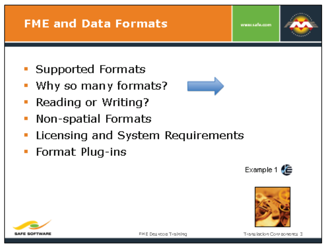
SAFE SOFTWARE

# Translation Components

## FME and Data Formats



FME and Data Formats

www.safe.com

- Supported Formats
- Why so many formats?
- Reading or Writing?
- Non-spatial Formats
- Licensing and System Requirements
- Format Plug-ins

Example 1

SAFE SOFTWARE   FME Desktop Training   Translation Components 3

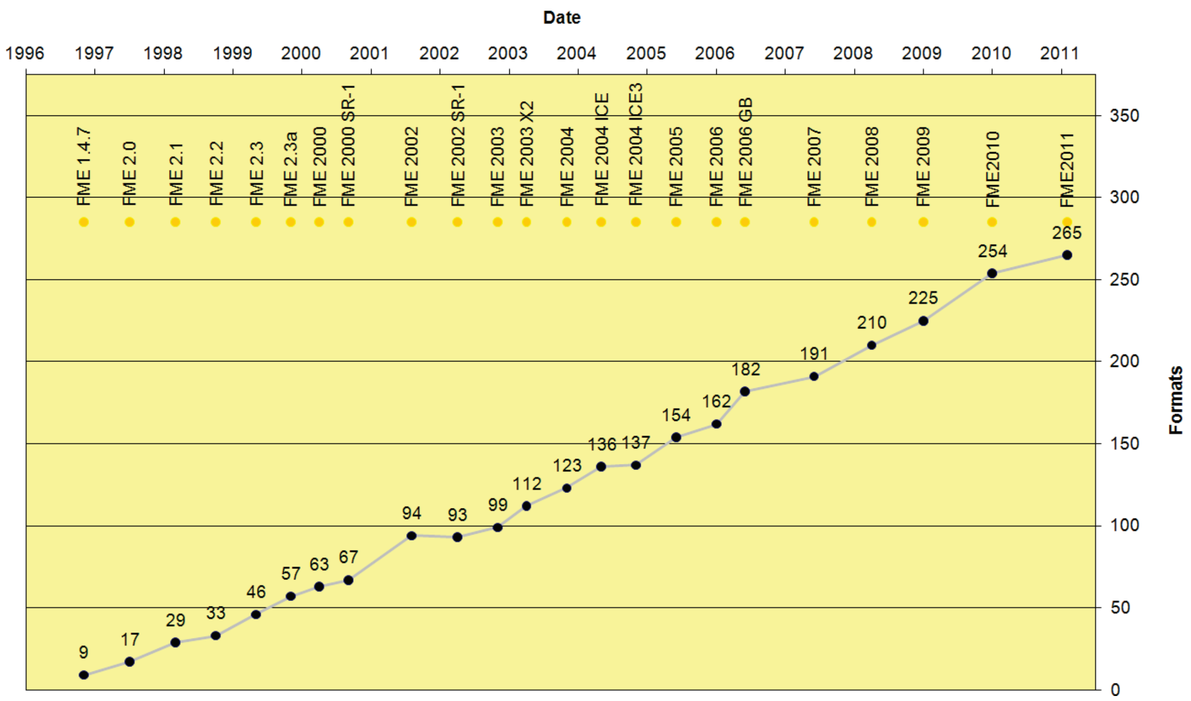
***Interoperability is the ability to exchange information between systems and applications. FME aids interoperability by supporting translations between numerous spatial and non-spatial data formats.***

### Supported Formats

FME supports over 250 data formats.

Surprisingly the rate of increase has been steady over the last fifteen years and shows little sign of levelling out (***below***).

**FME Supported Formats**

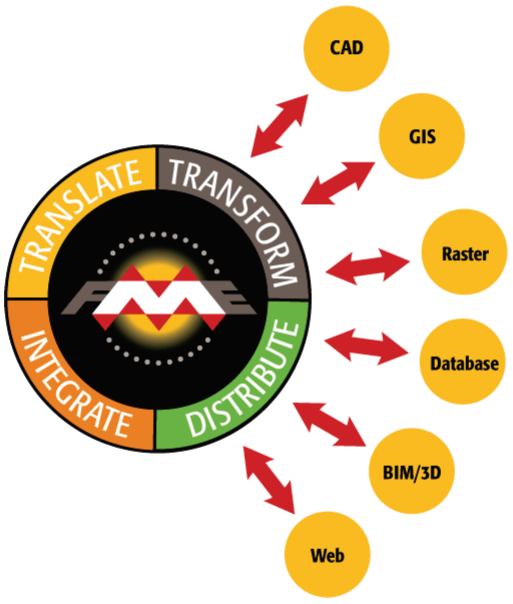


Mr. Saif-Investor says…

*'Although past performance is no guarantee of future results, this is one graph I don't see turning downwards any time soon.*

*FME's ability to work simultaneously with multiple formats – in a totally semantic manner – lets me bring together widely disparate datasets in the form I want and with minimal effort on my part.'*

**_Why So Many Formats?_**
The large number of formats arises because there are so many different fields that use spatial data. At Safe we sometimes call these *families* of data and each of these families has data with a set of characteristics that differentiates it from the structures of other data types.



| Type | Characteristics |
|---|---|
| CAD | *High-precision elements, traditionally backed by little or no underlying intelligence.* |
| GIS | *Spatial representation of real-world features, mostly with underlying intelligence.* |
| Raster | *Representations of spatial data as a series of cells – usually for GIS or visualization purposes.* |
| Database | *Data held as a centralized collection of records and usually processed using transactions.* |
| BIM/3D | *Data of buildings, often based on a data model where the building is divided into components.* |
| Web | *Any form of spatial data – vector or raster – that is stored online and distributed through a network.* |

To the preceding families, the following items could also be added:

| | |
|---|---|
| Cartographic | *Data optimized to visually highlight certain spatial characteristics or concepts.* |
| Transfer | *Formats specifically designed as a means to standardize the supply of data.* |
| Point Cloud | *Formats specialized for storing and transferring large quantities of point data* |

Of course, the big challenge is to preserve meaning and content when working with different types of data; for example, combining cartographic and database data into a GIS ready output. This is where in-depth knowledge of FME's readers and writers is a great benefit.

**_Non Spatial_**
FME also supports a number of non-spatial formats. Therefore, not only can FME work with the non-spatial attributes of spatial features, it's also able to work on a totally non-spatial basis.

**_Reading or Writing_**
It's worth noting that not every FME supported format permits both reading and writing. Some formats only support reading (for example, ESRI ArcGIS mxd files), whereas others only support writing; for example, SVG. However the majority do have support for both.

## Licensing and System Requirements

FME is available in various editions. The FME edition is determined by the license, rather than by the product that is installed, so the FME installation disk and downloads for FME are all identical.

Each different FME edition differs only in the formats available. Only the FME Desktop Base Edition has a different (more limited) set of functionality. Editions are often named for the type of formats they support; for example, support for GE Smallworld format datasets is provided by the FME Smallworld Edition.

Some formats are only supported when suitable application software is also installed on the user's system. ESRI Geodatabase is an example of this. Because FME uses ArcObjects to read from a Geodatabase, it's necessary that ArcGIS is installed and licensed to enable this format in FME.

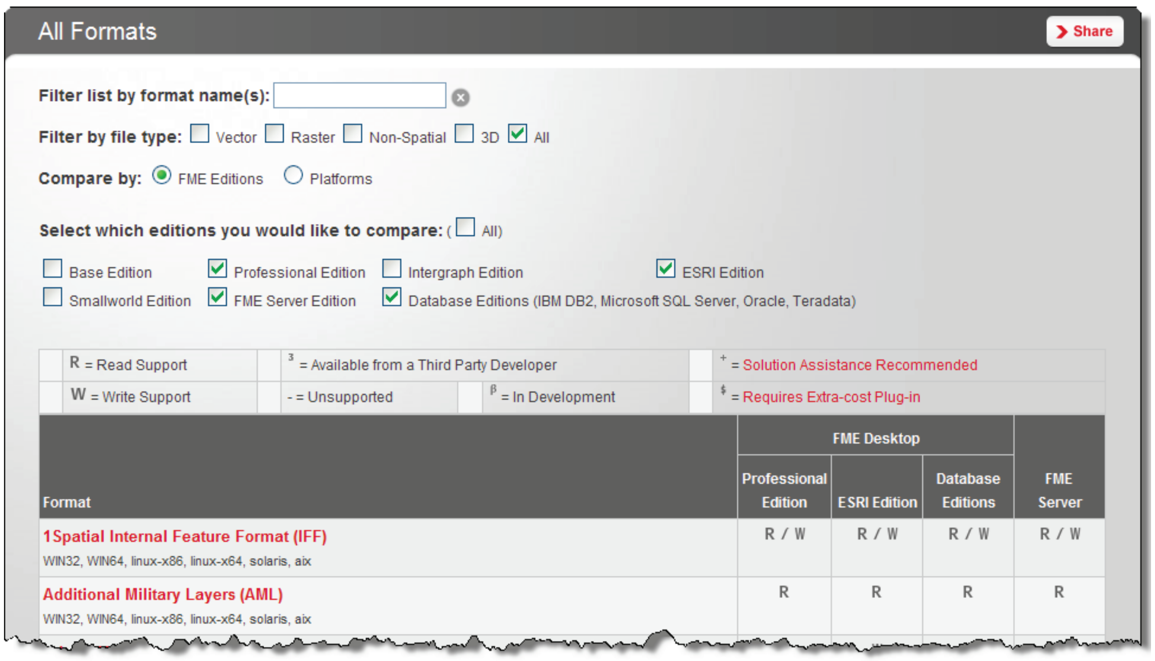
Similar limitations restrict the availability of formats on a particular platform; for example many formats aren't available on Linux or Windows 64-bit operating systems, because the required technology (or application) just isn't available for that platform.

### *Format Plug-Ins*

Some FME supported formats relate to very specialist data types. In these cases support is provided by an extra-cost plug-in. In some cases these plug-ins are created by third-party suppliers using the FME Plug-In SDK.

Ms. Surveyor says…

*'The full list of supported formats – including the edition they are supported by – is available on Safe Software's web site at www.safe.com/formats.'*



All Formats

Filter list by format name(s):

Filter by file type: ☐ Vector ☐ Raster ☐ Non-Spatial ☐ 3D ☑ All

Compare by: ● FME Editions ○ Platforms

Select which editions you would like to compare: ( ☐ All)

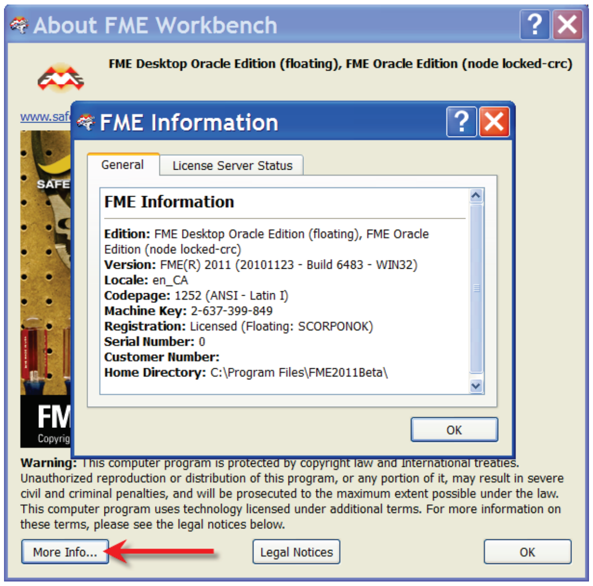
☐ Base Edition ☑ Professional Edition ☐ Intergraph Edition ☑ ESRI Edition
☐ Smallworld Edition ☑ FME Server Edition ☑ Database Editions (IBM DB2, Microsoft SQL Server, Oracle, Teradata)

| | | | |
|---|---|---|---|
| R = Read Support | 3 = Available from a Third Party Developer | | + = Solution Assistance Recommended |
| W = Write Support | - = Unsupported | β = In Development | $ = Requires Extra-cost Plug-in |

| Format | FME Desktop | | | |
|---|---|---|---|---|
| | Professional Edition | ESRI Edition | Database Editions | FME Server |
| 1Spatial Internal Feature Format (IFF)<br>WIN32, WIN64, linux-x86, linux-x64, solaris, aix | R / W | R / W | R / W | R / W |
| Additional Military Layers (AML)<br>WIN32, WIN64, linux-x86, linux-x64, solaris, aix | R | R | R | R |

*__Knowing the FME Version, FME edition, license type, and platform is key to determining whether a required format is supported by a certain FME installation.__*

*__This information can be obtained in FME Workbench by selecting Help > About FME Workbench from the menubar, and then clicking the More Info... button.__*



FME Desktop Oracle Edition (floating), FME Oracle Edition (node locked-crc)

**FME Information**

**Edition:** FME Desktop Oracle Edition (floating), FME Oracle Edition (node locked-crc)
**Version:** FME(R) 2011 (20101123 - Build 6483 - WIN32)
**Locale:** en_CA
**Codepage:** 1252 (ANSI - Latin I)
**Machine Key:** 2-637-399-849
**Registration:** Licensed (Floating: SCORPONOK)
**Serial Number:** 0
**Customer Number:**
**Home Directory:** C:\Program Files\FME2011Beta\

**Warning:** This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law. This computer program uses technology licensed under additional terms. For more information on these terms, please see the legal notices below.

| **Example 1: CAD to GIS** | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Roads (MicroStation Design v8, Autodesk MapGuide Enterprise SDF) |
| **Overall Goal** | Add CAD data from the engineering department to a public GIS dataset |
| **This Step** | Check pre-requisites for the translation |
| **Demonstrates** | Supported Formats |

You have been asked to add a set of CAD roads data (MicroStation v8 format) to an existing GIS dataset (SDF format). The first step in this task is to check for any format pre-requisites.

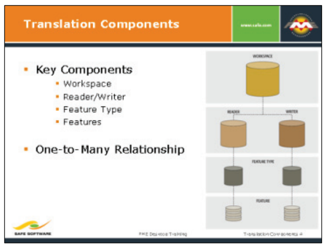
Autodesk MapGuide Enterprise SDF is an interesting format; it's a file dataset that behaves with all the characteristics of a database. In fact, underneath it's actually built on a SQLite database.

**1) Check Format Requirements**
Visit the Safe Software web site to check the requirements for the two formats.

**2) Start Workbench**
Start FME Workbench and use Help > About FME Workbench to determine whether the installation will support the format pre-requisites.

Remember that the task requires *reading* MicroStation data, and *writing* SDF data.

## Translation Components



Translation Components

- Key Components
  - Workspace
  - Reader/Writer
  - Feature Type
  - Features
- One-to-Many Relationship

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

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

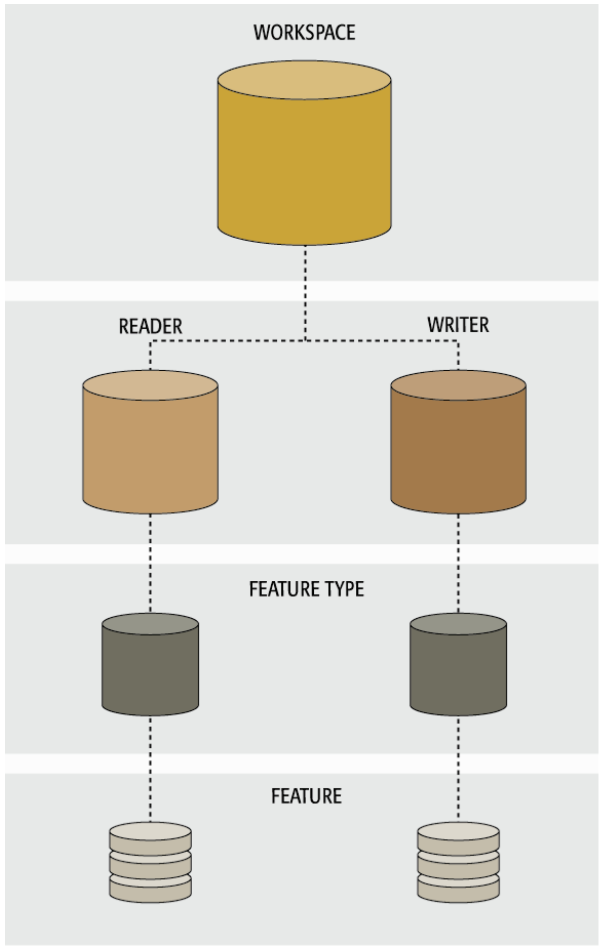
For each component there are tools within FME to create, add, or remove them; and parameters in Workbench in order to control them.

