FME Desktop ®

Raster Pathway Training

FME 2015 Edition

Safe Software Inc. makes no warranty either expressed or implied, including, but not limited to, any implied warranties of merchantability or fitness for a particular purpose regarding these materials, and makes such materials available solely on an "as-is" basis.

In no event shall Safe Software Inc. be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of purchase or use of these materials. The sole and exclusive liability of Safe Software Inc., regardless of the form or action, shall not exceed the purchase price of the materials described herein.

This manual describes the functionality and use of the software at the time of publication. The software described herein, and the descriptions themselves, are subject to change without notice.

Copyright

Revisions

Every effort has been made to ensure the accuracy of this document. Safe Software Inc. regrets any errors and omissions that may occur and would appreciate being informed of any errors found. Safe Software Inc. will correct any such errors and omissions in a subsequent version, as feasible. Please contact us at:

Safe Software Inc.
Suite 2017, 7445 – 132nd Street
Surrey, BC
Canada
V3W1J8

www.safe.com

Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice.

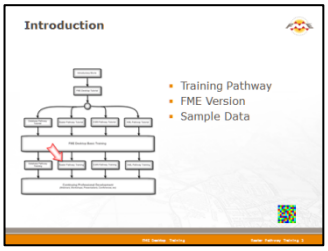
Trademarks

Documentation Information

The following is information about this document and the systems used to create it.

Document Name: FME Desktop Raster Pathway Training
Updated: March 2015
FME Version: FME 2015.0, WIN32
Operating System: Windows 2012, 64-bit.

Introduction



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

Raster Pathway

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

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

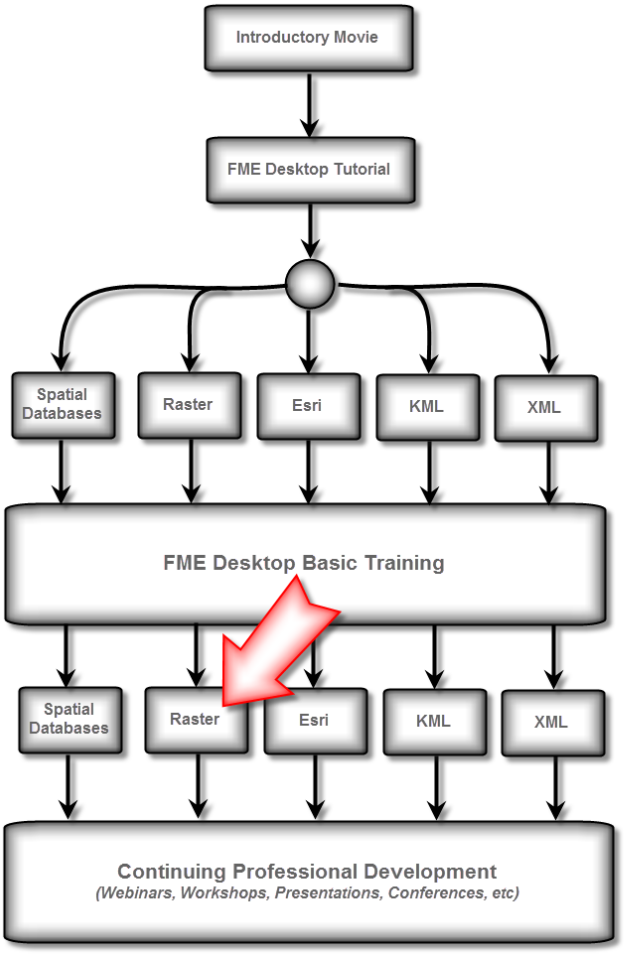
FME Version

This training material is designed specifically for use with FME2015. You may not have some of the functionality described if you use an older version of FME.

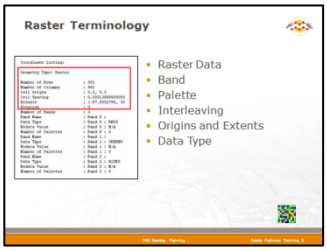
Sample Data

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

www.safe.com/fmedata



Raster Terminology



A review of FME's Raster Terminology

Terminology

This section is a brief review of FME terminology; you should already be familiar with most of this.

Raster Data

A raster dataset is a grid of data organized into rows and columns. Each data value – a location at the intersection of each row and column – is known as a pixel or *cell*.

Image datasets contain data values for each cell that represent a color or degree of shading. Numeric datasets represent spatial data that contains measured values.

Band

A raster dataset can contain a number of different "layers" known as *bands*.
Another way of looking at this is to say each cell can consist of a number of values.

Palette

A *palette* is a lookup table of discrete keys to color or string values; a means to map the value of a raster cell to a particular color or descriptive value.

Interleaving

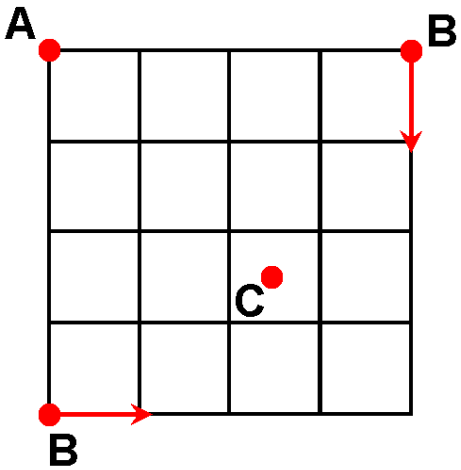
Interleaving is a technique that specifies how data is organized within a multi-band raster dataset.

Raster Origins and Extents

For an entire raster dataset, the origin point is the upper left corner (A).

The extents of a raster are the lower left X and Y of the lower left cell, to the upper right X and Y of the upper right cell (B).

For individual cells, 0.0, 0.0 is the lower left corner of the cell, and 1.0, 1.0 is the upper right corner. The origin point is usually the cell center; 0.5, 0.5 (C). This is the cell origin that FME uses.



Raster Data Type and Interpretation

Raster *Data Type* refers to the method by which values are stored within a raster dataset, for example as an integer number, or as a real number.

Raster *Interpretation* indicates how the raster data type should be interpreted; is it a numeric value, a grayscale value or a color within a multi-band dataset?

Like vector datasets, raster datasets may be either File based datasets or Folder based datasets.

File Based Raster Datasets
A file based dataset stores the complete dataset within a single file. File based datasets can range from a simple single band GeoTIFF to a multi-band, multi-resolution HDF4 dataset.

Folder Based Raster Datasets
A Folder based dataset stores the complete dataset within a single folder. An example would be satellite formats such as RADARSAT -2 product format, or IKONOS standard products where the individual bands of data are stored as separate GeoTIFF files and read as separate features.

Features
With raster data, an entire dataset is a single feature, which makes it easier to handle multiple raster datasets. Conversely, a raster feature is not the smallest unit of data, and is made up of smaller units such as bands, palettes and cells.

File Naming Conventions
With raster, each feature results in a separate output file. Therefore each raster Writer requires a mechanism to name the output files without overwriting each other.

FME adds a Feature Type 'Fanout' to each raster Writer, fanning out by the fme_basename format attribute. This ensures that each output file will get the same name as the source files.

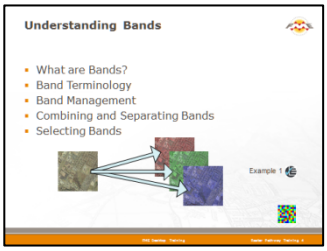
A simpler method is to use the Feature Type name as usual, but to append a numeric suffix. This is the default naming method used if the Feature Type Fanout is turned off.

