr. CAD says...

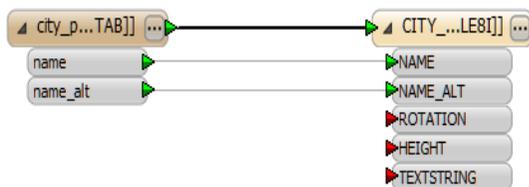
'FME supports destination attributes on DGN (V8) writers by generating and writing the values to MicroStation Tags.'

Attribute Names

Many formats have restrictions on the structure and format of attribute names. Esri Shape format, for example, only permits UPPERCASE names and has a maximum length of ten characters for attribute names.

When creating a workspace, FME converts attribute names to UPPERCASE in the destination schema if needed. In such a case attribute connections cannot be implied so FME generates schema mapping that joins source attributes to the destination.

Note, however, that this is just a starting point. Edits to the workspace may break the FME generated schema mapping and, therefore, require the user to create their own attribute connections.



In this translation to Oracle, FME generated destination attributes with UPPERCASE names and mapped them to the source schema.

If a transformer were inserted manually between source and destination, this schema mapping would be lost.

Attribute Field Size

Some formats carry restrictions on attribute sizes, for example a maximum length of character field or maximum number in a numeric field. Where possible, FME generates a destination schema that reflects any such limitations.



Writing Over-Sized Data

Writing over-sized data is not considered enough to terminate an FME process. Depending on the writer, the data is likely to be truncated to fit or ignored. The log file may carry a warning message when attribute data has been affected by a restrictive destination schema. However, do not rely on the log, but make your own checks on the output if this may be an issue.

Attribute Types

Different formats support different attribute types. Even when two formats support the same attribute type, it's often represented by a different name. In this situation, FME must decide how to map data between one attribute type and another. For example, what is the Oracle equivalent to the MapInfo 'decimal' field?

FME solves the problem by supporting all of these types internally within its enhanced data model. A 'metafile' for the reader format instructs FME how to map its attribute types to the FME model and a second metafile instructs FME how to map its model types to the destination schema.

As an example, here is the attribute type mapping for MapInfo data:

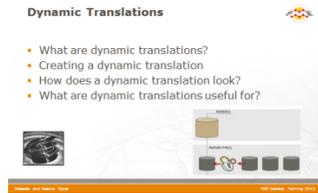
char(width)	fme_char(width)
date	fme_date
decimal(width,decimal)	fme_decimal(width,decimal)
float	fme_real64
integer	fme_int32
logical	fme_boolean
smallint	fme_int16

...and here is the attribute type mapping for Oracle data:

char(width)	fme_char(width)
float	fme_real32
number(width,decimal)	fme_decimal(width,decimal)
double	fme_real64
integer	fme_int32
logical	fme_boolean
smallint	fme_int16

These mappings confirm that a MapInfo 'decimal' type attribute becomes an 'fme_decimal' type in the FME data model, which then becomes a 'number' type attribute when written to Oracle.

Dynamic Translations

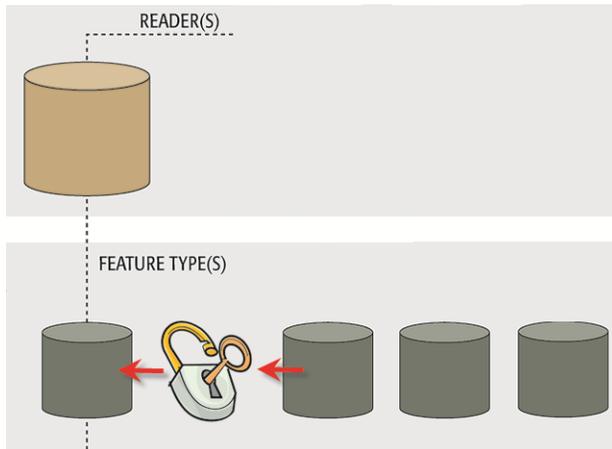


Dynamic Translations are a way to create 'schema-less' workspaces.

What are Dynamic Translations?

In previous chapters of the training, all of the translations have involved a schema being defined within the workspace. In other words, the source and destination schema reflect the structure of the source data (what we have) and the destination data the user requires (what we want).

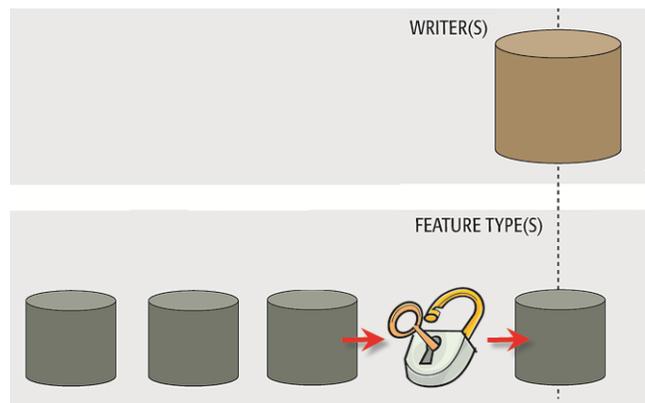
The layout of a dynamic translation does not reflect either the source or destination schema. It's a general layout, which is designed to handle data regardless of what schema is required.



On the reader side of things, a dynamic workspace is very similar to using Merge Parameters; feature types are given free entry to a workspace, regardless of whether they are yet defined in there.

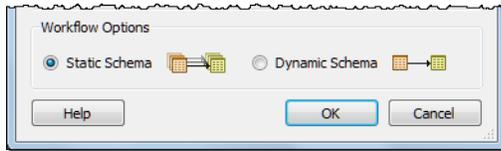
The writer side of a dynamic workspace mimics the reader part; feature types are written to the destination dataset, regardless of whether they have been defined in the workspace.

Additionally, all attributes are also written, regardless of whether they too have been predefined in a writer feature type.



Creating a Dynamic Translation

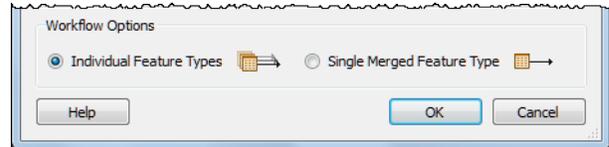
When an author creates a translation using the New Workspace dialog, there are two options for what is called workflow: static schema and dynamic schema.



The Static Schema option is the default for a workspace including schema. Choosing the Dynamic Schema option creates a schema-less workspace with dynamic Readers and Writers.

The Add

Writer dialog also has options for Static and Dynamic schema; so too does the Add Reader dialog, although there they are described a little differently.



You can use any Reader and Writer format in conjunction with a dynamic workflow.

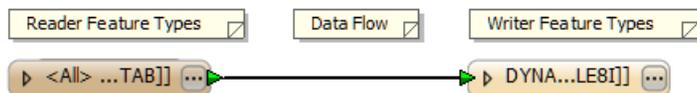
Don't get confused with the Generic Reader and Writer formats. The term "Generic" means "any format", while "Dynamic" means "any schema".

A workspace may be generic, or dynamic - or even both!

In this way it's possible to define individual readers and writers as 'dynamic'.

How Does a Dynamic Translation Look?

Both dynamic readers and dynamic writers each have a single Feature Type, regardless of the schema of the Reader datasets.



Notice that there is only a single feature type, regardless of whether the data is made up of several layers.

Also notice that the sole Reader Feature Type is named '<All>' (which provides a clue to what is happening here) and that the sole Writer Feature Type is named 'DYNAMIC'.

When the workspace is run, all of the source data is read through a single feature type. On the writer side, although there is only one output type, the data will be dynamically divided back into its component layers.

What are Dynamic Translations Useful For?

A dynamic translation has the advantage that it can accept any source data of the specified format and can write any data — regardless of schema.



Example 24: Dynamic Workspaces	
Scenario	FME user; City of Interopolis, Planning Department
Data	Various
Overall Goal	Improve workspaces using dynamic functionality
Demonstrates	Dynamic Workspaces
Starting Workspace	None
Finished Workspaces	C:\FMEData\Workspaces\DesktopManual\Example24aComplete.fmw C:\FMEData\Workspaces\DesktopManual\Example24bComplete.fmw

A few previous examples can now be re-created, this time using dynamic functionality.

1) Repeat Example 19

Repeat "Example 19: CAD to GIS" on page 187.

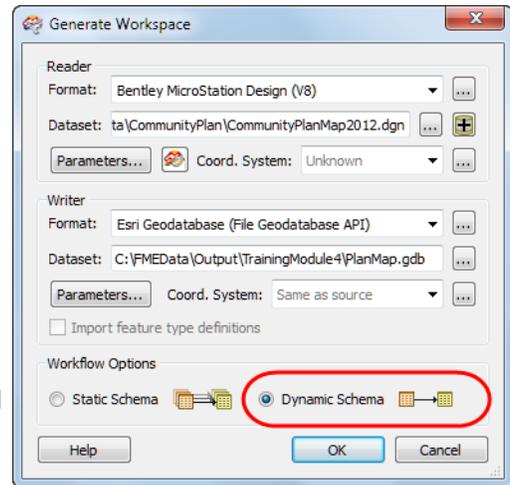
This time in the Generate Workspace dialog, click the Workflow Option 'Dynamic Schema' to create a dynamic workspace.

When the source dataset is changed and the workspace re-run, you should find that the Unexpected Input Remover does not appear as it did before.

That's because the workspace is set up to read any data. You could say it's been designed to expect the unexpected!

You'll also find that – despite there being only a single output Feature Type – the Geodatabase output is still arranged over several layers. This is because the workspace is working dynamically.

Also notice that you don't need to worry about the format's Allowed Geometries.



FME is taking care of this issue dynamically and will create one Feature Type per geometry as and when required. To expand on that, the name of the Feature Type will depend on an advanced setting under Feature Type > Dynamic Properties > Schema Definition > Geometry.

2) Repeat Example 22

Repeat "Example 22: Multi-Feature Type Reading" on page 200.

This time, instead of selecting a single file in the Generate Workspace dialog, select four files, but also choose the Dynamic Schema option.

What you'll find is that a dynamic schema automatically adds the Merge Filter option that was manually set up in example 22. So using a dynamic schema lets a user set up a workspace to read multiple feature types much quicker than would otherwise be the case.

Module Review



This session was designed to increase your knowledge of how FME handles different spatial data formats.

What You Should Have Learned from this Module

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

Theory

- The **Formats** supported by FME vary greatly in their licensing requirements, data types, geometries, and attributes.
- There are several types of **Dataset: File Dataset, Folder Dataset, Database Dataset, and Web Dataset.**
- The **Unexpected Input Remover** filters incoming data if it doesn't match a known feature type defined in the workspace.
- By default, multiple reader datasets create a single writer dataset; to do otherwise requires **batch processing.**
- FME carries out **semantic transformation** on outgoing data, modifying geometries as necessary to ensure they meet the specification of the format being written to.
- **Dynamic Translations** are a way to create schema-less workspaces that will process any incoming data.

FME Skills

- The ability to handle multiple source datasets regardless of feature type or attributes.
- The ability to create workspaces to process multiple reader and writer datasets.
- The ability to create and use dynamic workspaces.

Finding Transformers



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.

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.

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.

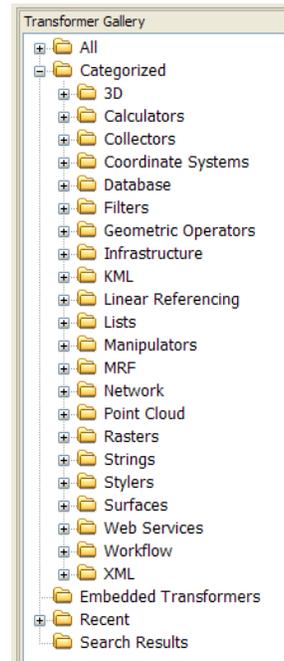
Surfaces: Work with surfaces; for example, create contours.

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



One new calculator transformer for FME 2013 is the VolumeCalculator.

The Transformer Description window is directly linked to the Transformer Gallery: a transformer selected in the gallery triggers help contents to display in the Transformer Description window.

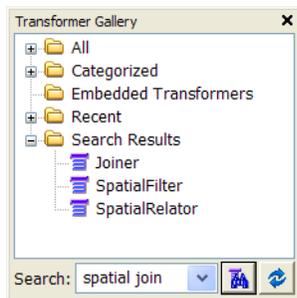
When a search returns multiple transformers then the Transformer Description is shown for the first transformer in the list.

Transformer Searching

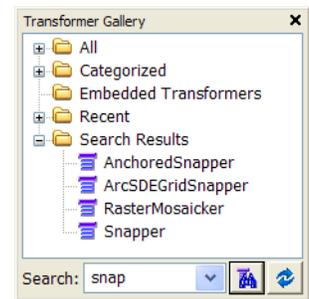
There are search functions in both the transformer gallery and Quick Add dialog.

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).

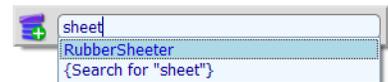
The transformer gallery search searches in both name and description. Therefore a search term may be the exact name of a transformer, or it may be a general keyword referring to functionality in general:



Search terms can either be full or partial words, and may consist of a number of keywords.



Quick Add search-terms can also be full or partial words:



Quick Add does not look in transformer descriptions, so the search term must be the actual name of a transformer. For example, although the Transformer Gallery search will return values for the keyword “clipping”, Quick Add will not

However, Quick Add does also have the ability to transfer the search term entered directly to the Transformer Gallery search.



i.e. clicking on {Search for “clipping”} will search for that term within the Transformer Gallery.



Quick Add results 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.

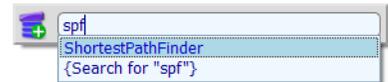


CamelCase

Quick Add also allows the use of CamelCase 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).

Quick Add will find all transformers whose upper case characters match the search term provided.

For example, Quick Add will return the *ShortestPathFinder* when the search term is the initials "spf":



Inserting Transformers



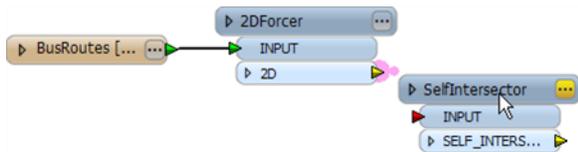
Once you've found the required transformer, there is still the need to place it into the canvas, and connect it into the right position.

Advanced Transformer Drag-and-Insert

The Transformer Drag-and-Insert function has a number of advanced capabilities.

Connecting a Transformer

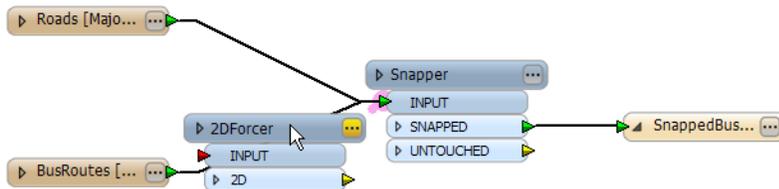
Drag-and-Insert also has the ability to connect a transformer rather than inserting it into an existing connection. This particularly helps when constructing a new workspace from scratch.



Here a user wants to attach a transformer on to the end of the current pipeline. The user drags it onto the final output port to connect it.

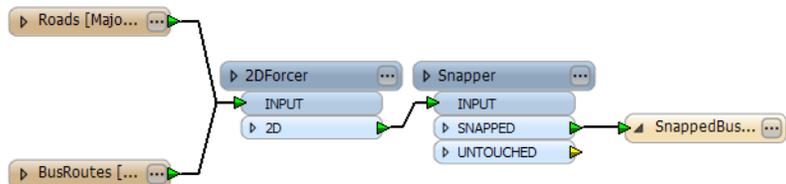
Multiple Connections

Did you know that transformers can be inserted into multiple connections simultaneously by highlighting an input or output port instead of a connection?



Here the user drags the transformer into a position where the Snapper:INPUT port is highlighted.

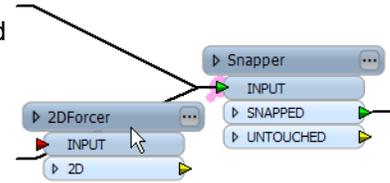
With the mouse released (and a little tidying up) the transformer is dropped into position with two input connections.



Reversing the Insertion Point

When the body of the transformer hides the connection into which it needs to be inserted, then the **Alt** key can be pressed when dragging a transformer. This switches the insertion highlight to the opposite corner of the transformer, clearing the view.

In this example, with the Alt key pressed down, the pink dot switches sides and the view is clearer.

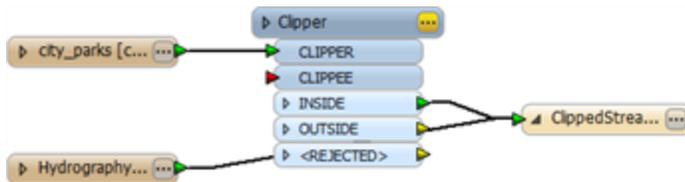
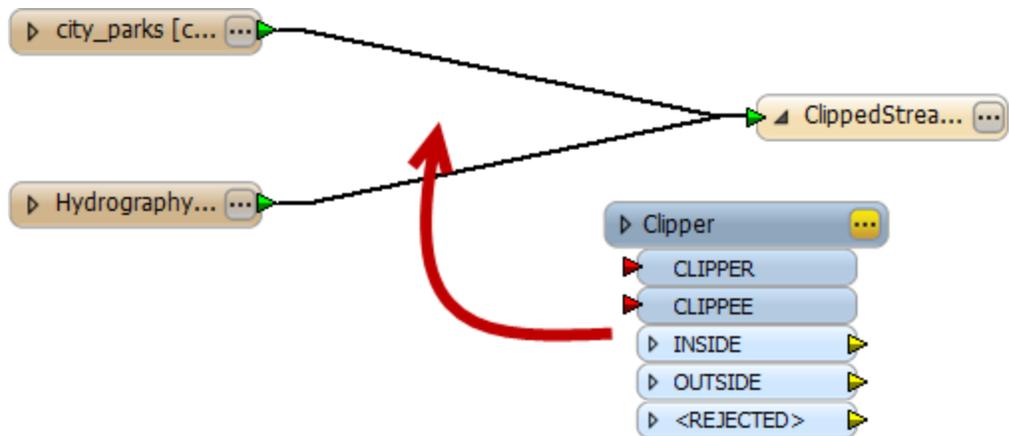


Inserting an Already Inserted Transformer

Dragging a transformer that's already inserted still causes a highlight point to show and highlights connections. The reason for this is that it's possible to connect a single transformer into multiple streams and this method lets you do it with remarkable ease.

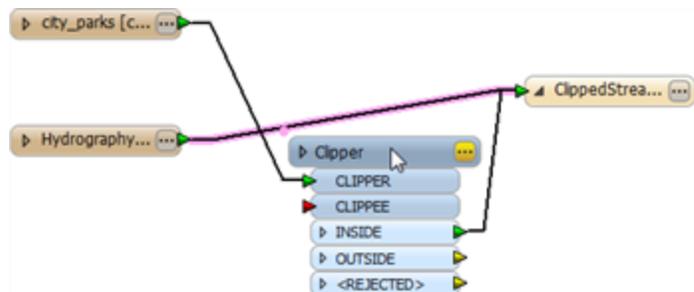
Here a user wants to insert a *Clipper* transformer.

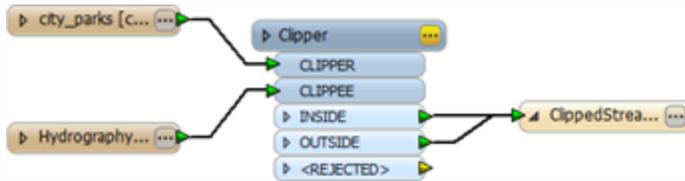
The difference from the previous multiple connection example is that there are multiple input ports requiring separate connections.



The first connection is easy to make using the drag and insert function.

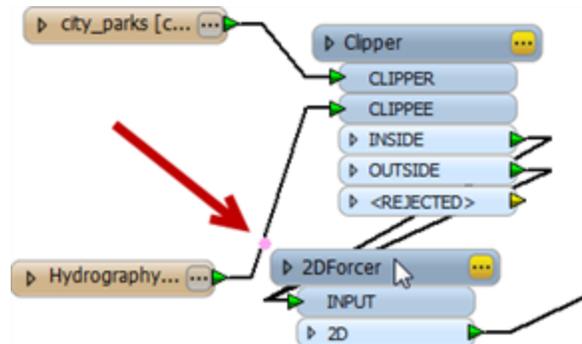
Now the user makes a second insertion; **with the same transformer**, but onto the other link.





And with a little tidying up, the *Clipper* is correctly inserted.

Note: A transformer cannot be inserted in a position that would cause a loop in the workflow. Such invalid connections are not even highlighted.



When a transformer is selected, *Ctrl+E* is a shortcut to toggle between enabled/disabled.
 For FME2013, a new shortcut is *Ctrl+D*, to duplicate the transformer.

Integrated Attribute Construction



FME encourages direct use of attributes by integrating string construction and arithmetic calculations within transformers.

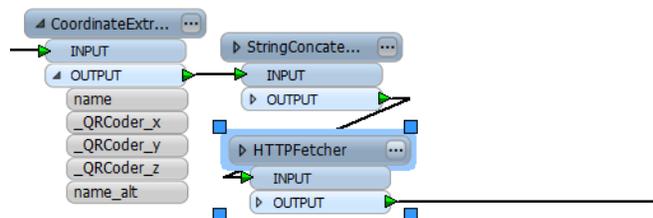
Reducing Workspace Congestion

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.

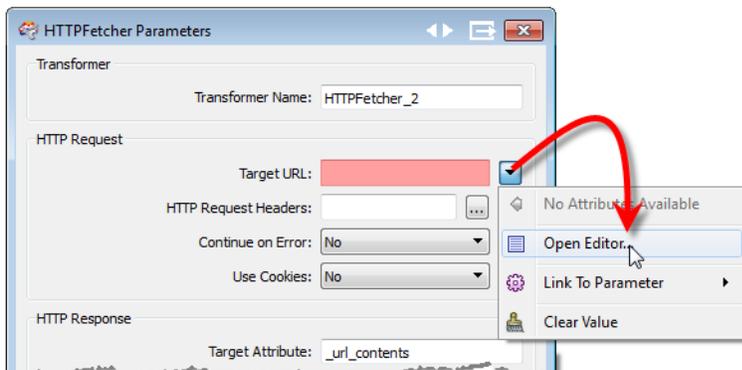
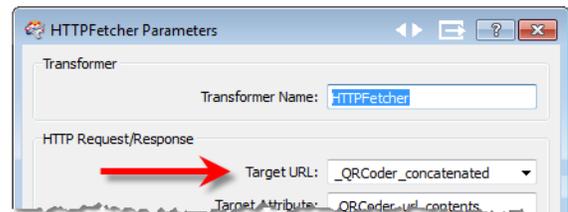
To avoid the need to use excessive numbers of attribute-related transformers, FME embeds attribute handling directly into many other transformers.

In this way the number of transformers required to carry out a single task is reduced.

For example, here a *CoordinateExtractor* and *StringConcatenator* are used to construct a URL that is then used in the *HTTPFetcher*.



However, the same action can also be carried out entirely within the *HTTPFetcher* by using the string building editor. The action of the *CoordinateExtractor* could even be replicated by using the FME functions @XValue and @YValue in the advanced dialog.



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.

Integrated Parameter Handling

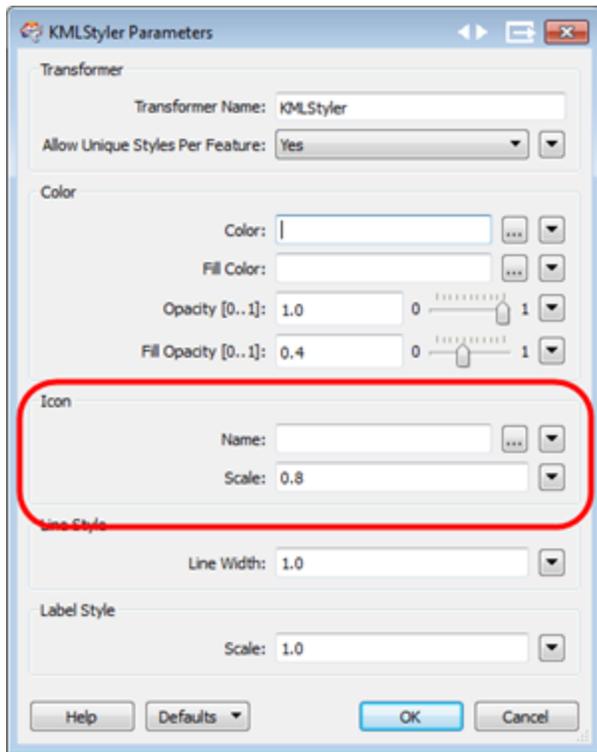


FME encourages direct use of user parameters by integrating related functionality within transformers.

Integrated Parameter Creation

Similarly to attributes, besides being separate transformers, FME integrates parameter-handling functionality directly within all other transformers.

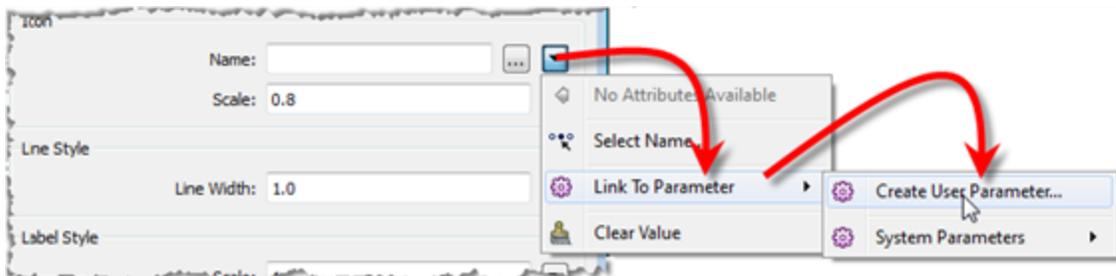
Again, because parameter use is embedded directly within an operational transformer, the number of additional transformers – and use of the Navigator window – is reduced.



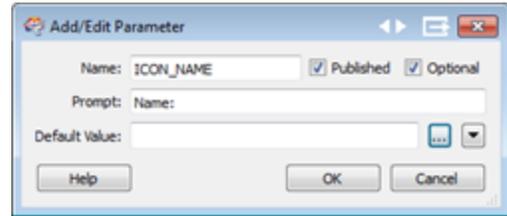
Here a workspace author wishes to prompt the end-user for the icon name in a KMLStyler:



The sequence of actions to set this up is to click on the drop-down arrow, select Link To Parameter, and then Create User Parameter.



This brings up the same Add/Edit Parameter dialog that is available in the Navigator window for setting up user parameters.



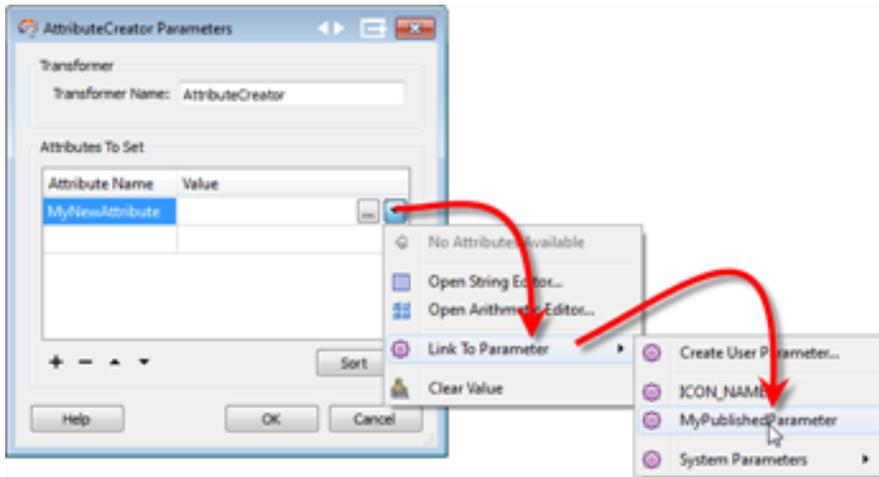
Accepting the dialog automatically links the user parameter to that transformer parameter.

Integrated Parameter Retrieval

To retrieve, and use, a parameter is similarly easy.

To retrieve the value of a user parameter as an attribute, the *AttributeCreator* can be used instead of the *ParameterFetcher*.

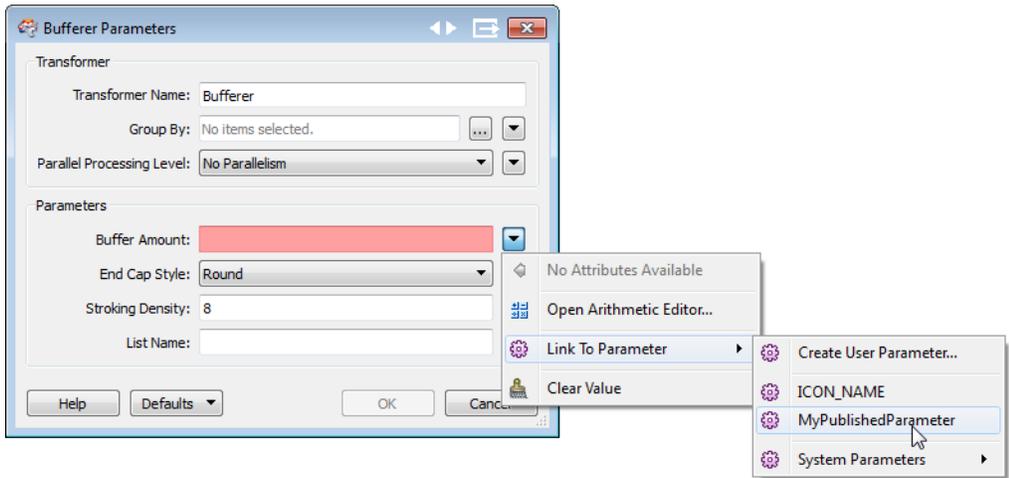
Simply select the Value drop-down arrow, choose Link To Parameter, and then choose the published parameter whose value is to be retrieved:



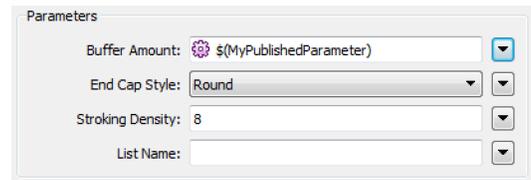
The value of the published parameter is then stored in the defined attribute.