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*

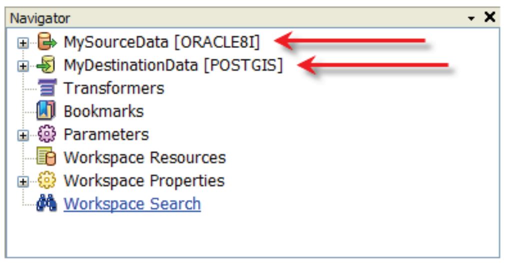
A workspace is the primary element in an FME translation and is responsible for storing a translation definition. A workspace is held as a file with a .fmw file extension. It can be run in either the Quick Translator or FME Workbench, but can only be opened for editing in Workbench.

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.

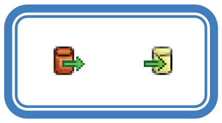
Each reader and writer in a workspace handles just a single format of data. To read or write multiple formats requires the use of multiple readers and/or writers.



Navigator
- MySourceData [ORACLE8I]
- MyDestinationData [POSTGIS]
- Transformers
- Bookmarks
- Parameters
- Workspace Resources
- Workspace Properties
- Workspace Search

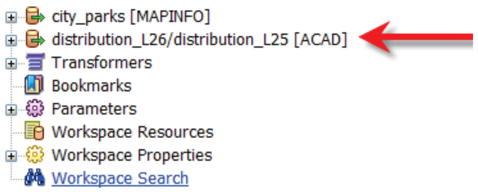
Readers and writers don't appear as objects on the Workbench canvas, but are represented by entries in the Navigator window. Each reader and writer appears as a separate entry in the list.

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.



- city_parks [MAPINFO]
- distribution_L26/distribution_L25 [ACAD]
- Transformers
- Bookmarks
- Parameters
- Workspace Resources
- Workspace Properties
- Workspace Search

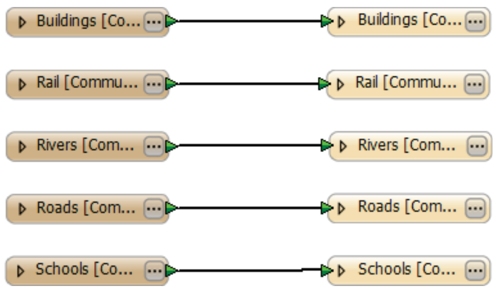
## *Feature Types*

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

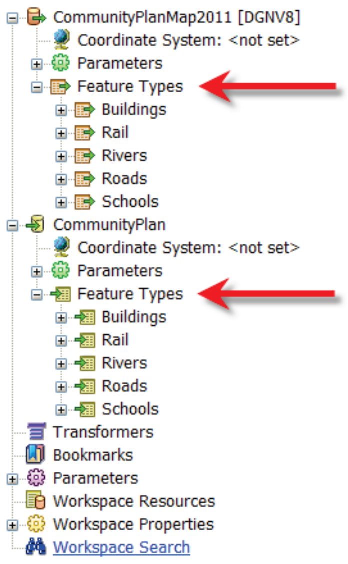
A Feature Type as a translation component is simply a defined layer in the process. If a specific layer is not defined by a feature type, then that data will not be either read and/or written.

