



FME Desktop[®]

Esri ArcGIS v10.1
Pathway Training

FME 2013-SP3 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.

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, BC
Canada
V3W1J8

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 mentioned herein may be trademarks or registered trademarks of their respective holders and should be noted as such.

Documentation Information

Document Name: FME Desktop Esri Pathway Training Manual
FME Version: FME 2013-SP3 (Build 13528) 32-bit
Operating System: Windows 7 SP-1, 64-bit
ArcGIS Version: Esri ArcGIS 10.1 (Build 3035)
Updated: August 2013

Introduction.....	5
Esri Pathway.....	5
FME Version.....	5
ArcGIS Version.....	5
Sample Data.....	5
Geodatabase Readers and Writers.....	6
Formats.....	6
Versions and Compatibility.....	6
Feature Datasets.....	10
How Does FME Handle Feature Datasets?.....	10
Handling Annotation.....	11
What is Annotation?.....	11
Annotation Schema.....	11
Reading Annotation.....	12
Writing Annotation.....	12
Multi-Part Annotation.....	12
Feature Linked Annotation.....	22
Text Size vs. Font Size.....	22
Domains.....	28
What is a Domain?.....	28
Domain Reading.....	28
Domain Writing Scenarios.....	28
Limitations.....	31
Subtypes.....	34
What is a Subtype?.....	34
Subtype Writing Scenarios.....	34
Limitations.....	35
Geometric Networks.....	39
Writing to Geometric Networks.....	39
Reading from Geometric Networks.....	39
Performance.....	39
Relationship Classes.....	40
Reading Relationship Classes.....	41
Writing Relationship Classes.....	42
ArcGIS Attachments.....	50
Attachments.....	50
Attachments and FME.....	50
Database Transformers.....	56
Why use a Transformer?.....	56
FME Database Transformers.....	56
Metadata.....	63
What is Metadata?.....	63
Reading Geodatabase Metadata.....	63
Writing Geodatabase Metadata.....	64
Updating Geodatabase Metadata.....	64
Integrating FME and ArcGIS.....	69
Spatial ETL Tools.....	69
Session Review.....	74
What You Should Have Learned from this Session.....	74
Appendix A – Useful debugging resources.....	75
Appendix B – FME, ArcGIS, and Python.....	75
Appendix C – Performance Considerations.....	75
Appendix D – Connecting to an Enterprise Geodatabase.....	76

Introduction



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

Esri Pathway

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

It contains advanced content and assumes that the user is familiar with all of the concepts and practices covered by the FME Database Pathway Tutorial, and the FME Desktop Basic Training Course.

The course looks at the interaction of FME with Esri formats and ArcGIS components. It uses ArcMap and examines the differences between FME and the Data Interoperability Extension.

FME Version

This training material is designed specifically for use with FME2013-SP3. You may not have some of the functionality described if you use an older version of FME.

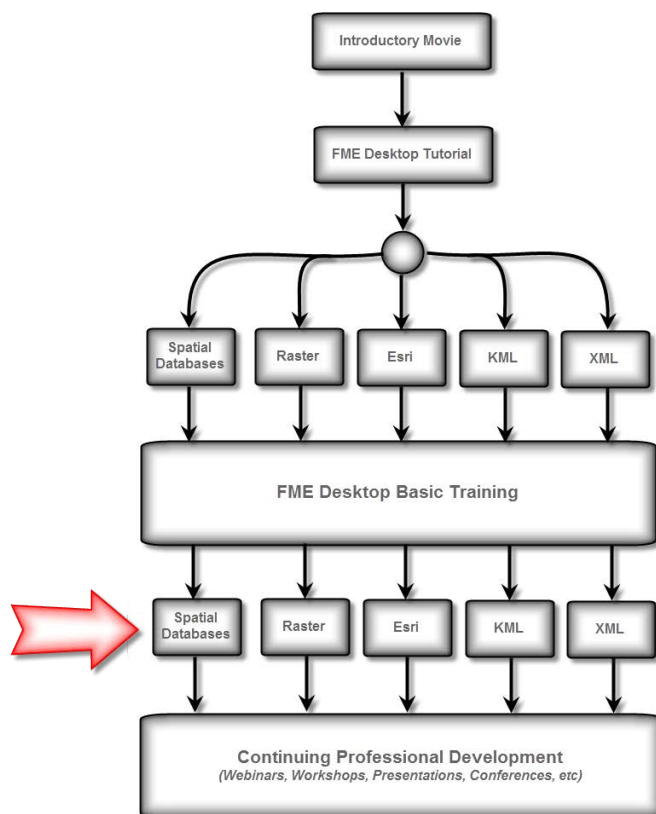
ArcGIS Version

This training material was designed around ArcGIS v10.1, using File Geodatabase.

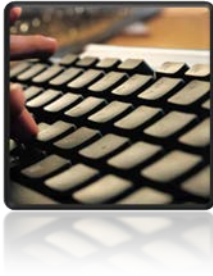
Sample Data

The sample data required to carry out the examples and exercises in this document can be obtained from:

www.safe.com/fmetadata



Geodatabase Readers and Writers



A review of Geodatabase Readers, and Writers

Formats

FME contains a number of formats for reading and writing Esri formats. In this course we'll concentrate on the ones for the various types of vector-based Geodatabase.

FME Format Name	FME Short Name
Esri Geodatabase (Personal Geodatabase)	GEODATABASE_MDB
Esri Geodatabase (File Geodatabase ArcObjects)	GEODATABASE_FILE
Esri Geodatabase (File Geodatabase API)	FILEGDB
Esri Geodatabase (XML Workspace Document)	GEODATABASE_XML
Esri Geodatabase (ArcSDE Geodatabase)	GEODATABASE_SDE

Each reader and writer has different capabilities depending on the technology used. For example, the File Geodatabase API has the ability to create indexes on attributes, whereas the ArcObjects reader/writer for File Geodatabase does not. However, both have the ability to create spatial indexes.

For more information see: <http://fme.ly/1967>

Versions and Compatibility

In general, FME will support an ArcGIS version for as long as Esri considers it still active. Of course, as new ArcGIS versions are released, FME functionality is updated to support it.

The current version compatibility is this:

ArcGIS Version	FME Version
ArcGIS 10.2	FME 2013-SP3 or higher
ArcGIS 10.1	FME 2012-SP3 or higher
ArcGIS 10.0	FME 2010-SP2 or higher
ArcGIS 9.3	FME 2009 or higher
ArcGIS 9.2 (now retired)	FME 2008 or higher

For more information see: <http://fme.ly/1957>

For info on connecting to Enterprise Geodatabase (SDE) see the Appendix at the end of this document.



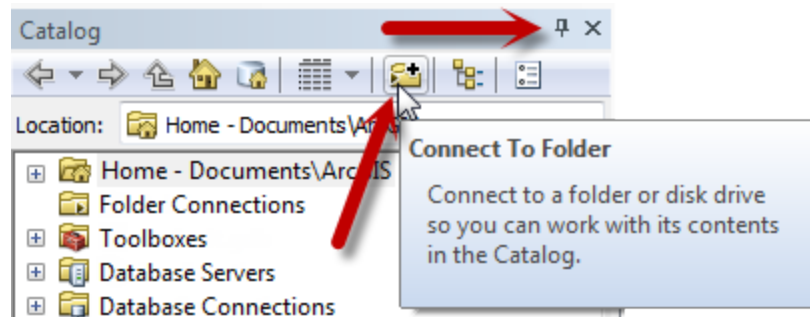
Example 1: Creating a Geodatabase with a Template	
Scenario	FME user; City of Interopolis, Planning Department
Data	Various
Overall Goal	Create a Geodatabase and populate it
Demonstrates	Geodatabase formats. Using ArcGIS XML Workspace Documents
Starting Workspace	C:\FMEData\Resources\Esri\LoadCityData2013.fmw
Finished Workspace	N/A

To get started, this is a basic example of creating a Geodatabase using an ArcGIS XML Workspace Document. These are created by exporting from an existing Geodatabase.

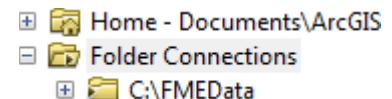
1) Start ArcMap

Start ArcMap. Open a catalog window (be sure to pin it down so it doesn't auto-hide).

Select the Connect to Folder option:



Select C:\FMEData as the folder to connect to.
This enables you to browse to the course data at any time.



Browse to C:\FMEData\Resources\Esri

Double-click on the entry CityDataSchema10.1.xml and, when it opens, notice that it contains a workspace definition (with subtypes, domains, etc.) that we can use as a template for a new Geodatabase.

```
<WorkspaceDefinition xsi:type="esri:WorkspaceDefinition">
  <WorkspaceType>esriLocalDatabaseWorkspace</WorkspaceType>
  <Version/>
  - <Domains xsi:type="esri:ArrayOfDomain">
    - <Domain xsi:type="esri:CodedValueDomain">
      <DomainName>AnnotationStatus</DomainName>
```




2) Start FME Workbench

Start Workbench. Open the workspace: C:\FMEData\Resources\Esr\LoadCityData2013.fmw




We will use this workspace to create a Geodatabase defined by the ArcGIS XML Workspace document, and then write some data to it.

Notice a few of the key points of this workspace:

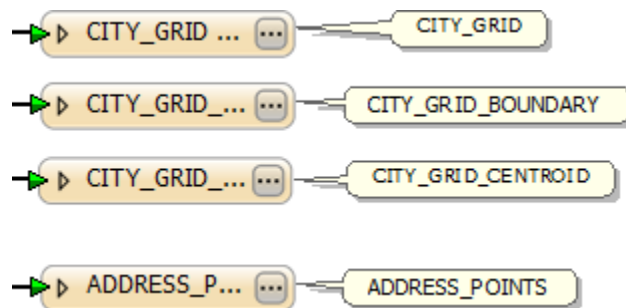
A Writer parameter to identify where the output is being written, and another to ensure any existing data is overwritten by this process:

-  Destination Esri File Geodatabase: C:\FMEData\Resources\Esr\CityData.gdb
-  Overwrite Existing Geodatabase: Yes
-  Transaction Type: Edit Session

A Writer parameter that sets the template (Workspace document) to use in creating the Geodatabase:

-  Transaction Type: Edit Session
-  Template File: C:\FMEData\Resources\Esr\CityDataSchema10.1.XML
-  Simplify Geometry: No

Four Writer Feature Types whose names match ones in the Geodatabase workspace document:



Three Feature Types that have a Geodatabase Feature Dataset set:

Configuration Keyword	DEFAULTS	▼
Feature Dataset	CityGrid	▼
Grid 1		▼

Note that although only four feature classes are defined, more feature classes than that will be created. The four feature classes are just ones that we want to add data to as they are created.

3) Run Workspace

Run the translation. It would be better to close ArcMap at this point to ensure nothing you are writing to is locked by ArcGIS.

The translation should succeed in about 20 seconds or so:

```

=====
|                                     Features Written Summary                                     |
=====
| ADDRESS_POINTS                                                                12292 |
| CITY_GRID                                                                    30   |
| CITY_GRID_BOUNDARY                                                            67   |
| CITY_GRID_CENTROID                                                            30   |
=====
| Total Features Written                                                         12419 |
=====
| Translation was SUCCESSFUL with 0 warning(s) (12419 feature(s) output) |
| FME Session Duration: 19.9 seconds. (CPU: 12.6s user, 5.1s system) |

```



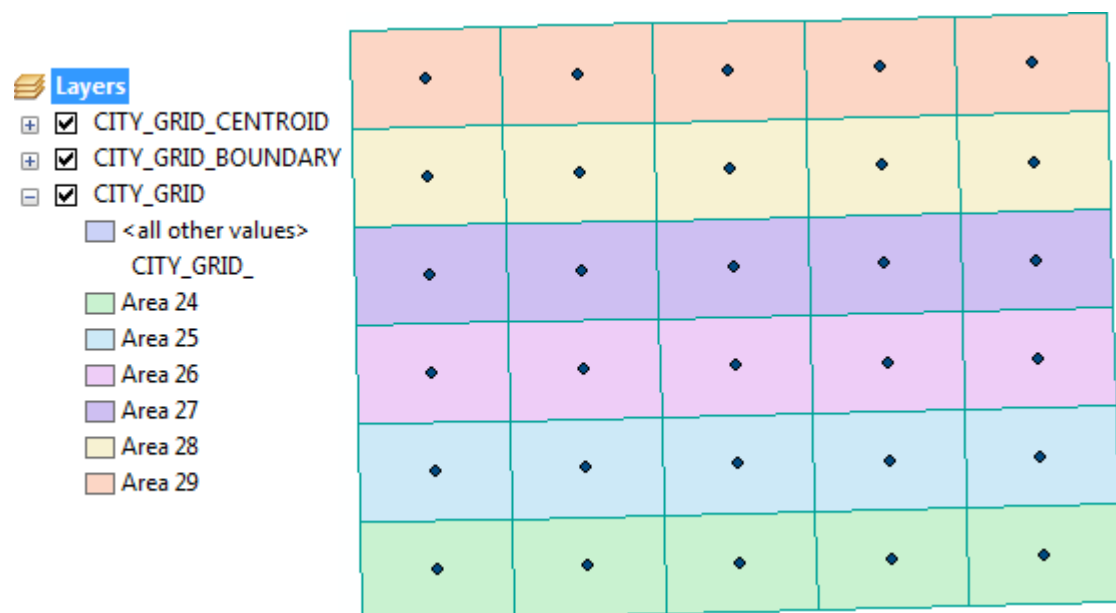
If, during this course, you ever need to revert to the original status of the Geodatabase, you can do so simply by opening and running this translation.

4) Switch to ArcMap

Switch back to (or restart) ArcMap. Browse to C:\FMEData\Resources\Esr\CityData.gdb

Add data from the CityGrid feature dataset to the ArcMap display.

Notice that all the features are colored differently; this is because subtypes are set for the data in that table (from the template) and ArcCatalog automatically colors each subtype differently.



Feature Datasets



Feature Datasets are Geodatabase objects that allow you to group together related tables (feature classes)

Feature Datasets are a way through which different tables (feature classes) can be grouped together. For example, all tables stored with the same coordinate system could be joined together as a Feature Dataset.

How Does FME Handle Feature Datasets?

The Feature Dataset for a particular table can be set in the feature type properties dialog, under the Format Parameters tab:

Shape Alias	<input type="text" value="SHAPE"/>
Configuration Keyword	<input type="text" value="DEFAULTS"/>
Feature Dataset	<input type="text" value="MyFeatureDataset"/> ←
Grid 1	<input type="text"/>
Avg Num of Points	<input type="text"/>



Feature Datasets can also be handled using an FME format attribute. Support for Feature Datasets using a Format Attribute is new in FME2013.

When reading data FME will create a format attribute called `geodb_feature_dataset` to identify the source Feature Dataset for each feature.

When writing data, FME will apply the value of this format attribute in order to write data to a specific Feature Dataset. Note that this attribute takes precedence over the equivalent parameter.

If the Feature Dataset does not already exist, and the data is being written to a new Feature Class, then a Feature Dataset is created using the first feature's spatial reference information.

The first feature to enter the Writer determines the Feature Dataset. It's not possible for two features in the same class to be written to two Feature Datasets, as this would require two Feature Classes, and a named class can only exist once in a Geodatabase.



It's important to remove the value for `geodb_feature_dataset`, when you are reading from one Geodatabase and writing to another, but don't want to write to a Feature Dataset. Otherwise the data will get written to that Feature Dataset regardless.

Handling Annotation



Annotation features are one of the most complex types of geometry to transfer into or out of a Geodatabase

Most users of FME who work with a Geodatabase will need to be able to handle annotation features. However, annotation is not the easiest type of data to work with, particularly when more than one format of spatial data is involved.

What is Annotation?

For our purposes, annotation is any sort of text entity, including multi-part text, leader lines, and feature-linked annotation.

Annotation Schema

An annotation feature class has a considerable number of fields that contain information about annotation fields. These fields include font, size, angle, offset, and alignment.

However these fields should never be used directly when using FME to read or write annotation. Instead, parameters and Format Attributes should be used to control FME, and FME left to fill in the annotation fields as part of the reading/writing process.

The full list of annotation Format Attributes is documented in the FME Readers and Writers Reference manual, but two key attributes are:

- `geodb_text_string`

This format attribute contains the content of the annotation. An FME Geodatabase Reader will return this as a UTF-16 encoded string. A Geodatabase Writer will convert any supplied values into UTF-16 before writing.
- `geodb_text_size`

This format attribute contains the size of the annotation. Although such information is stored in a Geodatabase in 'points', this attribute is in user units; i.e. a Reader will convert from points to user units, and a Writer will convert from user units to points.



Funkmeister F.M.E. says...

"You can see why it's important to let FME handle the Geodatabase attribute fields. Writing directly to a field could result in the wrong encoding or wrong units being used! ArcGIS won't let you edit them directly, so you shouldn't try here."

Reading Annotation

Annotation features viewed in ArcGIS may display different values to that held in the underlying database tables. For example, a "HorizontalAlignment" field that displays as "Left" in ArcCatalog actually has a value of "0" in the database.

FME helpfully retrieves both values: the attribute "HorizontalAlignment" will have a value of "0" and the format attribute "geodb_h_align" will show a value of "Left". Of course, it helps to understand the difference between the two – especially when writing the data to a different format – and the FME Data Inspector can be used to clarify any differences.



Inspecting features in the FME Data Inspector is useful because both Esri Geodatabase attributes and FME Format attributes will be displayed.

Writing Annotation

As noted already, format attributes are used to update fields in the annotation feature class when writing annotation. It is not necessary – and in fact is harmful to the process – to attempt to fill in these fields directly from a workspace.



When importing an annotation feature class into FME, you may find that some of the attributes from the table are included. These are for reference only – you shouldn't write data directly to them! In fact, best practice is to remove these altogether. The only attributes that should be defined on an annotation feature class are user attributes.

Multi-Part Annotation

The Geodatabase Reader has a parameter that specifies whether or not to split multi-part annotations into separate features for each 'element' when reading. It is of particular use when reading annotation that has a curved layout.

If set to yes, a single feature for each element (usually a word) in a multi-part annotation will be produced on reading, resulting in feature-specific attributes such as angle and text position being stored according to the location of each element.

The "Split Complex Annotation" parameter will break up curved annotation even more into individual letters whenever rotation or font changes. This can result in many additional features (so please use with care) but allows for very accurate annotation placement. It is sometimes possible to aggregate them back into a single text string using a series of transformers.



Example 2: Writing Annotation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Water Distribution
Overall Goal	Amend workspace to write annotation
Demonstrates	Basic annotation writing
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri2-Complete.fmw

This is a basic example of writing annotation to a Geodatabase.

This example shows some of the capabilities of FME as it extends ArcGIS functionality, with or without the ArcGIS Data Interoperability Extension, and then uses FME Desktop to expand upon this capability.

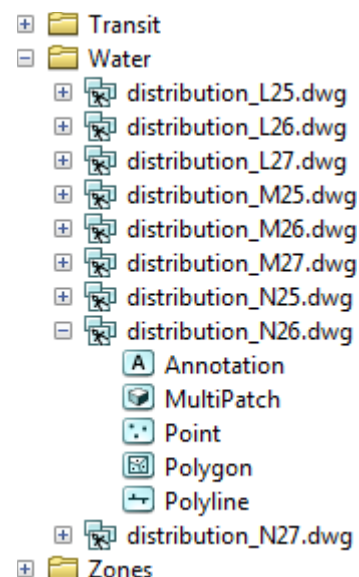
The scenario is creating an Annotation feature class for water_meters, which aren't yet in the workspace. Because these features are logically related to the rest of the Water Distribution data, we will add these to the Feature Dataset also.

1) Start ArcMap

A key part of any data translation is to inspect the data beforehand. In an ArcMap catalog window browse to this example's source dataset to review it.

Reader Dataset C:\FMEData\Data\Water\distribution_N26.dwg

A class/object can be displayed by dragging it from the catalog window into the main map display window.



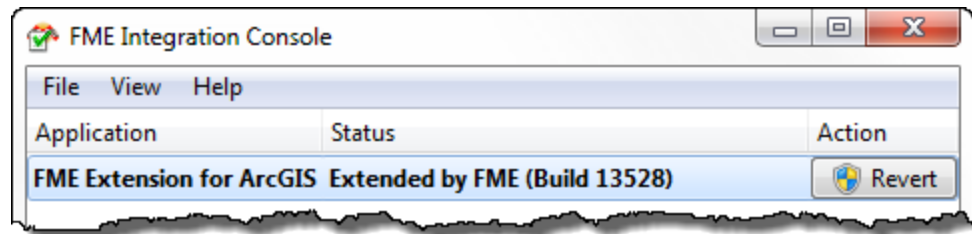
NB: Support for many non-Esri formats is included by simply installing FME. In this case, ArcGIS has its own built-in support for DWG, and so FME is not being used.

2) Activate FME Extension

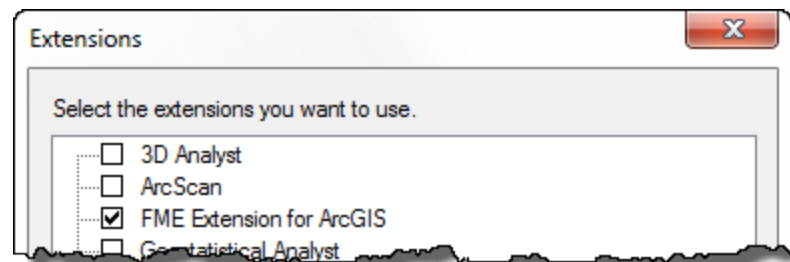
The problem with plain browsing (for DWG) is that ArcGIS separates the data by geometry, and not layer. To solve that, use the FME extension for ArcGIS (or Data Interoperability Extension).

To activate the FME extension for ArcGIS:

- Select Start Menu > FME Desktop > Utilities > FME Integration Console
- Ensure that the FME Extension for ArcGIS is set to Extended



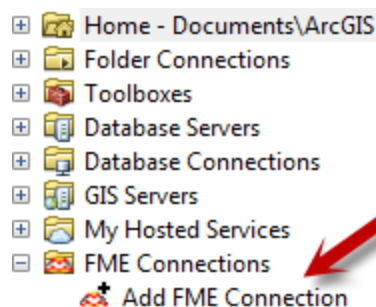
- In ArcMap (you may need to restart it), visit Customize > Extensions
- Ensure the FME Extension (or Data Interoperability Extension) is activated here too



3) Add FME/Interoperability Connection

Once activated, at the foot of the browse tree will be either the option "FME Connections" or "Interoperability Connections".