To fetch a parameter for use in another transformer is much the same technique, just selecting whichever transformer parameter the user parameter is to be applied to.

For example, here the author wishes to apply an existing user parameter to a *Bufferer*:



Once more, the parameter dialog is updated to reflect the new scenario:



Integrated List Handling



FME also encourages direct use of those useful – but mysterious – attributes called 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{0}.myAttribute* can contain multiple values as:

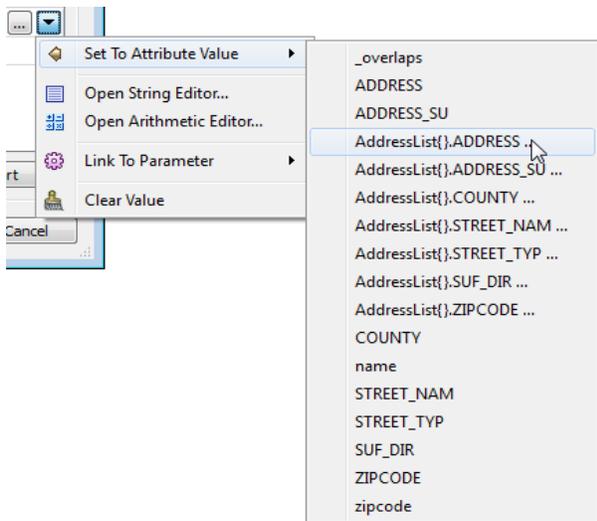
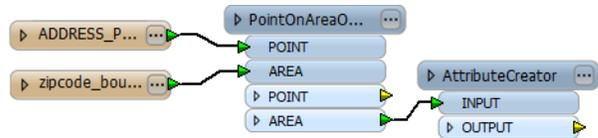
- myList{0}.myAttribute
- myList{1}.myAttribute
- myList{2}.myAttribute
- etc.

With integrated list handling, fewer list-related support transformers are required in a workspace.

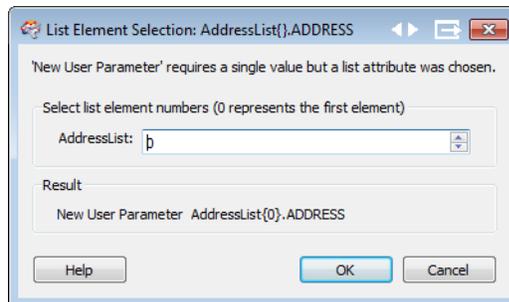
Selecting List Elements

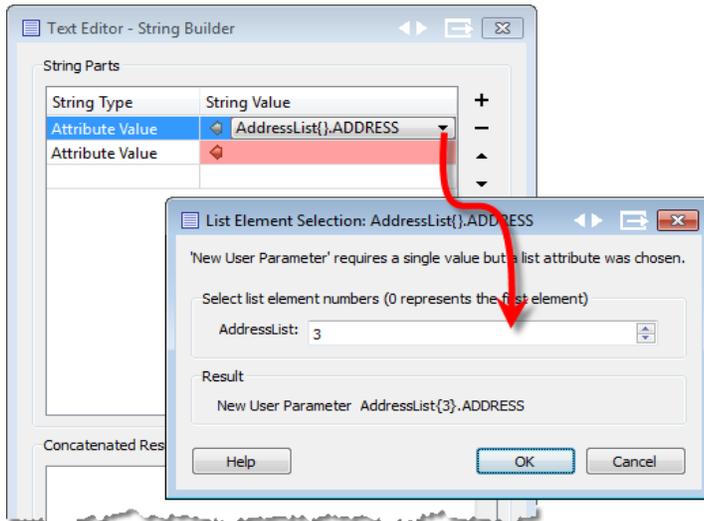
Transformers that allow attribute selection can be used to select list values. The only difference is that the user is prompted to select which element in the list is to be used.

In this workspace the author overlays address points with zip code boundaries. Because there are multiple addresses per boundary, a list (AddressList) is set in the PointOnAreaOverlayer parameters dialog:



In the *AttributeCreator*, when the author selects a list attribute from “Set to Attribute Value” a dialog pops up asking them to select which list element is to be used:

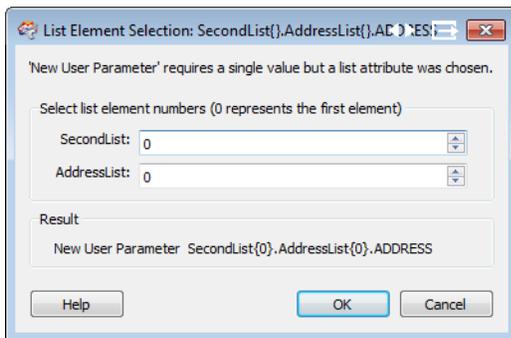




The same thing happens in a basic Text Editor – for example here where the dialog is opened by the author to concatenate address attributes together.

Simply selecting any list attribute opens a prompt to select which element in the list is required.

This list selection is useful because it avoids the need to expose list elements in advance of their use. It also works the same way when there are nested lists – the author is simply prompted to select multiple element numbers, one for each list:



Mr. R.G.B. Color says...

“FME is an absolutely iconic piece of software. Note the bold use of imagery within the AttributeCreator dialog, and how different functionality is represented symbolically using varying styles of iconography:



Value of an attribute



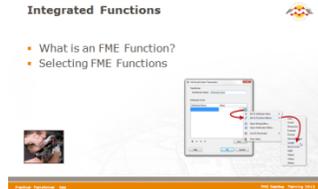
Value of a parameter



Constant value

The intrinsic visual amenity is incalculable, and so is the functionality! Be sure to check what happens when you type in a constant that has the same value as an attribute name”

Integrated Functions



Core FME Functions can now also be accessed directly within a transformer.

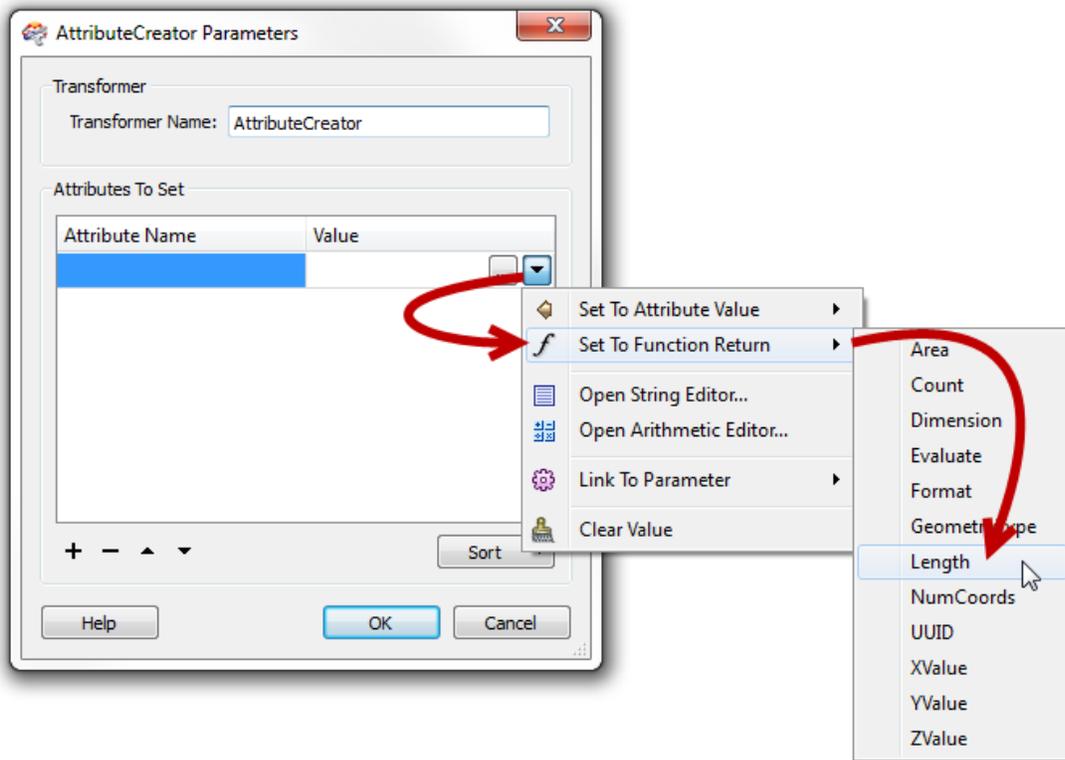
What is an FME Function?

A Function in FME is a built-in operation to be carried out on a feature. Functions are denoted by an “@” symbol prefix, and parameters enclosed in braces; for example @Area(1.0)

Ordinarily, Functions are hidden well below the level at which a workspace author or user operates. However, for FME2013, simple access to functions is provided through part of the Integrated Operations menu.

Selecting FME Functions

An FME Function can be selected within a transformer dialog using the “Set to Function Return” option, like in this AttributeCreator.



As an example, the Length (@Length()) function returns the length of the feature being processed. In this way, the author does not need to use the LengthCalculator transformer (for example) separately to obtain this result, the result can be obtained directly where it is to be used.

The Functions available on the menu are a subset of the full list of FME Functions. They are ones that produce an actual result value, rather than modifying existing attributes or geometry.



Example 25: Integrated Attributes	
Scenario	FME user; City of Interopolis, Planning Department
Data	Schools (Input: GML, Output: KML)
Overall Goal	Create a map showing school capacity rates
Demonstrates	Integrated Attribute Construction
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example25Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example25Complete.fmw

The task here is to create a map showing school capacity and attendance. Each school will be displayed as a circular buffer, the size of which is the ratio of capacity to attendance.

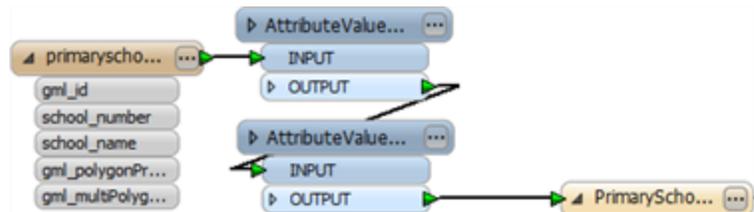
Integrated attribute construction will be used to calculate the size of the buffer and to create a title string for Google Earth.

1) Start Workbench

Start Workbench and open the beginning workspace.

Notice that it reads from a GML format dataset, constructs some numeric attributes, and writes to a KML dataset.

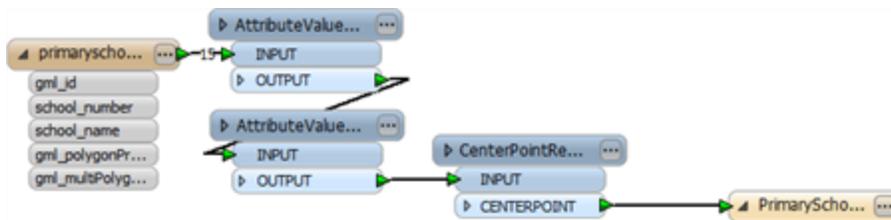
To meet Best Practice, you should now select Writers > Redirect to Inspection Application and then run the workspace, to see what data you are dealing with.



2) Add a CenterPointReplacer

Because the end user wants circular buffers, we should replace the school polygons with points.

So place a CenterPointReplacer transformer, connected to AttributeValueMapper_2:OUTPUT





Professor Lynn Guistic says...

“Instead of a CenterPointReplacer, why not try a CentrePointReplacer?!”

FME has transformer aliases - alternate names – to support cases like this where a transformer might be known by different names or spellings.

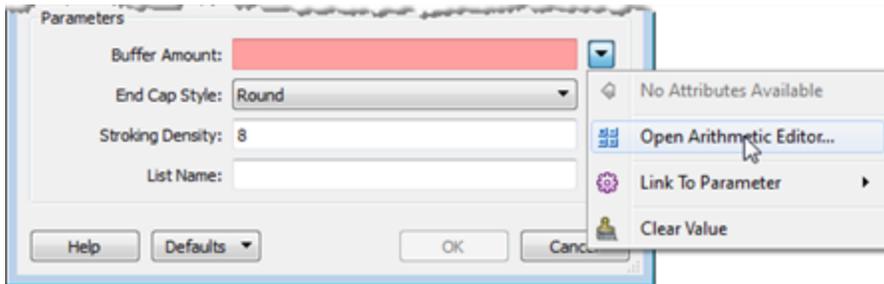
And if you have a moment, why not see if you can spot the humo(u)rours alias we have included for the Logger transformer?

3) Add Bufferer

Now add a Bufferer transformer, connected to the CenterPointReplacer.

Open the Bufferer parameters dialog.

Click on the Buffer Amount parameter down-arrow and select Open Arithmetic Editor.

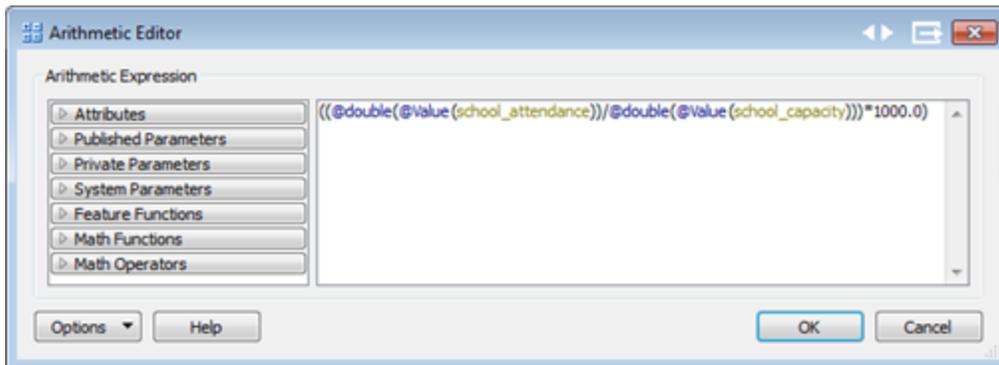


Now enter an arithmetic expression to calculate an appropriate buffer size.

You should make use of the attributes school_attendance and school_capacity

The idea is to display school attendance as a proportion of capacity, so a percentage value would be a good idea; but this will produce quite small buffers, so a larger multiplier might be of use.

A suggested expression is shown below:



`((@double(@Value(school_attendance)))/@double(@Value(school_capacity)))*1000.0`

Notice how adding an attribute also includes the @Value() function. You could get the same effect by manually entering the same content; the pick-list is just a shortcut.



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

“Are you wondering what the @double function does?
 It converts (casts) attributes to a floating point representation.
 It’s necessary because any proportion less than 100% would emerge as zero!
 For example, $80/100 = 0$ as an integer; whereas $80.0/100.0 = 0.8$

Google ‘Tcl Maths Functions’ for more information.”

Re-run the workspace to test the output. If you get a number of point features (i.e. with a buffer size of zero) then re-read Mr. Statistics-Calculator’s tip and edit your expression accordingly!

4) Add KMLStyler

Now add a *KMLStyler* transformer. Set a fill color and fill opacity for the features.

5) Add KMLPropertySetter

Now add a *KMLPropertySetter* transformer. This will be used to give a name and description to each feature.

Open the parameters dialog for the transformer.

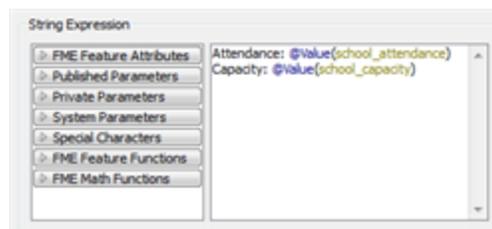
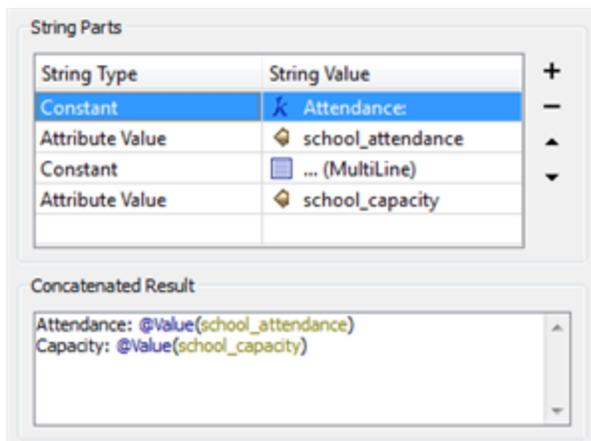
Under Navigation Tree > Name, click the down-arrow, select Set to Attribute Value and choose the attribute *school_name*

Under Description Balloon > Content, click the down-arrow and select Open Editor.

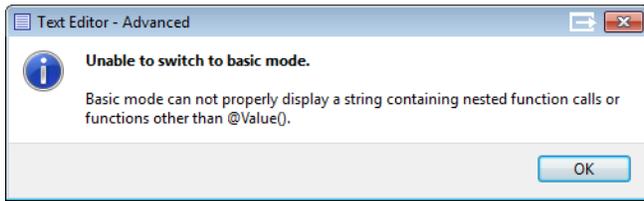
Using either the basic or advanced editor, set the concatenated output to read:

Attendance: <school attendance attribute>

Capacity: <school capacity attribute>



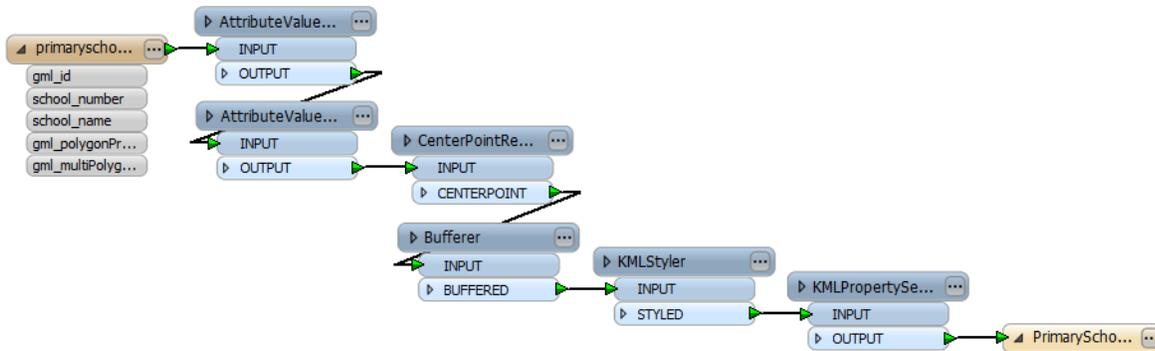
Notice how you can switch back from Advanced to Basic (and vice versa) up until you use something incompatible in the advanced editor (such as an FME function other than @Value). In that case you’ll get a warning message like this:



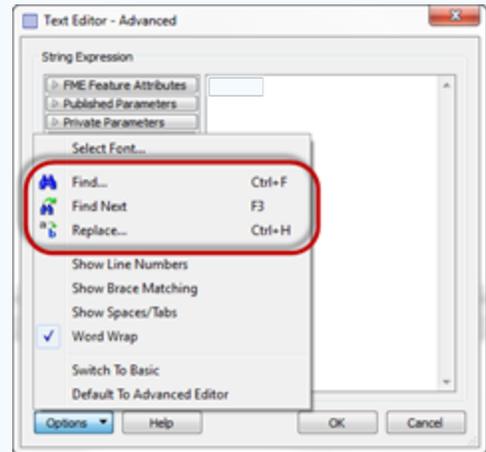
Also notice – if you construct the expression in the basic editor – that when you switch to advanced and back, the New Line string is merged into a general “Multiline” value. It’s still there, and still editable, we just chose not to parse out all new line characters in that scenario.

6) Run Workspace

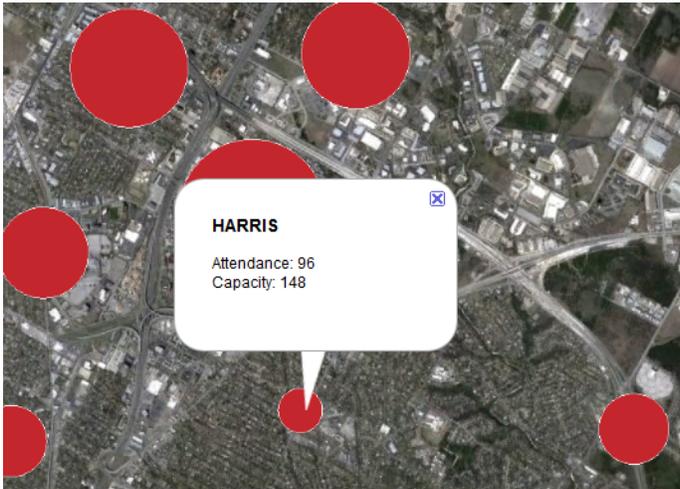
Now you can run the workspace, and open the output in Google Earth (don’t forget to turn off Redirect to Visualizer, if it is still turned on).



*“Quick tip: You can right-click on the writer feature type and choose Open Containing Folder.
Then the output dataset is ready and waiting for you to operate on.”*

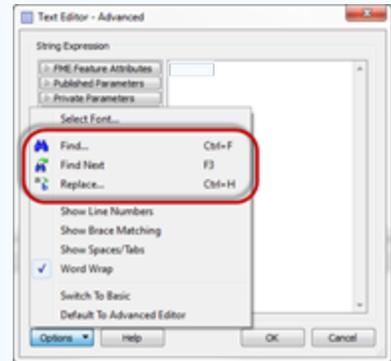


The output should look like this:



Did you notice the search options in the advanced text/arithmetic editors?

These features are new for FME2013.



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-common 25 transformers in 2011, calculated from user feedback. The table shows the percentage of all transformers used of that specific type.

Rank	Transformer	Change	%age	Operational	Support
1	Tester	-	5.0		X
2	Visualizer (now Inspector)	-	3.0		
3	AttributeCreator	-	2.5		
4	AttributeRenamer	-	1.9		
5	AttributeFilter	+4	1.8		
6	FeatureMerger	+5	1.6		
7	Creator	+3	1.5		
8	AttributeSetter	-3	1.3		
9	StringConcatenator	-1	1.2		
10	Clipper	New	1.2		
11	Reprojector	New	1.2		
12	AttributeKeeper	+3	1.2		
13	Counter	-	1.1		
14	GeometryFilter	+2	1.1		
15	AttributeRemover	New	1.0		
16	Joiner	New	1.0		
17	AttributeCopier	-10	1.0		
18	Aggregator	New	0.9		
19	TestFilter	New	0.9		
20	ExpressionEvaluator	-14	0.8		
21	AttributeExposer	-7	0.8		
22	Bufferer	New	0.7		
23	Dissolver	New	0.7		
24	2DPointReplacer	New	0.7		
25	Sorter	-8	0.6		



On the list of the top 25 transformers, indicate whether you think the transformer is Operational (it does something directly to a feature) or Support (it organizes features to be operated on)... or both. The Tester transformer is already filled in.

Do you think the results are a fair reflection of the relative importance of the two categories?

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 all others.



For FME2013, some transformers got renamed to make them easier to find and to better explain what their function is.

AttributeExpressionRenamer is now: BulkAttributeRenamer

AttributeExpressionRemover is now: BulkAttributeRemover

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. It can:

The ever-lengthening list of capabilities for this transformer is now:

- Manually Create an Attribute
- Set an Attribute Value
- Copy an Attribute
- Create an Attribute with an Editor
- Create an Attribute from a Function
- Create an Attribute from a Parameter

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).

Create an Attribute with an Editor (Can replace the StringConcatenator or ExpressionEvaluator)

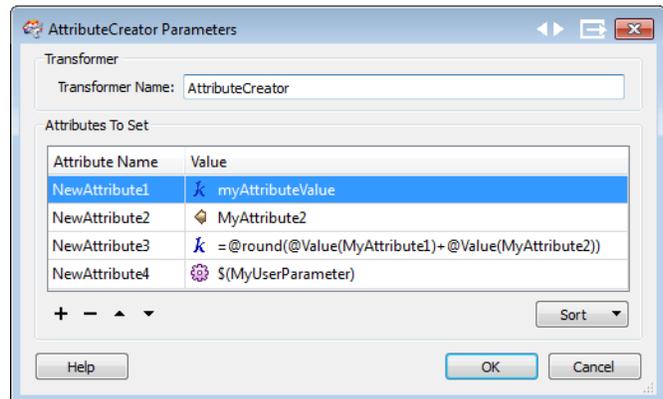
Because the string concatenation and expression evaluation components are included with the *AttributeCreator* dialog, a user can now create (or set) an attribute with a value defined in one of these editor dialogs.

Create an Attribute from a Parameter (Can replace the ParameterFetcher)

In the same way as other transformers, the *AttributeCreator* can fetch the value of a parameter with which to create (or set) an attribute. This parameter can either be a regular user parameter, or a pre-defined system parameter such as FME_BUILD_NUM

Here the user is:

- Manually creating an attribute
- Setting/Copying an attribute
- Creating an arithmetic attribute
- Setting an attribute to a parameter



AttributeRenamer

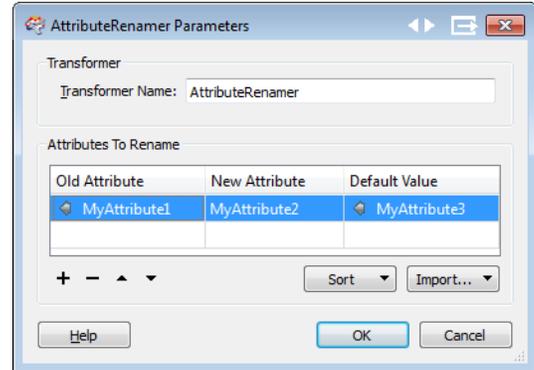
With string concatenation and expression evaluation also part of this transformer, it too is a fundamental transformer for authoring FME workspaces. It can:

- Rename an Attribute
- Rename and Set an Attribute
- Manually Create an Attribute
- Manually Create and Set an Attribute

- Create an Attribute with an Editor
- Create an Attribute from a Parameter
- Delete 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.



Rename and Set an Attribute

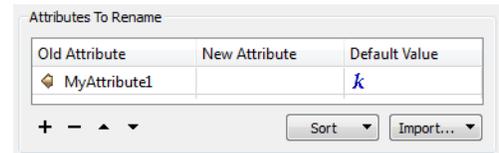
A field in the parameters dialog allows the user to set a default value. This way an attribute can be renamed, and a new value defined for it, simultaneously.

Manually Create an Attribute

Entering a value into the New Attribute field, without selecting an Old Attribute, means a new attribute is created; substituting for the *AttributeCreator*.

Manually Create and Set an Attribute

Entering a value 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*.



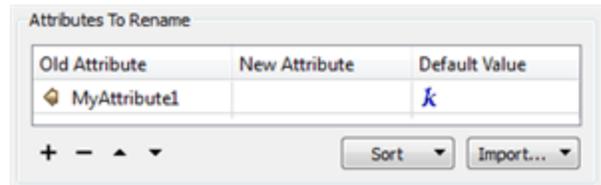


- Create an Attribute with an Editor**
- Create an Attribute from a Parameter**
- Create an Attribute from a Function**

The default value field allows all the same integrated functionality as other transformers; Set to Attribute Value, Open String Editor, Open Arithmetic Editor, and Link to Parameter. So a new attribute can be created and a value for it set using an editor or parameter. In 2013 it includes the new capability to fetch the value returned by an FME Function.

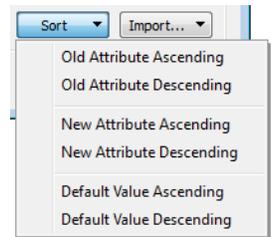
Delete an Attribute

By selecting an Old Attribute, but leaving the New Attribute field empty, the old attribute will be deleted.



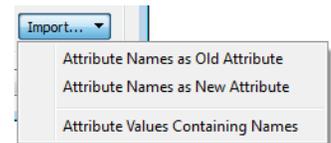
Sort Option

A button at the foot of the parameters dialog now allows the list of renamings to be sorted; by any field and in either direction:



Import Option

The Import button allows attributes to be imported from any supported FME dataset. This means schema mapping can be extracted from a dataset and used here.



The different import options are:

Attribute Names as Old Attribute

This option reads the attribute names from a dataset and imports them as attributes to be renamed. This option would be used when the attributes already exist somewhere.

In other words, if the dataset contains attributes A, B, and C, then the *AttributeRenamer* dialog will be populated with A, B, C.

For example, a user might read the schema of an XML dataset, or re-read the source data schema when they want to rename multiple attributes without typing them separately.

Attribute Names as New Attribute

This option also reads the attribute names, but imports them as attribute names to rename **to**.

For example, you might re-read the destination data schema that attributes must be renamed to.

Attribute Values Containing Names

This option reads the attribute values from a dataset, and imports them into the table.

This option would be used when schema mapping is stored in a lookup table (like a CSV file) and should be imported for use inside Workbench.

In other words, if you read attribute A from a dataset, whose values are X, Y, and Z, then the *AttributeRenamer* dialog will be populated with X, Y, Z.

This function can import the values of up to three attributes; one for Old Attribute Name, one for New Attribute Name, and another for the Default Value.