The different naming conventions and how they would be applied to a TIFF to JPEG translation.

Source	**Destination**	
	Fanout Method	**Suffix Method**
roads.tif	roads.jpg	JPEG.jpg
rivers.tif	rivers.jpg	JPEG_1.jpg
rail.tif	rail.jpg	JPEG_2.jpg

As with vector data, existing files will be overwritten if name collisions occur. For that reason, using multiple Writer instances targeted at the same directory is considered an unsafe practice.

Understanding Bands



Bands are one of the subjects that cause most confusion in using raster data with FME

Although it's possible to get by without, to master raster data translations with FME, it is necessary to understand the concept of bands, and how to manage them.

An understanding of bands may be required, for example, to do each of the following tasks:

- Translate raster from one format to another
- Overlay vector data into raster
- Resample raster resolution
- Mosaic raster datasets together
- Use transparency and compositing
- Carry out raster algebra on cells

What are Bands?

Bands can be considered as the layers in a raster dataset. Alternatively, you might find it easier to think of each cell having a number of values (like an FME list attribute).

For example, a color raster dataset commonly has three bands; Red, Green, and Blue.
You could say there are red, green, and blue layers; or that each cell has a value for each band.

Band Terminology

Bands are numbered from 0 onwards in FME. A single-band raster will have one band (0), whereas an RGB color raster will have multiple bands (0, 1, 2).

```
Number of Bands          : 3
Band Name                : Band 0 :
Data Type                : Band 0 : RED8
Nodata Value             : Band 0 : 0
Band Name                : Band 1 :
Data Type                : Band 1 : GREEN8
Nodata Value             : Band 1 : 0
Band Name                : Band 2 :
Data Type                : Band 2 : BLUE8
Nodata Value             : Band 2 : 0
```

Band Management

Workbench includes transformers to add or remove bands to/from a raster feature.

However, this functionality is rarely used. Changing the number of bands usually involves changing the interpretation of the feature (say from color to numeric), for which there is a set of specialized transformers that handle the bands automatically.

Combining and Separating Bands
Workbench also has transformers to combine or separate bands.

Combining Bands
Sometimes raster imagery is delivered as multiple files, each of which represents a single band. A user may wish to merge these into a single multi-band raster dataset.

Multiple raster features can be combined into multiple bands on a single feature, using the *RasterBandCombiner* transformer.

The output from this transformer is a single raster feature, with the number of bands equal to the sum of the input features.

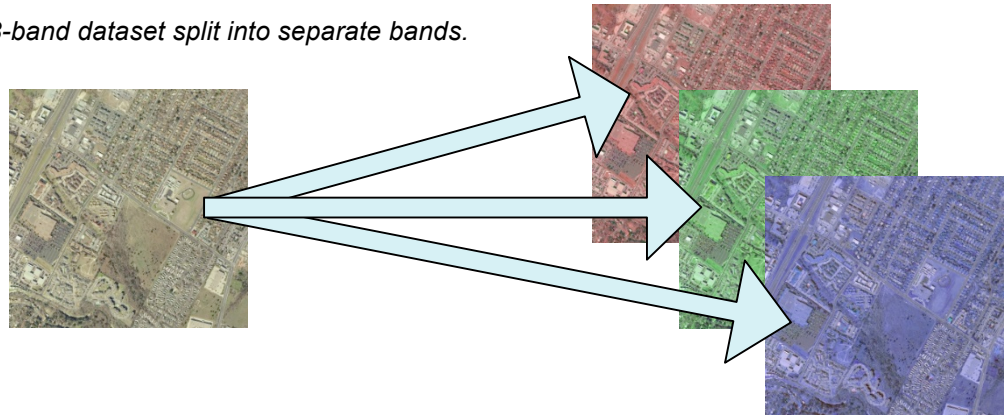
Bands to be combined must overlap exactly, and have the same number of columns and rows.

The order of the input features determines the order of the bands in the output feature. A *Sorter* transformer may be used to arrange the raster features into the desired order.

Separating Bands
Sometimes a user may wish to separate out a multi-banded dataset, into separate one band files.

This can be achieved using the *RasterBandSeparator* transformer.

A 3-band dataset split into separate bands.

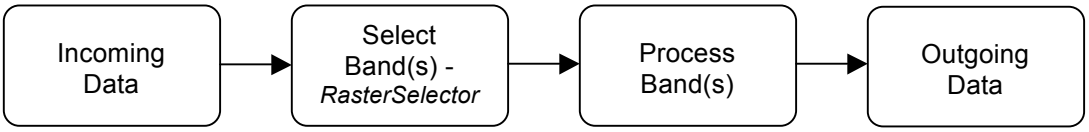


The actions carried out on the bands and palettes will depend upon the exact user requirements. There are a number of potential scenarios:

Source	Required	Action
Multi-Band, Multi-Palette	Single-Band, Multi-Palette	Split by Band
Multi-Band, Multi-Palette	Single-Band, Single-Palette	Split by Band and Palette
Multi-Band, Multi-Palette	Multi-Band, Single Palette	Split by Palette
Single-Band, Multi-Palette	Single-Band, Single-Palette	Split by Palette
Multi-Band, Single-Palette	Single-Band, Single-Palette	Split by Band

Selecting Bands
Many raster restructuring transformers have the ability to work on selected bands (or palettes).

However, the selection of bands is defined as a separate step. The reasoning is that a user may wish to carry out multiple operations on the same band, and not want to select them each time.



The selection of bands is carried out using the *RasterSelector* transformer.

RasterSelector
The *RasterSelector* transformer defines the active bands (and palettes) that subsequent raster transformers will operate upon.

For example, a user who wishes to offset cell values must first use the *RasterSelector* to determine which band(s) to offset.

The *RasterSelector* defines bands and palettes as a string in the format B:P where B is the band number and P the palette.

A single number defines a band upon which further operations are to be carried out; in this case band 0.

A semi-colon delimited list defines multiple bands; here band 0 and band 2 (Red and Blue in an RGB)

The word ALL can be used to denote all bands.

Notice that there is the option to select the bands from an attribute, as well as hard-coding them.

Example 1: Raster Bands	
Scenario	FME user; City of Interopolis, Planning Department
Data	InteropolisCentre (PNG format), City Parks (MapInfo TAB)
Overall Goal	Overlay vector park data onto a raster dataset
Demonstrates	Raster Bands and Band Management
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Raster1a-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Raster1b-Complete.fmw

The overall goal here is to get a raster image onto which vector features (representing parks) have been superimposed. The vector features must be semi-transparent, in order to see the raster data under them. To do this requires understanding and manipulation of raster bands.

1) Start FME Data Inspector
Start the FME Data Inspector and inspect the dataset:

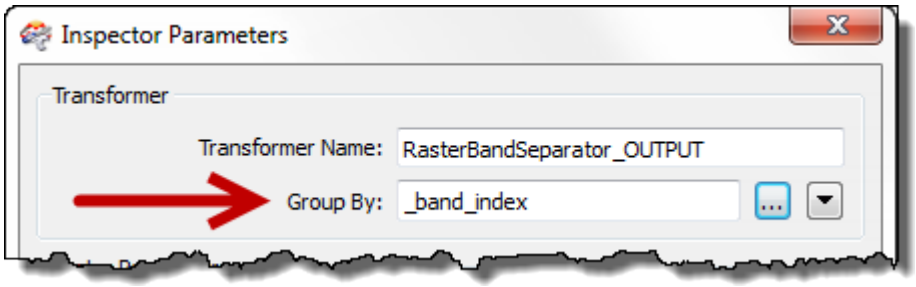
Reader Format PNG (Portable Network Graphics)
Reader Dataset *C:\FMEData\Data\Raster\InteropolisCentre.png*

Query the dataset to find out how many bands there are, and of what type.
As a matter of interest, also check out the cell origin and spacing.

