FME Desktop ®

Esri ArcGIS v10.3
Pathway Training

FME 2015.1.1 Edition

SAFE SOFTWARE

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

Introduction

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

This training material 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 FME2015.1.1. 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 v10.3, 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/fmedata

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.

A new option in FME 2015 allows a user to pick the version of geodatabase they wish to write to when using a File Geodatabase writer (does not include the File Geodatabase API).

For more information on such differences 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 10.3	**FME 2014-SP5 or higher**
ArcGIS 10.2	FME 2013-SP3 or higher
ArcGIS 10.1	FME 2012-SP3 or higher
ArcGIS 10.0	FME 2010-SP2 or higher

For more information see: **http://fme.ly/1957**

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

Feature Datasets

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

Feature datasets are a way through which different feature classes can be grouped together. For example, all feature classes 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 feature class can be set in the feature type properties dialog, under the Format Parameters tab:

Shape Alias	SHAPE
Configuration Keyword	DEFAULTS
Feature Dataset	MyFeatureDataset
Grid 1	
Avg Num of Points	

Feature datasets can also be handled using an FME format attribute.

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 feature class to be written to two separate feature datasets, as this would require two feature classes, and two feature classes of the same name cannot exist in a Geodatabase.

Feature datasets can also be handled by the File Geodatabase API. For feature classes within a feature dataset, the FME feature type name is formed as follows:
<Feature Dataset name>/<Feature Class name>.

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.

Exercise 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:\FMEData2015\Workspaces\Esri\LoadVancouverCity.fmw
Finished Workspace	N/A

To get started, this is a basic example of creating a Geodatabase. Rather than create the Geodatabase from scratch, we'll use an ArcGIS XML Workspace Document. Not to be confused with an FME workspace, these are templates created by exporting from an existing Geodatabase.

1) Start ArcMap
First let's take a look at the ArcGIS XML Workspace Document. Start ArcMap. Open a catalog window (be sure to pin it down so it doesn't auto-hide).

Select the Connect to Folder option:
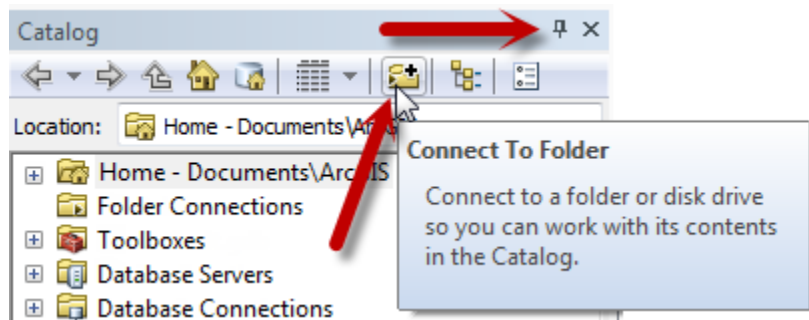
Select C:\FMEData2015 as the folder to connect to. This enables you to browse to the course data at any time.
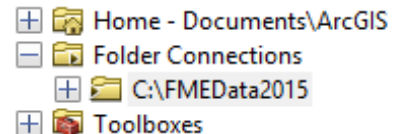
Browse to C:\FMEData2015\Resources\Esri

Double-click on the entry VANCOUVERCITYTEMPLATE.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 FME Workbench.

Open the workspace: C:\FMEData2015\Workspaces\Esri\LoadVancouverCity.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:

  VancouverCity [GEODATABASE_FILE]
      Destination Esri File Geodatabase: C:\FMEData2015\Resources\Esri\VancouverCity.gdb
      Coordinate System: <not set>
      Parameters
          Overwrite Existing Geodatabase: Yes
          Transaction Type: Transactions

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

  Transaction Type: Transactions
  Template File: C:\FMEData2015\Resources\Esri\VANCOUVERCITYTEMPLATE.XML
  Simplify Geometry: No

- Six Writer feature types (feature classes) whose names match ones in the Geodatabase workspace document:

- The AddressPoints feature type has a Geodatabase feature dataset set:

Configuration Keyword:	DEFAULTS
Feature Dataset:	Properties
Grid 1:	

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

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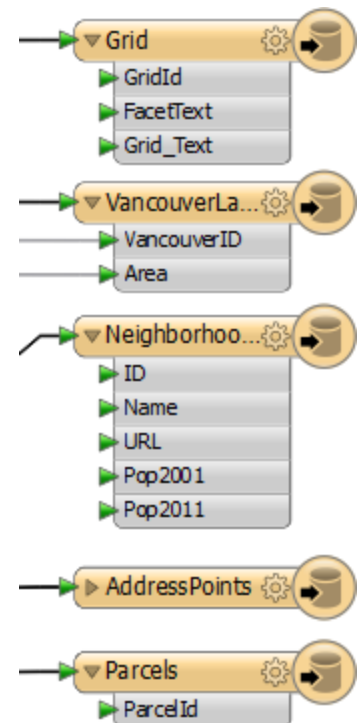
3) Run Workspace
Run the translation. It would be better to close ArcMap at this point to ensure nothing you are writing to is locked by ArcGIS.

The translation should succeed in about 40 seconds or so:

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
Re-open ArcMap. Browse to C:\FMEData2015\Resources\Esri\VancouverCity.gdb

Add data from the Grid feature class 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 ArcMap automatically colors each subtype differently.
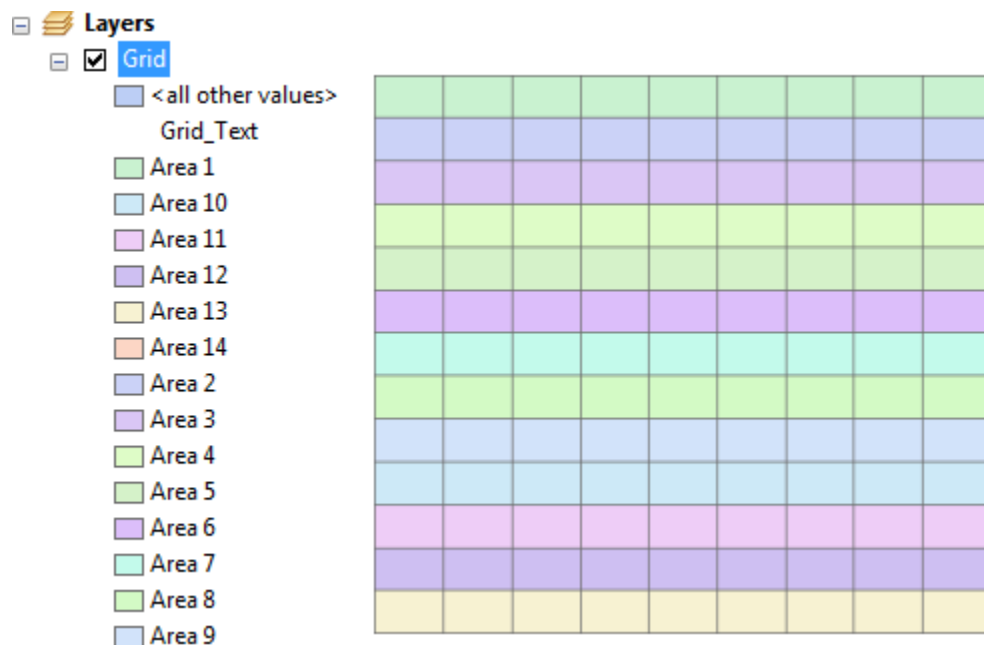
Esri Solutions

FME is the natural choice for dealing with Esri-related datasets; but it's not Safe Software's only solution!

There are two key products from Safe Software for dealing with Esri datasets:

- The FME Esri Edition
- The ArcGIS Data Interoperability Extension

The FME Esri Edition
The FME Esri edition is a version of FME with particular support for Esri data formats. It supports reading and writing from/to:

File Geodatabase
Personal Geodatabase
Enterprise/ArcSDE Geodatabase

…plus a whole other series of Esri formats. See www.safe.com for a full list of supported formats.

Besides being able to translate these formats of data – and transform their contents – the FME Esri Edition also builds extra functionality into ArcGIS (should it be installed on the same system).

This functionality includes the ability to read and write non-native formats of data directly inside of ArcGIS. It allows users to take advantage of FME functionality without leaving the ArcGIS environment.

The FME Extension for ArcGIS needs to be specifically activated using a utility called the FME Integration Console. This tool is how you can extend not 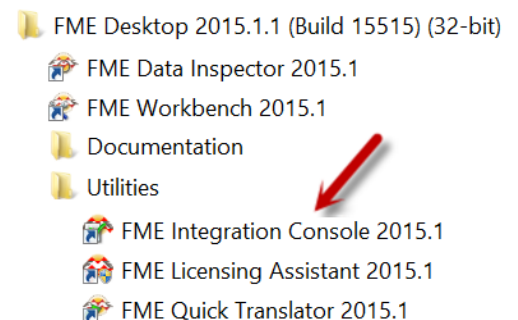just ArcGIS, but other FME-compatible applications too. Please note that depending on your system's configuration, you may have to run the Integration Console with administrative privileges (i.e. right click on the FME Integration Console and select "Run as administrator").

The FME Integration Console is found on the Windows start menu under a Utilities folder *(Depending on your operating system, the navigation process to the Utilities folder may differ)*:

📁 FME Desktop 2015.1.1 (Build 15515) (32-bit)
　🔲 FME Data Inspector 2015.1
　🔲 FME Workbench 2015.1
　📁 Documentation
　📁 Utilities
　　🔲 FME Integration Console 2015.1
　　🔲 FME Licensing Assistant 2015.1
　　🔲 FME Quick Translator 2015.1

It's then necessary to restart ArcGIS and turn on the extension there using Customize > Extensions on the menubar:

Customize	Windows	Help
Toolbars ▶		
Extensions...		
Add-In Manager...		
Customize Mode...		
Style Manager...		
ArcMap Options...		

Esri Solutions
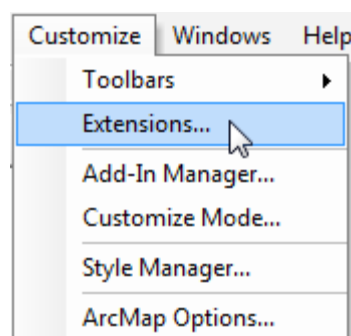
The ArcGIS Data Interoperability Extension

The ArcGIS Data Interoperability Extension is an entirely separate product to FME. It is similar to the FME Extension for ArcGIS in its functionality, but is sold and supported by Esri, rather than Safe Software.

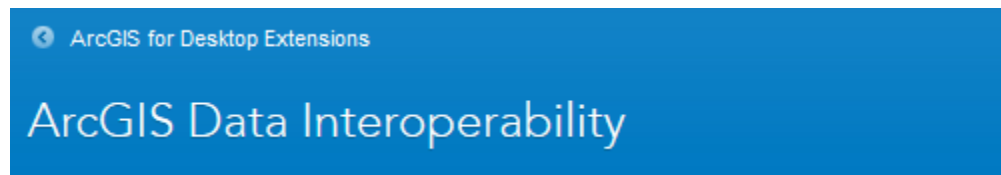

The extension provides FME-like functionality inside of the ArcGIS environment, but without the standalone FME product that is used throughout this manual.

Extension Functionality

Both the FME Extension for ArcGIS and the ArcGIS Data Interoperability Extension, have a common set of functionality. These functions are:

- **Direct Read**
  This is where the format reach of ArcGIS is extended, allowing it to view and use non-native formats of data, such as MapInfo TAB.

- **Quick Import/Export**
  These are tools added for the express purpose of translating other spatial data formats to or from a Geodatabase. It is a "Quick Translation" with no user-defined data transformation.
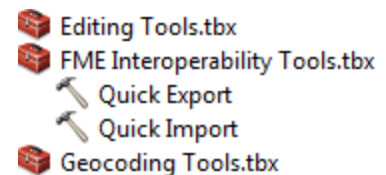
- **Spatial ETL Tools**
  Spatial ETL Tools are the ArcGIS equivalent of FME workspaces. A Spatial ETL tool can be created, edited, and run using FME functionality integrated into ArcGIS, and allows both format translation and data transformation.

The differences between the FME Extension and the ArcGIS Data Interoperability Extension are few. Apart from minor terminology differences (Spatial ETL Tools are called FME ETL Tools with the FME Extension) the main difference will be in the formats available (134 with the Interoperability Extension vs nearly 300 with the full FME).

Again, check www.safe.com for a full list of formats available in any edition of FME or the Data Interoperability Extension.

For an example of creating/using a Spatial ETL tool in ArcGIS, see Appendix A of this document

Exercise 2: Extending ArcGIS Functionality	
Scenario	FME user; City of Interopolis, Planning Department
Data	Transportation Data
Overall Goal	Compare FME Esri Edition and the Data Interoperability Extension
Demonstrates	Quick Translation/Import of Data
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise2-Complete.fmw

This is a basic example of translating data to an existing Geodatabase, either through FME or the ArcGIS Data Interoperability Extension

This exercise shows some of the capabilities of FME as it extends ArcGIS functionality, with or without the ArcGIS Data Interoperability Extension, and then expands upon this capability with the full FME Esri Edition.

The scenario is creating feature classes for a road network, which weren't in the XML workspace from Exercise 1. Roads can be classed as either Major, Minor, or Other, and we will create a feature class for each type.

1) Activate the FME Extensions in ArcGIS
To use FME inside of ArcGIS (for example to inspect our source data) we need to activate the extensions. 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
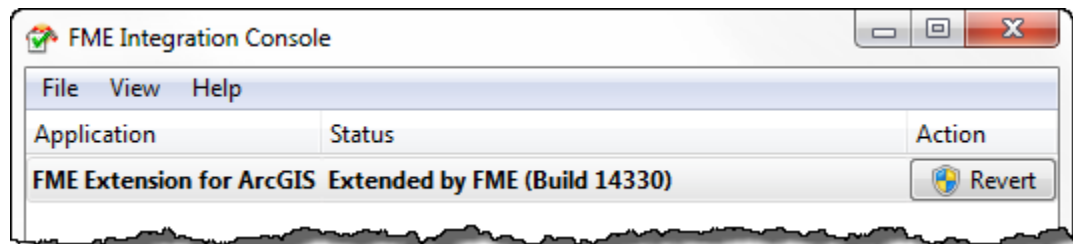
- In ArcMap (you may need to restart it), visit Customize > Extensions
- Ensure the FME Extension (or Data Interoperability Extension) is activated here too
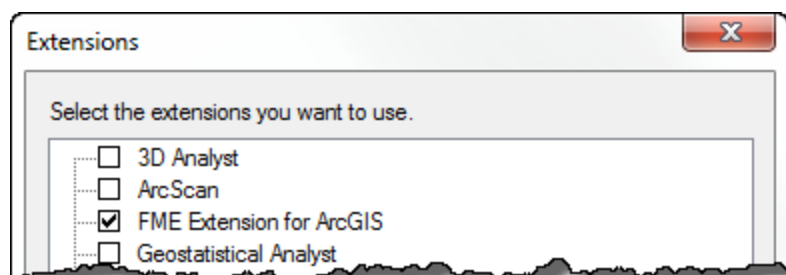
2) Inspect Source Data

With FME extending ArcMap, you will be able to view any FME-supported format of data inside ArcMap; including MapInfo TAB.

For example, browse to the sample parks dataset (C:\FMEData2015\Data\Parks\Parks.tab) using the Catalog window in ArcMap. You should be able to add that data to the ArcMap main display:

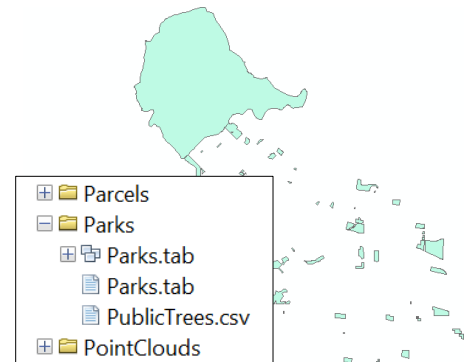You will also be able to inspect the source data for this translation, which is an AutoCAD Drawing dataset C:\FMEData2015\Data\Transportation\Roads.dwg

However, because ArcGIS has its own support for DWG datasets, FME is not being used when you view this data. If you do want to use FME then you have to use an FME Connection.

3) Add FME/Interoperability Connection

At the foot of the Catalog window's browse tree will be either the option "FME Connections" (for an FME-extended ArcGIS) or "Interoperability Connections" (for the Data Interoperability Extension).

Expand that entry and double-click the option to "Add […] Connection"
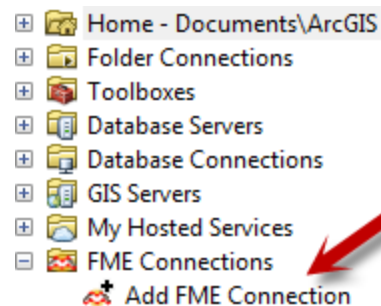
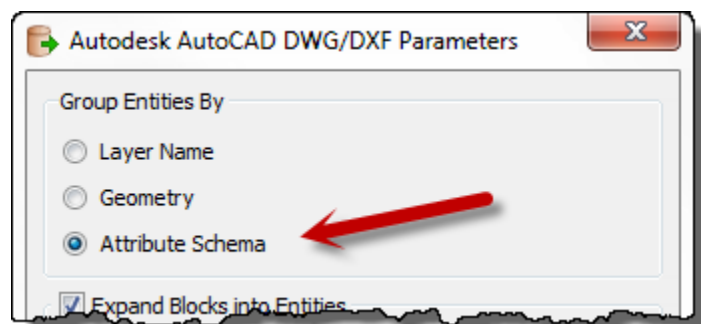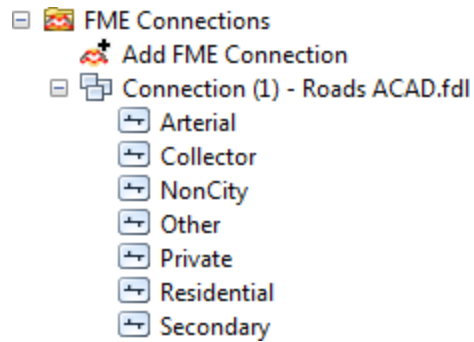When prompted – by a dialog identical to Add Reader in FME – set the information as follows:

Reader Format   Autodesk AutoCAD DWG/DXF
Reader Dataset   *C:\FMEData2015\Data\Transportation\Roads.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 have an FME Connection and can use it to view the Roads data arranged by layer/schema, rather than geometry.

4) Import Data
Another capability FME adds to ArcGIS is the import of supported formats directly into a Geodatabase. Let's try that out.

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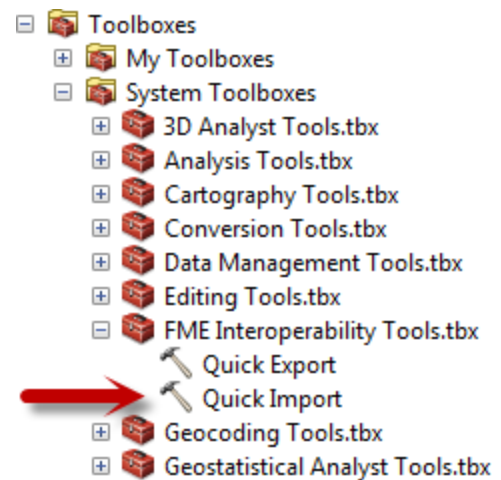
In the Quick Import dialog, click the Input Dataset browse button and set the Input Dataset as follows:

Reader Format   Autodesk AutoCAD DWG/DXF
Reader Dataset   *C:\FMEData2015\Data\Transportation\Roads.dwg*

Be sure to again click the Parameters button, and check that the "Group Entities By" parameter is set to *"Attribute Schema"*.
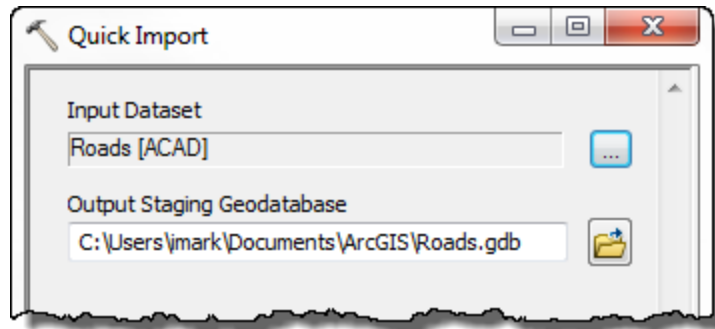
Set an output Staging Geodatabase, or use the default value:

Click OK and the translation will run – in the background – as shown by the status bar "throbber":

Note – you may have to turn off Background Geoprocessing in ArcMap if the Import fails.