With all these updates, the *AttributeRenamer* can almost replace the *SchemaMapper* transformer.



The SchemaMapper Dragon says ...

“I’ve always wanted a little baby dragon, and now here she is. I’m so proud! I’ll call her the AttributeRenamer Dragonette.

Dragonette can do nearly everything I can – except for reading schema at run-time – and is much simpler.



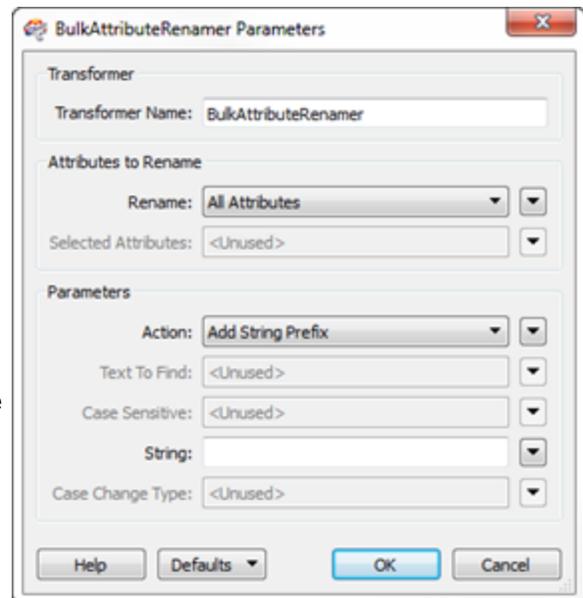
BulkAttributeRenamer

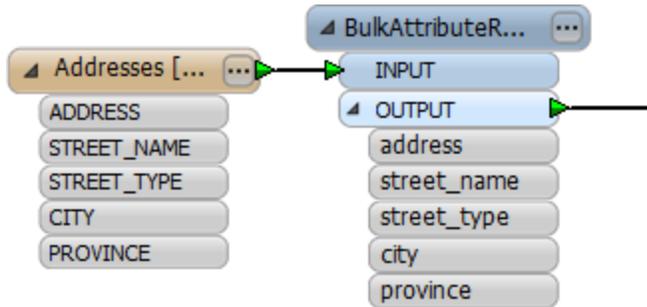
This transformer – newly renamed from the *AttributeExpressionRenamer* - carries out the core function of the *AttributeRenamer*, renaming attributes. But instead of manually specifying each attribute, this transformer lets the user enter a string-matching expression in order to define which attributes 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 upper case and need to be lower case on the output.

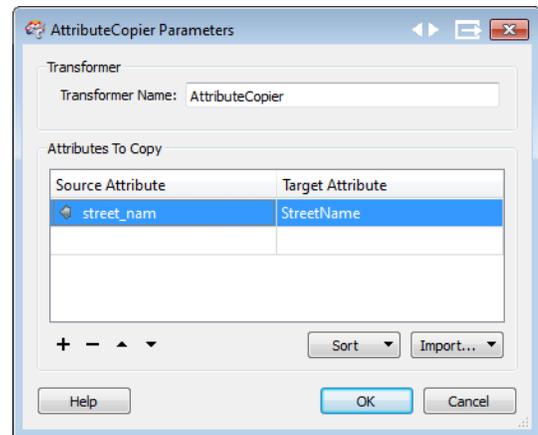
AttributeCopier

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 the same Sort and Import tools as the *AttributeRenamer*.



AttributeRemover

AttributeKeeper

These two transformers carry out the same function, but approach it from opposite directions.

Both remove 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 Workbench. 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 caches 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.

However, attributes **don't** need to be removed to avoid writing them to a destination dataset. If the attribute isn't defined on the destination schema, then it won't get written, regardless of whether the attribute exists or not. Only FME's internal format (FFS) breaks this rule.



BulkAttributeRemover

This transformer – renamed in 2013 from the *AttributeExpressionRemover* - 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.

AttributeExposer

Sometimes it is possible to have Workbench attributes that are invisible to the user.

There is an option to hide attributes in the Reader feature type dialog that can cause this, sometimes they won't appear because the workspace is dynamic, and sometimes they can be created by a transformer such as the Joiner or *SchemaMapper* that doesn't expose them because it can't know what attributes will be created until runtime.

The *AttributeExposer* transformer is used to manually expose attributes that are known to exist.

Because it includes the same Import tool as the *AttributeRenamer*, it can be used to expose an entire schema; for example one imported from an XML schema document or a comma-separated lookup table.



Example 26: Address Attribute Migration	
Scenario	FME user; City of Interopolis, Planning Department
Data	Address Points (Input: GeoMedia Access Warehouse, Output: File Geodatabase)
Overall Goal	Migrate and transformer address data
Demonstrates	Advanced Attribute Handling
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example26Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example26Complete.fmw

The task here is to migrate some address data from GeoMedia Access Warehouse to a File Geodatabase. In the process the schema must be mapped to a new Geodatabase structure.

1) Start Workbench

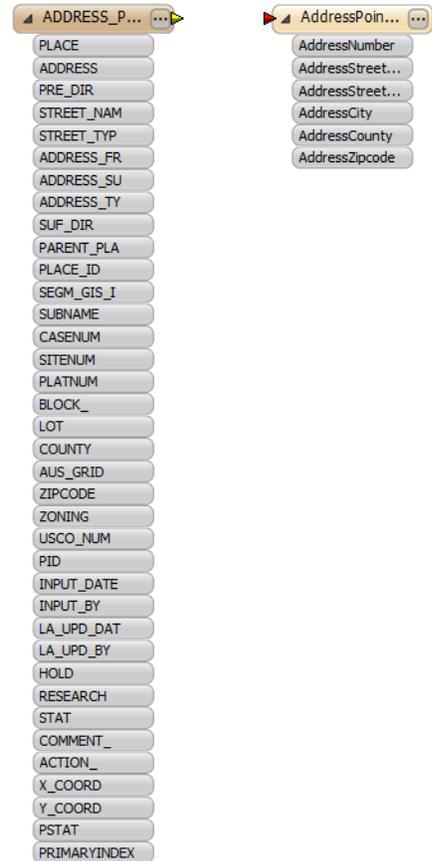
Start Workbench and open the beginning workspace.

Notice that it reads from a GeoMedia Access Warehouse with a rather extensive schema, and writes to a File Geodatabase using the File Geodatabase API writer.

The Geodatabase already exists, and contains a single dummy feature, but the Overwrite Geodatabase parameter is set to yes so that this will be deleted when the workspace is run.

At this point, no schema mapping has taken place.

To meet Best Practice, you should now inspect the source data to see what you are dealing with.



2) Hide Source Attributes

To make the task easier, unwanted attributes from the source schema should be hidden.

Open the reader Feature Type Properties dialog.

Click the User Attributes tab, and then click the 'Select All' checkbox to de-select all attributes.

Now reselect the attributes:

- ADDRESS
- STREET_NAM
- STREET_TYP
- ADDRESS_SU
- COUNTY
- ZIPCODE

3) Place AttributeRenamer

Place an *AttributeRenamer* transformer after the ADDRESS_POINT feature type.

4) Import New Attribute Names

Open the *AttributeRenamer* properties dialog.

Choose Import > Attribute Names as New Attribute

When prompted, set the Reader dataset to:

Reader Format	Esri Geodatabase (File Geodatabase API)
Reader Dataset	<u>C:\FMEData\Data\Addresses\Addresses.gdb</u>

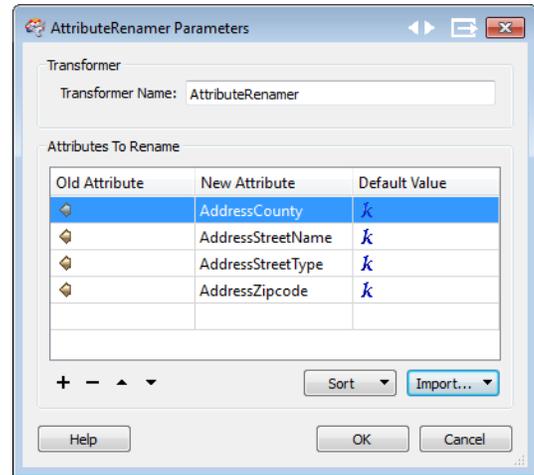
NB: You should be able to point to an existing GDB (Geodatabase) in that location. If it doesn't exist, copy it from the folder C:\FMEData\Resources\Miscellaneous

Click **Next**. Select AddressPoints as the Feature Type and click **Next** again.

Select the following attributes, and then click **Import**.

- AddressCounty
- AddressStreetName
- AddressStreetType
- AddressZipcode

The *AttributeRenamer* will now look like this:



Old Attribute	New Attribute	Default Value
COUNTY	AddressCounty	k
STREET_NAM	AddressStreetName	k
STREET_TYP	AddressStreetType	k
ZIPCODE	AddressZipcode	k

5) Set Old Attribute Names

Now select COUNTY, STREET_NAME, STREET_TYP and ZIPCODE as the old attribute names to map to the new attributes:

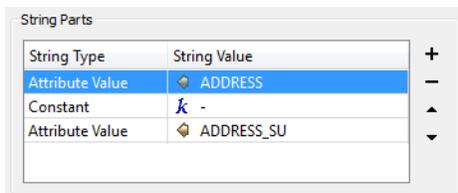
6) Create City Attribute

Now add an entry into the New Attribute field called AddressCity.
Set the Default Value field for this attribute to *Interopolis*.

Old Attribute	New Attribute	Default Value
COUNTY	AddressCounty	k
STREET_NAM	AddressStreetName	k
STREET_TYP	AddressStreetType	k
ZIPCODE	AddressZipcode	k
	AddressCity	k Interopolis

7) Create Number Attribute

Now add an entry into the New Attribute field called AddressNumber
In the Default Value field click the arrow and select Open String Editor from the drop-down list.
Set the text editor up to concatenate:
<ADDRESS>--<ADDRESS_SU>



Click **OK** to accept this, and then **OK** again to close the *AttributeRenamer*.



8) Run Workspace

Connect the *AttributeRenamer* to the writer feature type and then run the workspace.

Inspect the output to prove that it has been translated and mapped correctly.

Example 26b: Advanced Task	
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example26aComplete.fmw

Here is an advanced task to keep you busy!

9) Remove “dash” from Unsuffixed Numbers

AddressNumber is concatenated with a dash character to separate suffixes (for example, 999-A)

However, the dash still creeps in when there is no suffix.

Can you fix the workspace to make the dash only appear when there really is a suffix?

I think my solution is pretty elegant, and don't believe you could do it in fewer transformers, let alone using the Advanced Text Editor... but I am ready to be proven wrong!

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 *SurfaceModeller* transformer, which has output ports for contours, DEM points, TIN edges, triangles, and more.

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.

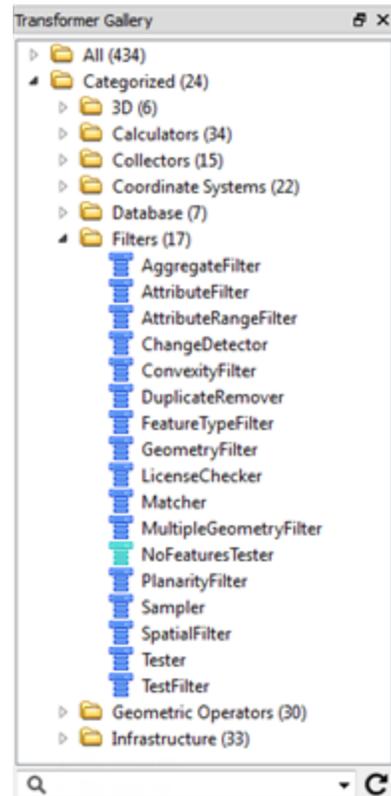
Transformers that Filter

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 *GeometryFilter*, *AttributeFilter*, *SpatialFilter*, and *Sampler*.

Because Filter transformers don't do content transformation (at most they could be said to aid structural transformation) they can be classified as support transformers.

Filter transformers have their own category within the Transformer Gallery.

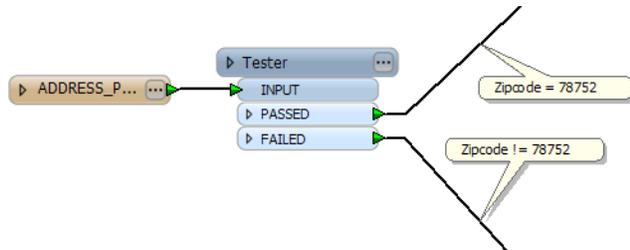


The Tester and TestFilter Transformers

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 creation of *Composite* tests, where a user can combine any number of AND clauses and OR clauses.

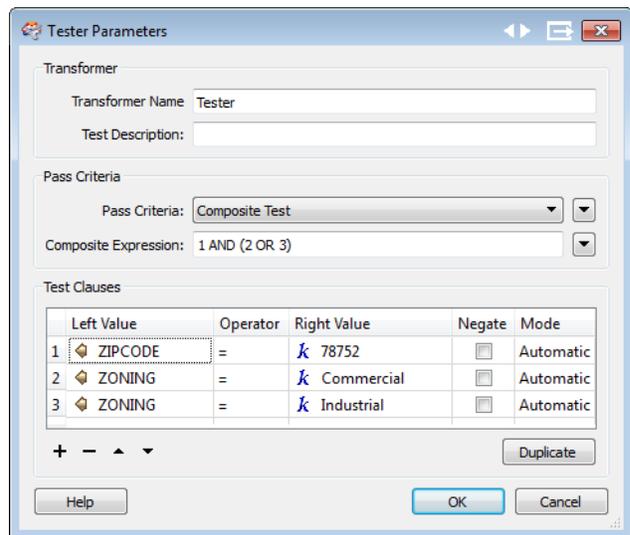


This *Tester* simply filters address features, depending on whether they are part of the 78752 ZipCode. It also demonstrates the usefulness of annotating a workspace!

But a composite test with ZipCode and zoning data is equally valid.

This is a composite test to find properties where the ZipCode is 78752 AND the zoning for that address is either Commercial OR Industrial.

The Composite Expression parameter is where this part of the test is defined:



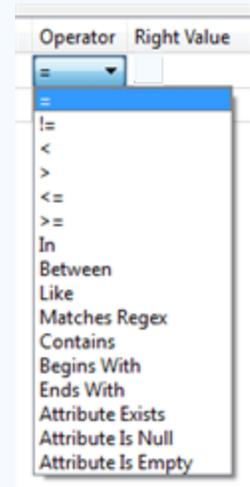


Besides the usual operators, there are also some based on a SQL where clause. These include:

- *In*
- *Between*
- *Like*
- *Matches Regex*
- *Contains*
- *Begins With*
- *Ends With*

...plus other tests such as:

- *Attribute Exists*
- *Attribute is Null*
- *Attribute is Empty*



TestFilter

The *TestFilter* (#19 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 such as Tcl.

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.

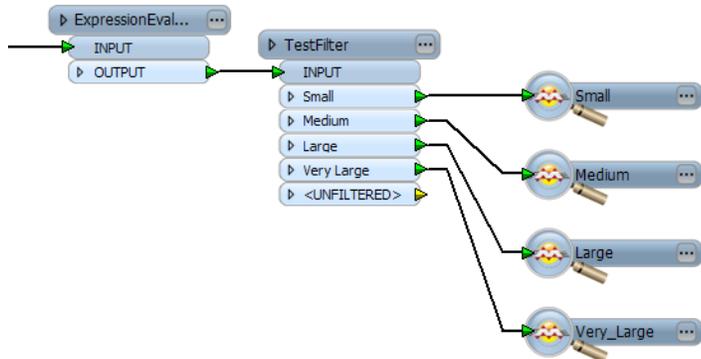
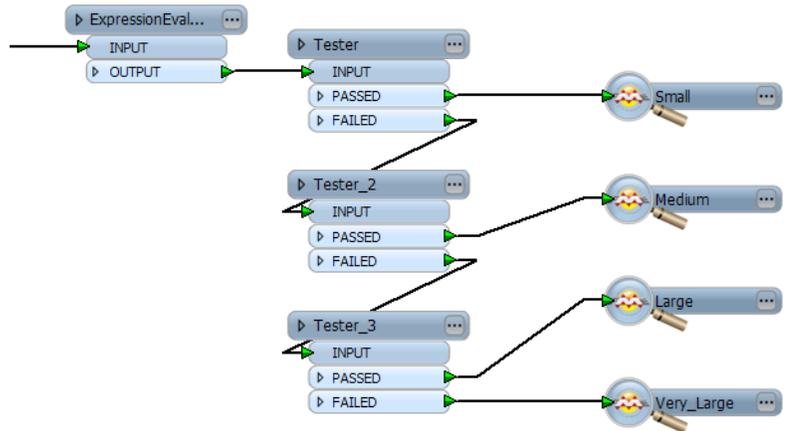
Features that pass are output through the matching output port. Features that fail are sent on to the next condition in the list. Therefore it's very important to get the conditions in the correct order.

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.

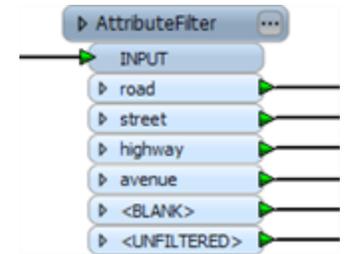
Other Key Filter Transformers

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 'road', 'street', 'highway', and 'avenue' according to the value of a chosen attribute.





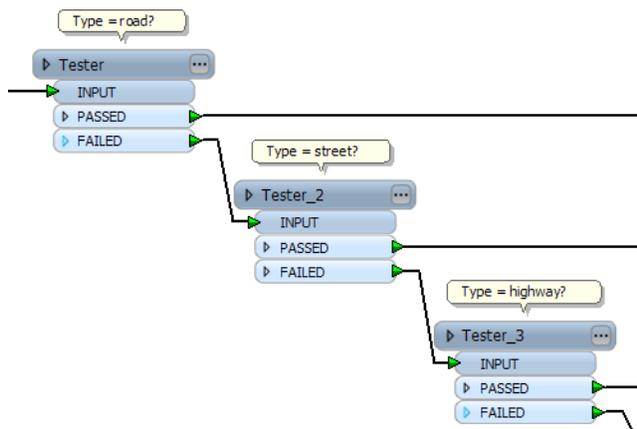
Dear Aunt Interop...

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?

'Use the *Tester* to filter one value of many attributes.

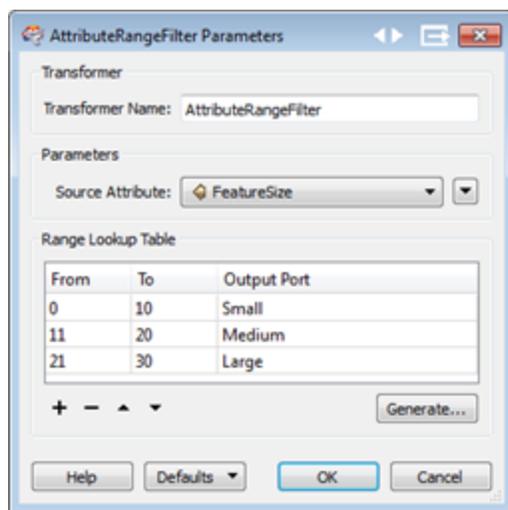
Use the *TestFilter* to filter many attributes and values using a mathematical clause.

Use the *AttributeFilter* to filter many values of the same attribute and when the clause is a simple string comparison.'



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.

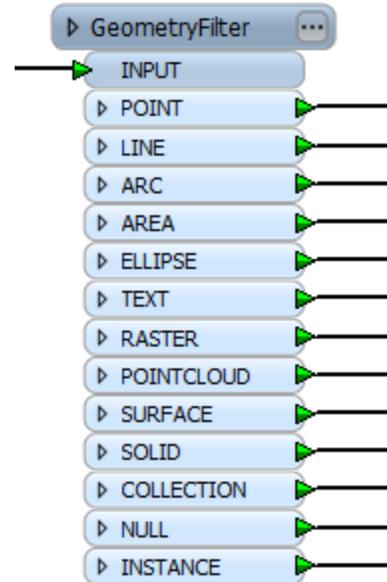
The *AttributeRangeFilter* parameters dialog has the option to generate ranges automatically from a set of user-defined extents.

GeometryFilter

The *GeometryFilter* (#14 in the top 25) directs features on the basis of geometry type; for example, point, line, area, ellipse.

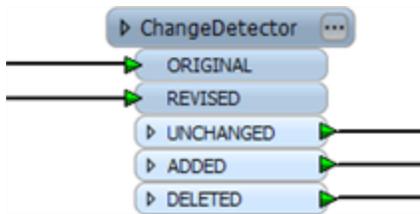
The *GeometryFilter* is useful for...

- 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.

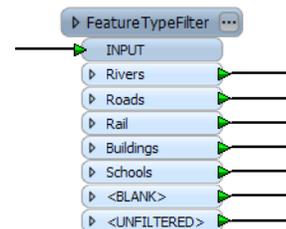
FeatureTypeFilter

A *FeatureTypeFilter* directs features on the basis of their original feature type. It can be used to separate data that has been merged to pass through

a single transformer together.

For example, instead of using an *AreaCalculator* for each feature type, they can be routed all into a single *AreaCalculator* and a *FeatureTypeFilter* used to split them back into different streams.

Here Rivers, Rail and Roads, Buildings, and Schools features are jointly passed into the transformer and emerge separated again into their original feature types.





Example 27: Conditional Filtering	
Scenario	FME user; City of Interopolis, Planning Department
Data	Addresses (GeoMedia), Zoning (GML)
Overall Goal	To find all residential addresses within 1000 feet of a class 1 highway.
Demonstrates	Conditional Filtering
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example27Complete.fmw

Because noise control laws are being amended in the City of Interopolis, residents living within 1000 feet 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 Address Reader

Use Readers > Add Reader to add a reader for address data.

Reader Format Intergraph GeoMedia Access Warehouse
Reader Dataset C:\FMEData\Data\Addresses\roadAllowancesAndAddressPoints.mdb

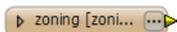
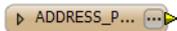
Reader Parameters
Table List ADDRESS_POINTS

3) Add Zoning Reader

Use Readers > Add Reader to add a reader for zoning data. The zoning data will be used to determine whether an address is residential or not.

Reader Format GML (Geography Markup Language)
Reader Dataset C:\FMEData\Data\Zones\zoning.gml

Feature Type zoning

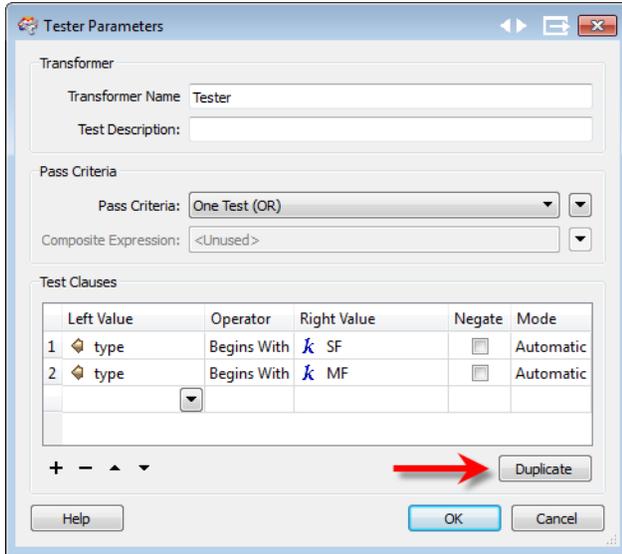


4) Add Tester Transformer

Add a *Tester* transformer to the Zoning feature type.

This *Tester* will be used to filter through residential zones.

All residential zones will start with either SF (Single Family) or MF (Multi-Family), so the *Tester* should be set up like this:



After the first clause is set up, the new Duplicate button will shorten the amount of time take to create the second clause. Give it a try!

The important thing is to set up the tests using the “Begins With” predicate, and set the Pass Criteria parameter to One Test (OR).

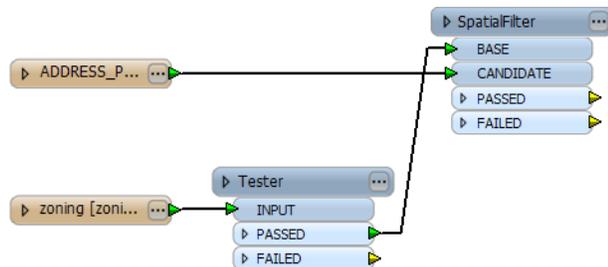
If the Pass Criteria is AND then a feature will need to be both single family AND multi-family, which is not possible.

5) Add SpatialFilter

Add a *SpatialFilter* transformer to the workspace. This will be used to test whether address features fall inside one of the filtered residential zones.

Connect the feature type FM0:ADDRESS_POINTS to the SpatialFilter:CANDIDATE port

Connect the Tester:PASSED output to the SpatialFilter:BASE port.



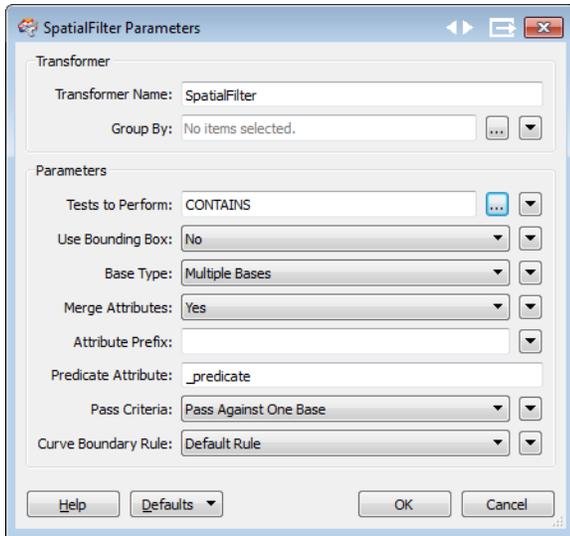
6) Set up SpatialFilter

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 Pass Criteria parameter should be “Pass Against One Base”.

This parameter is very important. The test will fail if it is not set correctly.

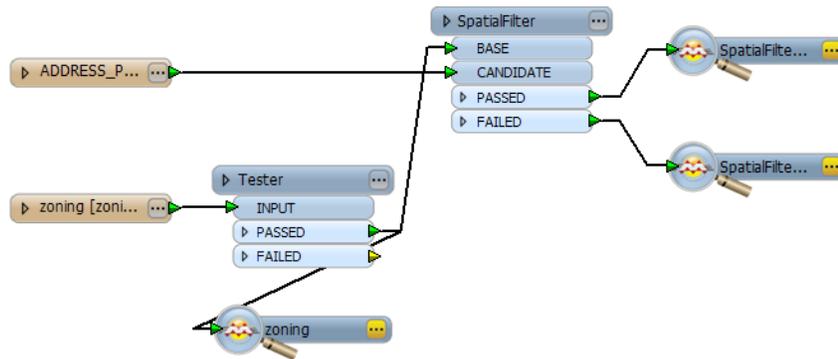


7) Add Inspectors

Add *Inspectors* to both *SpatialFilter* output ports, and the Tester:PASSED port.

8) Save and Run Workspace

Save the workspace. Run the workspace. Inspect the output – adding the zoning data to the same view – to prove that it has worked as expected.



Synchronous Read and Write



Normally data is input or output using Readers and Writers however some transformers have this ability as well.

Traditionally, workspaces read data at the start of the process (Extract), and write data at the end (Load). In between is where data transformation takes place.

However, in some cases data needs to be read or written during transformation. Because this occurs simultaneously to the transformation it's possible to call these *synchronous* reads and writes.

*Synchronous: [adjective]
Occurring at the same time; coinciding in time; contemporaneous; simultaneous.*

Why Read and Write Mid-Translation?

There are a number of common scenarios where a user wants to read or write data during a translation, and a number of transformers that exist to carry out these scenarios.

Scenario	Merging data stored within a database
Example	Querying features (spatially or with SQL) against a database
Transformer Category	Spatially filtering translated data against database objects such as a state/province/county boundaries Database Transformers
Scenario	Reading and writing data from/to a web site, or web service
Example	Extracting traffic information from a web site to add to translated road features
Transformer Category	Web Services Transformers
Scenario	Reading and writing data from/to local files
Example	Reading raster images related to features, to add to their database record
Transformer Category	Infrastructure Transformers

Database Transformers

These are transformers that read or write from/to a database, mid-translation.

FeatureReader

The *FeatureReader* transformer may come under the category of database, but it can be used to read any FME-supported format of data.

The first use of this transformer is to simply read a dataset, just like a reader.

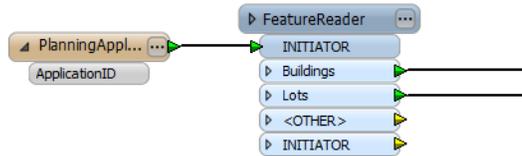
The transformer is initiated by an incoming feature to read an existing dataset. It then returns the contents of the dataset as features. In other words it is really doing the job of a Workbench Reader, but in the transformation phase of the workspace.

The initiator feature(s) can come from a Reader, or from a *Creator* transformer.

A second role of this transformer is to carry out spatial and non-spatial queries on the data being read. In this way *any* format of data may be treated as if it were a database.