2) Start Workbench and Add Raster Reader
Start Workbench and begin with an empty workspace. Add a new Reader (**Readers > Add Reader**) to read the same PNG raster data defined in the previous step.

3) Add RasterBandSeparator
This part isn't important for the actual task at hand but, as a test, add a *RasterBandSeparator* transformer to the workspace and connect it up to the PNG Feature Type.

Connect an *Inspector* transformer to the *RasterBandSeparator*. Open the parameters dialog and set **Group-By** to be *_band_index,* so each output gets a different layer in the FME Data Inspector.



Run the workspace. There are three output raster features, each of which represents a band of the original dataset. Notice each has small 'nodata' holes, where the value for that color is zero.

4) Add Vector Reader
OK. Back in Workbench remove the *RasterBandSeparator* and *Inspector* transformers. They were just used to demonstrate a concept.

Now select **Readers > Add Reader** from the menubar to add a second Reader, this time to read the MapInfo TAB parks dataset. This is the data that will be superimposed onto the raster image.

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

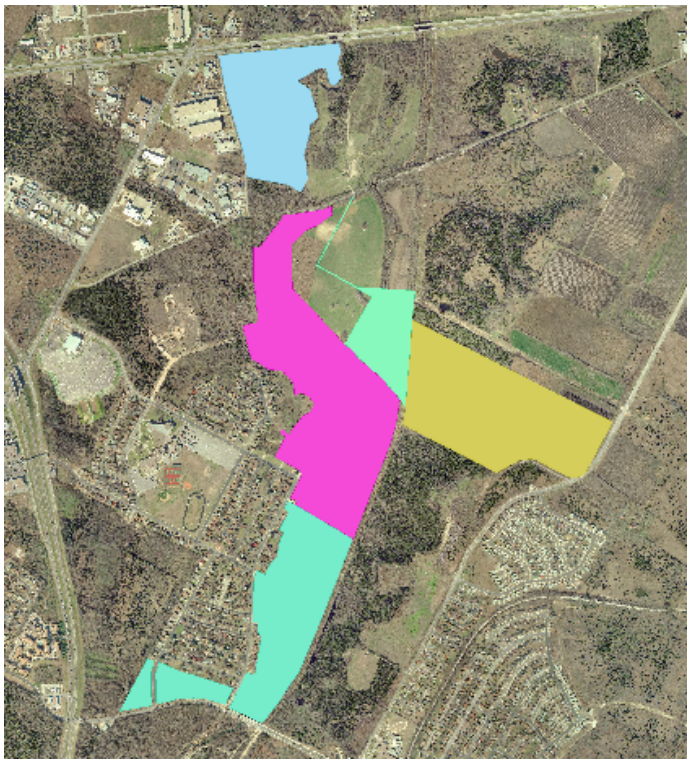
5) Add VectorOnRasterOverlayer
Add a *VectorOnRasterOverlayer* transformer to the workspace.

Connect the parks Feature Type to *VectorOnRasterOverlayer*:VECTOR and the PNG image to *VectorOnRasterOverlayer*:RASTER.

Connect the *VectorOnRasterOverlayer* output port to an *Inspector* transformer.

Run the workspace. Inspect the output data. Notice that the parks data has been overlaid onto the raster image. Query a cell and the value for all cells in any park will be the same.

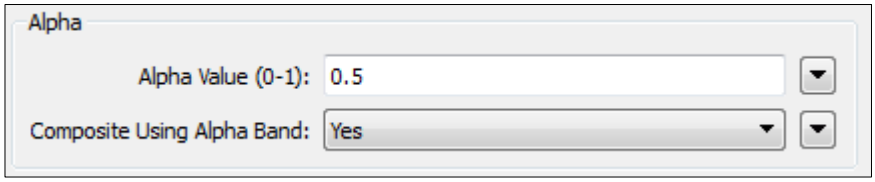
That's because the parks have overwritten the raster. They are not semi-transparent and this must be fixed to obtain the required output.



6) Set Parameters
Open the *VectorOnRasterOverlayer* transformer parameters dialog.

Notice that there are parameters for applying an alpha value to the incoming vector data. Set:

Alpha Value	0.5
Composite Using Alpha Band	Yes



The alpha value is a degree of transparency, from 0.0 (fully transparent) to 1.0 (fully opaque). But, by itself, this parameter will have little effect except lighten the color of the parks. We must also set the Composite parameter to tell FME to blend the vector features into the raster data.

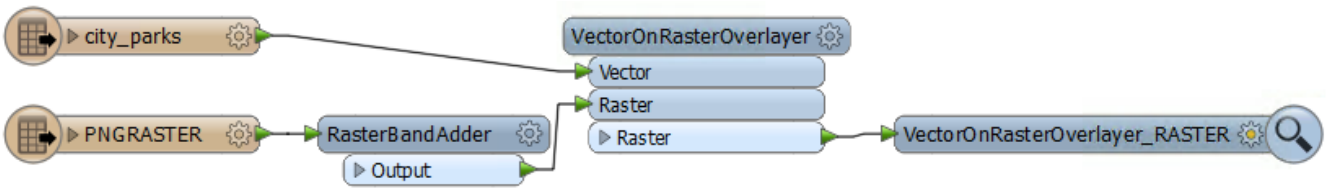
Re-run the workspace. This time it will fail with the following error:

```
VectorOnRasterOverlayer(VectorToRasterFactory): All input rasters must have
exactly one alpha band selected when compositing with the alpha band
```

If you didn't know, transparency is stored as a separate band (the alpha band) in a raster dataset.

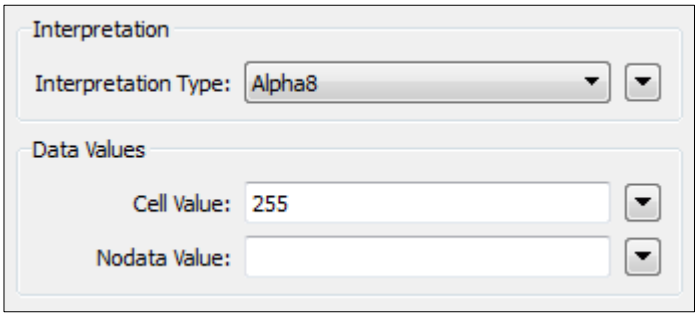
The problem here is that the workspace is trying to apply transparency to a raster feature that does not have an alpha band on which to store that information (remember it has only Red, Green and Blue). We can fix this by adding an alpha band.

7) Add RasterBandAdder
In Workbench, add a *RasterBandAdder* transformer. Connect it between the PNG Reader Feature Type and the *VectorOnRasterOverlayer* transformer.



Open the properties dialog for the RasterBandAdder transformer. Add a band of:

Interpretation Type Alpha8
Cell Value 255



"Alpha8" means the raster feature can now have a transparency value for each cell, from 0 (zero) to 255.

8) Run Workspace
Re-run the workspace.

Inspect the data. The park features will now be blended into the raster, rather than overwriting it.