Once complete a pop-up dialog will appear, like so:
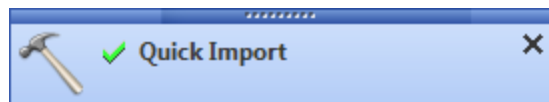
Click on this and an FME log file will appear in a Results window:

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
                Features Written
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Arterial                              957
Collector                               8
NonCity                                26
Other                                  43
Private                               191
Residential                          2507
Secondary                             263
==================================================================
Total Features Written               3995
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Translation successfully completed
```

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

☐ 🖿 Folder Connections
   ☐ 🖿 C:\FMEData2015
      ⊞ 🗀 Data
      ☐ 🗀 Output
         ⊞ 🗀 Training
         ⊞ 🗀 Tutorial
         ☐ 🗊 Roads.gdb
            ⊡ Arterial
            ⊡ Collector
            ⊡ NonCity
            ⊡ Other
            ⊡ Private
            ⊡ Residential
            ⊡ Secondary
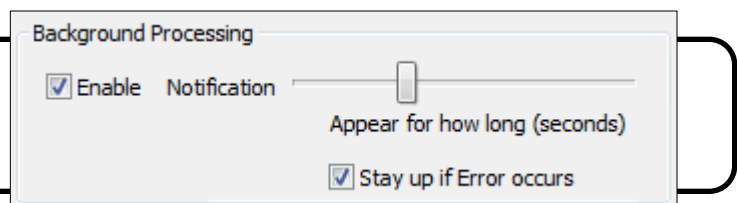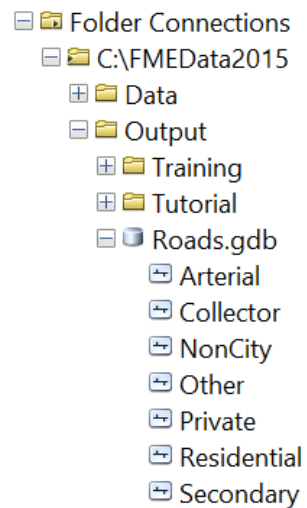
This was just a *Quick Import* using FME/Data Interoperability functionality; i.e. in the same way as an FME Quick Translation, there is no transformational stage.

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.

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

Reader Format   Autodesk AutoCAD DWG/DXF
Reader Dataset   *C:\FMEData2015\Data\Transportation\Roads.dwg*

Of course, it's not vital to do this, as we were just looking at the data in ArcMap, but it's still a good habit to get into.

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

Reader Format   Autodesk AutoCAD DWG/DXF
Reader Dataset   *C:\FMEData2015\Data\Transportation\Roads.dwg*

Writer Format   Esri Geodatabase (File Geodb ArcObjects)
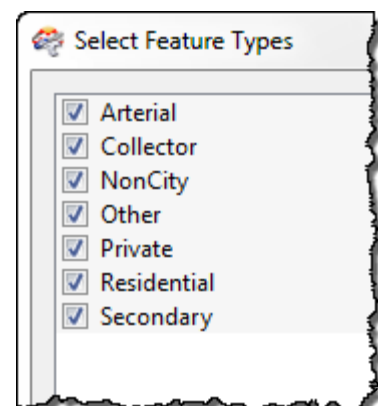**Writer Dataset**   *C:\FMEData2015\Output\Roads.gdb*

Use the following Reader parameters:

Note - both Use Block Feature Type for Component and Use Block Layer Information for Components can be found by expanding the Explode Blocks Into Entities area of the Parameters dialog while Resolve Entity Color is found by expanding the Advanced section of the dialog.

Group Entities By:	Attribute Schema
Use Block Feature Type for Components:	Yes
Use Block Layer Information for Components:	Yes
Read Visible Attributes as Text	Yes
Explode MText Entities:	Yes
Resolve Entity Color:	Yes

Select Feature Types
☑ Arterial
☑ Collector
☑ NonCity
☑ Other
☑ Private
☑ Residential
☑ Secondary

Click **OK** to accept this dialog. When prompted select all Feature Types to be processed.

7) Update Writer Feature Types
The benefit of FME Workbench over a Quick Import process is the ability to make edits to the output schema, and manipulate the data as it is being translated. Let's take advantage of this capability by collating the source feature types into just three feature classes.
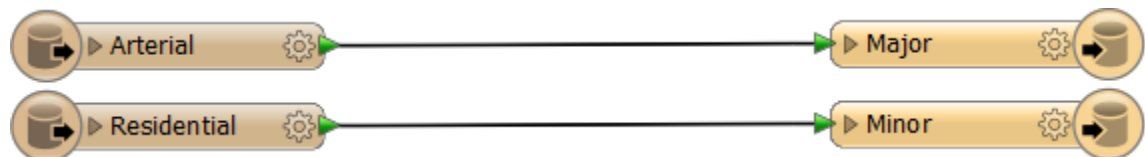
Firstly let's delete the Writer feature types we don't need. Delete the following by clicking on them and pressing the delete key:

- Private
- Secondary
- Collector
- NonCity
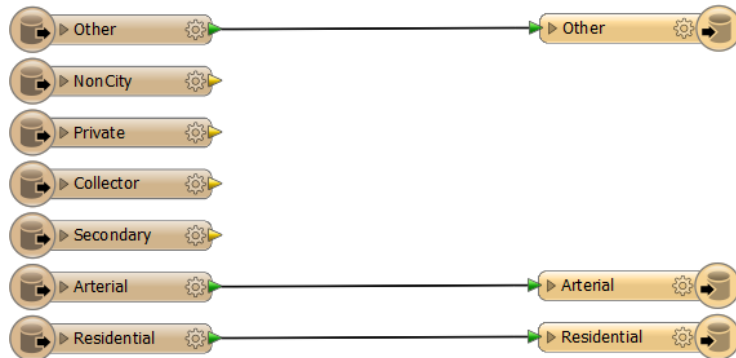
Now rename two of the existing feature types as follows:

Arterial   to   Major
Residential   to   Minor

8) Map Feature Types
Now let's map the Reader feature types to the Writer feature types.

Firstly delete any existing connections from Reader to Writer feature types.

Now connect the types as follows:

Other:	Other
NonCity:	Other
Collector:	Major
Secondary:	Major
Arterial:	Major
Residential:	Minor
Private:	Minor

Feel free to reposition the feature types to avoid overlapping connections.
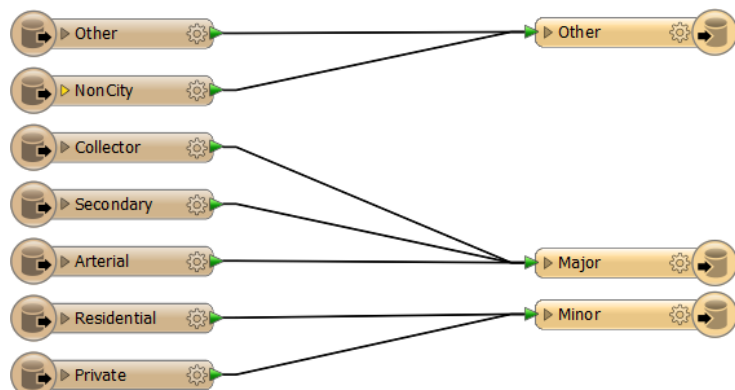
9) Join Linework

One way to transform this data is to join the linework together where the road name matches. For example, where "Point Grey Road" changes from Arterial to Secondary, we should just connect it together, as both types are now being classified as "Major" in our workspace.
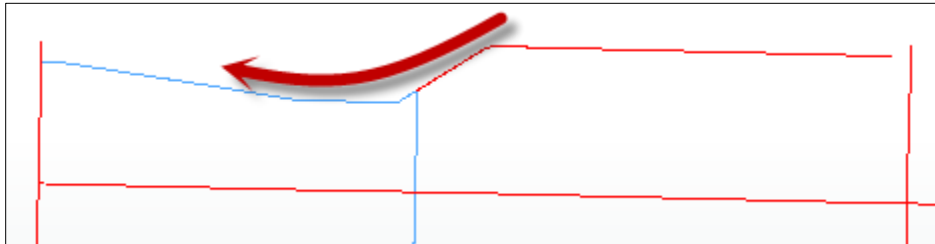
To achieve this, add a LineJoiner transformer before each Writer feature type/class, like so:
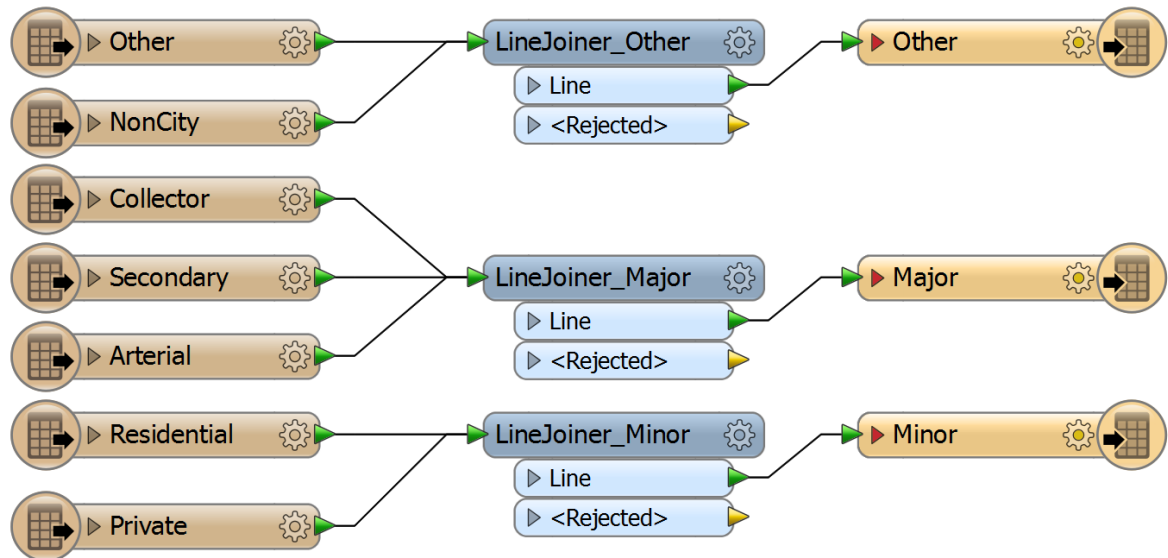
Hint: You can use Quick Add here by clicking on the input port arrow for each underline writer underline feature type, and then typing LineJoiner. This will make the required connections automatically.

For each LineJoiner transformer, open the parameters dialog and set the group-by parameter to group-by street name:
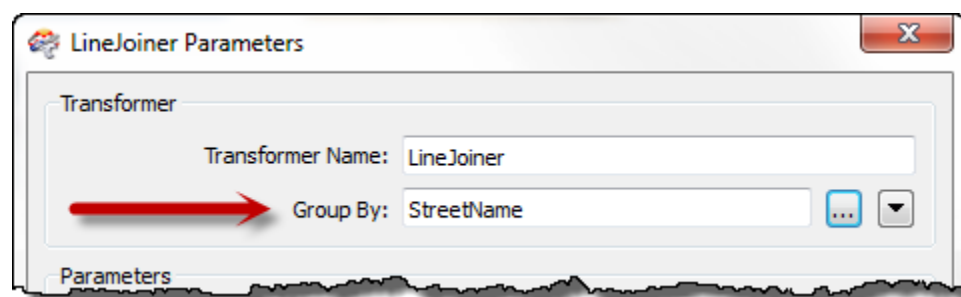
10) Save and Run Workspace

As a final step, ensure the Writer parameter "Overwrite Existing Geodatabase" is set to Yes; as this is a new Geodatabase we'll want to recreate it if we re-run the workspace.

Roads [GEODATABASE_FILE]
   Destination Esri File Geodatabase: C:\FMEData2015\Output\Roads.gdb
   Coordinate System: <not set>
   Parameters
      Overwrite Existing Geodatabase: Yes
      Transaction Type: Transactions
      Template File: <not set>

Folder Connections
   C:\FMEData2015
      Data
      Output
         Training
         Tutorial
         Roads.gdb
            Major
            Minor
            Other

Now save the workspace and then run it. Again, inspect the output in ArcMap. There should be a new Geodatabase in the Catalog window with three feature classes:

Examine the data closely and you'll see that the line features have been connected together where they had the same road name:

Esri Solutions
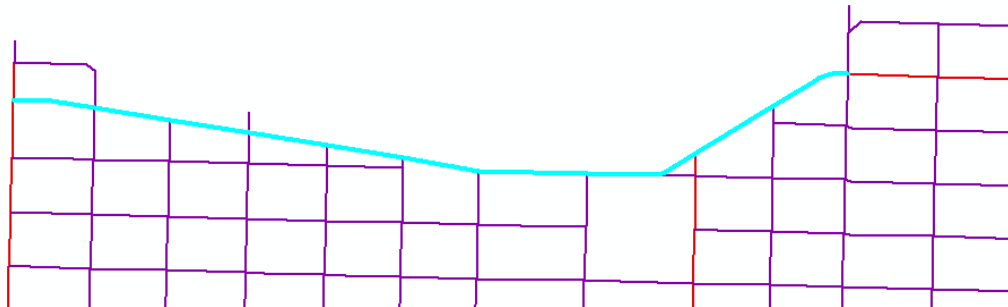
Working with Aliases

Aliases are alternatives names given to fields in a dataset that are often more descriptive than the actual field name.

Most users of FME who work with a Geodatabase will want to be able to create or maintain field aliases within their datasets.

When adding a File Geodatabase to a workspace, Alias control can be found under Schema Attributes > Alias Mode.

Alias Mode controls how Geodatabase aliases are used.

- **None**: Aliases are ignored.

- **Replace Attribute Names with Aliases** (Only applicable when adding a Reader): Attributes on feature types will be named based on their aliases rather than their official names. A geodb_feature_class_alias attribute will be included on each feature. Use this mode when the target format should create feature types using aliases as attribute names.

- **Expose Aliases as Metadata Attributes**: For each attribute read, a second <name>_alias attribute will be added that stores the alias for the attribute in question. A geodb_feature_class_alias attribute will also be included on each feature. Use this mode when the target format is Geodatabase and the aliases should be preserved during feature class and table creation.
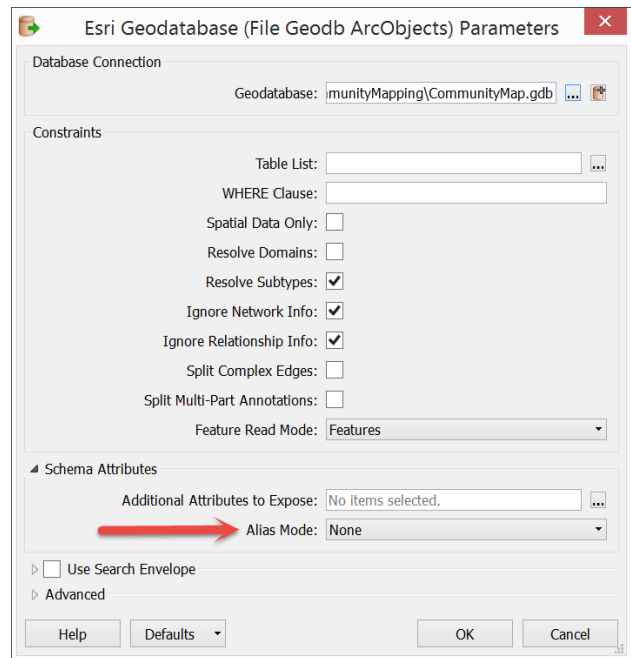
Exercise 3: Altering Alias Field Names	
Scenario	FME user; City of Interopolis, Planning Department
Data	Community Centre Data
Overall Goal	Create workspace to alter existing alias field names
Demonstrates	Alias control
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise3-Complete.fmw

The task here is to load Community Centre data into our VancouverCity Geodatabase. The Community Centre data has pre-existing alias field names; however, the planning department wishes to alter the alias names to make them more user friendly and to avoid field name size limits if the data ever needed to be exported to a different format.

1) Start ArcMap
Let's start by looking at the pre-defined Geodatabase.

Start ArcMap if necessary. In a Catalog window browse to
C:\FMEData2015\Data\CommunityMapping\CommunityMap.gdb\CommunityCentres

CommunityCentres is a feature class containing the Name, Address and Website URL of the various community centres located in the city.

You can examine the properties of this feature class to see that field aliases are present in the data.
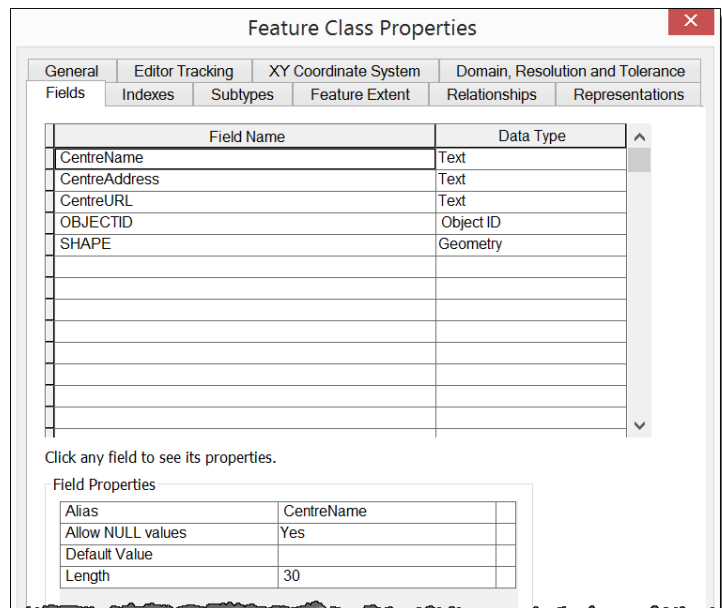
2) Start Workbench
Now let's translate this to Geodatabase and alter the alias field names.

Start Workbench and select Generate workspace.

Reader Format   Esri Geodatabase (File Geodb ArcObjects)

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

Select the Parameters button, click the Table List […] button and select the CommunityCentres Feature Type.

Expand the Schema Attributes area of the Parameters dialog and from the Alias Mode drop down menu select Expose Aliases as Metadata Attributes (<name>_alias).

Click Ok.
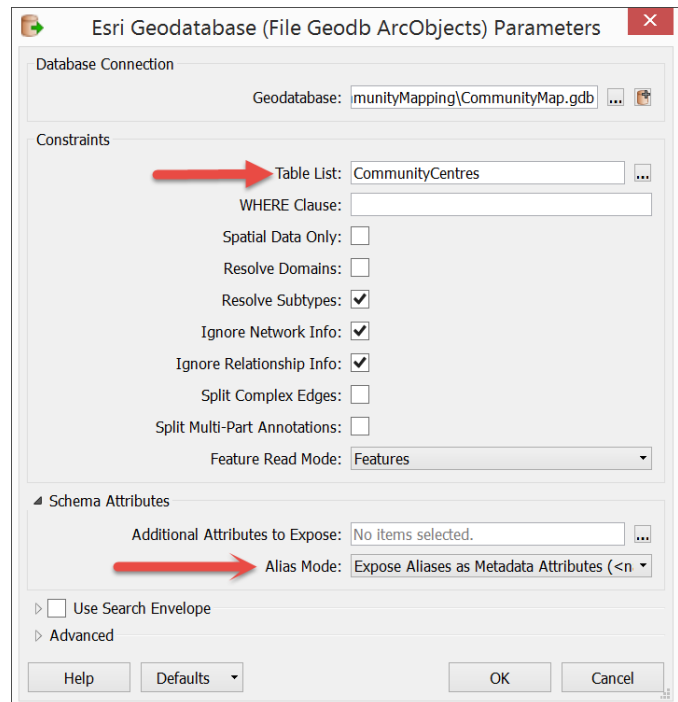
Writer Format   Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
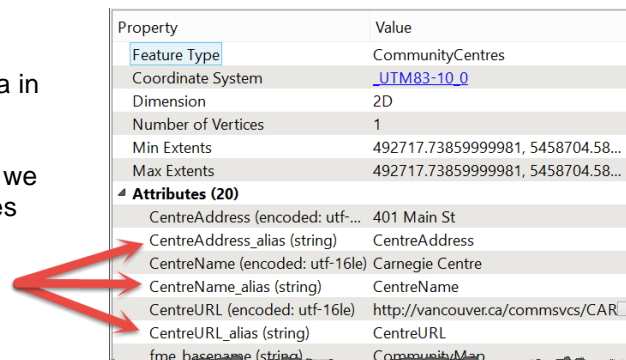
Click Ok.

5) Save and Run Workspace.
Let's see if our alias attributes are now available. Under the Writers menu, select Redirect to FME Data Inspector.

Run the workspace and inspect the data in the Data Inspector.

By selecting a Community Centre point we can see that the <name>_alias attributes are now available to us.

6) Add AttributeCreator
Place an AttributeCreator between the reader and writer.

Your workspace should look like the following:

7) Edit the AttributeCreator Parameters

Open the Parameters for the AttributeCreator and input the following values:
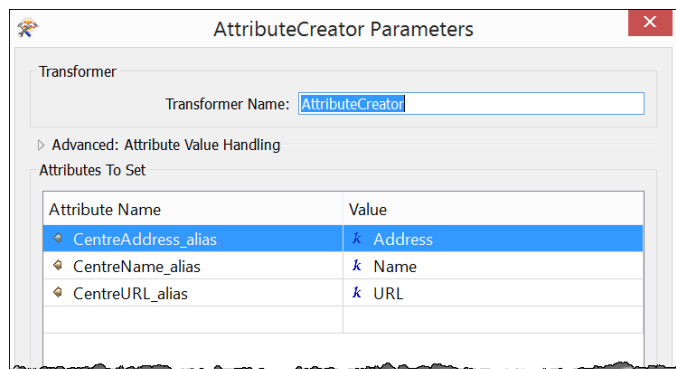
Click OK

8) Save and Run Workspace

Turn off the Redirect to Inspection Application option (if still on). Save the workspace and then run it.

Examine the results in ArcGIS and/or the Data Inspector *(remember to expose the aliases in the feature type parameters when opening the data in the Data Inspector)* to confirm the transformation worked.

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.

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.*

When a Reader feature type represents an annotation class, the General Properties will denote that like so:
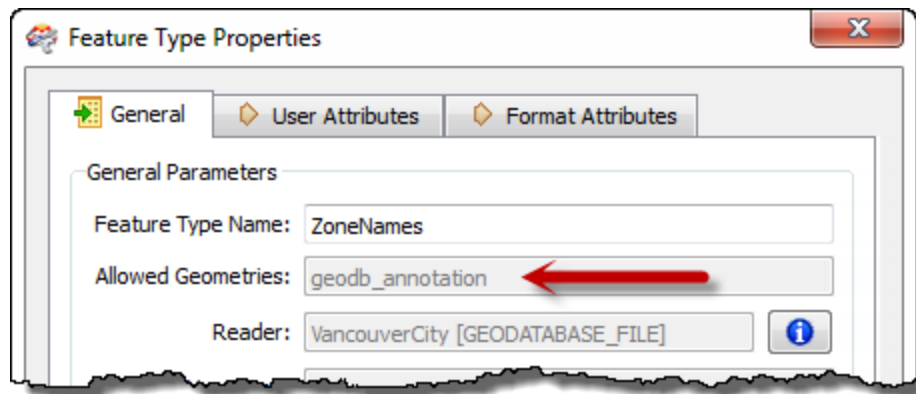
Format Attributes can be accessed by clicking the Format Attributes tab and exposed for use in the workspace by putting a checkmark in the 'Exposed' field:
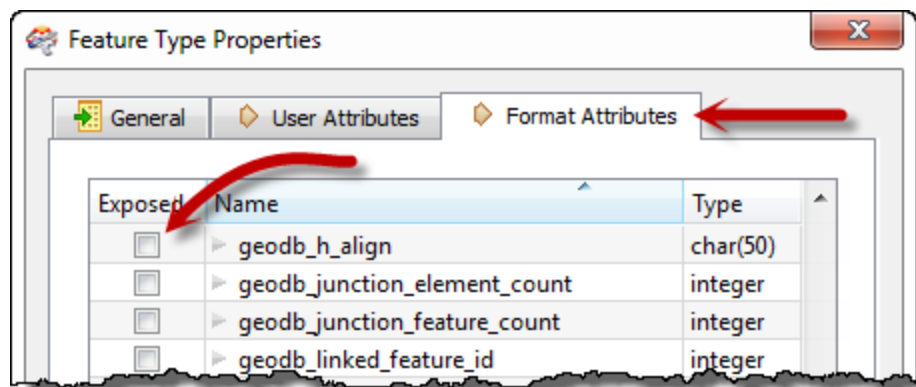
Multi-Part Annotation

The Geodatabase Reader has a "Split Multi-Part Annotations" 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.

⚙ Advanced
  ⚙ Split Multi-Part Annotations: No ⬅
  ⚙ Network Authentication: <not set>
  ⚙ Network Proxy: <not set>
  ⚙ Split Complex Annotations: No
  ⚙ Cache Multipatch Textures: Yes

⚙ Advanced
  ⚙ Split Multi-Part Annotations: No
  ⚙ Network Authentication: <not set>
  ⚙ Network Proxy: <not set>
  ⚙ Split Complex Annotations: No ⬅
  ⚙ Cache Multipatch Textures: Yes

The "Split Complex Annotation" parameter breaks up curved annotation into individual letters whenever rotation or font changes.

This results in many additional features, but allows for very accurate annotation placement.

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.
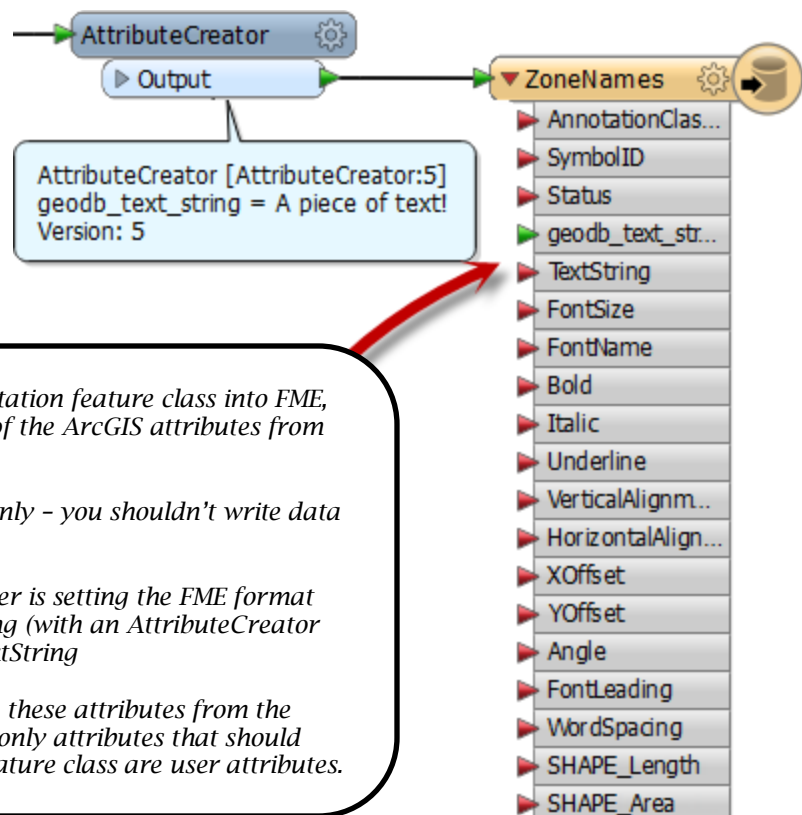
When importing an annotation feature class into FME, you may find that some of the ArcGIS attributes from the table are included.

These are for reference only – you shouldn't write data directly to them!

Here, for example, the user is setting the FME format attribute geodb_text_string (with an AttributeCreator transformer) and not TextString

Best practice is to remove these attributes from the feature type in FME. The only attributes that should exist on an annotation feature class are user attributes.

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:

⊟ 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, the fields will display the desired values because the attributes below correspond to some of the fields.

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.

