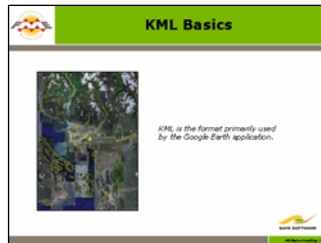

Advanced Module

FME and KML

KML Basics.....	3
What is KML?	3
KML Node Types	4
KML's Notion of Hierarchy	5
KML and FME Datasets + Feature Types.....	6
KML and Coordinate Systems	6
KML Format Attributes	6
KML Support within FME.....	7
Frequently Asked Questions:	7
Basic KML Translations.....	8
Reading KML	8
Writing KML	8
KML Styling	10
Styling Options in KML.....	10
KMLStyler Transformer.....	10
Feature Colour	10
Opacity.....	11
Icons	12
KML Descriptors	13
Feature Names	13
Feature Descriptions.....	14
Feature IDs	14
Special KML Feature Types	15
KML Hierarchy and Special Feature Types	15
KML Folders	15
KML StyleMaps.....	16
KML and Data Fanouts	19
KML and Dataset Fanouts	19
KML and Feature Type Fanouts	19
Raster Handling in KML	20
FME Rasters and KML.....	20
KML or KMZ?	20
Raster Handling Mode	21
Raster Output Format	22
Opacity.....	22
Network Links	24
What are Network Links?.....	24
How are Network Links Created?	24
Streaming KML with FME Server	30
Registering a Workspace for KML Streaming	30
Time Functions	33
Time Stamps.....	33
Time Spans.....	34
Session Review	39
What You Should Have Learned from this Module	39

KML Basics



Before getting into the FME side of the process, let's first take a brief look at the background of the KML format.

What is KML?

KML stands for Keyhole Markup Language. It is an XML-based format (or language if you prefer) intended to store data for use within the Google Earth™ and Google Maps™ applications.

The name “Keyhole” comes from the name of the original developers of the KML format and Google Earth product.

KMZ is an alternate form for a KML format dataset. A KMZ dataset is simply a KML dataset compressed by a ZIP type program and renamed with a new file extension. KMZ can incorporate both vector and raster data, but it is most frequently used as a means to store a set of raster images; the KMZ (zip) folder stores the raster files (as JPEG or GeoTIFF) plus a KML file that references them.

What do KML Datasets Look Like?

A KML dataset looks very similar to an XML or HTML dataset. In fact the analogy that Google uses is that Google Earth is to KML data what Internet Explorer is to an HTML document: simply a browser that lets you visualize the content of the dataset.

Google Maps is also a KML browser, but – at the time of writing – only supports a subset of KML. There are other KML viewers, but for this module we will stick to using Google Earth.

Like HTML, KML has a number of *tags* that affect how specific features are displayed. Since KML stores data of a spatial nature – whereas HTML tends to be non-spatial information – the type of tags are those that relate to spatial data symbology; for example, line styles, point symbols and area fill colours.

After cleaning up the data a little (by removing the style tags that define feature symbology), a sample KML dataset looks like this:

```
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
  <name>Safe Software HQ</name>
  <Placemark>
    <name>Safe Software HQ</name>
    <description>Safe's HQ, in Surrey, Canada.</description>
    <styleUrl>#msn_icon40</styleUrl>
    <Point>
      <coordinates>-122.8578661,49.1380373,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Above: This simple KML dataset defines a single point feature marking the home of FME.

KML Node Types

This is where the explanations can start to get very complicated; so we'll stick to basic generalities and not worry too much about precise definitions.

So, what is a node?

A node is a type of element within a KML dataset. KML has a number of node types...

Abstract Feature

An *abstract feature* node is a high level element that contains general information about the KML data, and defines several common elements that every other node type may have. A clearer, though not 100% accurate, description is to say that *abstract* loosely correlates with the term *metadata*.

Document

If you consider a KML file to be a dataset, then a *document* is like a sub-dataset; it's a one-to-many relationship, so each KML file can contain a number of distinct documents.

Folder

A *folder* is a sub-element of a document; the closest term it correlates to is layer. In FME-speak that would be a Feature Type, and in fact each destination feature type within a KML writer becomes (by default) a folder in the KML output.

Placemark

The easiest description of a *placemark* is to call it a vector feature, for example a point, line or polygon. Geometry as FME knows it comes under a *placemark* node.

Supported geometries are:

- Points
- Lines
- Polygons
- Donuts
- Text
- Aggregates

Unsupported geometries are:

- Circles
- Ellipses
- Arcs
- Non-Geometry Features

GroundOverlay

A *groundoverlay* element is simply a raster feature, or rather a pointer to a file containing a raster feature.

ScreenOverlay

A *screenoverlay* element is a raster-type feature used as a Google Earth screen decoration. It's not one that we'll worry about too much in this module.

PhotoOverlay

The *photooverlay* element is a raster feature that can be used to geographically locate a photograph on the Earth.

Network Link

A *network link* is an element that refers to other data sources such as a KML file or web URL. Interestingly it is an element in its own right, and not part of a vector feature.

Style

A *style* element is a tag for individual placemark features that defines symbology.

StyleMap

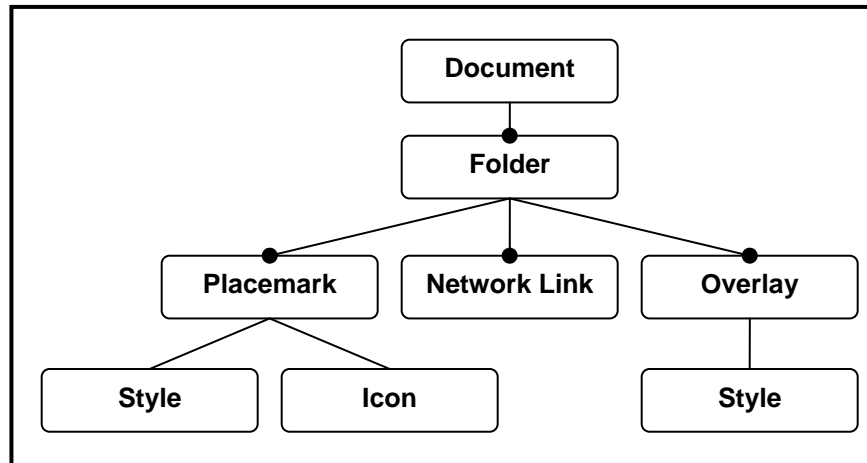
A *stylemap* is a set of default style tags for all placemark features.

Icon

An *icon* isn't really a node, but a symbol that represents a *placemark* point feature.

KML's Notion of Hierarchy

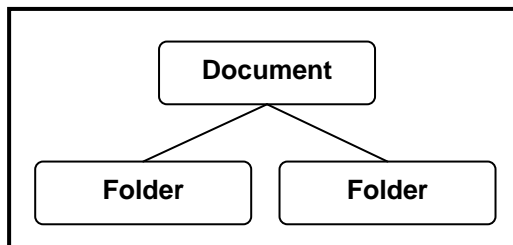
KML, like XML, has a hierarchical structure, which means that nodes appear at different levels, and that a higher level node acts as a container for a lower level node.



Left: The hierarchy of a KML dataset.

A connection ending in a black circle indicates a possible one-to-many relationship.

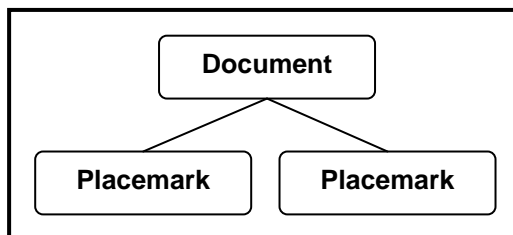
With such relationships each container may hold more than one item, or none at all, for example:



```

<Document>
  <Folder>
  ...
</Folder>
  <Folder>
  ...
</Folder>
</Document>
  
```

Above Left: A multi-folder dataset and **(right)** its equivalent KML definition.



```

<Document>
  <Placemark>
  ...
</ Placemark >
  <Placemark>
  ...
</ Placemark >
</Document>
  
```

Above: Left: A document containing multiple placemarks, but no folders and **(right)** its KML definition

KML and FME Datasets + Feature Types

As with any FME supported format it is important to be aware of how the format's structure relates to FME, and how FME defines that structure as a schema.

For KML, either a single KML file or set of KML files is counted as an FME *Dataset*, and each folder is treated as an FME *Feature Type*. A placemark relates to a single FME *Feature*.

FME	KML
Dataset	File or Set of Files
Feature Type	Folder
Feature	Placemark
Raster	

For example, if the dataset name is planning and the feature type name is ROADS, then the output would be a file called planning.kml containing...

```
<Folder id="kml_ft_ROADS">
```

KML and Coordinate Systems

KML stores coordinates as latitude and longitude values based on the WGS84 datum. These are the only coordinate systems that KML supports.

The FME equivalent coordinate systems are **LL84** and **EPSG:4326**

Data that is sent to the KML writer will not cause a problem provided that FME can ascertain the coordinate system of the source data. In that situation FME will automatically convert the data to LL84.

When FME is unable to ascertain the source coordinate system the translation will be terminated.

```

INFORM|Writing schema elements...
FATAL |Feature does not have a coordinate system specified
STATS |Storing feature(s) to FME feature store file 'mapping_log.fts'
FATAL |+++++
FATAL |Feature Type: 'pools'
FATAL |Attribute(string): '_wb_out_feat_type_' has value 'pools'
FATAL |Attribute(string): 'fme_feature_type' has value 'pools'
FATAL |Attribute(string): 'fme_geometry' has value 'fme_point'
FATAL |Attribute(string): 'fme_type' has value 'fme_point'
FATAL |Attribute(string): 'idrisi_type' has value 'idrisi_point'
FATAL |Attribute(string): 'kml_parent' has value 'kml_ft_pools'
FATAL |Attribute(string): 'kml_style_url' has value '#kml_style_ft_pools'
FATAL |Attribute(string): 'kml_type' has value 'kml_point'
FATAL |Geometry Type: Point (1)
FATAL |Number of Coordinates: 1 -- Coordinate Dimension: 2 -- Coordinate System: ''
FATAL |(3128413.25,10083640)
FATAL |=====
FATAL |Failed to finish writing to the default document.

```

Left: When data is sent to the KML writer from a source format which does not provide coordinate system support (and the user does not manually define it), then FME is unable to properly translate the data.

The process will be halted with a fatal error.

KML Format Attributes

FME uses format attributes to control individual KML features, for example *kml_name*.

The KML format attributes are very similar to the KML tags that they set, i.e. :

