Translation Components .................................................................................................................. 2

Key Components ............................................................................................................................... 2

Managing Components .................................................................................................................... 6
  Component Management Tools ........................................................................................................ 6
  Create Workspace ............................................................................................................................. 7
  Add Reader/Writer ........................................................................................................................... 8
  Add Feature Types ............................................................................................................................ 9
  Import Feature Types ....................................................................................................................... 14
  Remove Reader/Writer ..................................................................................................................... 15
  Remove Feature Types ..................................................................................................................... 15
  Update Feature Types .................................................................................................................... 16
  Move Feature Types ....................................................................................................................... 16

Controlling Translations .................................................................................................................. 19
  Parameters ......................................................................................................................................... 19
  Top-Down Effect .............................................................................................................................. 20
  Priority ............................................................................................................................................. 20
  Locating Parameters ......................................................................................................................... 20

Workspace Parameters .................................................................................................................... 22
  Basic Workspace Parameters .......................................................................................................... 22
  Advanced Workspace Parameters .................................................................................................... 23

Reader and Writer Parameters ........................................................................................................ 25
  Reader Parameters ........................................................................................................................... 25
  Writer Parameters ............................................................................................................................. 25

Feature Type Parameters .............................................................................................................. 29
  Feature Type Parameters ................................................................................................................. 29

Format Attributes ........................................................................................................................... 31
  Format Attributes ............................................................................................................................ 31
  FME Attributes ............................................................................................................................... 32
  Controlling Features with Format Attributes .................................................................................. 33

Parameter Documentation ............................................................................................................. 42
  FME Readers and Writers Reference Manual ................................................................................. 42

Published Parameters .................................................................................................................... 45
  Publishing Parameters .................................................................................................................... 46
  Using Published Parameters ............................................................................................................ 48

Module Review ............................................................................................................................. 50
  What You Should Have Learned from this Session ........................................................................ 50
Translation Components

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

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

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

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

Key Components

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

- Workspace
- Readers and Writers
- Feature Types
- Features

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

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

This section covers “official” FME components only.

For example, it won’t cover any user-defined Python scripting that might be used to exert control over several workspaces.

However, it’s easy to look at this hierarchy diagram and imagine where such custom components might fit it.
Workspace
A workspace is the primary element in an FME translation and is responsible for storing a translation definition. A workspace is held as a file with an .fmw file extension. It can be run in either the Quick Translator or FME Workbench, but can only be opened for editing in Workbench.

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

Readers and Writers
A Reader is the FME term for the component in a translation that reads a source dataset. Likewise, a Writer is the component that writes to a destination dataset.

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

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

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

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

The “MySourceData” and “MyDestinationData” parts of the screenshot are the names of the datasets being read/written. When multiple datasets are read they are all listed, like in this AutoCAD reader.
**Feature Types**

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

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

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

---

Don’t confuse the term ‘Feature Type’ with ‘Geometry Type’. Feature Type means “layer”; Geometry Type means “lines”, “points”, “polygons”.

---

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

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

---

**Features**

Features are the smallest single components of an FME translation.

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

Here 2,186 road features were translated.
One-To-Many Relationships

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

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

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

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

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

- Create Workspace
- Add Reader/Writer
- Add Feature Types
- Import Feature Types
- Remove Reader/Writer
- Remove Feature Types
- Update Feature Types
- Move Feature Types
Create Workspace
The most common way to start creating a translation definition is to create a workspace through one of the tools in FME Workbench.

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

Creating a workspace through the Generate options is a simple way to define a translation because it includes reader, writer and feature type components in the setup process.
Add Reader/Writer
Adding a reader or writer to a workspace is a common requirement. There are several reasons:

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

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

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

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

Although the usual workflow is to create a new workspace with the Generate dialog and then add extra components as necessary, there's nothing to prevent a user starting with an empty workspace and simply adding readers and writers one-by-one.
Add Feature Types
Adding feature types can take place either when adding a reader/writer, or by a manual process.

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

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

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

Each chosen type – and not all have to be selected – will be represented by a feature type object below the new Reader.
Adding Feature Types with a Writer
When adding a writer, FME will offer a chance to manually add a new feature type.

Responding yes to this question opens up the dialog for manually defining a new feature type. Only one feature type can be added at this time. Additional ones must be added manually.