Exercise 4: Writing Feature-Linked Annotation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Zoning Data
Overall Goal	Create workspace to write to existing feature-linked annotation classes
Demonstrates	Writing feature-linked annotation
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise4-Complete.fmw

The task here is to load zoning data into a feature class. Because there is already a relationship class defined, writing data will also create feature linked annotation. Linking will be done via the zone name

1) Start ArcMap
Let's start by looking at the pre-defined Geodatabase.

Start ArcMap if necessary. In a Catalog window browse to C:\FMEData2015\Resources\Esri\VancouverCity.gdb\City

City is a Feature Dataset containing the Feature Classes Zones (Polygons) and ZoneNames (Annotation). There is also a Relationship Class that defines the relationship between these two Feature Classes.

You can examine the properties of these classes to clarify the relationship that is defined.
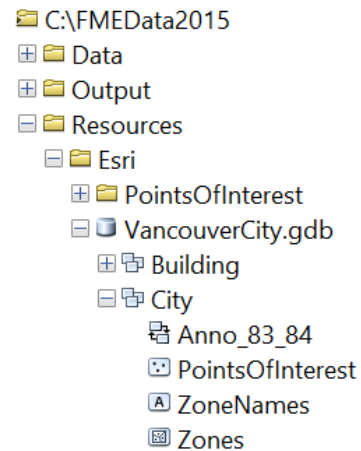
2) Inspect Source Data
Now let's inspect the source data we are to use.

In the ArcMap Catalog window browse to C:\FMEData2015\Data\Zoning\Zones.tab – this is a MapInfo TAB format dataset, but because we have extended ArcGIS with FME, you should be able to view it without problem.

Notice that the dataset consists of polygons only, each with a ZoneName attribute:
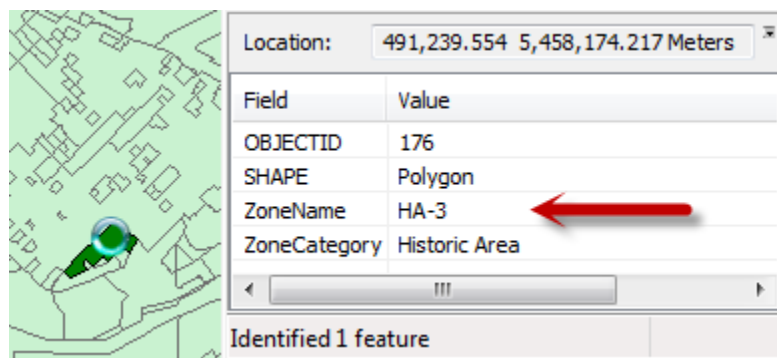
3) Start Workbench
Now let's translate this to Geodatabase and create Feature-Linked annotation.

Start Workbench and begin with an empty canvas.
On the menubar select Readers > Add Reader and add a reader to read the following dataset.

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2015\Data\Zoning\Zones.tab

4) 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	Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset	*C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
Feature Class or Table Definition: Import from Dataset	

The Import from Dataset option is selected because the Feature Class already exists in the Geodatabase; it can be imported more easily than being re-created.

5) Import Feature Type
When prompted set the format and select the file Geodatabase as in step 4 (it may – should – already be set by FME). Click **OK**.
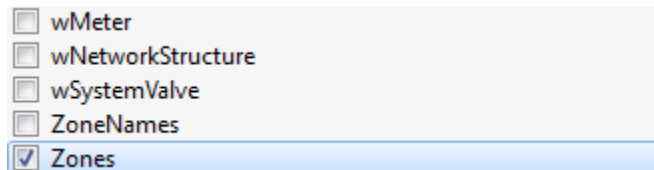
FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *Zones* only.

Now we have both a Reader and a Writer in the Navigator window and, on the canvas, a Feature Type for each of these. Go ahead and connect the two objects together:

6) Set Properties

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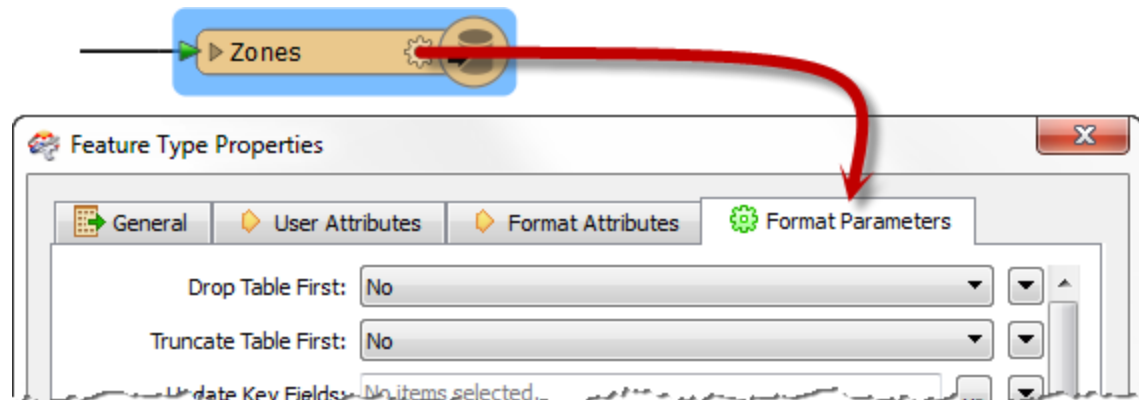
Setting these properties means that we will keep the schema of the existing table, and that we will be adding data – not replacing it – when we run the translation.

Now – in the User Attributes Tab – remove the user attributes flagged with a red arrow:

- 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.
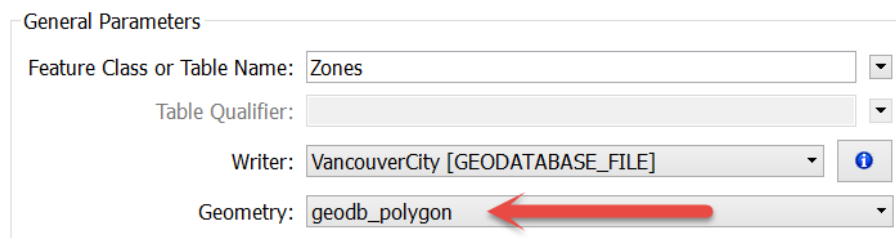
7) Set Transaction Type

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.

8) Save and Run Workspace

Save the workspace and then run it (close ArcMap first).

Edit sessions are always a little slower, so the translation should take a minute or two.

Inspect the output using ArcMap by adding the Zones Feature Class and the ZoneNames Annotation class to your map canvas.

The City feature dataset should now contain feature classes containing zones and zone (feature-linked) annotation.
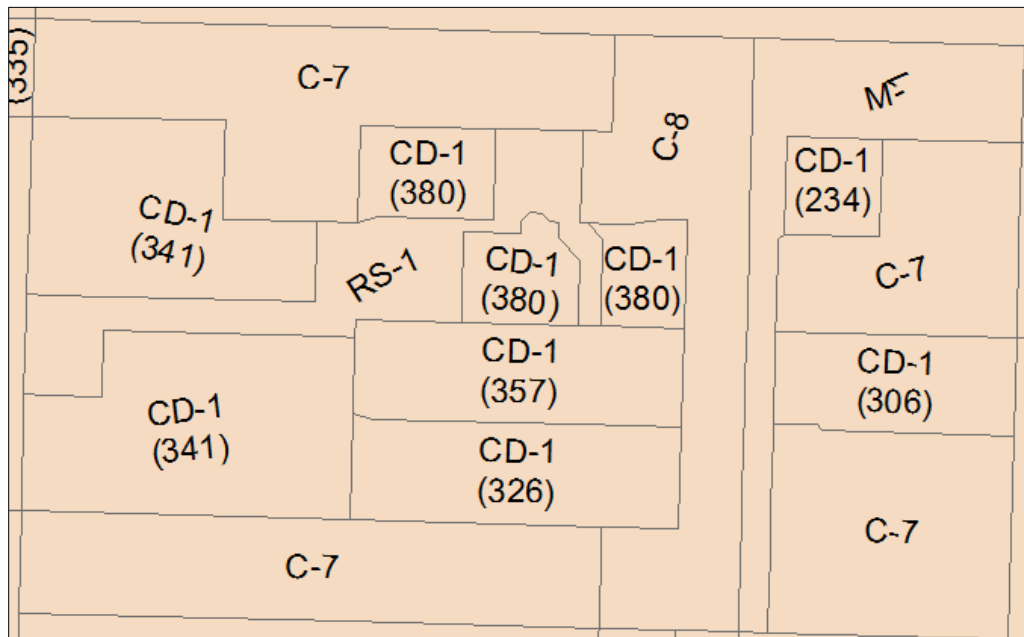
Remember: *If anything goes wrong with the translation, you can set up the Geodatabase again by simply running the workspace LoadVancouverCity.fmw*

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.

☑	► SymbolID	integer		
☑	► Status	coded_domain	View...	
☑	► Status_resolved	char	254	
☑	► 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:

<pre style="color:red">
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>.
</pre>

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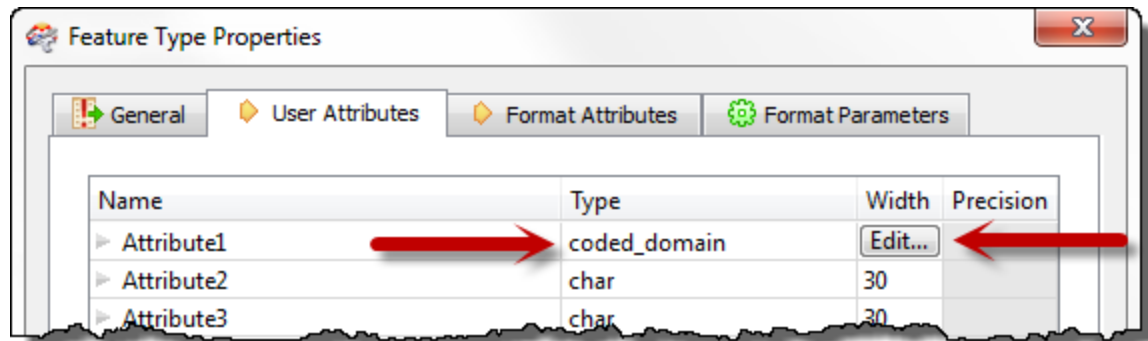
The next step is to click the Edit button in the attribute width field.

For an existing domain the dialog is very simple; you just fill in the name of the existing domain:
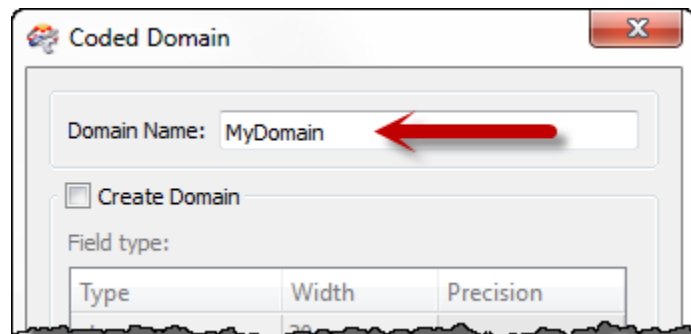
The range domain dialog is slightly different to the one for coded domain, but in this scenario it's still just one parameter (*Domain Name*) that matters.

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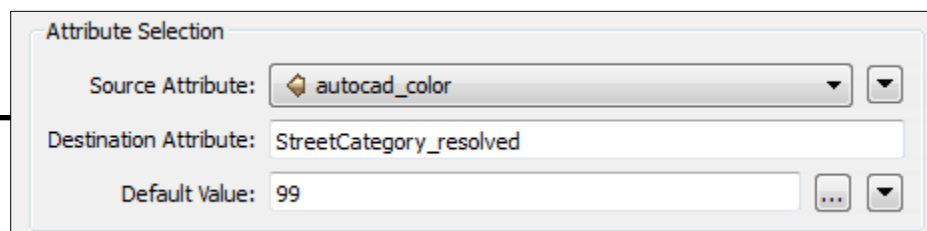
Writing to a new table and creating a new subtype/domain

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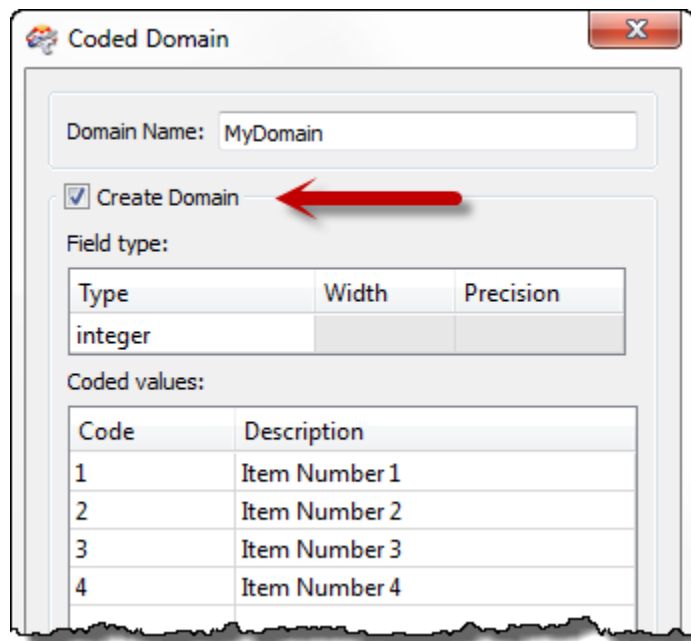
Here the user has a coded domain for a series of integers.

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.

Exercise 5: Writing a Coded Domain	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads Network (AutoCAD DWG)
Overall Goal	Create a workspace to write to a coded domain
Demonstrates	Handling Domains
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise5-Complete.fmw

In this example the requirement is to translate the roads network data again (that from exercise 2), this time adding it to the Vancouver Geodatabase.

1) Start ArcMap
Let's start by looking at the VancouverCity Geodatabase.

Start ArcMap if necessary. In a Catalog window browse to
C:\FMEData2015\Resources\Esri\VancouverCity.gdb

Examine the properties of this database (right-click > properties) and then inspect the available domains (Domains tab).
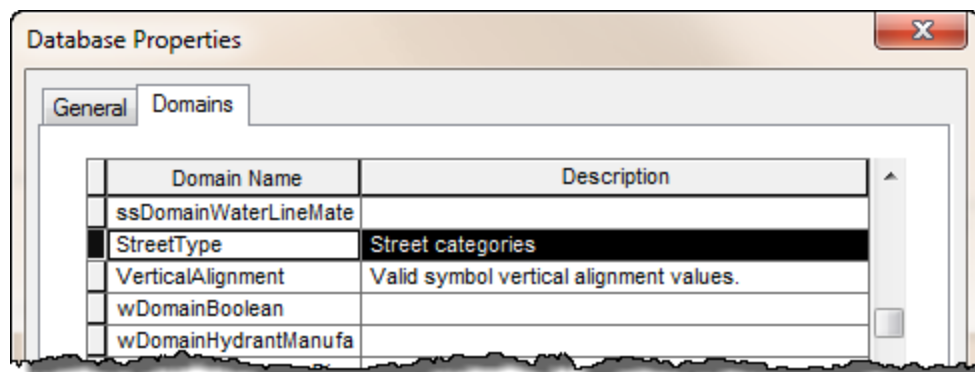
Notice that there is already a domain called StreetType. Its coded values are numbered from 1 to 5 and relate to the type of street (Arterial, Residential, etc.):

Domains

Also notice that there is a Feature Dataset (Engineering) with a Feature Class called Streets.

Notice how the Streets feature class has a field called StreetCategory that is connected to the StreetType domain:

Default Values and Domains:

Field Name	Default Value	Domain
StreetId		
From_HBlock		
HBlock		
StreetName		
StreetCategory		StreetType
SHAPE_Length		
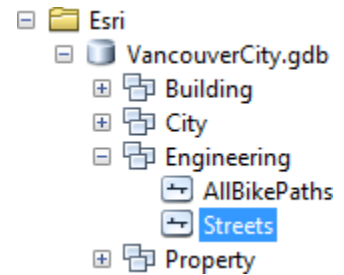
So here the task for us is to insert data into this feature class, making sure the StreetCategory values match those required by the domain.

2) Start Workbench
Start FME Workbench and begin with a new, empty workspace.

On the menubar select Readers > Add Reader and add a reader to read the following dataset.

Reader Format   Autodesk AutoCAD DWG/DXF
Reader Dataset   C:\FMEData2015\Data\Transportation\Roads.dwg

Use the Parameters button to change the Group Entities By parameter to Attribute Schema.
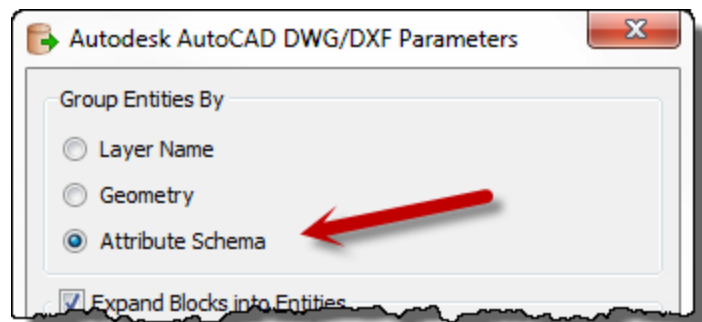
Also change the Workflow Option parameter from "Individual Feature Types" to "Single Merged Feature Type". This just means that we'll only get a single Reader feature type to read all the source data, which is perfectly fine here:
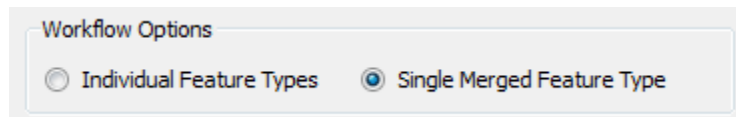
Click OK to add the Reader to the workspace.

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   Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
Feature Class or Table Definition: Import from Dataset

The Import from Dataset option is selected because the Feature Class already exists in the Geodatabase; it can be imported more easily than being re-created.

4) Import Feature Types

When prompted set the format and select the file Geodatabase as in step 3 (it may – should – already be set by FME). Click **OK**.
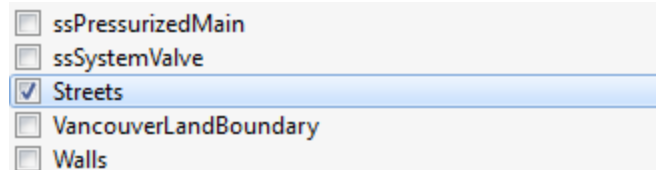
☐ ssPressurizedMain ☐ ssSystemValve ☑ Streets ☐ VancouverLandBoundary ☐ Walls	FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *Streets* only.

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

Go ahead and connect the two objects together, and then – on the Reader feature type – expose the Format Attribute called autocad_layer:

For this dataset the layer name matches the category of road/street, so we can use that to populate the domain field.

5) Edit Schema

On the Writer schema open the properties dialog and click the User Attributes tab. Remove the attributes that we don't need:

- OBJECTID
- SHAPE_Length

Under the StreetCategory attribute change the Type to coded_domain. Click the Edit button that appears and enter StreetType as the Domain Name.
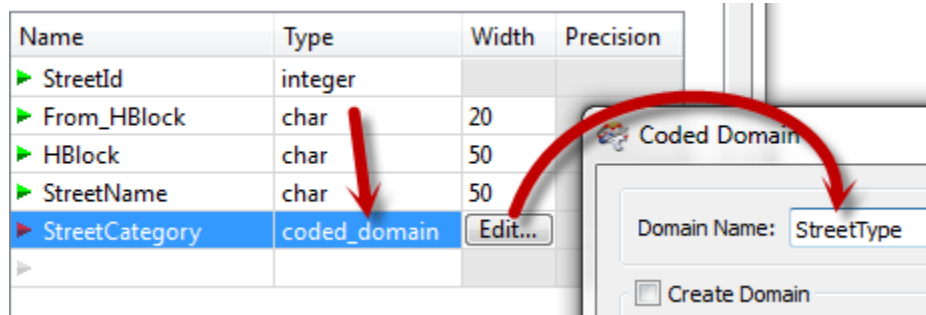
Now click on the Format Parameters tab. Ensure that Drop Table First is set to No (we want to keep the table already there) but Truncate Table First is set to Yes (we want to make sure it is empty).
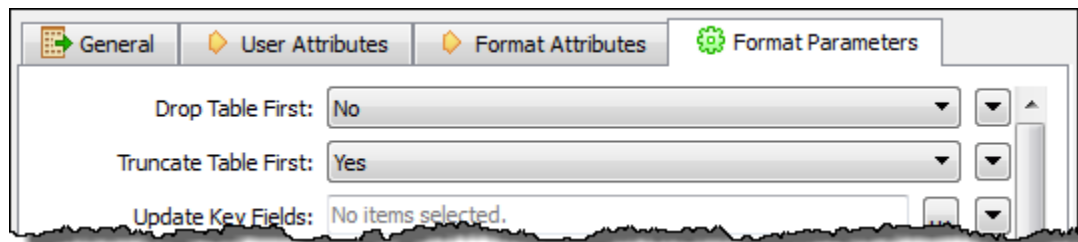
Also set Feature Dataset to Engineering:

Click OK to close the properties dialog.

6) Add AttributeRenamer

Now let's deal with the domain. In the first case, let's pretend we don't know what the actual domain codes should be; we know the resolved street types (Arterial, Residential, etc.) but we don't know the domain codes they map to.

To deal with this, add an AttributeRenamer transformer between the Reader and Writer feature types:
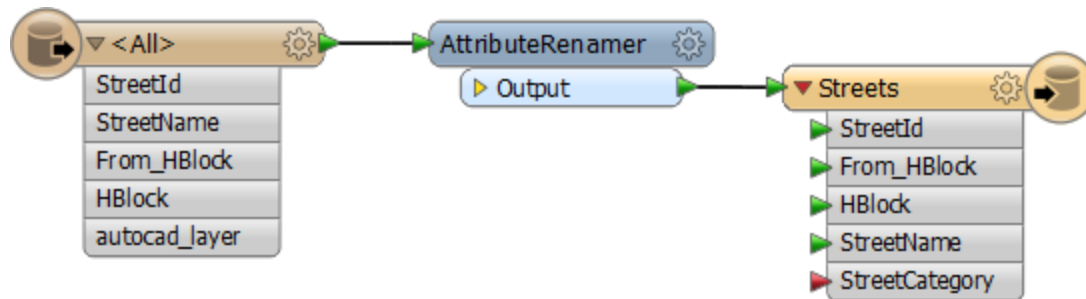
Open the parameters dialog. Rename *autocad_layer* to *StreetCategory_resolved* (you can pick *autocad_layer* from a list as long as you've exposed it, but need to type in *StreetCategory_resolved*)

Old Attribute	New Attribute	Default Value
◆ autocad_layer	StreetCategory_resolved	

This is how we handle not knowing the actual domain codes; we provide the resolved values (domain descriptions) in an attribute with *_resolved* as a suffix.

7) Save and Run the workspace

Check the Writer parameter for Overwrite Existing Geodatabase is set to No.

Now save the workspace and then run it.

The data will be written to the Geodatabase and can be inspected in ArcMap and/or the Data Inspector:
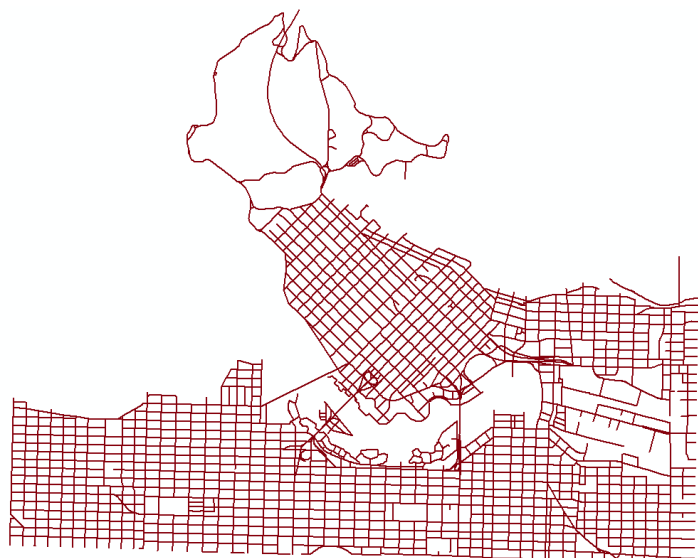
8) Add AttributeValueMapper
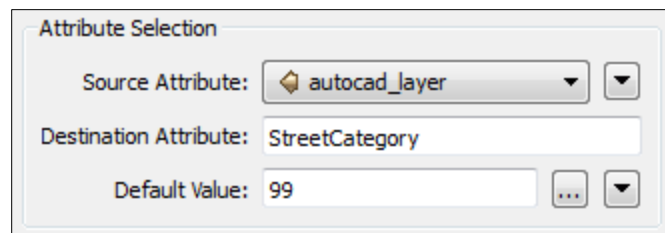
Now let's take the case where we DO know the actual domain codes.

Add an AttributeValueMapper transformer between the Reader and Writer features types, in parallel with the AttributeRenamer. Then disable the AttributeRenamer (Ctrl+E is the keyboard shortcut to use):
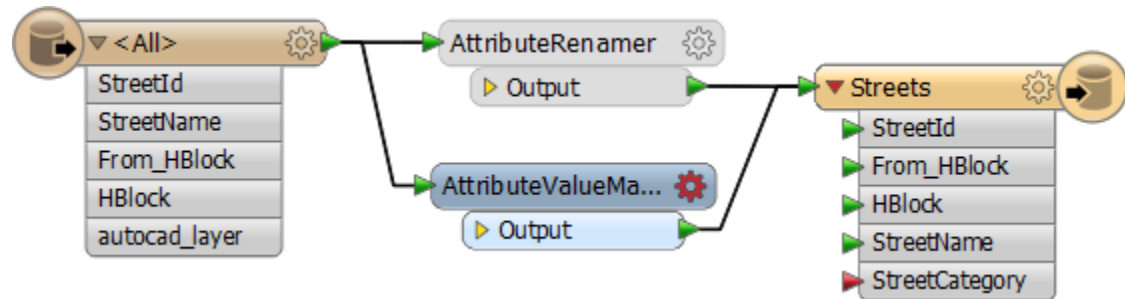
Open the AttributeValueMapper's parameters dialog. This is where we can set up a mapping between the source layer and the domain code.

Select autocad_layer as the Source Attribute and enter StreetCategory as the destination attribute. Enter 99 as the default value as – you may have noticed – this is the domain code for unknown.
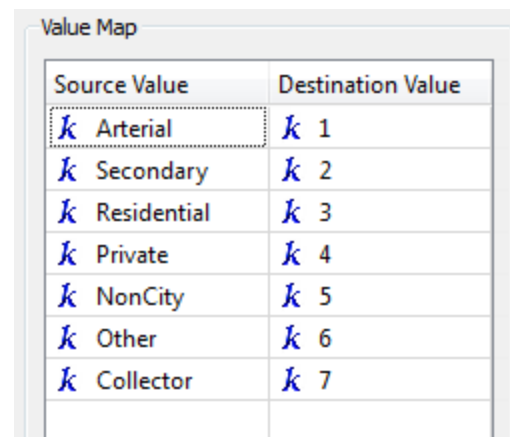
In the Value Mapping part of the dialog we need to map the following:

Source Value	Destination Value
Arterial	1
Secondary	2
Residential	3
Private	4
NonCity	5
Other	6
Collector	7

Click OK to close the dialog.

Domains

Field	Value	
OBJECTID	2890	
SHAPE	Polyline Z	
StreetId	12976	
From_HBlock	500	
HBlock	500 Union St	
StreetName	Union St	
StreetCategory	Residential	←
SHAPE_Length	141.51638	

9) Save and Run Workspace
Save the workspace and then run it.

Examine the output in ArcMap. You will find that, even though we wrote domain IDs, the result has been resolved to a proper name.

10) Validate Data
The one advantage of providing domain IDs – as opposed to the resolved values – is that the data can be validated when written.

Back in Workbench locate the Writer parameter *Validate Features to Write*. It appears under the advanced section. Change this to Yes.

- Annotation Units: unknown_units
- Compress Database When Done: No
- Validate Features to Write: Yes ←
- Simplify Network Features: No
- SQL Statement To Execute Before Translation: <not set>

Now re-run the workspace.

This time the translation will fail with an error message:

```
ERROR |Validation failed for a feature being written to the
table/feature class Streets.
The error message is: Field StreetCategory attribute value 7 is not
member of coded value domain StreetType.
```

The error occurs because we are trying to insert a feature with a value of 7, whereas (if you recall) only values 1-5 exist in the predefined domain.

Domains and the Geodatabase API
The Esri Geodatabase API treats domains slightly different when compared to the ArcObjects Geodatabase.

Coded domain fields are read and written using their decoded values (i.e., descriptions are used, not coded values). The Geodatabase API, in the current release, does not support creating tables or Feature Classes with domain restrictions. It can only read and write into tables and Feature Classes with existing domain definitions.

The following demonstrates how the Geodatabase API can be used to write to a Feature Class with existing domain constraints:

Open C:\FMEData2015\Workspaces\Esri\Exercise5_API-Complete.fmw in Workbench. The workspace contains the same input Roads ACAD file from Exercise 5 and it is still writing to the Streets Feature Class in the VancouverCity.gdb; however, the writer type is now Geodatabase API.
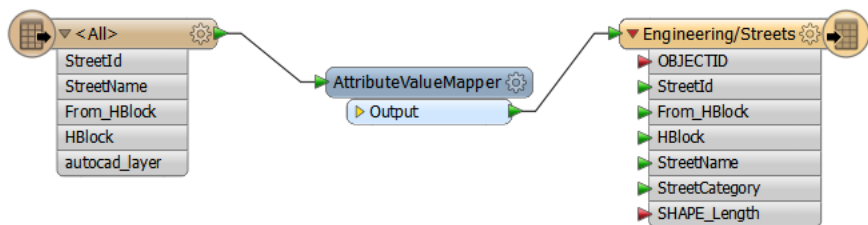
Click on the AtrributeValueMapper to open the parameters dialog window.

Notice how the Destination Values contained in the Value Map section must match the domain description of a predefined domain and not a coded value.
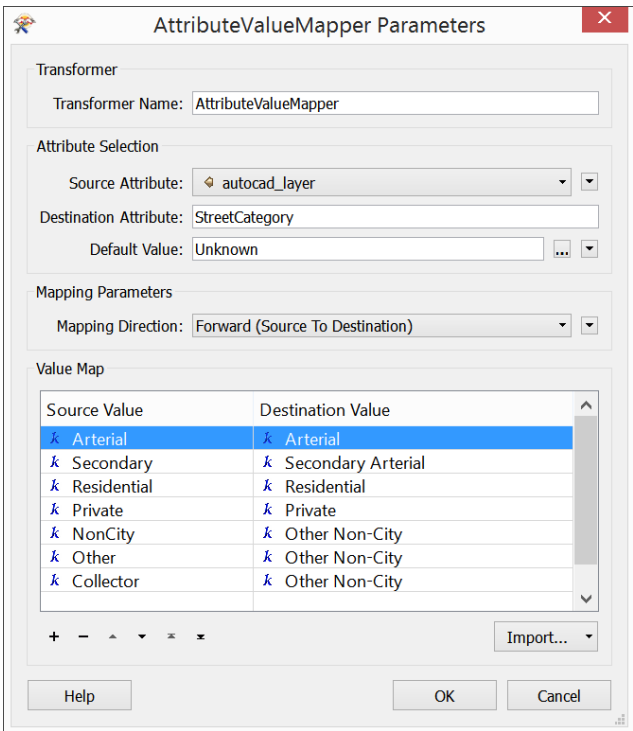
Save the workspace and then run it.

Examine the output in ArcMap. You will find that the result has been resolved to a proper name based on the domain constraints.

Field	Value
OBJECTID	36584
SHAPE	Polyline Z
StreetId	10724
From_HBlock	1900
HBlock	1900 Creelman Av
StreetName	Creelman Av
StreetCategory	Residential
SHAPE_Length	45.262971

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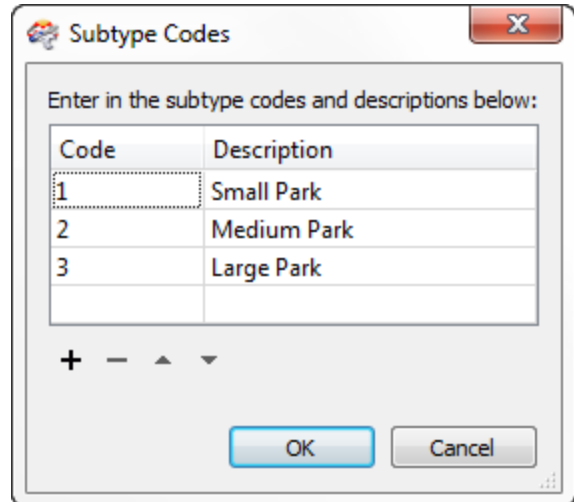
Of course, once again, creating subtypes is possible with FME.

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.
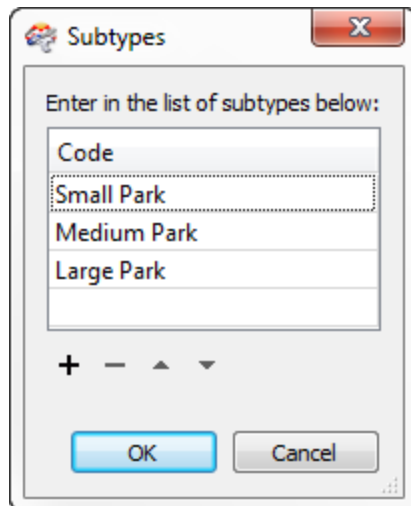
Subtype Codes

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.

Exercise 6: 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:\FMEData2015\Workspaces\Esri\Exercise6-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 (MITAB)
Reader Dataset   C:\FMEData2015\Data\Parks\Parks.tab

Writer Format   Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset   C:\FMEData2015\Resources\Esri\VancouverCity.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 Parks. It should have *geodb_polygon* as the permitted geometry type.

The workspace will now look like this:

3) Add Subtype Definition
Open the Properties dialog of the sole remaining Writer feature type.

Click the User Attributes tab and add a new attribute called ParkSize. Set the field type to *subtype_codes* then click the Edit button.

▶ Washrooms	char	1	
▶ SpecialFeatures	char	1	
▶ ParkSize	subtype_codes	Edit...	
▶			

Define the subtype codes as follows:

Code	Description
1	Small Park
2	Medium Park
3	Large Park
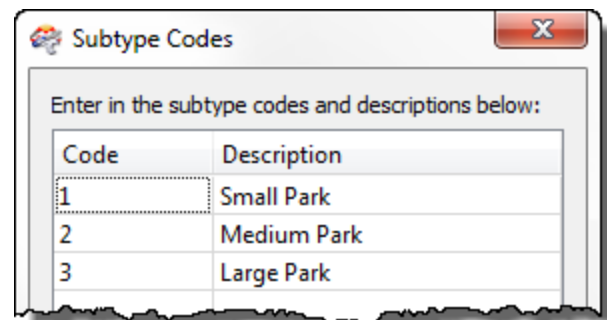
Click OK to close the dialog.

Next, still in the Properties dialog, click the Format Parameters tab. Ensure Drop Table First is set to No and that Truncate Table First is set to Yes.

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

5) Add AttributeRangeMapper
Now add an AttributeRangeMapper transformer. This will be used to create subtype values.
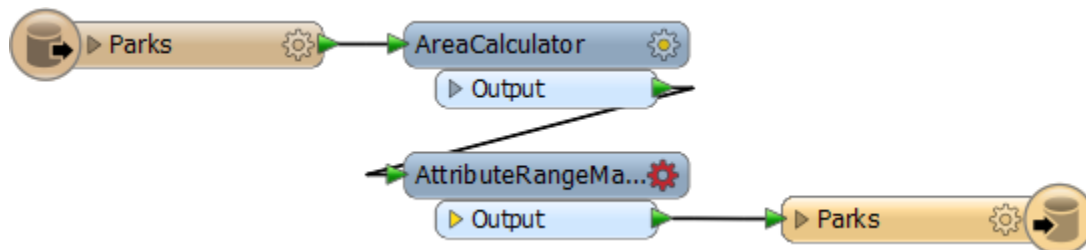
6) Set AttributeRangeMapper Parameters

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

Source Attribute   _area
Output Attribute   ParkSize

Lookup Table

From	To	Output Value
	10,000	1
10,000	1,000,000	2
1,000,000		3

7) 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:
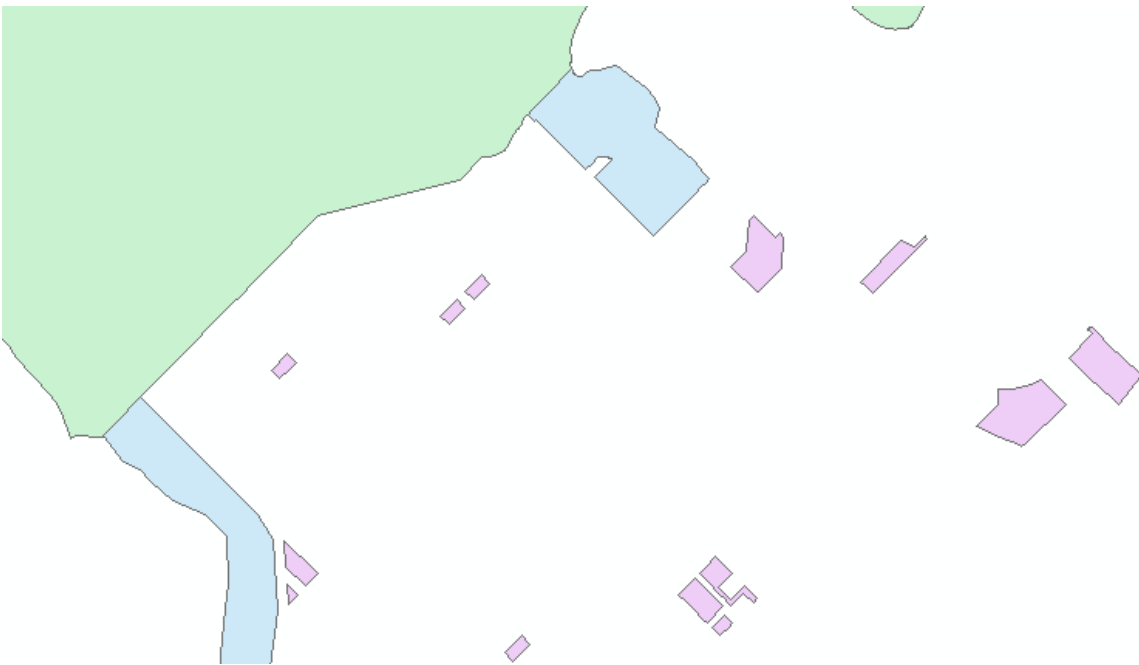
Field	Value	
OBJECTID	75	
SHAPE	Polygon	
ParkId	75	
RefParkId	18	
ParkName	Devonian Harbour Park	
NeighborhoodName	Downtown	
EWStreet	W Georgia Street	
NSStreet	Denman Street	
DogPark	N	
Washrooms	N	
SpecialFeatures	Y	
ParkSize	Medium Park	
SHAPE_Length	981.896659	
SHAPE_Area	43247.417914	

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.

Parks

	ParkId	ParkName	NeighborhoodName	ParkSize
	1	<Null>	Kitsilano	Small Park
	2	Rosemary Brown Park	Kitsilano	Small Park
	3	Tea Swamp Park	Mount Pleasant	Small Park
	4	<Null>	Strathcona	Small Park
	5	Morton Park	West End	Small Park
	6	Mcbride Park	Kitsilano	Medium Park
	7	Granville Park	Fairview	Medium Park
	8	<Null>	Mount Pleasant	Small Park
	9	Creekside Park	Mount Pleasant	Medium Park

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

Advanced
As an advanced exercise, open the workspace C:\FMEData2015\Workspaces\Esri\Exercise6-Advanced-Complete.fmw

This workspace reads the domain and subtype information from the Geodatabase XML template and writes it out as an Excel spreadsheet.

The idea is that it is very useful to be able to obtain this information in a format that is a bit more usable than XML!

Subtypes and the Geodatabase API
The Esri Geodatabase API treats subtypes slightly differently when compared to the ArcObjects Geodatabase.

Coded subtype fields are read and written using their decoded values (i.e., the descriptions are used, not coded values). The Geodatabase API, in the current release, does not support creating

tables or Feature Classes with subtype restrictions. It can only read and write into tables and Feature Classes with existing subtype definitions.
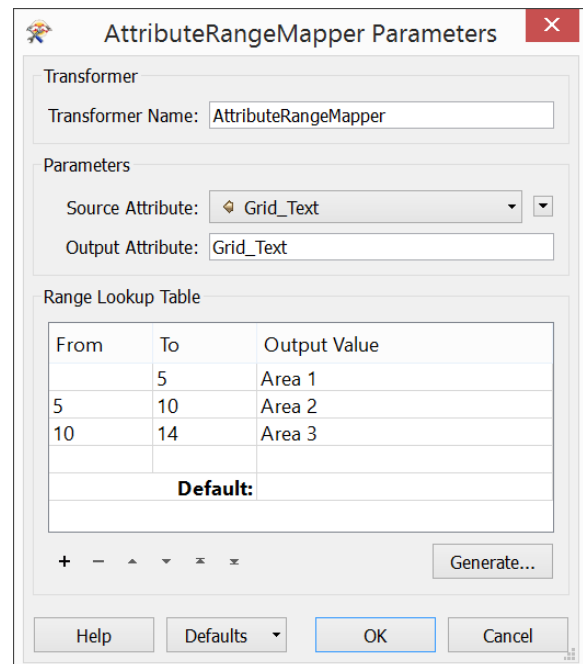
The following demonstrates how the Geodatabase API can be used to write to a Feature Class with existing subtype constraints:

Open C:\FMEData2015\Workspaces\Esri\Exercise6_API-Complete.fmw in Workbench. The workspace contains the same input CityGrid GML file from Exercise 1 and it is still writing to the Grid Feature Class in the VancouverCity.gdb; however, the writer type is now Geodatabase API.
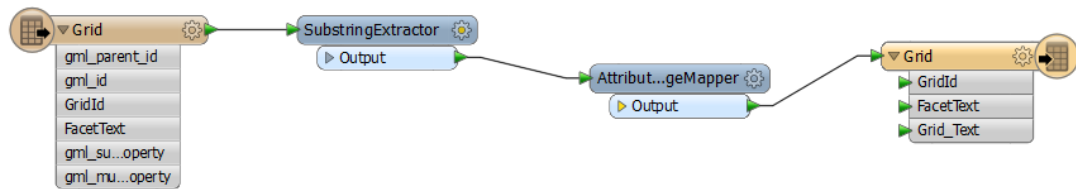
Click on the AtrributeRangeMapper to open the parameters dialog window.

Notice how the Output Values contained in the Range Lookup Table must match the subtype description of a predefined subtype and not a coded value.

Save the workspace and then run it.

Examine the output in ArcMap. You will find that the result has been resolved to fewer sections based on the subtype constraints.

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.

Reading from Geometric Networks

To read a network you need to set the parameters *Ignore Network Info* to 'No' and *Split Complex Edges* to 'Yes'. When this is done, reading from the network takes place and a number of format attributes (most of which store connectivity information) are populated.

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 full format attributes available.

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.

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.

However, FME **can** 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*.

Performance

When writing to a geometric network, all of the connectivity information must be constructed. Therefore, writing is faster if the network information is ignored. This can be achieved using the "Simplify Network Features" parameter for the Geodatabase Writer. A geometric network won't be formed by the features written, but the loading process will take place much more quickly.
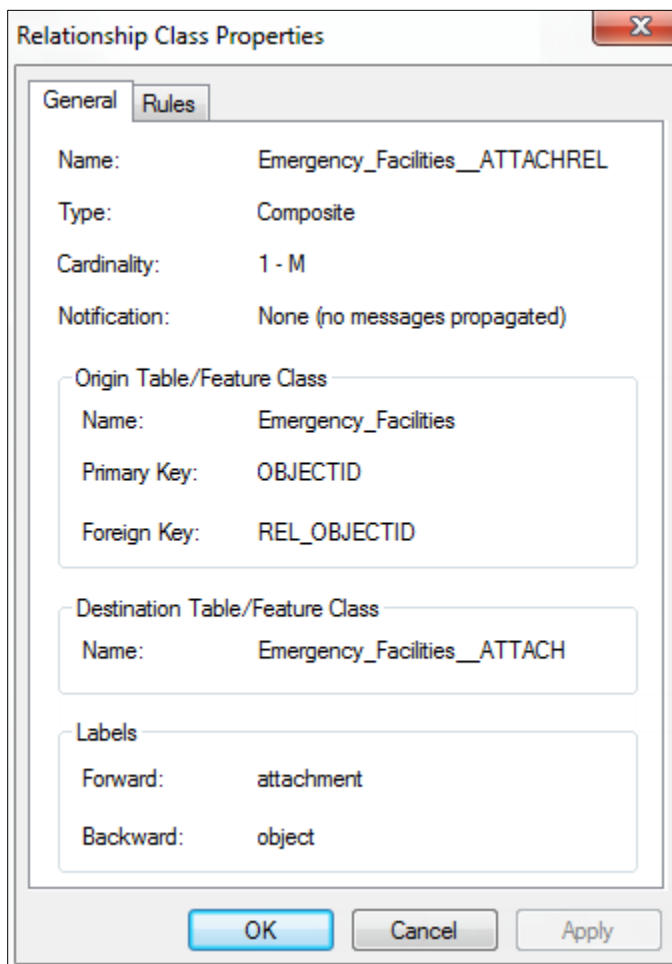
Relationship Classes

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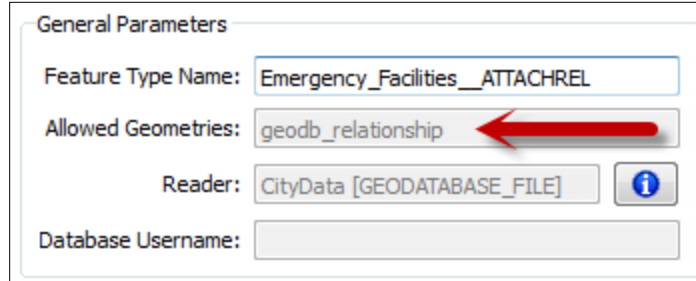
In this example, Emergency Facilities (the Origin) are related to a set of Attachments (the Destination).

In a "Simple" Relationship class, features can exist independently of each other – for example an emergency facility can exist without necessarily having an attachment, and any attachment can be deleted without affecting the facility.

In a "Composite" Relationship class, origin and destination objects are more closely connected. If an Emergency 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
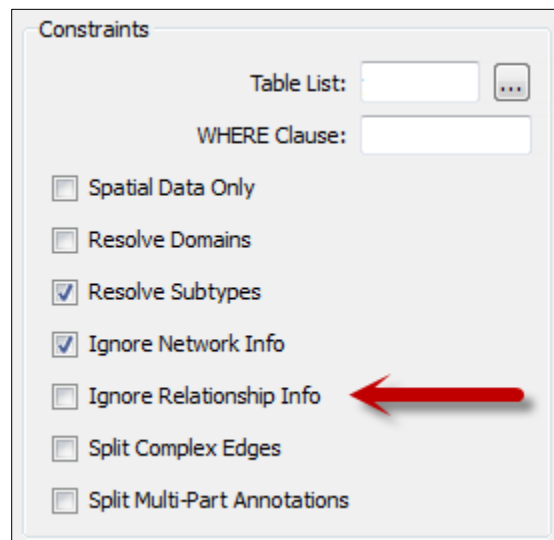
These attributes can be used to identify relationships between the origin and destination features within the workspace.

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	geodb_oid geodb_feature_has_relationships = *yes*
Destination feature class or table	geodb_oid geodb_feature_has_relationships = *yes*
Relationship class	geodb_rel_origin_oid geodb_rel_destination_oid geodb_type = *geodb_relationship* or *geodb_attributed_relationship*

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.

ArcGIS Attachments

Attachments are one 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. However, the attachments table has to already have been created in ArcGIS.

The key is to read the contents of the file to be attached into an attribute, and write 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.

Exercise 7: Attachments	
Scenario	FME user; City of Interopolis, Planning Department
Data	Points of Interest
Overall Goal	Attach photographic images to Points of Interest
Demonstrates	Esri ArcGIS Attachments
Starting Workspace	C:\FMEData2015\Workspaces\Esri\Exercise7-Begin.fmw
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise7-Complete.fmw

The city has a dataset containing various points of interest. However, before it is released for use, they wish to merge in documents, as attachments in a Geodatabase.

1) Start ArcMap
In an ArcMap catalog window, examine the VancouverCity file Geodatabase:
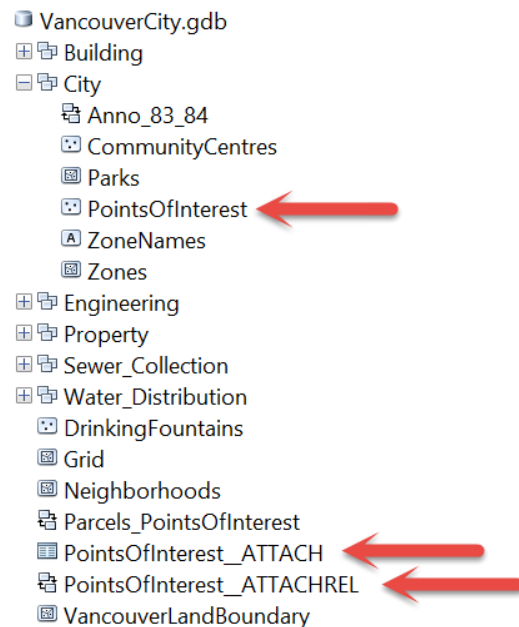
Notice there are simple feature classes for PointsOfInterest, and PointsOfInterest Attachments; plus and a relationship table to connect the two:

2) Start Workbench
Start Workbench and open the starting workspace for this exercise (Exercise7-Begin.fmw):

ArcGIS Attachments
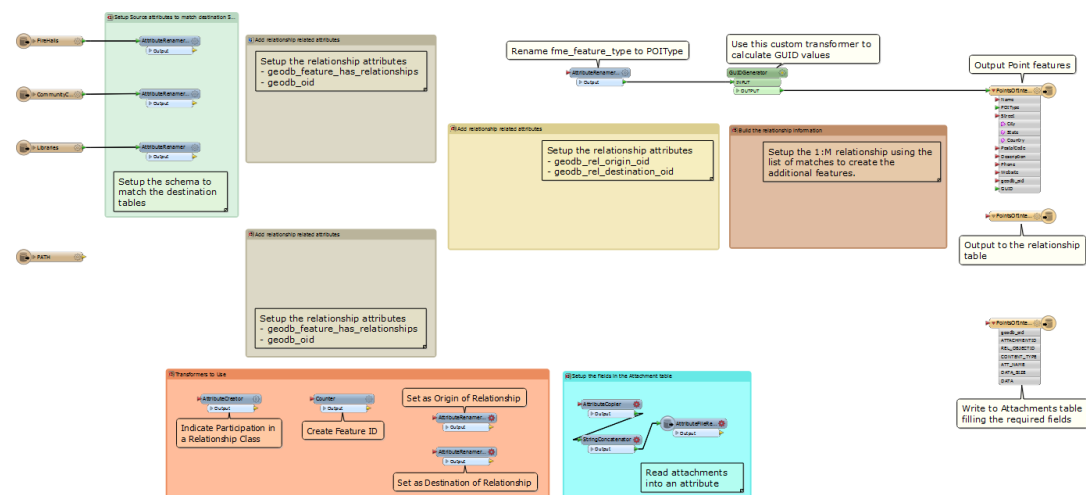
You'll immediately notice that the workspace is already partially set up. The source and destination schema objects are already in place and ready to read from/write to.

There are also a few transformers already in place to map the Reader and Writer schema.

The only things missing are the transformers required to set up the relationship; so adding them is the main task for this exercise.

However, we'll be helped by a number of predefined transformers that have been set up and copied to a bookmark:
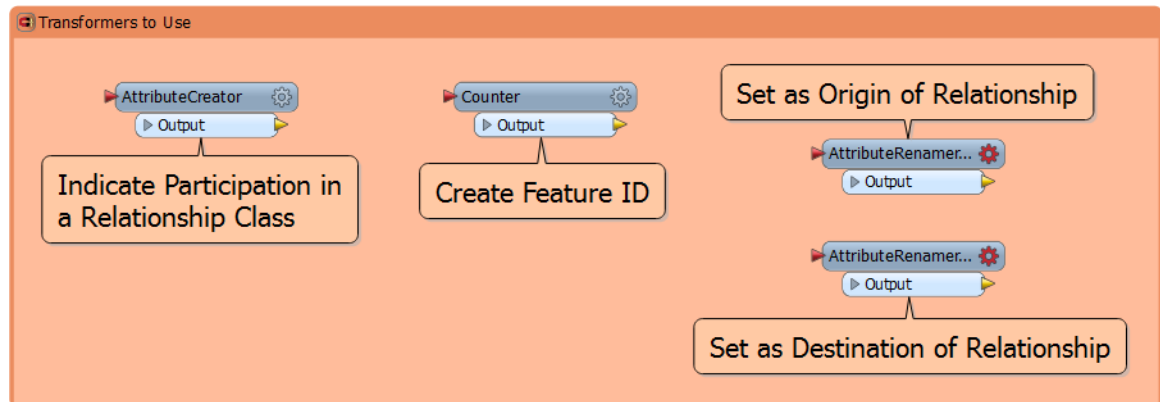
3) Place AttributeCreator Transformers
The first thing to do is flag all features to let FME know they are to participate in a relationship class. There is already a transformer in the "Transformers to Use" bookmark to use for this.
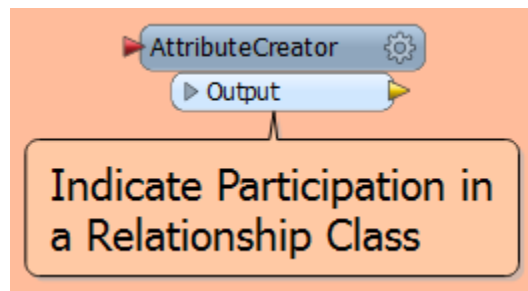
We'll need two instances of this transformer, because there are two streams of data (the Points of Interest and the Attachments).

So copy/move this transformer into both of the light brown bookmarks, and connect the source features to it:

For the Points of Interest, connect all three existing transformers to the new AttributeCreator:

ArcGIS Attachments
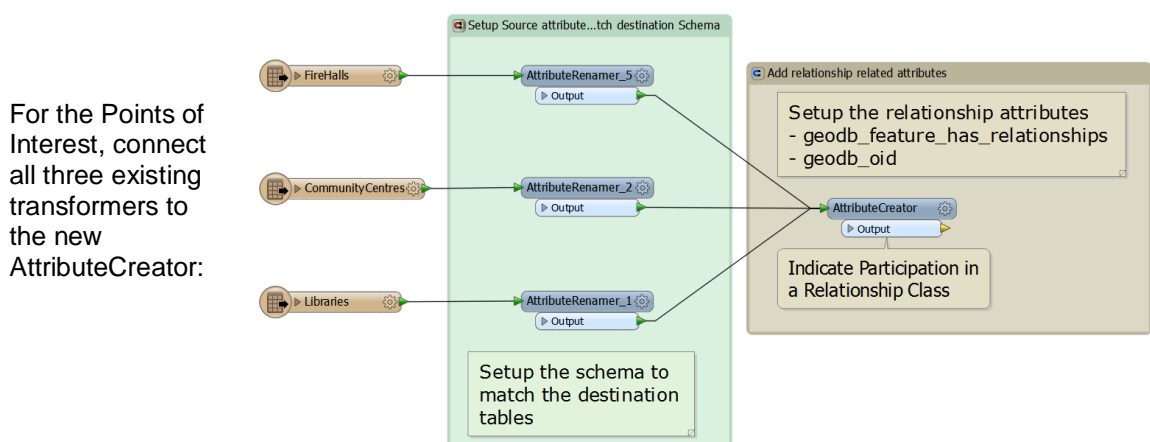
Open the properties dialog for the transformer and you'll see that all it does is set a format attribute called *geodb_feature_has_relationships* – this is all that is required to tell FME that these features participate in a relationship.
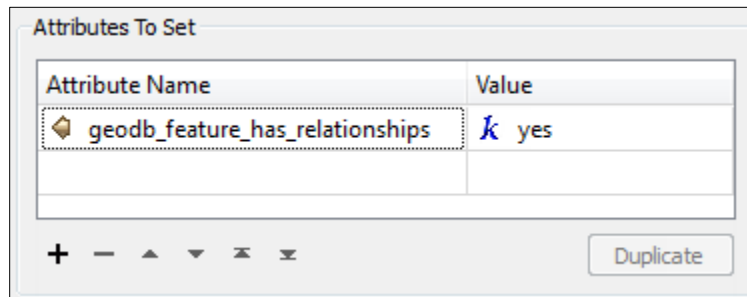
4) Place Counter Transformers

Each set of features also needs an ID number. Assuming that one does not already exist, we can create one with a Counter transformer. Again, we need one instance per set of features, and there is already a pre-defined transformer in the "Transformers to Use" bookmark.

So move/copy the Counter transformer from the bookmark and connect an instance of it after each of the AttributeCreator transformers from the previous step:
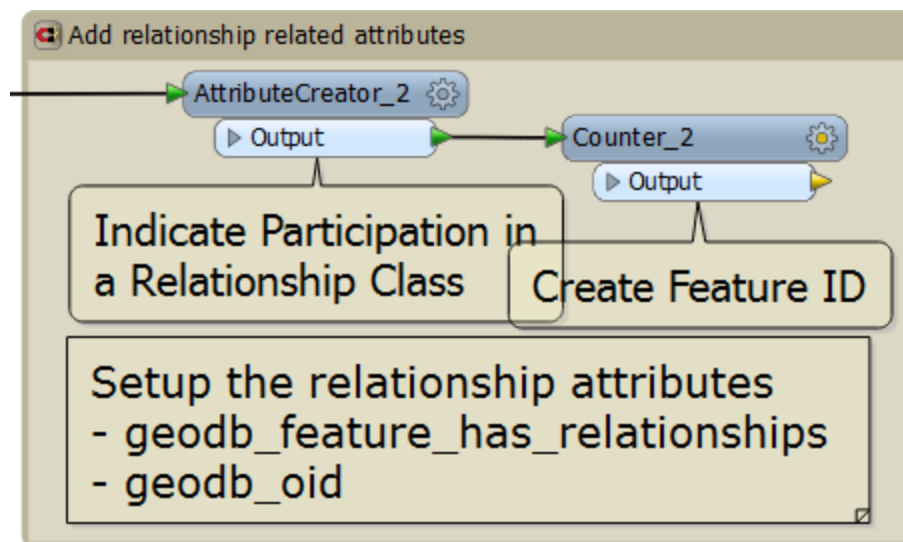
Inspect the Counter parameters and you'll find it is creating an attribute called geodb_oid:
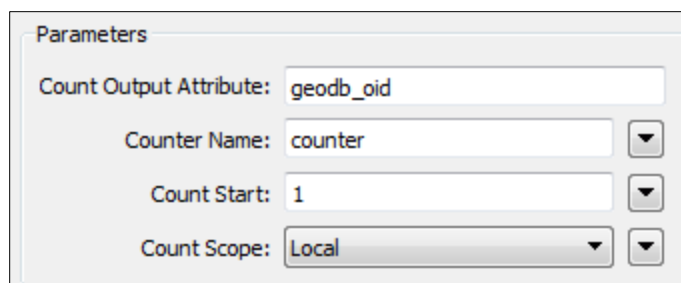
5) Connect Schema

At this point these features are all ready to write to the Geodatabase, so you can create connections from the Counter transformers to the existing AttributeRenamer transformer and PointsOfInterest__ATTACH feature type:
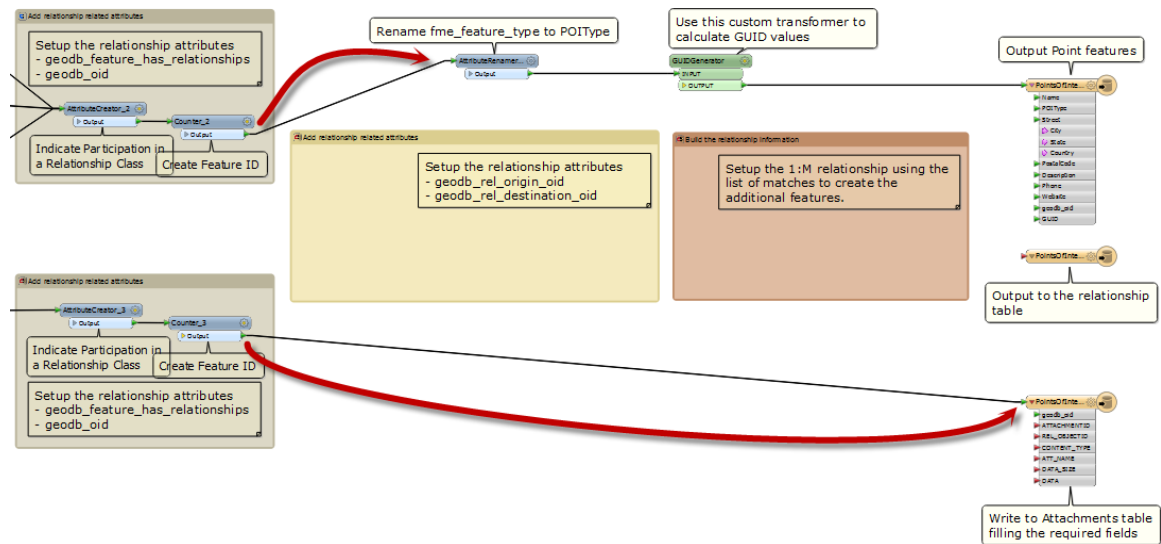
Open the parameters dialog for the AttributeRenamer_POIType transformer. The parameters should be set as follows:

Attributes To Rename

Old Attribute	New Attribute	Default Value
fme_feature_type	POIType	

However, what we still need to do is create the features to write to the relationship class.

6) Place AttributeRenamer Transformers

We now need to define the relationship between the two sets of features, and we do this by creating *origin* and *destination* ID numbers.

Again, we need one instance per set of features, and there are already two pre-defined transformers we can use.

Copy the two AttributeRenamer transformers from the "Transformers to Use" bookmark and into the yellow, centre bookmark.

Make a duplicate connection from each Counter transformer to one of the AttributeRenamers:
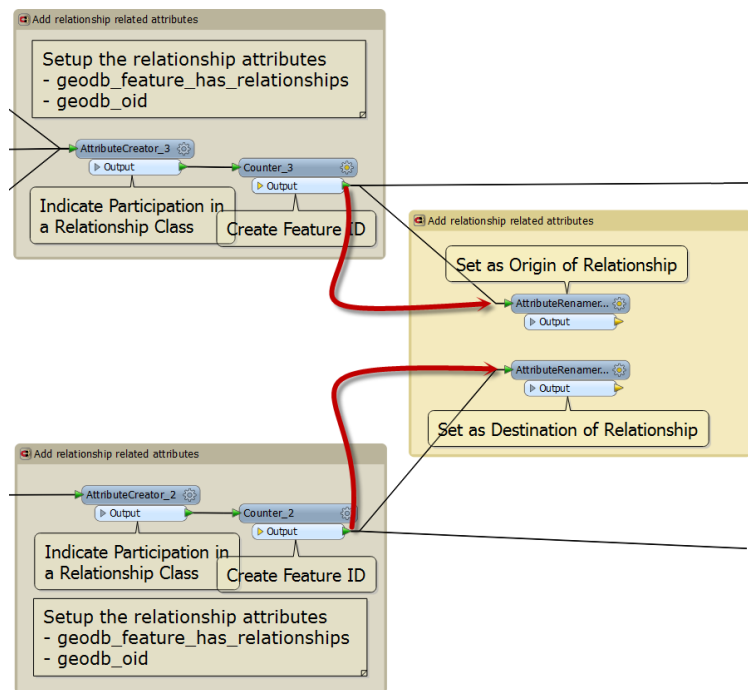
Check the parameters dialog for each AttributeRenamer. You'll see that each is renaming the newly created ID to either an Origin or Destination ID. This is what FME will use to write the relationship class.

The only task now is to join the two streams together to fully define how these IDs are related.

7) Place FeatureMerger Transformer
Add a FeatureMerger transformer. This is how the relationship will be built.

Connect the Origin AttributeRenamer to the Requestor port and the Destination AttributeRenamer to the Supplier; like so:
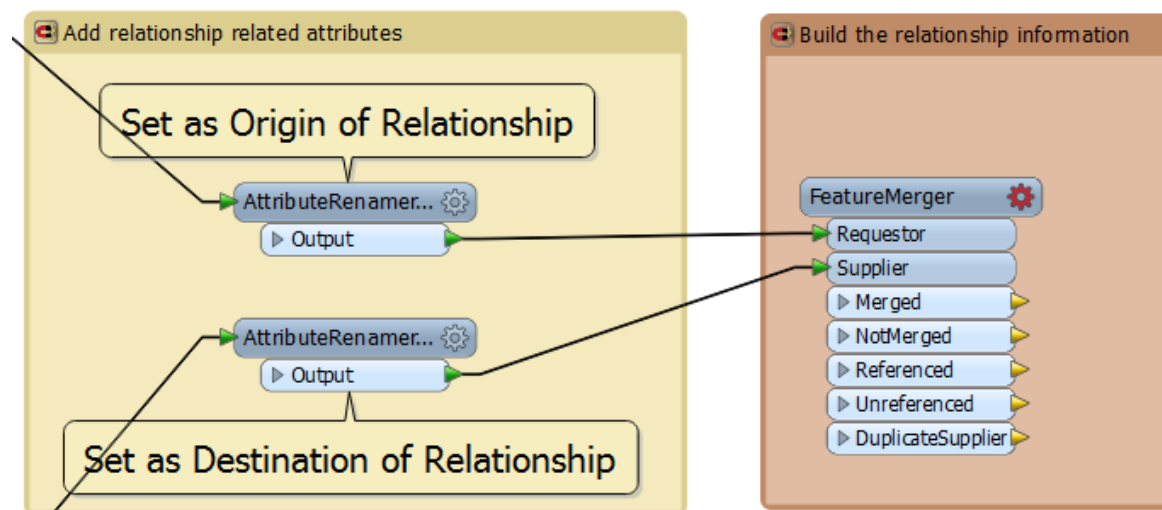
Open the parameters dialog for the FeatureMerger. Set the Requestor attribute to be *Name* and the Supplier attribute to be *path_rootname*:

Requestor	Supplier	Comparison Mode
◇ Name	◇ path_rootname	Automatic

The result of this is that data gets merged where the filename (path_rootname) of the attachment matches the name of the Point of Interest feature (Name).