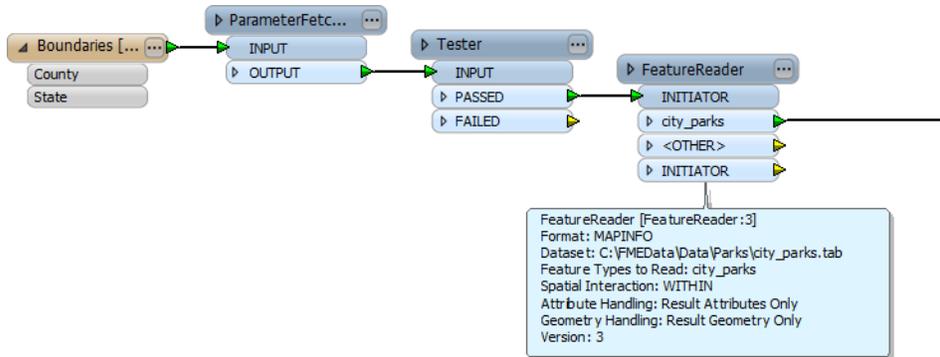
For instance, if the initiator feature is a polygon, the *FeatureReader* can be made to read point features from a selected dataset, where those points fall inside the incoming polygon.

For example, here a user reads a list of planning applications stored in a text/CSV dataset. Each record has an ID that is used in a *Where* clause in the *FeatureReader* to retrieve the appropriate features from a spatial database.



As a second example, here a set of county boundaries are read into the workspace. The user is prompted (with a user parameter) to select one of the counties. The user's choice is retrieved with a *ParameterFetcher* transformer, and a *Tester* used to keep only the matching county.

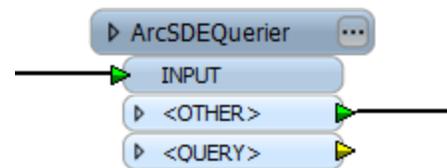
The selected county feature is used as an initiator in a *FeatureReader*; in this case the transformer is performing a spatial query, retrieving all parks within the county boundary, from a MapInfo dataset.



For FME2013, the *FeatureReader* replaces the *OracleQuerier* transformer, as it now replicates all of that format transformer's functionality.

Querier Transformers

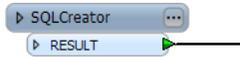
The *ArcSDEQuerier* transformer is similar to the *FeatureReader* but is optimised specifically for ArcSDE spatial databases. The main difference is that it can carry out inserts and updates to the database.



Other transformers in the Database category are *SQLExecutor* and *SQLCreator*.

Rather than being just a reader, the *SQLExecutor* can carry out any SQL statement on a database; but unlike the *FeatureReader*, it is limited to true database formats.

The *SQLCreator* is similar, but doesn't require a trigger, and simply creates features based on the results of a SQL query.



Joiner

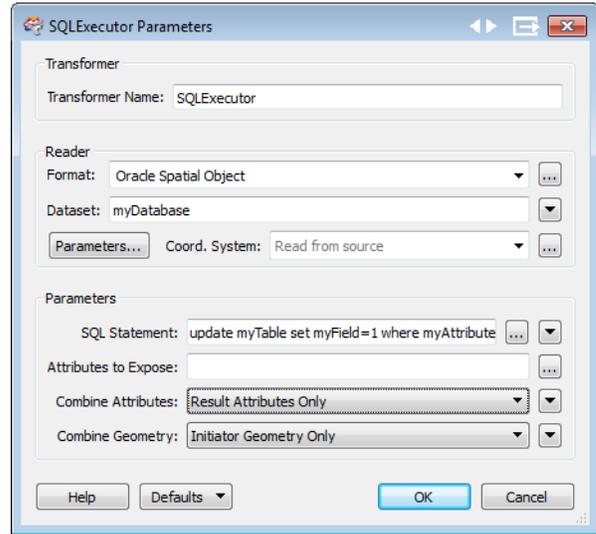
The *Joiner* transformer (#16 in the top 25) retrieves database records that match the ID (or key) of an incoming feature, and attaches that information to the feature. It can be described as a specialized version of the *SQLExecutor*, designed purely for retrieving non-spatial attributes.

The *Joiner* also replicates the scenario of retrieving database records with a reader, and then attaching them to spatial features using a *FeatureMerger* transformer.

However, the *Joiner* is usually (but not always) a more efficient way of doing this because:

- The *FeatureMerger* has limited functionality. The *Joiner* has advanced functionality, which is designed to improve performance.
- It is not particularly efficient to read all of the records from a database to only match some of them. It's better to only retrieve those records that are actually needed.

The *FeatureMerger* (#6 in the top 25) is more appropriate when the user needs to merge features within the same (non-database) dataset, or if they are trying to carry out a QA task and need to know which database features are (or are not) used during the join.



For FME2013, the Joiner transformer has been totally revamped, and its parameters setup changed from a wizard to a single dialog.

Web Services Transformers

There are numerous transformers that can be used to retrieve information from a web site.

All of these – along with a few other useful web-related transformers – can be found in the Web Services category of the transformer gallery.

**HTTPFetcher, HTTPUploader,
HTTPFileUploader, HTTPDeleter, HTTPMultipartUploader**

HTTP is a protocol for accessing online information.

It has a number of different request methods that indicate the action to be performed.

FME transformers support the following methods:

- PUT
- GET
- POST
- DELETE

In other words, these transformers can be used to Read, Write, or Delete datasets stored online.

For example, if the *HTTPFetcher* is directed to a web page (such as www.safe.com), then the contents of that page are returned and stored in an attribute.

However, if the *HTTPFetcher* is directed to a web service or feed, then data will be returned and stored in an attribute in a format such as XML, WFS, or JSON.

ImageFetcher

The *ImageFetcher* transformer is for retrieving an image from an online source specifically. It simply uses a HTTP GET operation on a specified URL.

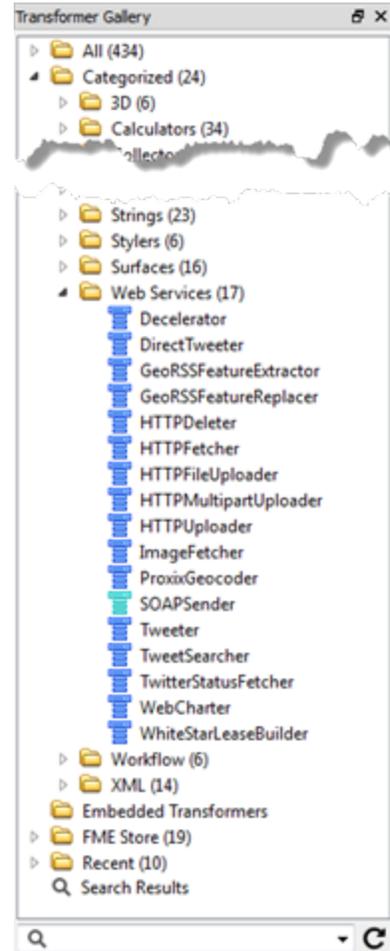
The reason to use this transformer (over the *HTTPFetcher*) is that this will output a feature with raster geometry. The *HTTPFetcher* would return the original geometry with the raster data stored as an attribute.

Support Transformers for Web Services

A number of transformers exist to support web-related operational transformers. In most cases, they're used to turn the data returned from a GET operation into standard FME features.

The *JSONExploder*, *JSONFlattener* and *JSONExtractor* transformers can extract information from a JSON format dataset. *JSONExploder* will create new features whereas the *JSONExtractor* and *JSONFlattener* will create new attributes.

XMLFeatureMapper allows a user to apply an xfMap to incoming XML data. An xfMap is a form of lookup table that maps XML features to an FME equivalent.



Infrastructure Transformers

Transformers that read and write to files are mostly in the Infrastructure category of transformers, although there are other transformers in the Raster category and even a Reader designed to support such synchronous read/writes.

AttributeFileReader and AttributeFileWriter

FME is a feature-based relational model. However, sometimes it's useful to read an entire dataset as a single document. These two transformers respectively read the contents of a file into an attribute on a feature and write the contents of an attribute out to a file.

The data being read or written can take any form, but the most common include:

- **Raster Images:** These transformers can be used to read a raster image (such as a photograph) into an attribute, or write a raster image out to a file
- **HTML:** It's becoming common for workspaces to generate HTML content (particularly when implemented on FME Server) and these transformers are ideal for reading in HTML templates or writing out HTML files.
- **Metadata:** These transformers are useful for reading and writing metadata – which is often stored as an XML document – during a translation. That way a workspace can read source metadata, update it, and write it back again.
- **Geometry:** These transformers can be used to read and write geometry when it is stored in a form such as well-known text (WKT) or well-known binary (WKB). Because of this they work particularly well in combination with the *GeometryExtractor* and *GeometryReplacer* transformers.



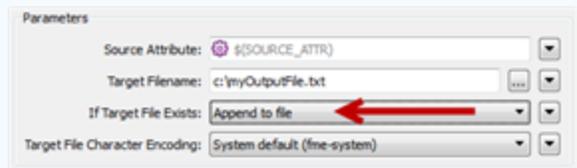
This section of a workspace converts feature geometry to WKB stored in an attribute.

The attribute is then written out to a file.

The *AttributeRemover* removes the geometry attribute because in terms of system resources it's expensive to keep it for the rest of the translation.



In FME2013 the AttributeFileWriter has a new option to append to a file, rather than overwriting it completely each time.

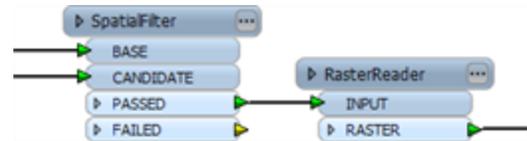


RasterReader

The RasterReader transformer reads raster datasets as if they were being read through a Reader rather than a transformer. Most raster formats are supported, the limitation being that source parameters are not available because they would differ for each format.

The typical scenario for this is when a set of raster datasets needs to be read, but the selection of these is dependent on a transformation within the workspace.

The scenario here is that a user has a set of vector tile outlines, each of which points to a matching raster dataset.



The tile outlines are filtered according to the results of a spatial overlay with another dataset, the ones that pass going on to retrieve their matching raster feature with the *RasterReader*.

This way, only raster features corresponding to the area of interest are returned to the translation.

RasterReplacer

The *RasterReplacer* isn't really a Read/Write transformer, but it gets a mention here as the sister transformer to the *RasterReader* and a useful support transformer for this type of functionality. The *RasterReader* reads an external raster dataset, whereas the *RasterReplacer* creates one from an attribute.

For example, if a raster dataset is read into an attribute using the *AttributeFileReader*, it can then be converted to a true raster feature using the *RasterReplacer* transformer.

Directory and File Pathnames Reader

Although not a transformer, this reader can be used to support reading and writing transformers. It reads a list of files within a specified directory and returns a feature for each file.

The filename (and path) is attached to each feature, so they can be routed through a reader transformer to read that particular file.



Example 28: Synchronous Reading	
Scenario	FME user; City of Interopolis, Planning Department
Data	Addresses (GeoMedia), Zoning (GML), Roads (MapGuide SDF)
Overall Goal	To find all residential addresses within 1000 feet of a class 1 highway.
Demonstrates	Synchronous Reading
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example28Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example28Complete.fmw

This example continues the previous one, using the *FeatureReader* transformer to match the chosen addresses against highway data.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Example 27.

Alternatively you can open *C:\FMEData\Workspaces\DesktopManual\Example28Begin.fmw*

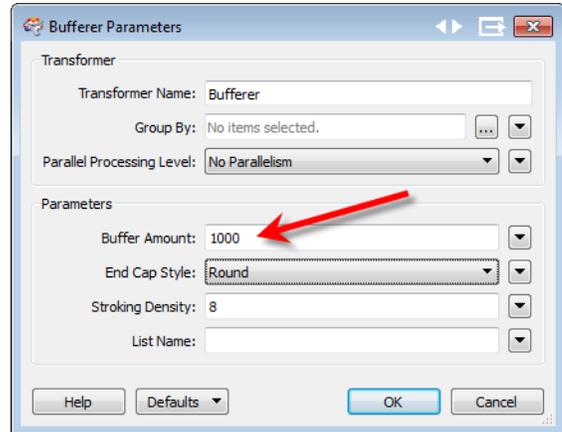
Remove any leftover Inspector transformers.

2) Add Bufferer

To find features within a set distance of another, a buffer can be used. Here (to find roads within 1000 feet of an address) a buffer should be added around the addresses (not the roads because they haven't been read yet).

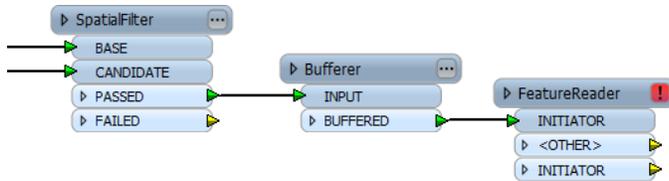
Add a *Bufferer* transformer after *SpatialFilter:PASSED* and set it up to create a 1000 foot buffer around each address point.

At this point you may wish to add an Inspector transformer after the *Bufferer* and run the workspace, just to make sure the output is as you expect.



3) Add FeatureReader

Add a *FeatureReader* transformer after the *Bufferer:BUFFERED* port.



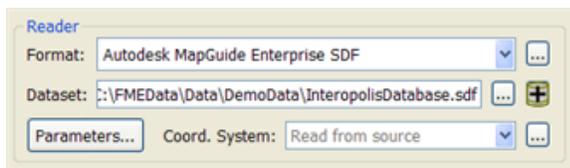
4) Set up FeatureReader

Open the properties dialog (actually a wizard) for the *FeatureReader*.

The dataset to read from is:

Reader Format Autodesk MapGuide Enterprise SDF
Reader Dataset *C:\FMEData\Data\DemoData\InteropolisDatabase.sdf*

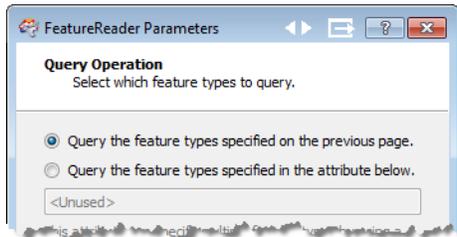
Parameters Remove Schema Qualifier = Yes





The Feature Types to Read parameter should be set to Roads (no other feature types are required).

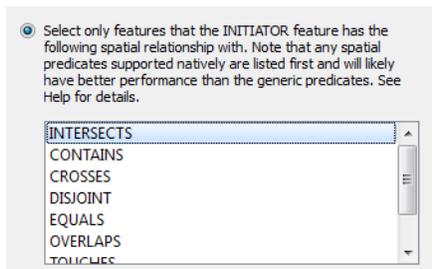
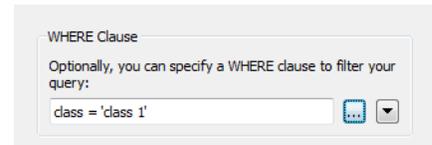
If the names of all types are preceded by "Default." then you didn't check Remove Schema Qualifier.



The Query Operation is the default (Query the feature types specified on the previous page)...

...but on the same dialog, the WHERE clause should be set to:

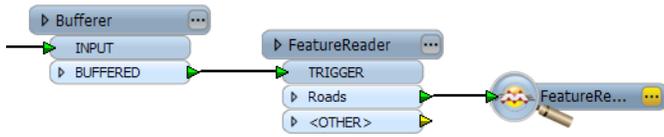
`class = 'class 1'`



The next dialog in the wizard should set up the Spatial Query. In this case an INTERSECTS operation is required (i.e. where the 1000 feet buffer intersects a class 1 road).

Finally, because the idea is to get a list of the addresses that intersect (not a list of the roads) the Merge Options dialog should specify that both attributes and geometry for output features should come from the original initiators (the addresses).





5) Run Workspace

Add an *Inspector* transformer to the FeatureReader:Roads port, save the workspace, and run the workspace.

6) Inspect Output

Inspect the output from the workspace. You will want to add the Roads data to the view to check that the output does actually intersect a class 1 highway.



Basic Custom Transformers



Custom Transformers are a very powerful tool at either a basic or an advanced level.

What is a Custom Transformer?

A custom transformer is a sequence of standard transformers condensed into a single transformer. Any existing sequence of transformers can be turned into a custom transformer.

Among other functions, custom transformers help to:

- Tidy up a busy workspace by moving large content onto subpages of the canvas.
- Efficiently reuse the same sequence of transformers in a number of places.

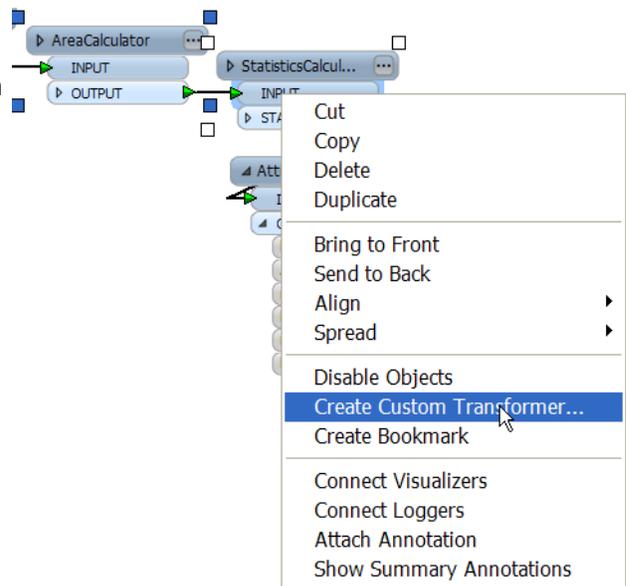
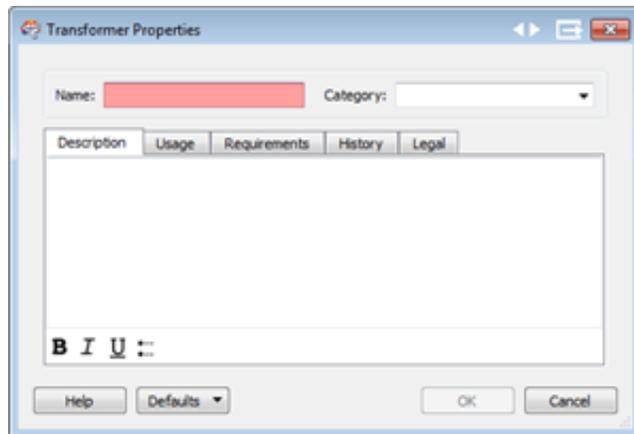
Creating a Custom Transformer

There are several ways to create a custom transformer.

An empty Custom Transformer is created using Insert > Custom Transformer from the menu bar or by right-clicking an empty place on the canvas and selecting Insert Custom Transformer...

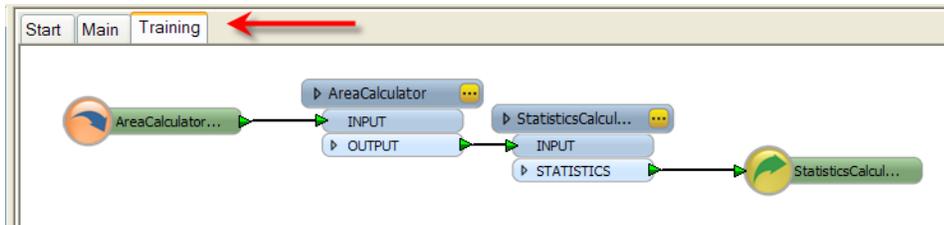
A custom transformer, based on existing transformers, is created by selecting those transformers, right-clicking, and selecting the Create Custom Transformer option.

All Custom Transformers require a name and (optionally) a category and description. This dialog also has other fields for legal and technical requirements, and allows rich text in some fields.

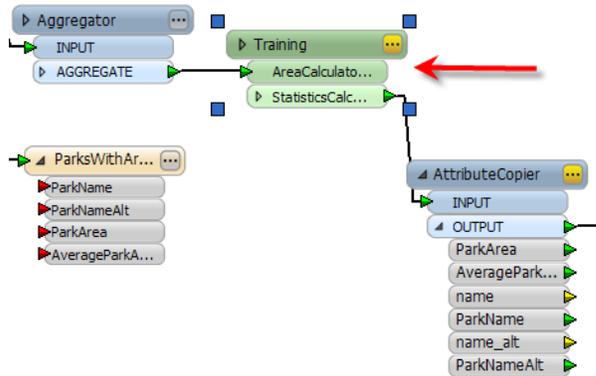


For FME2013, a custom transformer can be created by selecting the objects and pressing Ctrl+T.

The completed custom transformer appears in its own tab in the workspace...



...and in the Main tab the original sequence has been replaced by the custom transformer.

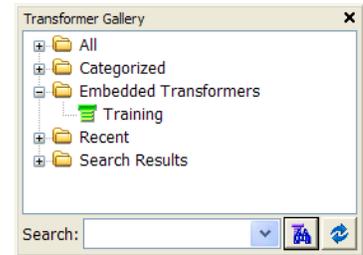


At this point the custom transformer definition is stored within the workspace file itself.

This is called an *embedded* custom transformer.

An embedded custom transformer can now be used within this workspace in the same way as any normal transformer.

For example, extra instances can be placed by looking it up in the transformer gallery (but not through Quick Add)



Custom transformers are edited in the same way as any other part of the workspace canvas. Remember that the definition applies to all instances of the custom transformer; if it's used multiple times, then definition edits apply to every one of these occurrences.

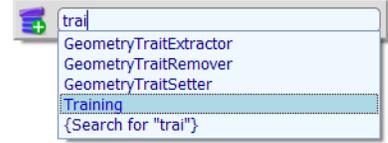
Nested custom transformers are permitted, so it's possible to create one custom transformer inside another. This 'nesting' can go on for a number of levels.

Custom Transformer Files

In order to facilitate sharing between users, custom transformer definitions can be saved to a file. A custom transformer file is created by using File > Export as Custom Transformer on the menubar, and is saved to the folder <user>\FME\Transformers with a .fmx extension.

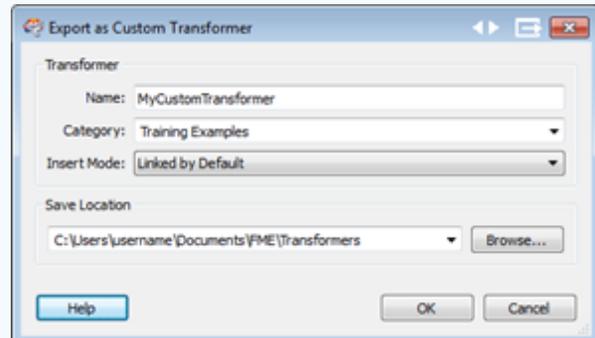
A custom transformer saved in this way now becomes available to all FME workspaces created by that user.

It can be placed in a workspace using either the transformer gallery, or Quick Add.



The Export function gives a preview of the properties that the transformer will be saved with.

This gives the user the chance to change the name, category, location, or insertion mode, before creating the transformer.



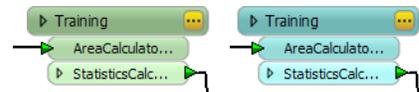
Insertion Mode

A custom transformer has two insertion modes: embedded or linked.

Placing such a transformer in embedded mode means that a copy of the definition is placed in the workspace, and it is stored and edited separately from the external file definition.

Placing such a transformer in linked mode means that a reference to the external file definition is placed in the workspace. The workspace does not contain a copy of the custom transformer definition, and so cannot be used unless it has access to the appropriate .fmx file.

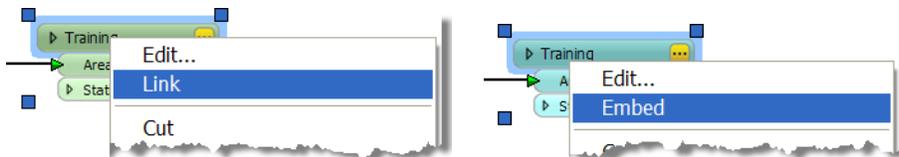
The colour of the transformer distinguishes embedded ones (green) from linked ones (cyan). There may even be two versions in the transformer gallery, also differentiated by color.



A linked transformer is affected by any editing that takes place in the fmx file definition. This, of course, can be useful, for a custom transformer used in multiple workspaces to be updated simultaneously.

Whether a new transformer is placed as embedded or linked by default is controlled by a parameter in the custom transformer itself; one which can be set when exporting.

However, it is possible to switch an instance of a custom transformer from linked to embedded (or vice versa) using right-click options on the transformer.





First Officer Transformer says...

'When embedded, a custom transformer can only be re-linked if it's un-edited. After you edit an embedded transformer the Link option disappears.

A word of advice—consider this when you tackle the following example!'



Example 29: Custom Transformer Creation	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	To set up an easy way for FME authors to calculate the average area of a set of polygon features.
Demonstrates	Custom Transformer creation and use
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example29Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example29Complete.fmw C:\FMEData\Workspaces\DesktopManual\AverageAreaCalculator.fmx

This example refers to the grounds maintenance workspace created in a prior session.

The calculation of average area is such a useful concept that you wish to turn it into a re-usable solution that can be shared among other FME users. At the same time the extent can be expanded to include other statistics such as maximum area and minimum area.

1) Start Workbench

Open the workspace *C:\FMEData\Workspaces\DesktopManual\Example29Begin.fmw*

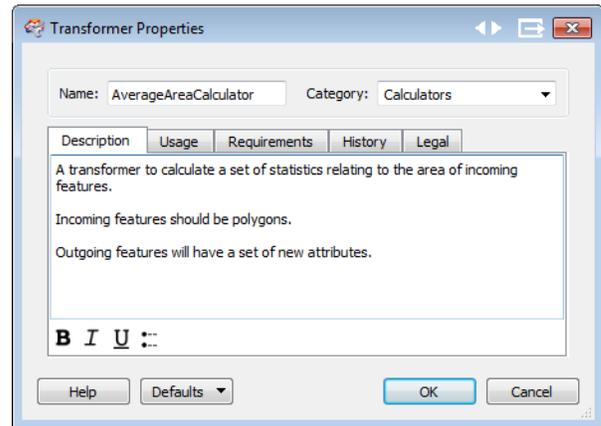
2) Create Custom Transformer

Select the *AreaCalculator* and *StatisticsCalculator* transformers in the workspace.

Right-Click the selection and choose the menu option 'Create Custom Transformer'.

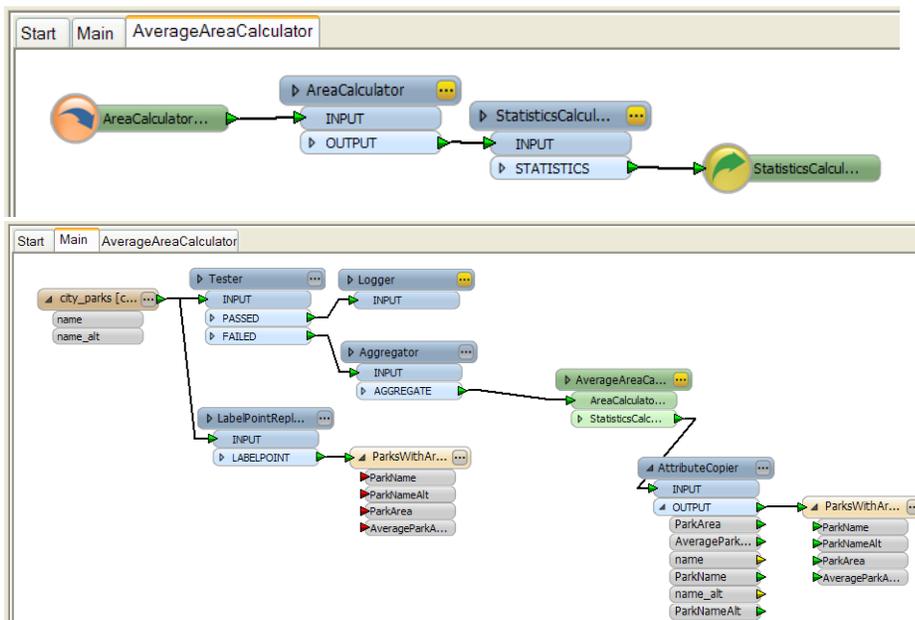
In the Transformer Properties dialog set the new transformer's name to *AverageAreaCalculator*. Enter the category as *Calculators* and add any description you want.

Click OK and a new custom transformer is generated.



3) Inspect Custom Transformer

Flip back and forth between the *AverageAreaCalculator* tab and the Main tab to see how the custom transformer is constructed, and how it is placed in the workspace itself.



4) Export Custom Transformer

At the moment the custom transformer is embedded within this workspace. It needs to be exported as an external definition.

In the *AverageAreaCalculator* tab, choose File > Export as Custom Transformer from the menubar. A dialog will appear with which to define the export settings. Name and Category should already be filled in, so simply set the Insertion Mode to "Embedded by Default".

Leave the default save location, or else FME will never be able to find the transformer.

A new instance of FME starts, which contains the transformer definition.

Notice on the window header how it's named as a .fmx file.

5) Test Transformer Modes

Back in the Main tab of the original workspace, you can right-click on the *AverageAreaCalculator* transformer and choose the option to Link. This turns the instance of the transformer to a linked version. You can then choose Embed to turn it back again.

Leave the transformer in embedded mode, as it does all that is required for this workspace.

6) Edit External Definition - 1

If it is to be used in a generic way – i.e. not for a specific purpose – the external custom transformer should really be tidied up.

The first task is to ensure that unwanted attributes used only within the transformer, are not carried back into a workspace.

Back in the fmx file session, open the *AreaCalculator* transformer parameters. *ParkArea* is one attribute that should be altered.

Firstly it should only be used within the custom transformer and removed before a feature exits.

Secondly it should be named in a way that is less likely to clash with any other existing attribute names (i.e. an incoming feature might already have a *ParkArea* attribute).

Rename the *ParkArea* attribute to *_myTransformerArea* and click OK to accept this change.

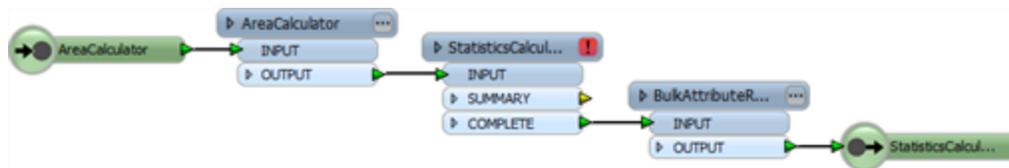
The *StatisticsCalculator* will be flagged as incomplete as an automatic response, and that will be dealt with later.