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.


```
CommunityPlanMap2011 [DGNV8]
    Coordinate System: <not set>
    Parameters
    Feature Types
        Buildings
        Rail
        Rivers
        Roads
        Schools
CommunityPlan
    Coordinate System: <not set>
    Parameters
    Feature Types
        Buildings
        Rail
        Rivers
        Roads
        Schools
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

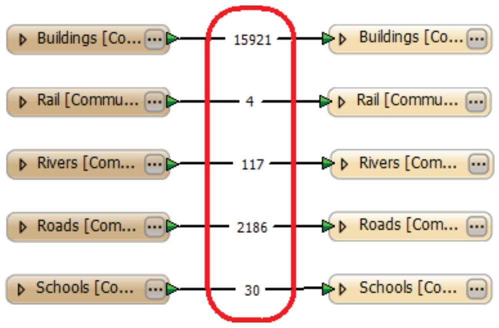
Beside canvas objects, feature types can be found listed in the Navigator window. They are the only component to be listed in two places in this way. A writer feature type icon looks like this:

## *Features*

Features are the smallest single components of an FME translation.

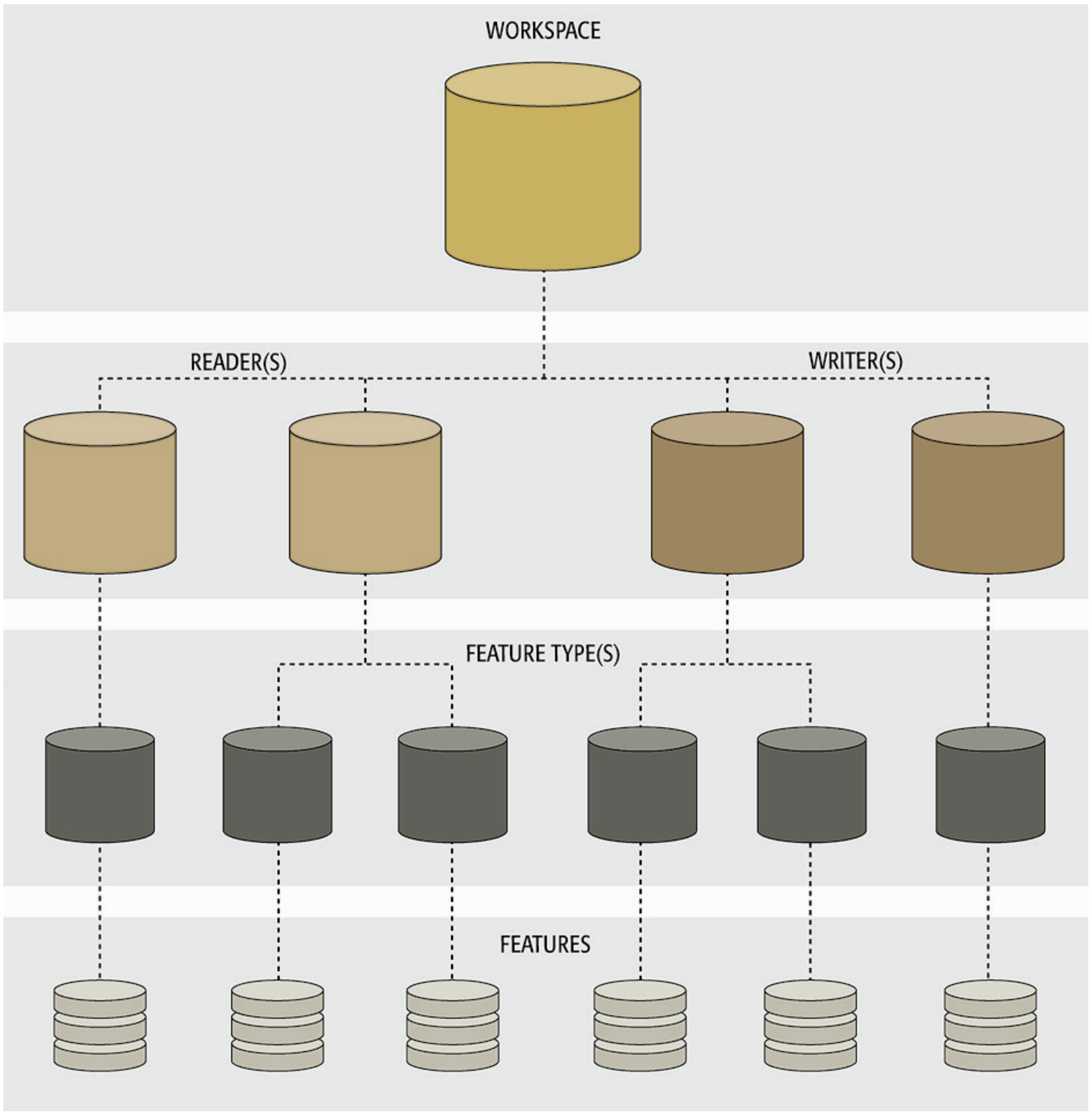
They aren't individually represented within a workspace, except by the feature counts on a completed translation.

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

Managing Components 1

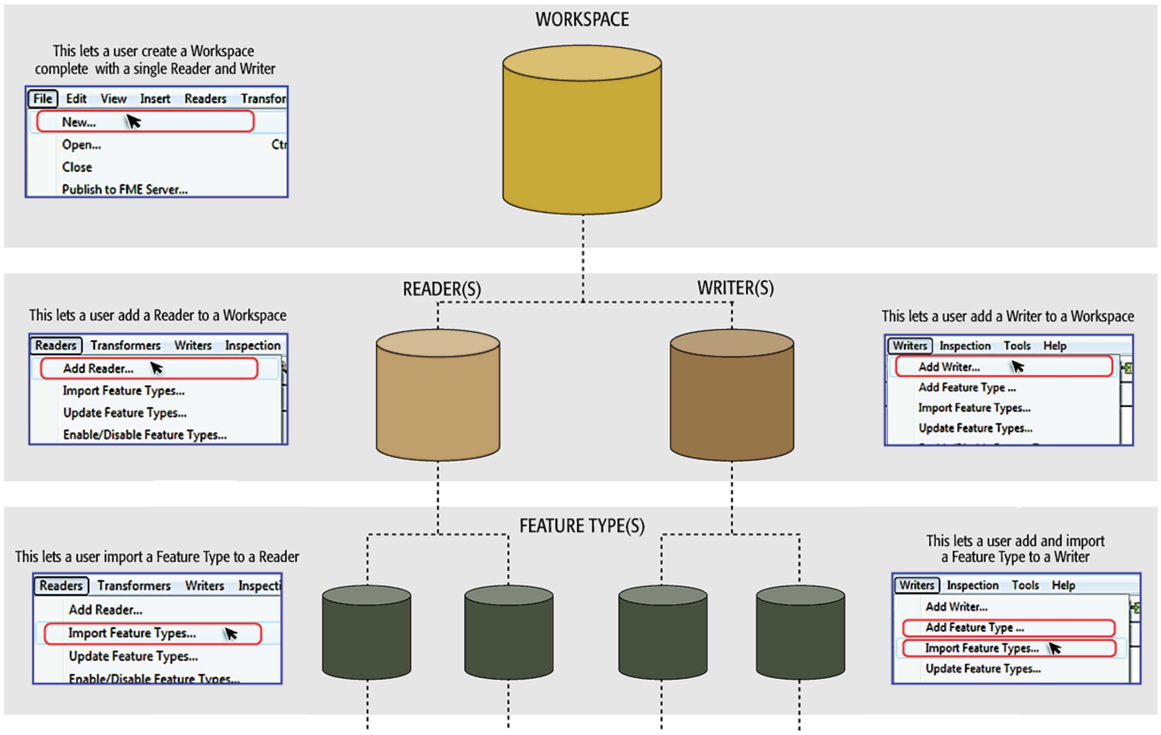
- Component Management Tools
- Create Workspace
- Add Reader/Writer
- Add Feature Types

Example 2

***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.


WORKSPACE
This lets a user create a Workspace complete with a single Reader and Writer
File Edit View Insert Readers Transfor
New...
Open...
Close
Publish to FME Server...
READER(S)
WRITER(S)
This lets a user add a Reader to a Workspace
Readers Transformers Writers Inspection
Add Reader...
Import Feature Types...
Update Feature Types...
Enable/Disable Feature Types...
This lets a user add a Writer to a Workspace
Writers Inspection Tools Help
Add Writer...
Add Feature Type ...
Import Feature Types...
Update Feature Types...
FEATURE TYPE(S)
This lets a user import a Feature Type to a Reader
Readers Transformers Writers Inspecti
Add Reader...
Import Feature Types...
Update Feature Types...
Enable/Disable Feature Types...
This lets a user add and import a Feature Type to a Writer
Writers Inspection Tools Help
Add Writer...
Add Feature Type ...
Import Feature Types...
Update Feature Types...

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

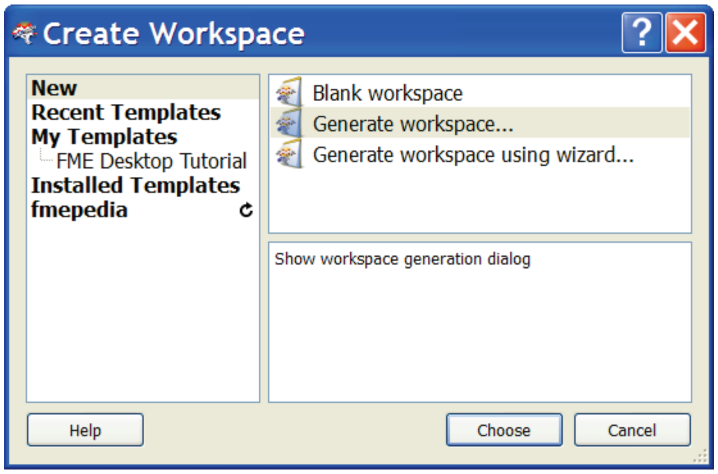
- Create Workspace
- Add Reader/Writer
- Import Feature Types
- Add Feature Types
- Remove Reader/Writer
- Remove Feature Types
- Update Feature Types
- Move Feature Type

## 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 form 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.

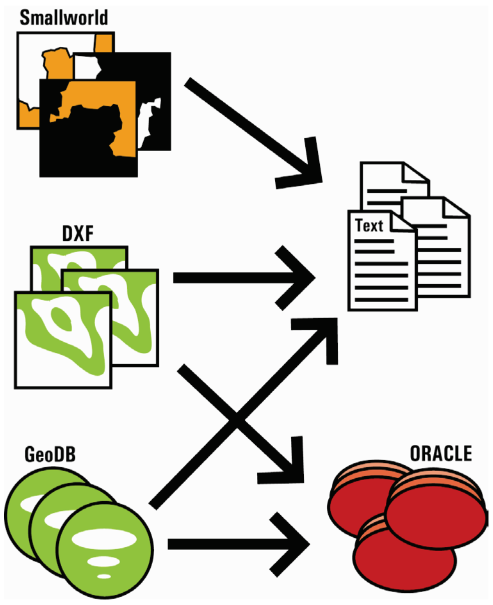


Show workspace generation dialog

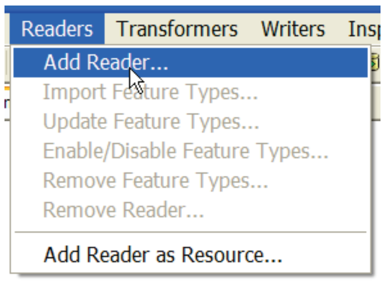
## Add Reader/Writer

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

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

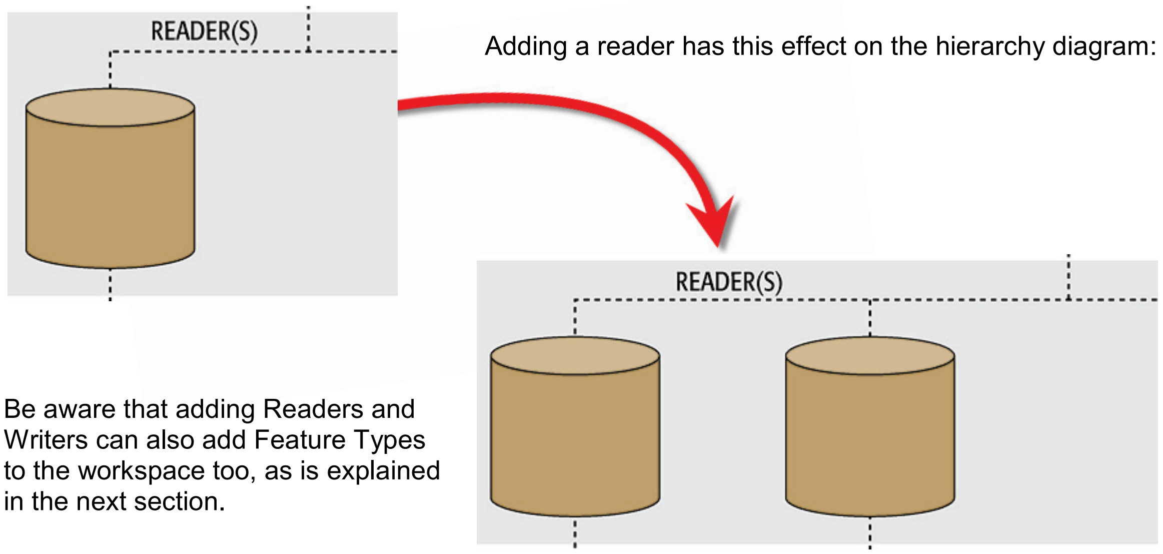


Therefore handling multiple formats of data – such as a workspace that reads Smallworld, DXF, *and* Geodatabase; and writes to both Oracle and a text file – requires multiple readers/writers.



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

Adding a reader has this effect on the hierarchy diagram:



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

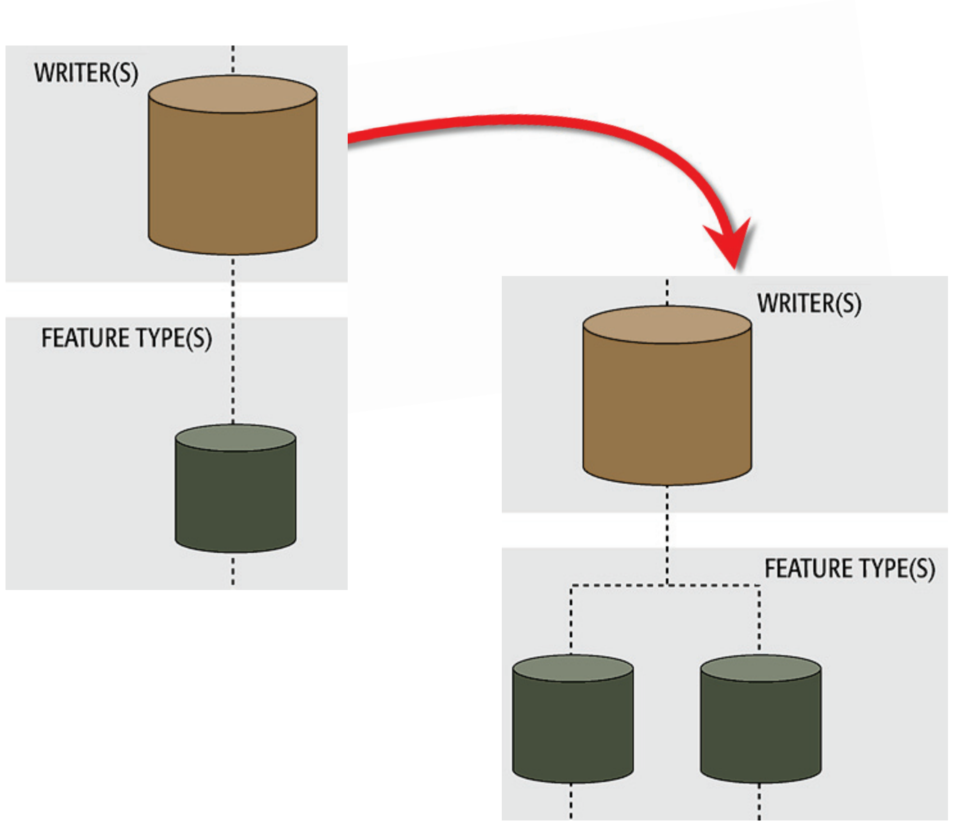
***Although the usual workflow is to create a new workspace with the generate dialog and then add extra components as necessary, there's nothing to prevent a user starting with an empty workspace and simply adding readers and writers one-by-one.***

## Add Feature Types

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

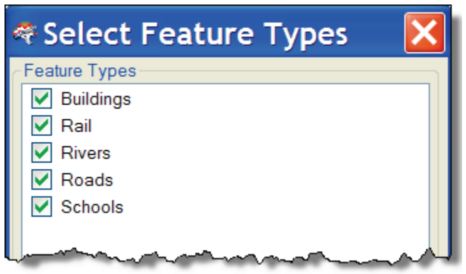
Generally manually adding a feature type is only permitted on the writer side of a translation. That's because the writer side is "What We Want" and is therefore open to manual editing.

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

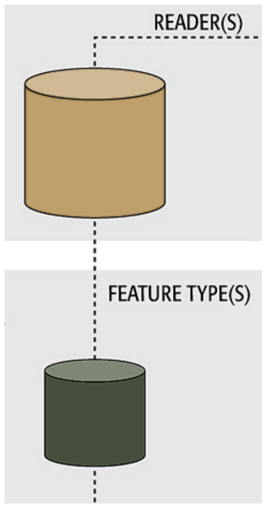


### *Adding Feature Types with a Reader*

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.



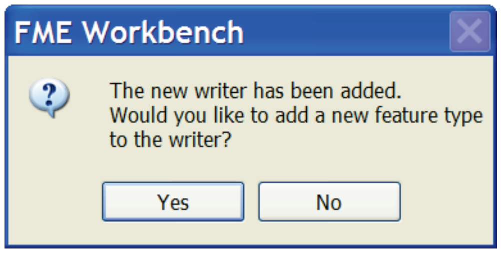
- [✓] Buildings
- [✓] Rail
- [✓] Rivers
- [✓] Roads
- [✓] Schools



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.



FME Workbench

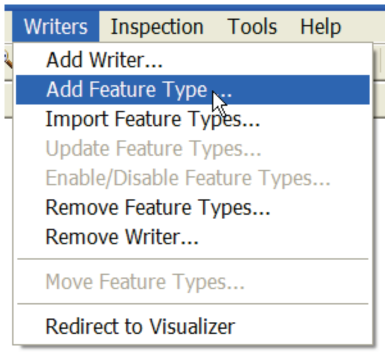
The new writer has been added.
Would you like to add a new feature type to the writer?

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***

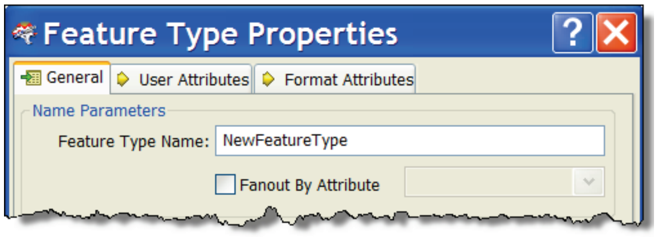
Feature Types can be added manually to a writer using Writers > Add Feature Type on the menubar.



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.

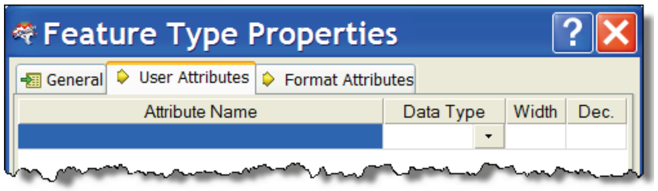
The general tab can be used to define the new feature type's name



Feature Type Name: NewFeatureType

Fanout By Attribute

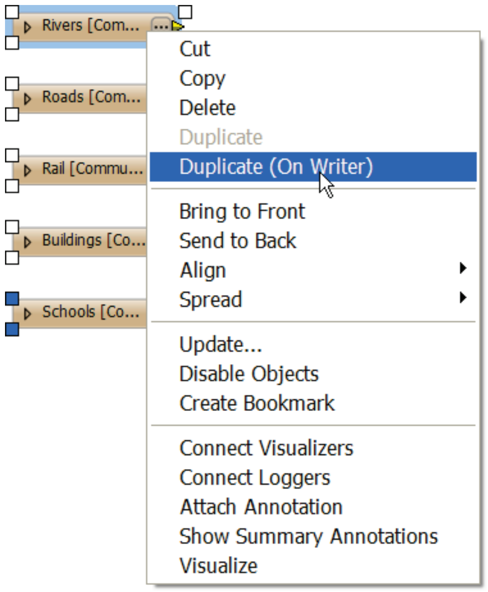
The user attributes tab can be used to define the new feature type's attribute schema



| Attribute Name | Data Type | Width | Dec. |
|---|---|---|---|
| | | | |

### *Copying from a Reader*

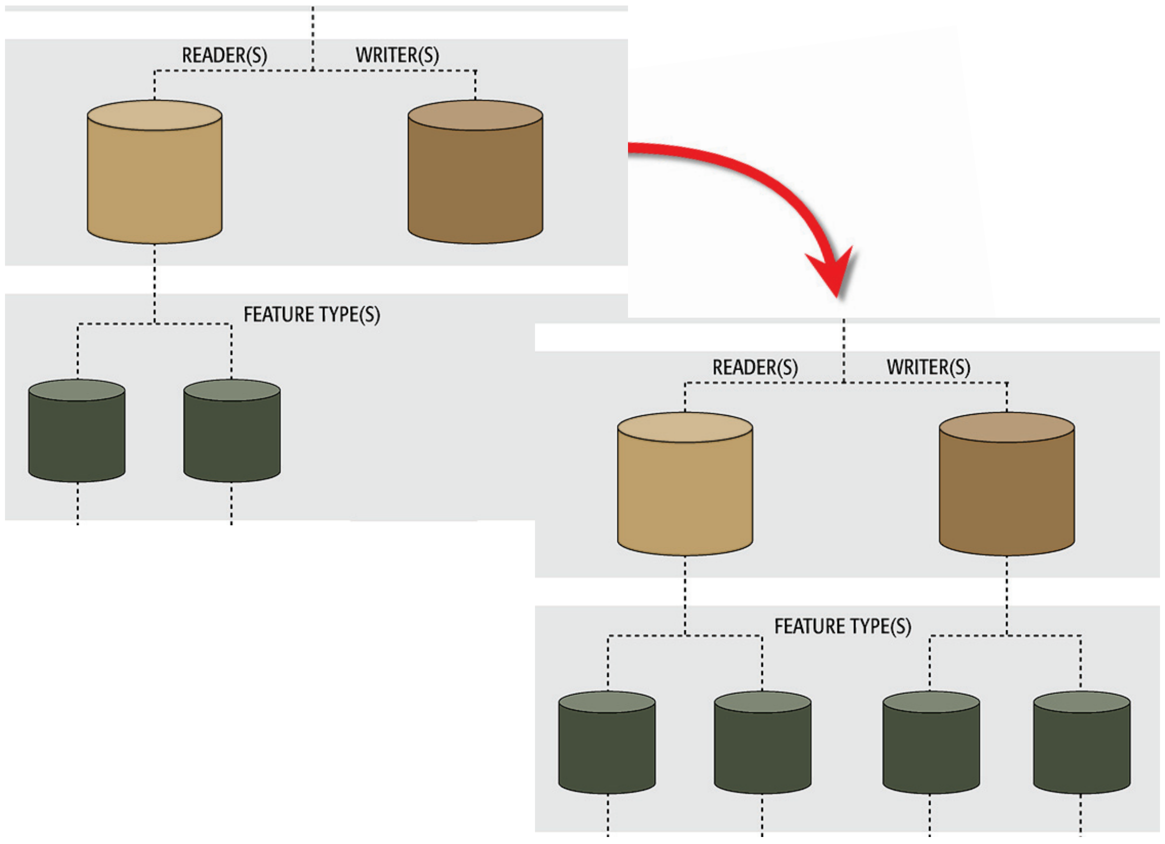
In some cases a user will have a reader with multiple feature types, and wish to add the same ones to a writer. This is simply done by selecting the source feature types, right-clicking them, and using the option Duplicate (on Writer).



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

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

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

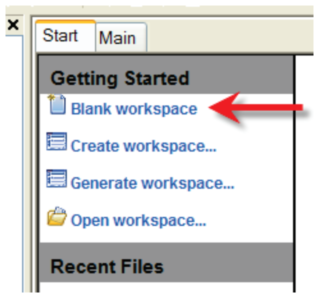


| Example 2: CAD to GIS | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Roads (MicroStation Design v8, Autodesk MapGuide Enterprise SDF) |
| **Overall Goal** | Add CAD data from the engineering department to a public GIS dataset |
| **This Step** | Add the required components to a new workspace |
| **Demonstrates** | Adding readers, adding writers |
| **Finished Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example2Complete.fmw |

After checking that the FME installation is suitable for the two formats, the next task is to create the workspace. Rather than generate a new workspace, this example will start with a blank canvas and just add components one by one.

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

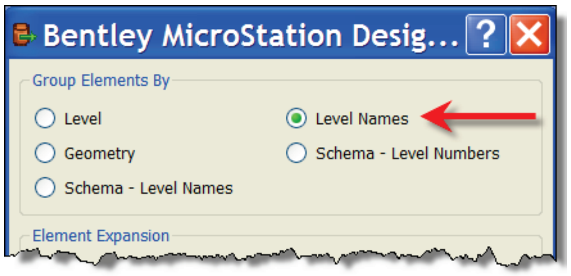
Since SDF format can be treated like a database, the translation can simply add data to it, rather than overwriting the whole file. That simplifies the workspace because there's no need to read the SDF data every time just to add it back to a new version of the file.



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

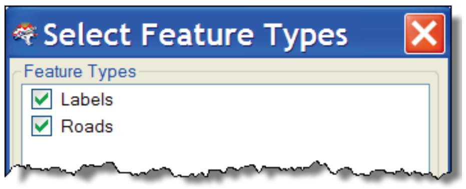
**2) Add a Reader**
Now start adding components by selecting Readers > Add Reader from the menubar. Fill in the Add Reader dialog as follows:

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



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

**3) Choose Feature Types**
When prompted, select both Roads and Labels as the feature types to add to the workspace.

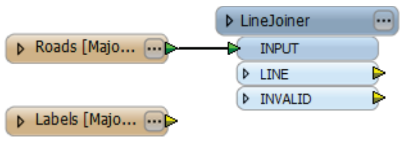