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

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

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

Choosing to add a feature type adds one to the translation, and then causes the Feature Type Properties dialog to appear in order to edit the feature type properties.

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

The User Attributes tab can be used to define the new feature type’s attribute schema
Copying from a Reader
In some cases a user will have a reader with multiple feature types, and wish to add the same ones to a writer. This is simply done by selecting the source feature types, right-clicking them, and using the option Duplicate (on Writer).

The command causes duplicates of the feature types to be added to the writer, and source/destination feature types to be automatically connected.

Again, at least one writer must exist in the translation hierarchy; else this option will be greyed out.

In the hierarchy diagram, copying feature types looks like this:
There are two readers/writers for Esri File Geodatabase format datasets. One requires ArcGIS to be installed and has greater functionality; the other (new for 2012) offers less functionality but operates without any application software requirements.

This case is interesting because the output dataset (transit.gdb) already exists.

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

### Example 1: CAD to GIS

<table>
<thead>
<tr>
<th>Scenario</th>
<th>FME user; City of Interopolis, Planning Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Roads (MicroStation Design v8, Esri File Geodatabase)</td>
</tr>
<tr>
<td>Overall Goal</td>
<td>Add CAD data from the engineering department to a public GIS dataset</td>
</tr>
<tr>
<td>This Step</td>
<td>Add the required components to a new workspace</td>
</tr>
<tr>
<td>Demonstrates</td>
<td>Adding readers, adding writers</td>
</tr>
<tr>
<td>Finished Workspace</td>
<td>C:\FMEData\Workspaces\DesktopManual\Session3Example1Complete.fmw</td>
</tr>
</tbody>
</table>

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

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

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

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

3) **Choose Feature Types**
When prompted, select both Roads and Labels as the feature types to add to the workspace.
4) Add a Writer
Now add a writer by selecting Writers > Add Writer from the menubar.

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

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

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

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

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

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

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

5) Copy Feature Type
Now a new feature type can be added to store the roads data.
Right-click the Roads feature type and choose Duplicate (on Writer).

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

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

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

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

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

7) Check Feature Type Parameters
Open the properties dialog for the Geodatabase Roads feature type.

Ensure the Allowed Geometries parameter is set to geodb_polyline

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

8) Save the Workspace
Save the workspace, but DO NOT run the workspace yet!
**Import Feature Types**

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

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

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

It only needs to read from a single table (roads), so there is a single Reader representing the database, and a single Feature Type representing the table.

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

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

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

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

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

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

It’s always preferable to import feature types, or copy them from an existing reader, rather than manually add them. That’s because a manual process is slower and more prone to user error; especially when case sensitivity is an issue.
Remove Reader/Writer
Tools exist to remove a reader or writer from a workspace, both on the menubar and in context menus in the Navigator window.

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

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

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

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

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

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

The workspace should not be run in this condition!

Performance suffers because all the source data is still being read. With a dangling reader it is discarded immediately, but with a dangling writer it is read and transformed too.
**Update Feature Types**
A frequent problem for users of spatial data occurs when – after setting up a data processing task – the structure of the source data changes; for example, a new attribute is added or the type of an existing attribute is changed from an integer to a floating point.

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

<table>
<thead>
<tr>
<th>Readers</th>
<th>Transformers</th>
<th>Writers</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Reader...</td>
<td>Import Feature Types...</td>
<td>Update Feature Types...</td>
<td>Enable/Disable Feature Types...</td>
</tr>
<tr>
<td>Remove Feature Types...</td>
<td>Remove Reader...</td>
<td>Add Reader as Resource...</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

Moving a feature type is not a common procedure, but it’s a very useful time-saver when necessary.
The next step in this CAD to GIS task is to convert some QLF data representing an airport, to the Esri File Geodatabase. This time, because a table already exists in the output dataset, Import Feature Type will be used.

1) Start Workbench
Start Workbench (if necessary) and open the workspace from the previous example. Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example2Begin.fmw

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

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

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

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

3) Import a Writer Feature Type
Select Writers > Import Feature Type from the menubar.

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

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

Since the only table required is Airport, when prompted uncheck all of the feature types except for Airport.
4) Map Schema
Map the reader feature type for Airport (it should be named qlf_record) to the writer feature type for Airport.