7) Edit External Definition - 2

Now place an *BulkAttributeRemover* transformer just before the transformer output object.

Open the parameters dialog and set *_myTransformer** as the expression to remove. This setting is very much case sensitive, so be careful to get it correct.

The transformer definition will now look like this:



Notice how attributes starting *_myTransformer* are removed. This is better than placing an *AttributeRemover* and choosing each attribute separately. As long as all temporary attributes are named with *_MyTransformer* they will be properly cleaned up before exit.

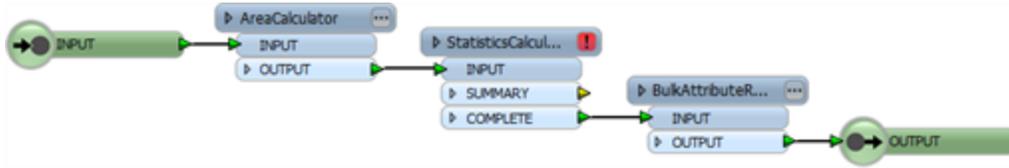
8) Edit External Definition - 3

The input and output objects in the transformer definition have named derived from the transformers used to generate the definition. In a generic custom transformer they should not be so named.

Double-click the Input object and rename it to INPUT.

Double-click the Output object and rename it to OUTPUT.

The FME standard is to use upper case for port names, although it doesn't really matter.



9) Edit External Definition - 4

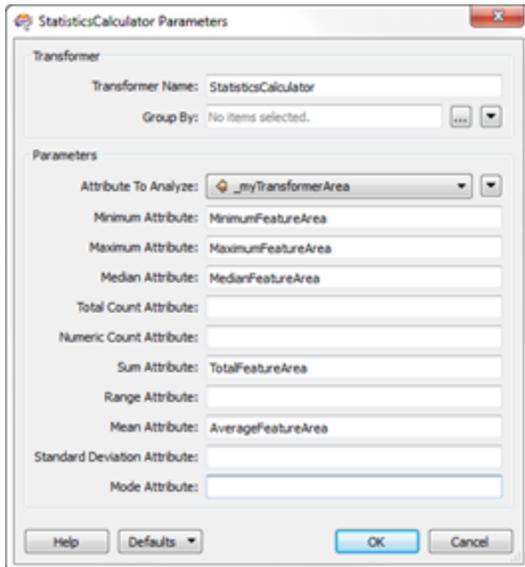
Finally open the parameters dialog for the StatisticsCalculator.

Change the Attribute to Analyze to `_myTransformerArea`

Change the Mean Attribute from `AverageParkArea` to `AverageFeatureArea`

Set other parameters that might be of use, for example:

- Minimum Attribute: `MinimumFeatureArea`
- Maximum Attribute: `MaximumFeatureArea`
- Median Attribute: `MedianFeatureArea`



10) Check Availability

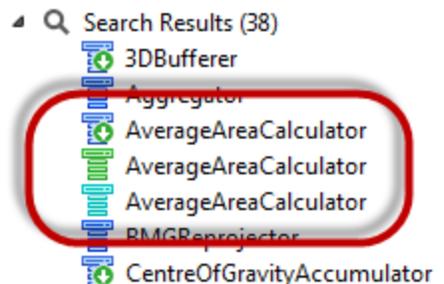
Back in the original workspace, search the Transformer Gallery for the word "average".

There should now be two entries in the gallery; one representing the embedded version, one representing the linked. You can tell the two apart by their color. There may also be a third entry linked to a copy of this transformer on the FME Store.

In any other workspace you would only ever see the linked (and store) version.

When you place a new instance of the linked transformer, you should find that the insertion mode is embedded by default.

Try the new transformer on any other dataset that contains polygon features.



Schema Mapping

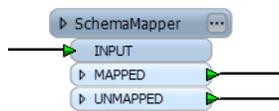


Schema Mapping is a key part to any FME workspace and transformers exist to assist with this task.

Many transformers described elsewhere, such as the *AttributeRenamer*, are useful for schema mapping, but the one with the most wide-ranging functionality is the *SchemaMapper* transformer.

SchemaMapper Transformer

The *SchemaMapper* transformer is specifically designed for mapping the source schema to the destination.



This transformer works by replacing the connecting lines of manual Schema Mapping with a user-defined external lookup table of mappings, in a format such as CSV or Excel.

The lookup table can include both Feature Mapping and Attribute Mapping, and also include 'where' clauses for use in the *SchemaMapper* wizard.

For example, a typical lookup table may look something like this:

```
SourceType,NumberOfLanes,DestinationType,SourceAttr,DestinationAttr
Roads,1,Road,Road_ID,Road_ID
Roads,2,Highway,Road_ID,Highway_ID
Roads,4,Freeway,Road_ID,Freeway_ID
```

...where a single Feature Type (Roads) is being mapped to different destination types (Road, Highway, Freeway) depending on the value of the NumberOfLanes attribute (field).



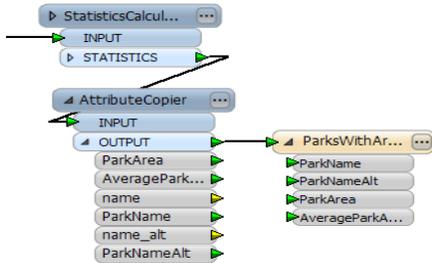
Example 30a: Schema Mapping with the SchemaMapper	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	To update the Grounds Maintenance workspace (session 2) to guard against source data changes, and allow schema mapping edits to be performed outside of FME Workbench.
Demonstrates	Schema Mapping
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example30aBegin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example30aComplete.fmw

This example continues the grounds maintenance workspace.

Ownership of the workspace is to pass out of your hands and into the hands of the Grounds Maintenance team themselves. Because that team is not comfortable using FME Workbench, you need to set up a solution they can edit when the source data schema changes.

1) Start Workbench

Open the workspace *C:\FMEData\Workspaces\DesktopManual\Example30Begin.fmw*



At the moment an *AttributeCopier* transformer manages schema mapping of the attributes name and name_alt to ParkName and ParkNameAlt

To avoid the need to edit this transformer (if the source attribute schema changes) this mapping should be entered into a lookup table, and the transformer replaced with a *SchemaMapper*.

2) Create Lookup Table

Open a text editor such as Notepad, UltraEdit or VI. Enter the following

comma-separated text:

```
ReaderAttribute,WriterAttribute
name,ParkName
name_alt,ParkNameAlt
```

Save the file as *ParksLookup.csv*

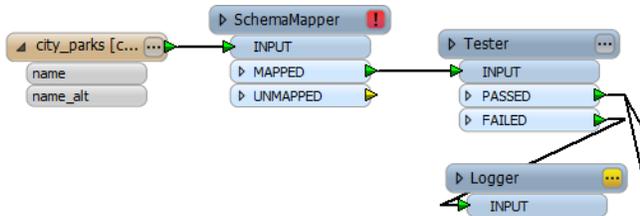
3) Place SchemaMapper Transformer

If the source data were to change there would be two problems:

- The attributes would no longer map correctly to the destination schema
- The transformers using pre-defined source attributes would no longer function

The solution is to replace the existing schema mapping with a *SchemaMapper* transformer, but also to carry out the mapping as the first step in the workspace.

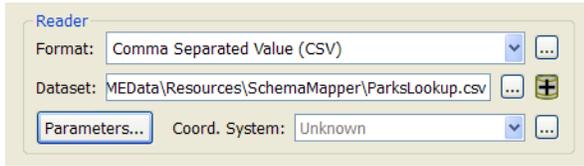
In the FME workspace, place a *SchemaMapper* transformer at the very beginning, before the data is split off into two streams with the *Tester*:



4) Adjust SchemaMapper Parameters

Open the *SchemaMapper* parameters dialog (actually, it's more like a wizard).

In the first dialog, set the format to Comma Separated Value, and select the CSV file created in step #2.



Check the parameters for this 'Reader' to ensure the first line is recognized as a header (i.e. put a check mark against 'File Has Field Names').

Click the 'Next' button to continue. Because, in a CSV dataset, there is only one 'table' of data, the wizard will skip a table selection dialog that could be there for other forms of data, and go directly to a final dialog.

5) Adjust SchemaMapper Parameters - 3

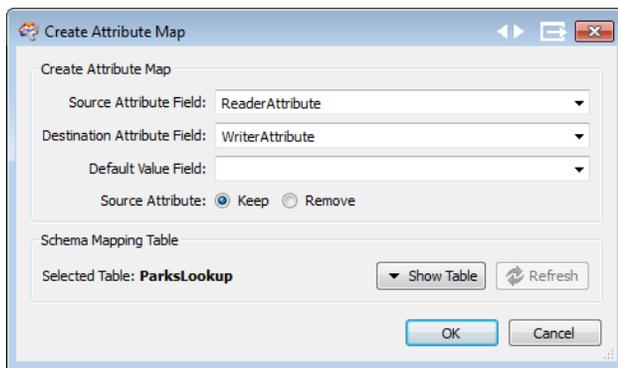
The third (and final) dialog is where schema mapping is defined.

Select Add > Attribute Map from the action button.

Select ReaderAttribute as the Source Attribute Field.

Select WriterAttribute as the Destination Attribute Field.

Default Value Field can be left empty.



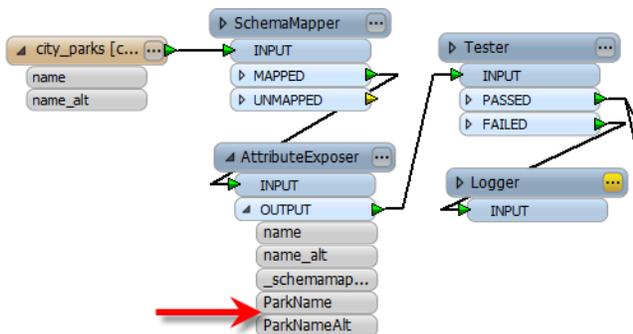
Click OK to accept the mapping, and then click the Finish button to continue.

6) Place AttributeExposer Transformer

The attributes have been renamed by the *SchemaMapper*, but have not yet been exposed to the workspace. This can be achieved with an *AttributeExposer* transformer.

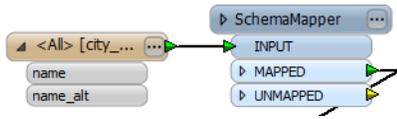
Place an *AttributeExposer* transformer after the *SchemaMapper*.

Type in the attributes ParkName and ParkNameAlt to expose them.



7) Set Merge Filter

Just in case the feature type name changes in the source data, set a merge filter of * in the reader city_parks feature type. See Chapter 4 if you need a reminder of how to achieve this.



For this case, no *SchemaMapper* feature type mapping is required, since there is only a single reader feature type, and only one writer feature type for the data to be written to.

8) Use Exposed Attributes

As it stands, if the source schema changes, transformers would still no longer function. They need to be changed to use the newly-exposed, renamed attributes.

Open the *LabelPointReplacer* parameters, and change the attribute used for labelling from name to ParkName.

Open the *Tester* parameters, and change the left value attribute from name_alt to ParkNameAlt.

Open the *Aggregator* parameters, and change the group-by from name to ParkName

9) Remove AttributeCopier

The *AttributeCopier* transformer is now redundant and can be removed.

Notice how the writer attributes still have a green connection, even once this transformer is gone.

10) Save and Run Workspace

Save the workspace and run the workspace to prove that the output is still as expected.

Example 30b: Advanced Task	
Finished Workspace	None

11) Inspect Changed Data

As an advanced task, let's assume that at some point in the future the data has changed. You can inspect the new data at C:\FMEData\Data\Parks\ParkUpdates\city_parks.tab

The source (ReaderAttribute) attributes are now called *NameOfPark* and *AlternateNameOfPark*

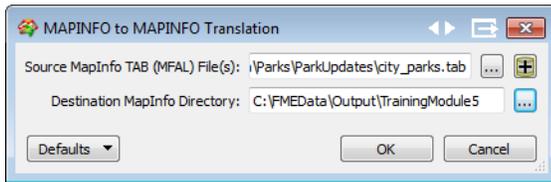
12) Fix Translation

Open the lookup table from step #2 and change the source attributes from name and name_alt to NameOfPark and AlternateNameOfPark (be sure to get the case of each letter correct)

13) Run Workspace in FME Quick Translator

Start the FME Quick Translator and run the workspace as an end-user might.

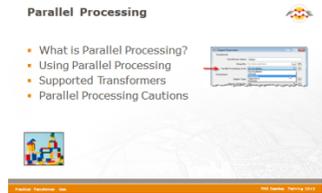
When prompted for the source data choose: C:\FMEData\Data\Parks\ParkUpdates\city_parks.tab



Inspect the output to prove that the translation has worked correctly.

This shows that the translation has been updated to handle new data, without having to make changes to the workspace itself!

Parallel Processing



Parallel Processing is a way to make transformations work more effectively on a multi-core computer.

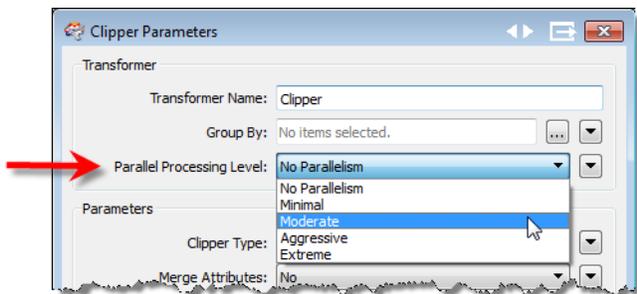
What is Parallel-Processing?

Each FME translation is usually a single process on a computer. Parallel processing is when several simultaneous processes are used. The fact that they run simultaneously means the translation will theoretically run several times quicker than it used to.

The idea is designed to make use of multiple-core processors that are so prevalent these days.

Using Parallel Processing

Each transformer that is capable of handling parallel processing will have a Parallel Processing Level parameter on it. The author can choose levels of aggressiveness from “No Parallelism” up to “Extreme”. Each level specifies the maximum number of processes to run at any one time:



Features are assigned to different processes using a Group-By setting. Here each *AreaOnAreaOverlay* process will work on data with a matching country value:



When a user sets a group-by and chooses to parallel process, each group gets run as a separate process. Because each group is a separate entity anyway, there is no need to communicate between processes.

If there are more groups than processes, then data will wait to be processed: there won't ever be more processes than the maximum specified.

Parallel Processing Parameter Values

The values for the parallel processing parameter are obviously quite subjective, but do map to a particular number of cores or processes, like this:



Parameter	Processes
No Parallelism	1
Minimal	Cores / 2
Moderate	Cores
Aggressive	Cores x 1.5
Extreme	Cores x 2

So (for example) on a quad-core machine, minimal parallelism results in two simultaneous FME processes ($4/2 = 2$).

Extreme parallelism on an 8-core machine results in 16 simultaneous processes ($8 \times 2 = 16$).

However, there is also a hard cap for each different level of license.

License Type	Process Cap
Base Edition	4
Professional Edition	8
Other Editions	16

So, a base edition license will never start more than four processes at a time, regardless of the machine type and parallelism parameter.

Supported Transformers

Not all transformers have parallel processing capability. Ones without a group-by parameter would not be able to benefit at all, and others would not benefit if they do not involve intensive processing.

A selection of some transformers with parallel processing is:

- Area/Builder
- AreaOnAreaOverlayer
- Bufferer (#22 in the top 25)
- Clipper (#10 in the top 25)
- DEMGenerator
- Dissolver (#23 in the top 25)
- DonutBuilder
- ImageRasterizer
- Intersector
- NeighborFinder
- NumericRasterizer
- RasterDEMGenerator

- SurfaceModeller
- TINGenerator



Incidentally, the AreaBuilder has been updated for FME 2013, and is now an amalgamation of its old self and the PolygonBuilder.

The PolygonBuilder, meanwhile, has been deprecated and replaced by an alias.

The AreaBuilder also has new snapping capabilities built directly into it.

Parallel Processing Cautions

Parallel processing can improve FME performance; but it can also degrade it or have very little effect, and it's worth knowing when this technique should be avoided.

Many, Small Groups

Avoid parallel processing when there are very many groups each with a small number of features. Remember, each group fires up an FME process and that takes time. For example, with 10,000 groups of 10 features, it may cost performance to start and stop FME 10,000 times than is saved in parallel processing. Conversely, 10 groups of 10,000 features would probably be more worthwhile.

Other System Resources

Other system resources – such as memory – must also be adequate for the task. An eight-core machine might be able to handle eight processes, but if memory is relatively low then the processes are just clogging up the machine and making everything slower.

Parallel processing is most efficient when the task is being offloaded elsewhere. For example multiple requests to be made via the *HTTPFetcher* might be worth parallel processing. Many processes could be started because each is such a tiny impact on the system resources.

Writing to Disk

When a translation involves writing to disk then that becomes a bottleneck that can't be solved by spawning multiple processes. Often there is a sweet spot with about 4 processes, and increasing the number beyond that makes very little difference in final translation time; the extra processes are just competing for disk time.

Really – when unsure about what will work – try a small subset of data in multi-processing mode. It will provide an estimate of whether it's worth trying it on the full dataset.

Group-By

One final issue - although not related to performance - is that the group-by and process groups are tied together. To separate them is a slightly more advanced but, in brief, the technique is to use a custom transformer. Then a group-by can be set on a transformer inside it, and a "process group" set on the custom transformer itself.

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

- Transformers can be either **Operational** or **Support** transformers. Operational transformers do the work of transforming data. Support transformers help prepare and structure data either before or after transformation.
- **Integrated** functionality allows the author to replace support transformers with tools built-into operational transformers.
- A large proportion of the most-used transformers are related to **attribute-handling**
- **Filtering** is the act of dividing data. **Conditional Filtering** is the act of dividing data on the basis of a test or condition.
- A **Custom Transformer** is a series of regular transformers condensed into a single object that is both reusable and sharable.
- Custom Transformers can be either **Embedded** or **Linked**, depending on whether they are being referenced from within the workspace or from an external file.
- **Parallel Processing** is when multiple FME processes are run simultaneously in order to make use of multiple CPU cores and provide performance benefits

FME Skills

- The ability to locate a transformer to carry out a particular task, without knowing about that transformer in advance.
- The ability to build strings and calculate arithmetic values using integrated tools.
- The ability to use common transformers, such as those for filtering and attribute management.
- The ability to turn a set of existing transformers into a Custom Transformer.
- The ability to create an external Custom Transformer definition, and to switch between the embedded and linked state within a workspace.
- Know when and how to use Parallel Processing

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 five 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.

Performance

This section describes some techniques to maximize the performance of FME translations. Performance includes both speed of translation and the amount of resources consumed.

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.



Recommendation: 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



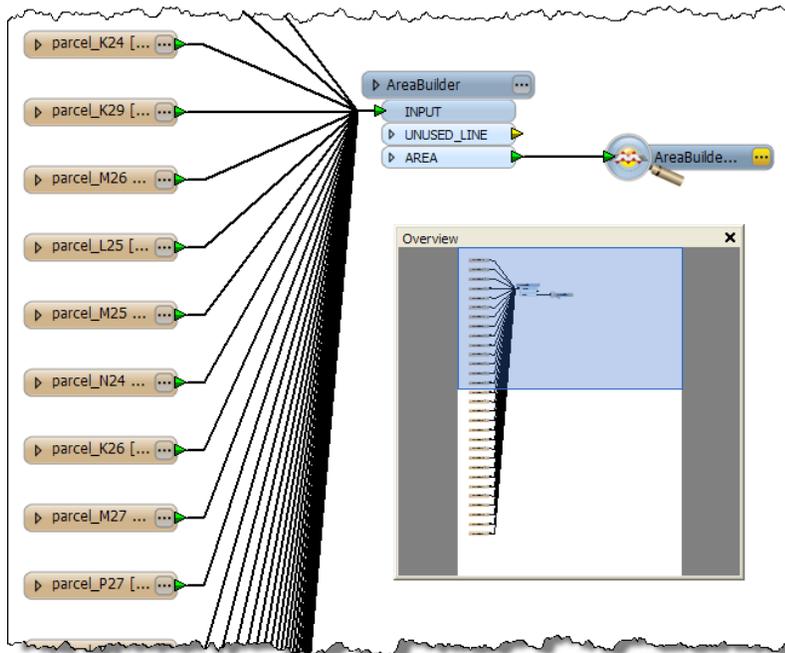
Using Workbench the right way makes for a more productive and efficient experience.

Format Methodology

Format best practices mostly involve setting up readers and writers in a clear and uncluttered manner in the workspace canvas, with some other ideas on how to avoid performance problems with “dangling” readers.

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.



Recommendation: 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.

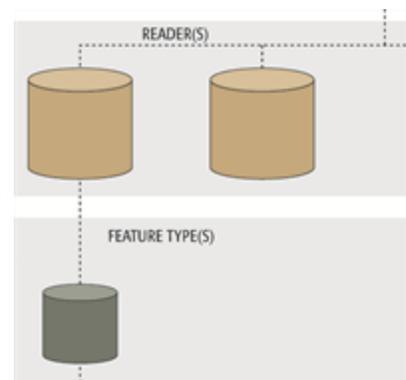
Recommendation: When source data has many Feature Types it can be difficult to manage. If all source types are being routed into the same transformer then you can (and should) replace them with one dynamic feature type.

Dangling Readers

An excess of Feature Types can be cleaned up in a workspace by removing the ones that are no longer required.

However, if all of the feature types in a particular reader (or writer) are deleted, the user is prompted to remove the reader too.

It's a good idea to remove the reader – provided it is no longer needed. A dangling reader (one without feature types) won't cause different translation results, but it is untidy and will slow down the translation.



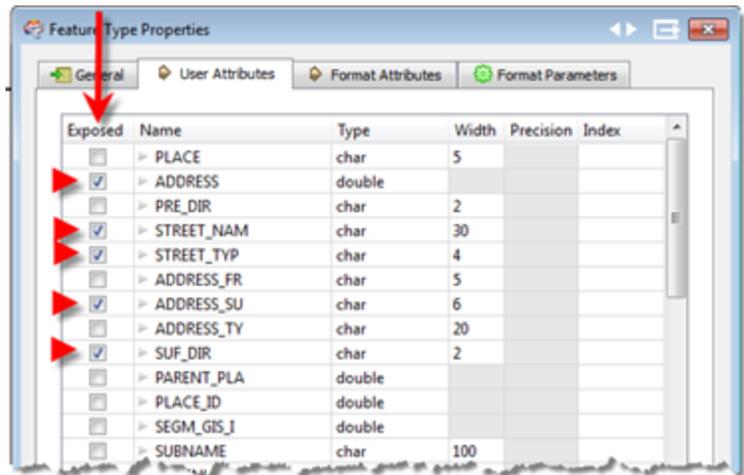
Recommendation: When you delete all the Feature Types for a particular Reader, be sure to delete the Reader too, unless you have a specific reason for keeping it.

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:



 **Recommendation:** Hide any attributes you don't need to use in the translation. It will make the workspace tidier and Workbench will even perform faster.

Format Defaults

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

However, a user may find they change a parameter regularly whenever a new workspace is created. For example, if they always connect to the same Oracle host, it would be useful to store the server name so that any new Oracle reader or writer automatically contains the correct connection parameters.

Best practice suggests that the Format Defaults options be used whenever the format settings dialog is opened; for example, through the settings button in the New Workspace dialog.

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

For example, a user of the WFS format may want to use a specific URL quite frequently.



By saving these values, whenever a WFS reader is added to a workspace its default URL is always the same as what was set.

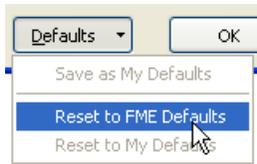
Of course, it may not be best practice to store default passwords inside a workspace. Users will want to consider the time-saving benefits against the potential security risks.



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

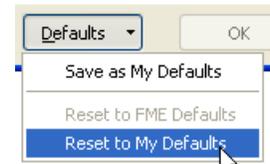
Resetting Defaults

Reset options are available to revert to the FME defaults and back again to the user defaults.



Reset to FME Defaults simply reverts the current settings back to the FME default values.

Reset to My Defaults reverts back to a previously saved set of parameter values, if the user defaults have since been changed.



To remove a default value permanently, delete the value (or Reset to FME Defaults) and then choose *Save as My Defaults* again.

Transformer Methodology

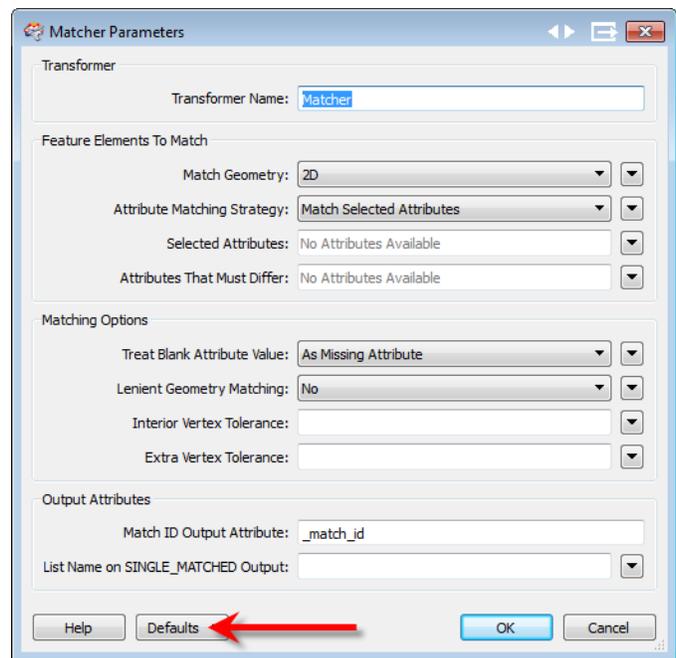
As part of the business of creating translations, there are a number of ways to use FME in an efficient and skillful manner. Some of these apply specifically to the use of transformers.

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 *Matcher* 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.



FME Functions inside Transformers

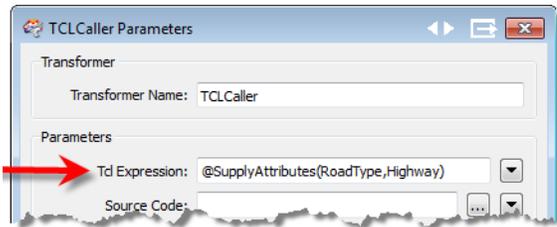
Long-time FME users, or new users with a computer programming background, often have a fascination with the underlying factories and functions of FME!

On one hand that's fine, because the more technical information a user has, the more efficient and better performing their translations can become.

On the other hand, this is a problem when a user uses an FME function where it isn't required, or isn't supported. Therefore the use of functions inside FME transformers is discouraged.

This user makes a double error. Not only do they use an FME function call inside a workspace, they don't even do it with an *FMEFunctionCaller* transformer.

It shows excellent knowledge of FME functions, and FME's built-in Tcl interpreter, but prompts the question; wouldn't it be better to simply use an *AttributeCreator*?!



 **Recommendation:** Remember, some FME Functions are now integrated into transformer dialogs, where they are guaranteed to be of use. But if you're tempted to use another FME function inside a workspace, then contact the support team at Safe who will either be able to find an alternate method, or file an enhancement request to have the same functionality supported directly.

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.

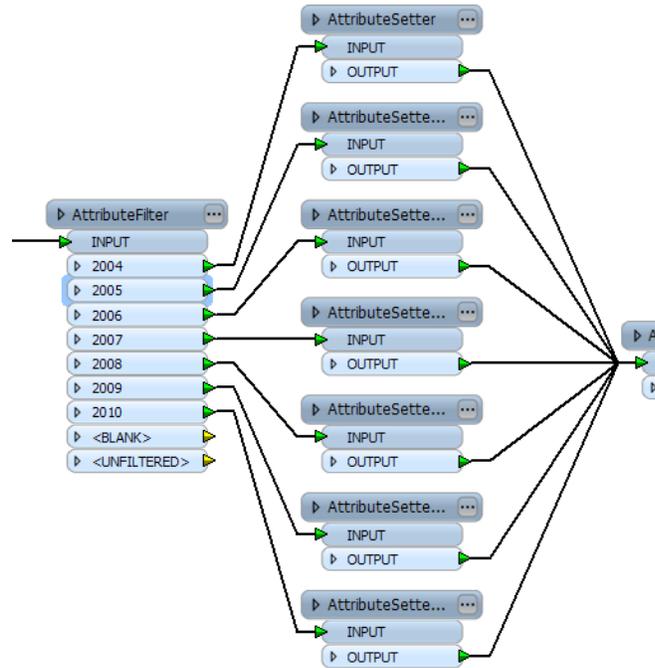
 **Recommendation:** If you're in a situation that requires the use of two workspaces to carry out one translation, then reconsider your decision and consult other FME users for advice.

The Right Transformer

As noted before, there are usually several ways to achieve the same goal with different transformers. However, 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.



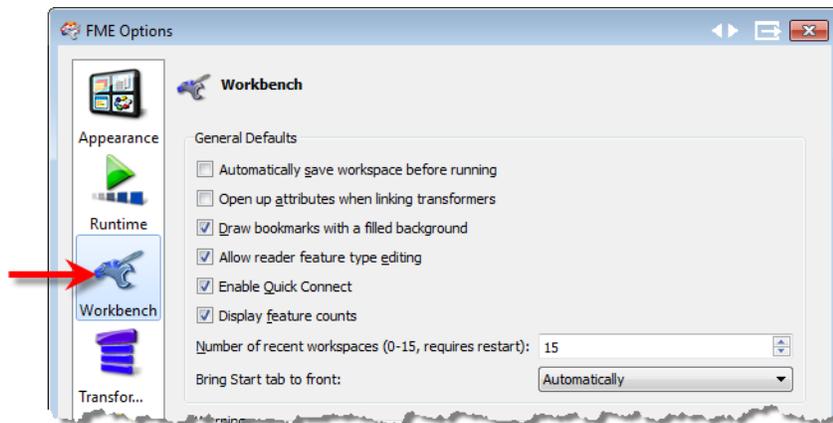
Recommendation: If you're unsure as to what transformer to use, or are worried that your workspace has multiple streams of data for no good reason, then consult other users in the FME community, or the Safe Software support team.