### 4) Clean Up Linework
When you inspected the source data before starting this example (did you inspect it?) you may have noticed that all road features are split up at intersection points.

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

Connect a *LineJoiner* transformer to the reader feature type DGNV8:Roads.

Default transformer parameters are sufficient for this example.



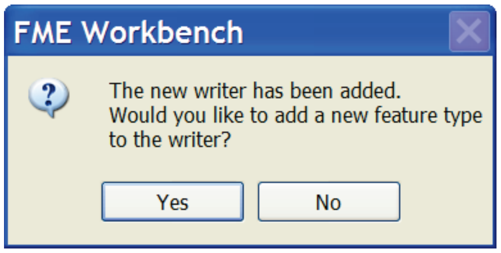
### 5) Add a Writer
Now add a writer by selecting Writers > Add Writer from the menubar.
Fill in the Add Writer dialog as follows:

| | |
|---|---|
| **Writer Format** | Autodesk MapGuide Enterprise SDF |
| **Writer Dataset** | *C:\FMEData\Data\Transit\Transit.sdf* |

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**.

Remember, it is better to copy or import feature types, rather than add them, if they already exist in a dataset.


FME Workbench

The new writer has been added.
Would you like to add a new feature type to the writer?

### 6) 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. It will also be automatically connected. Disconnect the two Roads' feature types and connect the writer one up to the LineJoiner:LINE output port.

### 7) Check Feature Type Parameters
Open the properties dialog for the SDF:Roads feature type.

Ensure the Allowed Geometries parameter is set to fdo_line

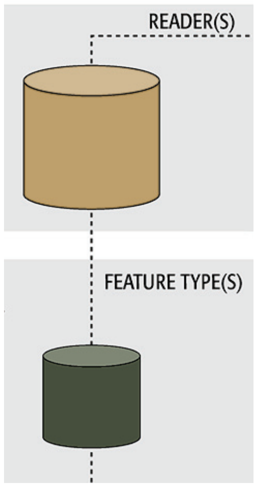
In the user attributes tab, remove any attributes (such as igds_class, igds_color) which have just been copied from MicroStation format attributes.

### 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.

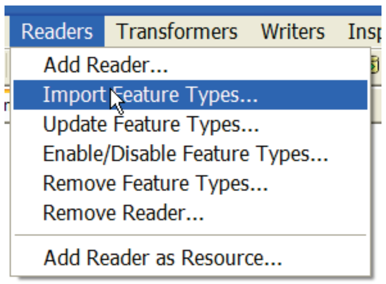
What the import tool does is essentially take a dataset's schema, and add it to a workspace.



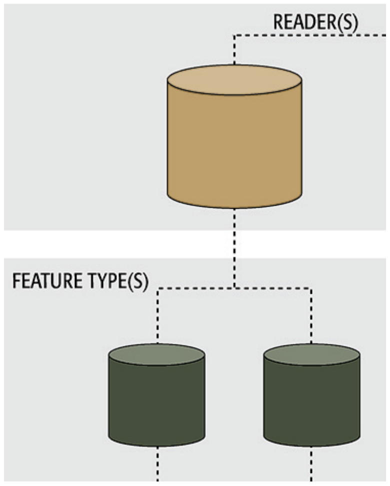
For example, a user has a workspace reading from a spatial database.

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.

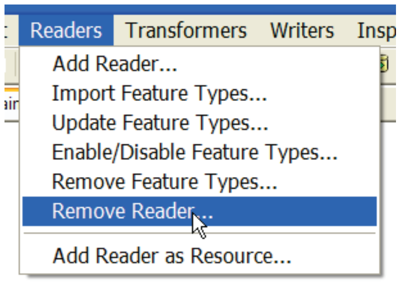
Interestingly, the import tool can import schemas from a dataset without that dataset being part of the actual translation. Even a different format is no impediment to using a schema this way.

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.



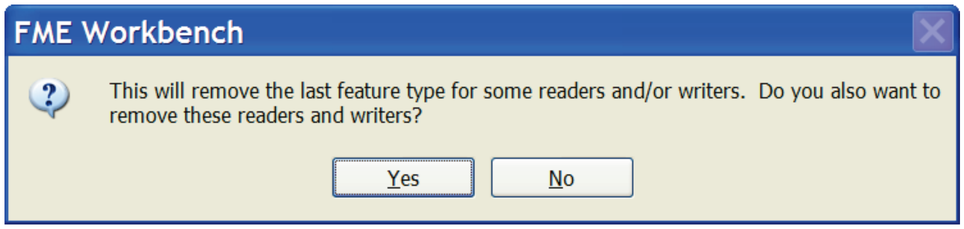
Note that whenever a reader or writer is removed, then all the related feature types will also be removed.

## Remove Feature Types

This remove 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.


FME Workbench

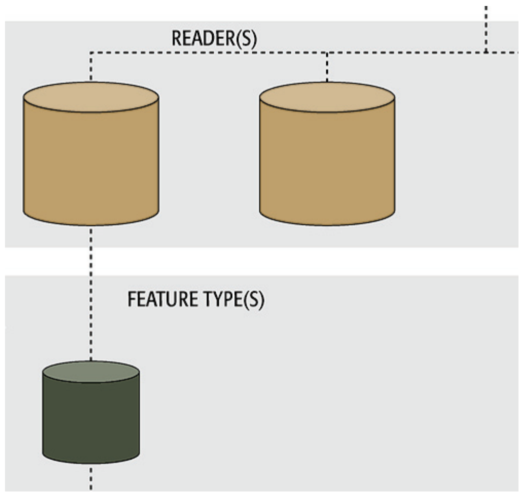
This will remove the last feature type for some readers and/or writers. Do you also want to remove these readers and writers?

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!***

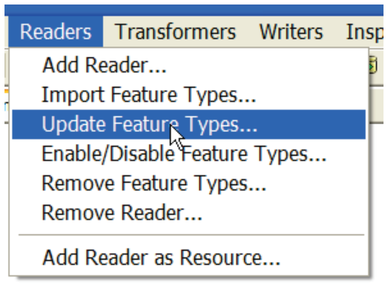
***There is no adverse effect on the output, but performance suffers because the reader will still read the source data (even though it is immediately discarded).***



## Update Feature Types

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

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



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

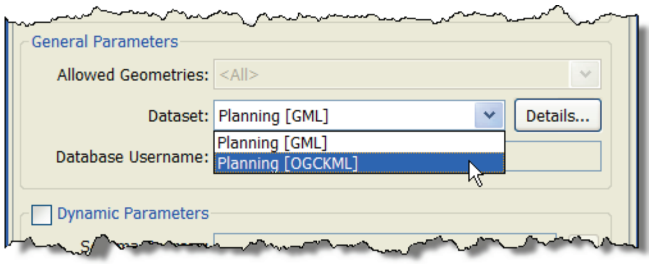
Both are a little like the Import Feature Type tool, except that the external schema is used to update a translation component of the same name, rather than to add a new one.

## Move Feature Type

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 you move the feature type from one writer to another.

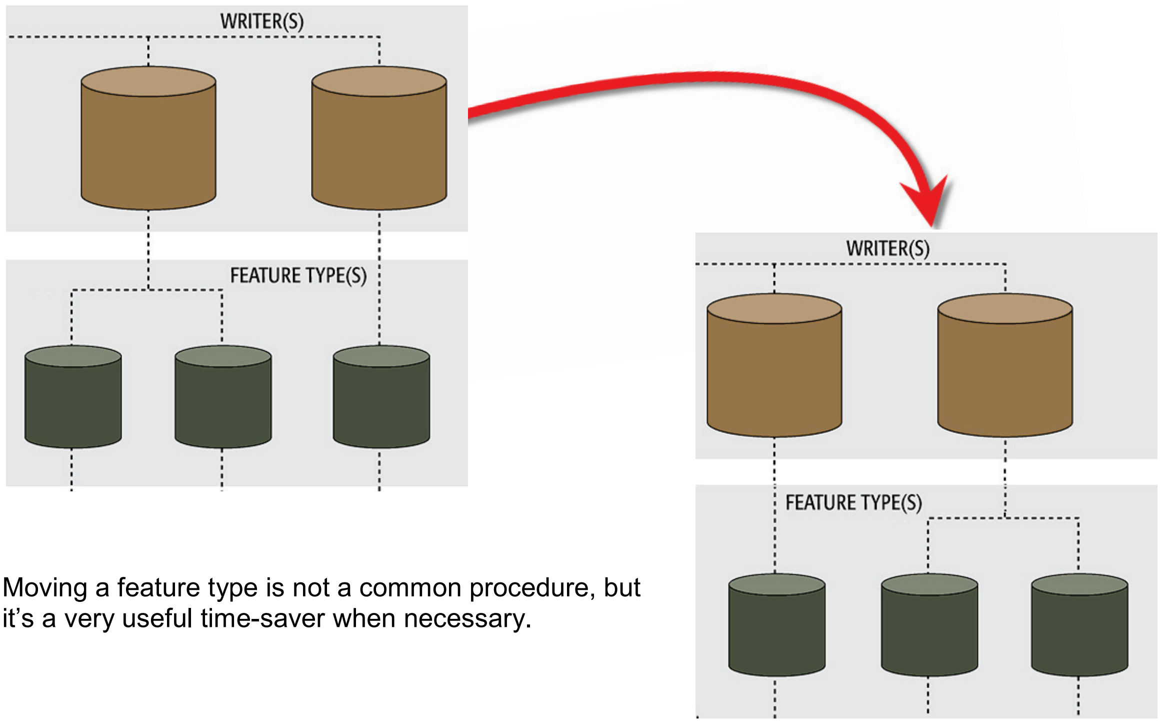


General Parameters

Allowed Geometries: <All>

Dataset: Planning [GML]

Database Username:



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

| Example 3: CAD to GIS | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Airport ( CITS Data Transfer Format (QLF), Autodesk MapGuide Enterprise SDF) |
| **Overall Goal** | Add CAD data from the engineering department to a public GIS dataset |
| **This Step** | Import feature types to a workspace |
| **Demonstrates** | Adding readers, importing feature types |
| **Starting Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example3Begin.fmw |
| **Finished Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example3Complete.fmw |

The next step in this CAD to GIS task is to convert some QLF data representing an airport, to the MapGuide SDF database. This time, because a table already exists in the SDF dataset, import feature type will be used.

**1) Start Workbench**
Start Workbench (if necessary) and open the workspace from Example 2.
Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example3Begin.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.

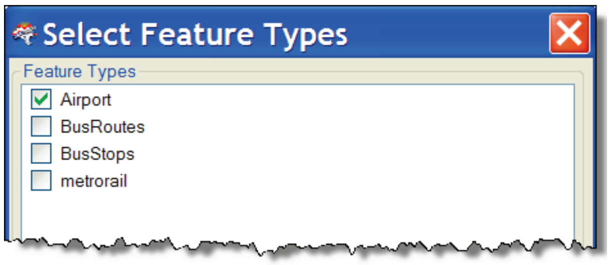
**3) Import a Writer Feature Type**
Select Writers > Import Feature Type from the menubar.

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

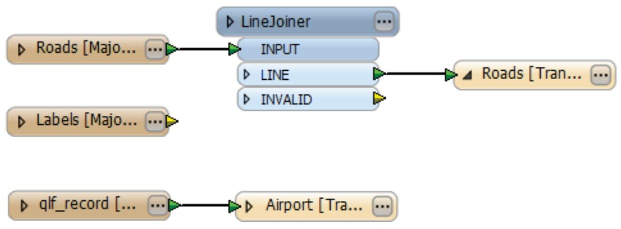
| | |
|---|---|
| **Writer Format** | Autodesk MapGuide Enterprise SDF |
| **Writer Dataset** | *C:\FMEData\Data\Transit\Transit.sdf* |
| | |
| **Parameters** | Remove Schema Qualifier = Yes |

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

If all types are preceded by "Default." then you didn't check Remove Schema Qualifier.

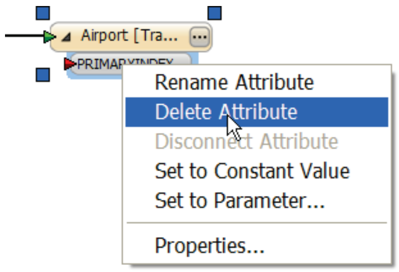


**4) Map Schema**
Map the reader feature type for Airport (it should be named *qlf_record*) to the writer feature type for Airport.



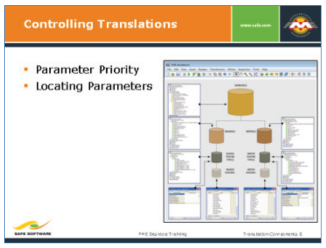
**5) Edit Schema**
The Airport feature type will have been imported with an attribute called PRIMARYINDEX. This needs to be removed, because it is a field that will be populated automatically, and does not need to be set as a user attribute.

Delete the attribute PRIMARYINDEX from the writer feature type Airport.



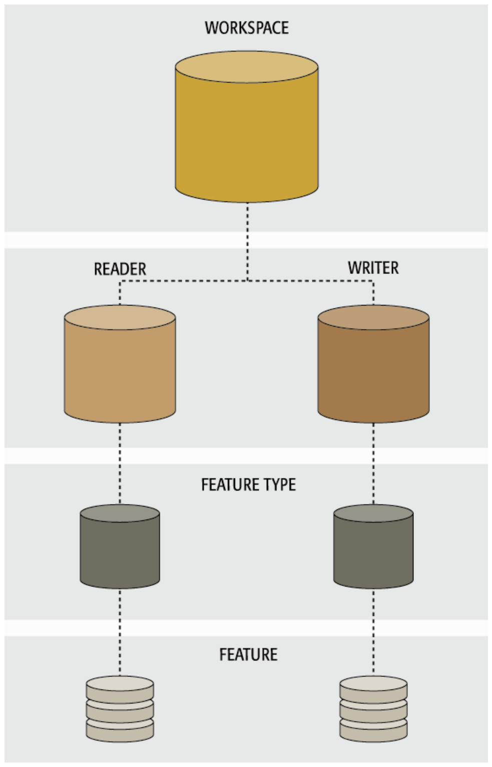
Save the workspace, but again, **DO NOT** run it yet!

## Controlling Translations


Controlling Translations

- Parameter Priority
- Locating Parameters

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



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

So there are parameters that control the workspace itself, parameters that control Readers and Writers, parameters that control feature types, and parameters that control actual features.

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

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

For example, a parameter set at the Reader level, affects all feature types that belong to that Reader.

This 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.

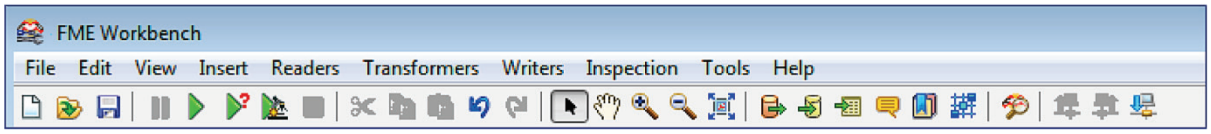
Finally, database writing mode can be set on individual features. Different features can be used to insert, update, or delete records – simultaneously – in the same table.

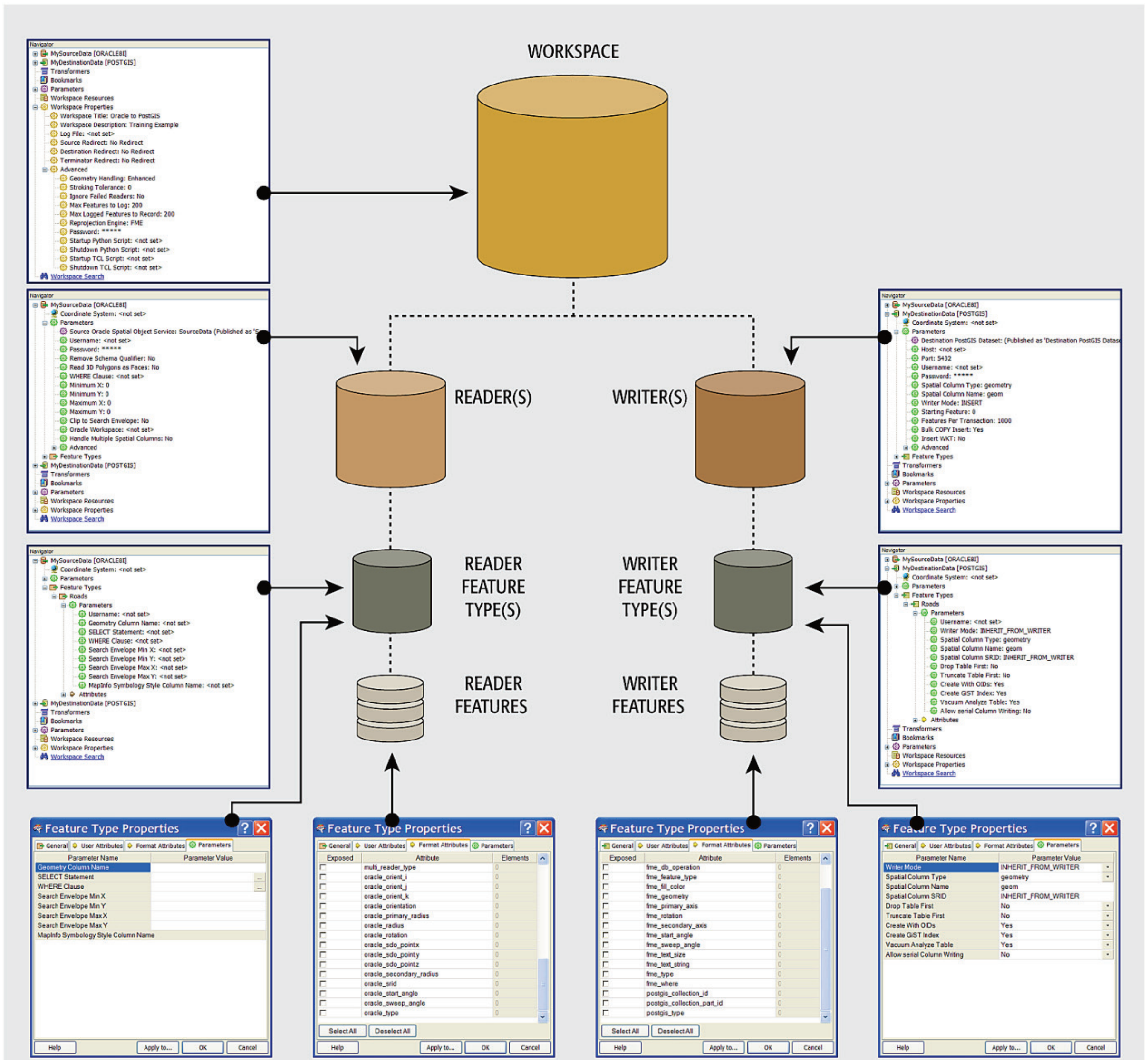
Interestingly, the higher-up parameter only applies when the lower-down parameters are not set. When the same parameter is set at different levels, then the lower-level parameter wins out.

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.

## 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.