Expand that entry and double-click the option to "Add [...] Connection"



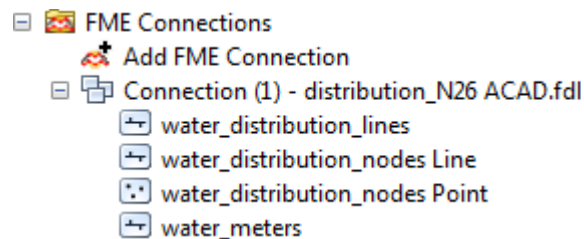
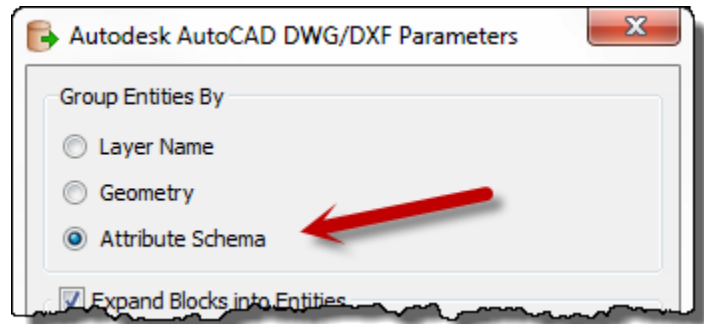
4) Create Connection

When prompted set the Reader information as follows:

Reader Format AutoCAD DWG/DXF
Reader Dataset C:\FMEData\Data\Water\distribution_N26.dwg

Before clicking OK, click the Parameters button, and check that the “Group Entities By” parameter is set to “Attribute Schema”.

Click **OK**, and then **OK** again to add the connection.



Now you can browse that connection and have the data arranged by layer/schema, rather than geometry.

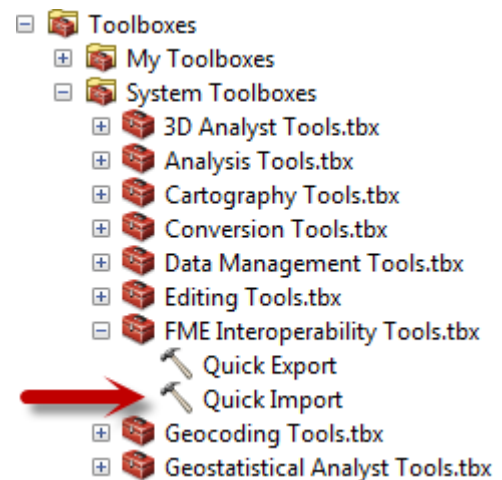
5) Import Data

Another capability FME adds is the import of supported formats directly into a Geodatabase.

In ArcMap, either open an ArcToolbox window or browse to Toolboxes > System Toolboxes in the Catalog window.

In the ArcToolbox tree, expand the FME/Data Interoperability Tools section, and double-click Quick Import to start that tool.

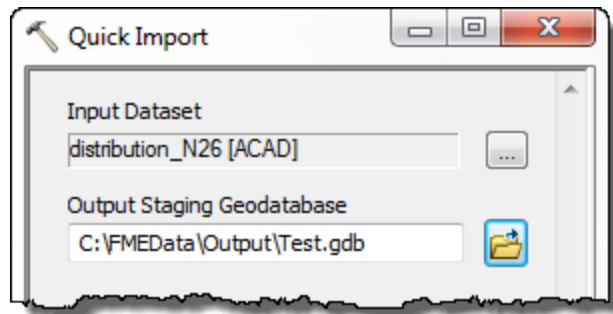
In the Quick Import dialog, click the Input Dataset browse button and set the Input Dataset as follows:




Reader Format AutoCAD DWG/DXF
Reader Dataset C:\FMEData\Data\Water\distribution_N26.dwg

Be sure to again click the Parameters button, and check that the “Group Entities By” parameter is set to “Attribute Schema”.

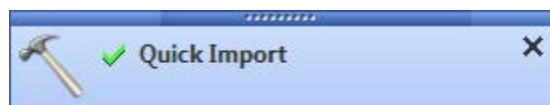
Set an output Staging Geodatabase, for example Test.gdb (or use the default value):



Click OK and the translation will run, as shown by the status bar “throbber”:

.Quick Import...Quic  1449.772 522.831 Unknown Units

Once complete a pop-up dialog will appear, like so:



Click on this and an FME log file will appear in a Results window:

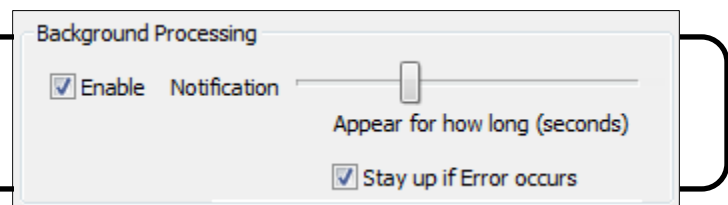
```

i =====
i water_distribution_lines                408
i water_distribution_nodes_line          772
i water_distribution_nodes_point         386
i water_meters                          458
i =====
i Total Features Written                  2024
i =====
i Translation successfully completed
i FME Session Duration: 10.5 seconds. (CPU: 5.1s user, 3.3s system)
i END - ProcessID: 8928, peak process memory usage: 210324 kB
i Translation was SUCCESSFUL
i Done Import. Read 2024 features.
i Succeeded at Thu Aug 01 15:40:16 2013 (Elapsed Time: 12.00 seconds)
QuickImport [153549_08012013]

```

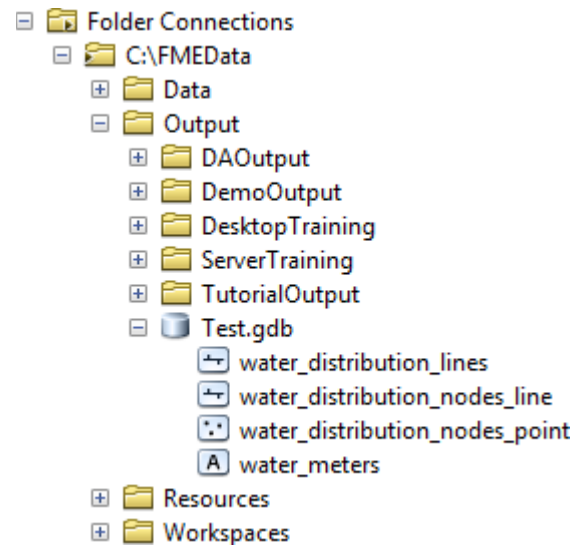


In ArcGIS you can turn off background processing using Geoprocessing > Geoprocessing Options on the menubar:



This was just a *Quick Import* using FME/Data Interoperability functionality; i.e. in the same way as an FME Quick Translation, there is no transformational stage.

Check the output to see what it looks like. Next we'll use an FME workspace to show what differences can be made using data transformation.



6) Start Data Inspector

As part of your translation routine, start the FME Data Inspector and inspect the source data:

Reader Format	AutoCAD DWG/DXF
Reader Dataset	<u>C:\FMEData\Data\Water\distribution_N26.dwg</u>

7) Start Workbench

Start FME Workbench. Generate a workspace to translate the DWG data as follows:

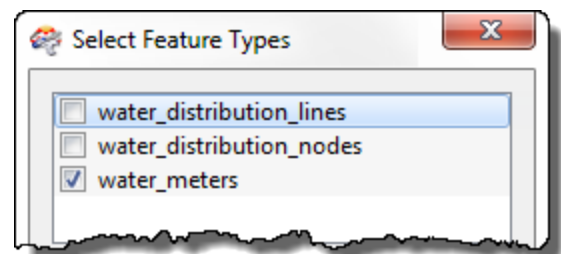
Reader Format	AutoCAD DWG/DXF
Reader Dataset	<u>C:\FMEData\Data\Water\distribution_N26.dwg</u>

Writer Format	Esri Geodatabase (File Geodatabase ArcObjects)
Writer Dataset	<u>C:\FMEData\Resources\Esri\CityData.gdb</u>

Use the following Reader parameters:

Group By:	Attribute Schema
Expand Blocks into Entities:	Yes
Use Block Header Layer:	Yes
Resolve Entity Color:	Yes
Read Visible Attributes as Text:	Yes
Explode MText Entities:	Yes

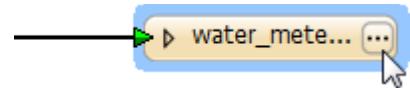
Click **OK** to accept this dialog. When prompted select **water_meters** (only) as the Feature Type to be processed.



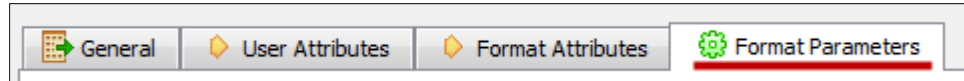
8) Update Writer Feature Type

Now a workspace has been created. The first thing we can do with this translation is to ensure the data is written to the correct Feature Dataset.

Open the properties dialog for the water_meters Writer Feature Type.



Click on the Format Parameters tab:



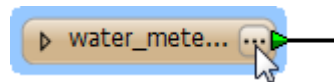
In the Feature Dataset parameter, enter WaterDistribution. Click **OK** to save the Properties.

Configuration Keyword	DEFAULTS
Feature Dataset	WaterDistribution
Grid 1	

9) Expose Reader Format Attributes

Various text format attributes must now be exposed on the Reader Feature Type.

This will provide information about the source (AutoCAD) annotation features that can be used to recreate them in ArcGIS.



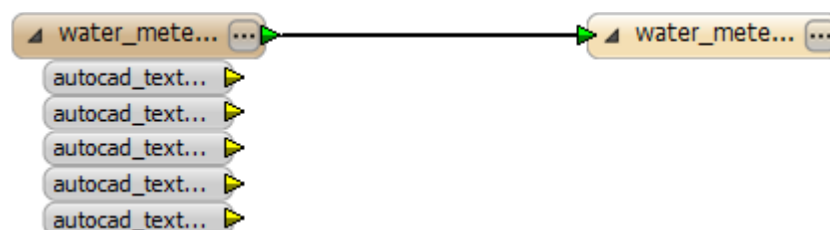
Open the feature type properties dialog and click the Format Attributes tab.

Put a checkmark against these Format Attributes to expose them:

- autocad_text_rotation
- autocad_text_size
- autocad_text_string
- autocad_text_x_pos
- autocad_text_y_pos

<input checked="" type="checkbox"/>	▶ autocad_text_rotation
<input checked="" type="checkbox"/>	▶ autocad_text_size
<input checked="" type="checkbox"/>	▶ autocad_text_string
<input checked="" type="checkbox"/>	▶ autocad_text_x_pos
<input checked="" type="checkbox"/>	▶ autocad_text_y_pos
<input type="checkbox"/>	▶ autocad_text_z_pos

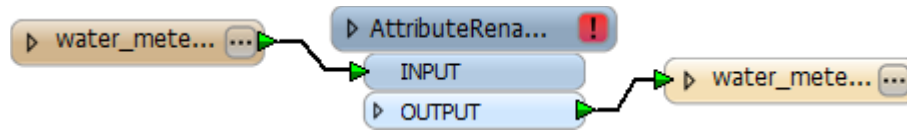
The result looks like this:



10) Map Format Attributes

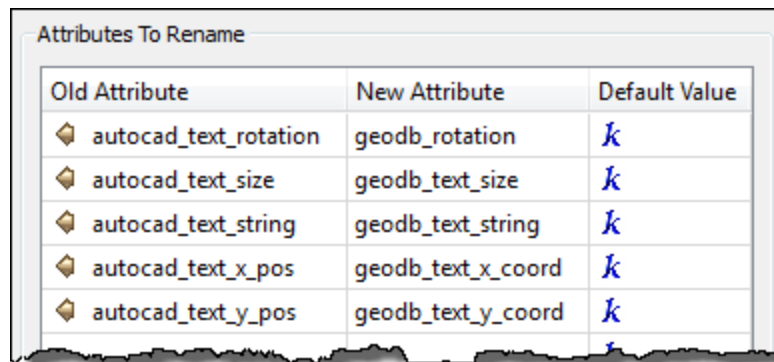
Some of the AutoCAD format attributes are a direct match for their Geodatabase equivalent. In that case they can be mapped directly from one to another.

Place an AttributeRenamer transformer between the Reader and Writer Feature Types.



Open its properties dialog and set the parameters to map the Reader and Writer schemas as follows:

autocad_text_rotation	►	geodb_rotation
autocad_text_size	►	geodb_text_size
autocad_text_string	►	geodb_text_string
autocad_text_x_pos	►	geodb_text_x_coord
autocad_text_y_pos	►	geodb_text_y_coord

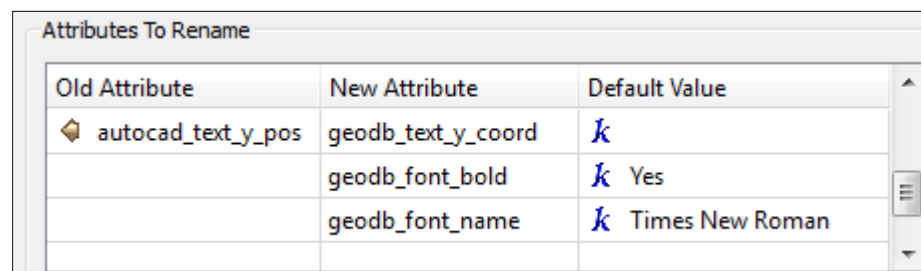


11) Add Format Attributes

Two more Geodatabase format attributes need to be set, but don't have AutoCAD equivalents. These can be created using the same AttributeRenamer transformer.

Open its properties dialog again. This time use it to create two new attributes by using the columns "New Attribute" and "Default Value".

geodb_font_bold	Yes
geodb_font_name	Times New Roman

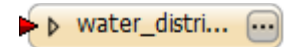




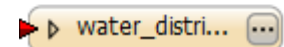
The AttributeRenamer is a jack-of-all-trades transformer. It can be used to rename attributes, copy them, delete them, or - like here - create them. In this example you'll need to ensure it is placed in position, between the two Feature Types, before you start to use it; otherwise it won't have access to the list of Writer Format Attributes.

12) Final Cleanup

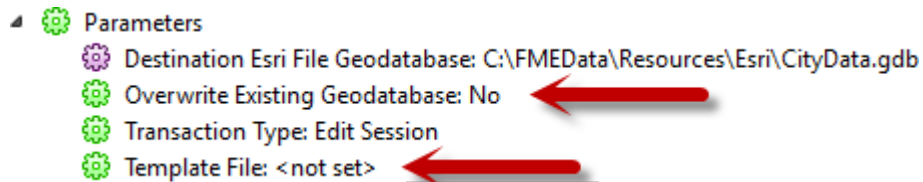
As a final step, we'll clean up some outstanding items.



If there are any unconnected Writer Feature Types (for example water_distribution_nodes) then they can be deleted. This will ensure we don't create any tables that are not required.



Ensure the Writer parameter "Overwrite Existing Geodatabase" is set to No; else the entire Geodatabase will get overwritten. Ensure also that the Template File parameter is now empty – otherwise FME will keep trying to use it to recreate the Geodatabase.



13) Save and Run the workspace

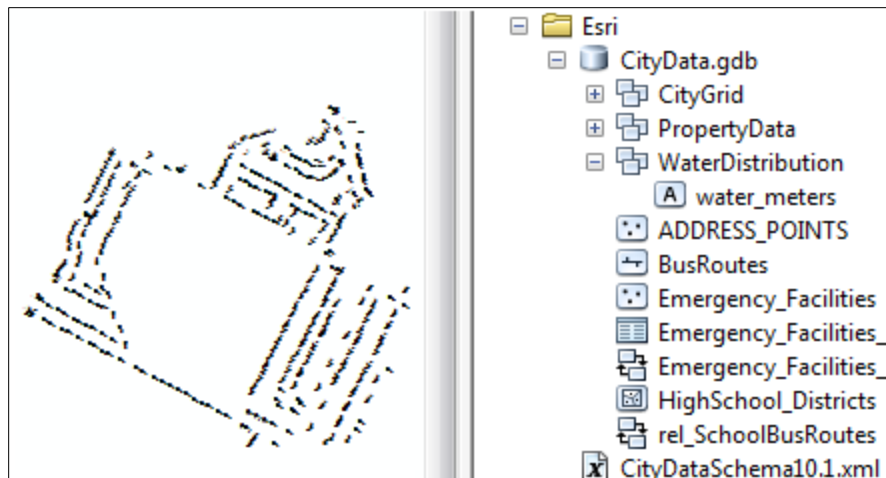
Save the workspace and then run it. Again you may wish to close ArcGIS first.

The Unexpected Input Remover might issue a warning, because some of the features in the Reader dataset are not represented by Feature Types in the workspace. This warning can be ignored, because these AutoCAD layers were deliberately left out.

14) Inspect Output

Again, inspect the output in ArcMap.

Check the values of the annotation attributes and how it displays in the map window.



Advanced Task

Check the values of the Writer Feature Type parameters Drop Table First and Truncate Table.

Try re-running the workspace using different combinations to see what happens (just DO NOT set "Overwrite Geodatabase" to Yes!)

Notice that if you write data to an annotation feature class that already exists and has some Annotation Classes defined then it is possible to expose the `geodb_anno_class_id` attribute and specify the class to be used.

Some of the `geodb_` format attributes will override the specifications in the class – see the FME Readers and Writers manual for more information on this.

Feature Linked Annotation

Although annotations are stored in a separate feature layer in a Geodatabase, they can be linked to other features through feature-linked annotations. Feature-linking occurs when there is a relationship between an annotation feature class and some other feature class.

Linking is carried out by defining a relationship through a common attribute. The relationship must already be defined in the Geodatabase before writing the data.

If the feature to be linked to by the annotation has not yet been written, then it is possible for the Geodatabase writer to write the feature, retrieve the object ID of the new feature and then write the annotation feature linking to it. The result is that the one FME feature contains enough information to write two features: one annotation feature and one non-annotation feature.

Text Size vs. Font Size

Some confusion exists over the difference between text size and font size for Geodatabase annotation.

There are two format attributes: `geodb_text_size` and `geodb_font_size`

`geodb_text_size` is the size of the text in user (ground) units. This value is converted to “points” when written, and text displayed at this size in ArcGIS. If no value is supplied then the default text size is the equivalent to 10.0 points.

`geodb_font_size` is the size of the font used to display the text string. Font size can only be set when the text size attribute is not present (i.e. `geodb_text_size` has priority).

See the FME Readers and Writers Manual for a complete list of annotation format attributes:

You are here: [FME Readers and Writers \(formats supported by FME 2013\)](#) > [Esri Geodatabase Reader/Writer](#) > [Feature Representation](#) > Annotation Attributes

Annotation Attributes

The following attributes are used to store the annotation information within an FME annotation feature.

In ArcGIS 9.1, the schema of annotation feature classes changed and a considerable number of additional fields were added. These new fields contain information about the annotation such as its font, size, angle, offset, leading, etc. When writing, these fields should **never** be set directly; instead the attributes in the table below must be used to control the properties of the annotation. After the translation,

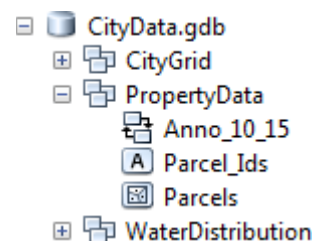


Example 3: Writing Feature-Linked Annotation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property Outlines
Overall Goal	Create workspace to write feature-linked annotation
Demonstrates	Writing feature-linked annotation
Starting Workspace	None
Finished Workspaces	C:\FMEData\Workspaces\PathwayManuals\Esri3-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Esri3-Complete-Advanced.fmw

The task here is to load property data into a feature dataset. The output should be one feature for every property parcel, plus a feature linked annotation. Linking will be done via parcel ID.

The source data is the property linework along with a centroid which holds the parcel ID. The complicating factor is getting the parcel ID from the centroid onto the related parcel features.

To write Feature Linked Annotation the feature class PropertyData has to exist and the annotation linkage be defined. Check the Geodatabase using ArcMap to see that this is so.








1) Start ArcMap

Let's start by looking at the source data we are to use.

Start ArcMap if necessary. In a Catalog window browse to C:\FMEData\Data\Properties\parcel_N26.mif

Notice how you can view this MapInfo data through the FME Extension (Data Interoperability Extension)

Add the data to a map and examine the contents. Notice there are line and point features only.

Name	Type
 parcel_N26 Line	FME Feature Class
 parcel_N26 NoGeometry	FME Feature Class
 parcel_N26 Point	FME Feature Class
 parcel_N26 Polygon	FME Feature Class
 parcel_N26 Text	FME Feature Class

2) Start Workbench

Now let's translate this to Geodatabase and create Feature-Linked annotation.

Start Workbench and begin with an empty canvas.

On the menubar select Source Readers > Add Reader and add a reader to read the following MIF/MID dataset. You may also want to inspect the data first – either in the FME Data Inspector or in ArcGIS – to see what we are dealing with.

Reader Format	MapInfo MIF/MID
Reader Dataset	<u>C:\FMEData\Data\Properties\parcel_N26.mif</u>

3) Add Writer

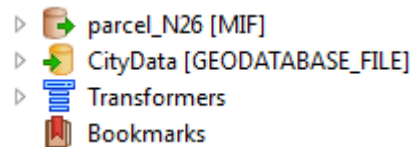
On the menubar select Writers > Add Writer and add the output Geodatabase. This already exists – in other words we are updating/adding to it.

Writer Format
Writer Dataset

Esri Geodatabase (File Geodatabase ArcObjects)
C:\FMEData\Resources\Esri\CityData.gdb

When prompted to add a Feature Type, respond **No**, because the Feature Classes already exist in the Geodatabase; they can be imported more easily than being re-created.

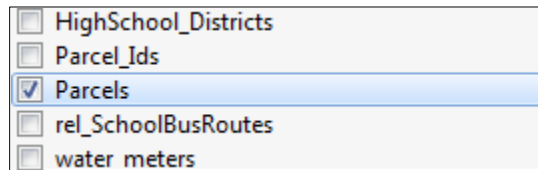
The canvas won't show any changes now, but a new Writer has been added in the Navigator window.