Advanced Tasks
If you have time, try the following challenges:

a) The parks are all different colors. What transformer can you use to make them all the same shade of green?

b) Add a third Reader to the workspace to read hydrography data:

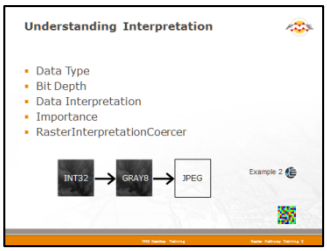
Reader Format MapInfo MIF/MID
Reader Dataset *C:\FMEData\Data\Hydrography\HydrographyLine.mif*

Connect the hydrography data into the *VectorOnRasterOverlayer*:VECTOR port.

How clear is the hydrography in the output raster? What transformer might be able to improve it? Maybe it should not be transparent like the parks? How could that be achieved?

Check the completed workspace to find out how these problems can be solved.

Understanding Interpretation



Interpretation is another subject that causes confusion in using raster data with FME

Like bands, interpretation is another concept that it is necessary to understand to make full use of FME raster functionality.

An understanding of interpretation may be required, for example, to do each of the following tasks:

- Translate raster from one format to another
- Mosaic raster datasets together
- Carry out raster algebra on cells
- Compress data to minimize file sizes

In general, the properties that define a cell value are:

- Data type
- Bit depth
- Data interpretation.

Data Type

Data Type and Bit Depth are key components of a raster interpretation.

Data Type classifies the nature of the data values. For example, the following are data types:

- Unsigned Integer (positive values only)
- Signed Integer (negative and positive values)
- Real (floating point) number
- String

Bit Depth

Bit-depth refers to the number of bits used to store values; this affects the range of numbers and the length of string that can be stored.

Together, Data Type and Bit Depth combine to define what values can be stored in a raster cell.

For example, a value could be stored as an 8-bit unsigned integer, UInt8.

In binary, 2^8 = 256 meaning there are 256 possible values. Including zero this gives a range of values of 0-255.

128	64	32	16	8	4	2	1	= 255
1	1	1	1	1	1	1	1	= 8

An 8-bit signed integer (Int8) would give 256 values from -127 to +128
Int16 would give 65536 values (-32767 to 32768 for a 16-bit signed integer).

Data Interpretation
Interpretation is used to denote what exactly Data Type and Bit Depth values mean.

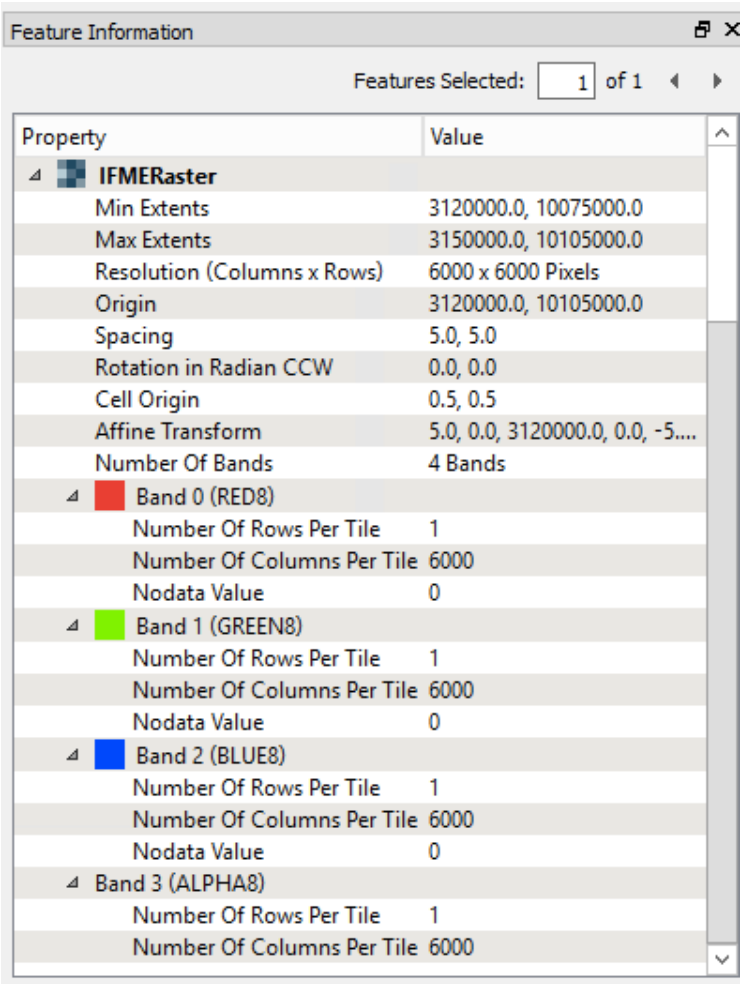
For example, the Data Type 'UInt8' can be used for any one of:

- A simple band of numeric values
- A range of shades on a grayscale image
- A color band in an RGB image
- A set of transparency values in an image
- A simple band of alphabetic values

The actual use in any given dataset is denoted by the Data Interpretation:

Use	Interpretation
Numeric Values	UInt8
Grayscale shade	Gray8
Color band in RGB	Red8, Green8, or Blue8
Transparency Values	Alpha8
Alphabetic Values	String

For example, here the data is UInt8 (8-bit integers). The band interpretations (RED8, GREEN8, BLUE8, and ALPHA8) denote three color bands and a transparency band.



When there are multiple bands in a color dataset, it's usual to describe the dataset interpretation as the sum of the band interpretations.

Dataset	***Band Interpretations***
RGB24	*Red8, Green8, Blue8*
RGBA32	*Red8, Green8, Blue8, Alpha8*
RGB48	*Red16, Green16, Blue16*
RGBA64	*Red16, Green16, Blue16, Alpha16*

So the raster here would have a dataset interpretation of RGBA32.

Importance
In general, these properties of a raster dataset are important because it is a restriction on the quality and precision of the data. The more bits of data that are stored, the more information can be held, but at the cost of larger datasets.

For example:

- A 16-bit color image will have a wider range of different shades (65536 colors) than an 8-bit color image (256 colors).
- A 64-bit real number will be able to store values with a greater precision, and a higher maximum value, than a 32-bit integer
- A signed integer will be able to store negative numbers

However, these issues are related to the data quality itself. FME is also concerned about being able to translate data and transform it; and this has its own set of issues.

Interpretation Incompatibility
One issue that arises is that not all raster formats support all data types. If a user is unaware of the concept they might try to translate data to a format that does not support that particular interpretation and data type.

For example, you could not write a raster DEM represented by a numeric value (say INT32) to an image format such as JPEG.

This can be particularly confusing to a user because when they open the DEM in a raster viewer it has a range of color. In fact the color is being generated by the inspection tool and is not part of the data itself.

The solution, of course, is to manipulate the data type and interpretation of the data. However, because data type is so integral to the quality of a dataset, and because there are different ways of doing the adjustment, FME will not do this automatically.

Rather, FME will record an error and stop the translation. The user must determine the correct interpretation, and convert the data with a transformer. This is why it is so important to be aware of this concept!


INT32
GRAY8
JPEG

Reducing Bit Depth
In general, higher bit-depth datasets are larger. They take up more space and consume more system resources to work with.

Therefore, a user may wish to trade off some of this size against having smaller and more efficient datasets. This is particularly true when the data is to be distributed via a network or the Web.

By reducing bit depth, in effect the data is being compressed into a smaller range of values.