```
Navigator
MySourceData [ORACLE8I]
MyDestinationData [POSTGIS]
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
    Workspace Title: Oracle to PostGIS
    Workspace Description: Training Example
    Log File: <not set>
    Source Redirect: No Redirect
    Destination Redirect: No Redirect
    Terminator Redirect: No Redirect
    Advanced
        Geometry Handling: Enhanced
        Stroking Tolerance: 0
        Ignore Failed Readers: No
        Max Features to Log: 200
        Max Logged Features to Record: 200
        Reprojection Engine: FME
        Password: *****
        Startup Python Script: <not set>
        Shutdown Python Script: <not set>
        Startup TCL Script: <not set>
        Shutdown TCL Script: <not set>
Workspace Search
```

```
Navigator
MySourceData [ORACLE8I]
    Coordinate System: <not set>
    Parameters
        Source Oracle Spatial Object Service: SourceData (Published as 'S
        Username: <not set>
        Password: *****
        Remove Schema Qualifier: No
        Read 3D Polygons as Faces: No
        WHERE Clause: <not set>
        Minimum X: 0
        Minimum Y: 0
        Maximum X: 0
        Maximum Y: 0
        Clip to Search Envelope: No
        Oracle Workspace: <not set>
        Handle Multiple Spatial Columns: No
        Advanced
    Feature Types
MyDestinationData [POSTGIS]
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

```
Navigator
MySourceData [ORACLE8I]
    Coordinate System: <not set>
    Parameters
    Feature Types
        Roads
            Parameters
                Username: <not set>
                Geometry Column Name: <not set>
                SELECT Statement: <not set>
                WHERE Clause: <not set>
                Search Envelope Min X: <not set>
                Search Envelope Min Y: <not set>
                Search Envelope Max X: <not set>
                Search Envelope Max Y: <not set>
                MapInfo Symbology Style Column Name: <not set>
            Attributes
MyDestinationData [POSTGIS]
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

```
Navigator
MySourceData [ORACLE8I]
MyDestinationData [POSTGIS]
    Coordinate System: <not set>
    Parameters
        Destination PostGIS Dataset: (Published as 'Destination PostGIS Datas
        Host: <not set>
        Port: 5432
        Username: <not set>
        Password: *****
        Spatial Column Type: geometry
        Spatial Column Name: geom
        Writer Mode: INSERT
        Starting Feature: 0
        Features Per Transaction: 1000
        Bulk COPY Insert: Yes
        Insert WKT: No
        Advanced
    Feature Types
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

```
Navigator
MySourceData [ORACLE8I]
MyDestinationData [POSTGIS]
    Coordinate System: <not set>
    Parameters
    Feature Types
        Roads
            Parameters
                Username: <not set>
                Writer Mode: INHERIT_FROM_WRITER
                Spatial Column Type: geometry
                Spatial Column Name: geom
                Spatial Column SRID: INHERIT_FROM_WRITER
                Drop Table First: No
                Truncate Table First: No
                Create With OIDs: Yes
                Create GiST Index: Yes
                Vacuum Analyze Table: Yes
                Allow serial Column Writing: No
            Attributes
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

Feature Type Properties (General | User Attributes | Format Attributes | Parameters)

| Parameter Name | Parameter Value |
|---|---|
| Geometry Column Name | |
| SELECT Statement | |
| WHERE Clause | |
| Search Envelope Min X | |
| Search Envelope Min Y | |
| Search Envelope Max X | |
| Search Envelope Max Y | |
| MapInfo Symbology Style Column Name | |

Feature Type Properties (General | User Attributes | Format Attributes | Parameters)

| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | multi_reader_type | 0 |
| ☐ | oracle_orient_i | 0 |
| ☐ | oracle_orient_j | 0 |
| ☐ | oracle_orient_k | 0 |
| ☐ | oracle_orientation | 0 |
| ☐ | oracle_primary_radius | 0 |
| ☐ | oracle_radius | 0 |
| ☐ | oracle_rotation | 0 |
| ☐ | oracle_sdo_pointx | 0 |
| ☐ | oracle_sdo_pointy | 0 |
| ☐ | oracle_sdo_pointz | 0 |
| ☐ | oracle_secondary_radius | 0 |
| ☐ | oracle_srid | 0 |
| ☐ | oracle_start_angle | 0 |
| ☐ | oracle_sweep_angle | 0 |
| ☐ | oracle_type | 0 |

Feature Type Properties (General | User Attributes | Format Attributes | Parameters)

| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | fme_db_operation | 0 |
| ☐ | fme_feature_type | 0 |
| ☐ | fme_fill_color | 0 |
| ☐ | fme_geometry | 0 |
| ☐ | fme_primary_axis | 0 |
| ☐ | fme_rotation | 0 |
| ☐ | fme_secondary_axis | 0 |
| ☐ | fme_start_angle | 0 |
| ☐ | fme_sweep_angle | 0 |
| ☐ | fme_text_size | 0 |
| ☐ | fme_text_string | 0 |
| ☐ | fme_type | 0 |
| ☐ | fme_where | 0 |
| ☐ | postgis_collection_id | 0 |
| ☐ | postgis_collection_part_id | 0 |
| ☐ | postgis_type | 0 |

Feature Type Properties (General | User Attributes | Format Attributes | Parameters)

| Parameter Name | Parameter Value |
|---|---|
| Writer Mode | INHERIT_FROM_WRITER |
| Spatial Column Type | geometry |
| Spatial Column Name | geom |
| Spatial Column SRID | INHERIT_FROM_WRITER |
| Drop Table First | No |
| Truncate Table First | No |
| Create With OIDs | Yes |
| Create GiST Index | Yes |
| Vacuum Analyze Table | Yes |
| Allow serial Column Writing | No |

| **Parameter** | **Location** |
|---|---|
| Workspace Parameters | Navigator Window |
| Reader and Writer Parameters | Navigator Window |
| Feature Type Parameters | Navigator Window *and* Feature Type Properties dialog |
| Feature Parameters | Feature Type Properties dialog |

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

## Workspace Parameters

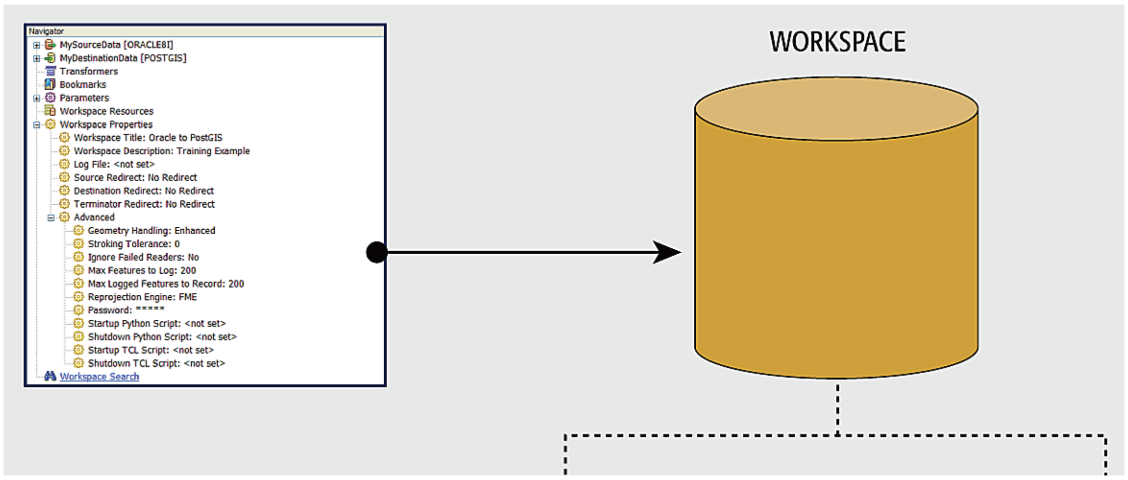
Workspace Parameters

- Basic Workspace Parameters
- Advanced 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.



```
Navigator
⊞ MySourceData [ORACLE8I]
⊞ MyDestinationData [POSTGIS]
  Transformers
  Bookmarks
⊞ Parameters
  Workspace Resources
⊟ Workspace Properties
    Workspace Title: Oracle to PostGIS
    Workspace Description: Training Example
    Log File: <not set>
    Source Redirect: No Redirect
    Destination Redirect: No Redirect
    Terminator Redirect: No Redirect
  ⊟ Advanced
      Geometry Handling: Enhanced
      Stroking Tolerance: 0
      Ignore Failed Readers: No
      Max Features to Log: 200
      Max Logged Features to Record: 200
      Reprojection Engine: FME
      Password: *****
      Startup Python Script: <not set>
      Shutdown Python Script: <not set>
      Startup TCL Script: <not set>
      Shutdown TCL Script: <not set>
  Workspace Search
```

Not all workspace parameters have an effect on the translation. Some settings are there only for reasons of Best Practice; for example, Workspace Title and Workspace Description. Others have a vital effect on how the translation is performed.

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

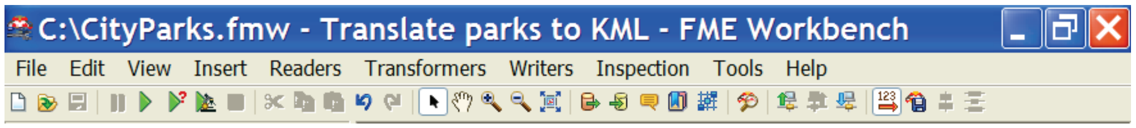
## Basic Workspace Parameters

There are a number of basic workspace parameters. The most important ones are:

### *Workspace Title*

The title of a workspace is purely for information purposes. The default title contains information about the Reader and Writer format, but this parameter permits it to be changed to anything.

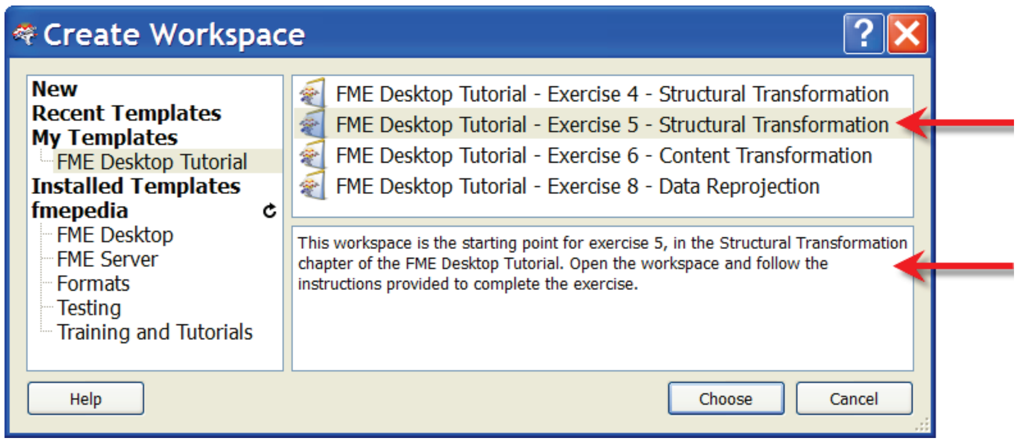
Workspace title is shown on the title bar of the Workbench window.


C:\CityParks.fmw - Translate parks to KML - FME Workbench
File Edit View Insert Readers Transformers Writers Inspection Tools Help

### *Workspace Description*

This is another information-only parameter, and allows a user to enter a description of what a workspace does.

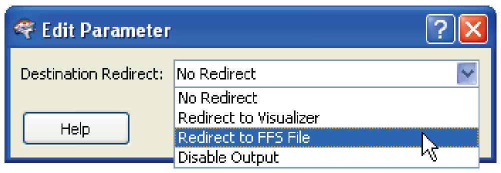
Workspace title and description are vitally important when either exporting the workspace as a template, or using the workspace on FME Server. This is because title and description are used as identifiers in various GUI components, such as the Create Workspace dialog.


Create Workspace
New
Recent Templates
My Templates
FME Desktop Tutorial
Installed Templates
fmepedia
FME Desktop
FME Server
Formats
Testing
Training and Tutorials
FME Desktop Tutorial - Exercise 4 - Structural Transformation
FME Desktop Tutorial - Exercise 5 - Structural Transformation
FME Desktop Tutorial - Exercise 6 - Content Transformation
FME Desktop Tutorial - Exercise 8 - Data Reprojection
Help
Choose
Cancel

This workspace is the starting point for exercise 5, in the Structural Transformation chapter of the FME Desktop Tutorial. Open the workspace and follow the instructions provided to complete the exercise.

### *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 Visualizer***: 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 (equivalent to a Null dataset).

***Above****: Workspace Parameters – Destination Redirect options.*

***'Redirect to Visualizer' 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 processing techniques for geometric transformations.

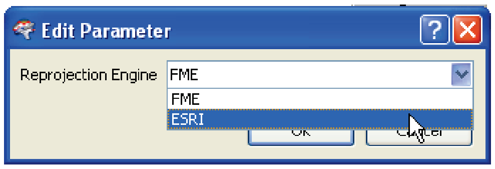
This parameter forces FME to use an older set of processing methods. It is used primarily to ensure backwards compatibility, when upgrading from a much older version of FME.

### *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 your existing data, this parameter permits a user to use the reprojection engine from a different application.



Reprojection Engine: FME

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

### *Password*

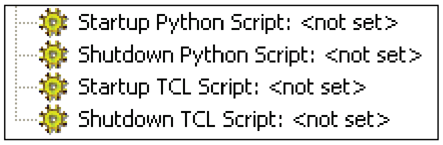
It's often desirable to pass a workspace to an FME user for them to run, but not to edit. A password-protected workspace cannot be opened for editing in Workbench without the password. It can, however, still be run within the FME Universal Translator or from the command line.

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.

***Right**: Script parameters in the workspace settings dialog:*

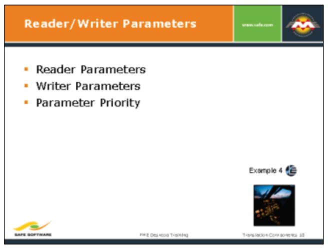


Startup Python Script: <not set>
Shutdown Python Script: <not set>
Startup TCL Script: <not set>
Shutdown TCL Script: <not set>

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



Reader/Writer Parameters

- Reader Parameters
- Writer Parameters
- Parameter Priority

Example 4

***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.

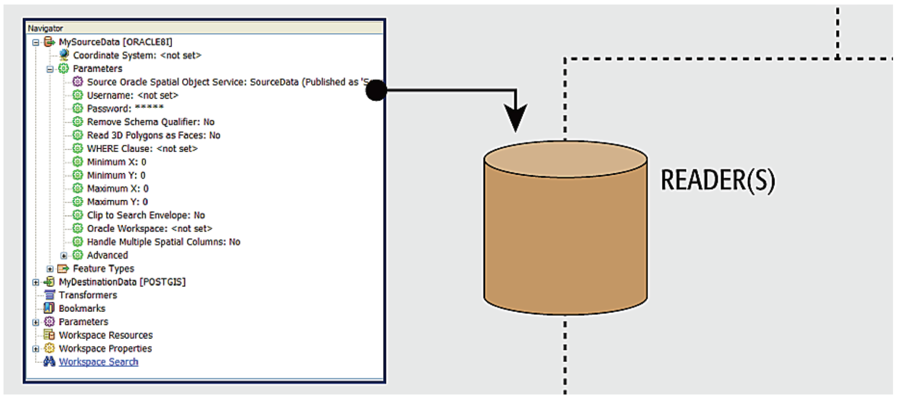
Because these parameters refer to specific components and characteristics of the related format, no two formats will have the same set of control parameters.

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

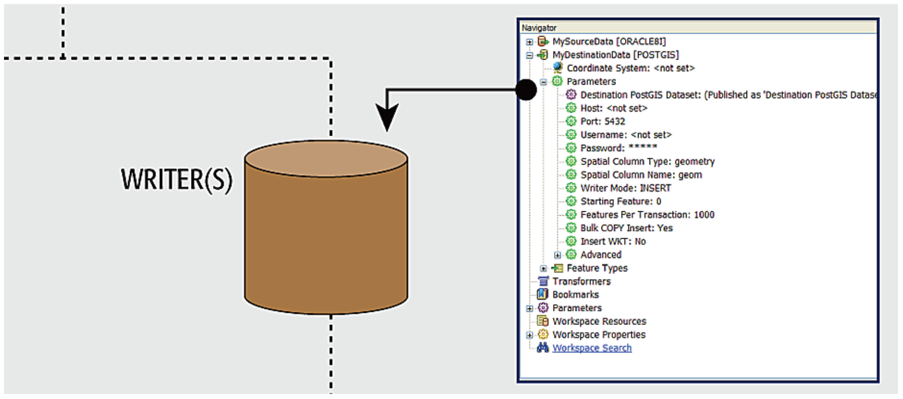
Reader parameters are shown and set in the Navigator Window.
To edit a parameter, double-click it. A dialog opens up where the parameter's value may be set.