```
<name>Simple placemark</name>
```

These are described in detail in the Google Earth KML reference guide:
<http://code.google.com/apis/kml/documentation/>

KML Support within FME



KML is a rapidly-changing format. Safe Software aims to support all the latest developments, in as timely a manner as possible.

Frequently Asked Questions:

Some commonly asked questions about FME and its support for KML are:

Which FME Editions Support KML Format?

KML support in FME is provided by all FME editions. (Support for this format was added to the Base Edition in FME2008).

What KML Readers and Writers Does FME Include?

The FME **OpenGIS KML Encoding Standard** reader will read KML datasets that conform to the KML 2.0, 2.1, and 2.2 specifications. The writer will write datasets that conform to the KML 2.2 specification.

If necessary, it is possible to configure FME to support the older version of the reader/writer (KML21) as still exists as a plug in and can be used with older workspaces, but it is highly recommended that all new workspaces use the new version.

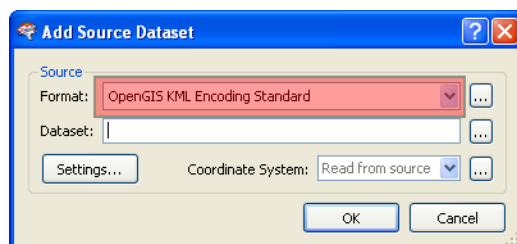
Gotcha!

Because KML goes under many names you may sometimes wonder where to find it in FME, particularly in the Formats Gallery and the Readers and Writers Manual.

At the time of writing the official FME name of the format is:

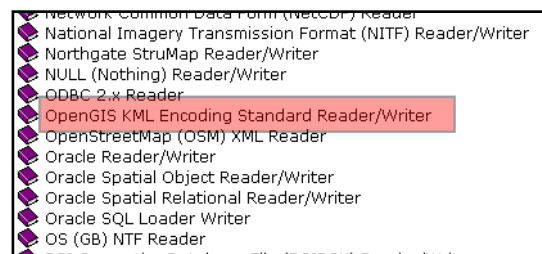
OpenGIS KML Encoding Standard

...so this is what you should look for when using the format. i.e. Because the format is owned by the OGC® we no longer associate it with Google Earth when naming it in our documentation.

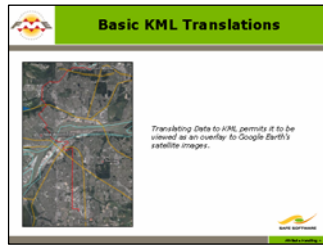


Left: KML as seen in the source dataset dialog...

Right: ...and as seen in the Readers and Writers Manual



Basic KML Translations



FME's support for KML allows users to translate other spatial datasets into a format that is used widely in today's society.

Reading KML

The OGCKML reader can read back all data written by FME. However, reading KML created by other applications may lead to problems; success can't always be guaranteed since that data may not be the same version of KML and may also not conform correctly to the KML specification.

Writing KML

Writing a basic KML dataset, with no concern about complex node types or feature styling, is as simple as choosing KML as the output format and running the translation. Where there is styling (symbolology) present on the source data, FME will attempt to preserve it when writing KML output.

KML Requirements

For the most part if there are any peculiarities about the KML format, FME takes care of them automatically.

- KML requires all features to be three-dimensional; if necessary FME will force compliance to this rule by setting a Z value of zero on all two-dimensional features.
- All nodes must have a unique ID. By default FME uses the format attribute `kml_id`, but if this is unset then FME will automatically create an id number in order to comply with this rule.
- As noted, KML requires all features to be held in the LL84 coordinate system. FME will automatically convert your data to LL84 provided that it knows the source coordinate system used. If it cannot deduce this information, and you do not provide it in the dataset parameters, then the translation will be stopped with an error.

KML or KMZ?

FME's KML writer provides the capability to write the output data as either a KML or a KMZ dataset. The type of dataset created depends upon the file extension you provide within the output dataset name; for example, call your output `myData.kml` to get an uncompressed dataset, or `myData.kmz` to have it created in a compressed form.

Writing Raster Data

When writing raster data to KML, FME automatically detects the type of data being used (vector or raster) and writes the appropriate KML dataset accordingly.

See the section to follow called "Raster Handling in KML" for more information on this topic.



Example 1 – Basic KML Writing

This simple exercise requires a data translation to KML format.

Using data from the hypothetical City of Interopolis, convert the City Grid features within the Planning Department's E00 dataset to KML.

Format	ESRI ArcInfo Export (E00)
Dataset	<u>C:\FMEData\Data\City Grid\city_grid.e00</u>

Do you need to set a coordinate system for the destination? If yes, then do so.

Open the output dataset in Google Earth to see what you have created.

Repeat the process, this time writing the data as KMZ. Compare the file size to the KML version.

Open the KMZ file using a zip file reader (renaming it to a .zip extension will help you to do this).

What are the contents? What is the name of the "root" document?

Open the dataset in Google Earth. Does it look any different to the KML version?

Save the workspace – we'll be using it in later examples.

KML Styling



Styling is another term for Symbology and refers to the look and feel of the visualized data

Styling Options in KML

KML format supports tags that define the style and symbology of the features within a dataset.

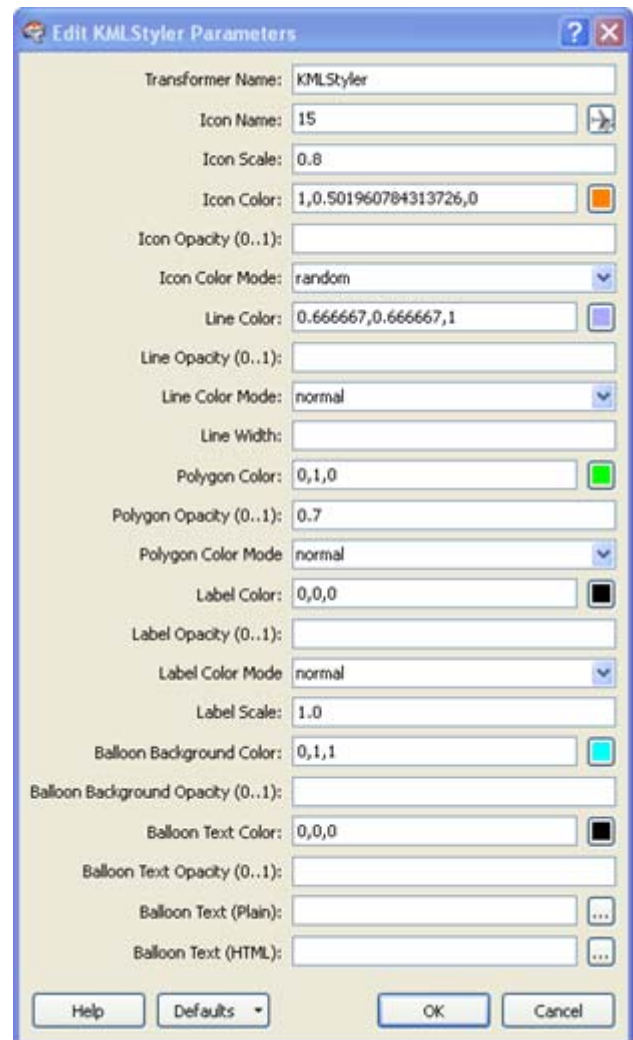
In FME these tags can be set using either a set of format attributes, or a specific KMLStyler transformer.

KMLStyler Transformer

Rather than have the user resort to format attributes to manipulate KML feature styling, FME Workbench has a *KMLStyler* transformer.

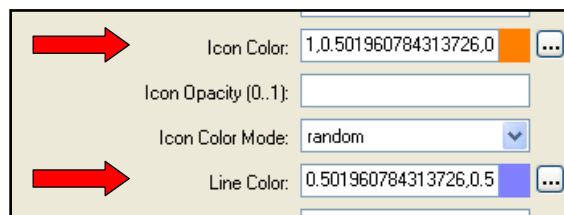
This transformer is one of the first dedicated to a single format.

Right: The parameters dialog box for the KMLStyler transformer.



Feature Colour

KML format does support colour, so that features can be set to display in different colours.



Left: Colour settings for Icon and Line features within the KMLStyler transformer.

Opacity

Features in KML can be given a value for opacity (or transparency), which is then reflected when the data is viewed within Google Earth.

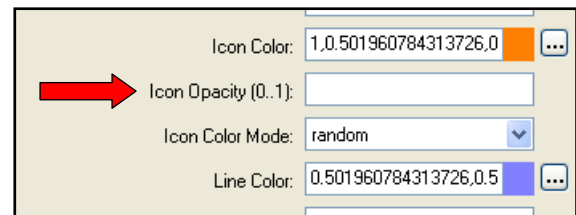
Opacity is measured as a value between 0 and 1, where 0 is fully transparent and 1 is fully opaque (solid). There are separate values for pen (line) opacity and fill opacity.



Above: From Left to Right – Green fill with 10% opacity (0.1), 50% (0.5) and 90% (0.9).

As with feature colour, opacity can be set within the *KMLStyler* transformer.

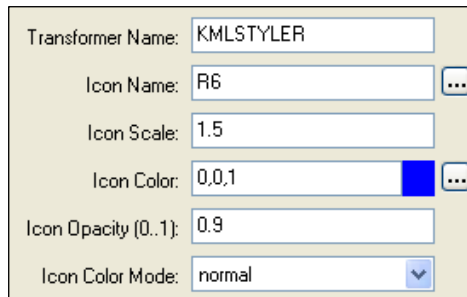
Right: *KMLStyler* opacity setting for icons.



Icons

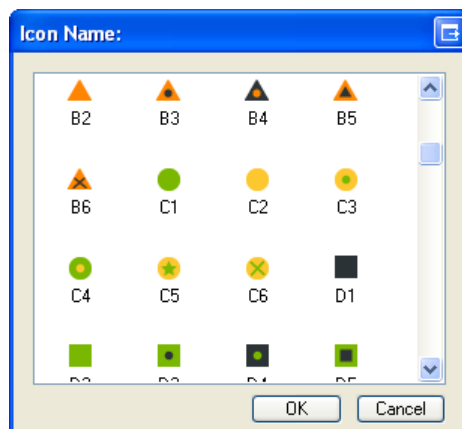
Point features in KML are assigned an icon as a map symbol. Only point features are selectable in Google earth. It is not possible to select areas or lines.

FME includes a set of built-in icons specifically designed for KML format translations and viewing. The parameters for these icons can be set using the option in the *KMLStyler* transformer. This includes icon type, colour, size and opacity.



Above: Creating Icons using the *KMLStyler* transformer

Above: The R6 icon seen in Google Earth.



The *KMLStyler* transformer has a gallery of available icon types from which a user can select.

There are similar attributes to define colour, scale and opacity – see the KML section of the FME Readers and Writers Manual for more information.

Left: The KML icon gallery in Workbench.

The equivalent Format Attribute to this icon selection is called *kml_icon*. *kml_icon* is a “magic” attribute in that it isn’t limited to this list, but can use any icon.

When FME writes a KML dataset it will copy icons defined by *kml_icon* into the dataset. If you want to merely reference an icon from elsewhere, and not include it as part of the data, then you can define it using the format attribute *kml_icon_href*



Example 2 – Symbology

Continue with the workspace from example 1. Use the *KMLStyler* transformer to set polygon colour (to red) and opacity (to 40%). Also use the *KMLStyler*, to apply an icon and symbology to the City Grid point features.

Use the following settings:

<i>FME Icon</i>	E1	<i>Icon Colour</i>	1,0,0 (Red)
<i>Icon Scale</i>	0.7		

Use Best Practice by using a method which requires only a single *KMLStyler* transformer, rather than placing one transformer for each stream of data (a *FeatureTypeFilter* will help).

Why is there more than one set of icons? The *KMLStyler* transformer will automatically create an icon and pair it to the polygon and line if an icon style is selected for polygon or line features.

KML Descriptors

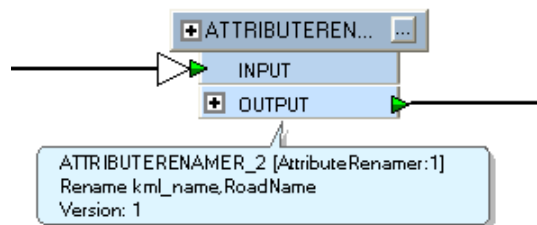


KML Descriptors are tags that provide identifiers to help a user recognize features when browsing a KML dataset.

Feature Names

Naming a KML feature allows it to be more easily identified within a KML browser such as Google Earth. Features can be named within FME by setting the format attribute *kml_name*. If *kml_name* is not set then Google Earth will use the *kml_id*.

In most cases a user would take an existing attribute name and apply it using an *AttributeCopier* or *AttributeRenamer* transformer.

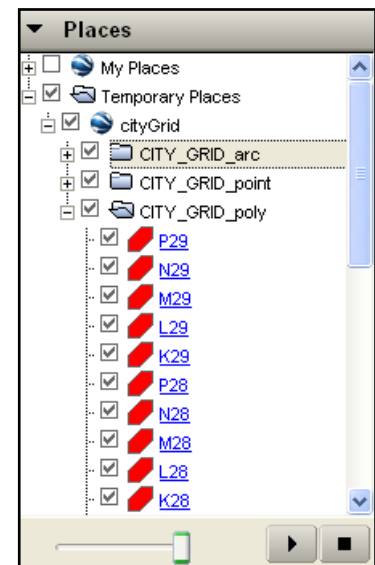


Left: Setting a name for features in a KML dataset using an AttributeRenamer

In Google Earth named features can be identified within the "My Places" panel.



Above: Points automatically display their name in Google Earth.

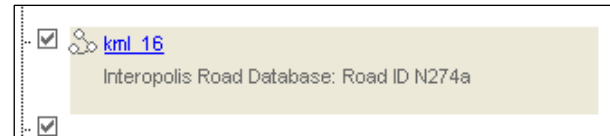
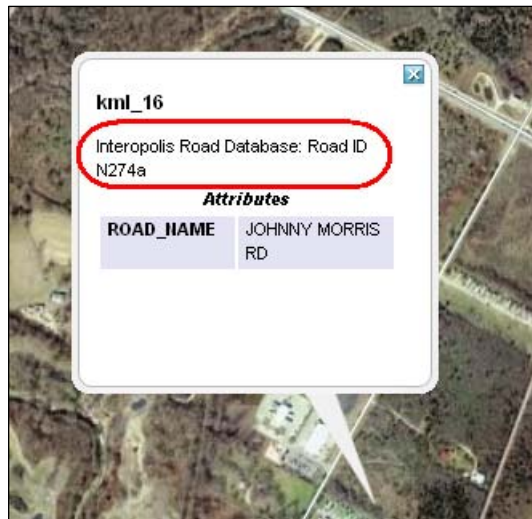


Right: Polygons show their names under the "My Places" legend.

Feature Descriptions

Feature descriptions are stored as part of a KML node and can therefore be displayed by a KML browser such as Google Earth.

Like KML feature names, feature descriptions are created by the FME user supplying a value for a format attribute, in this case *kml_description*



Above: A feature description in the Google Earth “My Places” legend.

Left: A feature description in the Google Earth data view.

Feature IDs

As noted, all KML nodes must have a unique ID. The ID must be unique within the entire dataset (not just a single folder). FME will automatically create an id number in order to comply with this rule, but the user can apply an override by assigning a value to the format attribute *kml_id*.

Where an id number pre-exists then *kml_id* can be set using an *AttributeCopier*, but where it does not then the user can create an id number using the *Counter* transformer.



Example 3 – KML Feature Naming and Feature IDs