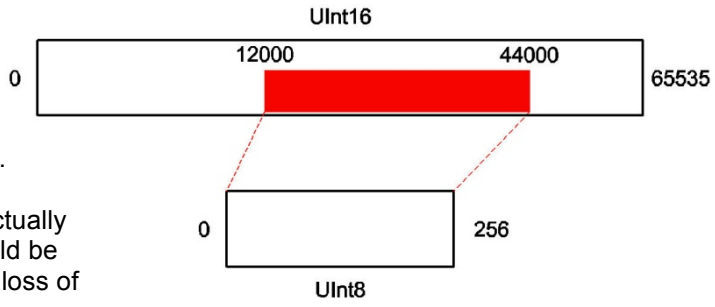
For example, take a raster dataset stored as UInt16 (a 16-bit unsigned integer).
This allows cell values between 0 and 65535. The user decides that he does not require such a large range of values and uses FME to convert the data to UInt8 with a range of 0 to 256.

However, in doing so, FME compresses the values from the higher range to fit them into a lower range, and this causes an obvious reduction in data precision!

This is because different values from the 16-bit dataset will end up squashed into the same value in the new 8-bit dataset. So this technique should only be used where the output doesn't require the full color or tone range provided by the higher bit-depth.

Note that, in order to preserve as much precision as possible, FME will only compress data where values exist.

Here, the values from 12000 to 44000 are compressed into a range of 0 to 256.



In this way, if the starting values were actually between 10,000 and 10,256, then it would be possible to reduce bit depth without any loss of precision, just a saving in file size.

However, reducing bit depth is problematic when a raster dataset uses zero to denote a cell where no data exists. Then FME wouldn't know what the real range of data was. Such values could cause an unnecessary loss of precision, or end up being merged with true data values.

Increasing Bit Depth
Increasing the bit-depth of a dataset is possible by stretching the existing range of values across a larger range of numbers.

For example, take a raster dataset stored as UInt8 (an 8-bit unsigned integer). The user decides to convert the data to UInt16. FME will expand the data to fit the higher range.

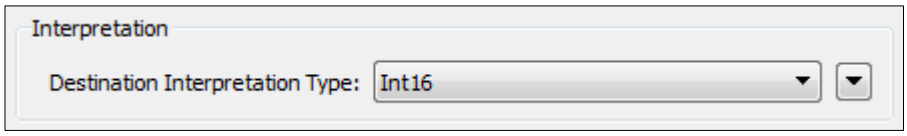
However, when expanding values to a higher range you can't really describe the data as more precise, despite the increase in bit-depth. That's because multiple values from the source are grouped together on the destination.

For example, an 8-bit color image expanded to 16-bit still only has a maximum of 256 different color values; they are just now spread over a wider range with gaps between.

So, the only reason to do an increase in precision like this is when further operations are to be carried out that do require this larger bit-depth.

RasterInterpretationCoercer
The *RasterInterpretationCoercer* transformer is the means by which a user can change the underlying interpretation of raster geometry.

The *RasterInterpretationCoercer* has a parameter to select the destination interpretation type.



By setting this to various values, in combination with different source data, this transformer can be used to:

- Increase Bit Depth
- Reduce Bit Depth
- Change Data Type
- Change Data Interpretation

Changing the data interpretation is probably the most interesting, since a user may convert numeric data to a true color image raster; or vice versa.

Conversion Options
The transformer has a number of options that define how data is manipulated as it is converted from one interpretation to another. Therefore it is important to not just understand the concepts involved, but also to be aware of the structure of the data being translated and the purpose of the transformation.

The two main operations are Casting and Scaling.

- Casting

Casting is when a range of data is simply moved into a new scale, acting as if there was no change in interpretation.

For example, casting the values 0 to 300 into an INT8 interpretation (range 0-255) causes values 256 or higher to exceed the limit. What happens then depends on the type of cast.

A Bounded Cast will cap all values at 255; so values 256-300 in the source become 255 in the output. Non-bounded casts will rollover the excess values to the bottom end of the scale; so values 256-300 in the source become 0-44 in the output!

- Scaling

Scaling is when the range of source data is either compressed or expanded in order to fit exactly into the new scale. There are two types of scaling.

Scaling by Data Values means that the range of existing values is scaled to fit exactly the new scale. For example, scaling values 0 to 300 into an INT8 interpretation will compress that range of values into the 0-255 range.

Scaling by Data Type means that the full range of possible values are used as the scale. For example, if the 0 to 300 is stored as UINT16, then the full possible range of values (0 to 65535) is compressed between 0-255. In this case most original values will become 0 or 1!

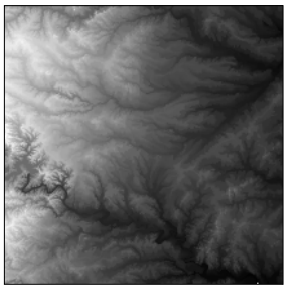
Example 2: Raster Interpretation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Elevation Model (Raster DEM, CDED format), JPEG
Overall Goal	Create a raster image from a DEM dataset
Demonstrates	Raster Interpretation
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Raster2-Complete.fmw

The overall goal here is to take a raster DEM model, and convert it into an image that a user can use in presentation media (document or slide presentation). The preferred format is JPEG.

1) Start FME Data Inspector
Start the FME Data Inspector and inspect the dataset:

Reader Format Canadian Digital Elevation Data (CDED)
Reader Dataset *C:\FMEData\Data\ElevationModel\RasterDEM-250K.dem*

Query the dataset to find out what the dataset interpretation is (and how many bands there are).



The FME Data Inspector does depict the data like this; can you tell how and why it is doing this?

2) Start Workbench
Start Workbench. Create a translation from the CDED dataset to JPEG format.

Reader Format Canadian Digital Elevation Data (CDED)
Reader Dataset *C:\FMEData\Data\ElevationModel\RasterDEM-250K.dem*

Writer Format JPEG (Joint Photographic Experts Group)
Writer Dataset *C:\FMEData\Output\DemoOutput*

3) Run Workspace
Run the workspace.

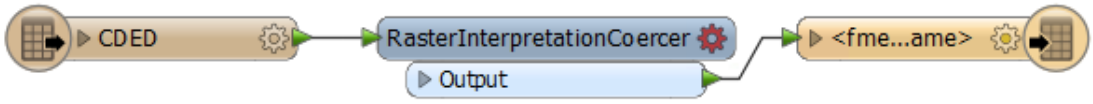
Does the workspace run to completion? Is the JPEG output created?
Why?/Why not?

4) Add RasterInterpretationCoercer
The workspace fails with the following error message:

```
JPEG writer: Unsupported band interpretation 'INT32'. Please convert the
interpretation to one of the following valid types: GRAY8, GREEN8, RED8, BLUE8
```

This is because the interpretation of the source data (INT32) is not supported by JPEG format. To write to JPEG, the data must be coerced into a new interpretation.

Add a *RasterInterpretationCoercer* transformer between Reader and Writer Feature Types.



5) Set Parameters
Open the *RasterInterpretationCoercer* properties dialog.

Experiment using different interpretation types, including:

- RGB24
- RGB48
- GRAY8
- RED8
- INT8

Some of these will work, some will be less successful! Can you guess in advance which types will work with this data translation?

Remember that the translation is from a numeric format to a color image, so if the “Numeric to Color” conversion option is grayed out, you can be fairly sure the interpretation will not be valid!

In particular, use the FME Data Inspector to open up outputs from the RGB24 and GRAY8 interpretations. Is there a difference in the number of bands, and the band interpretations?

6) Set Parameters
Re-open the *RasterInterpretationCoercer* properties dialog.