For example, if I have a point of interest called "Big Tall Statue" (Origin ID = 13) and there is a file named "Big Tall Statue.jpg" (Destination ID = 22) then the result will be a single feature ("Big Tall Statue", Origin ID = 13, Destination ID = 22) that defines a relationship between those features.

When I write this to the Relationship Class it will define the relationship between these features in the PointsOfInterest and PointsOfInterest_ATTACH tables.

ArcGIS Attachments

8) Set Geometry Type
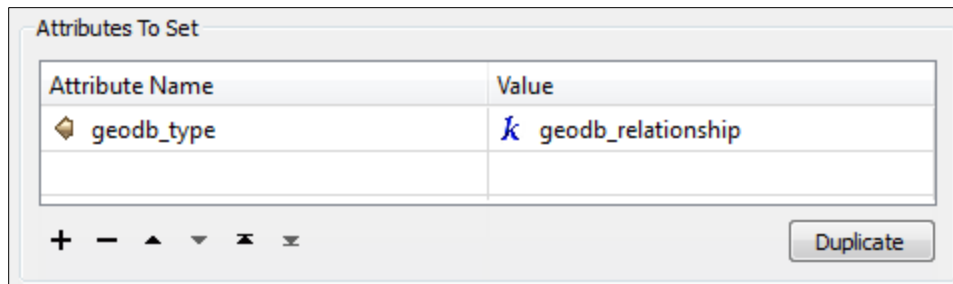
Each feature that gets sent to an FME Writer should have an attribute indicating the geometry type. In most cases you – the user – never need to know about this attribute and don't need to set it.

However, here these features will be currently flagged as point features (i.e. <u>points</u> of interest) and we need to tell FME they are actually non-geometry, relationship features.

So, add an AttributeCreator transformer after the FeatureMerger Merged output port:
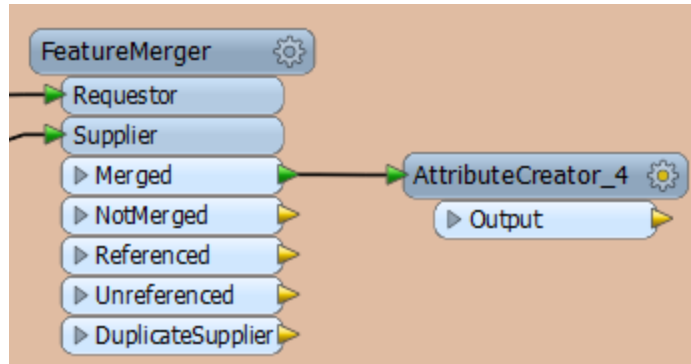
Open the parameters dialog and use it to create a new attribute called *geodb_type*. It should be given a value of *geodb_relationship*

This will tell FME to write these to the Geodatabase as relationship features.

You can now connect this transformer to the Writer feature type:

Output to the relationship table
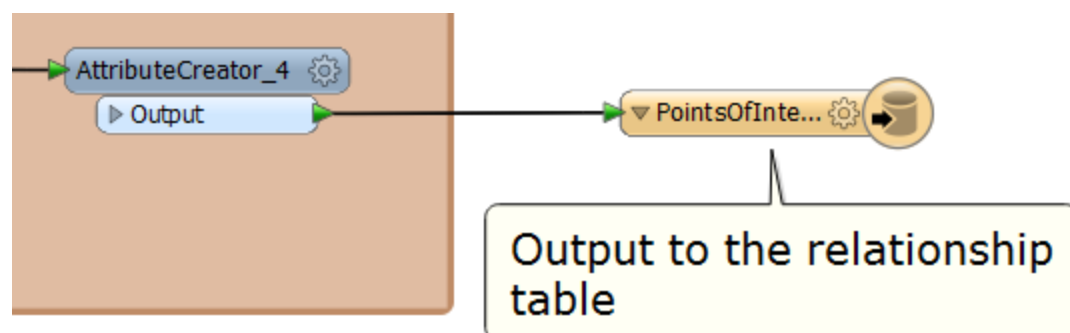
9) Save and Run Workspace

At this point we should check (in the Navigator window) that the transaction type is set 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.

If all of these conditions are met you may save and then run the workspace.

The log window will confirm that the features have been written:

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                          Features Written Summary
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
PointsOfInterest                                                       26
PointsOfInterest__ATTACH                                               17
PointsOfInterest__ATTACHREL                                            14
=========================================================================
Total Features Written                                                57
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

So, basically, we've now written a relationship class to ArcGIS. However, there are one or two items that we need to clean up for this particular exercise to work.

10) Handle Attachments

At the moment we're writing a relationship between features, but the destination features (attachments) are currently just a reference to a file, not the file itself.

We need to use these references to read the attachment file contents. A pre-defined section of workspace will do this for us. Look for the cyan-colored bookmark.
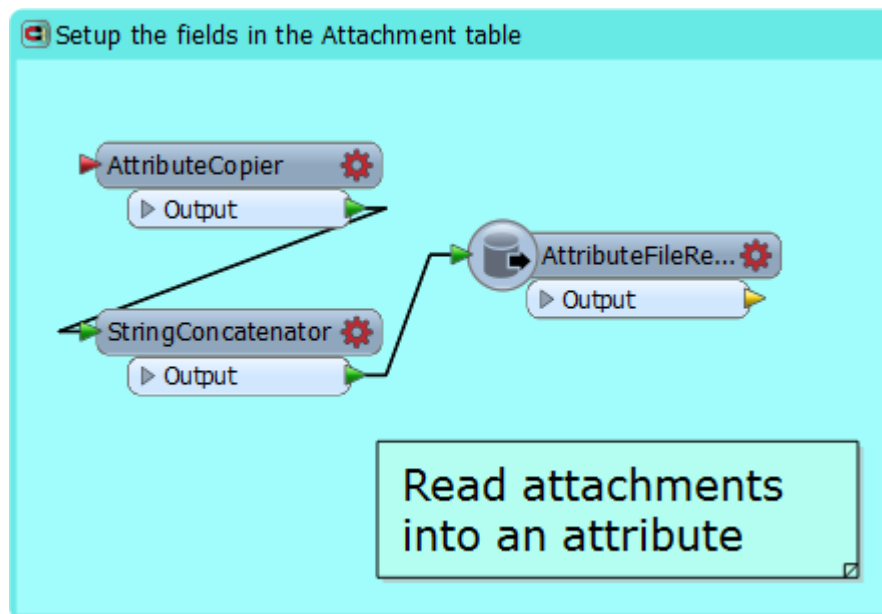
Examine what these transformers are doing. The AttributeCopier is just doing some basic schema mapping, then the StringConcatenator is setting an attribute to tell ArcGIS what the type of file being attached is. Finally, the AttributeFileReader reads the content of the attachment, using the filename obtained from the original Reader.

So, connect the contents of this bookmark into the main workspace, into the connection between the attachments Counter and the Writer feature type (PointsOfInterest__ATTACH). Remove any existing connection to PointsOfInterest__ATTACH.
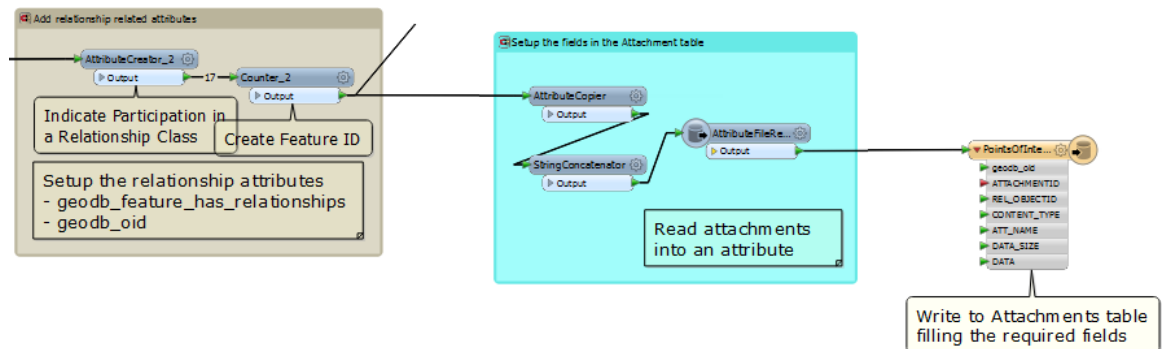
Now save the workspace again and re-run it (make sure ArcGIS is closed).

This time, when you inspect the PointsOfInterest features in ArcGIS (remember they are in the Feature Dataset called City) query them with the HTML PopUp tool.

HTML Popup

Launch an HTML popup by clicking on a feature.

Press F1 for more help.

Firehall.jpg

SHAPE	Point
GUID	<Null>
POIType	<Null>
Street	1455 W 10th Av
City	Vancouver
State	BC
Country	Canada
PostalCode	unknown
Description	Firehall
Phone	<Null>
Website	http://www.vpl.ca/branches/details/firehall_branch
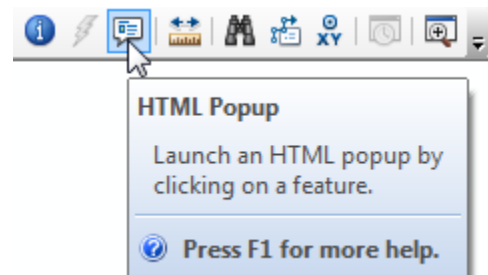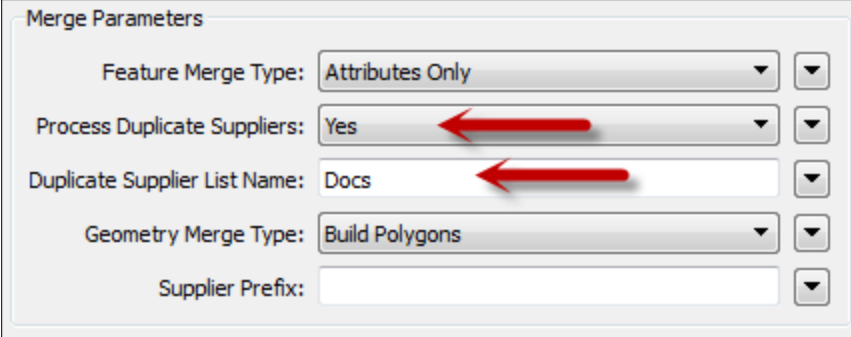
Now some of the features (but not all) will have photographic images attached to them:

11) Handle Multiple Attachments

The other outstanding issue to take care of is the case where there are multiple attachments for a particular point of interest.

Firstly we need to update the FeatureMerger to handle this. So open the FeatureMerger parameters dialog. Change "Process Duplicate Suppliers" to Yes and enter Docs as the list to create.

Now when the workspace is run, multiple files are stored in a list, like so:
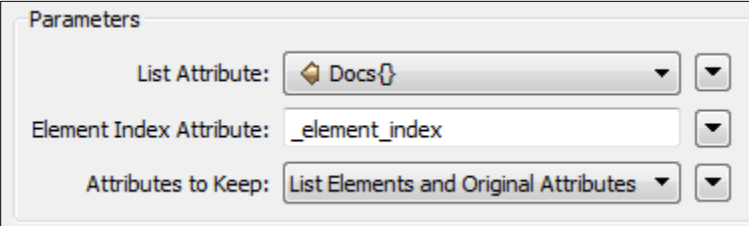
```
Docs{0}.path_filename
Docs{0}.geodb_rel_destination_oid
Docs{1}.path_filename
Docs{1}.geodb_rel_destination_oid
etc.
```

What we need to do is explode this list into individual features – so that there is a relationship record for each attachment – and this is done with a ListExploder transformer.

So, insert a ListExploder transformer between the FeatureMerger and AttributeCreator.
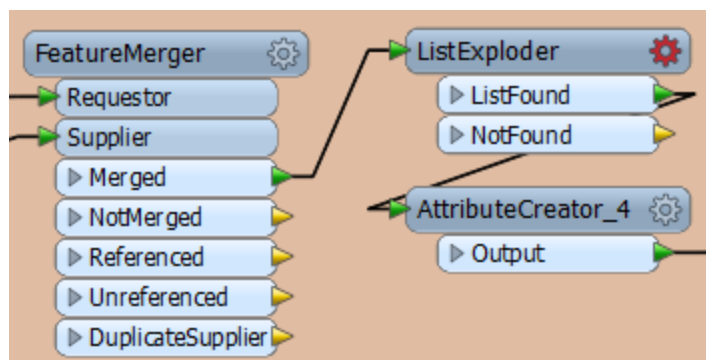
Then open the parameters dialog and select Docs{} as the list to explode:

ArcGIS Attachments

12) Save and Run Workspace
Now save and run the workspace for one final time.

Query the data and some features will have multiple attachments.

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.

Exercise 8: Relationship Classes	
Scenario	FME user; City of Interopolis, Planning Department
Data	Land Parcels, Points of Interest
Overall Goal	Create a relationship between Land Parcels and Points of Interest
Demonstrates	Writing to relationship classes
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise8-Complete.fmw

In this exercise, rather than write features and their relationships at the same time, we'll simply create a relationship between two existing feature classes.

The idea is that we'll fill-in relationships between the points of interest features and land parcels that they fall into. In other words, this is using a spatial join, rather than an attribute-only one.

***NB:** If you didn't complete the previous exercise, you'll need to run this workspace to prepare the Geodatabase for this example: C:\FMEData2015\Workspaces\Esri\Exercise7-Complete.fmw*

1) Start ArcMap
In an ArcMap catalog window, examine tables in the VancouverCity file Geodatabase.

The PointsOfInterest data will appear under the City Feature Dataset and Parcels under the Property Feature Dataset.

There is also a Relationship Class for Parcels_PointsOfInterest

VancouverCity.gdb
- Building
- City
  - Anno_83_84
  - PointsOfInterest ←
  - ZoneNames
  - Zones
- Engineering
- Property
  - AddressPoints
  - Parcels ←
- Sewer_Collection
- Water_Distribution
- Grid
- Neighborhoods
- Parcels_PointsOfInterest ←
- PointsOfInterest__ATTACH
- PointsOfInterest__ATTACHREL
- VancouverLandBoundary
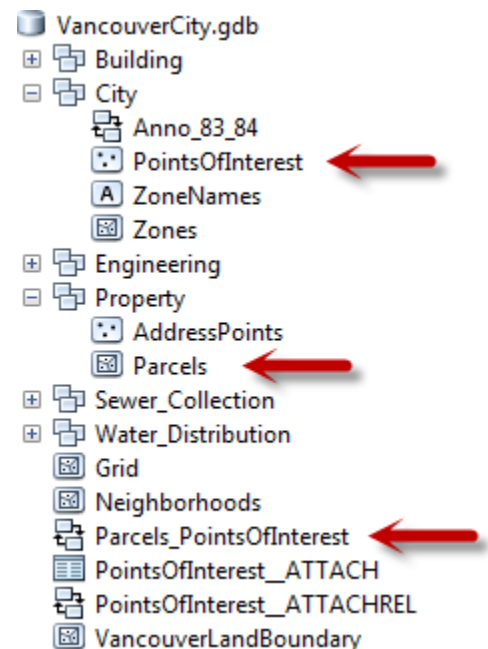
2) Start Workbench

Start Workbench and begin with an empty canvas.

Choose Readers > Add Reader from the menubar and when prompted enter the following:

Reader Format	Esri Geodatabase (File Geodb ArcObjects)
Reader Dataset	C:\FMEData2015\Resources\Esri\VancouverCity.gdb

Select the Parameters button and then click on the Table List […] button in order to choose Parcels and PointsOfInterest from the dialog window.
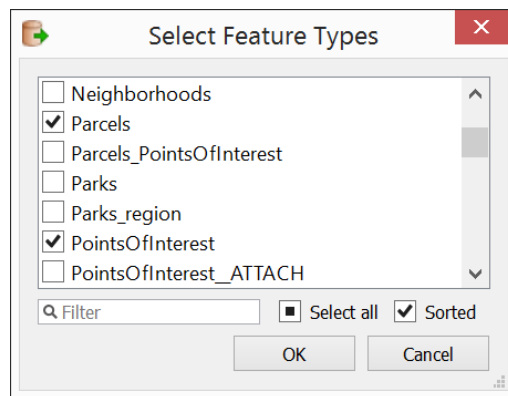
3) Add AttributeRenamer Transformers

As in the previous exercise, one set of data needs a format attribute called *geodb_rel_origin_oid* and the other needs one called *geodb_rel_destination_oid*

So, add two AttributeRenamer transformers and connect one to each Reader feature type:
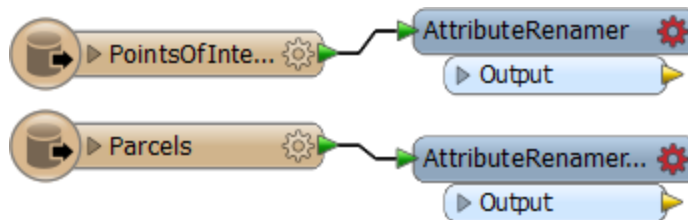
For the <u>Parcels</u>, rename *geodb_oid* to *geodb_rel_origin_oid* (i.e. it is the main feature)*:*

Old Attribute	New Attribute	Default Value
◈ geodb_oid	geodb_rel_origin_oid	

And for <u>PointsOfInterest</u>, rename *geodb_oid* to *geodb_rel_destination_oid*:

Old Attribute	New Attribute	Default Value
◈ geodb_oid	geodb_rel_destination_oid	

ArcGIS Attachments

4) Add SpatialFilter Transformer
Now let's join these features spatially. Add a SpatialFilter transformer. The Parcels should be connected to the Filter port, and the PointsOfInterest to the Candidate:
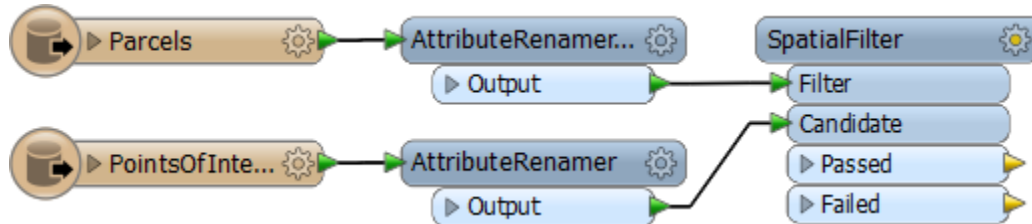
Feel free to reorganize the canvas (as above) to avoid crossing connections.

Now open the parameters dialog for the SpatialFilter.

Set the Tests to Perform parameter to CONTAINS (by default it will be INTERSECTS, but that is not what we want here).

Other parameters should be correct with the default values.

5) Add AttributeCreator
Finally we need to identify the features we've created as being part of a relationship (remember, we aren't writing either Parcels or PointsOfInterest – we're just writing to the relationship table that connects them).

So, add an AttributeCreator transformer connected to the SpatialFilter:Passed port.
Open the parameters dialog and create a new attribute called *geodb_type* set to a value of 'geodb_relationship'.
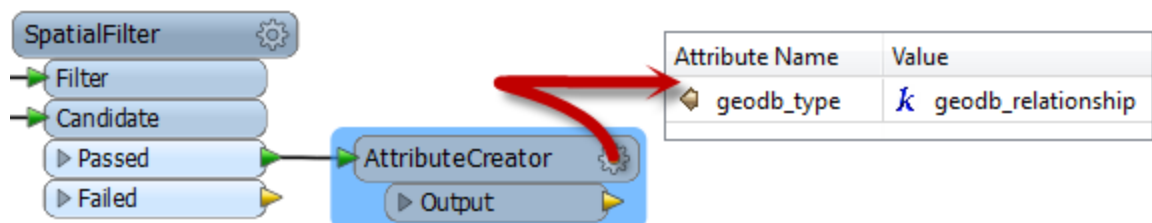
6) Add Writer
On the menubar select Writers > Add Writer and add the output Geodatabase. This already exists – in other words we are updating/adding to it.

Writer Format   Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
Feature Class or Table Definition: Import from Dataset

The Import from Dataset option is selected because the Feature Class already exists in the Geodatabase; it can be imported more easily than being re-created.

7) Import Feature Type
Now let's add the Geodatabase table to this Writer schema.

When prompted set the format and select the file Geodatabase as in step 6 (it may – should – already be set by FME). Click **OK**.
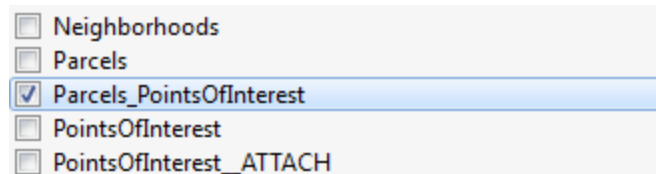
FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *Parcels_PointsOfInterest* only.

When the new feature type is added to the workspace, connect it after the AttributeCreator transformer:
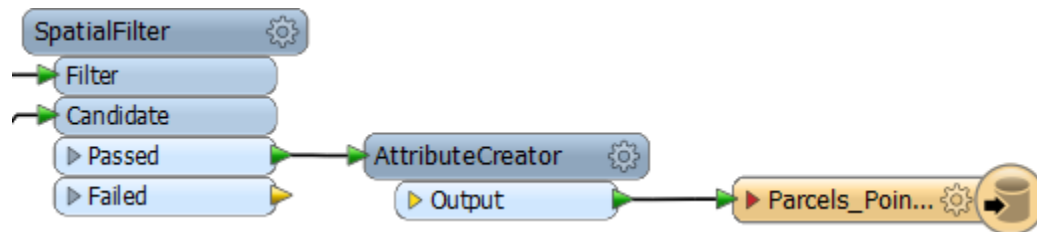
8) Adjust Writer Parameter – Transaction Type
In the Navigator window, set the transaction type to Edit Session. Relationship Classes can only be set in an Edit Session.

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.
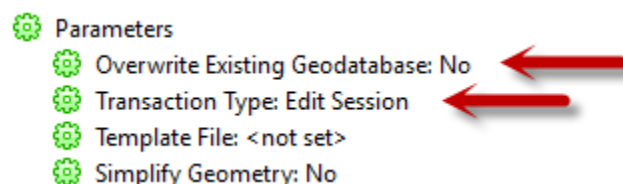
9) Run Workspace

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_rel_origin_oid* and *geodb_rel_destination_oid* is populated and that *geodb_type* is set to 'geodb_relationship'

◢ **Attributes (28)**

fme_geometry (string)	fme_point
fme_type (string)	fme_point
geodb_feature_dataset (string)	City
geodb_feature_is_simple (string)	yes
geodb_rel_destination_oid (32 bit integer)	2272
geodb_rel_origin_oid (32 bit integer)	20
geodb_type (encoded: utf-8)	geodb_relationship
GUID (encoded: utf-16le)	C37BC446-79F4-499E-A7E9-00801AEDCBAB

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.

NB: *If you receive the error:*

ERROR |Failed to retrieve destination row for a relationship feature with OBJECTID 'xxx' from table/feature class 'PointsOfInterest'

Check that you have the origin/destination correct for the Parcels/PointsOfInterest. They may be the wrong way around. Incidentally, although the tables are related using OBJECTID and GUID fields in ArcGIS, in FME we just use the geodb_oid format attribute. The Writer/ArcObjects sorts out the actual relationship.

Updating a File Geodatabase

File Geodatabases now support Update and Delete functionality.

Exercise 9: Updating Geodatabases	
Scenario	FME user; City of Interopolis, Planning Department
Data	Esri File Geodatabase, Updated Drinking Fountains (Esri Shape)
Overall Goal	Update a Geodatabase feature class from an updated GIS dataset
Demonstrates	Geodatabase Updates
Starting Workspace	None
Finished Workspace	C:\FMEData2015\Workspaces\Esri\Exercise9-Complete.fmw

The team responsible for managing municipal data have released updates to the Drinking Fountains table.

Three drinking fountains in particular were in the wrong position and have been moved.

However, the new data has been released as an "updates only" dataset and must be applied as an update to the existing drinking fountain table.

Since only individual features are being updated (not the whole table) this can be achieved using format attributes.
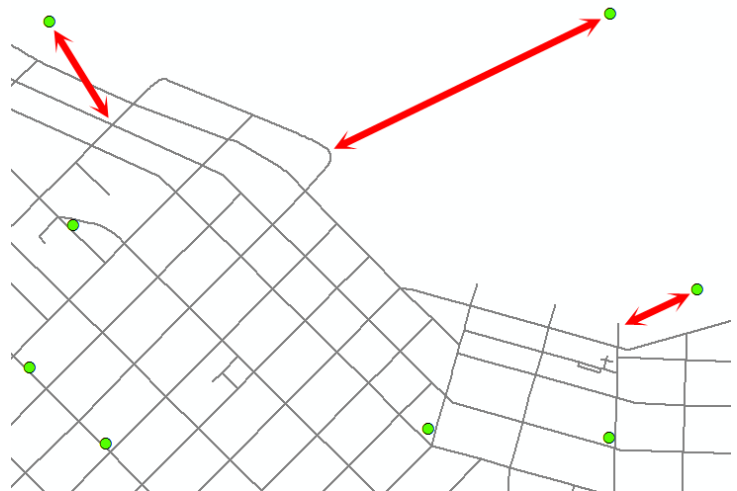
1) Start Workbench
Start Workbench and begin with an empty canvas.

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

Reader Format   Esri Shape
Reader Dataset   *C:\FMEData2015\Data\CommunityMapping\DrinkingFountains.shp*

2) 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	Esri Geodatabase (File Geodb API)
Writer Dataset	*C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
Feature Class or Table Definition: Import from Dataset	

The Import from Dataset option is selected because the Feature Class already exists in the Geodatabase; it can be imported more easily than being re-created.

3) Import Feature Type
Now let's add the Geodatabase table to this Writer schema.

When prompted set the format and select the file Geodatabase as in step 2 (it may – should – already be set by FME). Click **OK**.
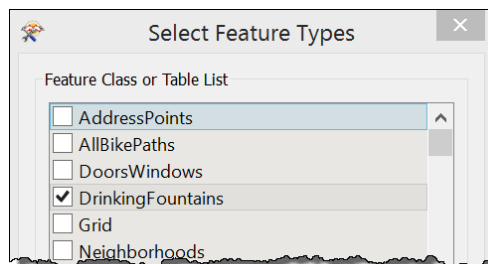
FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *DrinkingFountains* only.

Connect the reader to your Geodatabase API writer.

4) Edit Writer Format Parameters
Double click on the writer feature type to enter the Feature Type Properties dialog. Click the Format Parameters tab and set the following parameters:

Feature Operation:	Update
Table Handling:	Create If Needed
Update Spatial Column:	Yes

Select the WHERE Clause radial button and click the [...] button to open the Where Clause builder.

5) Set Where Clause:
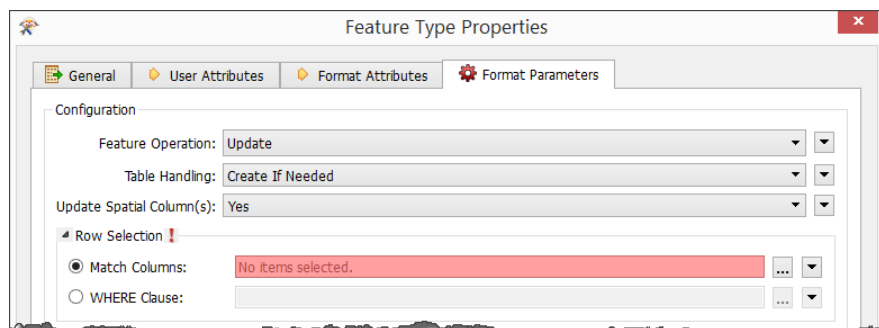The Where Clause builder dialog will be used to construct a 'where' clause for the update. Enter the following in the editor window:

FountainID=@Value(FountainID)
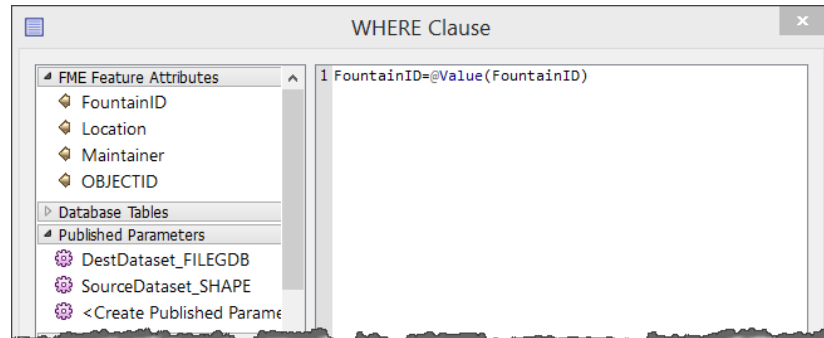