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

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

Parameters are what control a translation.

Chef Bimm says...

“Parameters in a translation are like the options when you order a coffee. You get to choose what ingredients you start with, how they are combined, and what the end result of the process will be like.

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

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

So there are:

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

Having parameters right down to the feature level provides a huge degree of control over every aspect of a translation.
Top-Down Effect
The basic rule is that any higher-level parameter affects every component below it.

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

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

To carry on Chef Bimm’s analogy, if you load a coffee machine with Decaffeinated coffee, then all of the drinks will be decaffeinated (Workspace Parameter).

But, each drink can still separately include cream and sugar (Feature Type Parameter)

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

Perhaps the best example of this is database writing mode.

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

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

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

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

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

Again it may help to return to the coffee maker analogy.

The coffee machine may have a temperature option to set the temperature of drinks. This parameter (being at the top level) will apply to all drinks.

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

Therefore, although the same temperature option occurs at each level, the lower level parameter takes priority (i.e. it overrides the higher-up parameter).
Locating Parameters
This diagram shows where the parameters are located for each translation component. Feature Types are interesting because their parameters are found in both the Navigator Window and the Feature Type Properties dialogs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Parameters</td>
<td>Navigator Window</td>
</tr>
<tr>
<td>Reader and Writer Parameters</td>
<td>Navigator Window</td>
</tr>
<tr>
<td>Feature Type Parameters</td>
<td>Navigator Window and Feature Type Properties dialog</td>
</tr>
<tr>
<td>Format Attributes</td>
<td>Feature Type Properties dialog</td>
</tr>
</tbody>
</table>

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

Workspace parameters relate to the workspace as a whole.

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

Workspace parameters are shown and set in the Navigator Window.

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

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

FME 2012 has a new section in the Navigator window called Workspace Properties.

This is a set of new metadata fields, but also includes Workspace Name and Workspace Description, which have now been moved from Workspace Parameters.

See the session on Best Practice for more information on Workspace Properties.
Basic Workspace Parameters
There are a number of basic workspace parameters. The most important one is Destination Redirect.

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

The destination redirect options are:

Redirect to Inspection Application: Output is sent directly to the FME Universal Viewer.

Redirect to FFS File: Output is sent to an FFS (FME Feature Store) file.

Disable Output: Output is ignored and not used (similar to a NULL format writer).

‘Redirect to Inspection Application’ can also be found on the menu bar, under the Writers menu.
Advanced Workspace Parameters
The advanced workspace parameters are perhaps not as valuable in everyday use, but have great importance in specific scenarios. Some particularly important ones are:

**Geometry Handling**
This parameter determines whether to use basic or enhanced geometry processing techniques.

The purpose of this parameter is to ensure backwards compatibility, when upgrading from an older version of FME. It has been removed in FME2012 as new workspaces default to ‘enhanced’ processing; however it will still appear when an older workspace is opened, in order to provide the option to run that workspace as it would have been in a prior version of FME.

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

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

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

**Password**
It’s often desirable to pass a workspace to an FME user for them to run, but not to edit. A password-protected workspace cannot be opened for editing in Workbench without the password. It can, however, still be run within the FME Universal Translator or from the command line.

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

**Start-up and Shutdown Scripts**
These parameters allow the ability to run a TCL or Python script before or after an FME translation.

Script parameters in the workspace settings dialog:

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

Potential uses of such scripts include:
- To check a database connection before running the translation
- To move data prior to or after the translation
- To write the translation results to a custom log or send them as e-mail to an administrator
- To run scripts from other applications; for example Esri ArcObjects Python scripts
Reader and Writer Parameters

Each reader or writer added to a workspace is controlled by a number of settings and parameters that are available.

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

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

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

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

Reader Parameters

Reader parameters are shown and set in the Navigator Window.

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

Doctor Workbench says...

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

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

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

Writer parameters are shown and set in the Navigator Window. To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.

---

**Parameter Priority**

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

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

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

*There isn't a different password per table!* Therefore it is a Reader/Writer level parameter.
The next step in this CAD to GIS task is to run the workspace.

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

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

1) Start Workbench
Start Workbench (if necessary) and open the workspace from the previous example. Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example3Begin.fmw

In the Navigator window, check the workspace parameter ‘Destination Redirect’. At the moment it will be set to ‘No Redirect’.