```
Navigator
MySourceData [ORACLE8I]
    Coordinate System: <not set>
    Parameters
        Source Oracle Spatial Object Service: SourceData (Published as 'S
        Username: <not set>
        Password: *****
        Remove Schema Qualifier: No
        Read 3D Polygons as Faces: No
        WHERE Clause: <not set>
        Minimum X: 0
        Minimum Y: 0
        Maximum X: 0
        Maximum Y: 0
        Clip to Search Envelope: No
        Oracle Workspace: <not set>
        Handle Multiple Spatial Columns: No
        Advanced
    Feature Types
MyDestinationData [POSTGIS]
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

## 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.



```
Navigator
MySourceData [ORACLE8I]
MyDestinationData [POSTGIS]
    Coordinate System: <not set>
    Parameters
        Destination PostGIS Dataset: (Published as 'Destination PostGIS Datase
        Host: <not set>
        Port: 5432
        Username: <not set>
        Password: *****
        Spatial Column Type: geometry
        Spatial Column Name: geom
        Writer Mode: INSERT
        Starting Feature: 0
        Features Per Transaction: 1000
        Bulk COPY Insert: Yes
        Insert WKT: No
        Advanced
    Feature Types
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

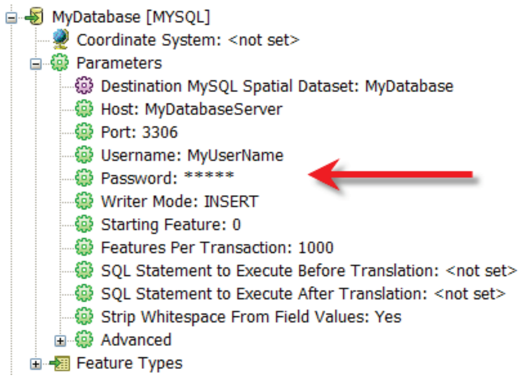
### *Parameter Priority*

It's worth emphasising 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.

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



```
MyDatabase [MYSQL]
    Coordinate System: <not set>
    Parameters
        Destination MySQL Spatial Dataset: MyDatabase
        Host: MyDatabaseServer
        Port: 3306
        Username: MyUserName
        Password: *****
        Writer Mode: INSERT
        Starting Feature: 0
        Features Per Transaction: 1000
        SQL Statement to Execute Before Translation: <not set>
        SQL Statement to Execute After Translation: <not set>
        Strip Whitespace From Field Values: Yes
        Advanced
    Feature Types
```

| Example 4: CAD to GIS | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Airport ( CITS Data Transfer Format (QLF), Autodesk MapGuide Enterprise SDF) |
| **Overall Goal** | Add CAD data from the engineering department to a public GIS dataset |
| **This Step** | Preview translation results, and adjust “over-write output” parameter |
| **Demonstrates** | Workspace Parameters, Reader/Writer Parameters |
| **Starting Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example4Begin.fmw |
| **Finished Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example4Complete.fmw |

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

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

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

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

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

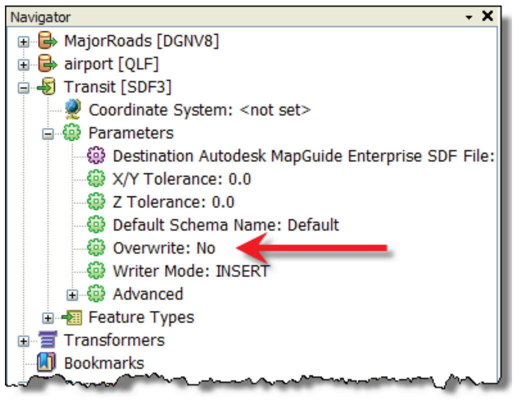
Changing this setting to ‘Redirect to Visualizer’ is the equivalent to using Writers > Redirect to Visualizer. In fact, you can try this to show how changing one automatically affects the other.

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.

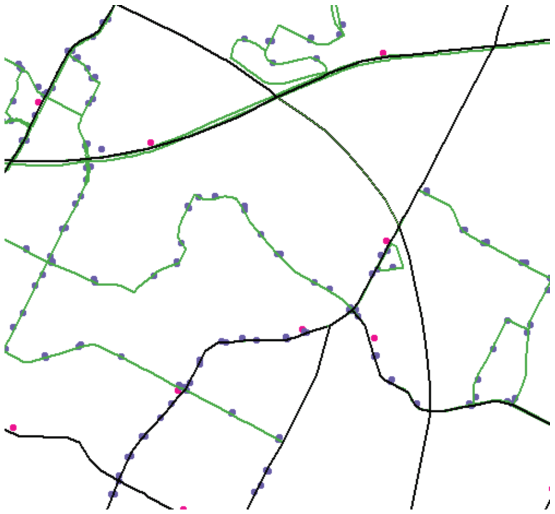
Experiment with setting the Destination Redirect option to either Redirect to Visualizer or Disable Output, and running the workspace.

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

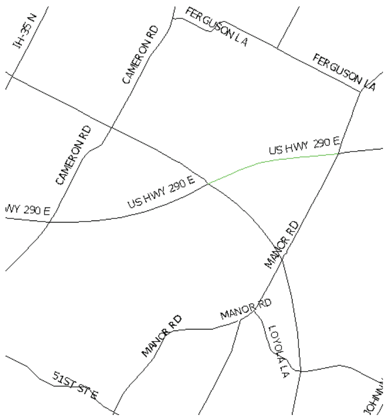
In the Navigator window locate the writer parameter **Overwrite file**. Make sure it’s set to **No**.


Navigator
MajorRoads [DGNV8]
airport [QLF]
Transit [SDF3]
Coordinate System: <not set>
Parameters
Destination Autodesk MapGuide Enterprise SDF File:
X/Y Tolerance: 0.0
Z Tolerance: 0.0
Default Schema Name: Default
Overwrite: No
Writer Mode: INSERT
Advanced
Feature Types
Transformers
Bookmarks

**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.



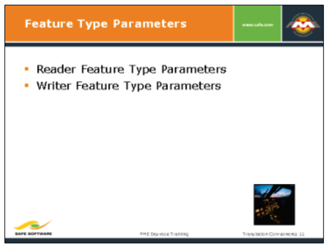
Inspect the SDF dataset contents to make sure the original data still exists and that the roads have been successfully added to it.



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.

***Note:*** *SDF datasets sometimes get locked when in use. If the workspace fails to write the data, close the FME Viewer and try again. Writing will fail if the data is locked in another application.*

## Feature Type Parameters


Feature Type Parameters

- Reader Feature Type Parameters
- Writer 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

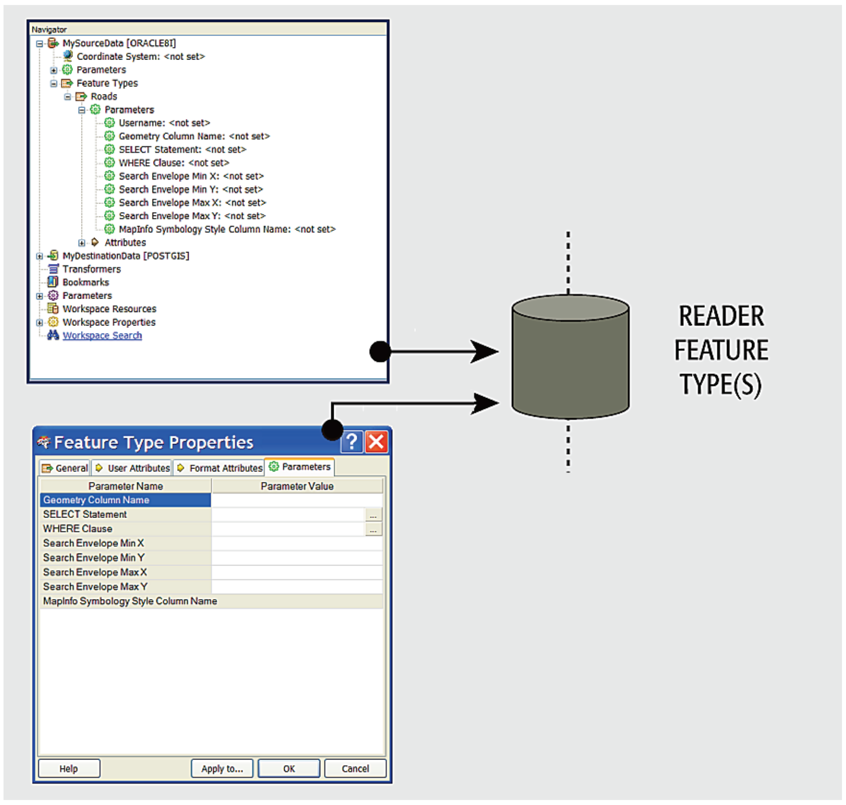
Feature Types are at a lower level in the hierarchy than readers and writers.

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.



```
MySourceData [ORACLE8I]
    Coordinate System: <not set>
    Parameters
    Feature Types
        Roads
            Parameters
                Username: <not set>
                Geometry Column Name: <not set>
                SELECT Statement: <not set>
                WHERE Clause: <not set>
                Search Envelope Min X: <not set>
                Search Envelope Min Y: <not set>
                Search Envelope Max X: <not set>
                Search Envelope Max Y: <not set>
                MapInfo Symbology Style Column Name: <not set>
            Attributes
MyDestinationData [POSTGIS]
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

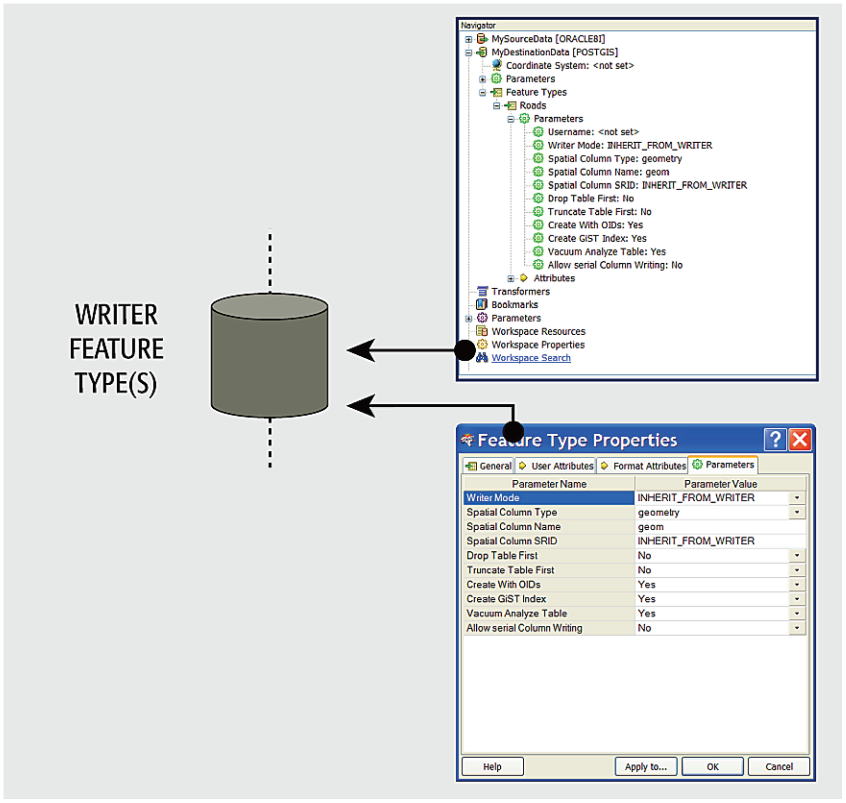
| Parameter Name | Parameter Value |
|---|---|
| Geometry Column Name | |
| SELECT Statement | ... |
| WHERE Clause | ... |
| Search Envelope Min X | |
| Search Envelope Min Y | |
| Search Envelope Max X | |
| Search Envelope Max Y | |
| MapInfo Symbology Style Column Name | |

***Left**: Feature Type Parameters can also be accessed through the Feature Type Properties dialog. Notice the tab named **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 reader feature type parameters, but a high proportion of file-based formats also have these.



```
MySourceData [ORACLE8I]
MyDestinationData [POSTGIS]
    Coordinate System: <not set>
    Parameters
    Feature Types
        Roads
            Parameters
                Username: <not set>
                Writer Mode: INHERIT_FROM_WRITER
                Spatial Column Type: geometry
                Spatial Column Name: geom
                Spatial Column SRID: INHERIT_FROM_WRITER
                Drop Table First: No
                Truncate Table First: No
                Create With OIDs: Yes
                Create GiST Index: Yes
                Vacuum Analyze Table: Yes
                Allow serial Column Writing: No
            Attributes
Transformers
Bookmarks
Parameters
Workspace Resources
Workspace Properties
Workspace Search
```

| Parameter Name | Parameter Value |
|---|---|
| Writer Mode | INHERIT_FROM_WRITER |
| Spatial Column Type | geometry |
| Spatial Column Name | geom |
| Spatial Column SRID | INHERIT_FROM_WRITER |
| Drop Table First | No |
| Truncate Table First | No |
| Create With OIDs | Yes |
| Create GiST Index | Yes |
| Vacuum Analyze Table | Yes |
| Allow serial Column Writing | No |

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

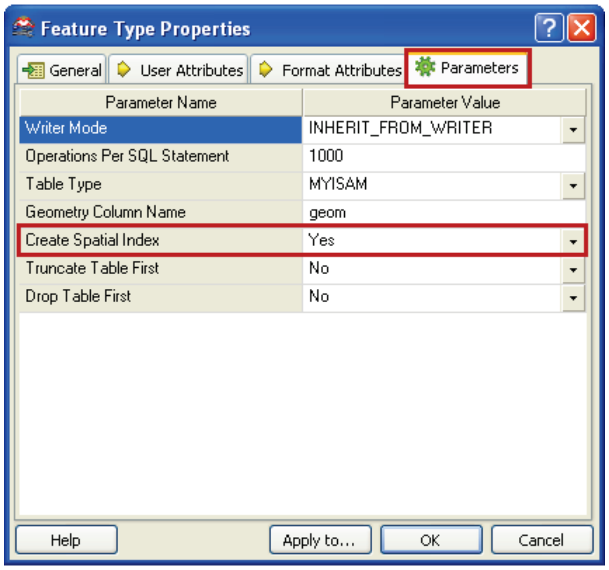
The decision about whether or not to apply an index is made on a table-by-table basis.

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

Therefore this is a feature type parameter.

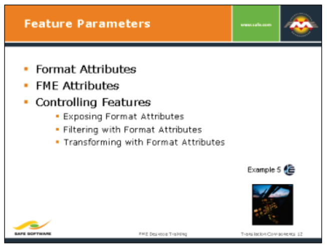
If it were a writer-level parameter, then ALL tables would get an index; not necessarily what the user wants.