4) Import Feature Type

Now let's add tables to this Writer schema.

On the menubar select Writers > Import Feature Types. When prompted set the format and select the file Geodatabase as in step 3. Click **OK**.



FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *Parcels* only.





Now we have both a Reader and a Writer in the Navigator window and, on the canvas, a Feature Type for each of these.



5) Set Transaction Type

In the Navigator window, set the transaction type to Edit Session. We are dealing with complex features that can only be edited in an Edit Session or Version.

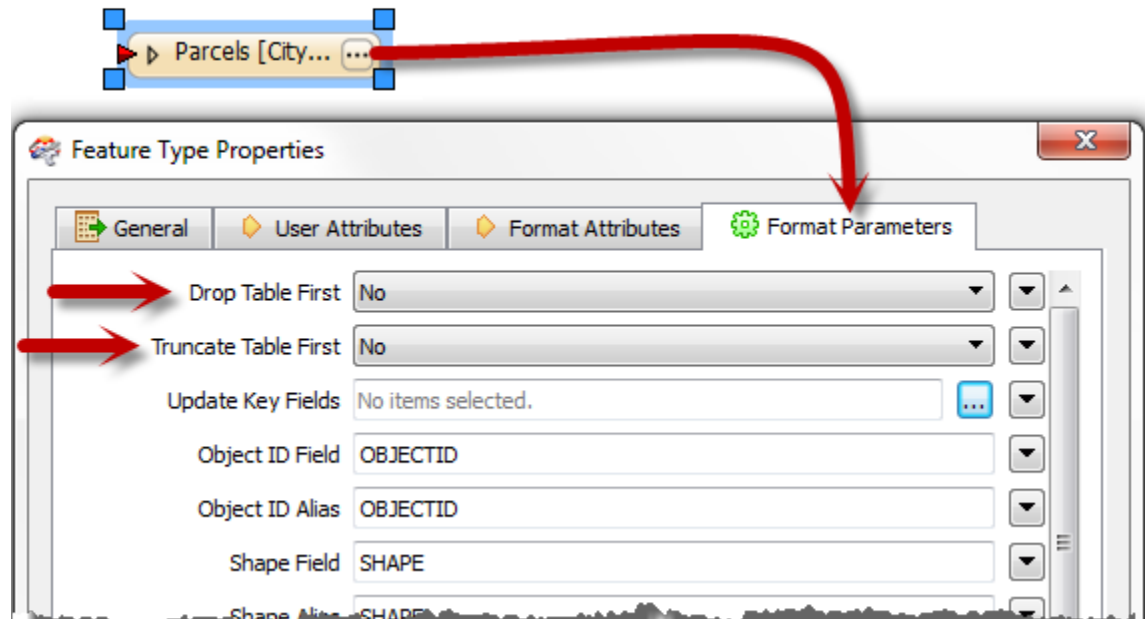
Also ensure that Overwrite Existing Geodatabase is set to No and that the Template File field is not set. We do not wish to overwrite the entire Geodatabase, nor add new tables.

-  Destination Esri File Geodatabase: C:\FMEData\Resources\Esri\CityData.gdb
-  Overwrite Existing Geodatabase: No
-  Transaction Type: Edit Session
-  Template File: <not set>

6) Set Properties

Once Parcels is imported (you don't need to connect it up yet) open the properties dialog and click the Format Parameters tab. Check (and set if required) the following parameters:

Truncate Table First	No
Drop Table First	No



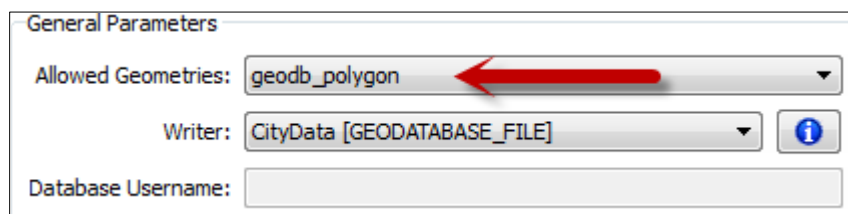
Setting these properties means that we will keep the schema of the existing table, and that we will be adding data – not replacing it – when we run the translation.

Now – in the User Attributes Tab – remove the user attributes:

- geodb_oid
- OBJECTID
- SHAPE_Length
- SHAPE_Area

These attributes are always auto-generated by ArcObjects and if we were to leave them on the schema, they would get renamed to OBJECTID_1, etc.

Finally, under the General tab, ensure that the Allowed Geometries parameter is set to geodb_polygon. If this were set to something else, then all the polygon data we wrote would be rejected by the feature class.



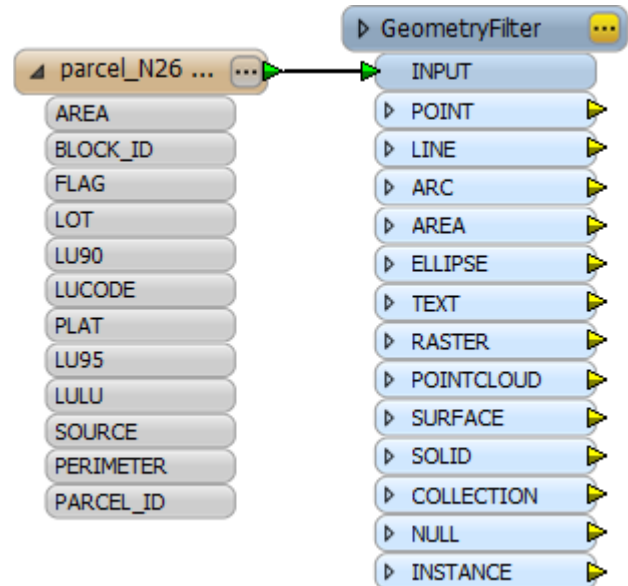
7) Add GeometryFilter

The Reader and Writer part of the translation are now complete. Now it's time to deal with the geometry and the tasks of building polygon features and getting an ID number from the centroid.

Add a *GeometryFilter* transformer. This will help us separate out the line features from the point features.

The point features hold the attributes so will be used to build the annotation.

The line features will be turned into polygons and be written out.



8) Add AreaBuilder

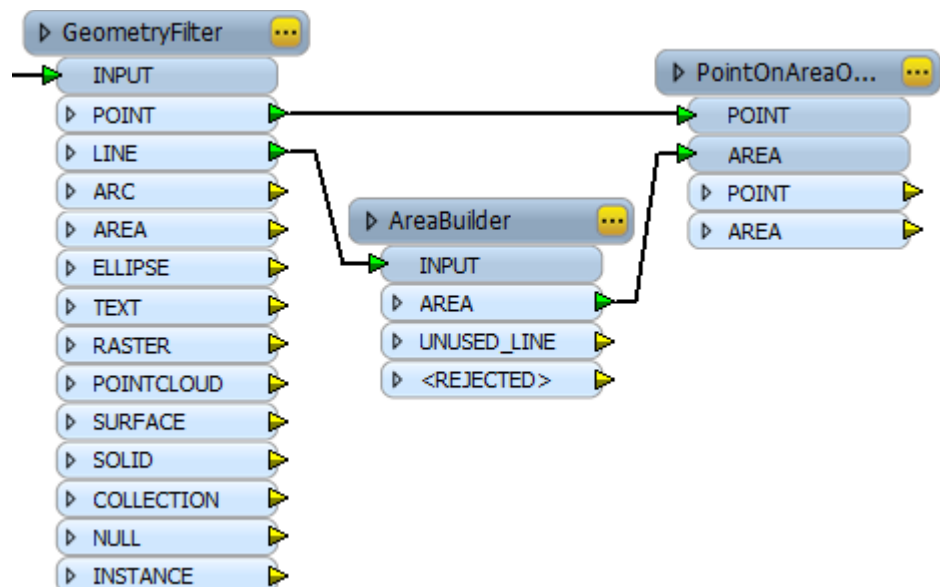
Add an *AreaBuilder* transformer to the output from the LINE port of the *GeometryFilter*. This will turn the line features in the source data into proper polygon features.

9) Add PointOnAreaOverlayer

Now add a *PointOnAreaOverlayer* transformer. This transformer is used to transfer attributes – in this case the parcel ID attribute – from point features onto surrounding polygons.

Connect the output port 'POINT' from the *GeometryFilter* and 'AREA' from the *AreaBuilder* to the 'POINT' and 'AREA' ports on the *PointOnAreaOverlayer* transformer.

No parameters need to be set this time.



10) Save and Run Workspace

Ensure the PointOnAreaOverlayer:AREA port is connected to the Writer feature type. Now save the workspace as Example3.fmw and then run it (close ArcMap first).

Inspect the output using ArcMap.

The `PropertyData` feature class should now contain parcels and feature-linked annotation.



Advanced Task

Attempting to rerun this workspace introduces an interesting problem.

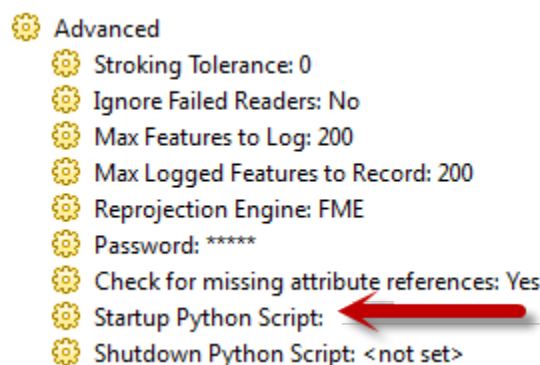
We can set the Truncate Table setting to Yes however this will only truncate the Polygon layer. It does not affect the Annotation layer. So, to truncate features from both layers it becomes necessary to use a Python script and invoke some ArcGIS geoprocessing tools to do the work.

Open the python script defined into a text editor:

Script: *C:\FMEData\Resources\Esrri\Parcels_delete.py*

This is some python code to invoke geoprocessing tools to delete the features from the two layers. Copy and paste the code into the **Startup** Python script editor, found in the Navigator Window:

Comment out or edit either line 12 or 13 (the path to ArcGIS) as necessary.



Mr. B. Lover, the Interopolis Zoo insect and reptile expert says...

"Python handling can be a tricky operation. To avoid a crushing disappointment please read the appendix on 'FME and Python' located at the end of this manual, and make appropriate changes to your environment"

14) Re-Run Workspace

Now the workspace may be re-run without creating duplicate annotation features. Examine the results in ArcMap to confirm the transformation worked.

Domains



A Domain is a data type that defines aspects of a Geodatabase schema related to data integrity.

What is a Domain?

A domain is a set of rules that define permitted values for an attribute. They are used to constrain data values and so ensure data integrity.

There are two types of domain, and both are supported by FME: coded domain, and range domain.




A coded domain is essentially a list of multiple valid values. A range domain is a single permitted range of values, therefore you would use it for numeric rather than character string attributes.

A domain is defined in a Geodatabase as a unique entity; i.e. it is a standalone item that can be applied to any attribute in any feature class within that Geodatabase.

If data in a feature class has been subdivided using a subtype, then different domains can be applied to each subtype.

Domain Reading

When reading a Geodatabase, FME has an option to resolve domains.

-  Spatial Data Only: No
-  Resolve Domains: Yes
-  Resolve Subtypes: Yes

When this is set to Yes, not only is the attribute value read, but <attribute>_resolved – the resolved version of the domain – is created too.

<input checked="" type="checkbox"/>	SymbolID	integer	
<input checked="" type="checkbox"/>	Status	coded_domain	View...
<input checked="" type="checkbox"/>	Status_resolved	char	254
<input checked="" type="checkbox"/>	TextString	char	255

Domain Writing Scenarios

FME has options to write to an existing table or to create a new table, but when a domain is added to the mix there are a number of scenarios:

- Write to an existing table using an existing domain
- Write to a new table using an existing domain
- Write to a new table creating a new domain

These scenarios will be controlled by a series of parameters, namely:

Data Type	coded_domain/range_domain
Validate Features to Write	Yes/No

Writing to an existing table using an existing domain

It's extremely simple to write to an existing table using an existing domain.

Any data written to a domain field is, by default, simply inserted as normal. Because the table already exists its attribute(s) will already be associated with the required domain, and there is no need to set any parameter to define this connection.

However, if you wish to validate incoming data – for instance compare it to a domain definition to ensure it has valid attribute values – then you must set the writer parameter “Validate Features to Write”:

- ⚙️ Advanced
 - ⚙️ Default Z Value: 0
 - ⚙️ Writer Mode: INSERT
 - ⚙️ Transaction Number: 0
 - ⚙️ Features to Write Per Transaction: 1000
 - ⚙️ Ignore Failed Features: No
 - ⚙️ Max number of features to ignore: -1
 - ⚙️ Dump Failed Features to File: No
 - ⚙️ Failed Feature Dump filename: <not set>
 - ⚙️ Annotation Units: unknown_units
 - ⚙️ Compress Database When Done: No
 - ⚙️ Validate Features to Write: Yes
 - ⚙️ Simplify Network Features: No

In this example the parameter Validate Feature to Write is set to Yes.

If, for example, permitted values for a field were S, M, or L, and one feature had the value XL, then the translation would fail with the following error:

Validation failed for a feature being written to the table/feature class
<ClassName>

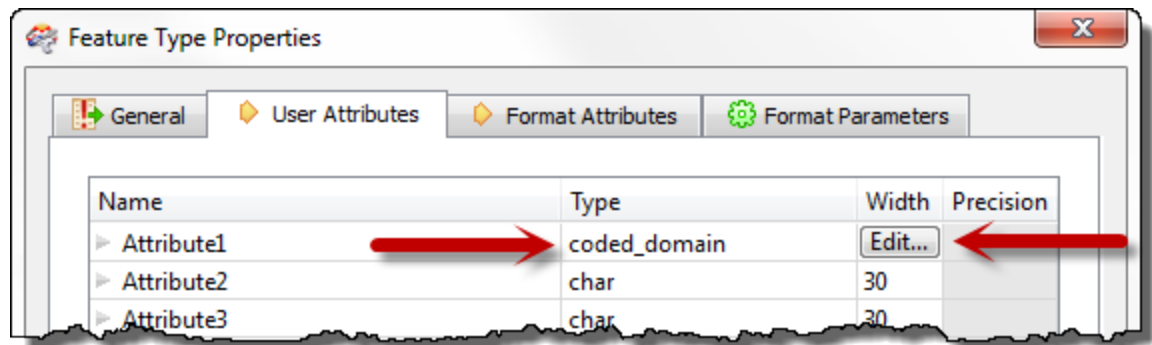
The error message is: Field <FieldName> attribute value XL is not member of
coded value domain <DomainName>.

If the validation parameter was set to No, then the data would pass into the Geodatabase without error, even though it would otherwise fail the domain rules.

Writing to a new table using an existing domain

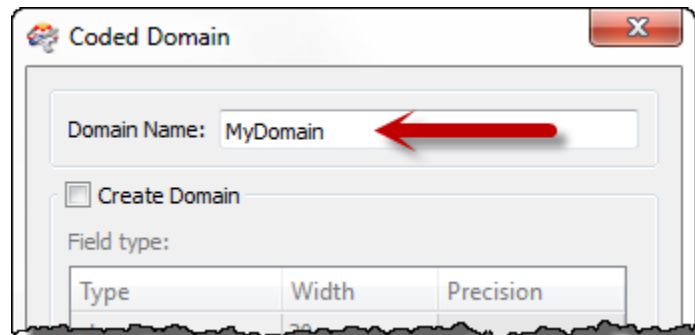
When creating a new table that is to use an existing domain, it's not much more difficult than when the table already exists.

The attribute that needs to be associated with a domain should be given the data type *coded_domain* or *range_domain* (depending on its type) in the schema definition, instead of the usual char, float, integer, etc.:



The next step is to click the Edit button in the attribute width field.

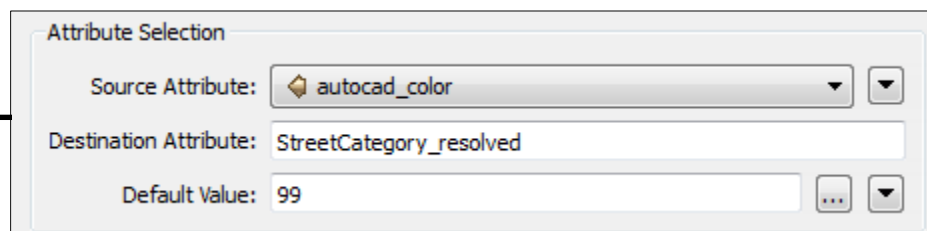
For an existing domain the dialog is very simple; you just fill in the name of the existing domain:



The range domain dialog is slightly different to the one for coded domain, but in this scenario it's still just one parameter (*Domain Name*) that matters.



When writing to a domain, you may understandably forget which attribute value is supposed to be used for which domain value. In that situation you can use the domain value directly, by renaming the attribute to <AttributeName>_resolved, like here where a user can remember the domain value (A Road, B Road, etc.) but not the attribute value that maps to it.

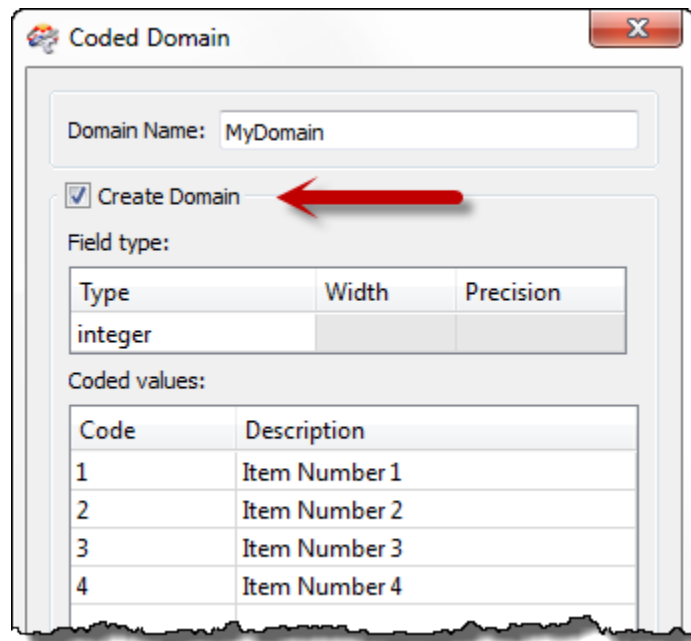


Writing to a new table and creating a new subtype/domain

Generally we recommended that you use ArcGIS to create and define domains, and simply use FME to associate attributes or validate data when inserting new features. That's because FME cannot hope to replicate the full extent of ArcGIS functionality, particularly in the relationship between domains and subtypes.

Having said that, creating Domains is possible with FME.

The process is the same as for using an existing domain, up to opening the edit dialog. At that point you check the "Create Domain" parameter, enter a new domain name, and define the values for that domain.



Here the user has a coded domain for a series of integers.

Note that, provided the "Validate Features to Write" parameter is set, incoming features will be automatically validated against any newly defined domain.

Limitations

There are a couple of limitations to domain writing.

Firstly, it is not possible to write to an existing table and to either create an association with an existing domain or create an entirely new domain. That's because this association is wrapped up in the table definition, and an existing table definition cannot be changed by FME. You would need to drop the existing table and re-create it entirely in order to be able to do this.

For the same reason, creating a domain is a one-off translation. You would set the data type to coded_domain for the initial process, but subsequent loads of the data should be done with the data type changed back the actual type of data (char, integer, etc.)

Finally, it is not possible to create a domain dynamically; i.e. the domain definition cannot be set as part of the workspace process, but must be manually defined prior to execution.

For more information, please see the Esri Geodatabase chapter in the *FME Readers and Writers* manual.



Example 4: Writing a Coded Domain	
Scenario	FME user; City of Interopolis, Planning Department
Data	Water Distribution (AutoCAD DWG)
Overall Goal	Create a workspace to create and write coded domain data
Demonstrates	Handling Domains
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri4-Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri4-Complete.fmw

In this example the requirement is to load some water distribution data into a feature dataset. There is a source drawing file that consists of a number of different layers, one of which has an attribute that is suitable for validation using a Domain table.

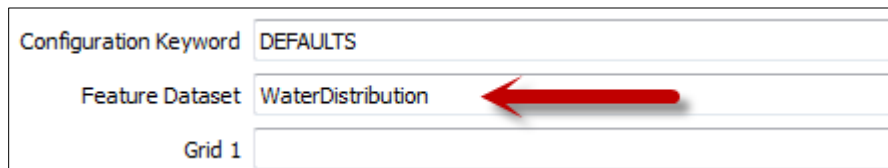
As a secondary task, because the source is a drawing file containing blocks (a form of symbol), we will keep AutoCAD-specific information about these special drawing objects so that the reverse translation can take place and the drawing recreated if necessary.

1) Start Workbench

Start Workbench and open the starting workspace provided.

Notice how this workspace reads two AutoCAD layers and writes out three feature classes.

Also, because a number of features are logically related, a *Feature Dataset* is used to hold them. This is specified by a parameter on the destination feature type settings.

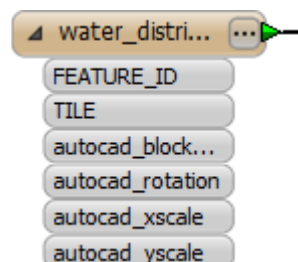


2) Examine Reader/Writer Attributes

On the Reader feature type *water_distribution_nodes* a number of format attributes have been exposed. These are written to the output Geodatabase so that they can be used to recreate the blocks if the data is ever written back to a DWG dataset.

This is a common way of maintaining information about a source format when converting into a Geodatabase feature class:

- autocad_block_name
- autocad_rotation
- autocad_xscale
- autocad_yscale



3) Build Domain Table

The *water_distribution_lines* feature type represents pipelines that have a fixed set of diameter values. A domain table can be applied to the DIAMETER attribute to validate these values.

Open the properties dialog for the writer feature type *water_distribution_lines*. Click on the User Attributes tab and locate the DIAMETER attribute. Click on the dropdown list of data types for that attribute and select "coded_domain".

▶ COMPTYPE	double		
▶ DIAMETER	coded_domain ▼	Edit...	
▶ DRAW	double		
▶ FEATURE_ID	char	25	

Now click the Edit button for the coded domain.