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

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

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

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

In the Navigator window locate the writer parameter Overwrite Geodatabase. Make sure it’s set to No.
3) Run the Workspace
Turn off the Destination Redirect and run the workspace. **Only run it once** else there will be multiple copies of the Road data.

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

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

Each reader or writer added to a workspace is controlled by a number of settings and parameters that are available.

Feature Type Parameters
Feature Types are at a lower level in the hierarchy than readers and writers. Therefore, Feature Type parameters don’t apply to datasets as a whole, but only to individual feature types within a dataset. They provide a degree of individual control over reading and writing different layers or tables.

Reader Feature Type Parameters
Reader feature type parameters apply to reading of specific layers/tables.

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

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

Not all feature types have parameters, so this tab is not always present.
Writer Feature Type Parameters
Writer feature type parameters apply to writing of specific layers/tables.

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

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

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

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

Therefore this is a feature type parameter.

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

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

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

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

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

Format Attributes

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

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

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

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

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

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

Notice how the attribute name starts with a format keyword, to differentiate the same format attributes for different formats of data.
FME Attributes
A particular set of Format Attributes has the prefix fme_. These attributes represent the data as it is perceived by FME and are sometimes known as FME Attributes or Generic FME Attributes.

When a translation is carried out, the following occurs:

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

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

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

FME merely converts everything to an FME standard and then from there to the Writer Format.
Controlling Features with Format Attributes
A user can make use of these attributes to carry out certain tasks by making the attributes part of the workspace.

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

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

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

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

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

Firefighter Mapp says…
‘An alternate method is to use the AttributeExposer transformer.’
Filtering with Format Attributes
One primary use of format attributes is as a means of filtering and directing source data within a workspace.

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

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

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

parcelList.treeType[0] = Pine, parcelList.treeType[1] = Oak, parcelList.treeType[2] = Cedar

Some format attributes can also be a list type of attribute.

Older versions of FME had a setting to expose a particular number of list elements for a format attribute, but that has been removed as transformer dialogs now ask which element in a list is to be used.

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

When they attempt to use that attribute in a Tester transformer, they are prompted to choose which element in the list is to be tested. Therefore it’s not necessary to have to expose multiple elements solely to get access to a single one.
**Transforming with Format Attributes**

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

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

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

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

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

This only really becomes a problem when reading and writing the same format, and the same format attributes exists on both reader and writer. In that case use the format attribute; it's not safe to use the fme equivalents.
Transformers for Setting Format Attributes
Format Attributes can actually be a little tricky to set.

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

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

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

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

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

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

The PenColorSetter (for example) “Sets the pen color of the feature”, but in reality all it does is set a new value for the format attribute fme_color.

The DGNStyler helps a user to define the symbology of features to be written to a MicroStation Design File. It is really just a more pleasant way of using format attributes.

Similar transformers are:
- DWGStyler
- KMLPropertySetter
- KMLStyler
- MapInfoStyler
- PDFStyler
On consideration, it’s a good idea to empty a table before writing data to it, in case there were any features in the airport table that we no longer need. The simplest method is to re-run the workspace, dropping the table before re-writing to it.

The same technique should also be applied to the Roads table, since that would get a double set of data if the workspace were to be simply re-run.

1) Start Workbench
Start Workbench (if necessary) and open the workspace from Example 3. Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example4Begin.fmw

2) Set Feature Type Parameter
In the Navigator window, locate the Geodatabase writer feature type parameter 'Drop Existing Table' ('Drop Table First' in ArcObjects writer) for each of the Airport and Roads feature types.

Set them both to Yes. This will ensure the tables are emptied out before any data is written to them.

3) Run the Workspace
At this point you can re-run the workspace to prove that the Airport and Roads tables only contains a single set of geometry.
When inspecting the data you may notice that the `LineJoiner` transformer did not do a perfect job joining line features. That’s because the `LineJoiner` works on geometry and lines won’t be joined where there is more than one possible connection that can be made; for example, where two roads cross.

This can be solved by using a Group-By; that is, joining only lines with the same road ID should remove unwanted connections and let FME determine where the proper join lies.