Continue to develop the workspace from examples 1 and 2.

1. Add a *Counter* transformer to create a unique ID number for each feature using the *kml_id*. Run the workspace and inspect the results. How have the icon names changed?
2. Use the attribute *TILE_NAME* to give names to the output KML features. To do this copy the value into the *kml_name* attribute. Run the workspace and inspect the results. How have the icon names changed?
3. Also set a description (*kml_description*) for each feature using the following:
Tile: <TILE_NAME> from City of Interopolis dataset: city_grid

A *Concatenator* transformer will help to create this description. The workspace will also be more flexible if you can take the dataset name (the “city_grid” part of the description) from a source format attribute, rather than as a hard-coded value.

By now you may have realized that the only destination feature type required is *CITY_GRID_poly*. You should disable the other two source Feature Types.

Congratulations – this workspace is now complete.

Special KML Feature Types



Some special components of KML require special FME Feature Types to create and handle them.

KML Hierarchy and Special Feature Types

KML – like XML – is a hierarchical type of dataset; there may be many KML documents in a dataset, and many folders in a document.

However, FME is not by nature a hierarchical based system. Therefore we use special Feature Types to describe different levels (tags) within a KML dataset.

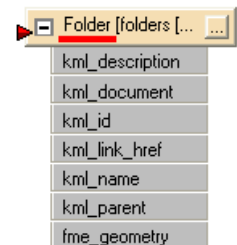
To trigger a special Feature Type usually only requires a single feature; usually a single feature is either extracted from the main flow of data or created with a Creator transformer.

Relationships are maintained between different levels of hierarchy with format attributes.

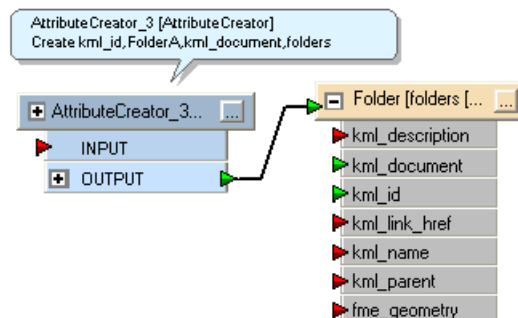
KML Folders

You may remember from the earlier diagram that KML Folders are a subset of a KML document. It is sometimes useful to be able to organize data inside a KML dataset by creating multiple folders and assigning features to each of them.

Because a folder is not part of a standard Feature Type definition, the only way to create such an entity is to set up a new Feature Type.



Right: Here a user has created a Feature Type to define one or more folders. The Feature Type must be named “Folder” for FME to recognize that this is the user’s intended purpose.

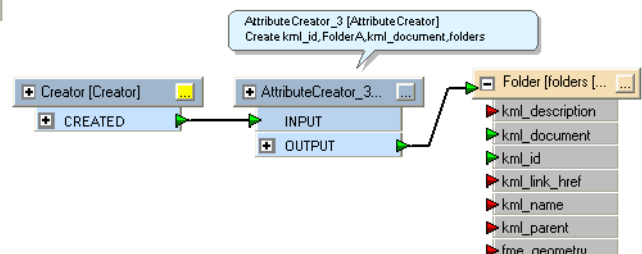


Left: Now the user creates a set of attributes to define the folder. These are:

kml_id: A unique ID (in this case “FolderA”

kml_document: The name of the KML document

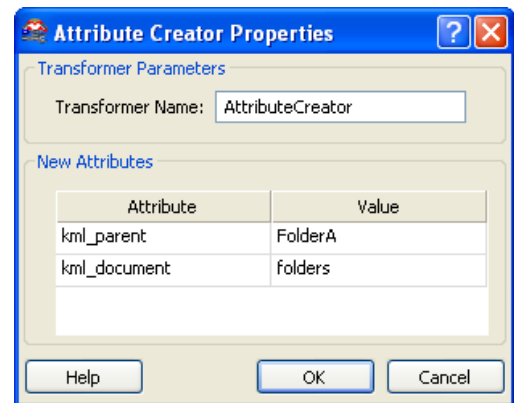
Right: Finally a Creator transformer triggers creation of the folder by outputting a single null geometry feature.



Linking Features to Folders

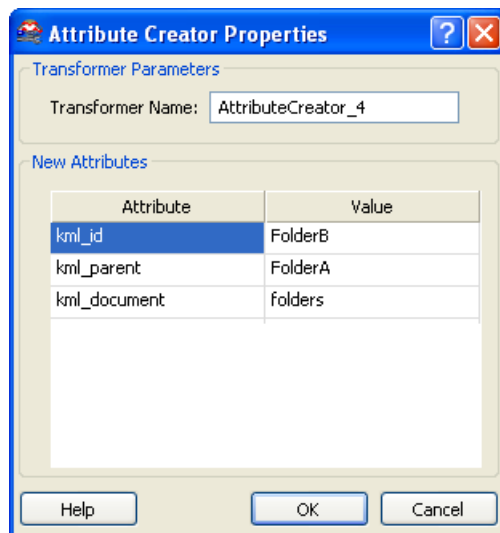
The final part of creating a KML Folder is to assign features to it. This is achieved using the format attribute *kml_parent*

Right: By dropping an AttributeCreator transformer into the main flow of features, the user has assigned all of the features into FolderA.



Multiple Folders

Multiple folders can be defined by simply submitting multiple features to the Folder feature type.



Nested Folders

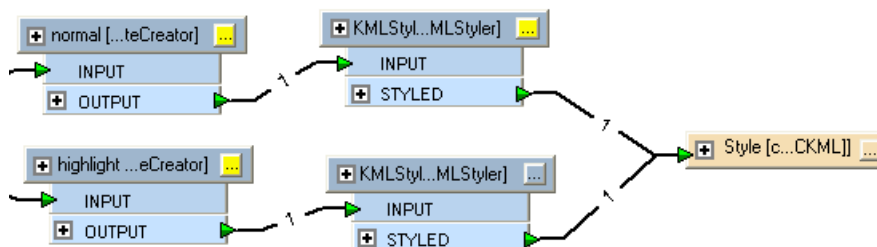
Nested folders can be created by simply defining one folder as the child of another.

Left: Here the user creates a new folder, FolderB, and by specifying a *kml_parent* value defines it as a child of FolderA

KML StyleMaps

KML uses a StyleMap element to map between two different Placemark display styles. It is typically used to specify the styling for the normal and highlighted states of a Placemark. This means that it is possible to change the style of an icon when the mouse moves over (highlights) a particular feature.

Like Folders, StyleMaps are defined by using a special Feature Type definition.



Left: A Style feature type shown connected to two KMLStylers; one defines the 'normal' style and the other defines the 'highlight' style.

Linking Features to StyleMaps

To assign a style to a particular feature, several attributes must also be added:

kml_target_style_highlight – This attribute points to the KML ID of the ‘highlight’ style

kml_target_style_normal – This attribute points to the KML ID of the ‘normal’ style

kml_create_info_point – This attribute will, if set to ‘yes’, force the creation of a point, which would be needed to display an icon for non-point features (such as lines or polygons). Create information point is also available under the feature type parameters.



Example 4 – Special Feature Types

Continue using the workspace from example 3.

Create a Folder in the output KML by creating a Folder feature type. The Folder name should be: CityGrid

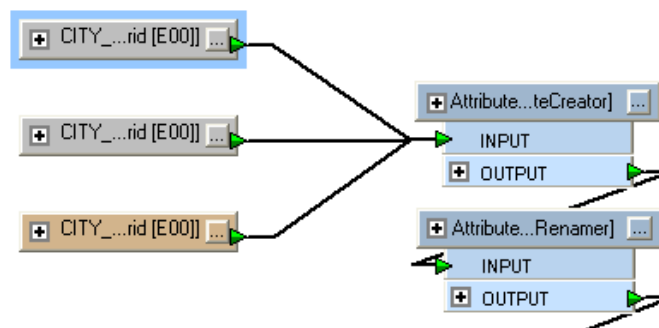
Assign all of the features being written to KML to the CityGrid folder.

Next create a highlight style for the polygon features in the dataset. You should ensure the arc and point feature types present in the workspace are disabled or deleted.

The KMLStyler that is currently in your workspace is specifying one style for every feature in the destination dataset. Because you want to define two different styles for the output polygons, you will need to delete the *KMLStyler*.

Replace the *KMLStyler* with an *AttributeCreator* that will create the following attributes and values:

kml_target_style_highlight	highlight_style
kml_target_style_normal	normal_style
kml_create_info_point	yes



Left: Shows the placement of the new *AttributeCreator*

Create the Normal and Highlight Styles

Create a new feature type in the destination dataset. Do this using the menu *Destination Data > Add Feature Type Definition* (or right click and select *Insert Feature Type Definition*). Change the feature type name to *Style*.

Connect a *Sampler* to the polygon input feature type. Use the following parameters:

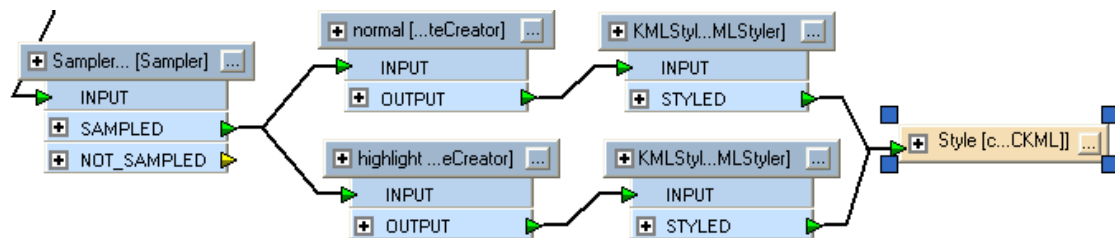
<i>Sampling Amount</i>	1
<i>Sampling Type</i>	First N Features

To create the 'normal' style, connect an *AttributeCreator* to the *Sampler*. Create an attribute called *kml_id* and give it the value, *normal_style*. Next, connect a *KMLStyler* and give it the same values you did in example 2:

<i>Icon Name</i>	E1
<i>Icon Scale</i>	0.7
<i>Icon Color</i>	1,0,0
<i>Polygon Color</i>	1,0,0
<i>Polygon Opacity</i>	0.4

To create the 'highlight' style, repeat the same steps as above, but, in the *AttributeCreator*, set the value of *kml_id* to *highlight_style* and use the following settings in the *KMLStyler*.

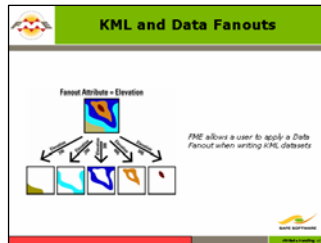
<i>Icon Name</i>	E5
<i>Icon Scale</i>	1.2
<i>Polygon Color</i>	1,0,0
<i>Polygon Opacity</i>	0



Above: The result of the above steps should look something like this.

Run the translation and open the output dataset in Google Earth. Notice how all features are stored under the CityGrid folder, and how each polygon's display style changes when you move the mouse over it.

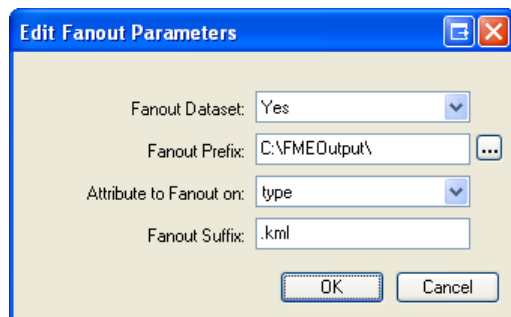
KML and Data Fanouts



A fanout splits up data into multiple destinations 'on the fly'. FME allows a user to apply a fanout when writing KML data.

KML and Dataset Fanouts

A dataset fanout diverts data to the same feature type within different datasets. For example, in a dataset consisting of cities within Canada, a fanout on the attribute *province* would create multiple datasets, each containing cities for a specific province.

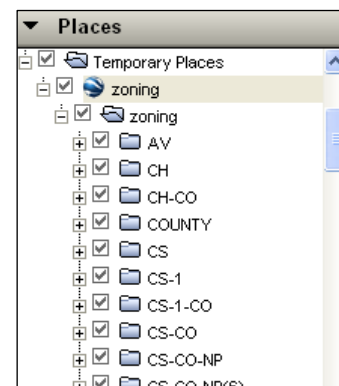
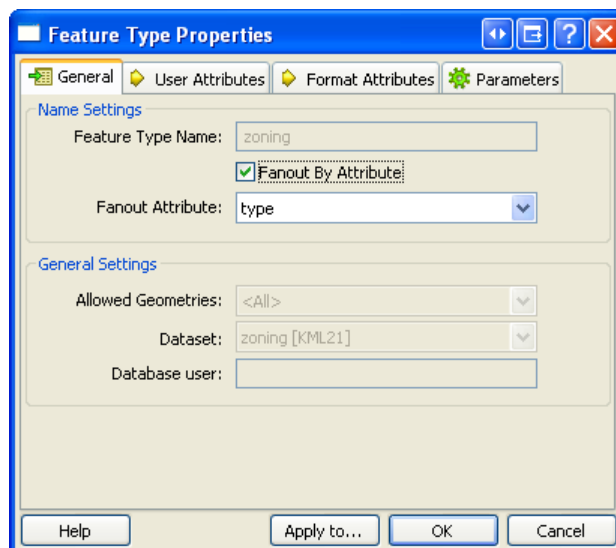


CH-CO.kml	11 KB	Google Earth KML	05/09/2007 11:42 AM
CH.kml	7 KB	Google Earth KML	05/09/2007 11:42 AM
CS-1-CO.kml	2 KB	Google Earth KML	05/09/2007 11:42 AM
CS-1.kml	28 KB	Google Earth KML	05/09/2007 11:42 AM
CS-CO-NP(6).kml	18 KB	Google Earth KML	05/09/2007 11:42 AM
CS-CO-NP(7).kml	2 KB	Google Earth KML	05/09/2007 11:42 AM
CS-CO-NP.kml	11 KB	Google Earth KML	05/09/2007 11:42 AM
CS-CO.kml	22 KB	Google Earth KML	05/09/2007 11:42 AM
CS-MU-CO-NP.kml	13 KB	Google Earth KML	05/09/2007 11:42 AM
CS-MU-CO.kml	2 KB	Google Earth KML	05/09/2007 11:42 AM
CS-MU-NP(1).kml	15 KB	Google Earth KML	05/09/2007 11:42 AM

Above: This zoning data has a Dataset Fanout applied to the type of zone. The result is a series of KML datasets, one for each type of planning zone.

KML and Feature Type Fanouts

A feature type fanout diverts data to different feature types (layers) within a single dataset. In the Canadian cities example, a feature type fanout on the *province* attribute would produce a single dataset containing multiple layers (sub-folders in KML), each containing cities for a specific province.



Left/Above: The same zoning data now has a Feature Type Fanout applied. The result is a single KML dataset (zoning.kml) that has a KML folder for each zone type.

Raster Handling in KML



KML format supports raster features and FME is equally able to supply data using that structure.

FME Rasters and KML

FME can write raster features directly to a KML (or KMZ) dataset. It is even able to write a combination of vector and raster features to the same output file!

The OGC KML writer works with raster features sent to the writer (as opposed to just copying the original source) and carries out proper raster reprojection, rather than just doing a vector reproject on the raster's bounding box.

KML or KMZ?

In most instances you will want to write your raster data to a KMZ output file. You do this by simply specifying KMZ as the output extension to use.

By default, writing data to a file with a KMZ extension will cause a single KMZ file to be created containing both a header document (*doc.kml*) and a sub-folder containing the raster image(s).

When the file extension is set to KML, then the output is somewhat different.

The KML file itself is the header document. It contains pointers to the raster image(s) which are stored in a completely separate folder called *images*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by Feature Manipulation Engine 2009 (Build 5658) -->
<kml xmlns="http://earth.google.com/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
<name>doc</name>
<visibility>1</visibility>
<Folder id="kml_ft_TIFF">
<name>TIFF</name>
<GroundOverlay id="kml_1">
<name>kml_1</name>
<Icon>
<href>images/interopolis.tif</href>
</Icon>
...etc
```