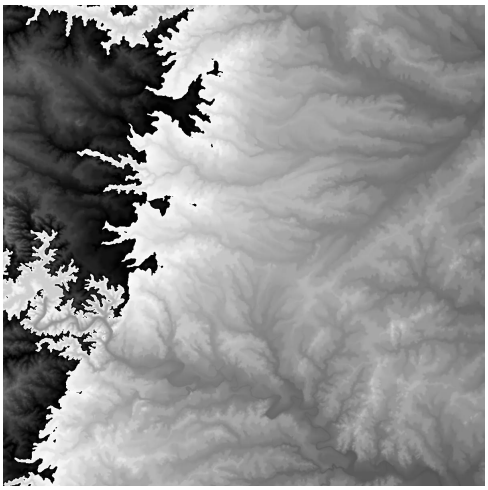
This time experiment using different conversion options:

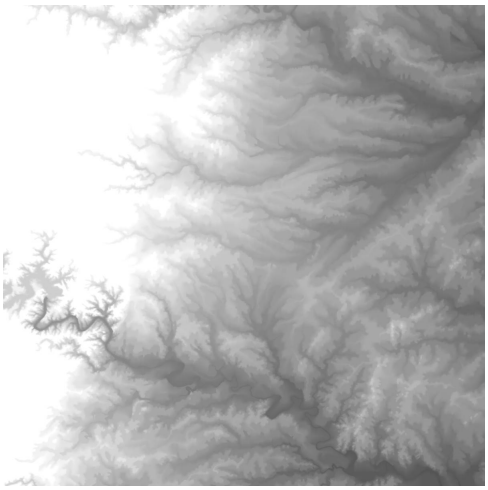
- Cast
- Bounded Cast
- Scale by Data Values
- Scale by Data Type

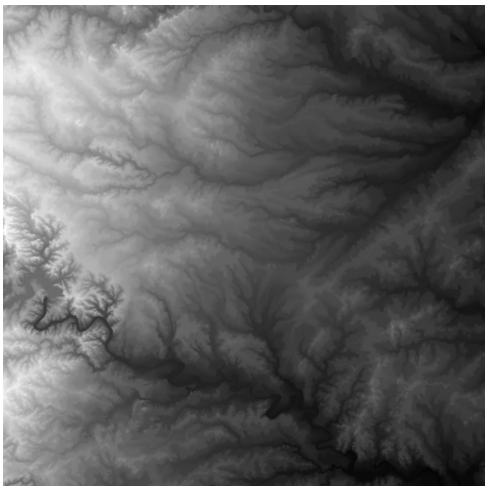
Inspect the difference between the different outputs. To do this it will help to turn off the Feature Type Fanout and manually give each output a different file name.

Name	Size	Type
BoundedCast.jpg	65 KB	JPG File
Cast.jpg	117 KB	JPG File
ScaleValues.jpg	73 KB	JPG File
ScaleType.jpg	18 KB	JPG File

Can you tell which output is from which conversion option?







It really helps to know the minimum and maximum values for the source data, and the minimum and maximum values in an 8-bit image, to understand what is going on.

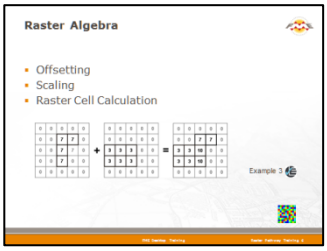
Why do the images look different with different options?
Which conversion option do you think is best? Which is worst?

7) Change Formats
Instead of writing to JPEG, try some different formats, for example:

- GIF
- TIFF
- PNG
- BMP

Do any of these have better interpretation support than JPEG?
If so, what makes them better?

Raster Algebra



Raster Algebra is the ability to carry out calculations and processing on individual raster cells.

Raster algebra is a set of techniques that carry out calculations and processing on individual cells.

There are a variety of raster algebra processes available in FME, including:

- Offsetting raster values (i.e. adding a fixed amount to each cell value)
- Scaling raster values (i.e. multiplying each cell value by a fixed amount)
- Evaluating values based on multiple raster features, for example;
  - Adding two or more raster cells together
  - Multiplying two or more raster cells together
  - Subtracting one raster cell from another

0	0	0	0	0
0	0	7	7	0
0	0	7	7	0
0	0	7	0	0
0	0	0	0	0

+

0	0	0	0	0
0	0	0	0	0
3	3	3	0	0
3	3	3	0	0
0	0	0	0	0

=

0	0	0	0	0
0	0	7	7	0
3	3	10	0	0
3	3	10	0	0
0	0	0	0	0

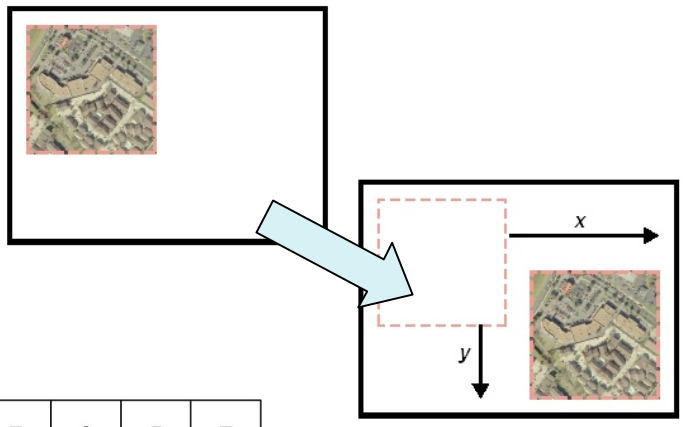
There are multiple uses for raster algebra too, including;

- Converting units (for example, from feet to metres)
- Summing a series of values recorded over time (for example rainfall)
- Best location analyses (including weighting of different factors)

Offsetting

Offsetting data can mean simply moving features by a certain specified distance along the X or Y axis.

However, with a raster dataset, an offset on the Z axis will cause the offset value to be added to the value stored in each cell.



3	4	1	3
1	2	4	2
2	3	3	4
1	4	1	2

7	8	5	7
5	6	8	6
6	7	7	8
5	8	5	6

Here a raster feature is offset by 4 units on the Z axis.

Note how the output values are all increased by 4.

Offsetting in FME Workbench is carried out using the *Offsetter* transformer. The offset value refers to whatever working units are being used in the dataset. For example, if the data is in feet, the offset will be applied in feet.

Scaling

Scaling is usually the act of multiplying each coordinate value by a specified factor, in effect causing a proportional increase or decrease in the size of each feature.

With a raster dataset, a Z scale factor will cause the values stored in each cell to be scaled by the specified proportion.

3	4	1	3
1	2	4	2
2	3	3	4
1	4	1	2

6	8	2	6
2	4	8	4
4	6	6	8
2	8	2	4

Here a raster feature is scaled by a factor of 2 on the Z axis.

Note how the output values are all twice the original.

Scaling in FME Workbench is carried out using the *Scaler* transformer. The scale values refer to scale factors and are unrelated to the current working units.

Raster Cell Calculations
Besides offsetting and scaling, cell calculations can also be carried out using multiple cells as the operands in an equation. Various operations could be carried out, including addition, subtraction, and multiplication.

For example, here two raster features are being added together. The result is the sum of overlapping cells.

0	0	0	0	0
0	0	7	7	0
0	0	7	7	0
0	0	7	0	0
0	0	0	0	0

+

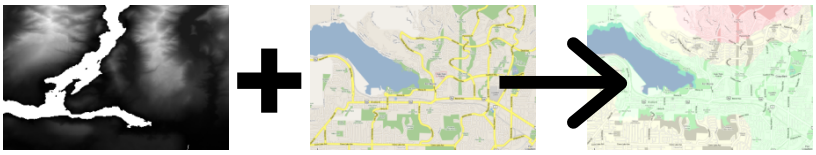
0	0	0	0	0
0	0	0	0	0
3	3	3	0	0
3	3	3	0	0
0	0	0	0	0

=

0	0	0	0	0
0	0	7	7	0
3	3	10	0	0
3	3	10	0	0
0	0	0	0	0

Interestingly, the cells being used can exist in two different raster features, or in two different bands on the same feature.