4) **Expose a Format Attribute**

The road ID to group by is not available as a user attribute. That’s because MicroStation data does not permit attributes. However, it does permit basic feature IDs called ‘mslinks’.

In FME this ID is available as a format attribute that must be exposed before it can be used by Workbench.

Open the properties dialog for the DGNV8:Roads reader feature type. Click the Format Attributes tab. Place a check mark next to the entry for `mslink_0` and click OK.

5) **Fix LineJoiner Transformer**

In the `LineJoiner` transformer set a group-by to group features by `mslink_0`.

6) **Run Workspace**

Run the workspace again and you should now find that all roads with the same ID are properly joined together.
The team responsible for managing transit data have released updates to the Bus Stops table. One bus stop in particular was in the wrong position and has been moved. However, the new data has been released as an “updates only” dataset and must be applied as an update to the existing bus stops table. Since only individual features are being updated (not the whole table) this can be achieved using format attributes.

7) Add a Reader
Select Readers > Add Reader from the menubar. Fill in the Add Reader dialog as follows:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>Esri Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData\Data\Transit\BusStopUpdates\Bus Stops.shp</td>
</tr>
</tbody>
</table>

8) Import Feature Type
As the workspace doesn’t yet have a WRITER feature type for Bus Stops, import it from the existing dataset Transit.gdb using Writers > Import Feature Types.

When prompted, only BusStops needs to be selected.
9) Place AttributeCreator Transformer
Format attributes for use on a writer are most easily defined using an AttributeCreator transformer. This is because exposing them on a writer does not let you set them within the workspace.

Place an AttributeCreator transformer connected to the reader feature type for BusStops.

10) Set AttributeCreator Parameters
Open the AttributeCreator parameters.
In the Attribute Name field, create two new attributes: fme_db_operation and fme_where

Click in the value field for fme_db_operation and enter the value: UPDATE
Click on the menu for fme_where and choose the option: Open String Editor

11) Set String Editor Parameters
The string editor dialog will be used to construct a ‘where’ clause for the update. It requires three parts to be concatenated.

Firstly, set a string type of Constant.
In the value field enter: STOPABBR='

Secondly, set a string type of Attribute Value.
In the value field, select the attribute STOPABBR

Finally, set a string type of Constant again.
This time enter just a single quote character: ‘

The Concatenated Result field should now show: STOPABBR='[STOPABBR]'

…this is basically a ‘where’ clause that means “where the field STOPABBR matches the value of the incoming attribute STOPABBR”. The quotes are required around the value because it is a text field (not just numeric).
12) Run Workspace
Connect the AttributeCreator:OUTPUT port to the Geodatabase BusStops feature type.

Run the workspace again and inspect the transit dataset.
You should now find that the erroneous bus stop has been moved into the correct position.

If you can’t find the new bus stop, it should be located close to:

X: 3129150
Y: 10097400
Parameter Documentation

The FME Readers and Writers Reference Manual documents all of the parameters available for each format, from the reader and writer level down to format attributes for controlling features.

FME Readers and Writers Reference Manual
The FME Readers and Writers Reference Manual is part of the help system included with FME Workbench. It is where the parameters for each level of the translation hierarchy are documented.

This manual lists – format by format – what parameters exist for the reader, writer, feature types and features.

Importantly, each parameter includes a description of what it does, and in some cases what values are acceptable.

However, because the terminology is oriented to mapping files, it differs slightly from what is used in this manual.

In general: Reader/Writer parameters are called Directives, and feature type parameters are documented under a particular directive called a DEF line. Format Attributes are listed in a section called Feature Representation.

<table>
<thead>
<tr>
<th>Translation Component</th>
<th>Reader and Writer Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Parameters</td>
<td>Reader Directives</td>
</tr>
<tr>
<td>Writer Parameters</td>
<td>Writer Directives</td>
</tr>
<tr>
<td>Reader Feature Type Parameters</td>
<td>Reader DEF Line Directive</td>
</tr>
<tr>
<td>Writer Feature Type Parameters</td>
<td>Writer DEF Line Directive</td>
</tr>
<tr>
<td>Format Attributes</td>
<td>Feature Representation Attributes</td>
</tr>
</tbody>
</table>

In the contents page Reader Directives (including the Reader DEF line) are accessed through Reader Overview.