Above: When the output file is a KML extension then the document contains a reference to separately stored image(s).

Raster Handling Mode

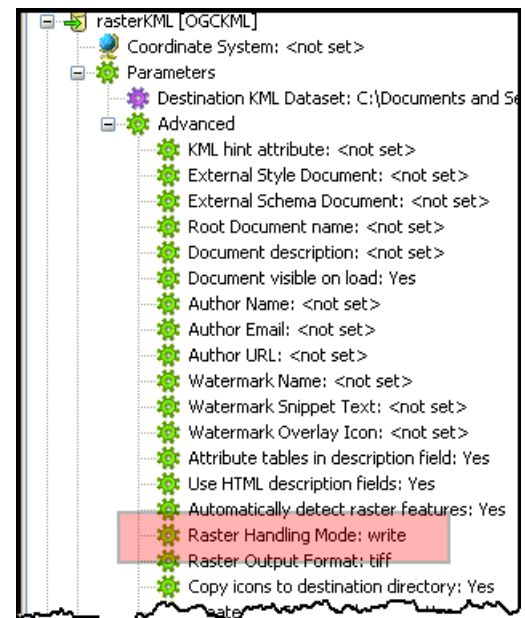
The Raster Handling Mode is a dataset setting, and so is found in the navigation pane in Workbench (*right*).

The three available modes are:

- Write
- Copy
- Relative

Write Mode

Write Mode means that the KML writer uses whatever raster features that are delivered to it. This means that it can handle any raster – regardless of its origin – and write it as it has been processed by any transformers within the workspace.

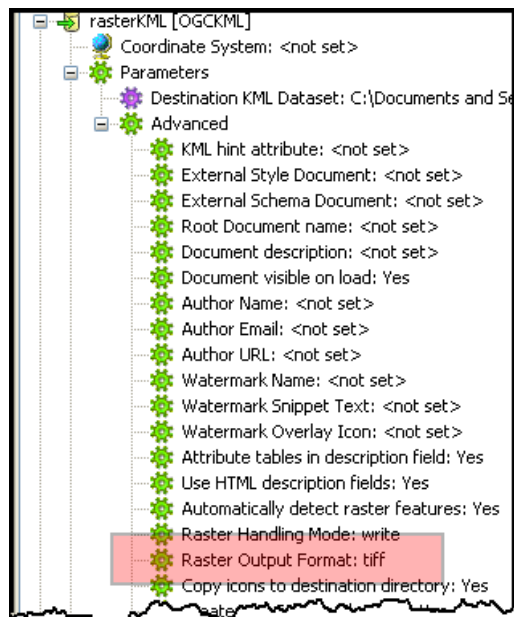


Copy Mode

Copy Mode means that the KML writer examines an incoming raster feature, locates its original source dataset, and makes a copy of that file into the output KML folder. The resulting GroundOverlay node will reference the file copy. Obviously this will only work for raster formats that KML supports (PNG, JPEG or TIFF), and could not be used for non-file raster formats such as an Oracle GeoRaster. Also it does not take into account any processing that has taken place within workspace transformers.