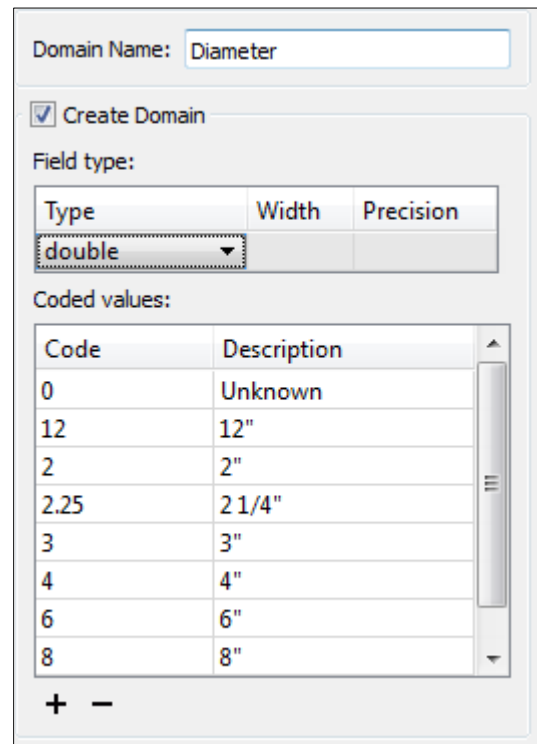
Since the domain does not already exist in the Geodatabase it is necessary to build it here.

Check the parameter "Create Domain", and give the Domain Table a name such as "Diameter".

Set the Field Type of the codes to "double", since not all pipe diameters are integers.

Enter the following codes and descriptions:

0	Unknown
12	12"
2	2"
2.25	2 1/4 "
3	3"
4	4"
6	6"
8	8"



4) Save and Run the workspace

Save the workspace and then run it.

View the output feature classes in ArcMap. Query a feature to show how the domain classes have been resolved. You can also right-click the Geodatabase, choose Properties, and then click on the Domains tab to get a list of domains. Clicking on the Diameter entry will show the properties and coded values for the domain you have just created.

Change one of the domain code definitions and re-run the translation. Did the domain definition change? Why/Why-not?

As a final test, use an *AttributeSetter* transformer to set the value of DIAMETER to an invalid code (i.e. one not in the coded domain list). Re-run the translation. Did the translation succeed or fail? Change the "Validate Features to Write" parameter (in the Navigator window) and try again. Is the result different this time?

Subtypes



A Subtype is a data type that defines aspects of a Geodatabase schema related to data classification.

What is a Subtype?

A subtype is a way to define a subset of features within a feature class, as an alternative to creating a different feature class for each set of features.

An attribute in the class stores integer values that define the subtype, and a subtype table contains definitions for each possible integer value.

For instance, a table named “road” may have a field called condition, whose values map to subtype values *good*, *moderate*, and *bad*.

In general, each table can have only one subtype, all the codes have to be unique and valid integers, and all the code:description pairs have to be unique.

A subtype is specific to a particular feature class. It cannot be shared by other classes in the Geodatabase in the same way that a domain can be.



“If data in a feature class has been subdivided using a subtype, then different domains can be applied to each subtype!”

Subtype Writing Scenarios

Because a subtype only applies to a single feature class, it is not possible to create a new table and associate it with an existing subtype. Therefore the scenarios are:

- Write to an existing table with existing subtype
- Write to a new table creating a new subtype

These scenarios will be controlled by a series of parameters, namely:

Data Type	subtype/subtype_codes
Validate Features to Write	Yes/No

Writing to an existing table with an existing subtype

No additional work is required to write to an existing table with an existing subtype. It's not even necessary to set the writer parameter “Validate Features to Write” in order to validate subtype value. A feature with an undefined subtype value will be rejected anyway, with the following error:

For the '<ClassName>' table/feature class the subtype code of '<Value>' is not valid for the subtype field '<SubtypeName>'

Writing to a new table and creating a new subtype/domain

Again it's recommended that you use ArcGIS to create and define subtypes, and simply use FME to enter subtype code values when inserting new features.

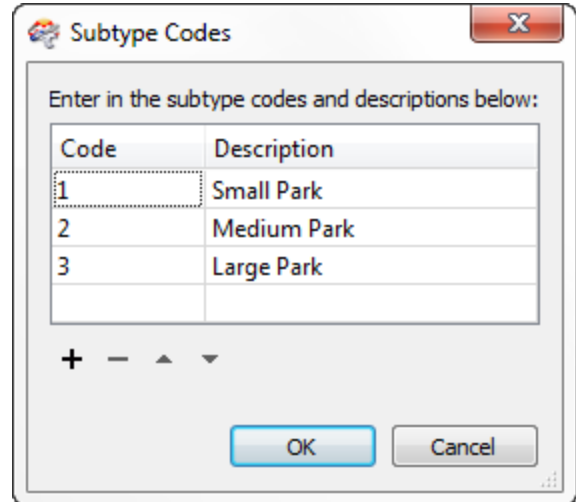
Of course, once again, creating Subtypes is possible with FME.

The process is to define an attribute for use as a subtype and set the correct data type. There are two data types: *subtype_codes* and *subtype*.

The *subtype_codes* data type allows the user to define a code number and description for the subtype. Its edit dialog looks like this:

Here the user has three subtype codes; one for Small Park, one for Medium Park, and one for Large Park.

Any attribute that is not one of these values will result in the translation failing with an error.



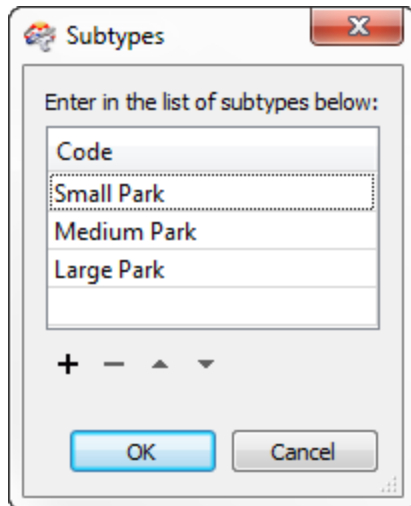
Subtype Codes

Enter in the subtype codes and descriptions below:

Code	Description
1	Small Park
2	Medium Park
3	Large Park

+ - ▲ ▼

OK Cancel



Subtypes

Enter in the list of subtypes below:

Code
Small Park
Medium Park
Large Park

+ - ▲ ▼

OK Cancel

The subtype data type has only a single field in its editing dialog:

The idea here is that the user does not care what code numbers are used for each subtype.

FME will create a unique code number for each incoming integer value.

This option might be useful when you do not know what particular values the incoming data might hold.

Limitations

At the time of writing, FME will not allow you to associate different domains based on a particular subtype. For instance – using the above example – you would not be able to set range domains of 0-50,000; 50,000-100,000; 100,000-250,000 and apply them to the Small, Medium, and Large park subtypes. You would need to create the domain:subtype relationship in ArcGIS to achieve this.

For more information, please see the Esri Geodatabase chapter in the *FME Readers and Writers* manual.



Example 5: Subtypes	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	Create workspace to create subtypes
Demonstrates	Using Subtypes
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri5-Complete.fmw

The task here is to load parks data into a Geodatabase. Parks will be assigned to a different subtype to signify small, medium, and large.

1) Start Workbench

Start Workbench and use the Generate Workspace dialog to create the following translation:

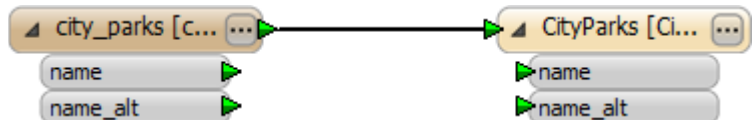
Reader Format MapInfo TAB (MFAL)
Reader Dataset C:\FMEData\Data\Parks\city_parks.tab

Writer Format Esri Geodatabase (File Geodatabase ArcObjects)
Writer Dataset C:\FMEData\Resources\Esri\CityData.gdb

2) Tidy Workspace

Because we know the park features are all polygons, remove any excess feature types and transformers that are inserted automatically. Rename the remaining writer feature type to CityParks. It should have *geodb_polygon* as the permitted geometry type.

The workspace will now look like this:

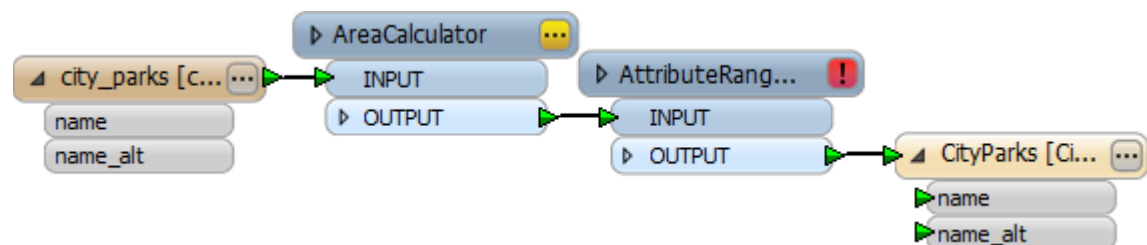


2) Add AreaCalculator

Add an AreaCalculator transformer to calculate each polygon's area. The default parameters will be fine for this example.

3) Add AttributeRangeMapper

Now add an AttributeRangeMapper transformer. This will be used to create subtype values.

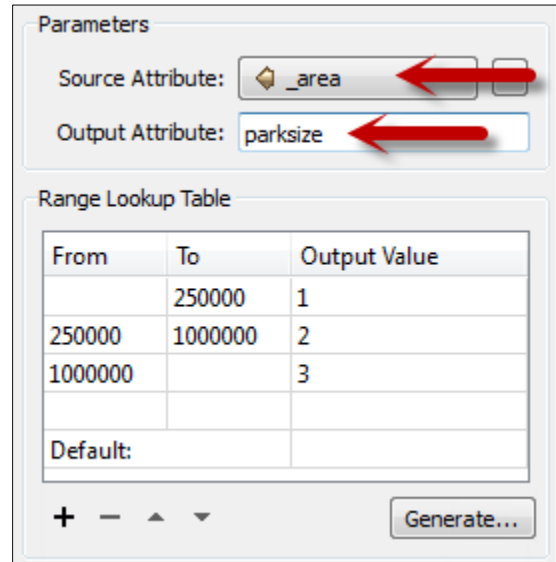


4) Set AttributeRangeMapper Parameters

Open the AttributeRangeMapper's parameters dialog. The parameters should be set as follows:

Source Attribute	Output Attribute	
		_area
		parksize

Lookup Table		
From	To	Output Value
	250000	1
250000	1000000	2
1000000		3



The dialog box shows the 'Parameters' section with 'Source Attribute' set to '_area' and 'Output Attribute' set to 'parksize'. The 'Range Lookup Table' section contains a table with the following data:

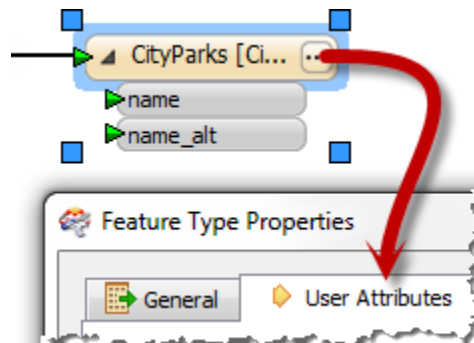
From	To	Output Value
	250000	1
250000	1000000	2
1000000		3
Default:		

Buttons for '+', '-', '▲', and '▼' are at the bottom left, and a 'Generate...' button is at the bottom right.

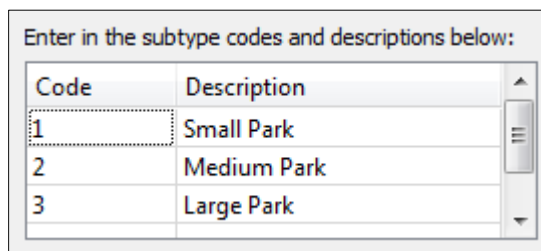
5) Add Subtype Attribute

Open the Writer feature type's properties dialog.

Click the User Attributes tab and add an attribute called *parksize* – the data type should be *subtype_codes*. Click the Edit button for the attribute.



Name	Type	Width	Precision
name	char	64	
name_alt	char	64	
parksize	subtype_codes		Edit...

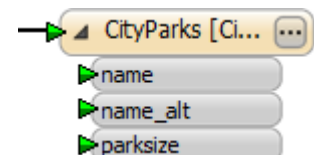


The dialog box is titled 'Enter in the subtype codes and descriptions below:'. It contains a table with the following data:

Code	Description
1	Small Park
2	Medium Park
3	Large Park

Set up subtype codes 1, 2, and 3 for Small, Medium, and Large parks.

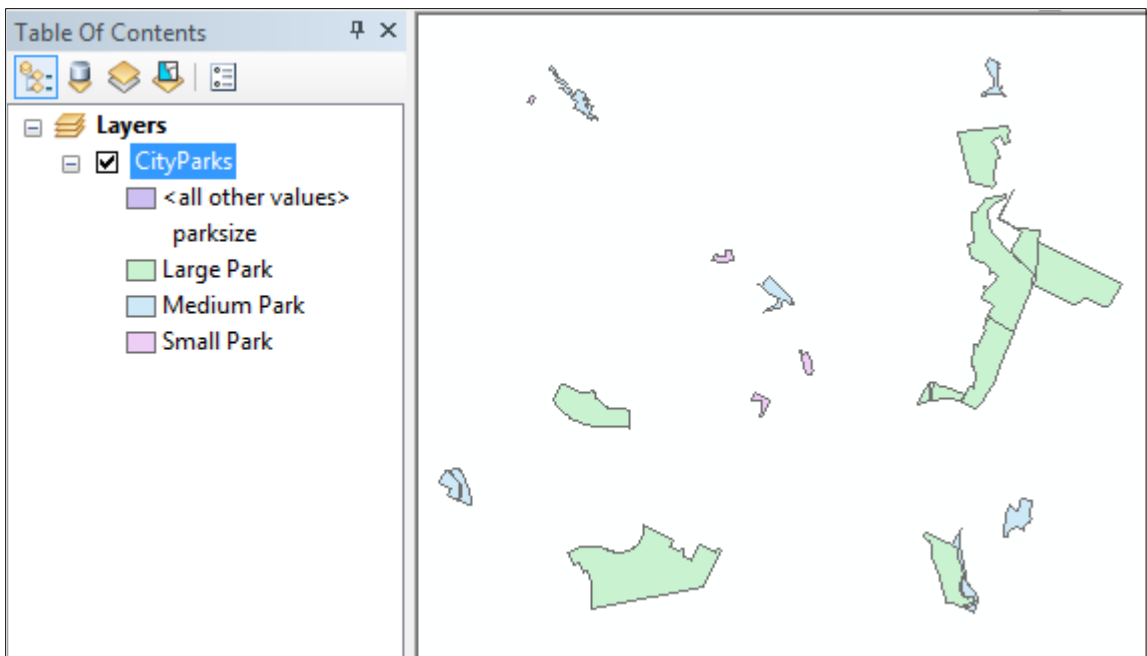
Click **OK** until all dialogs are closed. If all Writer feature type arrows are not green, then check the AttributeRangeMapper parameters.



6) Run Workspace

Check the Geodatabase Writer parameter will not overwrite the existing Geodatabase.

Now save and then run the workspace. Open ArcMap, browse to the park data and drag it into the mapping window. You will see the parks colored according to their subtype:



Field	Value
OBJECTID	10
SHAPE	Polygon
name	Big Walnut Creek
name_alt	Big Walnut Creek Preserve
parksize	Large Park
SHAPE_Length	13459.747948
SHAPE_Area	4735387.127402

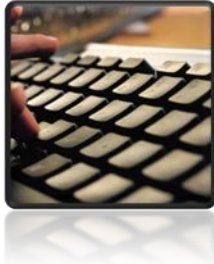
Query a feature and you will see each park has a parksize subtype:

The attribute table will also show a “parksize” subtype for all park features.

You can also see this information in the FME Data Inspector, if you set the Reader parameter “Resolve Subtypes”.

CityParks				
	SHAPE *	name	name_alt	parksize
►	Polygon	East Destination	<Null>	Large Park
	Polygon	Big Walnut Creek	Greenbelt	Large Park
	Polygon	Morris Williams	Golf Course	Large Park
	Polygon	T.A. Brown	School	Small Park
	Polygon	Bartholomew	District	Large Park
	Polygon	Pecan Springs	School	Small Park
	Polygon	Dottie Jordan	<Null>	Medium Park
	Polygon	Andrews	School	Small Park

Geometric Networks



Geometric Networks are relatively simple to understand and deal with in FME.

Geodatabase geometric networks store topological relationships between line and point feature classes in a feature dataset. They help enforce data integrity and connectivity between features and are commonly used for utility and hydrology data.

Writing to Geometric Networks

Currently, FME cannot be used to create geometric networks, or the feature classes participating in them. The geometric network and participating feature classes should be created before the translation using ArcGIS.

FME can therefore be used to populate existing network feature classes with point and line data or, alternately, to create simple point and line feature classes whose geometric network is created as a post-processing step.

A format attribute - `geodb_ancillary_role` - is used to define which features are sinks or sources. Possible values are *none*, *source* or *sink*.

Reading from Geometric Networks

A number of format attributes (most of which store connectivity information) are populated when reading from a Geodatabase network.

For example, the following attributes are populated when reading a simple edge feature:

- `geodb_element_id`
The logical network element ID of the junction
- `geodb_from_junction_element_id`
The junction element ID that corresponds to the from endpoint
- `geodb_to_junction_element_id`
The junction element ID that corresponds to the to endpoint

See the Readers and Writers Manual for the format attributes available for simple junction and complex edge features.

Performance

When reading from a geometric network, all of the connectivity information must be verified. Therefore, reading is faster if the network information is ignored. This can be achieved using the "Ignore Network Info" parameter for the Geodatabase reader.

A similar parameter exists for writing. A geometric network won't be formed by the features written, but the loading process will take place much more quickly.

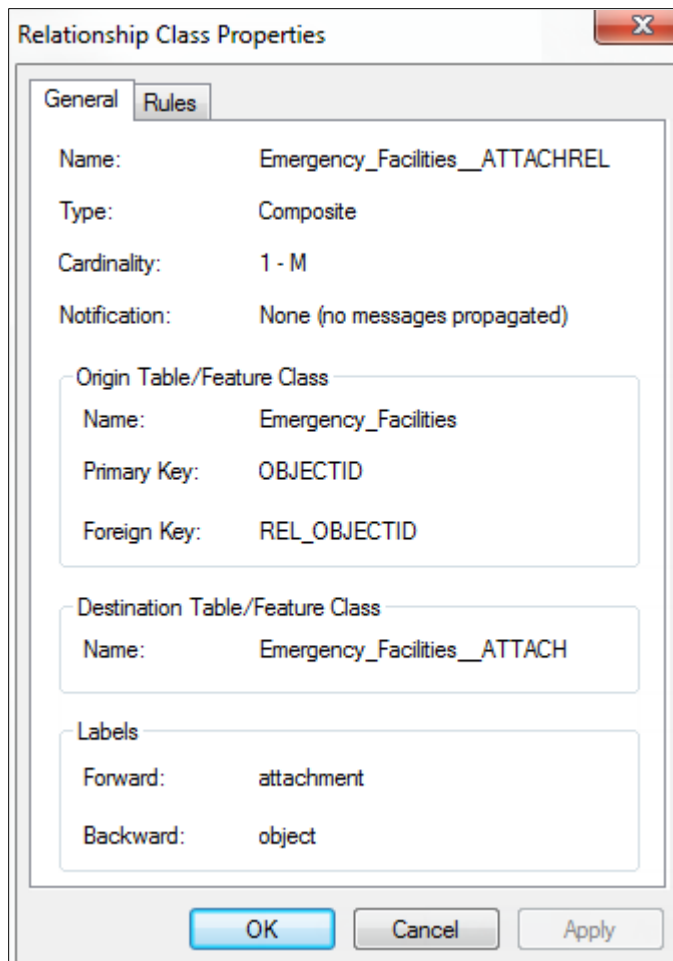
Relationship Classes



Handling Relationship Classes involves working with keys to define the relationships between features.

Geodatabase relationship classes are used to manage the relationships between features in one class with features in another. Attributed and non-attributed relationship classes can be read and written to with FME. Relationships are not rows in a table or feature class like other features, but rather implied through the primary and foreign key values of an origin and destination feature.

“Origin” features belong to the “Origin Table/Feature Class” specified when creating the relationship class in ArcCatalog, and “destination” features belong to the “Destination Table/Feature Class”.



Relationship Class Properties

General **Rules**

Name: Emergency_Facilities__ATTACHREL

Type: Composite

Cardinality: 1 - M

Notification: None (no messages propagated)

Origin Table/Feature Class

Name: Emergency_Facilities

Primary Key: OBJECTID

Foreign Key: REL_OBJECTID

Destination Table/Feature Class

Name: Emergency_Facilities__ATTACH

Labels

Forward: attachment

Backward: object

OK Cancel Apply

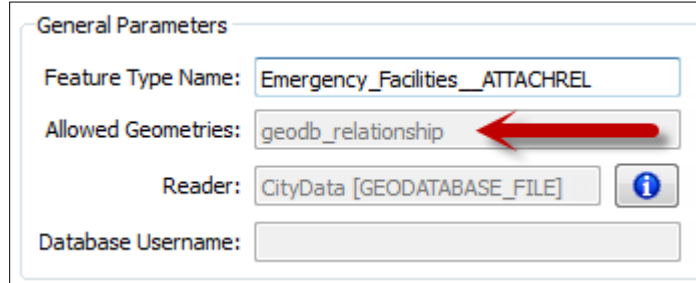
In this example, Emergency Facilities (the Origin) are related to a set of Attachments (the Destination).

In a “Simple” Relationship class, features can exist independently of each other – for example an emergency facility can exist without necessarily having an attachment.

In an “Attributed” (or “Composite”) Relationship class, origin and destination objects cannot exist apart. An Emergency Facility would always have an Attachment, and if the facility were deleted so too would be the related attachment record.

Reading Relationship Classes

When reading a relationship class both the origin and destination feature classes must be read at the same time as the relationship class. This is done by selecting all of these tables to be read.