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



| Parameter Name | Parameter Value |
|---|---|
| Writer Mode | INHERIT_FROM_WRITER |
| Operations Per SQL Statement | 1000 |
| Table Type | MYISAM |
| Geometry Column Name | geom |
| Create Spatial Index | Yes |
| Truncate Table First | No |
| Drop Table First | No |

## Feature Parameters



Feature Parameters

- Format Attributes
- FME Attributes
- Controlling Features
  - Exposing Format Attributes
  - Filtering with Format Attributes
  - Transforming with Format Attributes

Example 5

SAFE SOFTWARE FME Desktop Training Translation Components 12

***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.

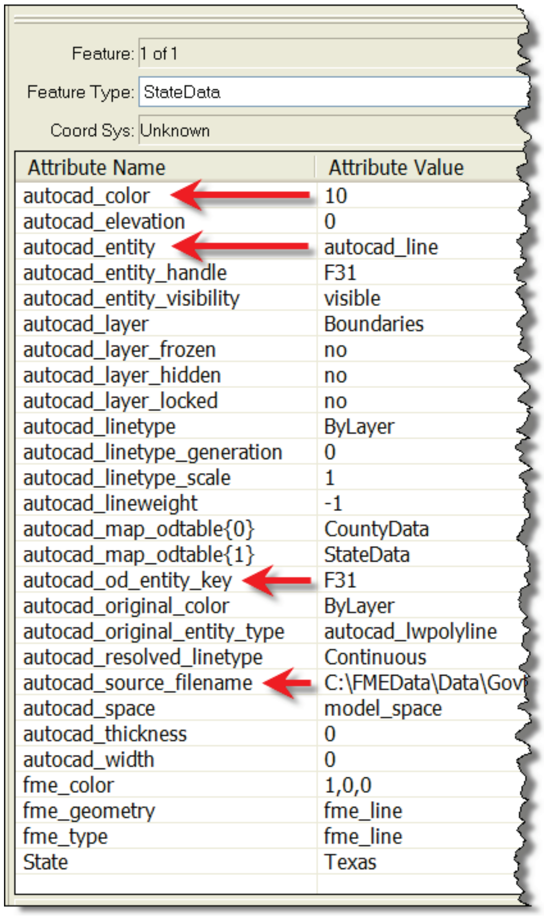
Format attributes are most obvious when viewing a dataset with FME Viewer. Querying a feature causes both user attributes and format attributes to be reported.

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

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

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

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



Feature: 1 of 1

Feature Type: StateData

Coord Sys: Unknown

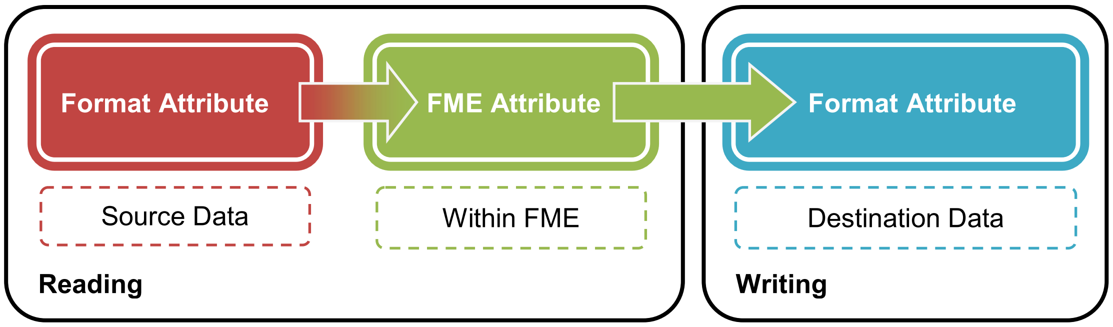
| Attribute Name | Attribute Value |
|---|---|
| autocad_color | 10 |
| autocad_elevation | 0 |
| autocad_entity | autocad_line |
| autocad_entity_handle | F31 |
| autocad_entity_visibility | visible |
| autocad_layer | Boundaries |
| autocad_layer_frozen | no |
| autocad_layer_hidden | no |
| autocad_layer_locked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

## FME Attributes

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

When a translation is carried out, the following occurs:

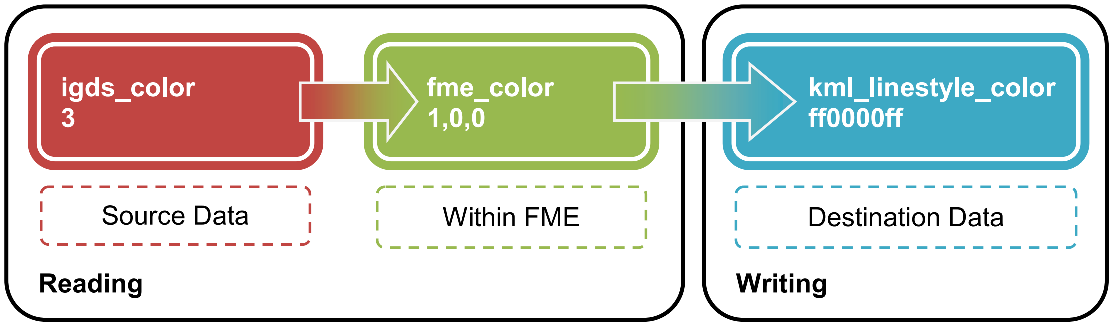
- FME reads the source data and stores information about its features as format attributes. These format attributes reflect the data that is stored in the original source data.
- FME converts the source data's format attributes into FME Attributes. These FME attributes reflect the source data as it is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

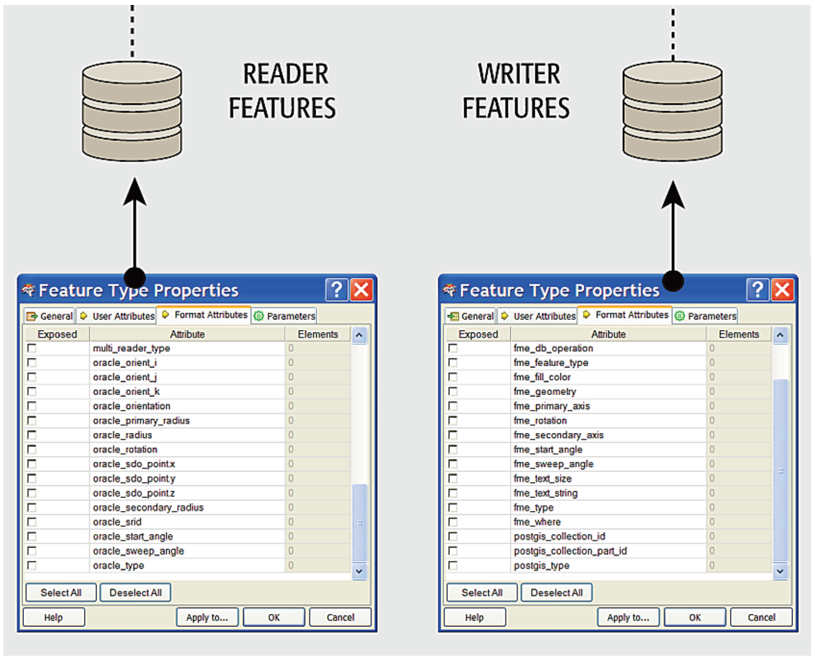
FME merely converts everything to an FME standard and then from there to the Writer Format.



## 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.



| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | multi_reader_type | 0 |
| ☐ | oracle_orient_i | 0 |
| ☐ | oracle_orient_j | 0 |
| ☐ | oracle_orient_k | 0 |
| ☐ | oracle_orientation | 0 |
| ☐ | oracle_primary_radius | 0 |
| ☐ | oracle_radius | 0 |
| ☐ | oracle_rotation | 0 |
| ☐ | oracle_sdo_point.x | 0 |
| ☐ | oracle_sdo_point.y | 0 |
| ☐ | oracle_sdo_point.z | 0 |
| ☐ | oracle_secondary_radius | 0 |
| ☐ | oracle_srid | 0 |
| ☐ | oracle_start_angle | 0 |
| ☐ | oracle_sweep_angle | 0 |
| ☐ | oracle_type | 0 |

| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | fme_db_operation | 0 |
| ☐ | fme_feature_type | 0 |
| ☐ | fme_fill_color | 0 |
| ☐ | fme_geometry | 0 |
| ☐ | fme_primary_axis | 0 |
| ☐ | fme_rotation | 0 |
| ☐ | fme_secondary_axis | 0 |
| ☐ | fme_start_angle | 0 |
| ☐ | fme_sweep_angle | 0 |
| ☐ | fme_text_size | 0 |
| ☐ | fme_text_string | 0 |
| ☐ | fme_type | 0 |
| ☐ | fme_where | 0 |
| ☐ | postgis_collection_id | 0 |
| ☐ | postgis_collection_part_id | 0 |
| ☐ | postgis_type | 0 |

### *Exposing Format Attributes*

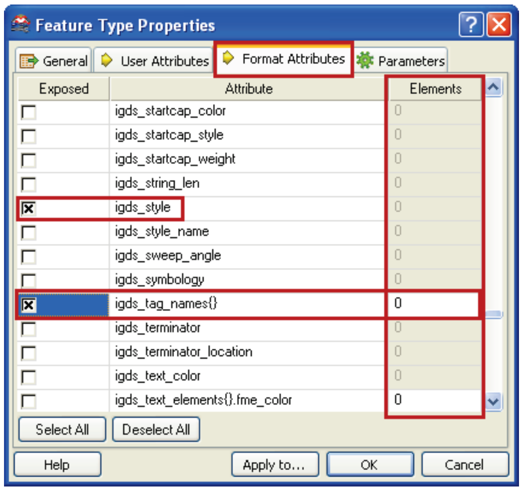
To expose a format attribute, open the Feature Type Properties dialog, and click the 'Format Attributes' tab. Locate the Format Attribute to expose and check the box provided. Click OK to make the format attribute available for use within Workbench.

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{})

For a list, the user can choose to expose not just the list, but also elements for that list by using the Elements column on the right side of the dialog.

Firefighter Mapp says…

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



| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | igds_startcap_color | 0 |
| ☐ | igds_startcap_style | 0 |
| ☐ | igds_startcap_weight | 0 |
| ☐ | igds_string_len | 0 |
| ☒ | igds_style | 0 |
| ☐ | igds_style_name | 0 |
| ☐ | igds_sweep_angle | 0 |
| ☐ | igds_symbology | 0 |
| ☒ | igds_tag_names{} | 0 |
| ☐ | igds_terminator | 0 |
| ☐ | igds_terminator_location | 0 |
| ☐ | igds_text_color | 0 |
| ☐ | igds_text_elements{}.fme_color | 0 |

### ***Filtering with Format Attributes***
One primary use of format attributes is as a means of filtering and directing source data within a workspace.

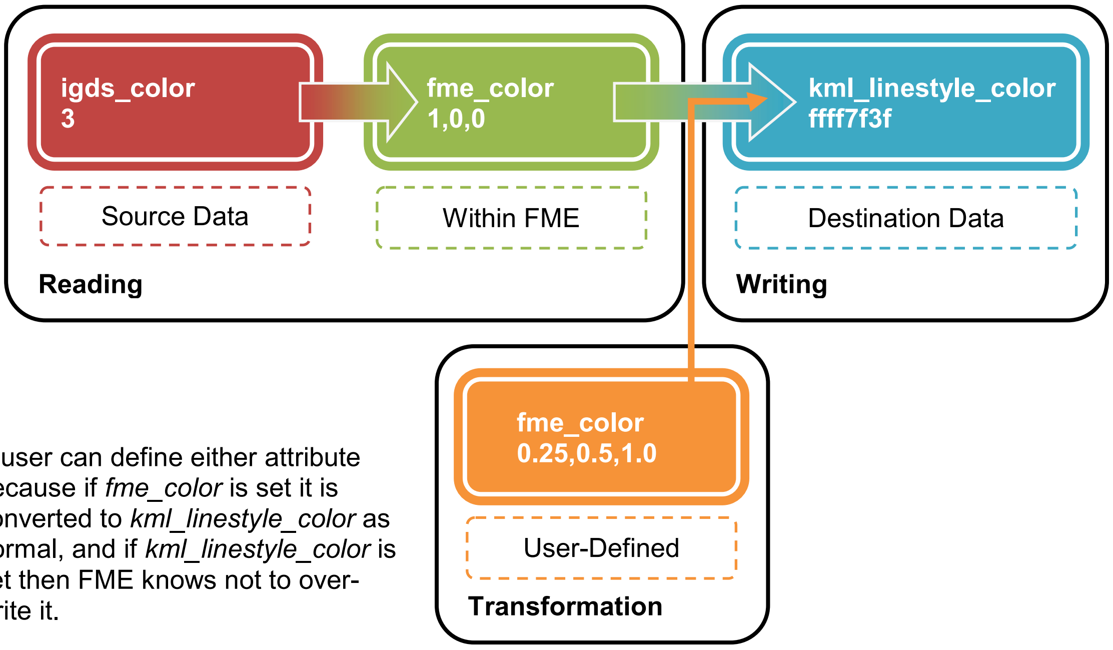
For example, suppose features in a source AutoCAD dataset are not divided into different layers as they should be. Because the user is able to determine the proper layer from maybe the color of the feature or the size of a text entity, he/she can expose the format attributes *autocad_color* or *autocad_text_size,*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

### ***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 over-write it.

If a user manages to define <u>both</u> 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.

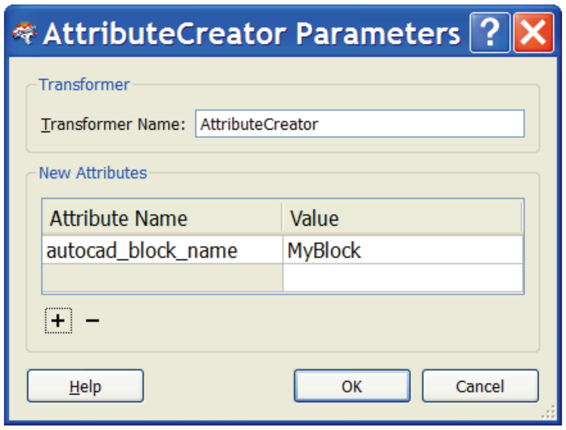
This only really becomes a problem when reading and writing the same format, and the same format attributes exists on both reader and writer. In that case use the format attribute; it's not safe to use the fme equivalents.

***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 *AttributeSetter* 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.

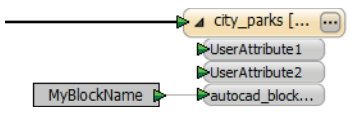
On the other hand, exposing a format attribute on a writer feature type doesn't make that attribute available in the workspace in the same way (i.e. it isn't exposed back upstream).

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



Transformer Name: AttributeCreator

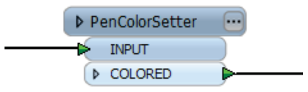
| Attribute Name | Value |
|---|---|
| autocad_block_name | MyBlock |



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.

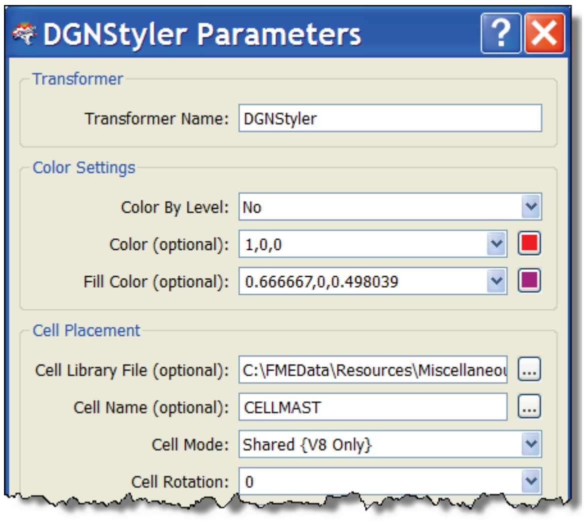


The *PenColorSetter* (for example) *"Sets the pen color of the feature"*, but in reality all it does is set a new value for the format attribute fme_color.

The *DGNStyler* is a new transformer for FME2011. It 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 *styler* transformers are:

- DWGStyler
- KMLStyler
- MapInfoStyler
- PDFStyler



Transformer Name: DGNStyler
Color By Level: No
Color (optional): 1,0,0 🟥
Fill Color (optional): 0.666667,0,0.498039 🟪
Cell Library File (optional): C:\FMEData\Resources\Miscellaneou
Cell Name (optional): CELLMAST
Cell Mode: Shared {V8 Only}
Cell Rotation: 0

| Example 5: CAD to GIS | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Airport ( CITS Data Transfer Format (QLF), Autodesk MapGuide Enterprise SDF), Updated Bus Stops (ESRI Shape) |
| **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\Session3Example5Begin.fmw |
| **Finished Workspace** | C:\FMEData\Workspaces\DesktopManual\Session3Example5Complete.fmw |

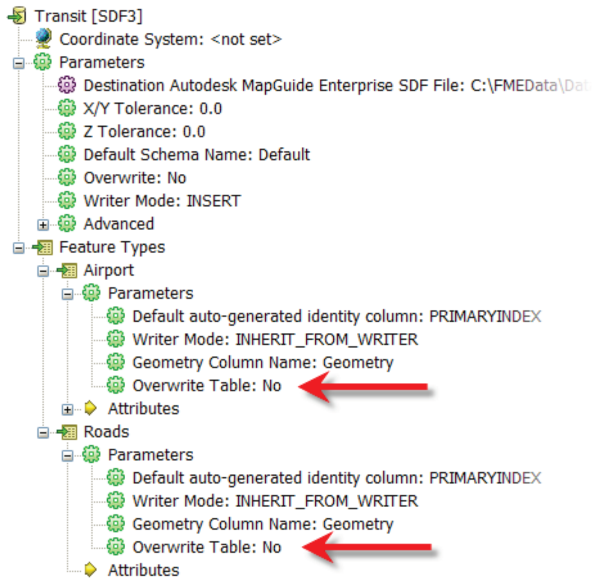
Inspecting the output from the previous example you may have noticed that the Airport table in the SDF dataset already held a dummy feature. This needs to be removed. The simplest method is to re-run the workspace, truncating the table contents before re-writing to it.