Relative Mode

Relative Mode is the same thing as Copy Mode, except that instead of making a copy of the source raster file, it writes a GroundOverlay feature that references the original. This avoids creating extra copies of the source, but does make data management slightly more difficult when you intend to pass on the KML dataset to another user.

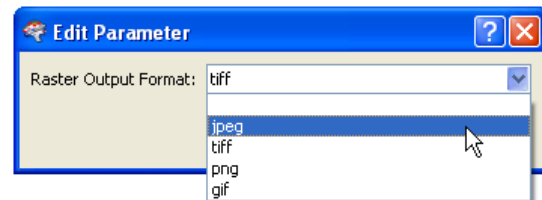


Raster Output Format

The Raster Output Format setting in the navigation pane (**left**) allows a user to change the type of raster being written to the KML dataset.

The four options are JPEG, TIFF, PNG, or GIF

In general terms, a TIFF output provides better quality, but at the cost of larger file sizes.



Obviously the Raster Output Format setting will only have an effect when the Raster Handling Mode is set to "write".

Opacity

Opacity in a raster dataset is set using the format attribute *kml_overlay_color*

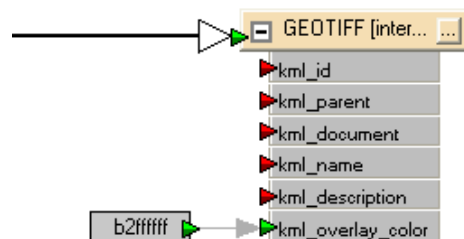
However, the value for this attribute is not on a simple 0 to 1 scale, but an ARGB value. ARGB is like a simple RGB value, but the extra A (for Alpha) refers to the opacity value.

Each value of A-R-G-B is on a scale of 0 to 255, but as a way to speed up graphic processing calculations within a KML browser, (and not intentionally to complicate things for a user) the values are each supplied in a two-digit hexadecimal format.

In simpler terms the value for *kml_overlay_colour* should be an 8 digit number between 00FFFFFF and FFFFFFFF. The last six digits (FFFFFF) are a mask that is a bitwise AND (if that sort of information interests you), but it is the first two digits – representing the opacity – that are the important part, giving:

00FFFFFF	No transparency	00 is hex for 0
40FFFFFF	25% transparency	40 is hex for 64
7FFFFFFF	50% transparency	7F is hex for 127
C0FFFFFF	75% transparency	C0 is hex for 192
FFFFFFFF	Fully transparent	FF is hex for 255

Right: Here a user is using a constant to set raster opacity. In general it's unlikely that you would want to set different opacities for different raster features, so that a constant is a good method for setting this format attribute.



Example 5 – Writing a Raster Dataset to KML

This example involves converting a raster DEM to a KML dataset.



Setup

Set up a translation to convert the following raster dataset to KMZ format.

Format	Canadian Digital Elevation Data (CDED)
Dataset	<u>C:\FMEData\Data\ElevationModel\RasterDEM-250K.dem</u>

Transformers

The source dataset contains raster information held as 32-bit integers (*INT32*). This type of data is incompatible with a KML raster dataset, so you will need to use a *RasterInterpretationCoercer* transformer to restructure the data to something more suitable. Reinterpret the data as a *Gray8* dataset.

The source data is also stored in a coordinate system incompatible with KML. Because the coordinate system information is held as part of the dataset, and because FME recognizes this information, the KML writer will automatically apply the coordinate reprojection.

Symbology

Use *kml_overlay_color* to set the opacity of the output to 70% (B2 is the hex code you need). You should set this as a constant, rather than trying to copy an existing attribute.



Above: The output from your workspace – as seen in Google Earth – should look something like this. See how the underlying Google Earth background is partly visible under the GroundOverlay.

Questions

Open the output KMZ file using a zip file reader. What are the contents? What format are raster datasets stored in? Check the FME help system to find out how to change the raster format used.

Write the raster data as a KML file (instead of KMZ). What is the difference in the output?

Network Links



Network Links are another special feature type – so special they deserve a section all to themselves!

What are Network Links?

A Network Link is a connection from one KML dataset to another. The linked-to dataset can be held either locally or remotely.

Opening a dataset containing a network link should cause the KML browser to also retrieve the dataset that is being linked to.

Network Links are useful for reading datasets that are maintained and stored by someone else, without having to download and open those datasets.

Network Links are also a useful way to handle large datasets, by dividing them up into regions and fetching those regions on demand via a link.

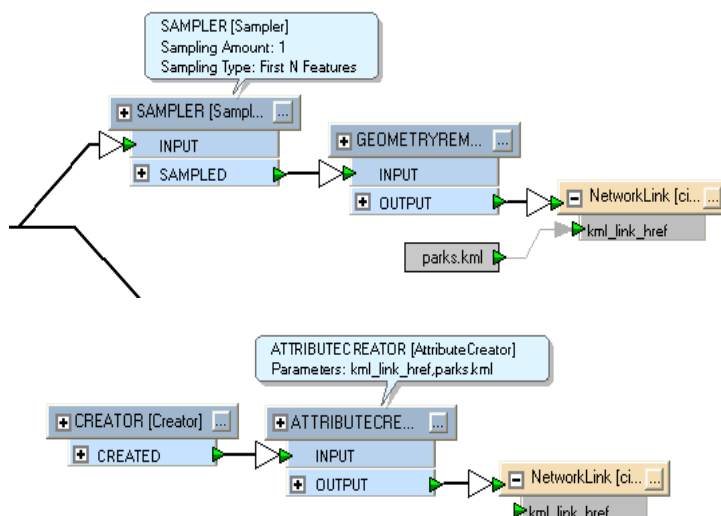
How are Network Links Created?

FME sometimes creates Network Links automatically in response to some situation such as multiple KML Documents. However, a user may also create a Network Link manually.

A Network Link works at a dataset (or document) level, rather than on individual features. In other words, when user defined, it needs a separate feature type within an FME workspace.

The process to create a Network Link manually is:

- Create a feature type called NetworkLink
- Create a single non-geometry feature for each required link
- Define the link using the format attribute *kml_link_href* – with the file name.
- Send the link feature(s) to the NetworkLink feature type



Left: Here a user creates a Network Link by skimming off features from the main workflow, reducing it to a single feature (using a Sampler), removing the geometry and writing it to a NetworkLink feature type. *kml_link_href* is set with a constant – the file name of the link.

Left: This user makes a Network Link by creating a null-geometry feature with the appropriate *kml_link_href* value.

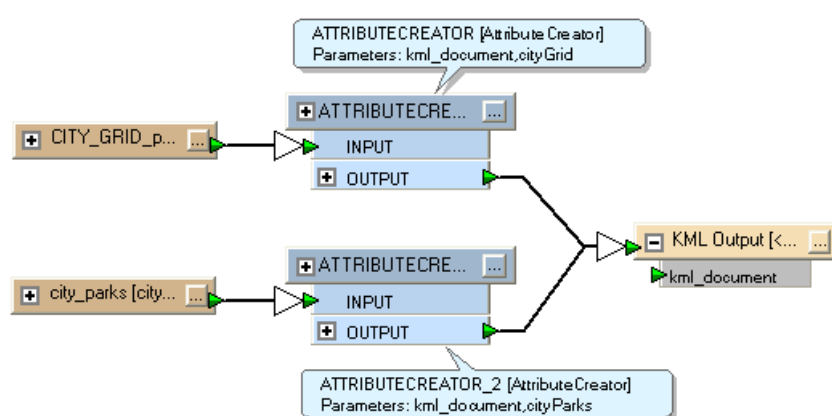
Network Links and Multiple Documents

FME is also able to create network links automatically when a KML dataset is written containing multiple documents.

Remember that a KML *Document* is like a sub-dataset. FME can assign different features to different Documents by applying a value for the format attribute *kml_document*

When features are written to multiple documents, the header document is named with the user-defined dataset name, and the underlying documents are named according to the values for *kml_document*

By default FME creates network links from doc.kml to the other kml Documents.



Left: This user is writing data to a KML dataset called *planningData*.

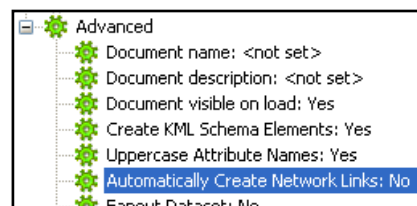
The features are assigned different document names: *cityGrid* and *cityParks*

Below: The output from this process is 3 files. *planningData.kml* has network links to the other two files



This automatic creation of network links is controlled by the advanced KML writer parameter “*Automatically Create Network Links*”. The default behaviour is to create these links. In some cases a user may wish to turn off this functionality by setting the value of this parameter to No.

Right: Here the user has chosen not to create automatic network links. The output will be the same as above, but the file *planningData.kml* will not contain links to the other two files.





Example 6 – Working with Network Links

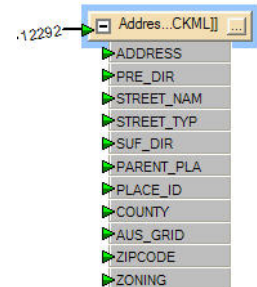
The City of Interpolis Engineering department would like to publish their address database in KML for the use of city residents. However, they are worried that the data will be too large to be handled in KML. They plan to use Network Links to a set of tiled data to resolve this issue.

In this example we will take the address database, tile it, create KML documents for each tile and apply Network Links to join the documents together.

Setup

Set up a translation to convert the following GeoMedia (Access Warehouse) dataset to KML. The GeoMedia table name can be chosen by clicking the Settings button in the New Workspace dialog.

Format GeoMedia Access Warehouse
Dataset C:\FMEData\Data\Addresses\roadAllowancesAndAddressPoints.mdb
Table ADDRESS_POINTS



Remove some of the redundant attributes from the KML
 ADDRESS_POINTS feature type.

Add Second Source Dataset

Now we want to tile the data. A good way to do this is to clip the data against a city grid dataset.

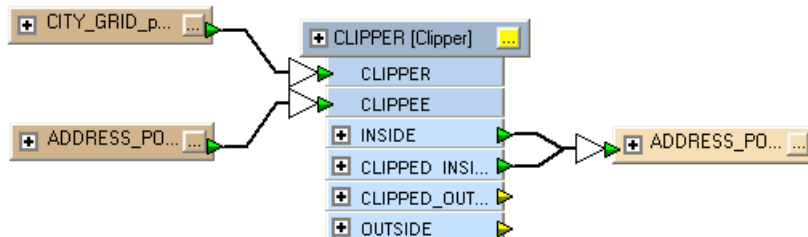
Add the city grid dataset as a new source using Source Data > Add Dataset.