A relationship feature type shows an allowed geometry of either *geodb_relationship* or *geodb_attributed_relationship*

Each relationship feature has the following Format Attributes stored on it when read from a relationship class:

- *geodb_rel_origin_oid*
the OID (ObjectID) of the related origin feature
- *geodb_rel_destination_oid*
the OID (ObjectID) of the related destination feature

Exposed	Name	Type
<input checked="" type="checkbox"/>	geodb_rel_destination_oid	integer
<input checked="" type="checkbox"/>	geodb_rel_origin_oid	integer
<input type="checkbox"/>	geodb_right_to_left	boolean

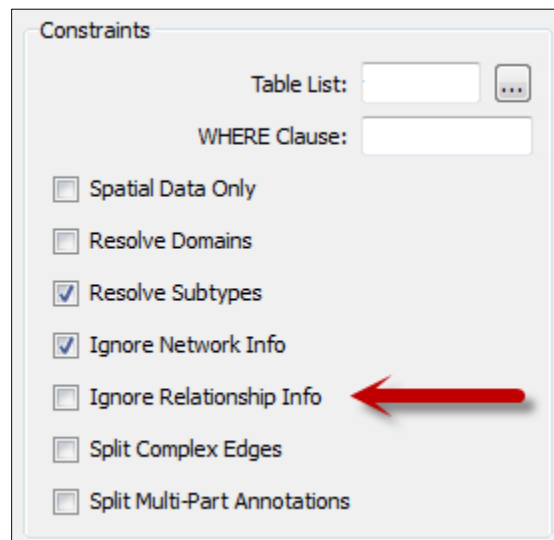
These attributes can be used to identify relationships between the origin and destination features within the workspace.

Performance

Note that reading from relationship classes is very slow since each relationship is validated when read.

Because this functionality is not often required, FME includes a parameter that, by default, turns off relationship reading to improve performance.

Therefore, to read relationship classes you must first locate and deactivate this parameter.



Writing Relationship Classes

Relationship classes cannot be created through FME, and must be set up through ArcGIS prior to running the translation.

Then, in FME, the following attributes must be stored on features written to a relationship class:

- *geodb_rel_origin_oid*
the OID (ObjectID) of the related origin feature
- *geodb_rel_destination_oid*
the OID (ObjectID) of the related destination feature
- *geodb_type*
either *geodb_relationship* or *geodb_attributed_relationship*

The following attribute must be stored on features written to the origin and destination feature classes/tables:

- *geodb_oid*
the OID (ObjectID) of the feature (matches the *geodb_rel_origin_oid* on the origin and the *geodb_rel_destination_oid* on the destination)
- *geodb_feature_has_relationships*
Set to "YES" to specify that a feature participates in a relationship as an origin or destination.

Note that the OID (ObjectID) values mentioned above are not transferred to the Geodatabase. They are only supplied so the Geodatabase writer knows which features are related.

Required attributes for writing to relationship classes:

Object	Required Attributes
Origin feature class or table	<i>geodb_oid</i> <i>geodb_feature_has_relationships</i> = yes
Destination feature class or table	<i>geodb_oid</i> <i>geodb_feature_has_relationships</i> = yes
Relationship class	<i>geodb_rel_origin_oid</i> <i>geodb_rel_destination_oid</i> <i>geodb_type</i> = <i>geodb_relationship</i> or <i>geodb_attributed_relationship</i>

For example, if an origin feature has *geodb_oid* = 1 and a destination feature has *geodb_oid* = 2, the feature written to the relationship table must have these attributes:

geodb_rel_origin_oid = 1
geodb_rel_destination_oid = 2



Be careful if you have more than one Geodatabase writer in your workspace. The origin and destination feature classes (or tables) that participate in the relationship and the relationship class must be written by the same Geodatabase writer (i.e. you cannot write to feature classes with one Geodatabase writer and to relationship classes with another).

Attributed relationships can be inserted, updated and deleted, while non-attributed relationships can only be inserted and deleted. Attributed relationships have intermediate tables associated with them, which can be updated by providing an RID (relationship id) as a key field, much as an OBJECTID must be provided when updating a table or feature class.

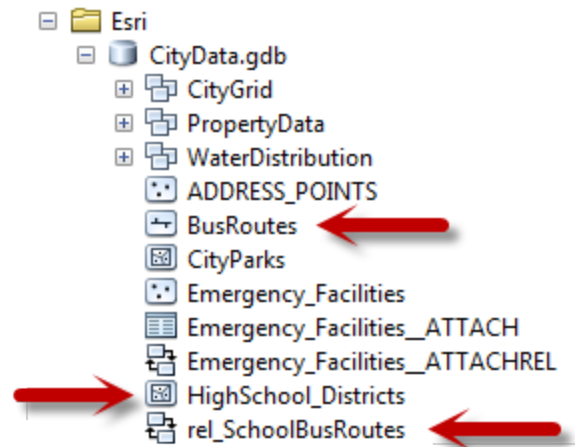


Example 6: Relationship Classes	
Scenario	FME user; City of Interopolis, Planning Department
Data	Transit, Schools
Overall Goal	Create a relationship between transit and school features
Demonstrates	Creating relationship classes
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri6-Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri6-Complete.fmw

The City of Interopolis would like to know which bus lines service their high school districts. They have GIS datasets of their high school districts and bus routes, and wish to migrate this data to a Geodatabase in order to populate a relationship class between districts and bus routes.

1) Start ArcMap

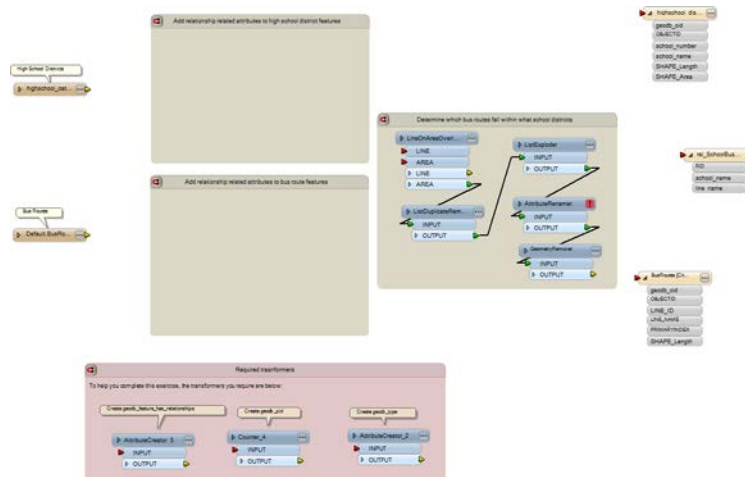
In an ArcMap catalog window, examine tables in the file *Geodatabase C:\FMEData\Resources\Esrri\CityData.gdb*.



Notice there are tables for bus routes and school districts, plus a relationship table.

2) Start Workbench

Start Workbench and open the pre-defined workspace.







This workspace has been partially set up. All that needs adding are the relationship attributes.

The transformers in the pink-colored bookmark can be used, rather than fetching new ones from the transformer gallery.

3) Adjust Writer Parameter – Transaction Type

In the Navigator window, set the transaction type to Edit Session. Relationship Classes can only be set in an Edit Session.

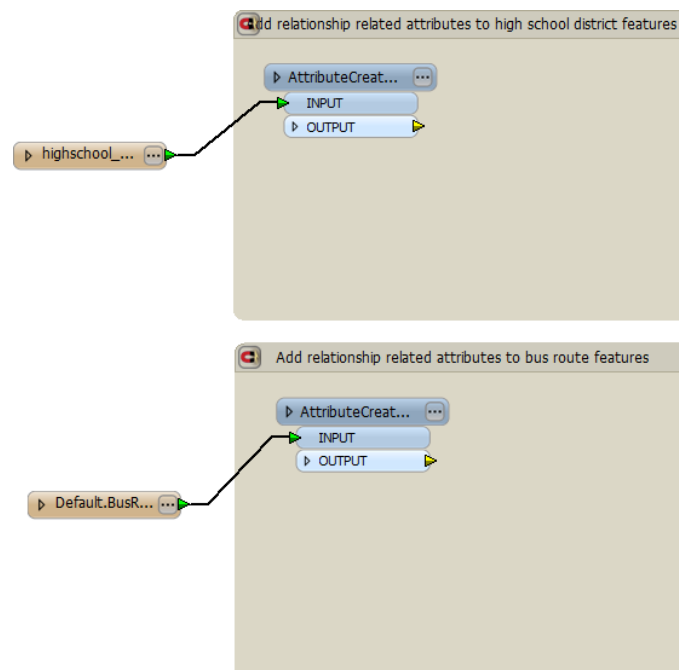
Also ensure that Overwrite Existing Geodatabase is set to No and that the Template File field is not set. We do not wish to overwrite the entire Geodatabase, nor add new tables.

-  Destination Esri File Geodatabase: C:\FMEData\Resources\Esri\CityData.gdb
-  Overwrite Existing Geodatabase: No
-  Transaction Type: Edit Session
-  Template File: <not set>

4) Copy AttributeCreator

To create one of the required relationship attributes (*geodb_feature_has_relationships*) copy the first *AttributeCreator* in the “Required Transformers” bookmark, and paste it after each of the source feature types.

Open the parameters dialog for each in turn, and set the value of the attribute *geodb_feature_has_relationships* to “yes”.

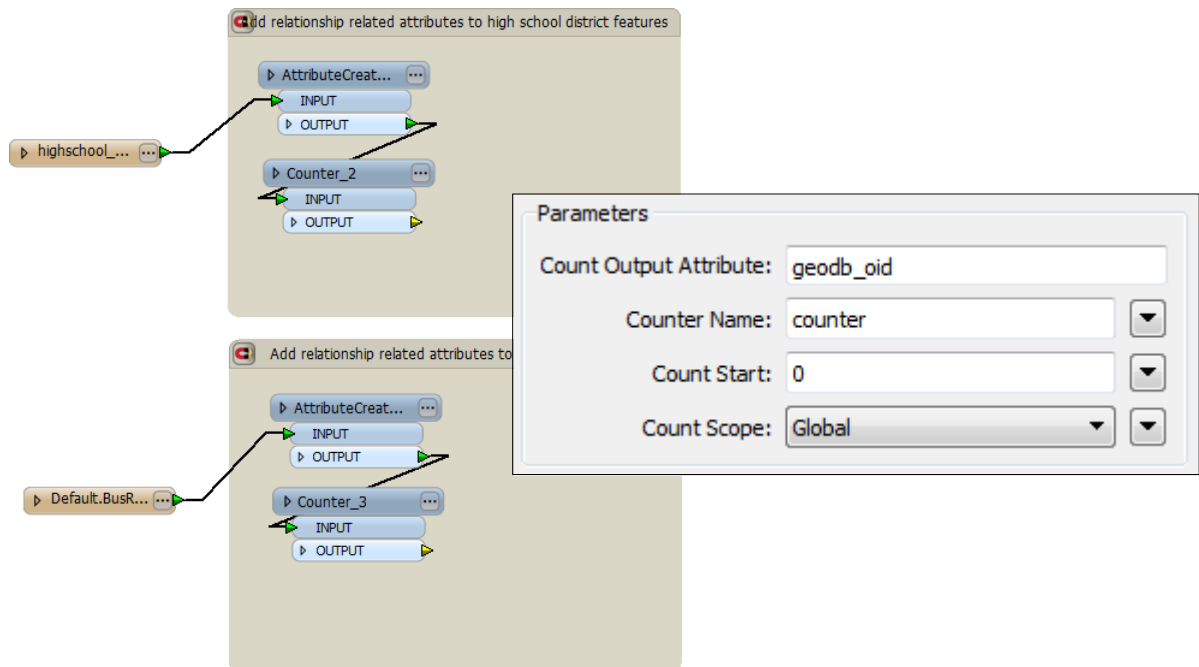


Attributes To Set	
Attribute Name	Value
geodb_feature_has_relationships	<i>k</i> yes

5) Copy Counter

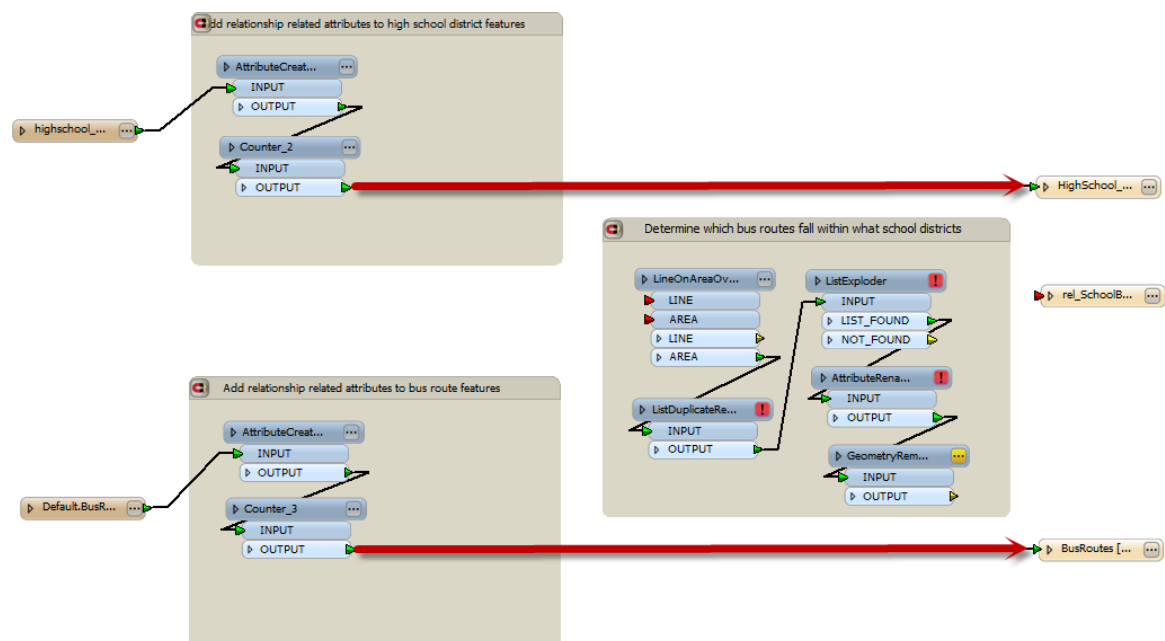
A unique id (geodb_oid) must exist on features that participate in the relationship class (high school districts and bus routes). These IDs are referenced when writing to the relationship class.

Copy and paste the *Counter* transformer from the “Required transformers” bookmark and place one after each *AttributeCreator*. Notice that the count output attribute is set to *geodb_oid*.



6) Make Connection to Destination Feature Types

Connect the outputs from the *Counter* transformers to the two Writer feature types (high school and BusRoutes). These are the actual features to participate in the relationship.

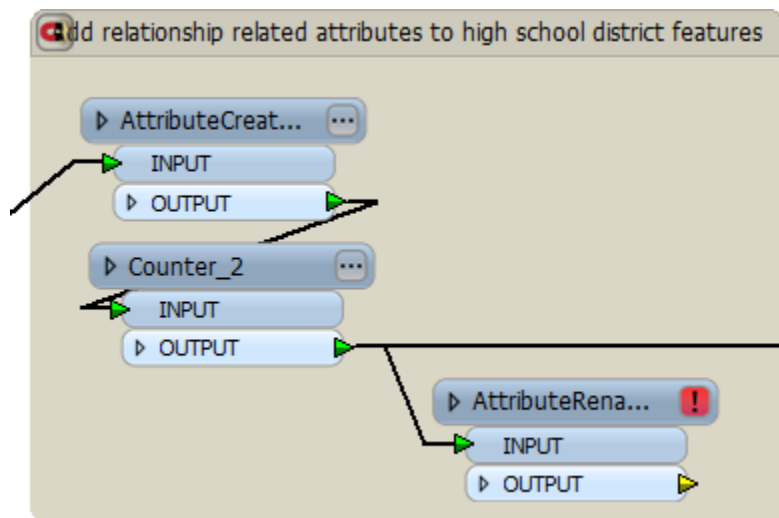


7) Add AttributeRenamer

Now the basic tables have been defined, it's time to work on the setting up the attributes for the relationship class.

Features written to the relationship class must have both a *geodb_rel_origin_oid* and a *geodb_rel_destination_oid*. The values for these attributes must match the *geodb_oid* values written to the origin and destination features.

Add an *AttributeRenamer* transformer (using Quick Add) after each *Counter* and set it up to rename the *geodb_oid* attribute to either *geodb_rel_origin_oid* or *geodb_rel_destination_oid*.



Points to Note!

Because these features are destined for the relationship table, the *AttributeRenamer* should be placed in parallel (i.e. a duplicate stream of data).

The *geodb_oid* attribute will need renaming to either *geodb_rel_origin_oid* or *geodb_rel_destination_oid* (depending on which set of features is the origin and which the destination – if unsure then check the feature class properties in ArcMap)

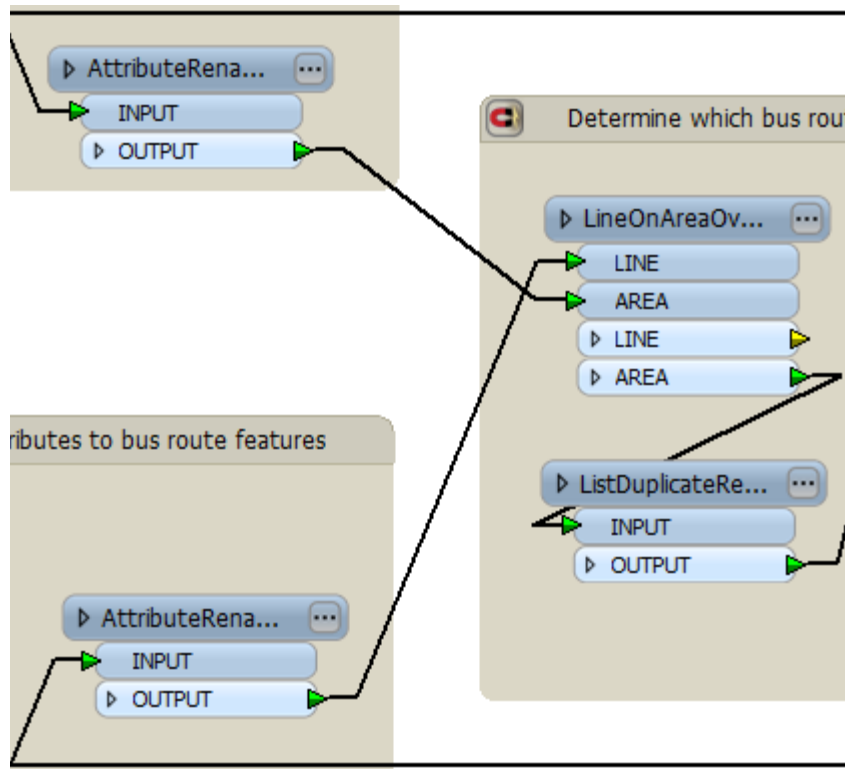
8) Connect to LineOnAreaOverlayer

The bus route features now have a *geodb_rel_destination_oid* and the high school district features should now have a *geodb_rel_origin_oid*.

However, features written to the relationship class must carry both of these attributes, and this can be achieved by merging/joining the bus/school features on a spatial relationship basis.

Notice the *LineOnAreaOverlayer* transformer. This will effectively carry out a spatial join.

Connect bus route features to the LINE input port and schools to the AREA input port.



School features will emerge from the AREA output port of the *LineOnAreaOverlayer*. These features carry a "list" of all of the bus routes that overlap them.

Notice that duplicate list items are removed from the features using a *ListDuplicateRemover*, and the list is exploded with a *ListExploder* in order to create one feature only for each unique school district/bus route combination.

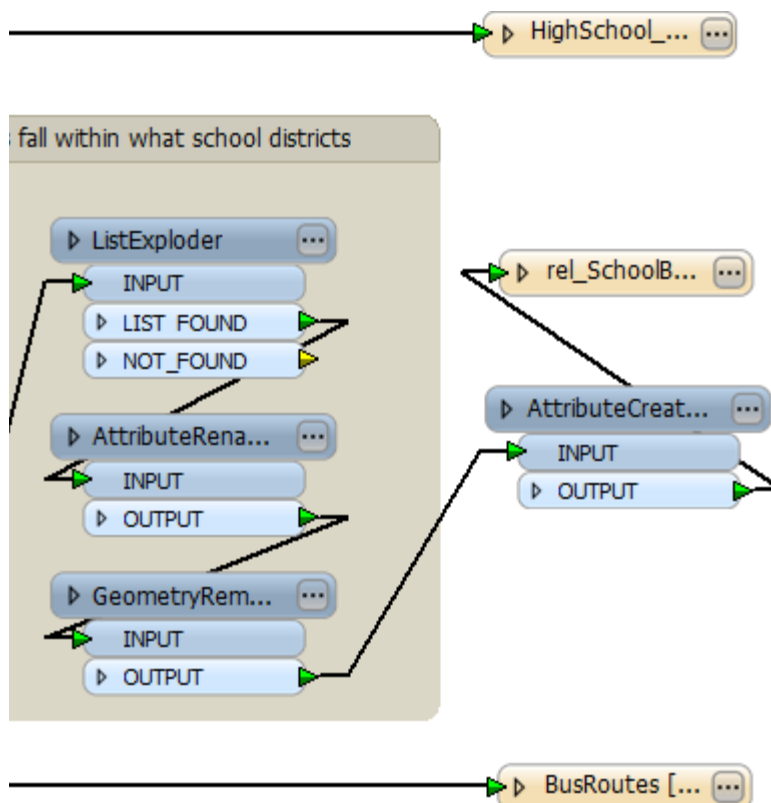
LINE_NAME is renamed to line_name to reflect the case of the field in the Geodatabase, and finally the geometry removed with a *GeometryRemover*.

9) Copy AttributeCreator

The final task is to add an attribute called *geodb_type*

Do this with an *AttributeCreator* transformer. You can copy and paste one from the “Required Transformers” bookmark.