Uses of this technique include processing color/brightness in a raster image:



Or to combine raster datasets of different types together:

Cell Calculations in FME
Raster cell calculations in FME are carried out using the transformers *RasterExpressionEvaluator* and *RasterCellValueCalculator*.

The *RasterExpressionEvaluator* transformer is designed to operate on multiple bands within a single raster feature, or selected bands from multiple features.

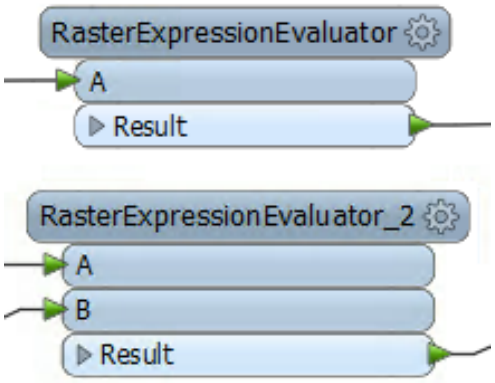
The *RasterCellValueCalculator* is designed to operate on the same band on multiple features.

When using multiple raster features each cell in the first raster must be paired with a cell in the second raster; therefore a condition of raster algebra in FME is that multiple raster features used in cell calculations must have the same number of rows and columns.

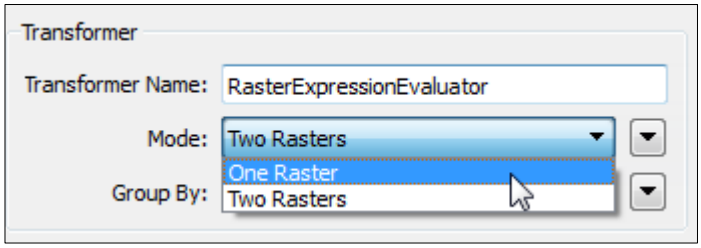
Also, in some operations, multiple raster features must possess the same number of bands.

RasterExpressionEvaluator

The *RasterExpressionEvaluator* transformer can have a single input port for one raster feature (A), or dual ports for Raster features A and B.



This capability is controlled inside the parameters dialog with a parameter called Mode:



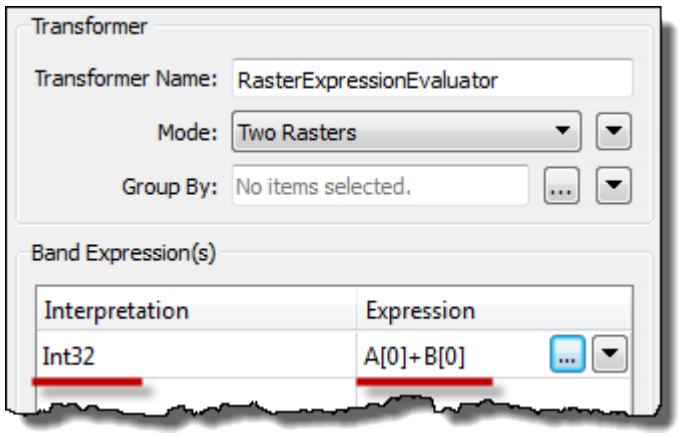
With only one input port, the transformer is used for creating complex expressions that operate on multiple bands within a single raster feature; for example:

Result = Raster A Band 0 + Raster A Band 1

In that scenario there would be one output feature per input feature. However, with two input ports the transformer can calculate values on selected bands from multiple incoming raster features; for example:

Result = Raster A Band 1 + Raster B Band 0

In that case there would be a single A feature, that can be processed against one or more B features. There would be one output feature for each pair of A/B inputs.



The Interpretation field tells us what interpretation is to be given to the output, by the calculation carried out by the Expression field.

Here two incoming raster features (single band, numeric) are being processed to create a new raster feature of interpretation INT32.

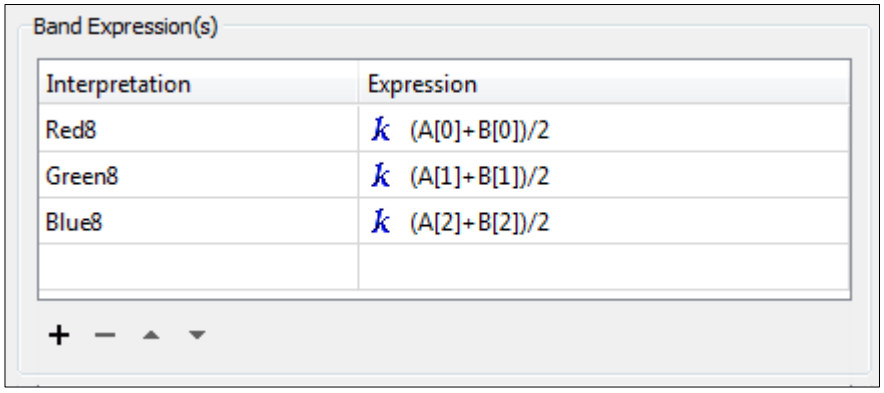
The expression is: A[0] + B[0]

The cell values in the output will be the sum of the matching cells in the two raster features.

For FME2013 the parameters dialog for this transformer has undergone revision to make it easier to use. In particular, it can be set to an attribute or parameter value, or clicking the [...] button opens up an editor specific to raster calculations.


NEW

In this example the input is two color raster images of interpretation RGB24. They are being processed to create another RGB24 with three 8-bit bands (Red8; Green8; Blue8).



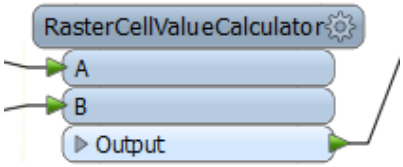
Each expression produces a new band in the output data. The expressions are:

(A[0] + B[0])/2;
(A[1] + B[1])/2;
(A[2] + B[2])/2

…which makes the output cell values an average of the incoming cell values.

RasterCellValueCalculator

The *RasterCellValueCalculator* transformer has a fixed set of two input ports for Raster features A and B.

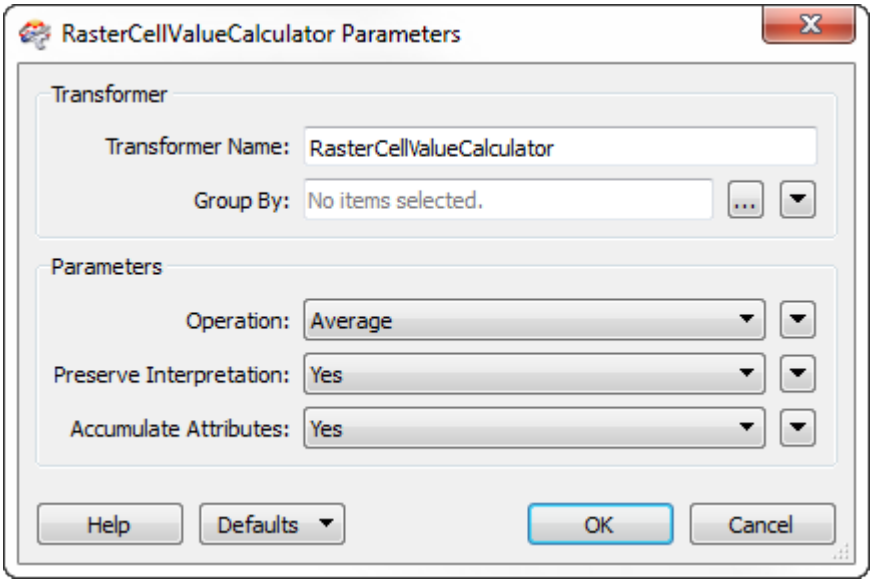


It is used when it's necessary to carry out calculations on the paired raster bands of two input raster features; for example:

Result 1 = Raster A Band 0 + Raster B Band 0
Result 2 = Raster A Band 1 + Raster B Band 1
Result 3 = Raster A Band 2 + Raster B Band 2

There would be one output per pair; i.e. in the above example, one output feature per result.