Writer Directives (including the Writer DEF line) are accessed through Writer Overview

Feature Representation is where the list of format attributes is accessed.
**Reader/Writer Directives**

Taking the format Oracle Spatial Object as an example, the writer parameters are nicely mirrored in the Writer Directives section of the Manual.

For example, there are directives for transaction interval, chunk size, and writer mode.

Username and password are documented under the Oracle Reader Directives, and so aren't repeated here.

**Feature Type Directives**

Notice one of the writer directives is called DEF. This is where all of the writer feature type parameters are documented; for example Drop Table, Truncate Table, and Update Key.

**DEF**

Required/Optional: **Required**

Each Oracle Database table must be defined before it can be written. The general form of an Oracle Database definition statement is:

```sql
OracleSpatialDB.DEF <tableName> 
   [oracleSQL-encoded <sqlquery>] 
   [oracleUpdateKeyColumns <column>[,<column>]...] 
   [oracleDeleteKeyColumns <column>[,<column>]...] 
   [oracleDropTable (yes|no)] 
   [oracleTruncateTable (yes|no)] 
   [oracleParams <creationParams>] 
   [oracleSequencedCols column[.:seqname][;column[:seqname]]...] 
   [<fieldName> <fieldType>]*
```

**Feature Representation**

The Feature Representation section is divided up into different geometry types.

Each geometry has a list of applicable format attributes; for example a geometry type of oracle_circle possesses format attributes that describe circle radius (oracle_radius) and circle rotation (oracle_rotation)
## Example 5: CAD to GIS

<table>
<thead>
<tr>
<th><strong>Scenario</strong></th>
<th>FME user; City of Interopolis, Planning Department</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>File Geodatabase</td>
</tr>
<tr>
<td><strong>Overall Goal</strong></td>
<td>Add CAD data from the engineering department to a public GIS dataset</td>
</tr>
<tr>
<td><strong>This Step</strong></td>
<td>Check Labels feature type</td>
</tr>
<tr>
<td><strong>Demonstrates</strong></td>
<td>Use of Readers and Writers Manual</td>
</tr>
</tbody>
</table>

Notice that the Labels feature type from the source MicroStation dataset has not been connected up to any table in the Geodatabase output.

As a very quick example, check the Quick Facts section of the Readers and Writers Manual for the two Esri Geodatabase reader/writers.

Can you tell if it is even possible to write text features to a Geodatabase?
Published Parameters

Publishing parameters is a method by which FME prompts to change read and write control parameters at runtime.

As shown, each different level of translation hierarchy has a related set of control parameters.

However, there are two potential users of FME; a workspace author and the end-user.

Ideally, if the end-user wanted to change one or more of these parameters, they should not need to edit the workspace in the same way that an author would.

Published Parameters provide this functionality.

Publishing parameters is a way to prompt the end user to enter a value, in much the same way that FME will prompt for any undefined mandatory parameters.

Prompting the user for values thus avoids the need to make manual edits in the workspace.

Any setting that can be defined through the Navigator window is capable of being set as a published parameter. This includes most workspace parameters, all reader and writer parameters, and all feature type parameters.

Chef Bimm says...

“Think of Published Parameters as your coffee options printed on the side of a paper cup.

They are a way for a customer to have their wishes defined without setting the machine themself”.
Publishing Parameters
To publish a parameter, simply locate it in the Navigator window, right-click it, and choose the **Create User Parameter** option.

This opens a dialog for defining the publishing settings. After setting a name for the published parameter, its icon changes to a purple color to denote its published status.

Here the user is choosing to publish a reader parameter that determines what character encoding the source data is in.

Notice how the parameter definition includes:

- Parameter name
- Parameter prompt
- Default value
- Optional Flag
- Published Flag

The ‘published’ flag exists because there are two different parameter states: public and private.

- Public means the parameter is published for an end-user to set.
- Private means it can only be used inside the workspace, but shareable in several places.

For ease of editing, all published parameters also appear in a separate section of the Navigator.
Publishing Workspace Parameters
To publish a workspace parameter simply locate the parameter in the Navigator, right-click, and choose Create User Parameter.

The limitation here is that many basic parameters (such as Destination Redirect or Workspace Password) can’t be published because it doesn’t make sense to do so.