Adjust the parameters to create an attribute *geodb_type* with the value *geodb_relationship*

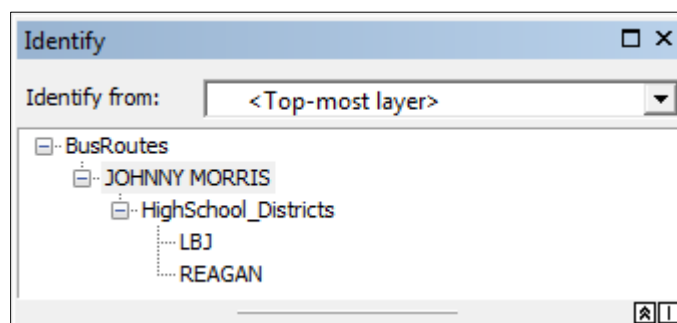


10) Run Workspace

Let's check the data before it's written out to the Geodatabase. From the Writers menu, select the option to Redirect to Inspection Application. Run the workspace.

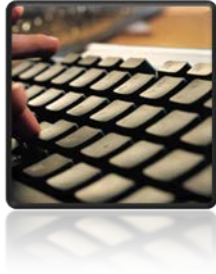
Verify that *geodb_oid* is populated on spatial features, and that *geodb_rel_origin_oid* and *geodb_rel_destination_oid* is populated on non-spatial features.

Turn off the Redirect to Inspection Application option. Save the workspace and then run it. Examine the results in ArcGIS to confirm the transformation worked.



Be careful if you decide to re-run this workspace and set the “Truncate Table First” parameter to “Yes” on all of the destination feature types. You will get a warning in the log saying the relationship class cannot be truncated and end up with duplicate features in your relationship class. You must delete the rows of the relationship class in ArcGIS before re-running this workspace – something you could again do with a Python script.

ArcGIS Attachments



Attachments are another use of a relationship class in ArcGIS.

Attachments

ArcGIS attachments are a way to connect additional information to features, in the form of a specific file; for example an image, a PDF, or a text document.

As the ArcGIS documentation mentions:

“For example, if you have a feature representing a building, you could use attachments to add multiple photographs of the building taken from several angles, along with PDF files containing the building's deed and tax information.”

You can attach one or more files to a feature, and then retrieve the information using query tools in ArcGIS.

Attachments and FME

Because attachments are handled by a relationship class, FME is capable of easily creating this sort of connection. Attachments are added into a new table and FME can also create this. The key is to read the contents of the file to be attached into an attribute, and writing that attribute to a DATA field in the attachments table.

An *AttributeFileReader* transformer or the new Data File reader can be used to read the contents of a file into an attribute. In the case of the Data File reader, be sure to set the reader parameter “Read Whole File at Once” to Yes.

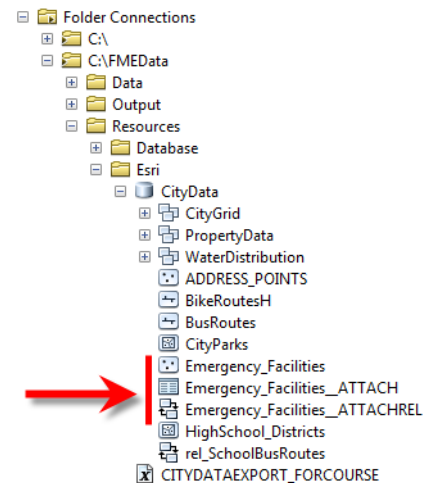
Example 7: Attachments	
Scenario	FME user; City of Interopolis, Planning Department
Data	Emergency Facilities
Overall Goal	Attach photographic images to emergency facility features
Demonstrates	Esri ArcGIS Attachments
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri7-Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Esri7-Complete.fmw

The city has a database of emergency facilities and wishes to merge in photographic images as attachments in a Geodatabase.

1) Start ArcMap

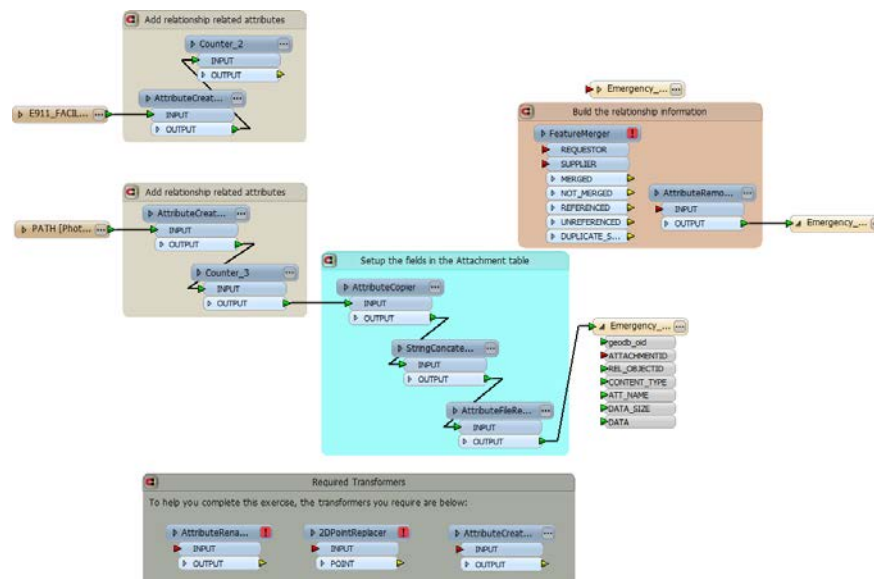
In an ArcMap catalog window, examine tables in the file Geodatabase *C:\FMEData\Resources\Esri\CityData.gdb*.

Notice there are tables for emergency facilities and a relationship table:



2) Start Workbench

Start Workbench and open the pre-defined workspace.



This workspace is partially set up.





As this is similar to the previous example, there are more transformers already in place.

The transformers in the grey-colored bookmark can be used, rather than fetching new ones from the gallery.

3) Adjust Writer Parameter – Transaction Type

In the Navigator window, set the transaction type to Edit Session. Relationship classes must be written to in an edit session.

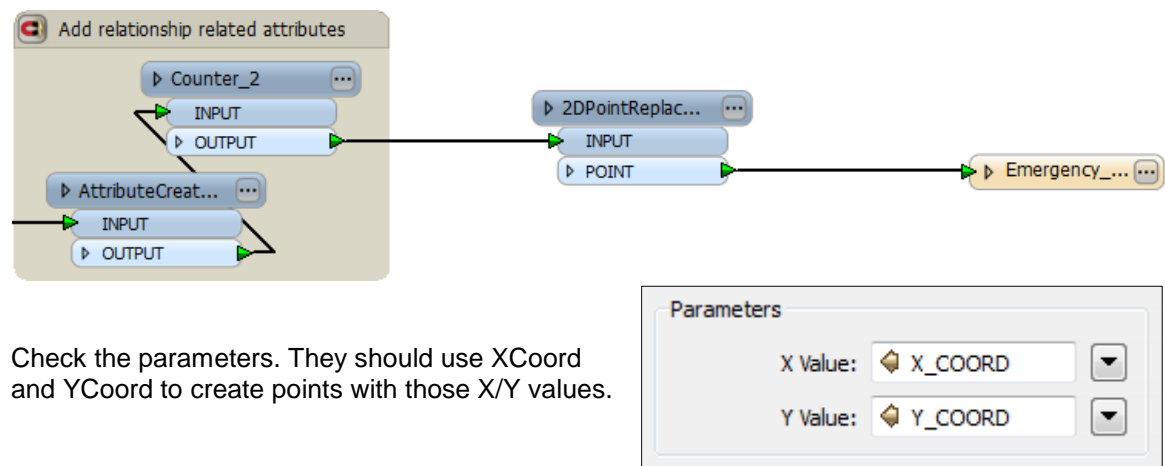
Also ensure that Overwrite Existing Geodatabase is set to No and that the Template File field is not set. We do not wish to overwrite the entire Geodatabase, nor add new tables.

-  Destination Esri File Geodatabase: C:\FMEData\Resources\Esri\CityData.gdb
-  Overwrite Existing Geodatabase: No
-  Transaction Type: Edit Session
-  Template File: <not set>

4) Add 2DPointReplacer

The emergency facility features are – currently – non-spatial, because they have been read from a Microsoft Access database. They must be turned into spatial features.

Insert the 2DPointReplacer transformer from the Required Transformers bookmark, into the connection to the emergency facilities Writer feature type.

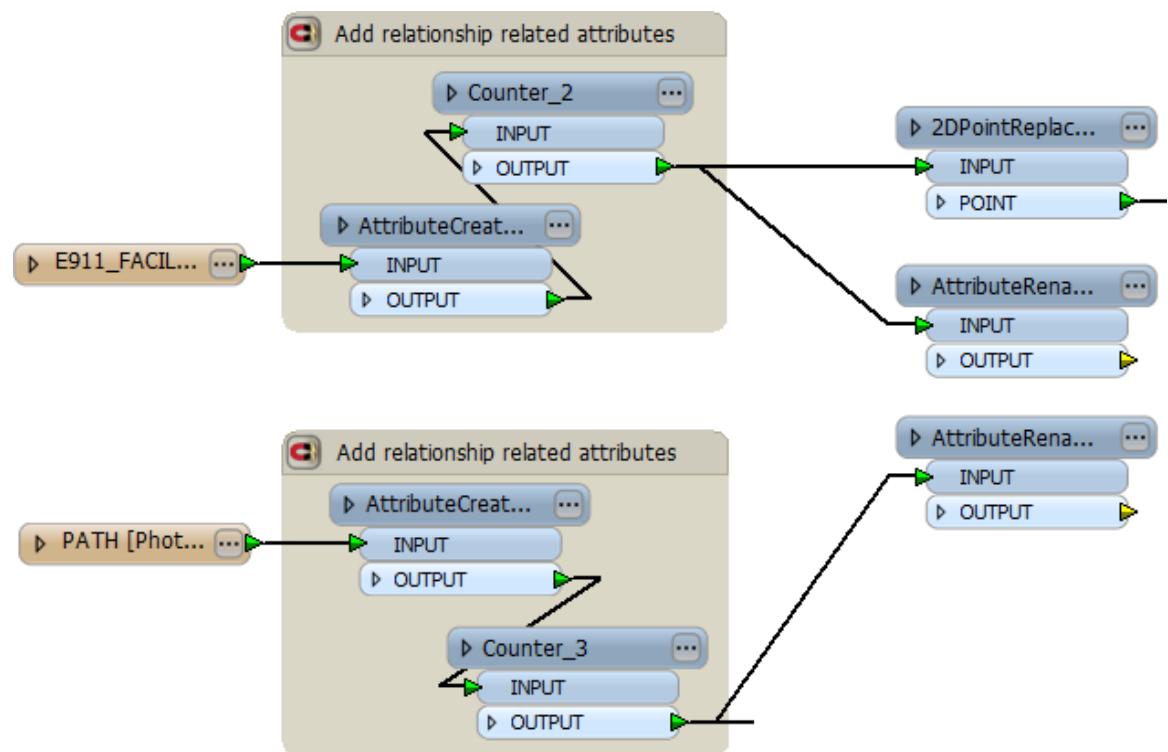


5) Add AttributeRenamers

Now the basic tables have been defined, it's time to work on the setting up the attributes for the relationship class.

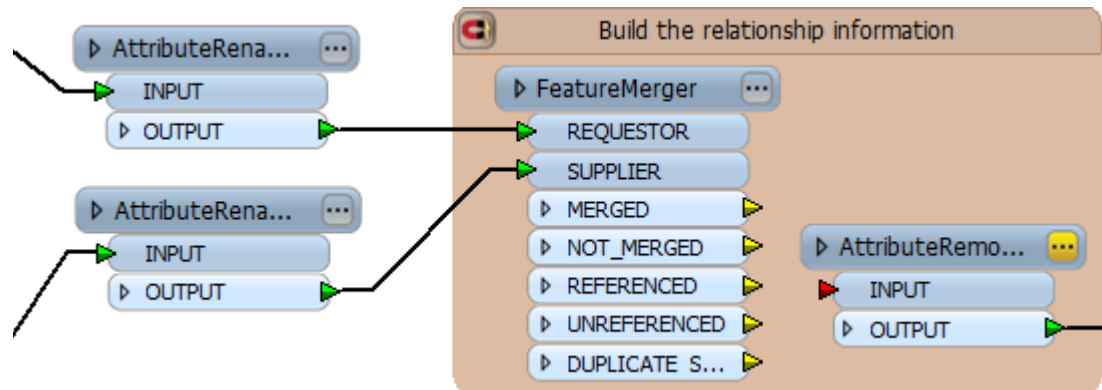
Features written to the relationship class must have both a *geodb_rel_origin_oid* and a *geodb_rel_destination_oid*. The values for these attributes must match the *geodb_oid* values written to the origin and destination features.

Add an *AttributeRenamer* transformer (copied from the bookmark again) after each *Counter* and set it up to rename the *geodb_oid* attribute to either *geodb_rel_origin_oid* or *geodb_rel_destination_oid*, depending on which set of features is the origin and which the destination – if unsure then check the feature class properties in ArcMap.



6) Connect to FeatureMerger

To make a join, connect the two AttributeRenamers to the pre-existing FeatureMerger:



This will join the features together based on the name of the photo image. This name comes from the filename (in the case of the image) and an attribute (in the case of the emergency facility).

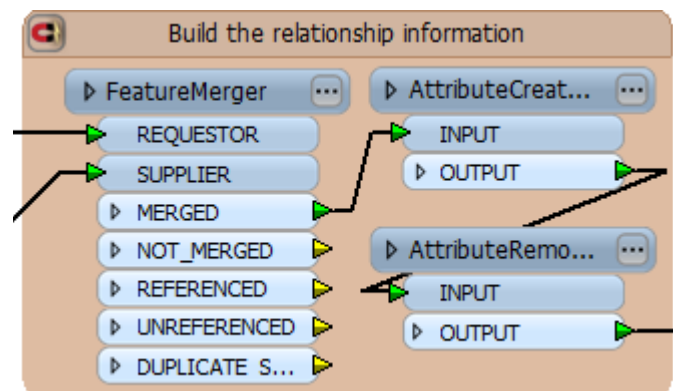
7) Copy AttributeCreator

The next task is to add an attribute called *geodb_type*

Do this with an *AttributeCreator* transformer. You can copy and paste the one from the "Required Transformers" bookmark.

Attach it to the FeatureMerger:MERGED output port and then afterwards to the AttributeRemover:INPUT port.

Adjust the parameters to create an attribute *geodb_type* with the value *geodb_relationship*



8) Save and Run Workspace


Save the workspace and then run it.

Examine the table
Emergency_Facilities in ArcGIS to
confirm the transformation worked.

If you query a feature using the
HTML Popup button, you should see
the attached photo image.

STATION 18

STATION 18



AustinFireStation18.JPG

SHAPE	Point
IDR_ID0	2
IDR_ID	2
ADDRESS	6311 BERKMAN DR
NAME	STATION 18
FACILITY	FIRE STATION
TYPE_	10
STATUS	IFD BUILT
STA_NUM	18
COMMAND	
ENTITY	
PROP_DATE	
FACILITY2	
COLOCATE	N
NAME2	
UNIT	(E18, T18)
GAATN	Yes
LADDER	L18
SECTORS_	6
JURIS_	FULL
NAME_FULL	AustinFireStation18

Database Transformers



A number of transformers exist specifically to communicate with databases

FME contains a number of Workbench transformers specifically designed for use with databases. These come under the Database category in the transformer gallery. Such transformers are commonly used to query a database, but can also be used to dispatch updates and insertions or perform spatial relationships with data residing in a database.

For our purposes, Geodatabase is one example of such a spatial database.

Why use a Transformer?

Transformers are sometimes preferable to using the writer to carry out updates, because you may wish to only apply a change to a small subset of data, or you may wish to use a special where clause that isn't available when you choose a writer UPDATE mode.

However, unless you need these functions for a specific reason, you should use a writer instead.

FME Database Transformers

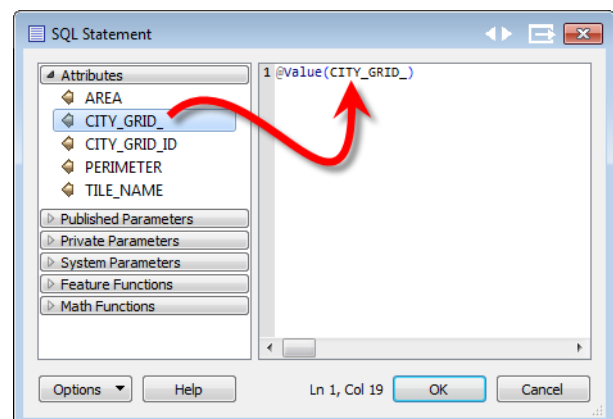
For these transformers, one SQL or other database command is issued for each input feature. Output features may be entirely new, if selected from the database by a query.

SQLExecutor

The *SQLExecutor* is a transformer for issuing SQL commands to a database.

For Geodatabase these are usually simple *select* statements. We don't recommend using a *SQLExecutor* to insert, delete or update records in a Geodatabase as the actions occur directly to the underlying spatial database and bypass the versioned Geodatabase that manages the data.

Attributes are accessed using the SQL editing dialog, and are represented by the FME function `@Value()`:



For example:

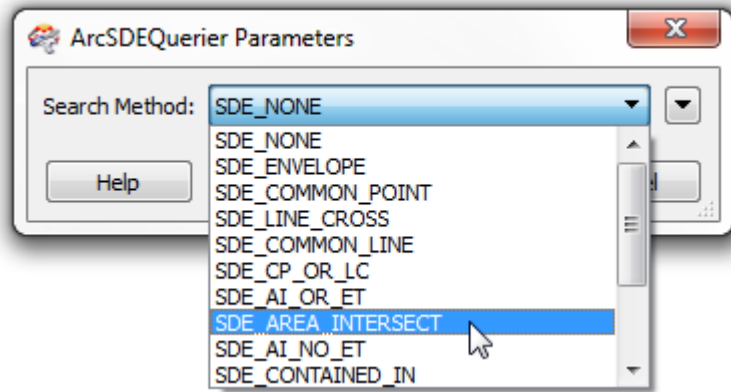
```
delete from ADDRESS_POINTS
where PRIMARYINDEX = @Value(PRIMARYINDEX)
```


ArcSDEQuerier

The *ArcSDEQuerier* is a transformer for issuing commands to an ArcSDE database.

This transformer can issue update and delete commands, but – as mentioned – it's better to use the SDE writer where possible.

In query mode a full set of spatial interactions is available.



FeatureReader

The *FeatureReader* transformer can be used to read any FME-supported format of data.

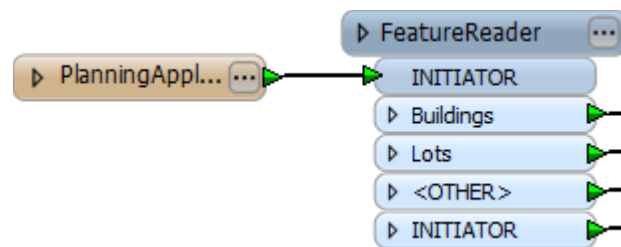
The first use of this transformer is to simply read a dataset, just like a Reader.

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 example, 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 Geodatabase.



Example 8: Database Transformers	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Grid, Addresses (File Geodatabase)
Overall Goal	To read all addresses within a user-defined grid square
Demonstrates	Database transformers, particularly the FeatureReader
Starting Workspace	None
Finished Workspaces	C:\FMEData\Workspaces\PathwayManuals\Escri8-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Escri8-Complete-Advanced.fmw

The task here is to read all addresses within a (user-selected) specific city grid. Because the addresses do not have a city grid cross-reference this will have to be done with a spatial, rather than non-spatial, query.

One method would be to read the entire address dataset, then filter it against the chosen grid square. However, a more efficient way will be to use a *FeatureReader* transformer.

1) Start Workbench

Start Workbench and begin with an empty workspace. The first task is to read the city grid.

Use Readers > Add Reader to add a reader to read the CITY_GRID class from the CityData Geodatabase:

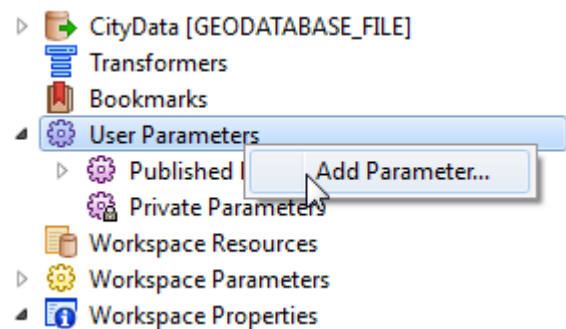
Source Format Esri Geodatabase (File Geodatabase ArcObjects)
Dataset C:\FMEData\Resources\Escri\CityData.gdb

When prompted (or in the parameters dialog) choose CITY_GRID as the table to read.

2) Add User Parameter

A published parameter will allow the user to select which grid square to read.

In the Navigator window, locate and right-click User Parameters. Choose Add Parameter.

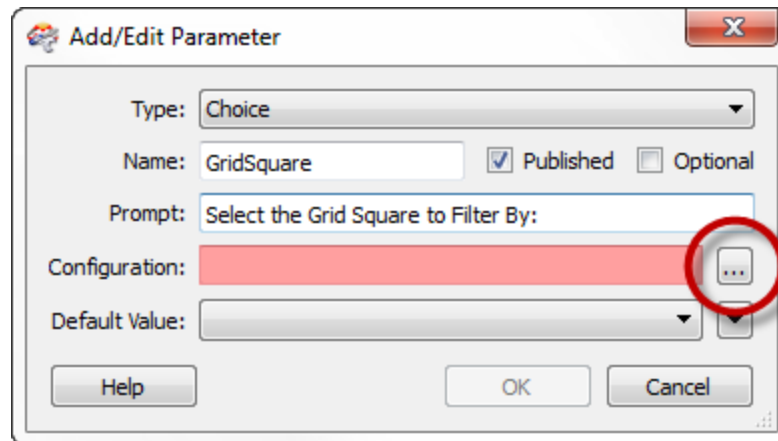


Define the new parameter as follows:

Type	Choice
Name	GridSquare
Published	Yes
Optional	No
Prompt	Select the Grid Square to Filter By:

A Choice parameter lets the user select a grid square from a list, rather than entering it manually.

3) Configure User Parameter



The 'Add/Edit Parameter' dialog box is shown. The 'Type' is set to 'Choice'. The 'Name' is 'GridSquare'. The 'Published' checkbox is checked. The 'Prompt' is 'Select the Grid Square to Filter By:'. The 'Configuration' field is highlighted in red. The 'Default Value' is empty. The 'Import...' button is circled in red.