Click OK twice.

…this is basically a 'where' clause that means *"where the field FountainID matches the value of the incoming attribute FountainID"*. Quotes are not required around the value because it is a numeric field.

6) Save and Run Workspace
Save and Run the workspace and inspect the DrinkingFountains dataset.
You should now find that the erroneous drinking fountains have been moved into the correct position.

If you can't find the new drinking fountains, they should be located close to the following:

Fountain 24 X: 491772   Fountain 49 X: 491288   Fountain 69 X: 492418
         Y: 5459404   Y: 5459477   Y: 5459028

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.

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.

ArcSDEQuerier Parameters

Search Method: SDE_NONE

Help

SDE_NONE
SDE_ENVELOPE
SDE_COMMON_POINT
SDE_LINE_CROSS
SDE_COMMON_LINE
SDE_CP_OR_LC
SDE_AI_OR_ET
SDE_AREA_INTERSECT
SDE_AI_NO_ET
SDE_CONTAINED_IN
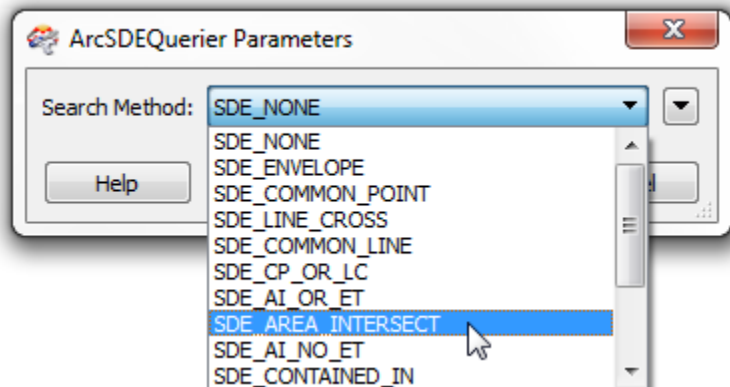
FeatureReader

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.
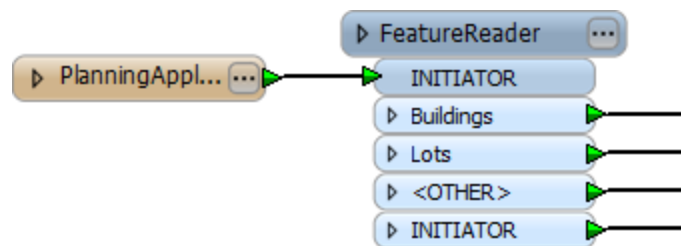
For example, here a user reads a list of planning applications stored in a text/CSV dataset.

Each record has an ID that is used in a Where clause in the *FeatureReader* to retrieve the appropriate features from a Geodatabase.

Example 10: Database Transformers	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Grid, Addresses
Overall Goal	To read all addresses within a user-defined grid square
Demonstrates	Database transformers, particularly the FeatureReader
Starting Workspace	None
Finished Workspaces	C:\FMEData2015\Workspaces\Esri\Exercise10-Complete.fmw C:\FMEData2015\Workspaces\Esri\Exercise10-Advanced-Complete.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, which is already stored in the VancouverCity Geodatabase.

Use Readers > Add Reader to add a reader as follows:

Source Format   Esri Geodatabase (File Geodatabase ArcObjects)
Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*

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

2) Add User Parameter
A published parameter will allow the user to select which grid square to read.

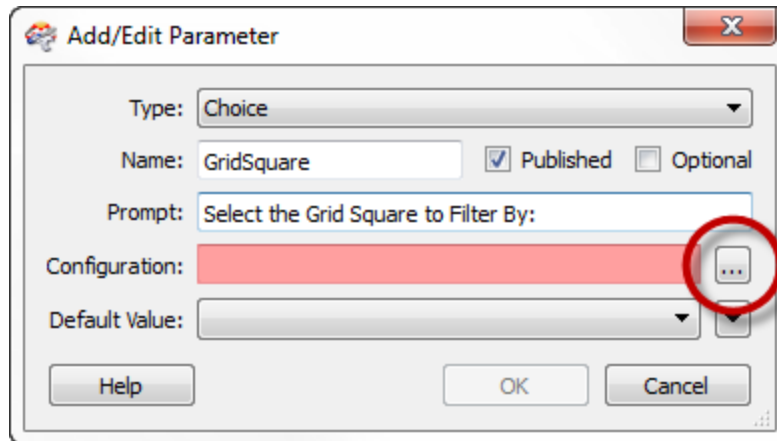In the Navigator window, locate and right-click User Parameters. Choose Add Parameter.
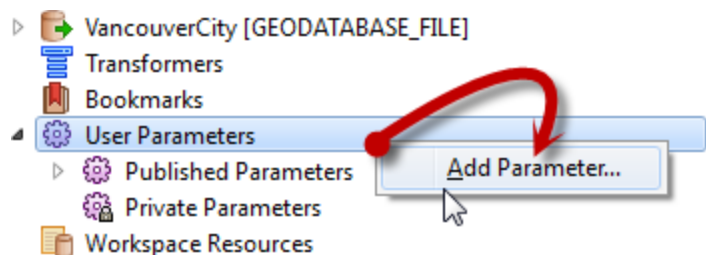
Define the new parameter as follows:

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

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 grid square values.

In the Import Wizard, set the format to File Geodatabase and select the VancouverCity Geodatabase. Select Grid as the feature type to read. Select *FacetText* as the attribute to read.

FME will now scan the Grid table for a list of tile names (there should be 117 values).

Click **Import** to complete the process. The choice list will now look like this:
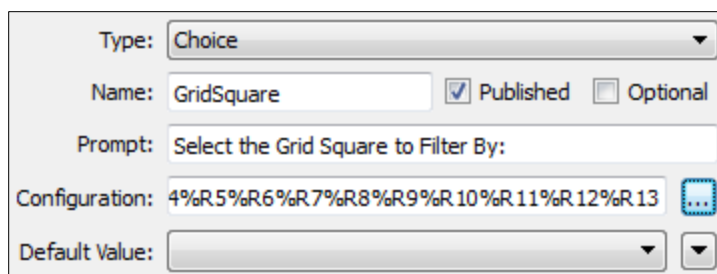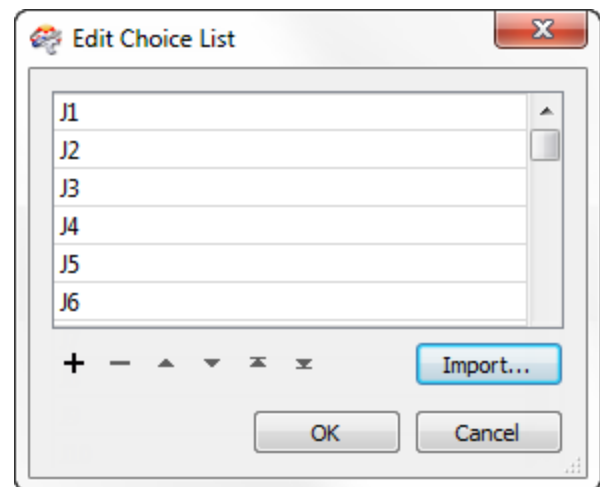
Click OK to close all dialogs and create the new User Parameter.

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 from the drop-down menu and then FacetText as the attribute to be tested.

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

In the Right Value field choose User Parameter from the drop-down menu and then GridSquare as the (newly created) user parameter to be tested:
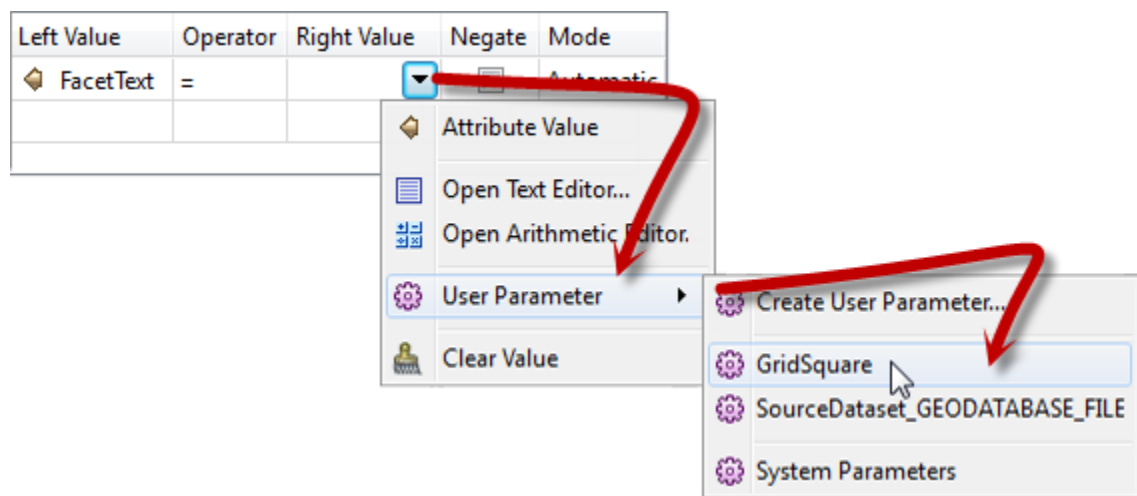
5) Test Workspace
At this point you may wish to test this part of the workspace.

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.
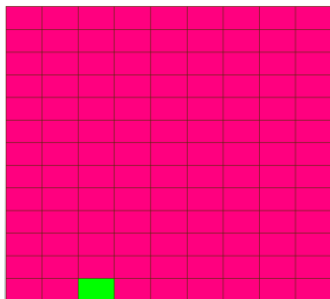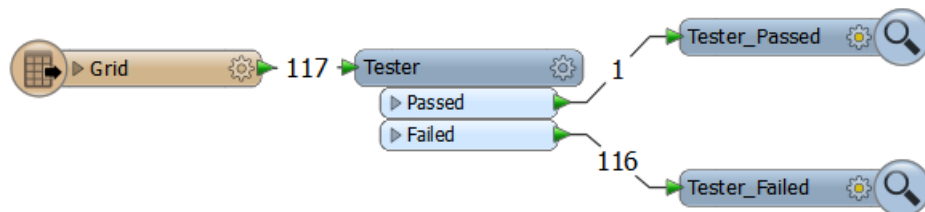
The result should be one grid square – chosen by the user – passing the Tester and the rest failing.

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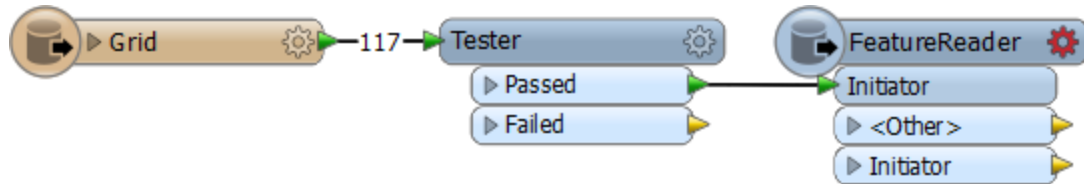
Once we have isolated the required grid feature, we can use it in a spatial query using the FeatureReader transformer.

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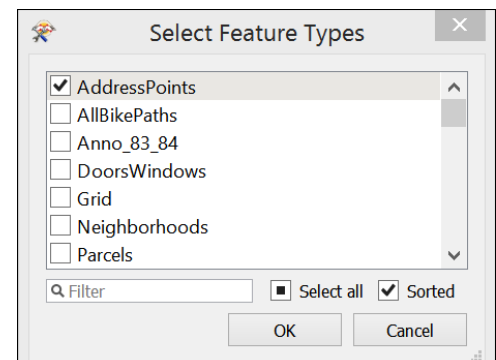

Open the FeatureReader parameters dialog.

The Reader portion of the dialog asks for the location of the data to query. This is the location of the address data:

Source Format   Esri Geodatabase (File Geodatabase ArcObjects)
Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*

Next, under the Feature Types to Read section of the dialog, ensure that the Feature types radial button is selected and then click on the [...] button.

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

The WHERE Clause can be left empty (we already selected AddressPoints; however, we do need to select a spatial interaction). This is where we can define a spatial query between the Initiator (City Grid) and the table being queried (Address Points).

Select Contains from the Spatial Filter drop down menu. When complete, your FeatureReader parameters dialog should look as follows:

FeatureReader Parameters

Transformer

Transformer Name: FeatureReader_2

Reader

Format: Esri Geodatabase (File Geodb ArcObjects)

Dataset: C:\FMEData2015\Resources\Esri\VancouverCity.gdb

Parameters... Coord. System: Read from source

Feature Types to Read

○ Feature Types: AddressPoints

○ From Attribute:

◢ Constraints

WHERE Clause:

Spatial Filter: Contains

◢ Output Ports

◉ From Named Feature Types To Read
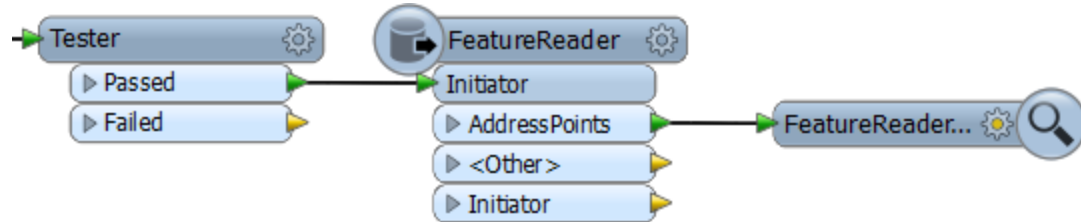
○ Single Output Port

○ Output Ports:

▷ Advanced

Help Defaults ▾ OK Cancel
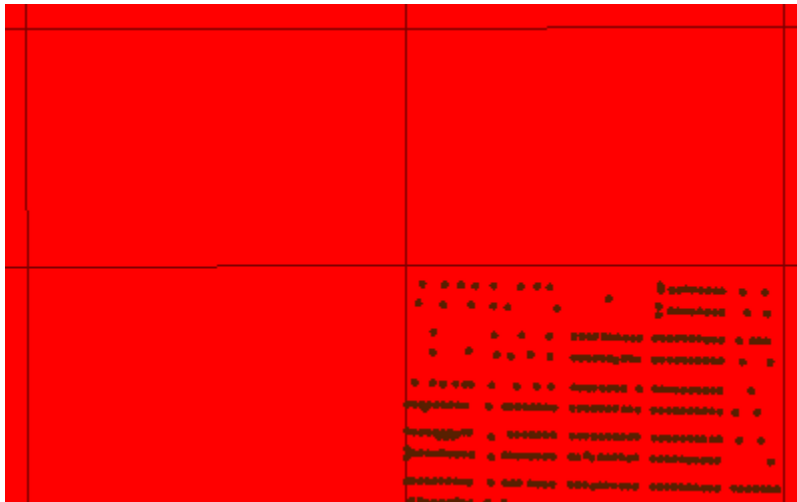
8) Run Workspace

Connect an Inspector transformer to the AddressPoints 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 (L13 is a good choice for testing) 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 status field whose value is 'Retired'– indicating an address that is no longer valid. You can see this in the Data Inspector table view:

12		Vancouver	BC	V6R 1K8	Canada	Pending	20130707	GIS Technician
13		Vancouver	BC	V6R 0M9	Canada	Current	20130707	GIS Technician
14		Vancouver	BC	V6R 1M8	Canada	Pending	20130707	GIS Technician
15		Vancouver	BC	V6R 4V5	Canada	Other	20130707	GIS Technician
16		Vancouver	BC	V6R 2O7	Canada	Retired	20130707	GIS Technician
17		Vancouver	BC	V6K 6H3	Canada	Current	20130707	GIS Technician
18	..	Vancouver	BC	V6R 7U2	Canada	Current	20130707	GIS Technician

Use the FeatureReader WHERE clause to filter out addresses with this status.

You may 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. For example, square R8 should have 124 addresses in total, or 109 excluding retired addresses.

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 *PointOnAreaOverlayer* transformer? Why?

Database Transformers

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 a Geodatabase Reader parameter – *Feature Read Mode* – to 'Metadata'.
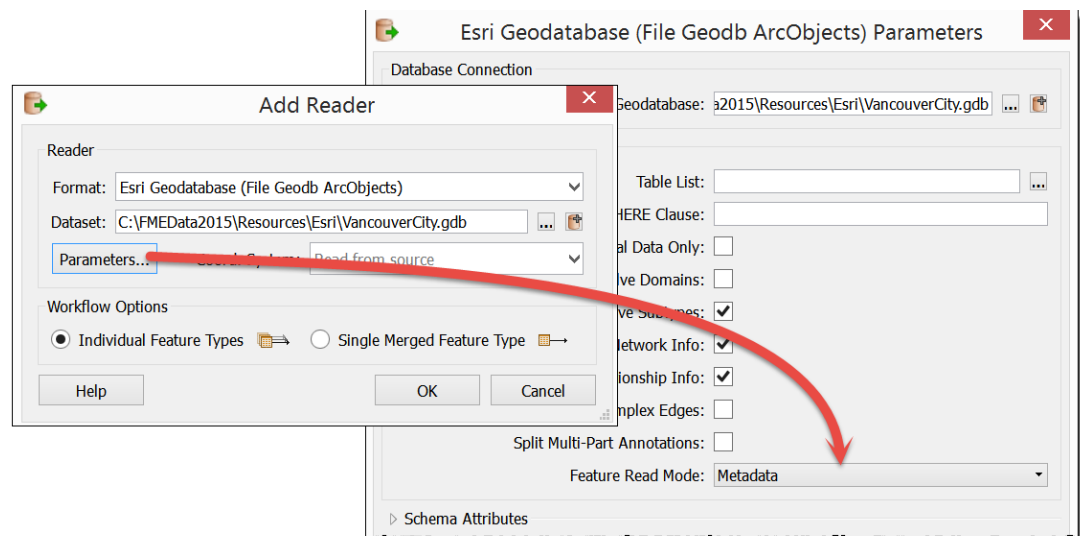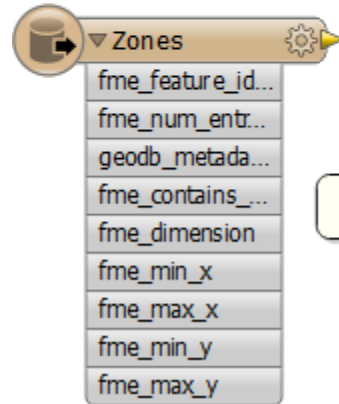

Be aware that this option is only available at the time the Reader is added, or the workspace generated. You can't (as in previous versions) add a Reader and then change it from reading features to reading metadata.

▽ Zones
| geodb_oid |
| OBJECTID |
| ZoneName |
| ZoneCategory |
| SHAPE_Length |
| SHAPE_Area |
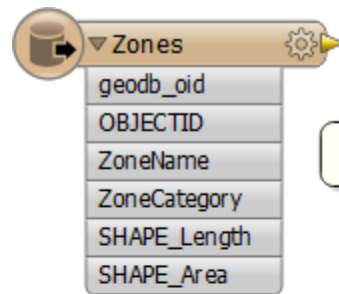
Reading Features

When reading metadata, the Reader feature types have a completely different schema than when reading features.

Instead of listing user attributes, a metadata feature type lists a number of metadata fields, including *geodb_metadata_string* – an XML string containing the Geodatabase metadata.

Other format attributes store information such as dimension, spatial column, geometry, etc.

▽ Zones
| fme_feature_id... |
| fme_num_entr... |
| geodb_metada... |
| fme_contains_... |
| fme_dimension |
| fme_min_x |
| fme_max_x |
| fme_min_y |
| fme_max_y |

Reading Metadata

In metadata mode, only a single feature is output from the feature type, 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.

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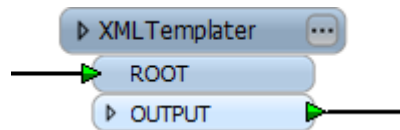
To find out more information about Esri metadata in FME, browse the information in the Readers and Writers Manual under the section *Geodatabase > Feature Representation > Metadata*

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.

Exercise 11: 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\Esri\Exercise11-Complete.fmw C:\FMEData\Workspaces\Esri\Exercise11-Complete-Advanced.fmw

The city has a Shape dataset of Bike Routes. The task is to import the Bike Route data into a Geodatabase, at the same time writing metadata.

1) Inspect Data
Let's take a look at the source and destination datasets' existing metadata.

In the folder C:\FMEData2015\Data\Transportation\Cycling you'll find XML files that represent metadata for various bike path datasets. Open one of these files (in an XML editor or web browser) to view the metadata contents.

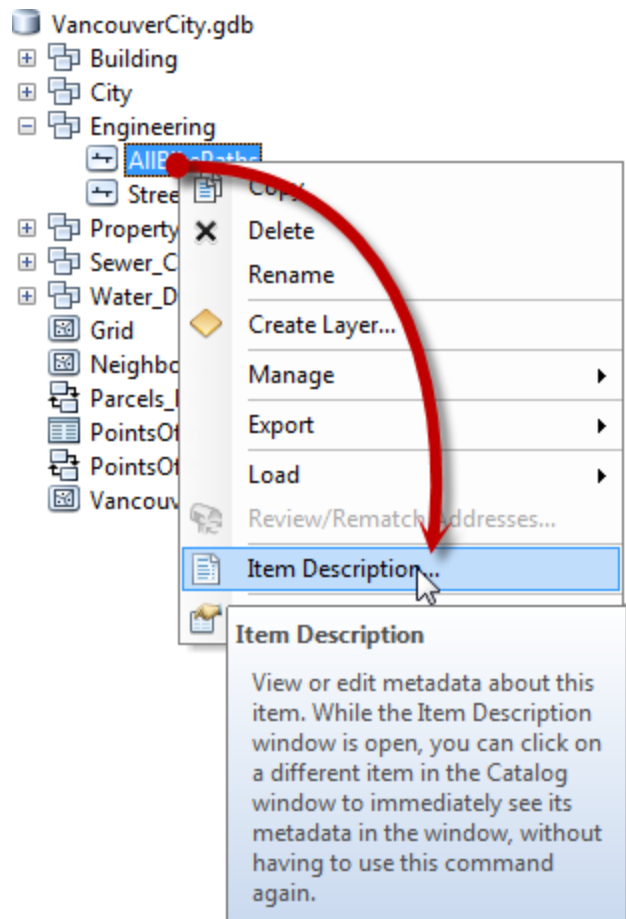To view the metadata in ArcMap requires a change in setting.

Set Customize > ArcMap Options > Metadata > Metadata Style to *ISO 19139 Metadata*.

In a catalog window in ArcGIS, browse to: *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*

Locate the "AllBikePaths" feature class (under Engineering), right-click it, and choose Item Description.

This opens a dialog in which the existing metadata is displayed.

Currently the AllBikePaths class is empty, so the task here is to insert data and update the metadata at the same time.

Item Description

View or edit metadata about this item. While the Item Description window is open, you can click on a different item in the Catalog window to immediately see its metadata in the window, without having to use this command again.

2) Start Workbench
Start FME Workbench. Add a Reader to read one of the Bike Route Shape files

Reader Format   Esri Shape
Reader Dataset   C:\FMEData2015\Data\Transportation\Cycling\BikePaths_L.shp

3) Add Writer
On the menubar select Writers > Add Writer and add the output Geodatabase.

Writer Format   Esri Geodatabase (File Geodb ArcObjects)
Writer Dataset   *C:\FMEData2015\Resources\Esri\VancouverCity.gdb*
Feature Class or Table Definition: Import from Dataset

The Import from Dataset option is selected because the Feature Classes already exist in the Geodatabase; they can be imported more easily than being re-created.

4) Import Feature Type
When prompted set the format and select the file Geodatabase as in step 3 (it may – should – already be set by FME). Click **OK**.
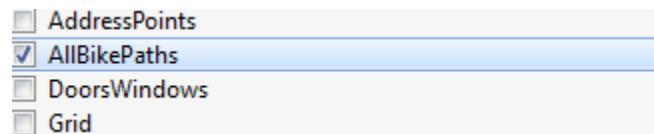
FME will now scan the Geodatabase to confirm what tables exist. When prompted with a list of classes, select *AllBikePaths* only.

Now we have both a Reader and a Writer in the Navigator window and, on the canvas, a Feature Type for each of these.
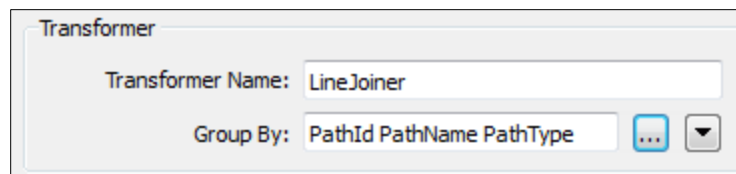
5) Add LineJoiner Transformer
Before getting on to the metadata, there are a couple of actions we need to perform on the bike route linework.


Firstly add a LineJoiner transformer.
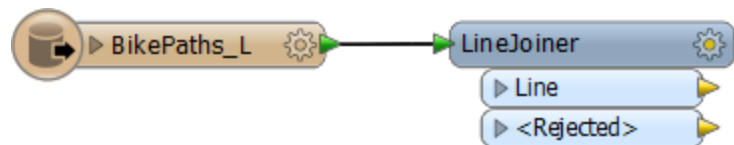
Open the parameters dialog for the LineJoiner. Click the [...] button for the Group-By parameter and choose PathId, PathName, and PathType as the attributes to group by.

In other words, the transformer will connect line features together where they have the same ID, Name, and Type.

Metadata   *Page 91*

6) Add LengthCalculator Transformer
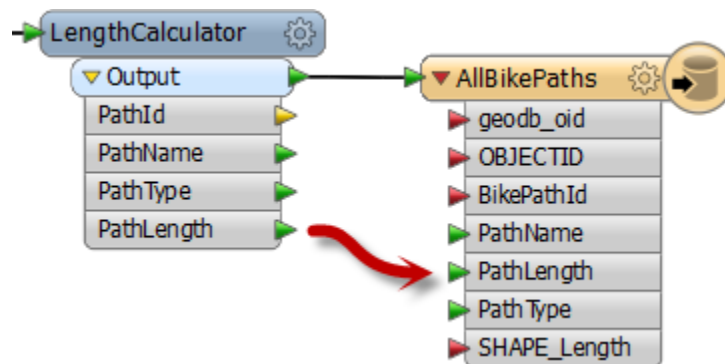
Now add a LengthCalculator transformer.
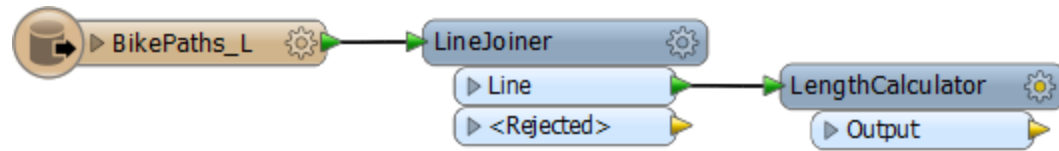
Open the parameters dialog and ensure the transformer is creating an attribute called PathLength:


If you now connect this transformer to the Writer feature type, you'll see that PathLength matches the schema and is automatically connected:

7) Add Creator Transformer

Now let's deal with the metadata.

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.
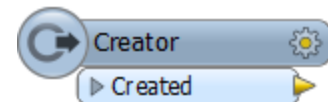
8) Add AttributeFileReader Transformer

Add an *AttributeFileReader* transformer after the *Creator*.
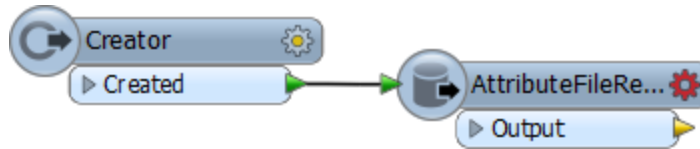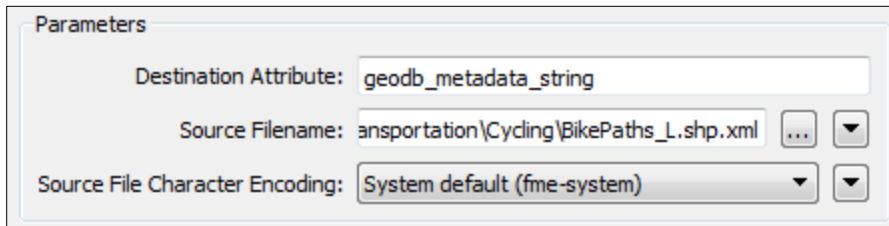

An *AttributeFileReader* reads the contents of a file and stores it in an attribute. In this case we'll use it to read the contents of the existing metadata file.

Set the destination attribute to be called *geodb_metadata_string*

Open the parameters dialog. Set the source filename to be the XML Metadata document associated with the source shape (*C:\FMEData2015\Data\Transportation\Cycling\BikePaths_L.shp.xml*)

Set the Source File Character Encoding to be System Default (when you opened the file in an XML editor did you notice it had no encoding set?)

9) Add AttributeCreator

Finally 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*

10) Map Schema
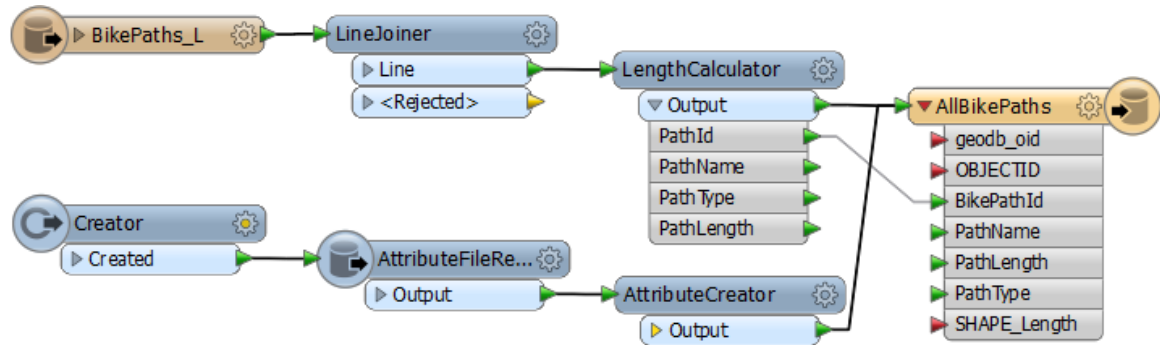
Connect the *AttributeCreator* transformer to the Writer feature type. Also drag an attribute connection from PathId (on the LengthCalculator) to BikePathId (on the Writer) to map that.

At this point the workspace will look something like this


11) Save and Run Workspace

Check the Writer and Feature Type parameters are correct for the workspace (i.e. that we aren't overwriting the Geodatabase and we aren't dropping the table first).

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

You'll see that, whereas the metadata used to say "Bike paths in the City of Vancouver" credited to "Robyn", the metadata now reports "Data for Bike Paths that are over xx km long" and is credited to "City of Vancouver Open Data Portal".

Bike Paths Long
File Geodatabase Feature Class

Thumbnail Not Available
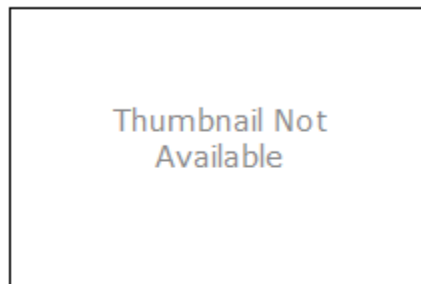
Tags
Long Bike paths

Summary
Data for Bike Paths that are over xx km long

Description
There is no description for this item.

Credits
City of Vancouver Open Data Portal

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 change "Bike Paths that are over xx km long" to actually state the correct distance.

1) Inspect XML

If you view the source XML file once more, you'll notice that the summary information is held in a tag labelled <idPurp>