Workbench Options

Workbench Options are not workspace-specific and in fact don't really apply to a workspace at all. They relate to settings that affect user operation of Workbench itself.

For that reason they can be used to improve how a user works with FME, and so assist best practice.

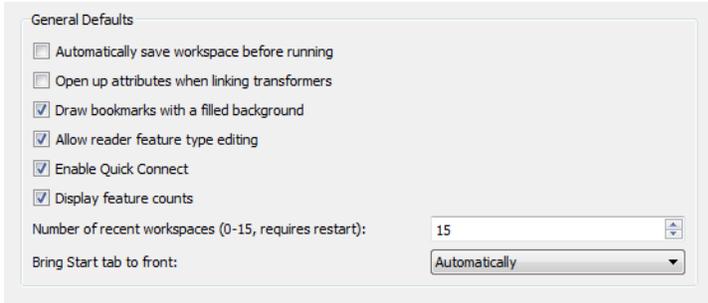
These settings can be edited in the FME Options dialog, which is opened using Tools > FME Options > Workbench on the menu bar.



Workbench options include the following.

General Defaults

Because these options are oriented towards beginners, a user may wish to change some options after they become more experienced.



- ### Automatically Save Workspace before Running

When activated, this option causes the workspace to be saved each time it's run.



Recommendation: Turn off the FME option “Save Workspace Before Running”. This sounds like a potentially useful precaution, but you'll often want to test an update without overwriting the original workspace.

- ### Open Up Attributes When Linking Transformers

When activated, this option causes transformers to expand and show their attributes when another object is connected to them.



Recommendation: Turn off the FME option “Open Attributes when Linking”. The times you won't want the transformers expanded far outweigh the times you will.

- ### Draw Bookmarks with a Filled Background

When activated, this option causes bookmarks to be drawn as a solid (see-through) fill.



Recommendation: Turn on the FME option “Draw Bookmarks with a Filled Background”. It adds better visibility to bookmarks.

- ### Allow Reader Feature Type Editing

The ability to edit source Feature Type schemas is not visible by default because of the dangers inherent in changing this part of a translation. A source schema is (in effect) a filter on the source data, and so changing it can affect how a Reader operates.



Recommendation: Turn off the FME option “Allow Reader Feature Type Editing” and only turn it on when you have edits to make. This ability poses some dangers to a translation, and turning it off prevents accidental edits!

- ### Enable Quick Connect

This toggle turns the Quick Connect functionality on and off.



Recommendation: Turn on the FME option “Enable Quick Connect”. There is no particular reason to turn off this functionality.

- **Display Feature Counts**

This toggle turns the Display Count functionality on and off.



Recommendation: Turn on the FME option “Display Feature Counts”. There is no particular reason to turn off this functionality.

- **Number of Recent Workspaces**

This option specifies how many recent workspaces will be listed in the File menu. The default is 4; the maximum is 15.



Recommendation: Increase the FME Option “Number of Recent Workspaces” to the maximum. This is a personal preference more than anything else, but the ability to locate long-gone workspaces in a more efficient manner is very useful.

- **Bring Start tab to front**

This option controls how often Workbench shows you the start screen.



Recommendation: Set the FME option “Bring Start Tab to Front” to ‘Once a Day’. It’s useful to have access to the recent files list and workspace shortcuts, but usually only the first time you start FME.

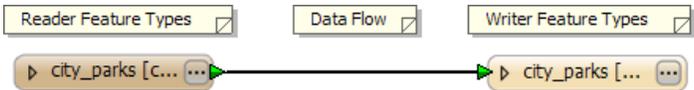
Annotation Defaults

Again, there is no right or wrong value for these options.



- **Automatically Generate Header Annotation**

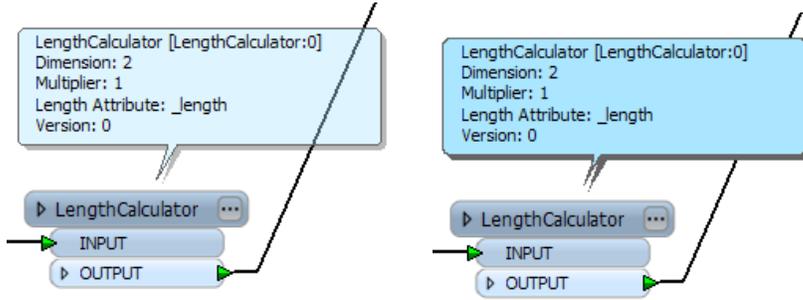
This option controls the yellow labels that appear at the top of a new workspace.



Recommendation: Turn off the FME option “Automatically Generate Header Annotation”. Anyone who’s done this course won’t really need it.

- **Use Transparent Annotations**

This option toggles annotations from semi-transparent to opaque.



 **Recommendation:** Turn on the FME option “Use Transparent Annotations”. It’s useful to be able to view objects through annotation.

■ Auto-Create Summary Annotation

This option causes transformers to be placed with summary annotations attached automatically.

 **Recommendation:** Turn off the FME option “Auto-Create Summary Annotation”. It’s good policy to use Summary Annotation, but not every transformer needs it. Just create Summary Annotation as and when required. Ctrl+Shift+K is the shortcut key to do so.

Save Options



Save Recovery Data Every...

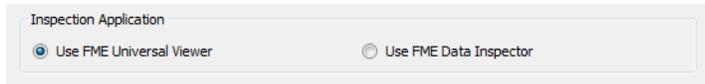
These settings define whether to save recovery data and the time period in which to do so.

These automatic workspace backups are useful in the event of a system crash.

 **Recommendation:** Go with the default for the FME option “Save Recovery Data Every”. Unless you’re working on an exceedingly immense workspace, the automatic backup is barely noticeable anyway.

Inspection Application

This single option determines whether output from Workbench for Data Inspection should be sent to the current FME Universal Viewer or the FME Data Inspector, which (for FME2013) is still a technology preview.



The FME Data Inspector is designed to be quicker and easier to use than the FME Universal Viewer, but its full range of functionality is not yet complete.

 **Recommendation:** *If you want to inspect true 3D data, or look at non-spatial data in a table view, then you'll want to use the FME Data Inspector. Otherwise the choice is yours whether to stay with the old or move to the new.*

It's anticipated that the FME Data Inspector will replace the FME Universal Viewer for FME2014.

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 zoom scale bar to 100% (new for FME2013)
- Double-click anywhere on the zoom scale bar (new for FME2013)
- Double-click with the mouse wheel anywhere on the canvas.

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



Canvas Attributes

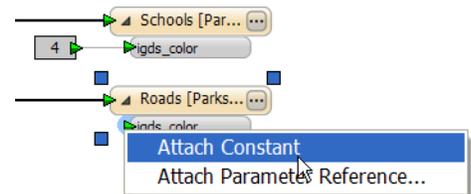
Not many users know, but there is a way to set writer attribute values directly within the canvas, rather than through a transformer. This is done by either a Constant or a Parameter Reference.

Constants

Constants are created by right-clicking an attribute and choosing 'Set to Constant Value' or by right-clicking the port itself and choosing 'Attach Constant'.

Used this way it's like a Feature Type level format attribute; that is, it applies to all features that get written to that feature type.

Here data is being written to two MicroStation feature types, and the format attribute `igds_color` set to a constant value.

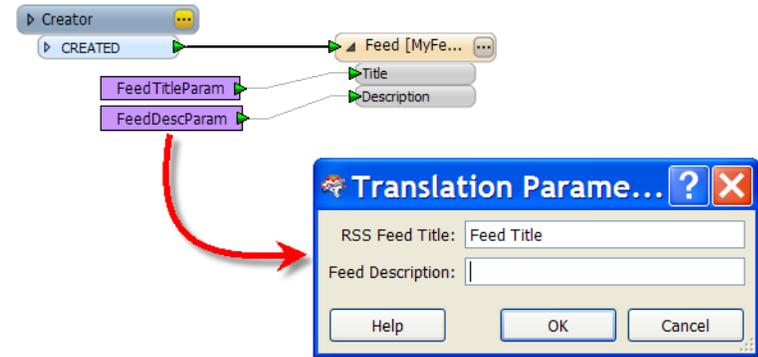
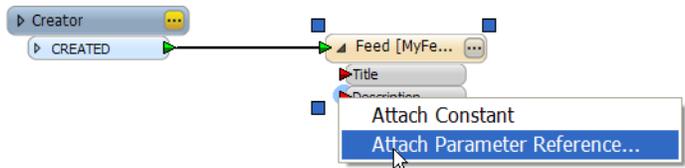


Recommendation: Use Constants sparingly. It better visualizes what is happening, but connections are more easily disrupted. Instead, use an AttributeSetter transformer. For format attributes, look for a transformer designed to set them; for example a DGNStyler, FeatureColorSetter, ArcPropertySetter, or KMLViewSetter.

Parameter References

Like a constant, a published parameter can be attached directly to a writer attribute. These are called Parameter References and are for use when the destination attribute must be a constant and user-defined value.

Here an author is writing a GeoRSS feed, and wishes the end-user to set the feed title and description at run time.



The Parameter Reference option opens a dialog in which to select the correct parameter.

Now when the workspace is run (with 'Prompt and Run') the user is prompted to enter values that will be used for these two fields.

Recommendation: Use Parameter References sparingly. Integrated Parameter Handling makes it less likely than ever that this functionality will be needed.



Example 31: Transformer Best Practice	
Scenario	FME user; City of Interopolis, Planning Department
Overall Goal	Answer Miss Vector's questions!
Demonstrates	Methodology Best Practice
Starting Workspaces	C:\FMEData\Workspaces\DesktopManual\Example31-1Begin.fmw C:\FMEData\Workspaces\DesktopManual\Example31-2Begin.fmw C:\FMEData\Workspaces\DesktopManual\Example31-3Begin.fmw
Finished Workspaces	C:\FMEData\Workspaces\DesktopManual\Example31-1Complete.fmw C:\FMEData\Workspaces\DesktopManual\Example31-2Complete.fmw C:\FMEData\Workspaces\DesktopManual\Example31-3Complete.fmw

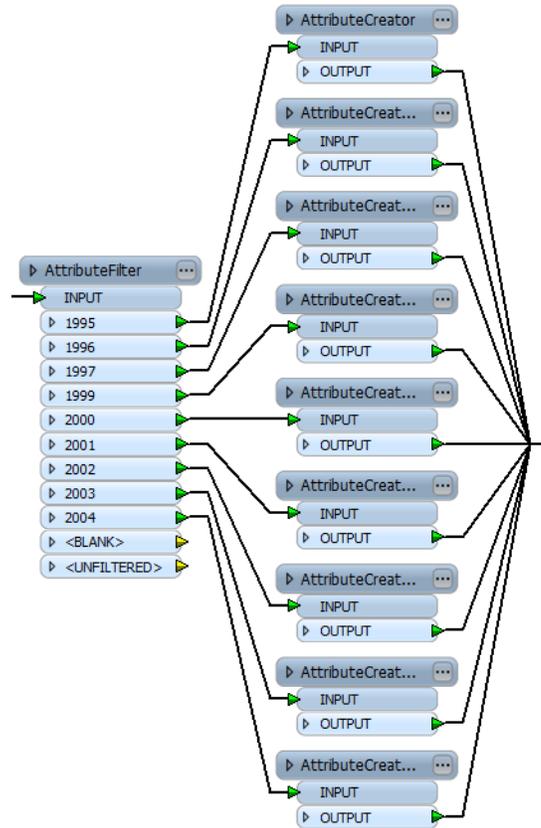


Miss Vector says...
'Can you see where these workspace authors went wrong?
Which transformer should they have used?
The answers are at the end of this manual.'

Question 1

Here the user is dividing the data on the basis of year and then setting the same attribute to a different value depending on the year.

Is this the most efficient method? Which transformer would be better – and why?

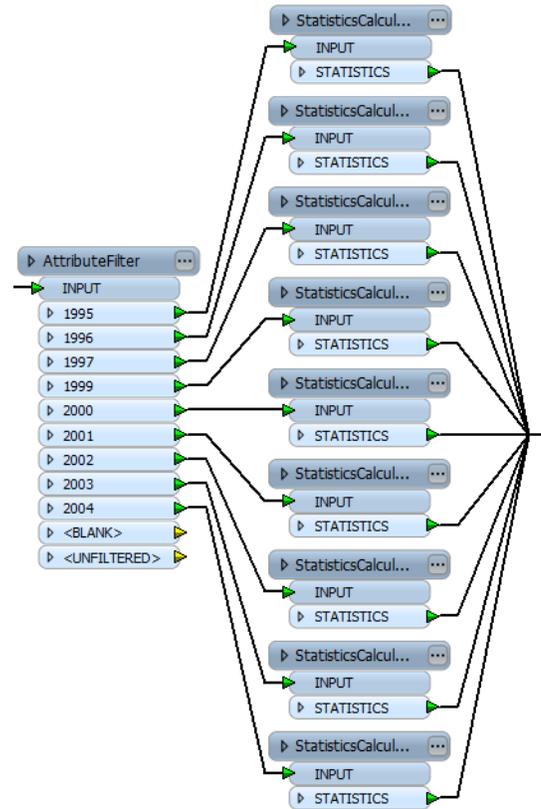


Question 2

This is very similar to Q&A 1. Again the user is dividing the data by year, but this time the user is calculating the same set of statistics for each different year.

Is this the most efficient method?

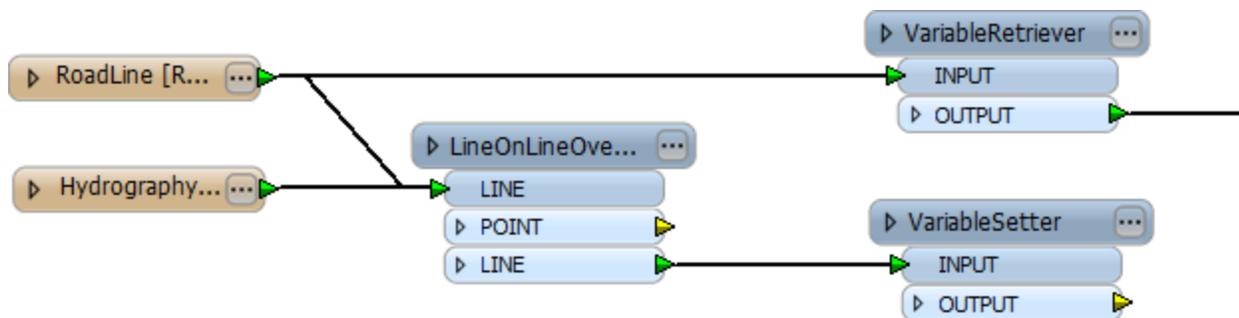
Is there another transformer that could do the same thing or could one of these transformers be used in a better way?



Question 3

In this example a user is trying to find how many times each road feature crosses a river. The LineOnLineOverlayer provides the result, which is passed back to the original features with variables. This is a valid concept – since the LineOnLineOverlayer will destroy the original geometry – but the method of using VariableSetter/VariableRetriever transformers is not.

Is there an alternative way to do this? Why is it better?



Recommendation: Always check for alternative methods of carrying out a particular process, and try to assess the relative benefits of each. As you become more experienced this technique will be easier and more natural.

Style



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

An FME Workspace Style Guide

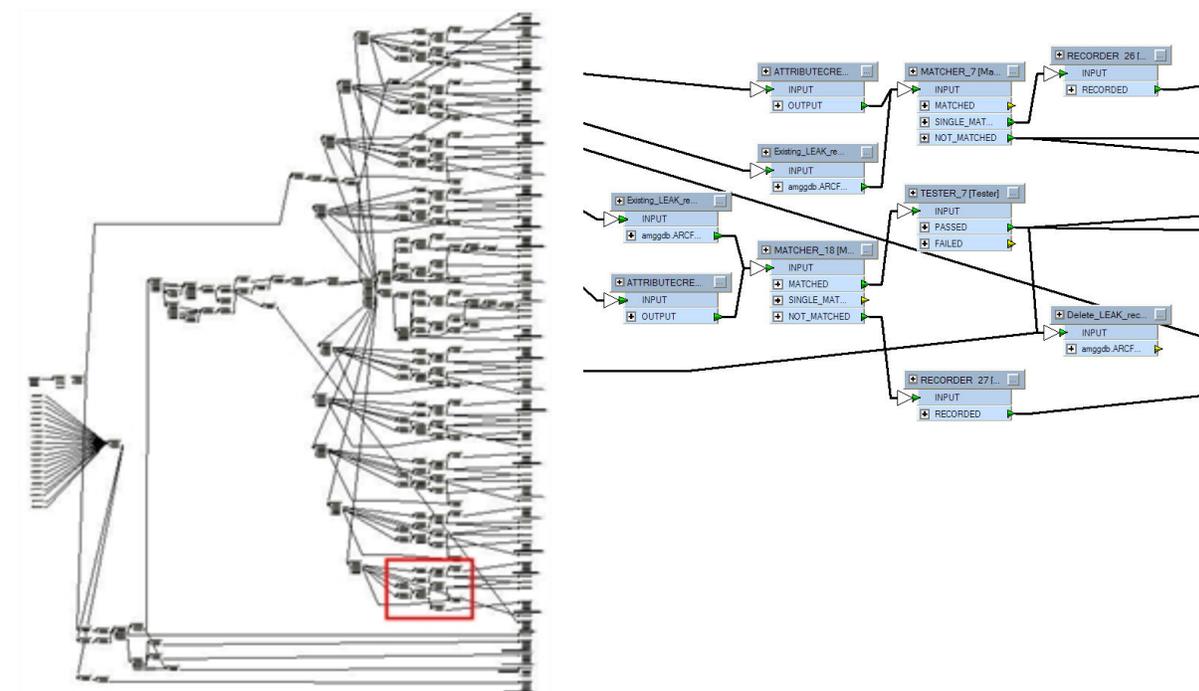
Workspaces are rarely static and used repeatedly without being edited; sooner or later the source or destination schema will change, or edits will be needed to take advantage of new or updated FME functionality.

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 is 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

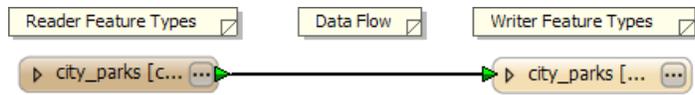
Annotation is a key method for a clear and comprehensible design.

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.

Default 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

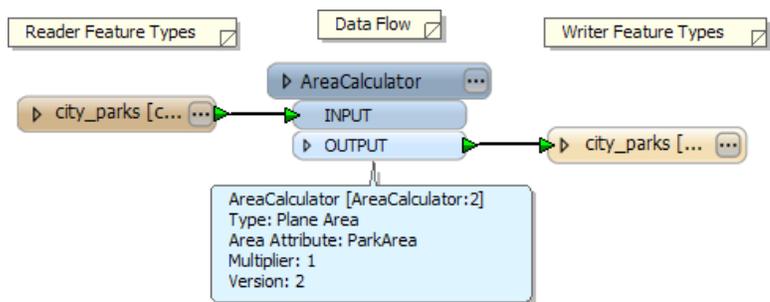
Summary annotation is an FME-generated comment that provides information about any object within the workspace. This item can be a source or destination feature type, or a transformer.



To create summary annotation for an object, right-click it and select 'Show Summary Annotation', or use the (new for FME2013) shortcut key Ctrl+Shift+K.

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.

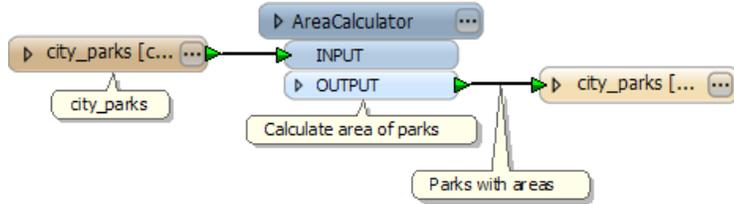


Aunt Interop says...

'Size doesn't matter! You should always use Best Practice on small workspaces and training exercises – as well as large scale projects – to get into the habit and to make them scalable.'

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 Attach Annotation.



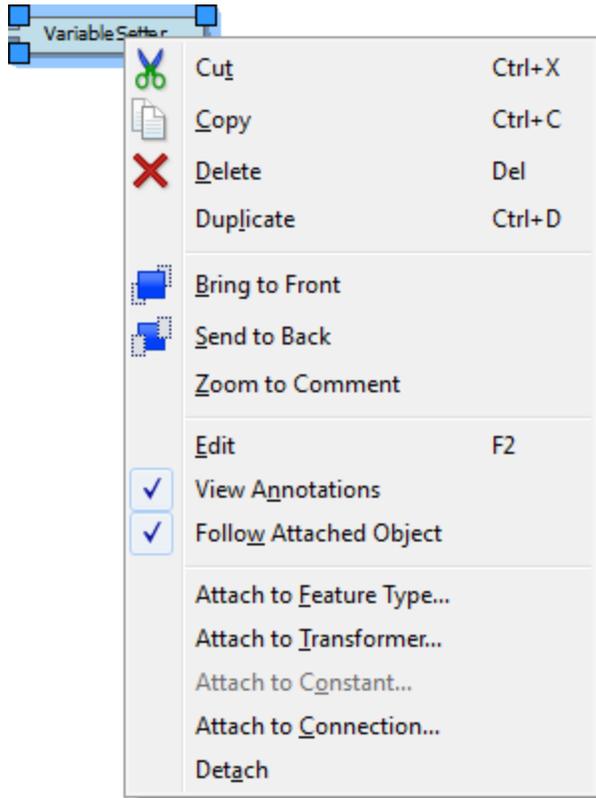
The (new for FME2013) shortcut key for attached user annotation is Ctrl+K



Recommendation: 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:



- View Annotations** – uncheck to turn off annotation display in the canvas
- 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 Constant** – attach to a specific constant
- 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 sections of transformers at a time.

Adding a Bookmark

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



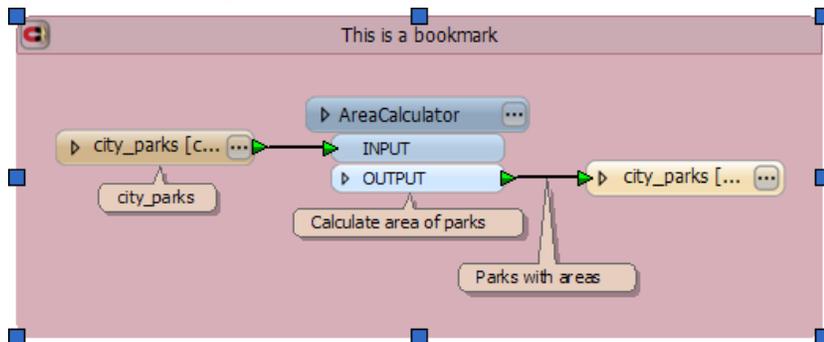
Whereas a traditional bookmark marks just a single page in a book, the FME bookmark can cover a wide area of the canvas. A single workspace can be divided into different sections by applying multiple bookmarks.



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 click it to select it and show the resizing handles. The handles can be dragged around the canvas to change the bookmark size or shape.



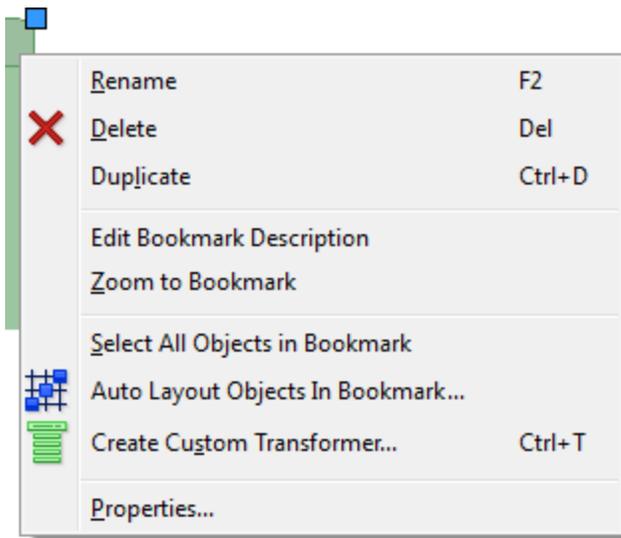
Double-clicking a bookmark frame allows the color and title of a bookmark to be edited. Double-clicking the title lets the title only be edited.



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:



Rename: Edit the bookmark title.

Delete: Delete the bookmark

Duplicate: Create a duplicate bookmark

Edit Bookmark Description: Edit the description of the bookmark.

Zoom to Bookmark: Zoom the canvas in or out to cover the same area as the bookmark.

Select All Objects in Bookmark: All objects (transformers included) in the bookmark are selected.

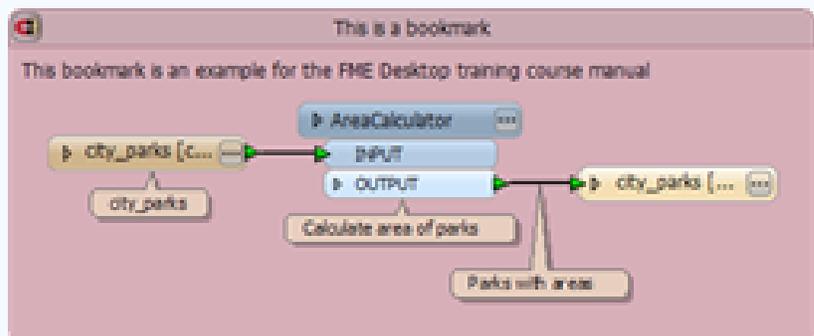
Auto-Layout All Objects in Bookmark: All objects (transformers included) in the bookmark are laid out in a new arrangement.

Create Custom Transformer The transformers in the bookmark are turned into a custom transformer.

Properties: Open a dialog to set bookmark name and color.

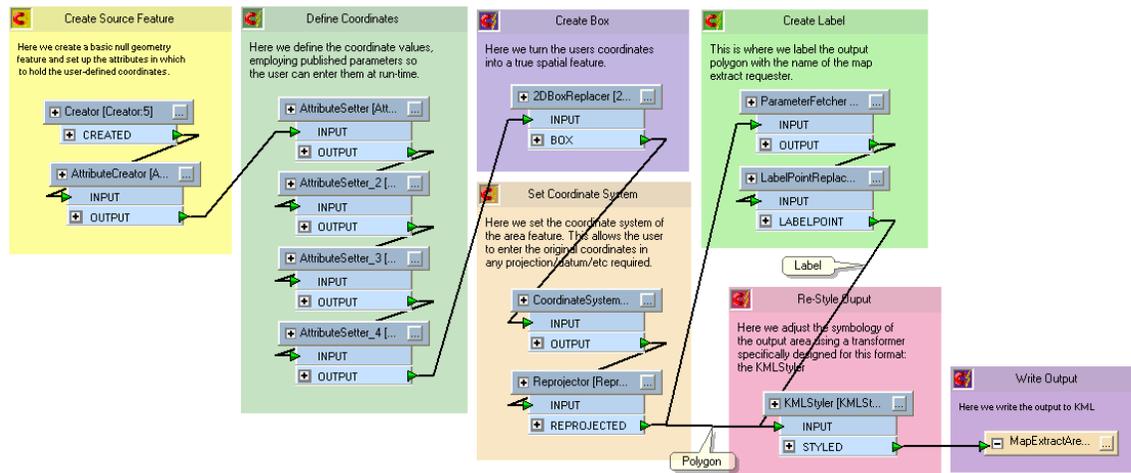


A bookmark description shows as plain text inside the bookmark.



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!

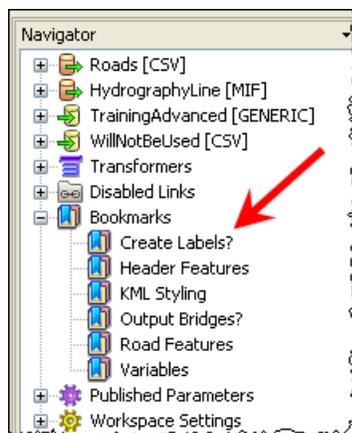


This workspace illustrates nicely how to mark up different sections of a workspace using Bookmarks.

 **Recommendation:** *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.

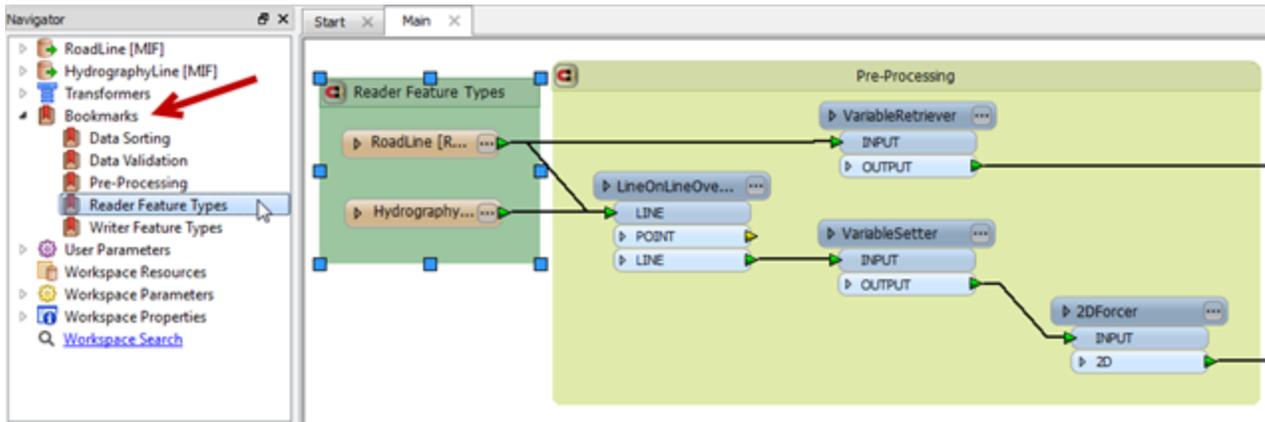


Bookmarks are listed on the Workbench Navigator.