Format ESRI ArcInfo Export (E00)
Dataset C:\FMEData\Data\CityGrid\city_grid.e00
Feature Type CITY_GRID_poly

Clip Data

To clip the data add a *Clipper* transformer. Connect the Address database to the Clippee input port and the City Grid dataset to the Clipper.

Connect the INSIDE and CLIPPED_INSIDE output ports to the destination dataset.

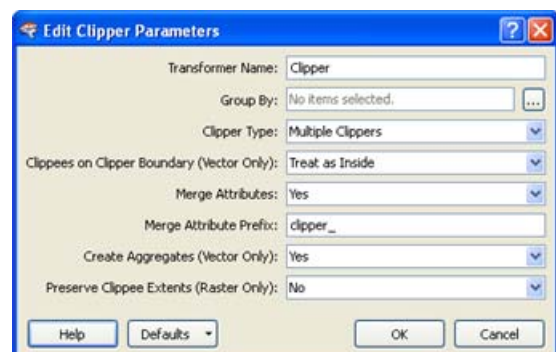


Left: At this point your workspace should look something like this.

Right: Set up the Clipper using the settings shown on the right. The important ones are:

Clipper Type
Merge Attributes

Multiple Clippers
Yes



Set Document Names

Each set of tiled addresses can be stored in a Document bearing the name of the related tile. This is simply done by using an *AttributeCopier* transformer to copy *clipper_TILE_NAME* to *kml_document*

At the same time set the *kml_name* to the the ADDRESS attribute.

NB: If you don't have an attribute called *clipper_TILE_NAME* then you probably did not set the *Clipper > Merge Attributes* setting to Yes.

Style the Features

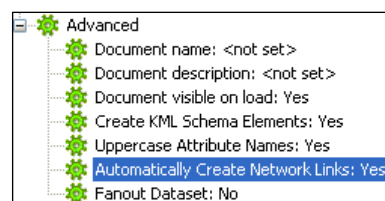
Use a *KMLStyler* transformer to give the address features an appropriate style.

Hint: Icon C2, Scale 0.5 is a good choice.

In the workbench navigation window, under the KML writer, ensure the advanced parameter *Automatically Create Network Links* is set to Yes (**right**) and run the workspace.

On running the workspace you will be prompted for an output location. For clarity's sake, use an empty folder, such as:

C:\FMEData\Output\TrainingAdvanced\addressDatabase\addresses.kml



Left: The output folder will look like this.

Each tile has been given a separate KML Document in order to avoid creating a single, over-large dataset.

Right: The output file *addresses.kml* contains a number of network links to the individual Documents.

If you do not see these links, then you probably omitted to set the *Automatically Create Network Links* parameter.

```
<NetworkLink id="linkto_K24.kml">
<name>K24.kml</name>
<Link>
<href>K24.kml</href>
</Link>
</NetworkLink>
<NetworkLink id="linkto_K25.kml">
<name>K25.kml</name>
<Link>
<href>K25.kml</href>
</Link>
</NetworkLink>
<NetworkLink id="linkto_K28.kml">
<name>K28.kml</name>
<Link>
<href>K28.kml</href>
</Link>
</NetworkLink>
```

After opening the output in Google Earth and showing it to your manager he has a new request. He would like to see zipcode boundaries displayed alongside this data.

You realize that this too can be done using a Network Link.

Translate Zipcodes to KML

Start a new instance of FME Workbench – or save and close the current workspace – and perform a quick translation of the zipcode boundary dataset to KML.

Format	MapInfo MIF/MID
Dataset	<u>C:\FMEData\Data\Addresses\zipcode_boundaries.mif</u>
Output	<u>C:\FMEData\Output\TrainingAdvanced\addressDatabase\zipcodes.kml</u>

Create Network Link Feature Type

Return to the original Address Database translation workspace.

Create a new feature type in the destination dataset. The simplest way is probably to use the menubar command *Destination Data > Add Feature Type Definition...*

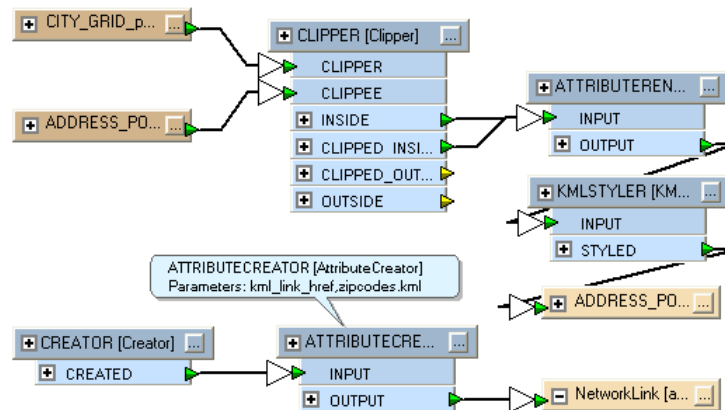
The new feature type should be named NetworkLink

Create Network Link

Manually create a Network Link using either of the two methods previously discussed: creating a non-geometry feature specifically for the purpose, or taking a feature from the main workflow.

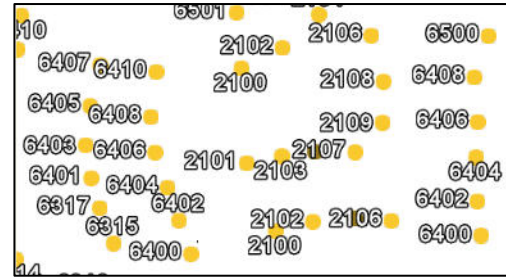
Whichever method you chose you need to set the *kml_link_href* to point to the *zipcodes.kml* file.

The former method will result in a workspace that looks something like this (***below***):



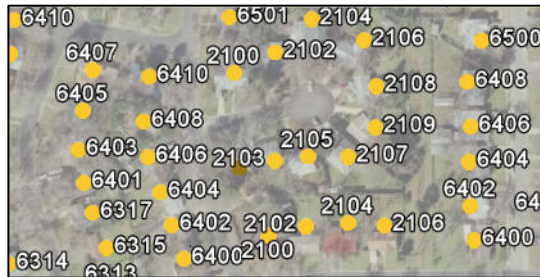
Open the output dataset in Google Earth (*right*).

Oh! That's not good. The zipcode features are obscuring the underlying background images.



In the zipcode MID/MIF to KML translation use a *KMLStyler* transformer to give the zipcode polygon features a colour of white (1,1,1) and an opacity of 0.6

Re-run the zipcode translation.



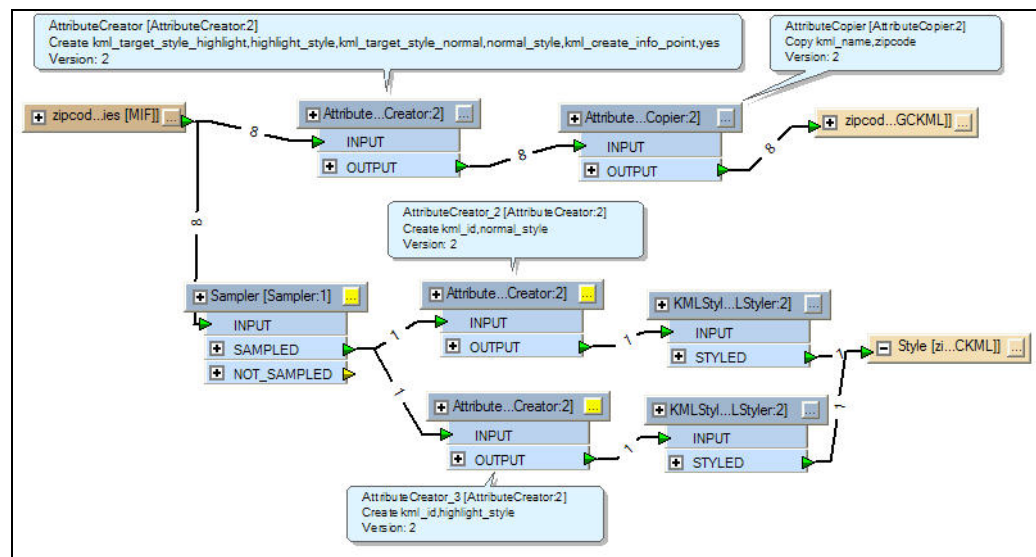
Re-open addresses.kml within Google Earth (*left*)

Notice how the zipcodes are now semi-transparent, without even having to re-run the address conversion workspace again!

That's because the address dataset merely links to the zipcode dataset and does not contain that data.

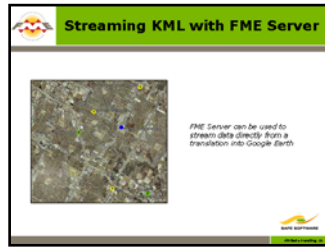
This illustrates how network links can be used to configure KML files that another team maintains, independently of the primary dataset.

Advanced: Enhance the zipcodes data with an icon and highlight color, as you did in Example 4.



Above: final zipcodes workspace

Streaming KML with FME Server



Data Streaming is a way to have Output KML Data open directly in Google Earth

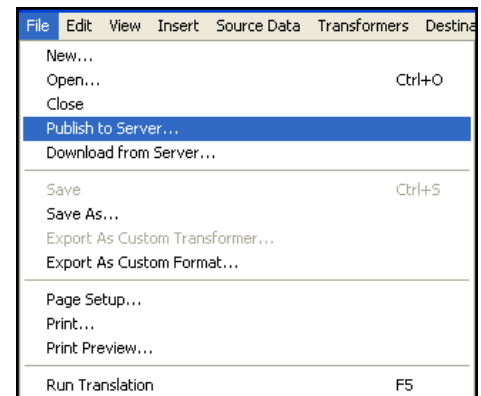
Because KML is a format for use by non-GIS users, it's unlikely that you would want such persons to use FME to create their KML. However, neither do you really want to be responsible for generating content for them.

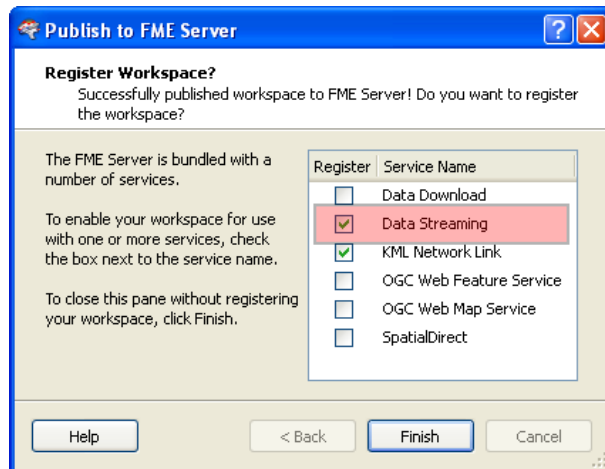
This is where FME Server functionality comes in useful. It allows a user to run a KML translation – without the need to use FME – and have the results streamed live into their KML browser.

Registering a Workspace for KML Streaming

There are two ways to register a workspace with FME Server to stream KML data; as a Data Streaming Service and as a KML Network Link Service.