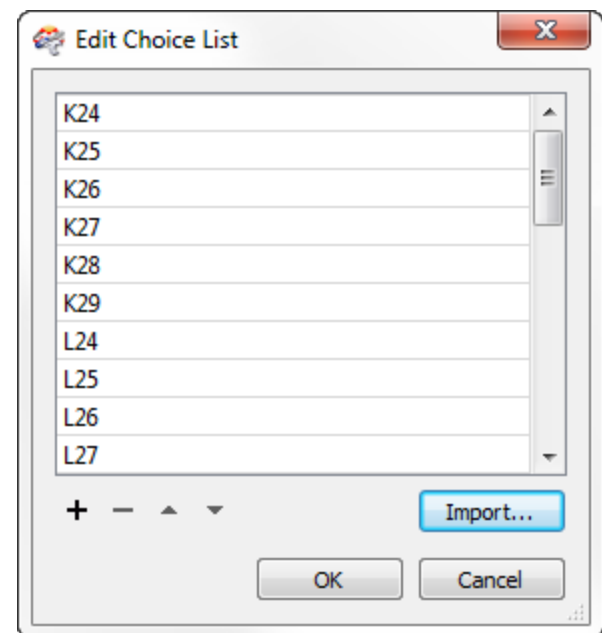
For configuration of the available choices, click the [...] button to open the Edit Choice List dialog.

We could define all the tiles manually, but this would take time and we might make an error. Instead, click the Import button. This will open the Import Wizard, which can be used to scan the grid table to ascertain all possible TILE_NAME values.

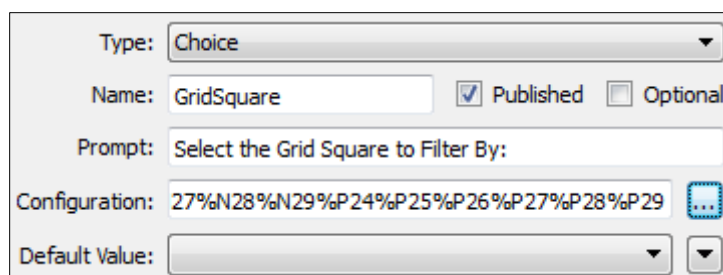
In the Import Wizard, set the format to File Geodatabase and select the CityData Geodatabase. Select CITY_GRID as the feature type to read. Select TILE_NAME as the attribute to read.

FME will now scan the CITY_GRID table for a list of tile names (there should be 30 values).

Click **Import** to complete the process. The choice list will now look like this:



The 'Edit Choice List' dialog box is shown. It contains a list of tile names: K24, K25, K26, K27, K28, K29, L24, L25, L26, L27. The 'Import...' button is highlighted in blue.



The 'Add/Edit Parameter' dialog box is shown again. The 'Configuration' field now contains the text '27%N28%N29%P24%P25%P26%P27%P28%P29'. The 'Import...' button is still highlighted in blue.

Click OK to close all dialogs and create the new User Parameter.

4) Add Tester

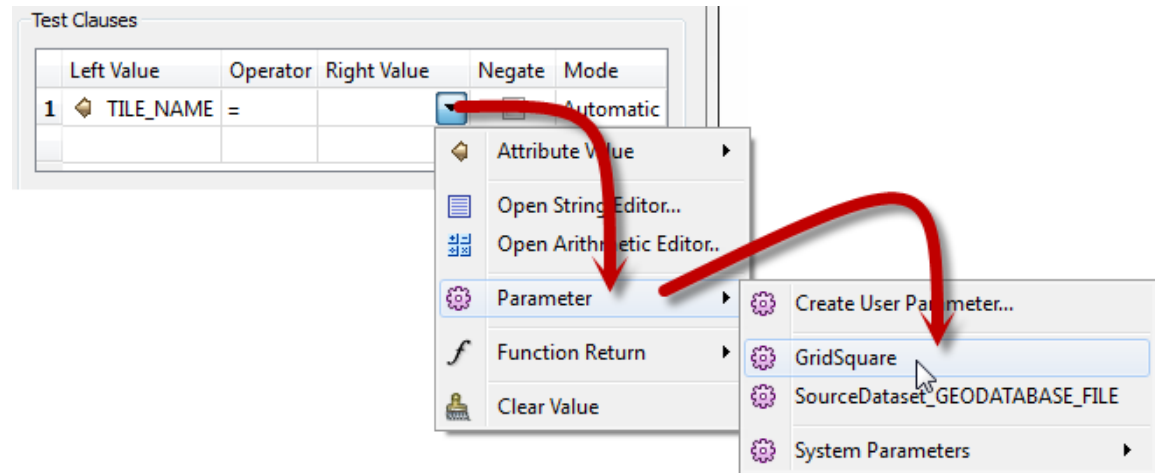
Now the user can choose which tile to use, we need to apply that choice to the incoming data. This can be done with a Tester transformer.

Add a *Tester* transformer. Open the *Tester* parameters dialog.

For the Left Value, select Attribute Value > TILE_NAME.

For the Operator field, select the equals (=) sign.

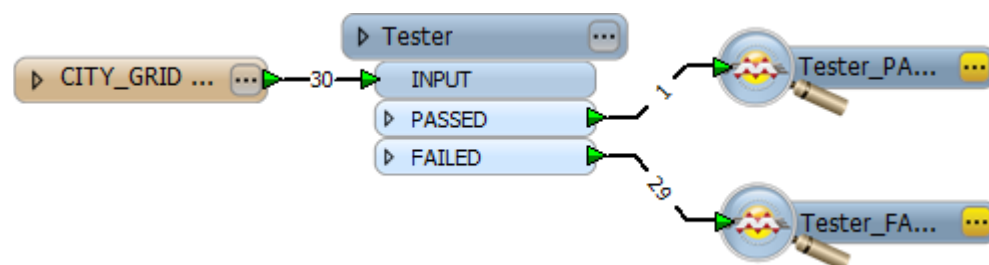
In the Right Value field choose Parameter > GridSquare (the newly created user parameter):



5) Test Workspace

At this point you may wish to test this part of the workspace.

Connect Inspector transformers to the PASSED and FAILED ports and run the workspace using File > Prompt and Run to ensure the Tester and Published Parameter components work correctly.



6) Add FeatureReader

Once we have isolated the required grid feature, we can use it in a spatial query using the FeatureReader transformer.

Add a FeatureReader transformer connected to the PASSED port of the *Tester* transformer.

7) Define FeatureReader Parameters

Open the FeatureReader parameters wizard.

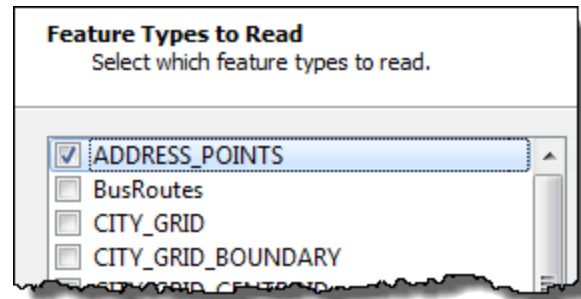
The first panel asks for the location of the data to query. This is the location of the address data:

Source Format
Dataset

Esri Geodatabase (File Geodatabase ArcObjects)
C:\FMEData\Resources\Esri\CityData.gdb

The second panel asks which feature types (tables/classes) to read.

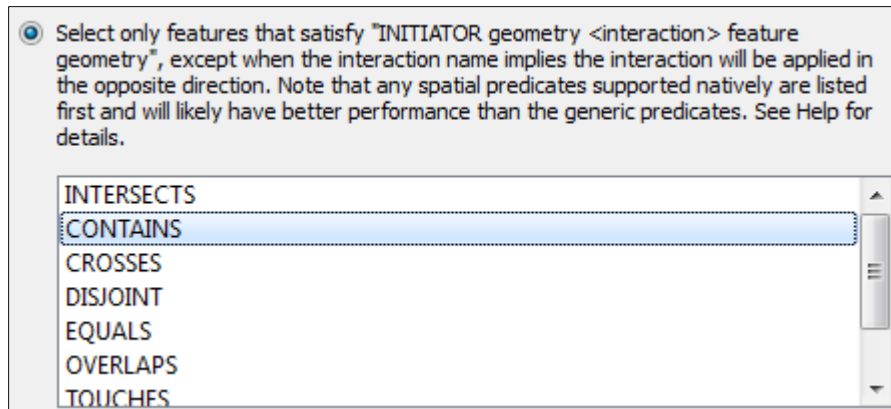
We are trying to locate all addresses related to the Initiator feature, so choose the table ADDRESS_POINTS



The Query Operation panel can be left as-is (we already selected ADDRESS_POINTS and we don't need a Where clause) so simply click **Next**.

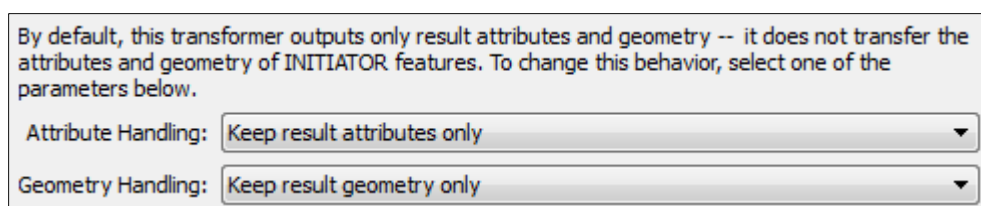
The next panel asks for spatial interaction. This is where we can define a spatial query between the Initiator (City Grid) and the table being queried (Address Points).

Select the third option and choose the CONTAINS operator. Click **Next**.



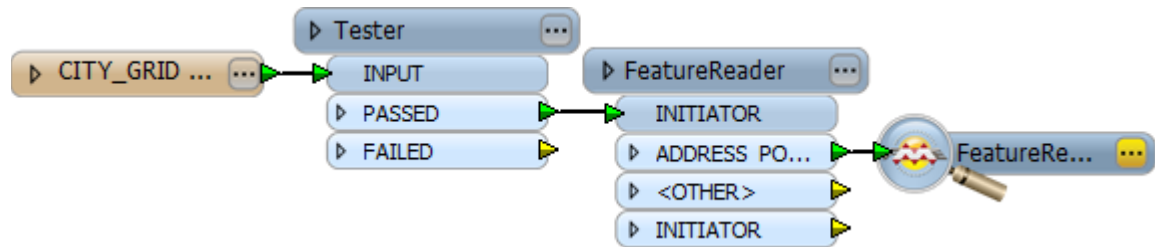
The final panel asks for merge options. It would let us do things like merge the address attributes onto the city grid geometry. But – in this example – we want to output only the address features.

So, leave the defaults (Keep Result Attributes Only and Keep Result Geometry Only) and click **Finish** to close the wizard.



8) Run Workspace

Connect an Inspector transformer to the ADDRESS_POINTS output port on the *FeatureReader*.



Run the workspace using File > Prompt and Run.

You will now be able to select a tile of addresses to be read and can inspect that output in the FME Data Inspector. Overlay the original grid data to prove the query worked correctly.

Advanced Task

Some addresses in the dataset have a STAT field whose value is INAC – indicating an address that is no longer valid. You can see this in the Data Inspector table view:

RESEARCH	STAT	COMMENT_	ACTION_	X_COORD
145 N	C			3130770
146	GEO	Geocoded poin...		3124989.25
147	GEO	Geocoded poin...		3124960
148	GEO	Geocoded poin...		3125078
149	GEO	Geocoded poin...		3127199.25
150	GEO	Geocoded poin...		3125719.5
151	INAC	Geocoded poin...		3124982.25
152	INAC	Geocoded poin...		3131431.5
153	INAC	Geocoded poin...		3130959.25
154	INAC	Geocoded poin...		3131049.25
155	INAC	Geocoded poin...		3131049.25

Use the FeatureReader WHERE clause to filter out addresses with this status. You will need to use the syntax <tablename>.<column> in this query.

Make a note of feature counts for your chosen grid square with and without the WHERE clause. This will help confirm the query is operating correctly.

Q) Why is this the most efficient way of querying the data?

In general, would the performance be better or worse using either a *Clipper* or *PointOnAreaOverlay* transformer? Why?

Metadata



FME allows users to both read and write Geodatabase metadata – and with its transformational capabilities, can make updates too!

What is Metadata?

A metadata record is a file of information - usually presented as an XML document - that captures the basic characteristics of a data or information resource.

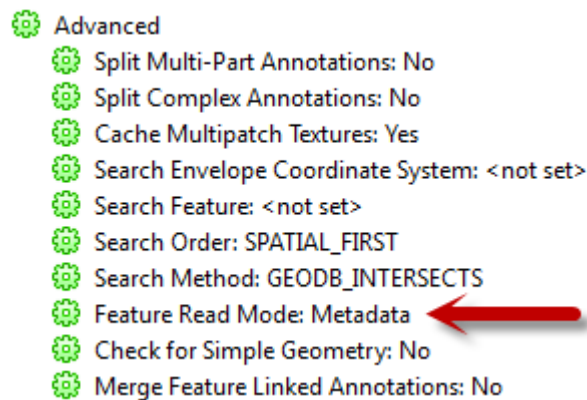
It represents the - who, what, when, where, why and how of the resource. Geospatial metadata are used to document geographic digital resources such as Geographic Information System (GIS) files, geospatial databases, and earth imagery.

A geospatial metadata record includes core library catalog elements such as Title, Abstract, and Publication Data; geographic elements such as Geographic Extent and Projection Information; and database elements such as Attribute Label Definitions and Attribute Domain Values.

FME supports reading and writing Geodatabase metadata.

Reading Geodatabase Metadata

Reading Geodatabase metadata is triggered by setting the Geodatabase reader advanced parameter 'Feature Read Mode' to 'Metadata'.



When set this way, this parameter causes the Reader to read one metadata record rather than all the normal feature class records. This means that to read features and metadata you need two Geodatabase Readers: one to get the features and one to get the metadata.

The metadata is stored as an XML string in the format attribute *geodb_metadata_string*. Other format attributes store information such as dimension, spatial column, geometry, etc.

Writing Geodatabase Metadata

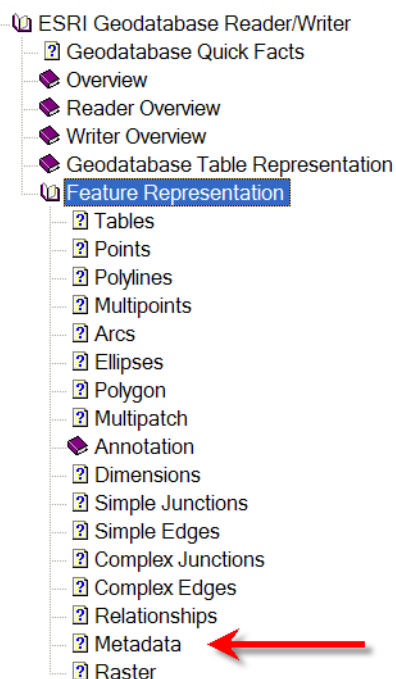
Writing metadata is triggered by a feature with the correct type being written to the table to which the metadata applies.

A metadata feature must have the type (i.e. have the `geodb_type` format attribute set to) `geodb_metadata`

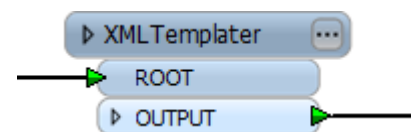
The metadata to be written should be held as XML in the format attribute `geodb_metadata_string`. It will overwrite any previous metadata that was on the table/feature class. If multiple metadata features are written to the same table, then the final feature is used.

Viewing newly created metadata within ArcMap will update certain fields, such as table name and record count, if they were incorrect on the XML passed in. However, reading data back with FME (if not already viewed in ArcMap) will not correct the data in the same way.

Note that the destination feature type (table) should be that of the geometry type (point, polyline, etc.). If you are handling metadata only then these fields may be set automatically. However, to be sure it is always safest to expose and set them explicitly. This also lets you view field values if you need to diagnose a problem.



The *XMLTemplater* and *XMLUpdater* transformers are good tools to use for creating and updating XML metadata.



To find out more information about Esri metadata in FME, browse the information in the Readers and Writers Manual under the section *Geodatabase > Feature Representation > Metadata*

Updating Geodatabase Metadata

Updating metadata is achieved by simply writing new metadata back to the table.

If read from a Geodatabase as an XML string, metadata strings can be updated through use of XML-related transformers such as the *XMLUpdater*

If read from another format of metadata, the *XSLTProcessor* might be a better transformer to use.



Example 9: Metadata	
Scenario	FME user; City of Interopolis, Planning Department
Data	Bike Routes
Overall Goal	Load data with metadata. Update metadata
Demonstrates	Creating, updating, and reading Geodatabase metadata
Starting Workspace	None
Finished Workspaces	C:\FMEData\Workspaces\PathwayManuals\Esr9-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Esr9-Complete-Advanced.fmw

The city has a Shape dataset of Bike Route. Metadata for SHAPE files is stored as a separate file with the .xml extension.

The task is to import the Bike Route data into a Geodatabase, at the same time writing metadata.

1) Inspect Data

Take a look at the source and destination datasets' existing metadata using ArcCatalog.

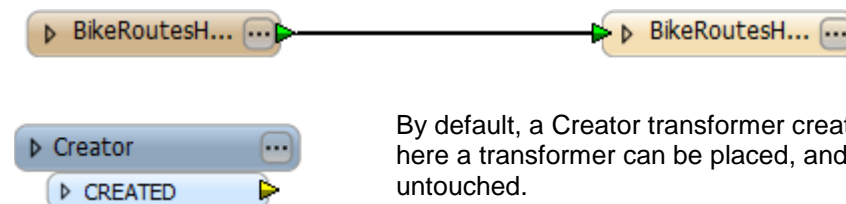
2) Start Workbench

Start FME Workbench and generate a workspace to load Bike Route information into a file Geodatabase

Reader Format	Esri Shape
Reader Dataset	C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesH.shp
Writer Format	Esri Geodatabase (File Geodatabase ArcObjects)
Writer Dataset	C:\FMEData\Resources\Esr\CityData.gdb

3) Add Creator

Place a Creator transformer to create one null feature. This feature will be used to trigger the metadata reading/writing process, and is a common method of creating parallel data flows.



By default, a Creator transformer creates a single null feature, so here a transformer can be placed, and the parameters left untouched.

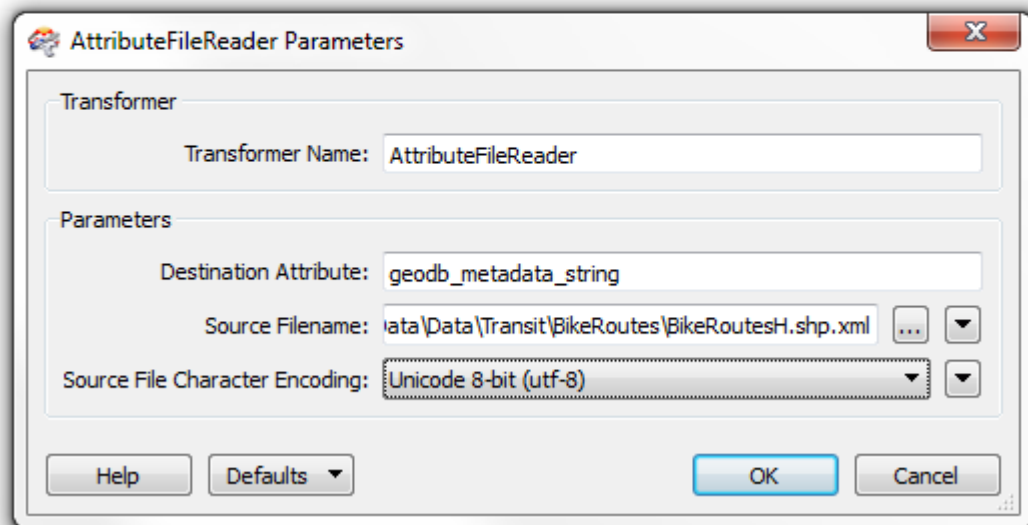
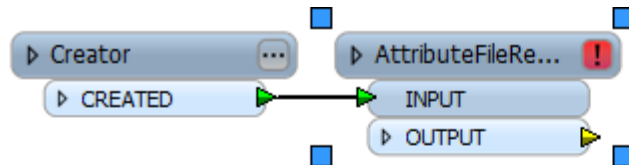
4) Add AttributeFileReader

Add an *AttributeFileReader* transformer after the *Creator*. An *AttributeFileReader* reads the contents of a file and stores it in an attribute.

Open the parameters dialog and set the source filename to be the XML Metadata document associated with the source shape dataset (*C:\FMEData\Data\Transit\BikeRoutes\BikeRoutesH.shp.xml*)

Set the destination attribute to be called *geodb_metadata_string*

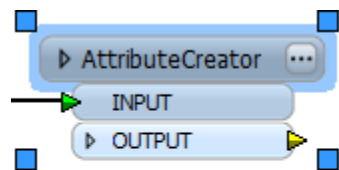
Set the target encoding to be utf-8



5) Add AttributeCreator

We must define this feature as being metadata, which is done with a format attribute.

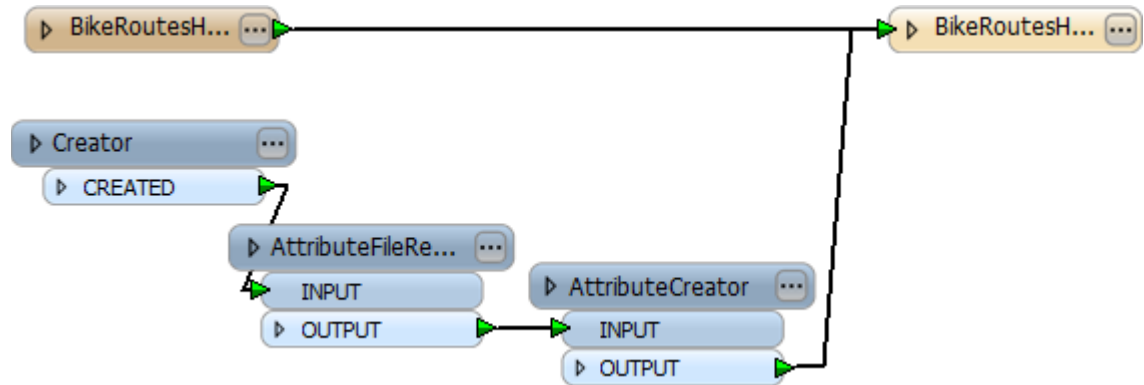
Add an *AttributeCreator* transformer after the *AttributeFileReader* and use it to create a format attribute called *geodb_type* with the value *geodb_metadata*



Attributes To Set	
Attribute Name	Value
geodb_type	geodb_metadata

6) Map Schema

Connect the *AttributeCreator* transformer to the writer feature type.
At this point the workspace will look something like this



7) Save and Run Workspace

Save the workspace and then run the workspace. Examine the results in ArcMap.