Clicking a bookmark here selects that bookmark.

Double-clicking it zooms the canvas view to that bookmark.

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



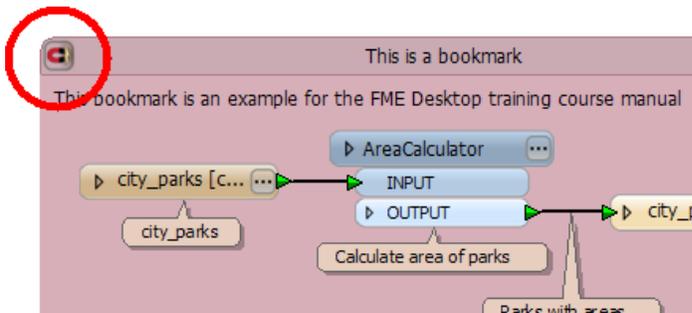
Bookmarks for Organization

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 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.

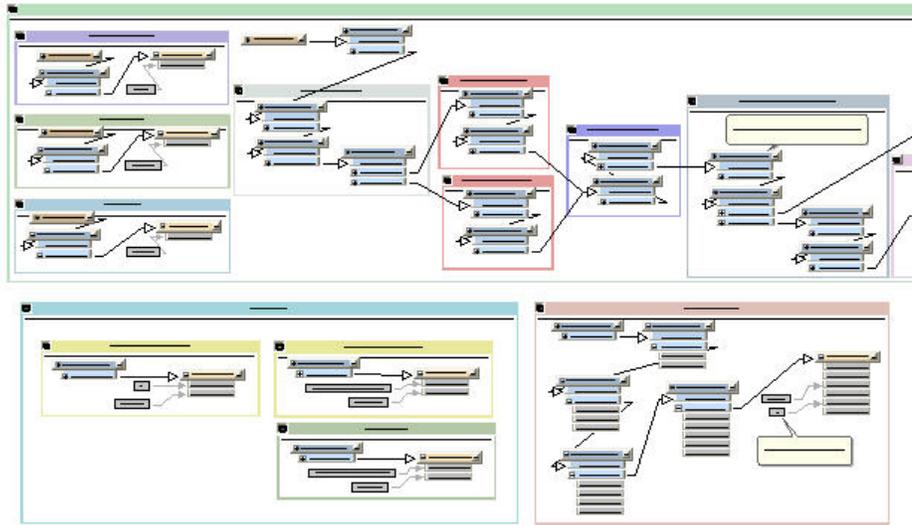
An inactive magnet allows the bookmark to be moved without disturbing the contents, which is also useful in some situations.





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.'

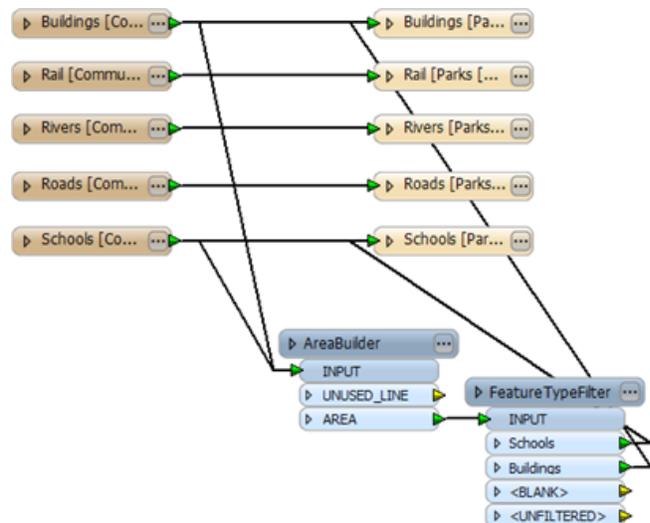


General Style Suggestions

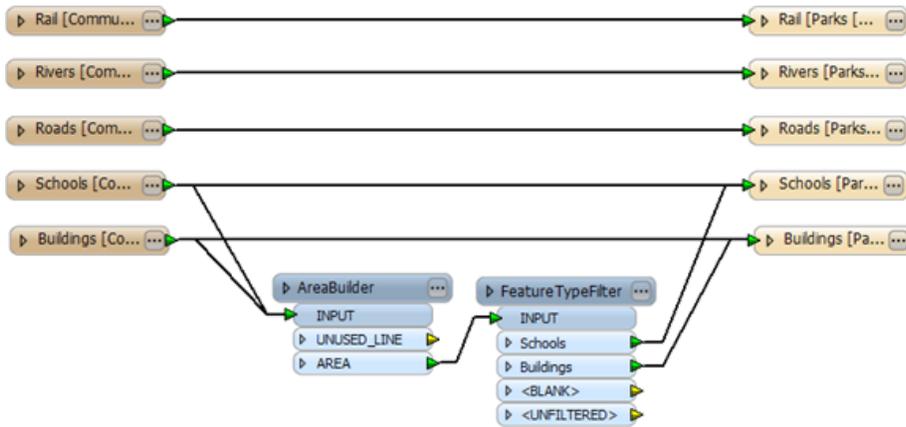
These items are general suggestions for what the style of an FME workspace should look like.

Non-Overlapping Connections

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

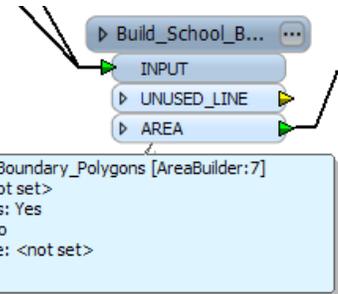
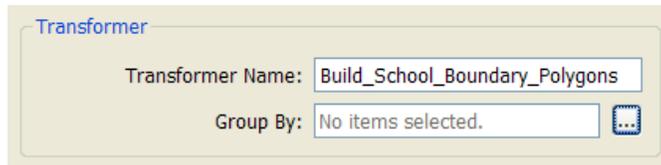
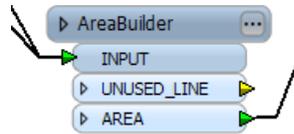


Recommendation: Take the effort to tidy your workspace. It will be so much clearer with just a little reorganization.



Renamed Transformers

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

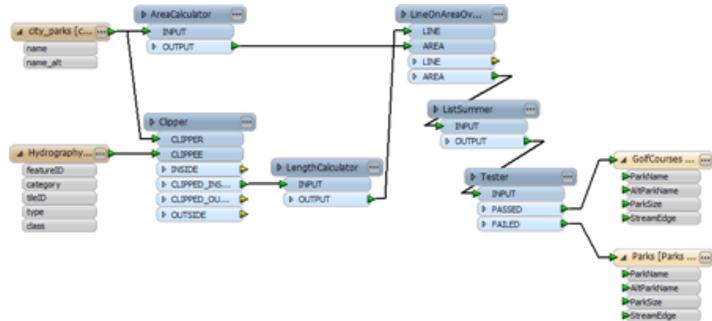


Recommendation: Rename transformers to help identify them both on the canvas and in the navigator window.

Transformer Layout

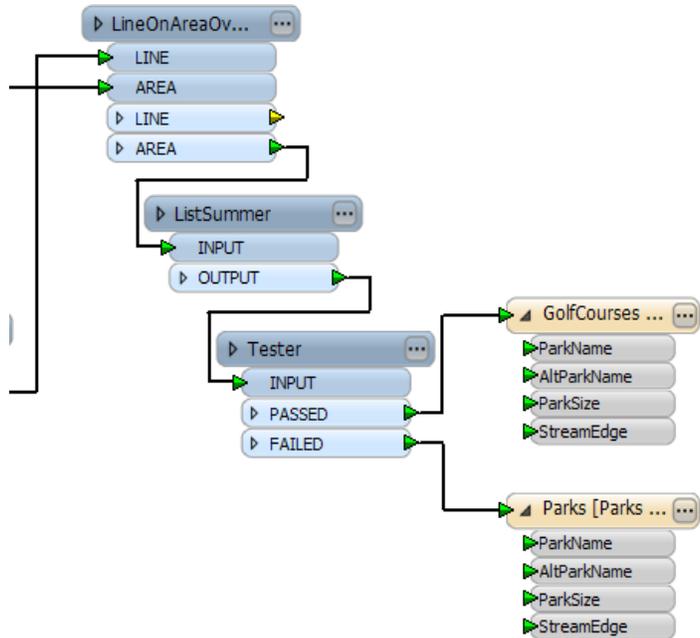
The layout of transformers – and, in particular, a consistent method of positioning – can really make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

The tools View > Show Grid and View > Snap to Grid on the menu bar help users position transformers in a consistent manner.

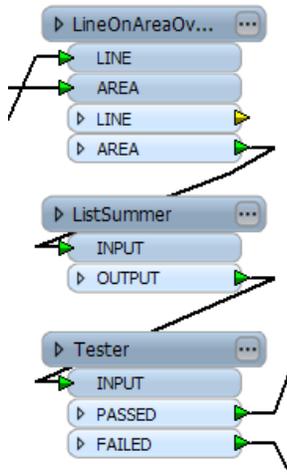


 **Recommendation:** Using straight connection lines between objects – rather than crooked - is one way to enhance the look of any workspace.

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



 **Recommendation:** 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 are easier on the eye when vertically aligned.

A toolbar button (Align Vertical Centers) will do this; and there are similar tools for aligning horizontally, or spreading objects at an equal distance apart.



Auto-Layout

Although Workbench does have an Auto-Layout tool, it's not particularly good practice to make use of it. It would be difficult to be consistent in workspace design when such a tool is applied frequently.

 **Recommendation:** Try not to use Auto-Layout; rely instead on your own design skills. If you do use it, apply it to small sections only instead of an entire workspace.



Example 32: Applying the Style Guide	
Scenario	FME user; City of Interopolis, Planning Department
Data	Unknown!
Overall Goal	Clean-up workspace and apply concepts of style guide
Demonstrates	Style Best Practice
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example32Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example32Complete.fmw

You have been given a workspace created by a previous holder of the post of FME “superhero” at the City of Interopolis, and asked to make some improvements.

However the workspace is such a mess that, before you can start work, you must try to understand what it is doing, and how, and then document it better for future users.

1) Open the Workspace

Open the workspace:

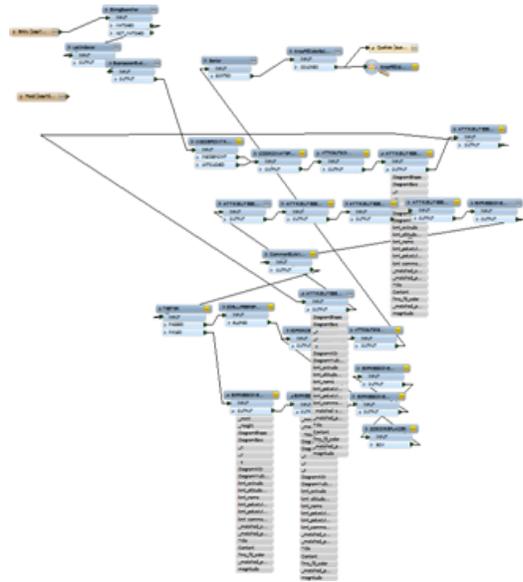
C:\FMEData\Workspaces\DesktopManual\Example32Begin.fmw

As a first step try to figure out what the workspace is meant to do. It may help to run the workspace and inspect the output.

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 won't have the same problems that you are facing.



Performance



There are a number of very simple tips that can have a dramatic effect on translation performance.

Reader Efficiency

All formats have two sets of parameters that speed up feature reading by filtering the amount of data being read.

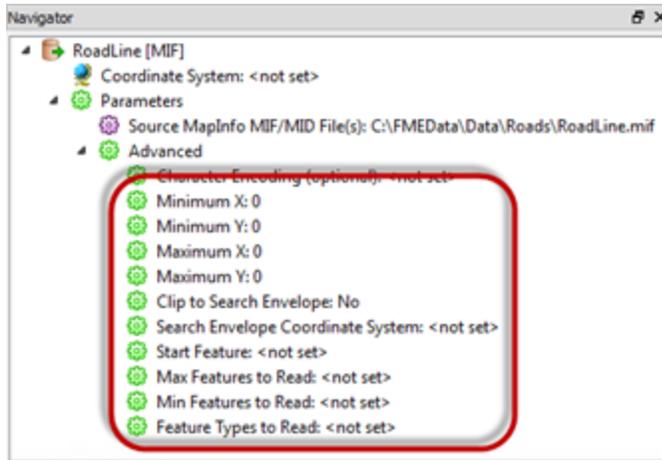


Chef Bimm says...

'How sensible is it, to go into a restaurant and order the entire menu, when you only intend to eat some of the dishes? Ordering is quicker, but way longer to be delivered, and it will certainly be more expensive!.'

The same applies reading data with FME. If you read the entire contents of a dataset, when you only need a part of that data, then you're wasting resources and slowing down the process. Not to mention putting stress on the CPU (Chef Processing Unit)

The two sets of parameters are for a search envelope and for the features to read.



Search envelope parameters allow reading just a defined area of data. They are available on every reader, but really have the most effect when the source data is spatially indexed. Then FME can query the dataset and retrieve only the data that is needed.

Similarly, the “Features to Read” parameter allows selection of which features to read.

There are also parameters to define a maximum number of features to read, and what feature to start at.

By using these judiciously, the amount of data being read can be reduced and the translation sped up considerably.

Other formats have additional clauses that can help reduce the data flow. When reading an Oracle database, for example, a ‘where clause’ can be applied. This is more efficient than reading all of the data and using a *Tester* transformer in the workspace.

Remember, any extra data that is read, also goes through the transformation process too, making the whole translation even slower.



Recommendation: When you want to filter source data, and can use a specific reader parameter to do so, it is more efficient than reading all the data then using a transformer.

Writer Efficiency

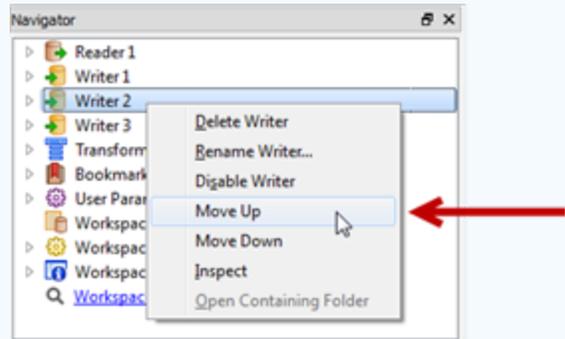
The main tip for writer efficiency is to get the writers into the optimum order.

Writers can be re-ordered by moving them up and down in the list in the Navigator window:



Writers can be moved up and down in the list using the context (right-click) menu.

Alternatively – new for FME 2013 – they can be simply dragged up and down with the mouse cursor.



The key is to ensure that the writer that is to receive the largest amount of data is at the top of the list. In brief, the first writer in a workspace starts to write data as soon as it is received. Other writers cache theirs until they are ready to start writing.

Therefore, if the largest amount of data is written immediately, lesser amounts of data have to be written to, and stored in, a cache.

This can improve performance tremendously, particularly when the translation is especially unbalanced; for example one million features go to one writer, and only ten features go to another.



First Officer Transformer says...

“Think of it like an airport. It’s more efficient when you load the busiest flights first, because it empties the terminal waiting areas quicker.”



For more information see the FME Evangelist article at: <http://fme.ly/FirstWriter>



Recommendation: For maximum performance, determine which writer is receiving the most data, and ensure it is top of the writers list.

Transformation Efficiency

There are a number of ways to make transformations operate more efficiently too.

Group-based Transformations

For performance, Feature Based transformers operate more efficiently than Group Based ones.

Remember (from Chapter 2):

Feature-based Transformation

'In feature-based transformation (or restructuring) a transformer performs an operation on a feature-by-feature basis. A single feature at a time is processed and the results of processing one feature have no bearing on the processing of other features.'

Group-based Transformation

'In group-based transformation a transformer performs an operation on a group or collection of features. All features are grouped together before processing starts and each feature can have a direct effect on how other features are processed.'

Group-based transformations are expensive in terms of performance because they consume more system resources. This occurs because the 'group' of features must be stored together (cached either to memory or disk) to be processed.

Turning Group-based Transformers into Feature-based Transformers

Obviously, when a group-based transformer is needed, then it must be used. However, some group-based transformers have parameters that, in effect, turn them into feature-based.

FME can't apply these parameters automatically because they rely on a condition being met first. However if the user can meet the condition, they can use the parameter to improve performance.

Key transformers in this category are shown in the following table.

Transformer	Parameter	Condition
AttributeAccumulator	Input is Ordered By Group	<ul style="list-style-type: none"> • When using a Group-By in conjunction with this parameter, it's assumed that the data is pre-sorted into group order.
Aggregator	Input is Ordered By Group	<ul style="list-style-type: none"> • When using a Group-By in conjunction with this parameter, it's assumed that the data is pre-sorted into group order.
Clipper	Clippers First	<ul style="list-style-type: none"> • The full set of Clipper features is assumed to arrive at the transformer before the

Transformer	Parameter	Condition
NeighborFinder	Candidates First	Clippee features. <ul style="list-style-type: none"> The full set of Candidate features is assumed to arrive at the transformer before the Base features.

For example, with the *Aggregator*, if FME can assume the data is pre-sorted into group order, then it can process each group individually and not have to cache the entire set of data.

In fact, it may be even more efficient to use a *Sorter* transformer beforehand to use a parameter in this way.



Recommendation: *Whenever you use a group-based transformer, always consider whether one of its parameters can be used in a way that helps performance.*

Transformer Selection

As already mentioned, it's likely that any particular transformation could be carried out with any number of transformers. As well as being more efficient for a particular design, some transformers are designed specifically to carry out a certain task and therefore have better performance.



Recommendation: *Your translation may benefit if you're careful in your choice of transformer—consider the benefits of choosing one transformer over another.*

Attributes and Transformation

During a translation - as noted several times - FME will either be holding data in memory or caching it to a disk. Obviously, the smaller the dataset the less memory used and the better the performance; and this includes the number of attributes.

One particular problem is carrying around spatial data as attributes. Spatial database formats - for example Oracle or GeoMedia - usually store geometry within a field in the database; for example GEOM. When FME reads the data it converts the GEOM field into FME-style geometry and drops the field from the data.

So, when reading a geometry table with a non-geometry reader, the translation could end up with the geometry stored as an FME attribute. A similar thing could happen when a workspace reads only one geometry column of a multiple geometry table.

Geometry will create very large and complex attributes, which take up a great deal of resources.

Another type of attribute to beware of is a List. A list can carry many, many sets of attributes, which is a big drain on resources. For example, use a Joiner to join a feature to 1000 records and the list will have 1,000 sets of records. This is bad enough, but if the list is exploded and all of the original attributes kept, then there will be 1,000 features each with 1,000 sets of attributes!



Recommendation: *Only carry through the translation any geometry and attributes you need for transformation or that you intend to be available on the output. Otherwise remove excess data as early as possible in the translation. An AttributeRemover helps.*

Databases Efficiency

There are also a few ways to improve performance when reading or writing databases. Most of these are common sense when there is an understanding of exactly how a database works.

For example, drop spatial indexes before doing a bulk load. If not done, the index could get re-built with every feature that gets inserted. This is one reason that dropping a table is more efficient than truncating it; the drop action also removes the indexes.

Similarly, remember that a database has a network read/write overhead. Look at the parameters for a database reader/writer (for example Features Per Transaction) and try to optimise them to give a balance between network traffic, database performance, and the risk of losing uncommitted data.



Recommendation: Make use of the *FME Readers and Writers Manual*, to find out which database reader/writer parameters can be used to enhance performance.



Example 33: Performance Best Practice	
Scenario	FME user; City of Interopolis, Planning Department
Data	Address point (GeoMedia Access Warehouse), Emergency Facilities (Idrisi)
Overall Goal	Find addresses within 1,000 feet of emergency facilities
Demonstrates	Performance Best Practice
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example33Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example33Complete.fmw

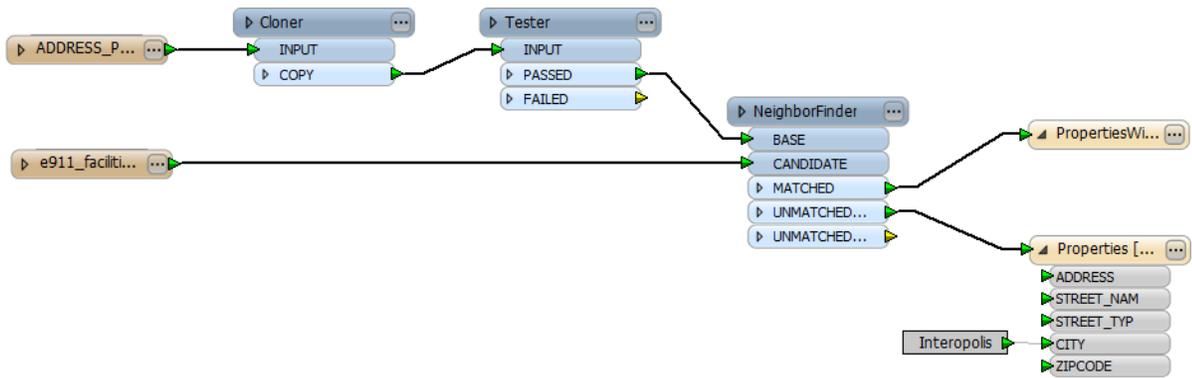
This FME author has created a workspace to find all addresses that are/aren't within 1,000 feet of the nearest emergency facility in zipcode 78723. But is it as efficient as it could be?

1) Start Workbench

Start FME Workbench and open the starting workspace.

You will see two readers (one for each source), a *Tester* transformer to filter out anything not in the correct zipcode, a *NeighborFinder* to locate the nearest facility, and two writers (one for a 'Within' dataset, one for a 'Without' dataset).

The *Cloner* transformer is there to increase the amount of data, to make this a worthwhile test.



Run the workspace. In the log file note the values for:

- FME Session Duration
- Peak Process Memory Usage

2) Fix Workspace

Make performance improvements to the workspace, using the knowledge you have gained from this chapter. Re-run the workspace to see how much time and memory you can save.

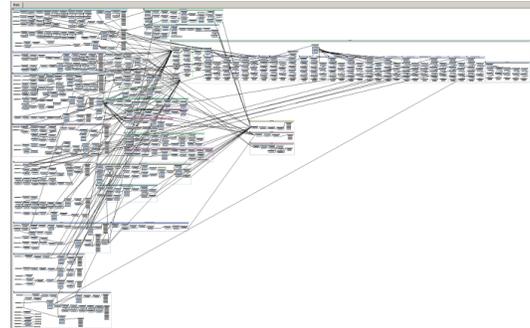
On my machine I managed a 5% saving in session duration, and a 15% saving in peak memory.

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:



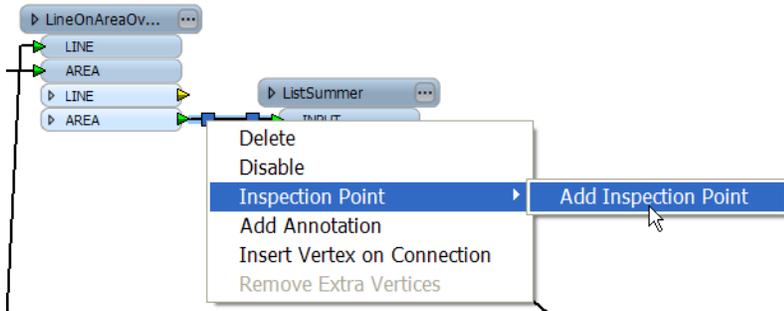
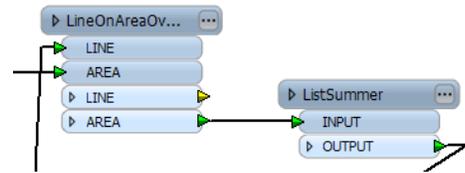
Feature Inspection

Feature Inspection 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.

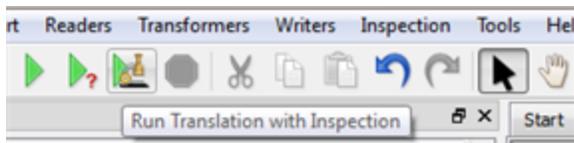
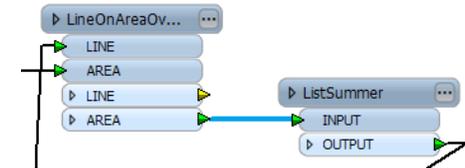
Feature Inspection is triggered by Inspection Points; workspace connections that are flagged by the user as a location where features should be inspected.

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

Right click > Inspection Point > Add Inspection Point is used to set it up.



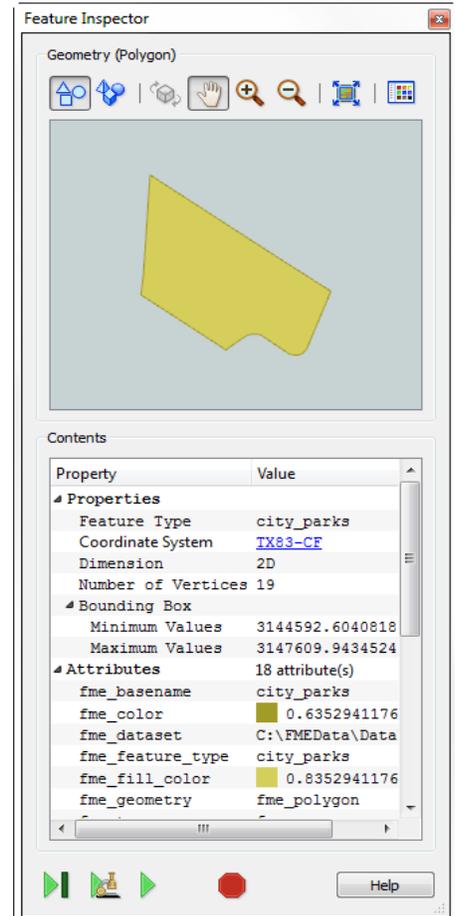
The connection is highlighted blue to denote its new status. Now the workspace is run using "Run translation with inspection"



When the first feature arrives at the inspection point, the translation is temporarily paused and information about the feature displayed in a pop-up 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.

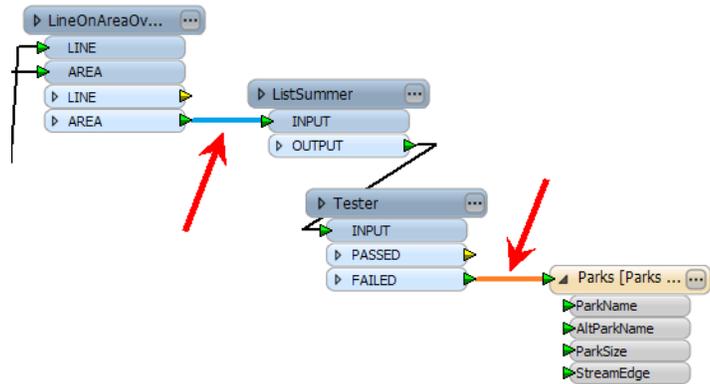
There are four buttons at the foot of the Feature Inspector window:



	Step to Next Link	This tool steps through the workspace one transformer at a time, showing the status of a feature as it is processed.
	Step to Next Inspection Point	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 inspection points.
	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 inspection point when the “Step to Next Link” tool has been used.



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



 **Recommendation:** Use Feature Inspection 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.



Example 34: Feature Inspection	
Scenario	FME user; City of Interopolis, Planning Department
Data	XML Feed of New York City Traffic
Demonstrates	Debugging Best Practice
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example34Begin.fmw
Finished Workspace	None

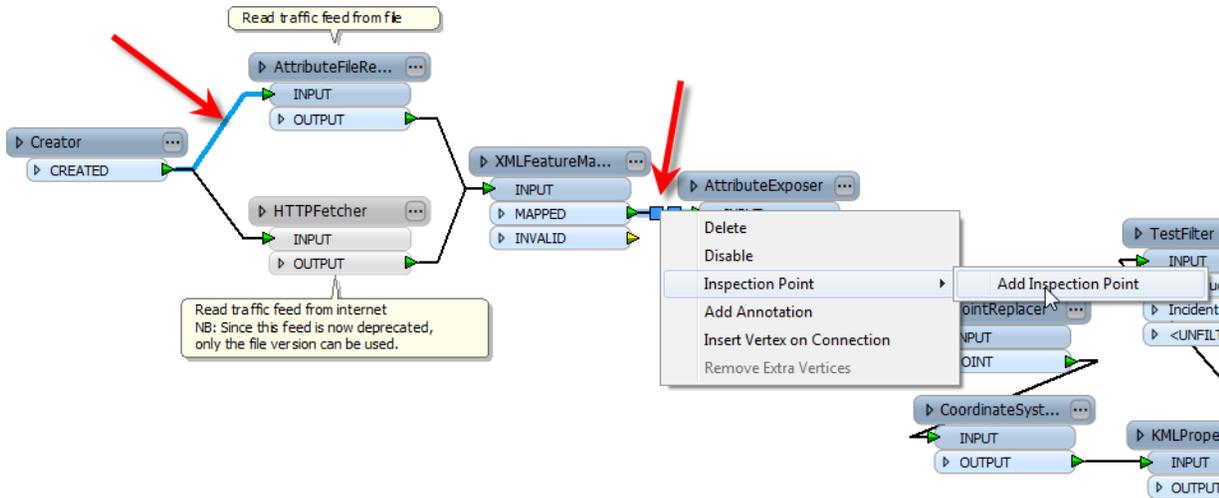
The idea here is just to open a pre-existing workspace, place some inspection points, and use Feature Inspection to see how the data is transformed as it is processed.