In such cases that option is simply left off the menu:

Publishing Reader/Writer Parameters
Publishing reader and writer parameters is the most common use of this functionality. It is achieved by using the same right-click context menu in the Navigator window.

Notice that some parameters are automatically published by FME. That’s because these are common parameters the user will often need to set; for example reader input file and writer output file locations.

If it’s not appropriate for a user to select these, then the published parameters can be simply removed.

Publishing Feature Type Parameters
For Feature Type parameters, it’s important to notice that these can only be published in the Navigator window. There are NO right-click > publish options in the Feature Type Properties dialog.

Publishing Format Attributes
For publishing Format Attributes, or any other attribute or transformer values, see session 5.

First-Officer Transformer says...
‘Besides translation components for reading and writing, you can also publish parameters for transformers; again by locating the parameter to be published in the Navigator window, right-clicking, and choosing Create User Parameter.’
Using Published Parameters
Published parameters are activated whenever a workspace is run using the option *File > Prompt and Run Translation* from the menu bar in FME Workbench.

The shortcut for this command is *Ctrl+R* and there is a related Prompt and Run button on the toolbar.

**Note:** The simpler *File > Run* (shortcut *F5*) will not prompt for values, but re-use existing ones.

When a workspace is run in prompt mode, the user receives a dialog prompting them to enter new values.

Running a workspace in the FME Quick Translator always prompts for parameters; which is why it’s so suitable for non-FME-authoring end-users to run a workspace.

### Published Parameters on the Command Line
In the same way that FME translations can be run from the command line, they can be passed values to published parameters on the command line using the syntax:

```
--<Parameter Name> <Parameter Value>
```

Notice that the parameter name matches that supplied in the published parameter definition.

*Windows command-line to run this workspace:*

```
fme.exe ExS-StructuralTransformation-begin.fmw
    --SourceDataset_MAPINFO C:\FMEData\Data\Parks\city_parks.tab
    --DestDataset_GML C:\FMEData\Output\TutorialOutput\Parks.gml
```

As usual, the log window reveals the command line used in the above translation.

The command line in the Log window specifically states ‘Windows command-line’.

The biggest difference between Windows and UNIX shells is how parameters with spaces in them are quoted.
Example 6: CAD to GIS

Scenario | FME user; City of Interopolis, Planning Department
Data | Esri File Geodatabase
Overall Goal | Add CAD data from the engineering department to a public GIS dataset
This Step | Set “over-write table” parameter,
Demonstrates | Feature Type Parameters, Format Attributes
Starting Workspace | C:\FMEData\Workspaces\DesktopManual\Session3Example6Begin.fmw
Finished Workspace | C:\FMEData\Workspaces\DesktopManual\Session3Example6Complete.fmw

1) Start Workbench
Start Workbench (if necessary) and open the completed workspace from Example 4. Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example6Begin.fmw

2) Delete Parameters
Because it’s not necessary for end-users to change the source or destination datasets, locate the published parameters for all reader and writer dataset parameters.

Either delete them from the Published Parameters section or un-publish them from the Reader/Writer sections.

3) Publish Parameter
Publish the ‘Overwrite Table’ parameters for both Geodatabase Roads and Airport tables so that users are prompted at runtime whether to drop the contents or not. Make sure you alter the prompts so that the user can tell which parameter refers to which table.

4) Run the Workspace Again
Use File > Prompt and Run to run the workspace again to prove that the published parameters now prompt for a value.
Module Review

This session was designed to increase your knowledge of how FME handles different spatial data formats.

What You Should Have Learned from this Session
The following are key points to be learned from this session.

Theory

- A Workspace has a hierarchy of **Readers** and **Writers**, **Feature Types**, and **Features**

- Translations are controlled by **Read and Write Parameters** and **Format Attributes**, which have a similar hierarchy to the structure of Reader-Dataset-Feature Type.

- **Format Attributes** store information related to the structure and symbology of a feature. Format attributes can be used to **filter** source features, or to **re-symbolize** or **re-structure destination** features.

- **Published Parameters** are a way to prompt users to set values at run time.

FME Skills

- The ability to set up and manage the components of a translation.
- The ability to control a translation with read and write parameters.
- The ability to apply Format Attributes on either source or destination features.
- The ability to find out how features will be affected when converting between formats.
- The ability to publish parameters from readers and writers.