To view the metadata in ArcMap requires a change in setting.
Set Customize > ArcMap Options > Metadata > Metadata Style to ISO 19139 Metadata.

Now right-click on BikeRoutesH in the catalog window and choose the option Item Description.

This will open the metadata under ArcGIS Metadata:

ArcGIS Metadata ▶

Citation ▶

* **TITLE** BikeRoutesH

PRESENTATION FORMATS * digital map

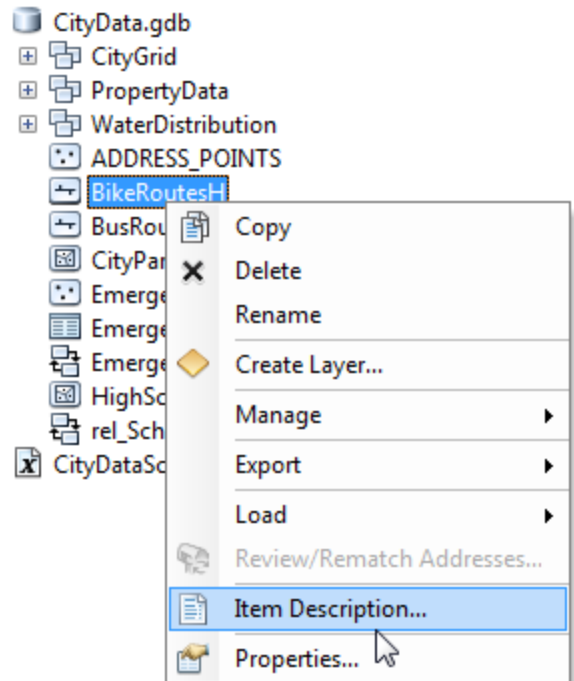
[Hide Citation ▲](#)

Resource Details ▶

DATASET LANGUAGES * English

SPATIAL REPRESENTATION TYPE * vector

* **PROCESSING ENVIRONMENT** Microsoft
Windows XP Version 5.1 (Build 2600)
Service Pack 3; ESRI ArcCatalog
9.3.1.3000



Advanced Task

The method used in this example is straightforward and with no updates to the metadata.

When updates are required, the *XMLUpdater* transformer is a good solution. Underneath this transformer uses XQuery to do its work. It's said that XQuery is to XML as SQL is for databases.

So, with this transformer, the example can be extended to update the metadata and add the keywords "Austin City Data – Bike Routes".

8) Inspect XML

Open up the XML file in a text editor, web browser, or similar.

Notice the tag `<themekey>` :

```
<themekey>Austin City Data</themekey>
```

It is this that we wish to update to say Austin City Data – Bike Routes.

9) Set Parameter

Because a second run of the workspace will rewrite the same data, it's necessary first to change the writer feature type parameter *Drop Table First* to *Yes*, so do that.

10) Add XMLUpdater Transformer

Add an *XMLUpdater* transformer between the *AttributeCreator* and the writer feature type. The input port to connect is *DOCUMENT* as the incoming feature contains the xml document.

Open the parameters dialog. Set:

XML Input	Text or Attribute
XML Text	Set to Attribute Value > geodb_metadata_string
Result Attribute	geodb_metadata_string

Now for the update part. Set:

Update Type	Replace Contents
XML Path	/metadata/idinfo/keywords/theme/themekey
Value Type	Plain Text
Value	Austin City Data – Bike Routes

11) Add Creator

The *XMLUpdater* requires an update feature – even a null one – to carry out an update.

So, add a *Creator* transformer and connect it to the *XMLUpdater:UPDATE* port.

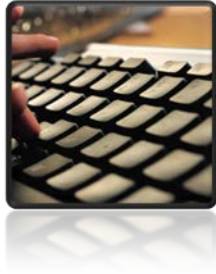
12) Save and Run Workspace

Close ArcMap to ensure the data is not locked. Save the workspace and then run it.

Re-open ArcMap and examine the results to view the changed metadata.



Integrating FME and ArcGIS



FME and ArcGIS share a high level of integration that allows scripts from one to be run in the other.

Spatial ETL Tools

Spatial ETL Tools are what FME workspaces are known as in the ArcGIS environment.

A Spatial ETL tool can be created, edited, and run using FME functionality integrated into ArcGIS by the FME installation process. This includes an equivalent to FME Workbench.



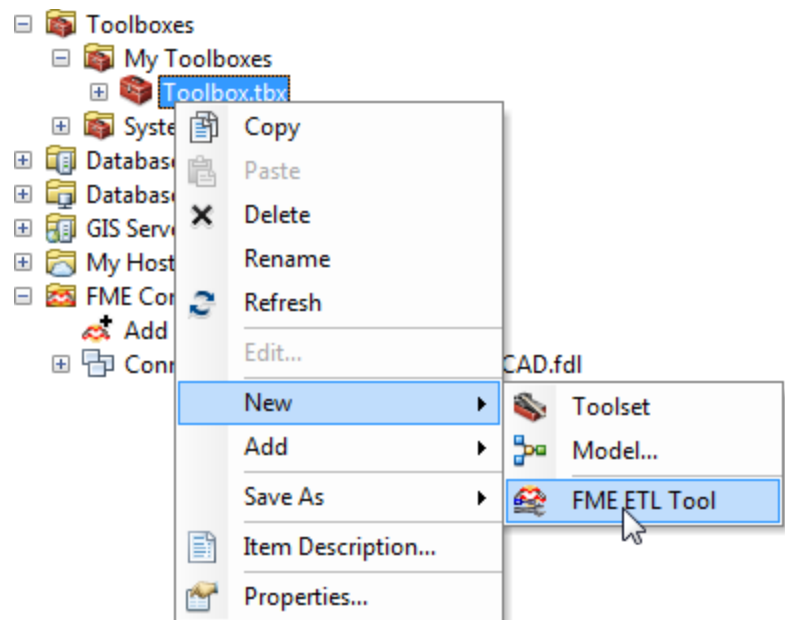
Example 10: Integration	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks
Overall Goal	Convert data to Shape and create spatial index
Demonstrates	FME/ArcGIS integration

The city has a dataset of city parks in MapInfo TAB format. It wishes to convert it to Esri Shape format and create a spatial index.

This can be done with a combination of ETL tool and ModelBuilder.

1) Start ArcMap

Start ArcMap. Right-click on Toolbox in a Catalog window, and choose New > FME ETL Tool



2) Create Workspace

Create a new “workspace” using the Workspace Wizard.

Source Format	MapInfo TAB (MITAB)
Source Dataset	C:\FMEData\Data\Parks\city_parks.tab
Destination Format	Esri Shape
Destination Parameters	None
Workflow Options	Static Schema

Click **Finish** and the ETL tool is created and opened for editing.

3) Update Workspace

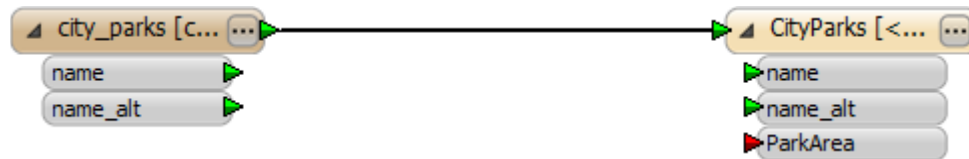
FME will automatically insert a GeometryFilter transformer to divide the data up into different Shape files for each geometry type. However, since the source data is composed entirely of polygons, this is not necessary.

Delete the GeometryFilter and all of the writer feature types except for city_parks_polygon

Rename the feature type city_parks_polygon to CityParks

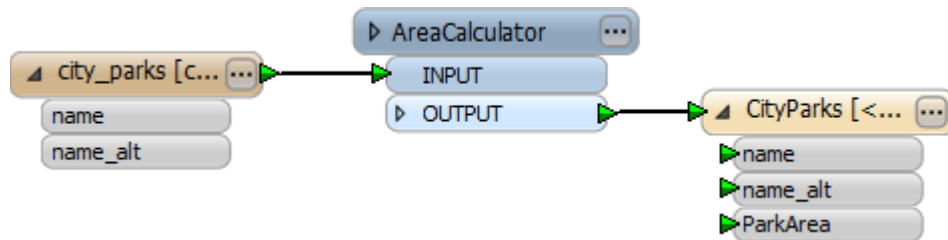
Add a new attribute called ParkArea, of type "number" (16,2).

The ETL tool will now look something like this:



4) Add Transformer

Insert an AreaCalculator transformer to calculate the area of each park. Ensure the output attribute is called ParkArea in order to match the output schema.

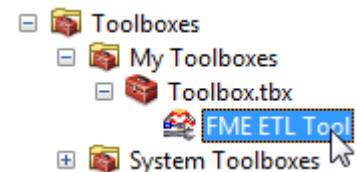


5) Save and Run ETL Tool

Save the ETL tool and exit Workbench.

Back in ArcMap, double-click the newly created ETL tool to run it.

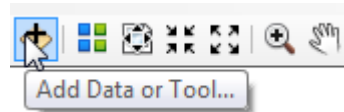
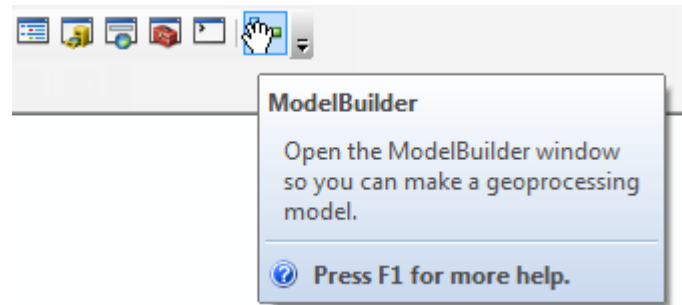
Select an output folder when prompted and click OK to run the tool.



Once complete, check the FME log in the Results window. Then examine the output files. Notice there are no files (.sbn, .sbx) for a spatial index. Although there is a setting in FME to create this, we will try and create these using ModelBuilder.

6) Start ModelBuilder

Click the button on the ArcMap toolbar to open a ModelBuilder window.



Now click on the ModelBuilder button to add data or a tool

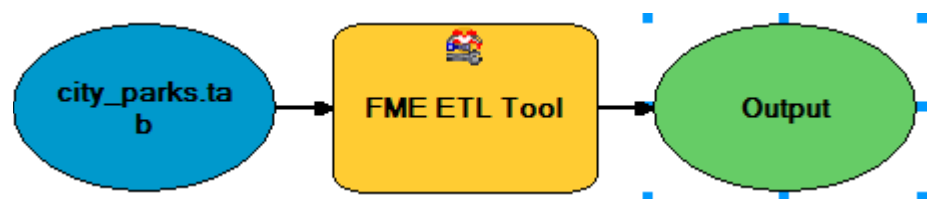
Double-click Toolbox.tbx (look under Toolboxes > My Toolboxes) to open it.

Select the newly created FME/Spatial ETL tool with a single click, and click **Add**.
Do not double-click the tool - else it will run again!



7) Set Output

Double-click *Destination Esri Shape Directory* and set an output folder. Click **OK**.
The objects in ModelBuilder should all change color to show they are ready to run.



8) Run Model

Press the blue play button to run the model.



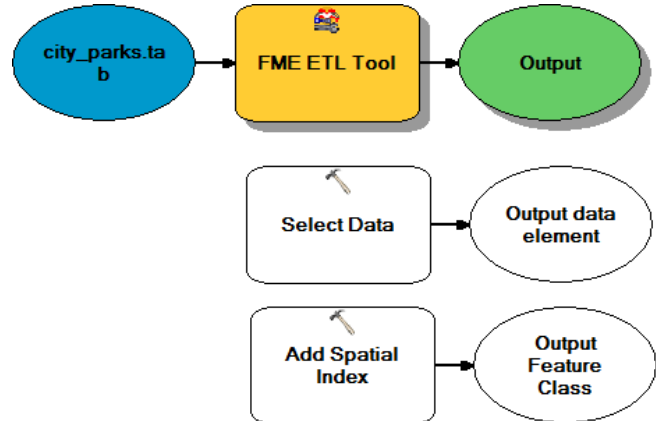
The model will run and the initial shape output is created.
Check the files to prove there are no files for the spatial index.

9) Select Data

Again, choose Add Data or Tool from the menubar. This time select Toolboxes > System Toolboxes > Data Management Tools > General > Select Data

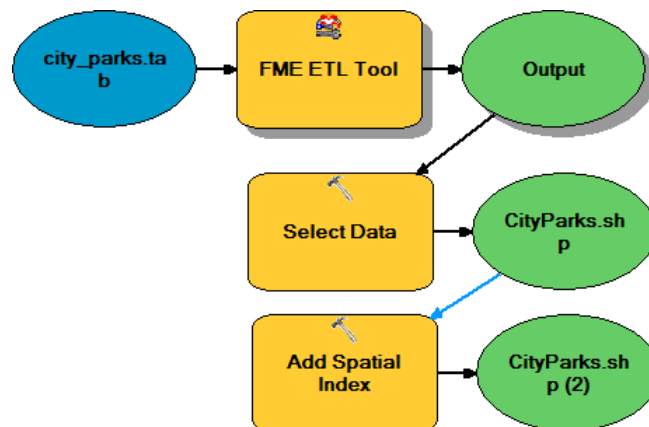
10) Create Index

Again, choose Add Data or Tool from the menubar. This time select Toolboxes > System Toolboxes > Data Management Tools > Indexes > Add Spatial Index



11) Make Connections

Now select Connect from the toolbar.

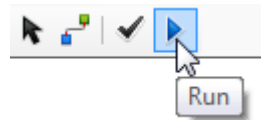


Draw connections from the output folder to *Select Data* (choose *Input Data Element* in the popup dialog) and from *Output Data Element* to *Add Spatial Index* (choosing *Input Features* in the popup dialog).

The new objects in ModelBuilder should all change color to show they are ready to run.

12) Run Model

Again, press the blue play button to run the model.

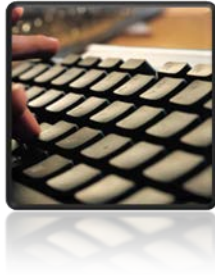


The model will run and the output is created. Check the files to prove there are now files for the spatial index.



A similar outcome could have occurred by creating the workspace directly in FME Desktop, writing an ArcGIS Python script to form the spatial index, then adding the Python as an FME shutdown script in Workbench.

Session Review



This session was all about Esri data and FME.

What You Should Have Learned from this Session

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

Theory

- FME can read and write **Annotation**, including Feature-Linked Annotation.
- Reading and Writing **Geometric Networks** is much faster when the network functionality is turned off
- **Relationship Classes** are handled by a primary/foreign key relationship that must be defined in ArcGIS before FME can write such data
- FME can read and write **Metadata** and update it using the **XMLUpdater** transformer
- FME and ArcGIS have a close **integration** where workspaces (ETL models) can be created and run in ArcGIS, and Esri Python scripts can be run in FME

FME Skills

- The ability to read and write ArcGIS annotation
- The ability to read and write Relationship classes
- The ability to read, update, and write ArcGIS metadata
- The ability to read non-Esri format datasets in ArcGIS, do a Data Interoperability Quick Import or Quick Export, create a Spatial ETL model in ArcGIS, and run a Python script in FME

Appendix A – Useful debugging resources

When a workspace uses an ArcObjects-based Geodatabase reader or writer, then all reported error messages usually consist of an ArcObjects error number and an error message.

An Esri webpage that is useful for interpreting the error numbers can be found via:
<http://fme.ly/ArcObjects>

Other information – including troubleshooting – can be found on our community knowledgebase (FMEpedia) via: <http://fme.ly/Geodatabase>

The FME Readers and Writers manual also has lots of information on feature representation in a Geodatabase and the definitions for all the format attributes.

Appendix B – FME, ArcGIS, and Python

FME has a built-in installation of Python. Because ArcGIS scripts are Python-based, FME can run one of these scripts (provided ArcGIS is installed and licensed on the same system).

Python Versions

If the version of Python used by ArcGIS is different to that used by FME, then you need to properly set up the Python environment in order to get FME to run a script containing ArcGIS Python components.

Select Tools > FME Options > Runtime from the Workbench menubar.
Check the box for Custom Python Interpreter and choose a Python DLL of the correct version.



There are various locations the DLL might be found, depending on the operating system used. The ArcGIS 10 Desktop installer puts its Python DLL into the windows folder by default.

On our systems we also have to set PYTHONPATH to:

```
C:\Program Files\ArcGIS\Desktop10.0\bin;C:\Program  
Files\ArcGIS\Desktop10.0\arcpy;C:\Program  
Files\ArcGIS\Desktop10.0\ArcToolbox\Scripts
```

...in order for FME to locate the arcpy libraries.

For more information on how to select a different Python interpreter for FME, see:
<http://fme.ly/2014>

Appendix C – Performance Considerations

There are a number of factors and techniques you can take into consideration to improve performance when reading and writing to Geodatabase with FME.

For full and up-to-date information see the article at <http://fme.ly/1968>

Appendix D – Connecting to an Enterprise Geodatabase

The exercises in this document all use file-based Geodatabase. However, it is worth being aware of how to connect to ArcSDE Geodatabase.

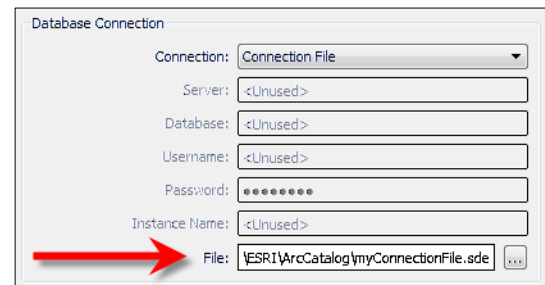
Connecting to an Esri Enterprise Geodatabase can be done either through a set of parameters, OS Authentication, or a Connection File. The Connection File is probably the simplest method to use.

Connection File

Connection files are a store of connection information. They are created by ArcGIS when a connection is made in one of the ArcGIS applications, and can be used by FME to connect.

The files are usually stored in the Esri ArcCatalog user application directory; for example:
C:\Users\<username>\Application Data\Esri\ArcCatalog\ConnectionFile.sde

Simply select the Connection File method and then use the file browser provided to select the connection file to be used.



Connection Parameters

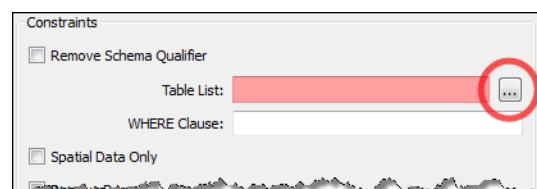
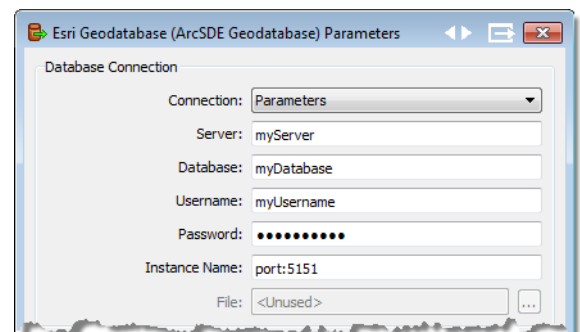
When a reader dialog requires the definition of an ArcSDE connection, firstly fill in the format field as follows:

Reader Format: Esri Geodatabase (ArcSDE)

Click the Parameters... button to open the parameters dialog.

Ensure 'parameters' is selected in the connection type drop down list.

Fill in the server, database, username, and password parameters, plus the instance name.



Optionally click on the browse button to the right of the Table List parameter to select specific tables

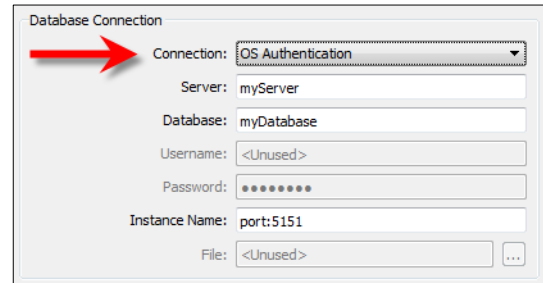
If the connection was successful, and there is already data in the database, then a list of tables will be presented. If the connection was unsuccessful, then a dialog will appear with an error reporting the nature of the problem:

```
ERROR | Could not open the Enterprise Geodatabase. Please check that the connection
       | parameters specified are correct. The error number from ArcObjects is: '-
       | 2147216118'. The error message from ArcObjects is: {Bad login user}
WARN  | A fatal error has occurred. Check the logfile above for details
INFORM | Merged 0 schema features read from 1 datasets into 0 resulting feature types
```

OS Authentication

Connecting to an Esri Geodatabase using Operating System authentication is a similar process to connecting through parameters.

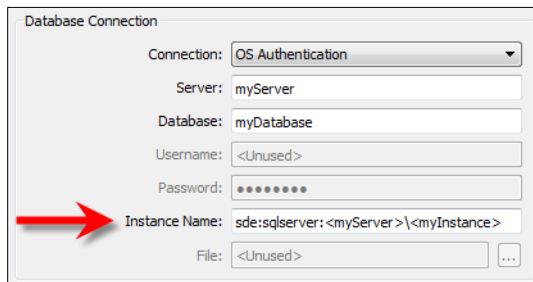
The main difference is that when the OS Authentication option is selected, the Username and Password parameters are grayed-out.



Username and password will get supplied to the SQL Server database from the user's OS login information.

OS Authentication (Direct Connect)

The OS Authentication mode is also used when carrying out a Direct Connect (aka 2 Tier Connection). This method is the only way by which to connect to a Personal SDE Geodatabase.



Here the Instance Name parameter is all-important. It is where the instance and connection information is defined, and takes priority over the other Server field.