1) Start Workbench

Start FME Workbench and open the starting workspace.

The workspace reads from a live feed of traffic information, transforms it, and writes the output to a KML format dataset.

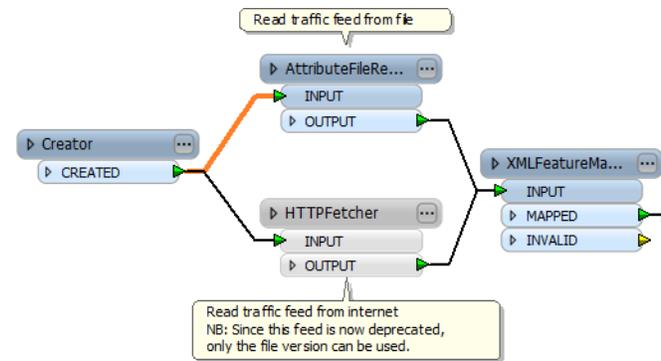
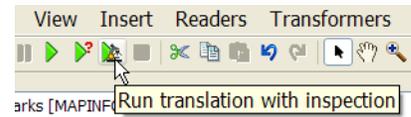
2) Add Inspection Points



Add inspection points to the first and third connections in the workspace.

3) Run Workspace in Inspection Mode

Choose the toolbar button to 'Run translation with inspection'.



The workspace will run as far as the first inspection point then pause.

The Feature Inspection window will open and show the single non-geometry feature that is formed by the Creator transformer.

4) Step through Workspace

Use the 'step to next link' button on the Feature Inspection dialog to step through to the next connection.

The *HTTPFetcher* transformer fetches the traffic data from an online feed and returns it as XML. The Feature Inspection dialog will now show the results of this operation as an attribute called *TrafficFeed*.

Contents	
Properties	
Feature Type	Creator_CREATED
Coordinate System	Unknown
Attributes	
_creation_instance	0
fme_geometry	fme_undefined
fme_type	fme_no_geom
Geometry	
IFMENull	

5) Step to next Connection

Click the 'step to next link' button again.

You will see that the *XMLFeatureMapper* has turned the XML query response into a proper set of attributes that can be used in the output dataset.

6) Step to 2dPointReplacer Connection

Click the 'step to next link' button until the *2dPointReplacer* has been processed.

You will now see that the original feature now has geometry – a single point feature – although the Coordinate System is flagged as 'Unknown'.

7) Step to next Connection

Click the 'step to next link' button.

The last transformer was the *CoordinateSystemSetter*. You will now see that the feature has its coordinate system flagged as 'LL84'.

8) Step through Remainder of Workspace

Click the 'step to next link' button until this feature reaches the writer feature type.

As the feature is processed you will see KML symbology applied to it, according to the type of item (incident or construction) and the severity.

9) Step to next Feature

By continuing to select the 'step to next link' button, the feature will pass through the workspace, eventually progress to the next feature in the pipeline, and show that being processed.



Now click the 'step to next inspection point' button.

The translation will skip to the next feature as it is output from the *XMLFeatureMapper*.

10) Continue Translation

By continuing to select the 'step to next inspection point' button, the feature inspection process will continue to skip to the next feature in the pipeline.



Now click the button 'Continue translation'. This will cause the translation to continue to the end, with no further feature inspection taking place.

Logging

The log window 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 WriterFormat. 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.

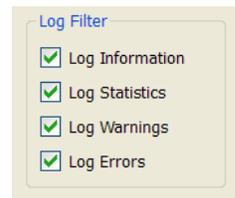


When FME functionality is implemented using a third-party library (for example, some Geodatabase writers make use of Esri's ArcObjects), information in the log window is reported directly from that product and FME has no control over the quality of the messages shown.

Log Options

Workbench has the ability to filter different message types from the log window. This is done using Tools > FME Options > Runtime.

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

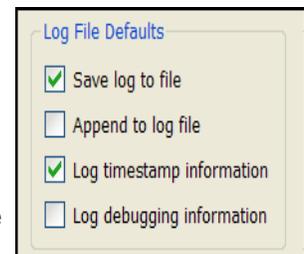


Recommendation: *Turning off STATS and INFORM log messages helps to highlight WARN and ERROR messages in the log so there is less chance of missing them; though it does feel strange to watch a translation take place without the log window scrolling past as usual!*

Log Timings

Another useful option under Tools > FME Options > Runtime 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.



 **Recommendation:** *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.*



Performance

In previous FME versions, performance could be slowed by the contents of the log window scrolling past, particularly when there are many features that all trigger the same warning.

For FME2013 the logging system has been improved and should no longer slow the translation. In fact, the translation may run so fast you're not sure if it even happened!

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.

 **Recommendation:** *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 – when it usually writes it to a spatial log file.

 **Recommendation:** *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

If the log window was closed during the translation, then any warning messages that may have passed by will have been missed; so 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. That is only likely to happen when the transformer is group-based and does actually process all features simultaneously.

Output

If the workspace writers have been redirected to an output other than a destination dataset, 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 translation!

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 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 add unwanted source feature types to the workspace.

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*.

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.



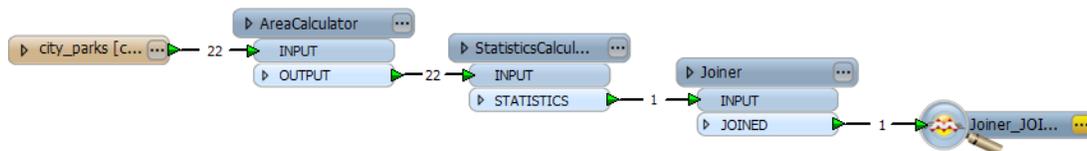
The Logger transformer has a setting called 'Log Message:' – when a feature is logged, this message appears with it. Setting a unique message helps you locate logged features by using the log search function to find the message. FME also adds the transformer name to the message, helping you to identify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

Feature Counts

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.

However, different interpretations can be placed upon the same numbers.

This example is similar to a workspace created earlier to calculate the average size of park polygon features, only this time a *Joiner* transformer is included at the end.



When the workspace is run, it fails. The question is: Where does the failure occur?

The feature counts are open to two interpretations:

- 22 features entered the *StatisticsCalculator*, but only 1 emerged. Therefore the *StatisticsCalculator* must be at fault.
- The *StatisticsCalculator* didn't get the chance to output 22 features. The first feature to emerge caused the *Joiner* to fail. Therefore the *Joiner* is at fault.

So, it's not entirely clear where the problem occurred, but at least it can be narrowed down to be fairly certain that the reading of the data and the *AreaCalculator* were not the locations of the problem. If the failure was an ERROR, then it's likely the *Joiner* is at fault.

 **Recommendation:** Use Feature counts to identify where a problem occurred.

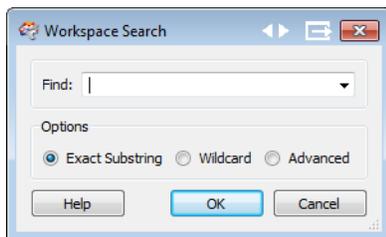
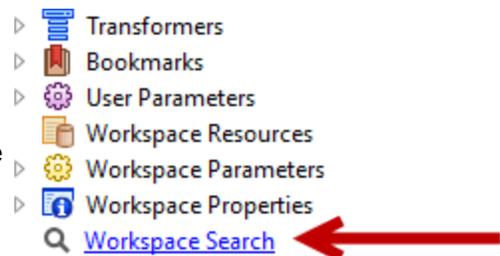


You should also be clear that these numbers don't work as well for identifying warnings. Most warnings still allow the feature in question to pass, unprocessed, so the counts do not reflect warnings as well as they reflect errors that stop the translation in its failed state.

Workspace Searching

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

Clicking this link opens up a text dialog in which to enter search terms. The results of the search include any attribute names, feature types, transformers, parameter names, and parameter values that include the search term entered.

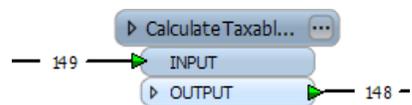


Workspace search is important for debugging because sometimes the transformer that fails does so because of a fault in a completely different part of the workspace. When that happens, the ability to search a workspace helps to track down the source of the problem.

For example, take a workspace that fails in an *ExpressionEvaluator*.

```
CalculateTaxableValue: @Evaluate -- failed to evaluate expression
`expr (111.395433278693*)' -- missing operand at _@_
```

CalculateTaxableValue is what the user named this transformer (very good), which makes it easy to find, and the feature count confirms this is the source of the error (149 features input, but only 148 output).

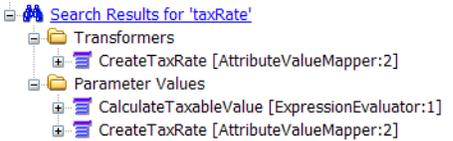


However, the problem is not in the construction of the expression (because 148 features have passed correctly) but with a missing operand called taxRate.

A quick check of the Log file proves this:

```
Attribute(string): `taxRate' has value ``
```

So the problem must lie in where taxRate is constructed. A search for that term reveals the transformer in which that attribute is created:



Therefore a reasonable starting point for fixing the problem is to check out the *AttributeValueMapper* transformer called *CreateTaxRate*.

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

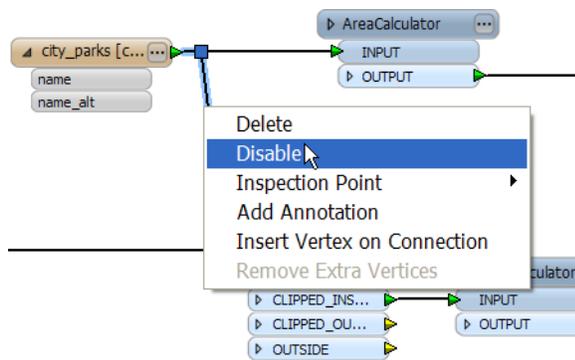
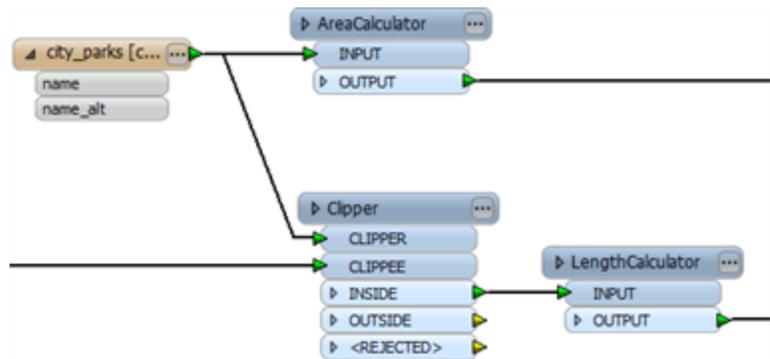
Testing Isolated Sections

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 disable connections. This renders a connection inoperative in much the same way as if it had been deleted, and no features will pass through.

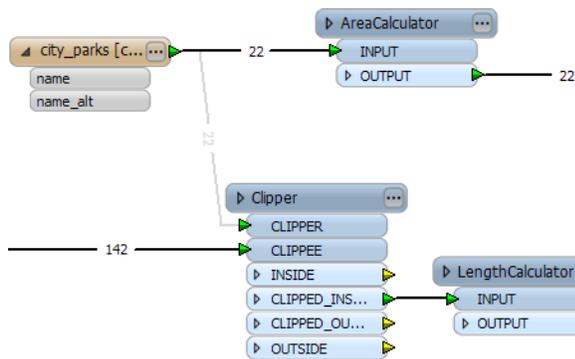
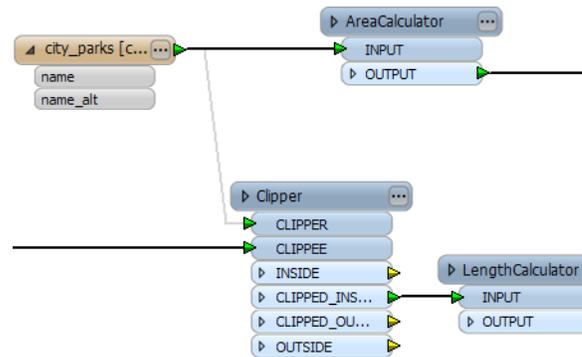
For large workspaces the disable functionality is very useful. It enables the author to isolate sections of a workspace from being executed, so that tests run much more efficiently.

Here the workspace has two outputs from the source feature type, and the user wishes to run the workspace only to one section.



Right-click > Disable disables a connection (as does the shortcut Ctrl+E).

Disabling a connection turns it grey within the workspace.



The feature count shows how many features were dropped at that point, but subsequent connections will not show a feature count.



Example 35: More Debugging	
Scenario	FME user; City of Interopolis, Planning Department
Data	GPS Road data (CSV)
Overall Goal	Debug workspace
Demonstrates	Debugging Best Practice
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Example35Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Example35Complete.fmw



On his application for FME Certified Professional status, Shirley U Jest says...

"This workspace takes a set of GPS points, converts them to road lines, and then writes the output to an Esri Shape dataset."

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 the error message.

Track down where the error occurs, trace it to its cause, and fix it.

2) Fix Translation

Track down all the other problems in the workspace (I count about seven) and be sure to fix them.

Remember to make use of:

- The FME Universal Viewer
- The Logger and Inspector transformers
- The Feature Inspection tool
- Feature Counts
- Log window WARNings and ERRORs.
- Workspace Searching
- Disabled Connections

Organization



It's great to be able to re-use components, especially when you can share them with fellow FME users.

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.

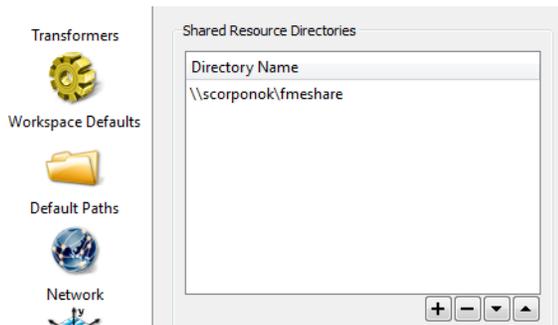
Shared Resource Directories

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.



Recommendation: 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.

Installing a Shared Resource

The simplest method of installing a shared resource is to copy the file into the required location. Note that each shared directory has a subdirectory for each type of resource.

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

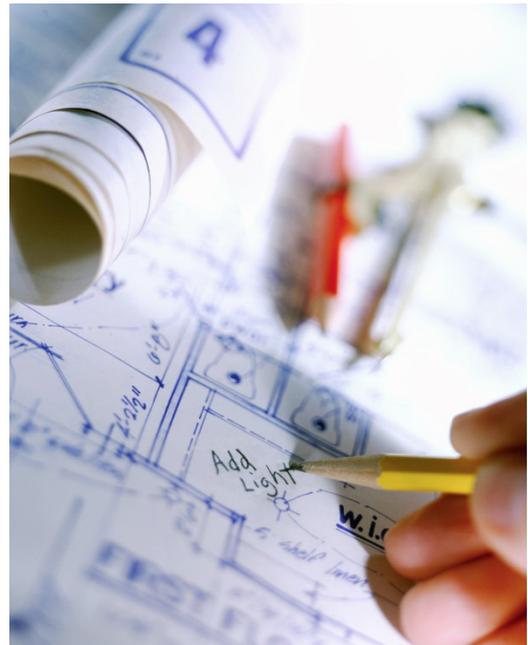
They have their own extension (*.fmwt) and their own storage folder (<user>/FME/Templates).

The most interesting thing about FME templates is perhaps 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:

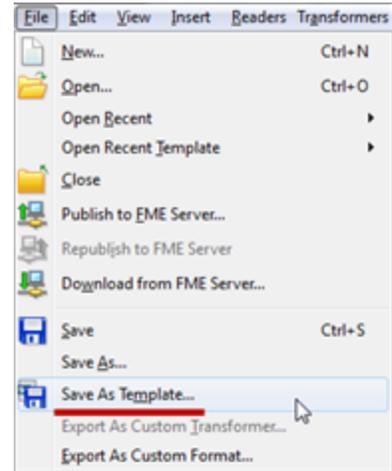
- **A Complete Translation**
Wrapping up source data and workspace inside a single file makes it a very elegant way of providing a set of related files to another user.
- **Pre-Defined Data**
When a series of workspaces are all to use the same source or destination data, a template allows the author 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**
An FME template is a great way to provide an example to a user, for them to study and copy. It can either be a complete (i.e. fully working) example, or just a piece of a workspace that they can then add to.



Recommendation: Use templates when you have a fixed design (layout, schema, or transformation) that you will use again and again in a series of workspaces.

Creating a Template

FME templates are very simple to create. Simply open an existing workspace and save it using File > Save as Template on the menubar.



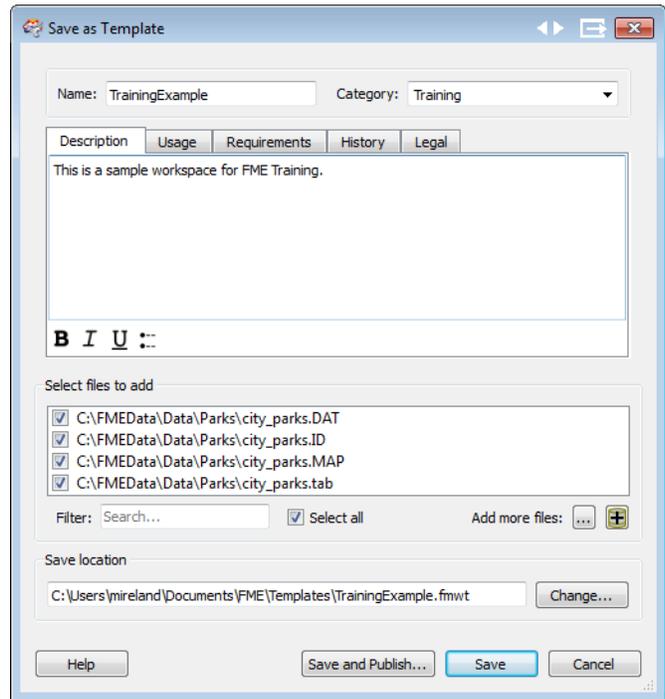
On creating a template the user is prompted to enter some information. This includes:

- A template title
- A template category
- A template description
- The data to include
- The template filename/location

The template category can be any value. It is used to organize templates within the Create Workspace dialog.

Although all source data is selected by default, it does not need to be included. Also notice how the other new fields for workspace properties (Usage, etc.) are now also added to this dialog.

Confirmation of the template creation is provided in the Workbench log window:



```

Saving as template...
Title : Training Materials
Category: Training
Location: C:\Documents and Settings\mireland\My Documents\FME\Templates\Training Materials.fmwt
    
```

```

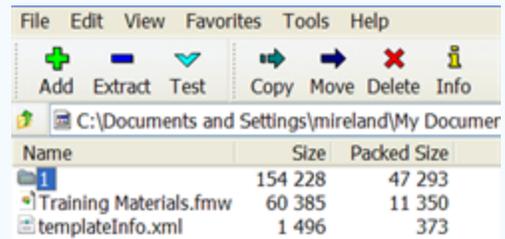
Adding files to template...
Training Materials.fmw
C:/FMEData/Data/Roads/MajorRoads.dgn
C:/FMEData/Data/Airport/airport.qlf
    
```

```

Saving as template completed successfully
    
```



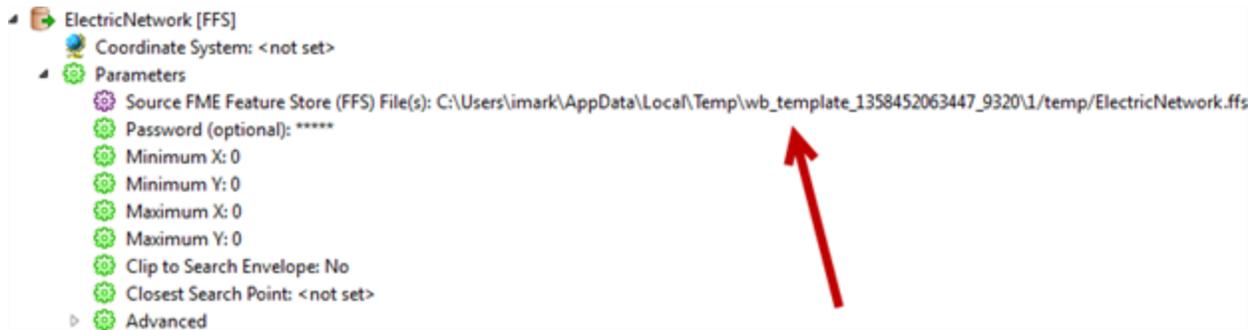
In case it is of interest, a .fmwt file is simply a compressed folder, and can be opened using any tool capable of viewing such a file.



Using a Template

Using File > Open on a template file doesn't open the template itself, but creates a new workspace that is a copy of the template workspace.

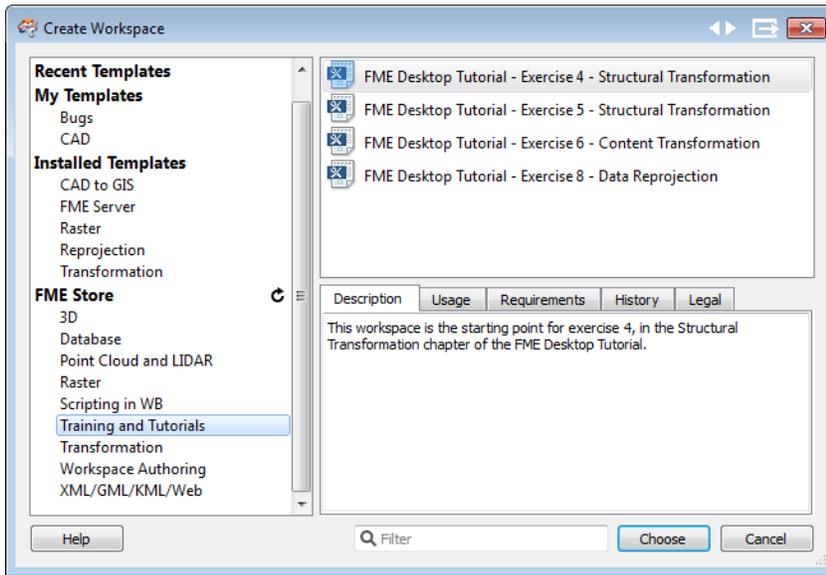
Opening a template also extracts the data and files inside the template into a temporary location:



When the workspace is saved by the user then the data is copied over into that location, and the source/destination dataset parameters automatically updated to match (the Navigator dialog will need refreshing):



Templates can also be used directly from the Create Workspace dialog.

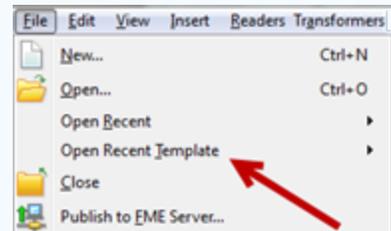


Notice there are sections for recently-used templates, “My Templates”, “Installed Templates” (examples provided as part of FME Desktop), and FMEpedia. FMEpedia templates are ones available online as workspace examples for users to download and try out.

The “My Templates” section is divided into categories defined when the templates were created.



In FME2013 the File menu has a new entry for Open Recent Template:



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:\P2013-999_PROPERTY_UPDATES

Follow this with subdirectories, such as:

- .\FMEworkspaces which could itself have subdirectories such as
 - .\includes for include files
 - .\Tcl for 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



Recommendation: 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.fmw.
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 2013-03-21 or 20130321 so files are listed in the correct order. Do not use a date such as Mar-21-13, 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_clnt'. 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:\P2013-999\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.

Example 36: 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



Example 36: Using a Template	
Finished Workspace	None

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 FMEpedia.

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 understand performance issues and tune various translation components.
- 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.

Chapter 7 - Course Wrap-Up

Product Information and Resources



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



Although your FME training is now at an end, there is a good supply of expert information available for future assistance.

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



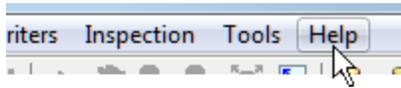
Behind FME are passionate, fun, and knowledgeable experts, ready to help you succeed, with a support and services philosophy built on the principle of knowledge transfer.

Safe Software Blog

The Safe Software blog ("It's All About Data") provides thoughts on spatial data interoperability from the folks who make it their passion.



FME Manuals and Documentation



Use the Help function in FME Workbench to access context-sensitive help, and various manuals and documentation for FME Desktop.

Community Information and Resources



Safe Software actively promotes users of FME to become part of the FME Community.

FMEpedia

This encyclopaedia is our FME community website. It provides one-stop access to all FME community resources. You'll find a wealth of FME information, including tips, tricks, and FAQs.



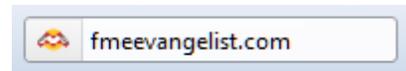
FME Talk

FME Community Answers

The FME community group (FME Talk) has 1,500 members who post FME related messages and questions and share in answering other users' questions.

The FME Evangelist

The FME Evangelist delivers inside news about cutting edge examples and the latest developments in FME functionality.

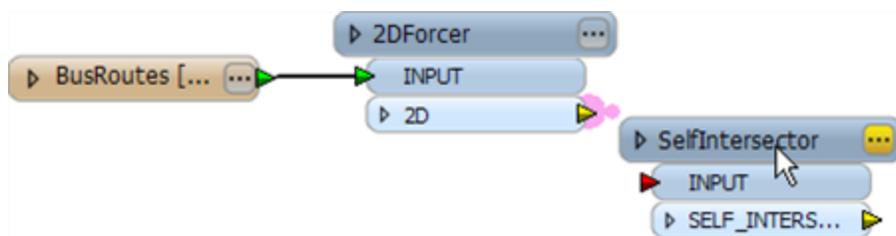


The FME Channel

This FME YouTube channel is for those demos that can only be properly appreciated through a screencast or movie.



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 Training Feedback Form

Please take a moment to fill in the survey below. Your feedback allows us to continually improve our courses to meet the needs of our customers.

Course Details
FME Desktop, Online | November 17th-18th

Attendee Information
Name (optional): Company (optional):

General Feedback

I can apply the course content to my job. Please Select... ▾	The exercises were directly applicable to the material covered. Please Select... ▾
The course length was adequate to cover the content. Please Select... ▾	The graphics and overheads were clear and concise. Please Select... ▾
There was an appropriate balance between lecture and lab. Please Select... ▾	

What did you think?

Overall course rating: Please Select... ▾	Sometimes we like to use comments for promotional purposes. Please let us know if we can use yours. <input type="radio"/> Yes <input type="radio"/> No
Overall instructor rating: Please Select... ▾	
Overall classroom environment: Please Select... ▾	
Overall usability of FME: Please Select... ▾	

Would you recommend this course to other FME users?
 Yes
 No

What was the most impressive aspect of this course?

What was the least impressive aspect of this course?

I would have preferred to spend more time discussing:

and less time discussing:

Additional Comments:

If you have comments or concerns on items not covered by the feedback form then please contact our [Training Manager](#) directly

Certificates



All FME training course attendees receive a certificate.

Congratulations!

With the presentation of your certificate of achievement, you have now officially completed the FME Desktop 2013 Training Course.



Thank You



Thank you for attending this FME training course.

All of us funky dudes at Safe Software Inc. wish you good luck with your use of FME.



Congratulations!



"As a reward for reading this far, here's a nice word search for you to carry out. There are twenty transformers hidden in the grid. But as a punishment for not working fast enough, I'm not going to tell you which transformers are included!"



U B W N J A B W W G O B O Q H R U A C R G X C S G
T H J C C Z G O N H V O N N W E B M A P T I L E R
Q W U K D I W F Z G W I N A P D J S B Y U L X K O
S E P W X Y J U Y T K K U F G M T E T X X A B Q T
Z I U U Q B N A C O R O T A R E L E C E D W Q N A
X D D I U W H J M T D N B R R O T A E R C Z L O L
C R J F E L F N D M S H E B R D T T K C I J E H U
O I H F H R L P R E P P A W S E T A N I D R O O C
F D E L S B E R E M P N P F N M H H M Q D E A F L
H D X X A E T V L I D H F L T W W C Q A A W S V A
O U D L M I R J L N P P J W D W H H T T G R Z W C
M O E K P T E C A O O I E E D C M S K A Z L E V A
R M C F L U R M C B S F Q G L M O I T C M N A I E
I E O R E X E D N I T S I L C K Y S J X Z F K M R
N L D I R S F P O F P P I B M C N H E V E W N P A
W I E X E J F Z I X W A V D E R Q V X H M Y Y Q K
P A R T C O U N T E R M I N A T O R U D E T B Q X
X R T M O N B E C E B I J S M L Z Q F E U Y H D L
V E P Q R W V U N T T R Y E I U T J I Q U E P T R
R U B X D Z Z I U F K D W Z C E M F I U C G B M N
N P F M E R O X F A X X F B J Y F G K H F J B T Q
Q S I L R J T Y E K H L P L N Z G T A N V P S U N
Q R Z V G M C C M N N K B K A Y S V U R O L S Q W
M D G K O R I J F K F S J D F S Y D D I F H H F P
D W L G M D T D J Z C Y Y R D T X W L I C Y M K G