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 4.
Alternatively you can open C:\FMEData\Workspaces\DesktopManual\Session3Example5Begin.fmw

**2) Set Feature Type Parameter**
In the Navigator window, locate the SDF writer feature type parameter 'Overwrite Table' 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.



```
Transit [SDF3]
  Coordinate System: <not set>
  Parameters
    Destination Autodesk MapGuide Enterprise SDF File: C:\FMEData\Da
    X/Y Tolerance: 0.0
    Z Tolerance: 0.0
    Default Schema Name: Default
    Overwrite: No
    Writer Mode: INSERT
    Advanced
  Feature Types
    Airport
      Parameters
        Default auto-generated identity column: PRIMARYINDEX
        Writer Mode: INHERIT_FROM_WRITER
        Geometry Column Name: Geometry
        Overwrite Table: No
      Attributes
    Roads
      Parameters
        Default auto-generated identity column: PRIMARYINDEX
        Writer Mode: INHERIT_FROM_WRITER
        Geometry Column Name: Geometry
        Overwrite Table: No
      Attributes
```

**3) Run the Workspace**
At this point you can re-run the workspace to prove that the Airport dummy feature has been removed, and that the Roads table only contains a single set of geometry.

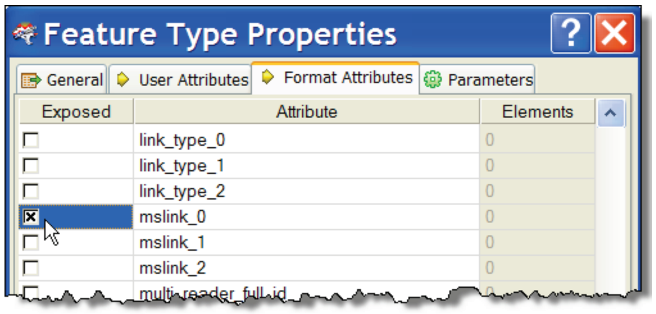
When inspecting the data you may notice that the LineJoiner transformer did not do a perfect job joining line features. That's because the LineJoiner works on geometry and lines won't be joined where there is more than one possible connection that can be made; for example, where two roads cross.

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.



| Exposed | Attribute | Elements |
|---|---|---|
| ☐ | link_type_0 | 0 |
| ☐ | link_type_1 | 0 |
| ☐ | link_type_2 | 0 |
| ☒ | mslink_0 | 0 |
| ☐ | mslink_1 | 0 |
| ☐ | mslink_2 | 0 |
| ☐ | multi_reader_full_id | 0 |

**5) Fix LineJoiner Transformer**
In the *LineJoiner* transformer set a group-by to group features by mslink_0.

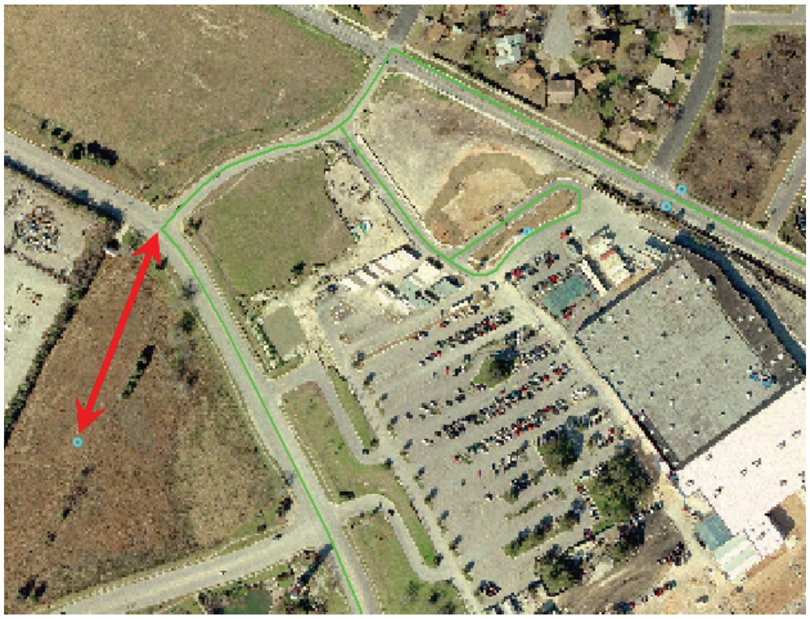
**6) Run Workspace**
Run the workspace again and you should now find that all roads with the same ID are properly joined together.

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.

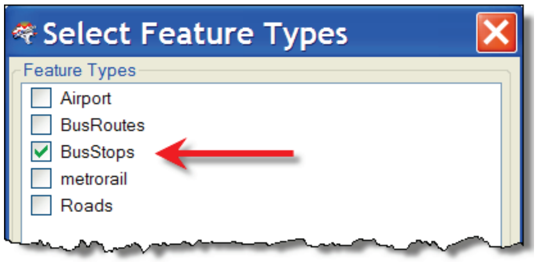


**7) Add a Reader**
Select Readers > Add Reader from the menubar. Fill in the Add Reader dialog as follows:

| | |
|---|---|
| **Reader Format** | ESRI Shape |
| **Reader Dataset** | *C:\FMEData\Data\Transit\BusStopUpdates\BusStops.shp* |

**8) Import Feature Type**
As the workspace doesn't yet have a WRITER feature type for BusStops, import it from the existing dataset Transit.sdf using Writers > Import Feature Types.

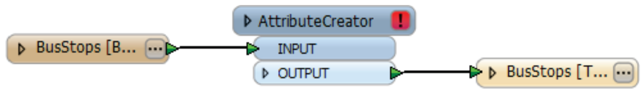
When prompted, only BusStops needs to be selected.

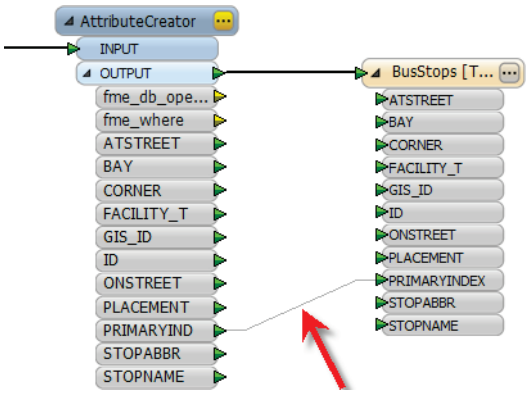


- [ ] Airport
- [ ] BusRoutes
- [x] BusStops
- [ ] metrorail
- [ ] Roads

**9) Place AttributeCreator Transformer**
An ESRI reader won't have format attributes for database operations to expose. In this case the easiest way to obtain them is to create them with an AttributeCreator transformer, as they can be set at the same time.

Place an AttributeCreator transformer and connect it between the reader and writer feature types for BusStops.





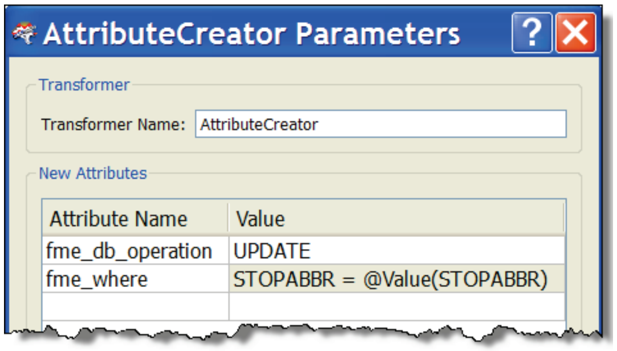
**10) Map Schema**
Map the BusStops updates reader attribute PRIMARYIND to the writer attribute PRIMARYINDEX

**11) Set AttributeCreator Parameters**
Open the AttributeCreator parameters and create two new attributes: fme_db_operation and fme_where

fme_db_operation defines the operation to be carried out on the data.
Set it to a value of: UPDATE

fme_where defines a clause which features must pass to be updated.
Set it to a value of: STOPABBR = @Value(STOPABBR)

In this case the where clause means *“where the field STOPABBR matches the value of the incoming attribute STOPABBR”*.


AttributeCreator Parameters

Transformer

Transformer Name: AttributeCreator

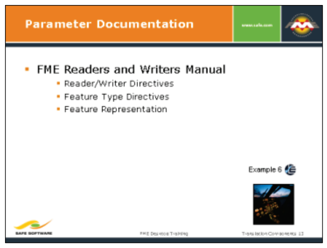
New Attributes

| Attribute Name | Value |
|---|---|
| fme_db_operation | UPDATE |
| fme_where | STOPABBR = @Value(STOPABBR) |
| | |

**12) Run Workspace**
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.

## Parameter Documentation

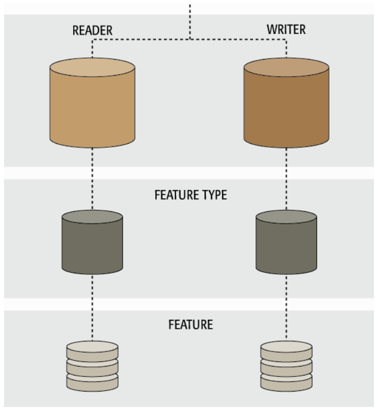


Parameter Documentation

- FME Readers and Writers Manual
  - Reader/Writer Directives
  - Feature Type Directives
  - Feature Representation

Example 6

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



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

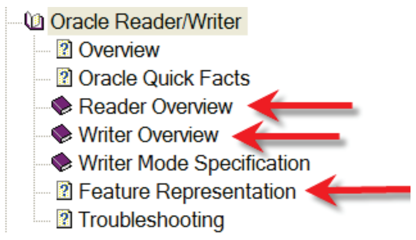
This manual lists – format by format – what parameters exist for the reader, writer, feature types and features.

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

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

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

| Translation Component | Reader and Writer Terminology |
|---|---|
| **Reader Parameters** | Reader Directives |
| **Writer Parameters** | Writer Directives |
| **Reader Feature Type Parameters** | Reader DEF Line Directive |
| **Writer Feature Type Parameters** | Writer DEF Line Directive |
| **Format Attributes** | Feature Representation Attributes |

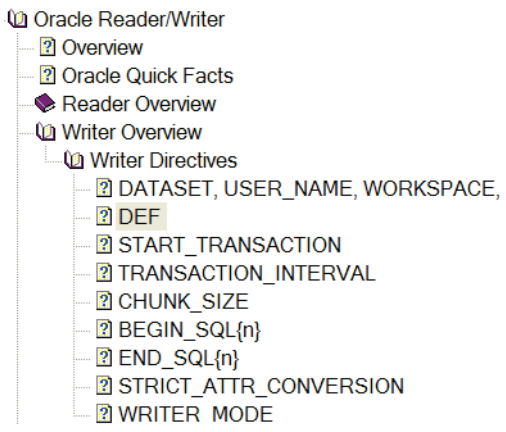


Oracle Reader/Writer
- Overview
- Oracle Quick Facts
- Reader Overview
- Writer Overview
- Writer Mode Specification
- Feature Representation
- Troubleshooting

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.

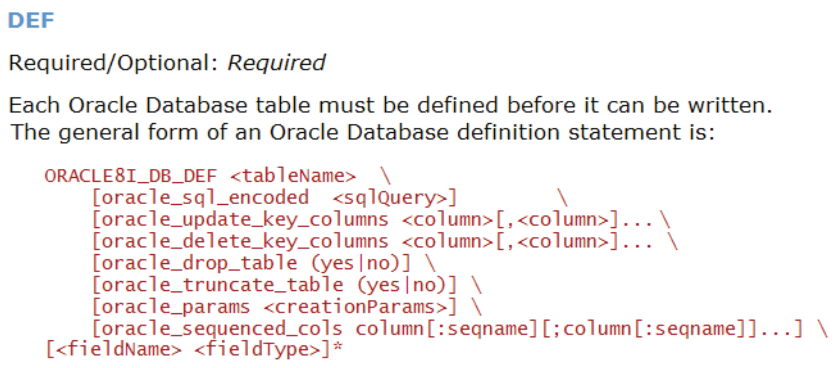

Oracle Reader/Writer
- Overview
- Oracle Quick Facts
- Reader Overview
- Writer Overview
  - Writer Directives
    - DATASET, USER_NAME, WORKSPACE,
    - DEF
    - START_TRANSACTION
    - TRANSACTION_INTERVAL
    - CHUNK_SIZE
    - BEGIN_SQL{n}
    - END_SQL{n}
    - STRICT_ATTR_CONVERSION
    - WRITER_MODE

***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.

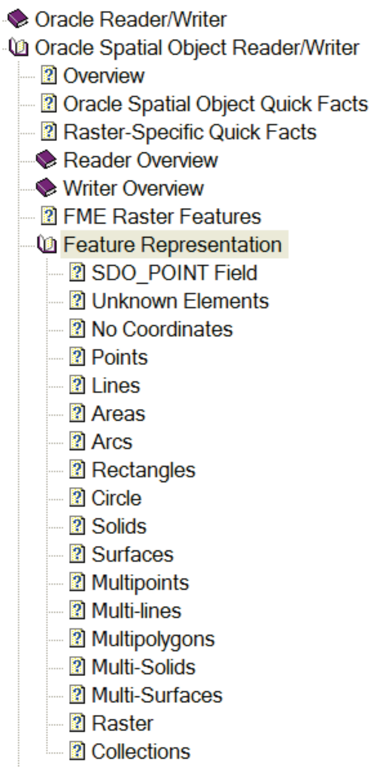
***Feature Type Directives***
Notice one of the writer directives is called DEF. This is where all of the writer feature type parameters are documented; for example Drop Table, Truncate Table, and Update Key.


**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:

```
ORACLE8I_DB_DEF <tableName>  \
    [oracle_sql_encoded  <sqlQuery>]        \
    [oracle_update_key_columns <column>[,<column>]...\
    [oracle_delete_key_columns <column>[,<column>]... \
    [oracle_drop_table (yes|no)] \
    [oracle_truncate_table (yes|no)] \
    [oracle_params <creationParams>] \
    [oracle_sequenced_cols column[:seqname][;column[:seqname]]...] \
[<fieldName> <fieldType>]*
```


Oracle Reader/Writer
Oracle Spatial Object Reader/Writer
- Overview
- Oracle Spatial Object Quick Facts
- Raster-Specific Quick Facts
- Reader Overview
- Writer Overview
- FME Raster Features
- Feature Representation
  - SDO_POINT Field
  - Unknown Elements
  - No Coordinates
  - Points
  - Lines
  - Areas
  - Arcs
  - Rectangles
  - Circle
  - Solids
  - Surfaces
  - Multipoints
  - Multi-lines
  - Multipolygons
  - Multi-Solids
  - Multi-Surfaces
  - Raster
  - Collections

***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 6: CAD to GIS | |
|---|---|
| **Scenario** | FME user; City of Interopolis, Planning Department |
| **Data** | Autodesk MapGuide Enterprise SDF |
| **Overall Goal** | Add CAD data from the engineering department to a public GIS dataset |
| **This Step** | Check Labels feature type |
| **Demonstrates** | Use of Readers and Writers Manual |

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

As a very quick example, check the Quick Facts section of the Readers and Writers Manual for the format Autodesk MapGuide Enterprise SDF Reader/Writer, to investigate why this exercise doesn't even bother to try writing text geometries.

## Module Review

FME Training Presentation

FME® Desktop 2011 version

www.safe.com

Translation Components

SAFE SOFTWARE

***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*

- The ***Formats*** supported by FME vary greatly in their licensing requirements, data types, geometries, and attributes.
- 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.

#### *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.