```
</dataExt>
<idPurp>Data for Bike Paths that are over xx km long</idPurp>
<idCredit>City of Vancouver Open Data Portal</idCredit>
```

In actual fact, the full path to this tag is: `/metadata/dataIdInfo/idPurp`

2) 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.
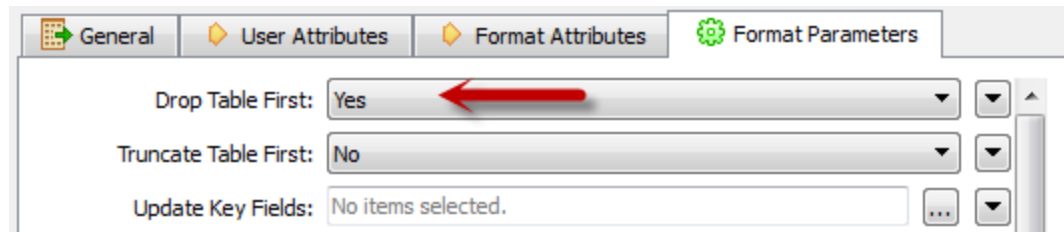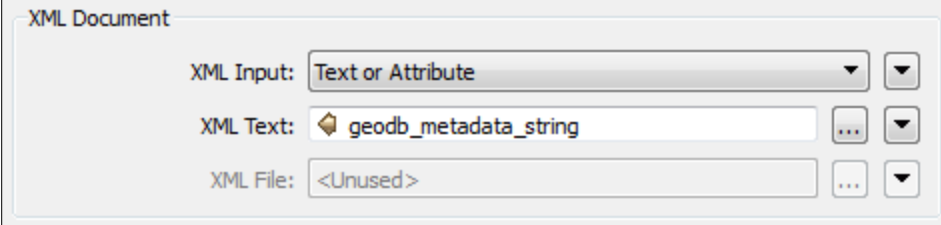

3) Add XMLUpdater Transformer

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.
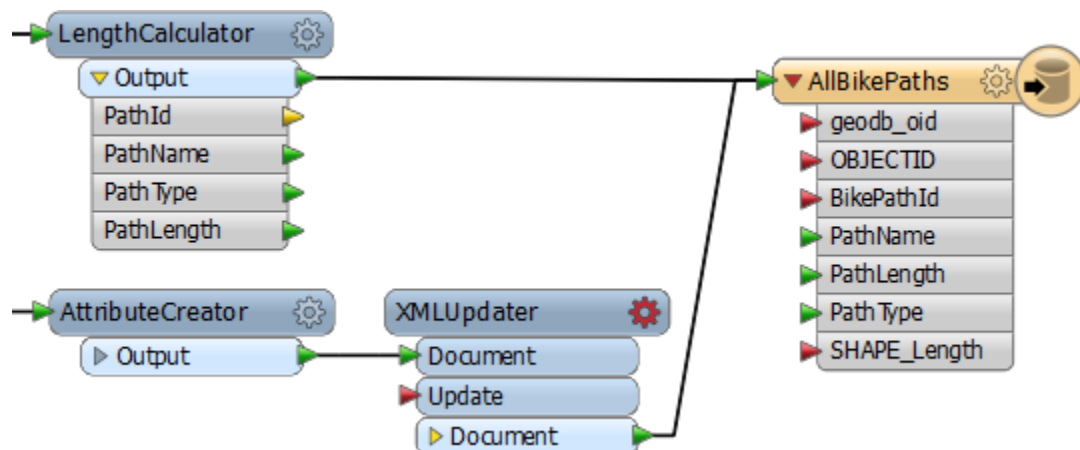
Metadata

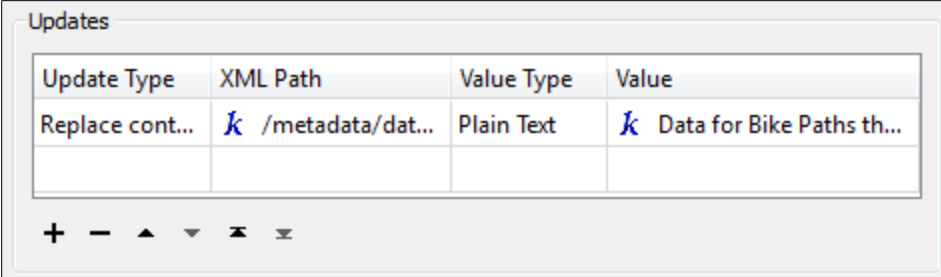Open the parameters dialog for the XMLUpdater. Set:

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

Now for the update part. Set:

Update Type   Replace Contents
XML Path   /metadata/dataIdInfo/idPurp
Value Type   Plain Text
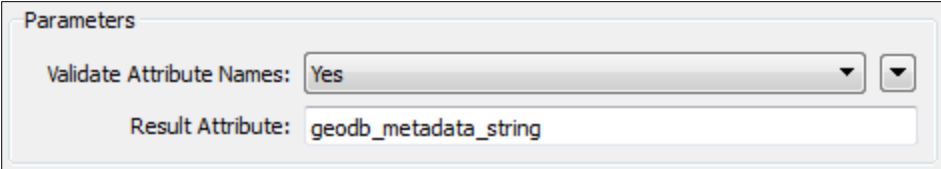Value   Data for Bike Paths that are over 15 km long

Finally set the output parameters:

Validate Attribute Names   Yes
Result Attribute   geodb_metadata_string

Basically this will take the metadata string, replace the path we chose with the new value, and write back to the same attribute.

4) Connect Update Port

To trigger the *XMLUpdater* transformer, we require an update feature – even a null one – to force an update.

So, make a second connection from the existing Creator transformer, and connect it to the XMLUpdater:Update port:
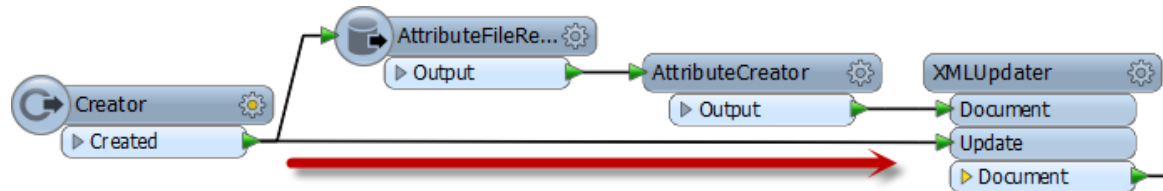
This will provide the update feature that triggers the transformer.

5) 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.

Summary
Data for Bike Paths that are over 15 km long

Description
There is no description for this item.

Credits
City of Vancouver Open Data Portal

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.

- FME can read and write **Subtypes** and **Domains**.

- 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 Subtypes and Domains

- 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, and create a Spatial ETL model in ArcGIS.

Appendix A – Spatial ETL Tools

Spatial ETL Tools are the ArcGIS equivalent to an FME workspace. They can be created and used as follows.
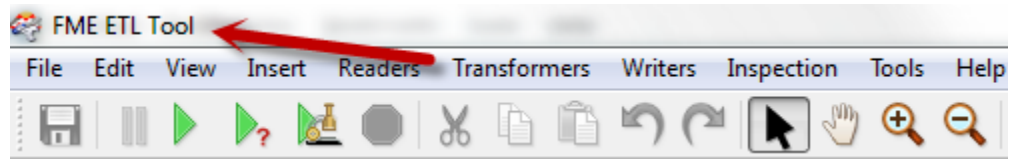
In ArcMap, create a toolbox, right-click on it, and choose the option FME ETL Tool (with the Data Interoperability Extension it will be Interoperability ETL Tool)
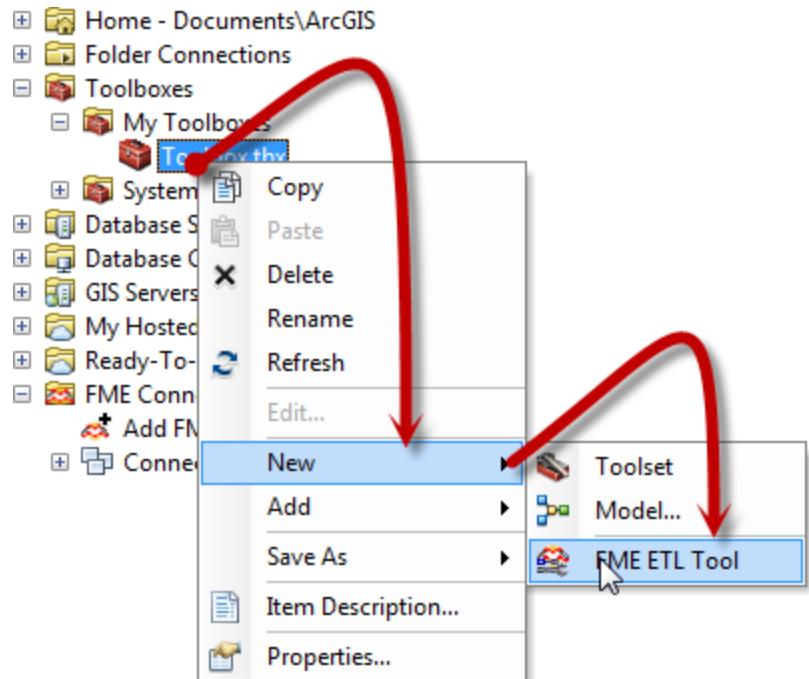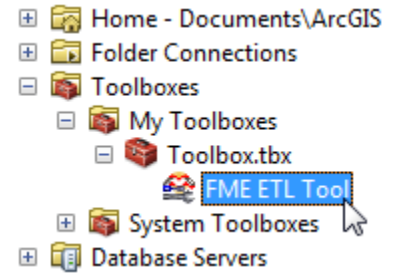

This action opens up a dialog window in which a new workspace can be defined:
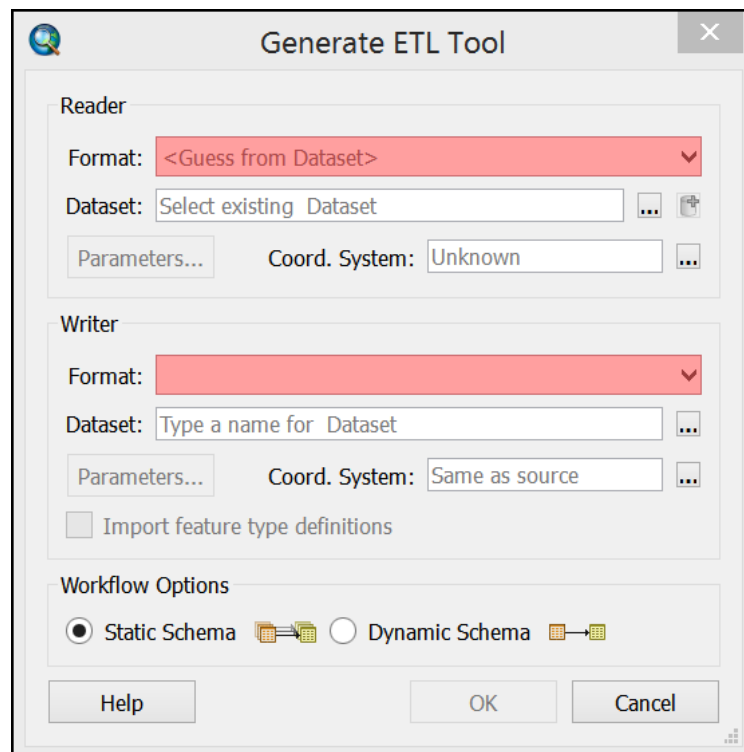
This creates a workspace and opens it up in an ArcGIS-specific version of Workbench:

The ETL Tool can be run inside of this ArcGIS-Workbench, or it can be run by double-clicking on the tool itself in a catalog window:

It's even possible to include such an ETL Tool as a step inside a ModelBuilder model:

Appendix B – 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/kgv

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 C – 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 > Translation from the Workbench menubar.
Check the box for Custom Python Interpreter and choose a Python DLL of the correct version.
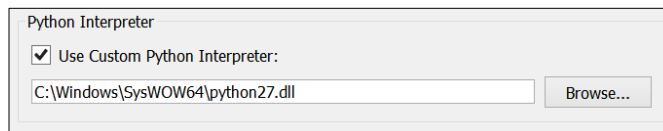
There are various locations the DLL might be found, depending on the operating system used. The ArcGIS 10.3 Desktop installer puts its Python DLL into the windows folder by default.

On our systems we also have to set PYTHONPATH to:

```
C:\Program Files\ArcGIS\Desktop10.3\bin;C:\Program
Files\ArcGIS\Desktop10.3\arcpy;C:\Program
Files\ArcGIS\Desktop10.3\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 D – 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 E – 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 Esri Enterprise Geodatabase.

Connecting to an Esri Enterprise Geodatabase (ArcSDE Geodb) is done through a Connection File. The Connection File is probably the simplest method to use and is the only method available for ArcSDE Geodb connections.

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

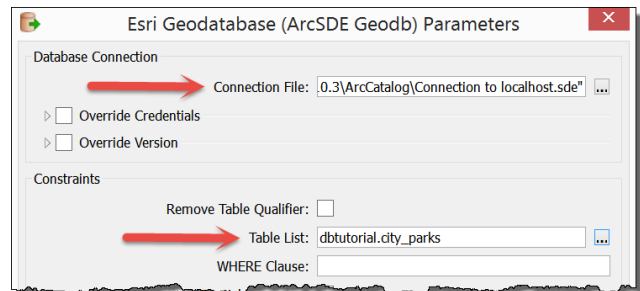**Reader Format:**   Esri Geodatabase (ArcSDE Geodb)

Then click the Parameters… button to open the parameters dialog.

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>\AppData\Roaming\ESRI\Desktop10.3\ArcCatalog\ConnectionFile.sde

Simply select the Connection File […] and then use the file browser provided to select the connection file to be used. Then select the Table List […]

However, it should be noted that connecting to the Esri ArcSDE reader/writer can be done either through a set of parameters, OS Authentication, or Populate from Connection File.
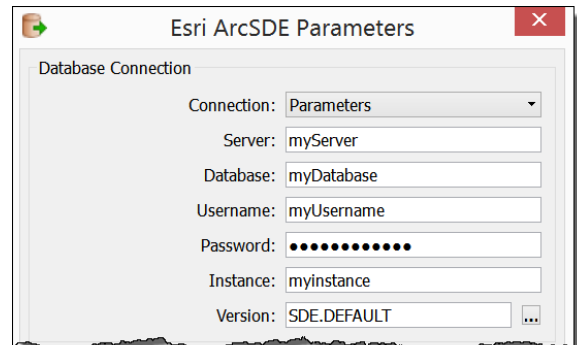
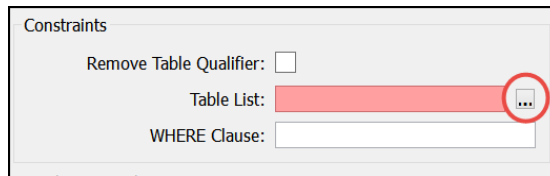**Reader Format:**   Esri ArcSDE

Connection Parameters
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.
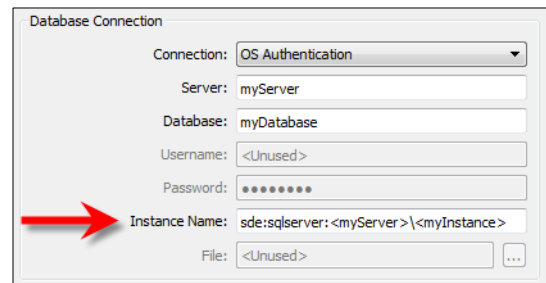
Click on the browse button to the right of the Table List parameter to select specific tables
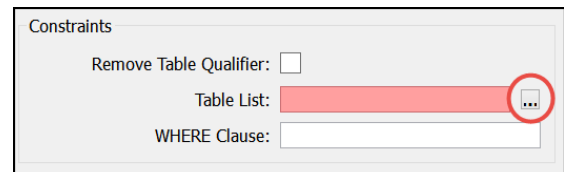

OS Authentication

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 database from the user's OS login information.

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


Populate from Connection File

The Populate from Connection file option is a similar process to connecting through parameters.

The main difference is that when the Populate from Connection File option is selected, a Connection File is selected which is then used to populate the ArcSDE parameters dialog.
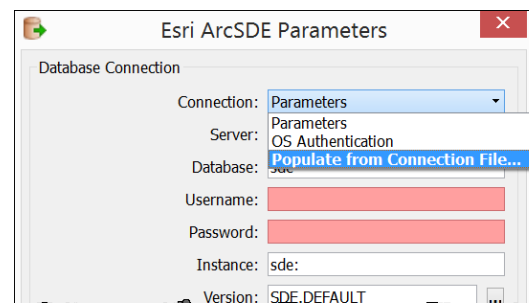
However, the password associated with the database connection and a table from the Table List will still need to be provided by the user.
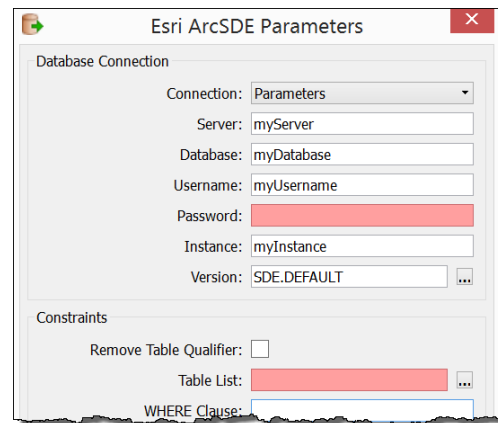
In all cases, the key connection parameter is Instance Name. For older Geodatabases it might be port:5151, where the application server is running. However, for new installations, it might need to be the direct connect string, i.e. sde:<dbclient>:<host> etc.

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
```