Both methods are activated using the Publish to Server function (*right*):





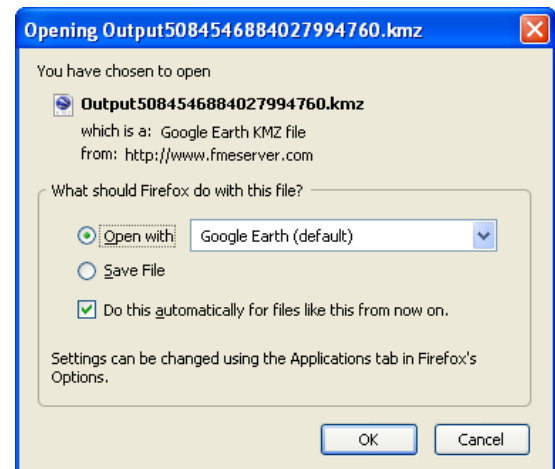
Data Streaming Service

Publishing a workspace as a Data Streaming Service (**left**) is basically a static solution.



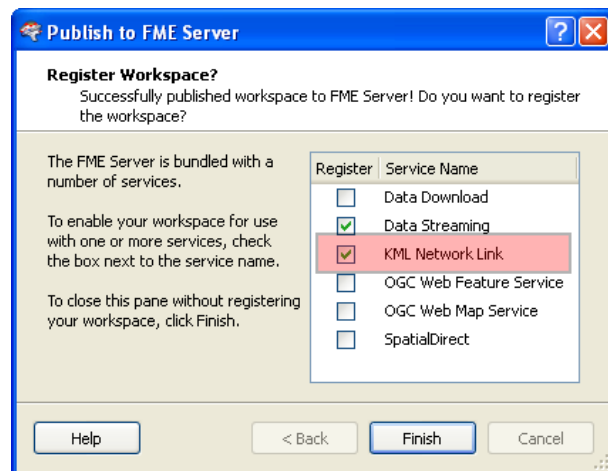
When an FME Server translation to KML is triggered – here (**above**) using the Run button – the user is presented with a dialog (**right**) asking to open the results directly in their default KML browser; in this case Google Earth.

This is a static solution because the output is fixed and unchanging. The user's view of the data will not be updated regardless of what might change within the original source data.



Access an example Data Streaming workspace on fmeserver.com using the link:

<http://www.fmeserver.com/fmeserver/services/fmedatastreaming/Samples/airports.fmw>

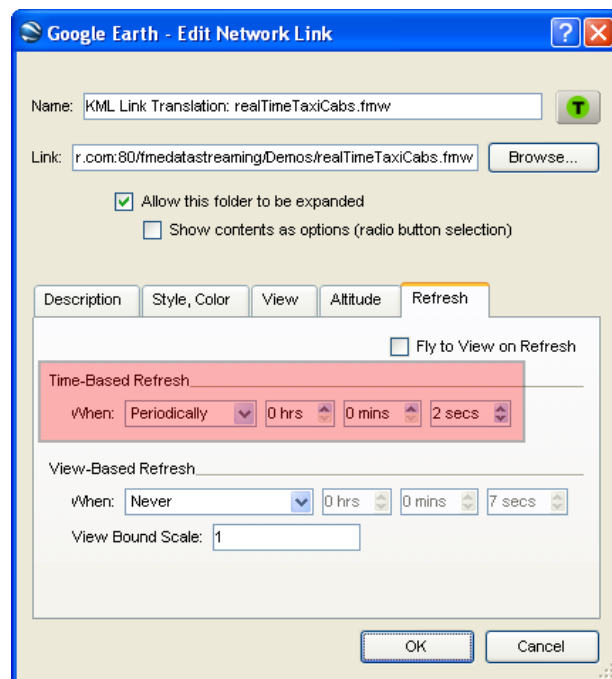


KML Network Link

Publishing a workspace as a KML Network Link (left) is basically a dynamic solution.

When the translation is triggered - again using the Run button (Right) – the data opens automatically in Google Earth...

Repository / Workspace	Title	Actions
Demos		Browse
realTimeTaxiCabs.fmw	Unset	Configure Run



But as a Network Link the properties dialog has options to refresh the data at regular intervals.

Left: Here the user has chosen to refresh the view every two seconds. This means a new request is sent to FME Server, the translation is re-run, and a new output dataset is streamed directly into Google Earth.

Access an example KML Network Link workspace on fmeserver.com using the link:

<http://www.fmeserver.com/fmeserver/services/fmekmlink/Demos/realTimeTaxiCabs.fmw>

Time Functions



To allow temporal mapping all features written to KML format can be assigned a number of time-oriented attributes.

Each feature written to a KML dataset – be it a line point or polygon – may have a related *TimePrimitive* element. This element can either be a single time stamp (i.e. this feature relates to time X) or a time span (i.e. this feature relates to the period from time X to time Y).

KML times adhere to the standard dateTime value defined in the XML specification. For detailed information the full XML dateTime specification can be found at:

<http://www.w3.org/TR/xmlschema-2/#dateTime>

For example a date of:

- 2004-05-22T15:41:00-08:00

...equates to:

- 3:41pm (Pacific Standard Time) on the 22nd May 2004.

In actual fact the time component of this is not obligatory.

Equally valid is:

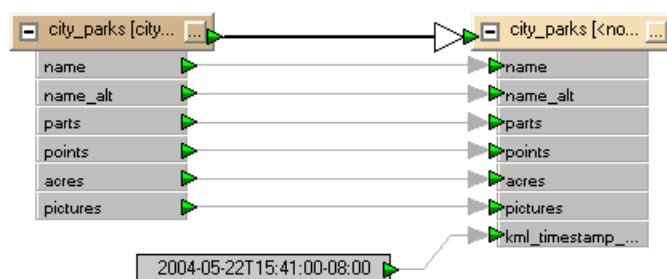
- 2004-05-22
(note the 2 digit month & day – 2004-5-22 is not valid)

FME does not validate KML dates to provide a warning or error in the log. You will only find that there is a problem when you open the dataset for display within Google Earth. To reduce problems use the DateFormatter transformer to construct dates in FME.

Time Stamps

KML time stamps can be set using the format attribute *kml_timestamp_when*

Right: A user setting a KML timestamp using a constant value for all features.



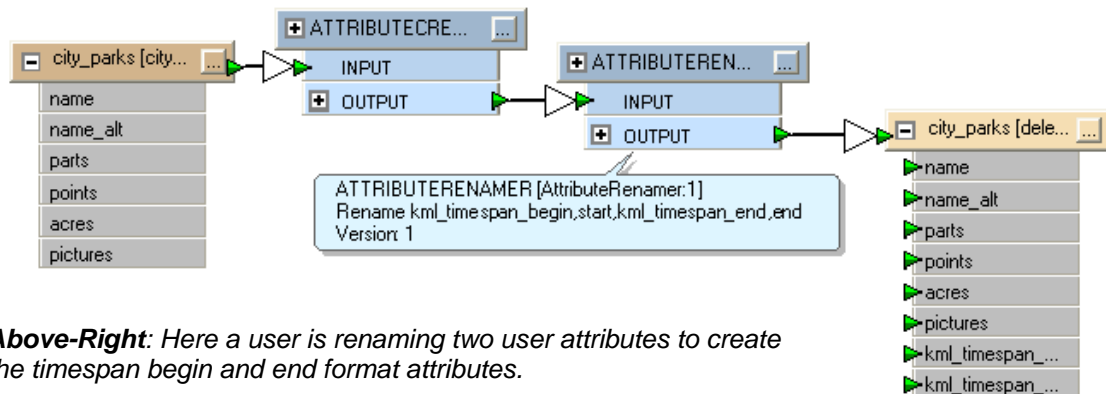
Below: The KML tag defining the timestamp.

```
<TimeStamp>
<when>2004-05-22T15:41:00-08:00</when>
</TimeStamp>
```

Time Spans

A time span is very similar to a time stamp, except that it defines a range of time instead of a single point.

To do this requires two format attributes: *kml_timespan_begin* and *kml_timespan_end*



Above-Right: Here a user is renaming two user attributes to create the timespan begin and end format attributes.

Which Google Earth Functions Use Time Stamps and Time Spans?

On opening a dataset containing TimeStamp or TimeSpan tags, Google Earth automatically detects the tags and adds an extra control for managing the display of these.



Above: In addition to the standard navigation controls, Google Earth has recognized a TimeSpan tag and added a control to manage its display.

In the Google Earth time navigation control, make particular note of the “play” or “run” button to the right-hand side of the control. Clicking this will cause the whole display to be animated.

The time bar will move slowly along the scale from the minimum to the maximum value, and features within the main display will be turned on and off according to whether their time stamp (or time span) falls within the current date range.

**Example 7 – Creating a Time Span Dataset for use in Google Earth**

The City of Interopolis disaster management team is tasked with analyzing the potential impact a hurricane would have upon the city.

In this example we will take a set of records for hurricane locations and create a time-based Google Earth dataset to help display the hurricane tracks. From this the city will be able to see just how close hurricanes come, and how often this occurs.

For the sake of simplicity, we'll start out by using just the data from 2005.

Setup

Set up a translation to convert the following GML (Simple Features Profile) dataset to KMZ format.

Format	GML Simple Features Level SF-0 Profile
Dataset	<u>C:\FMEData\Resources\FME and KML\hurricanes2005.gml</u>

Inspect the data to check out what attributes exist.

Transformations

Create a value for *kml_timespan_begin*.

You can use the *Concatenator* transformer to join together the values for the attributes YEAR, MONTH, DAY and HOUR – plus a set of constant values – to create a value that meets the XML timeDate specification, i.e. 2005-05-22T00:00

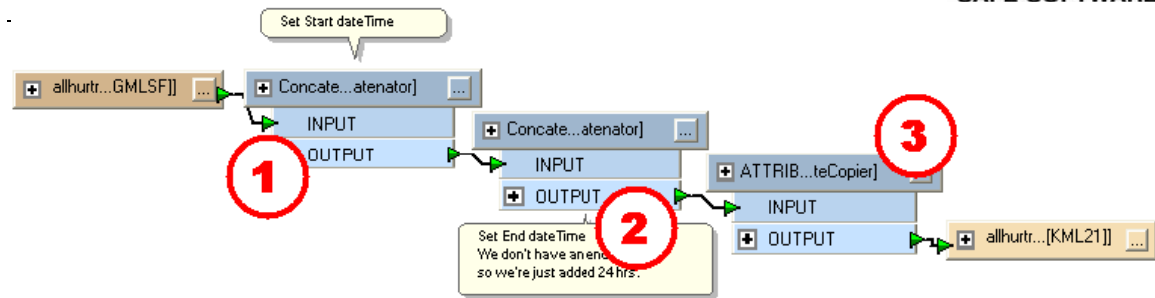
Create a value for *kml_timespan_end* that is the same as the begin time, plus twenty four hours (T:23:59). This is easier than adding a single day, since you would then have to check for dates that rolled over from one month to the next (ie 31st January +1 day is not 32nd January!)

Two final steps:

Firstly set the *kml_icon* format attribute to use the icon *hurricane.gif*.

You can either use the *KMLStyler* or the format attribute, but should copy the GIF file to the same folder as the workspace and use its name as the value.

Secondly, copy *kml_timespan_begin* to *kml_name* to provide a useful label for each point.

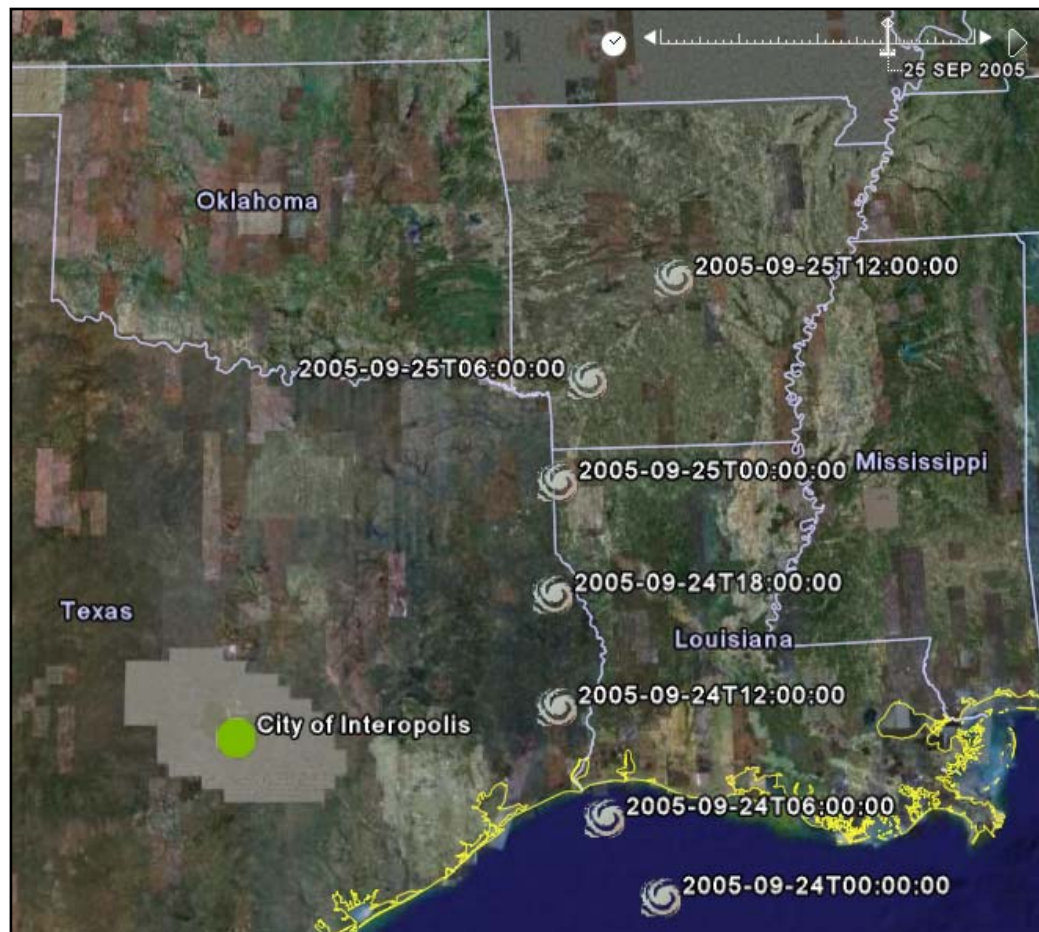


Above: The solution workspace.

- 1) The first Concatenator creates the value for `kml_timespan_begin`
- 2) The second Concatenator creates the value for `kml_timespan_end`
- 3) The AttributeCopier copies the time of the hurricane to the `kml_name` attribute, in order to provide a useful label within Google Earth.
- 4) [Not Shown] The `kml_icon` attribute is set to point to a user defined icon using a constant value.

Run the workspace and open the data in Google Earth. Press the timescale “play” button to get an animated display of hurricane activity.

Below: On September 24th and 25th, 2005, Hurricane Rita came perilously close to the City of Interopolis.





Advanced Exercise – Dynamic Highlighting

Continue to develop the workspace from example 7, by using what you have learned about KML to highlight the counties that the hurricane tracks pass over as they move across the land.

Start by adding the following dataset to your workspace:

Format Autodesk AutoCAD DWG/DXF
Dataset C:\FMEData\Data\GovtBoundaries\StateCountyBoundaries.dwg

You will also need to add another destination dataset to the workspace (call it counties).

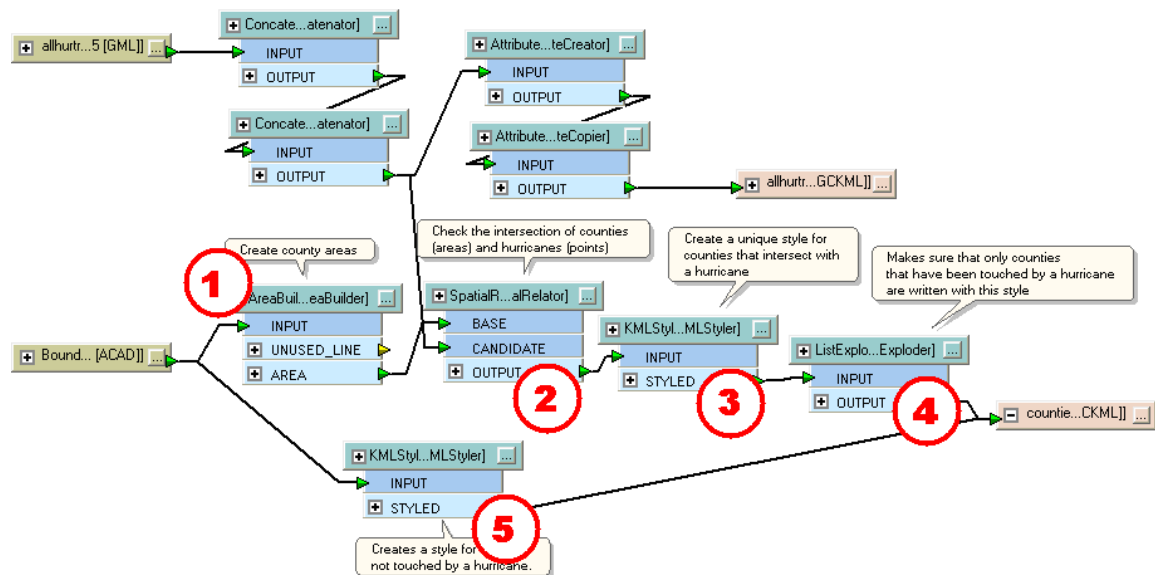
Transformations

You will need to create two different KMLStyles: one to display all the county boundaries and another to be assigned to the counties that had a hurricane pass over them.

You can use the *SpatialRelator* transformer to test whether or not a hurricane moves into a county. (Hint: you will need to convert the county boundaries into areas before you can do this).

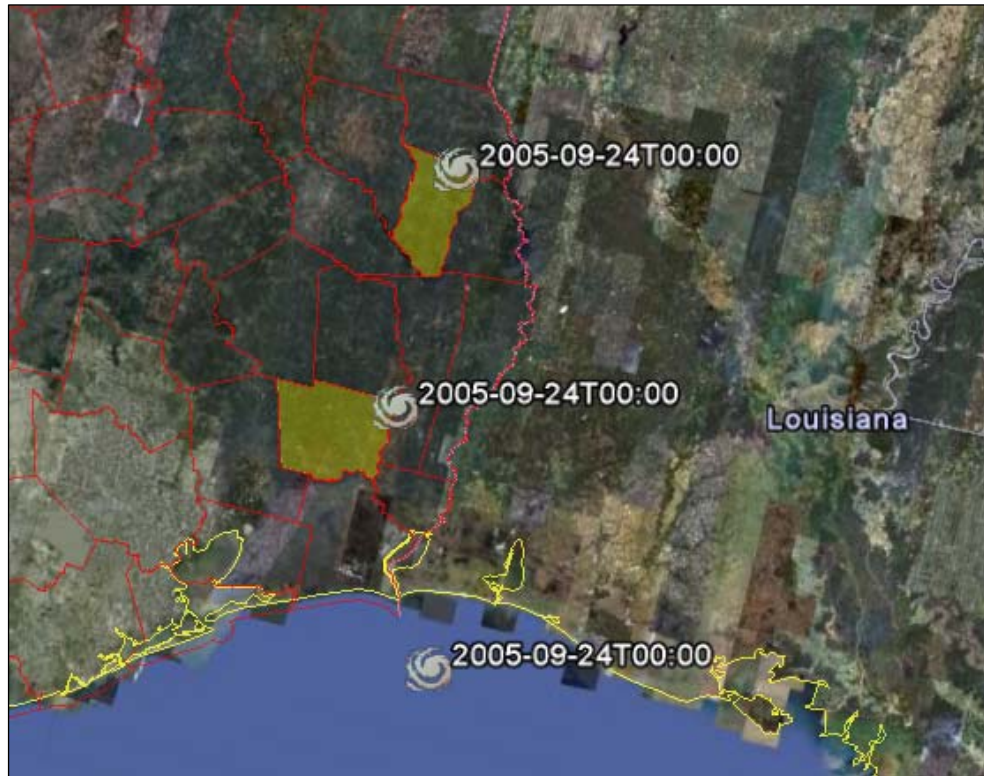
The *SpatialRelator* determines topological (spatial) relationships between sets of features. When one of the comparisons is true a list attribute is added that contains all the attributes of both the BASE and CANDIDATE features that passed. In other words, any county which intersects a hurricane will be given the timespan attributes from the corresponding hurricane feature.

You can use a *ListExploder* transformer to create features for each of the counties touched by a hurricane by setting the List Attribute parameter to the list created by the *SpatialRelator*.



- 1) The *AreaBuilder* creates polygons for each of the counties
- 2) The *SpatialRelator* checks which counties are intersected by a hurricane
- 3) The *KMLStyl* sets the style for the intersected counties
- 4) The *ListExploder* sends the intersected counties to the output file
- 5) The *KMLStyl* (at the bottom of the workspace) creates a style for the county boundaries.

Run the workspace and open the data in Google Earth. When you press the timescale “play” button, you will see some of the counties change colour as a hurricane passes over them.



Above: Shows two counties that have been highlighted as a hurricane moves over.

Session Review



This module was designed to assist you in using FME to generate useful and interesting KML format datasets.

What You Should Have Learned from this Module

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

Theory

- **KML** is a format that is to spatial data as HTML is to plain text.
- **KML** has a hierarchical structure with a number of primary node types.
- **KML** is restricted to using the coordinate system LL84 (or EPSG:4326), all features must be three-dimensional, and all must have a unique ID number.

FME Skills

- Use FME to **convert vector data** to KML format.
- Use FME to **convert raster data** to KML format.
- Use the **KMLStyler transformer** to apply basic KML symbology.
- **Describe** features within a KML dataset using the format attributes kml_id, kml_name and kml_description
- Use special KML feature types such as **Folders**, and **StyleMaps**
- Create **Network Links** to reference one KML dataset from another
- Create and use **KML data streams** on FME Server
- Create KML **Timestamps and Timespans** to describe data through time