This transformer is simpler to use (than the *RasterExpressionEvaluator*) but therefore supports only a simpler set of expressions. Rather than user-defined expressions, it has a fixed set of operators.



In this case the user has set the 'average' operation.

The result will be a raster feature, per input pair, that is an average of equivalent cells.

The output raster feature will have the same number of bands as the incoming features.

Notice that the operations are a pre-defined set, selected through a drop-down list. The available operations are:

- Addition (+)
- Subtraction (–)
- Multiplication (*)
- Division (/)
- Minimum
- Maximum
- Average

Say, for example, that a user had a series of monthly rainfall totals in raster form.

They could use this transformer to add the monthly totals (to find an annual total), find the month with the minimum rainfall, find the month with the most rainfall, or find the average rainfall for each cell.

Example 3: Raster Algebra	
Scenario	FME user; City of Interopolis, Planning Department
Data	Zoning (GML), Pools (Idrisi), City Parks (MapInfo TAB)
Overall Goal	Heat map of most livable locations
Demonstrates	Raster Algebra
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Raster3a-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Raster3b-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Raster3c-Complete.fmw

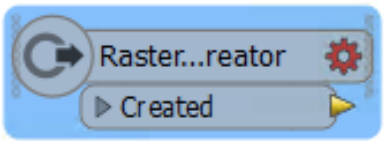
The overall goal here is to identify the "most livable" locations in the city of Interopolis. This will be based on factors such as closeness to parks, pools, and shopping.

One way to do this is to rasterize the available vector data, and use raster algebra.

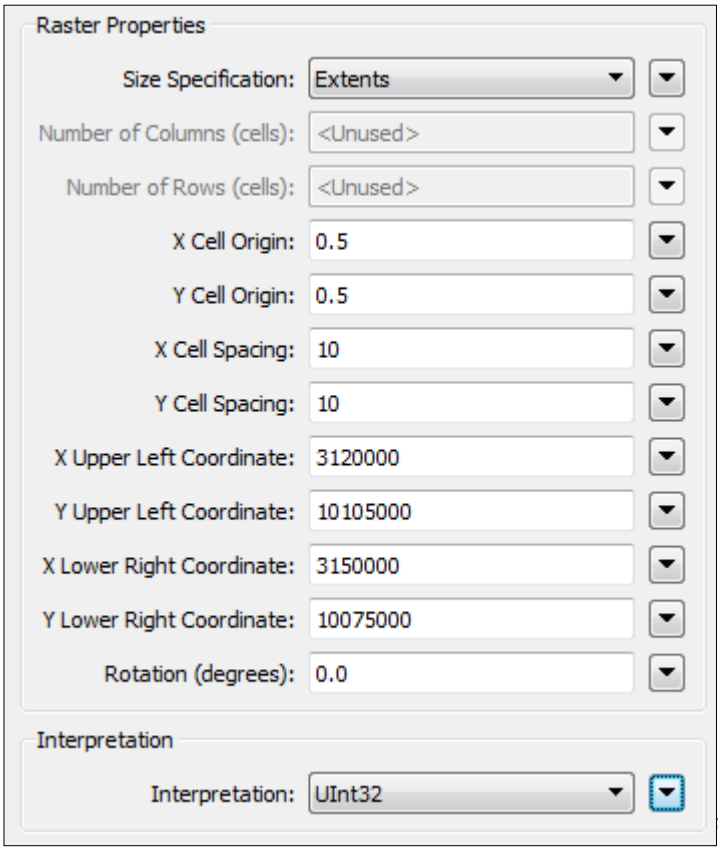
1) Start Workbench
Start Workbench. Start with an empty workspace.

2) Add RasterNumericCreator
The easiest way to rasterize vector data is to overlay it onto an existing raster feature, and the easiest way to obtain such a raster feature is to create it manually.

Place a *RasterNumericCreator* transformer into the workspace.



Open the properties dialog and enter values for the parameters as shown:



The advantage of using this transformer (instead of, say, the NumericRasterizer) is that we have more control over the exact size, shape, and resolution.

Size Specification	Extents
X Cell Spacing	10
Y Cell Spacing	10
X Upper Left	3120000
Y Upper Left	10105000
X Lower Right	3150000
Y Lower Right	10075000
Interpretation	UInt32

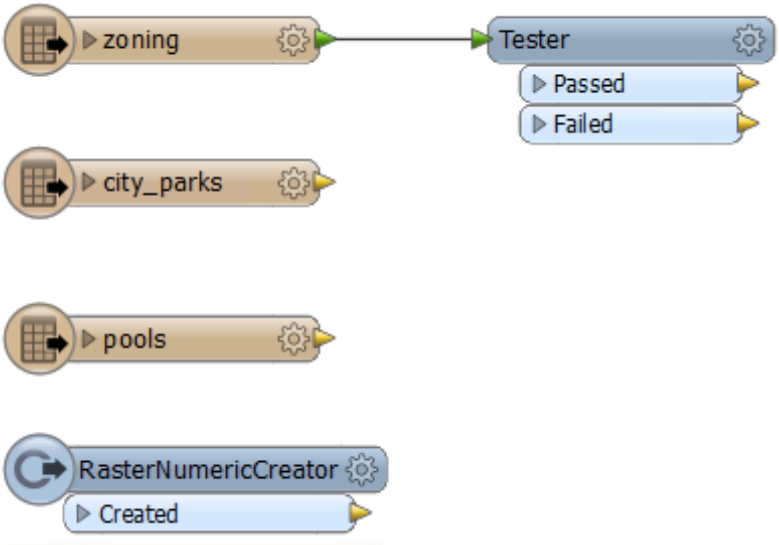
Notice that the extents are upper-left to lower-right (rather than lower-left to upper-right). Cell spacing will control the precision of the results. A lower value would be more precise, but it would take longer to run the translation.

The Interpretation, UInt32, means this is a numeric raster.

3) Add Readers
Add Readers for the following three vector datasets:

Reader Format	GML (Geography Markup Language)
Reader Dataset	*C:\FMEData\Data\Zones\zoning.gml*
Reader Format	MapInfo TAB (MFAL)
Reader Dataset	*C:\FMEData\Data\Parks\city_parks.tab*
Reader Format	IDRISI Vector Format
Reader Dataset	*C:\FMEData\Data\Pools\pools.vct*

The “Feature Collection” Feature Type for the GML (zoning) dataset is not required and can be safely ignored or even deleted.



4) Add Tester
In the case of the zoning data, this is being used to identify shopping areas.

Therefore add a *Tester* transformer and set it up to filter out data that is not zoned type “CS”.

The output port (PASSED/FAILED) that will be connected (next step) will depend on the test you created, and whether you chose to search for ‘CS’ features or features that are not ‘CS’.

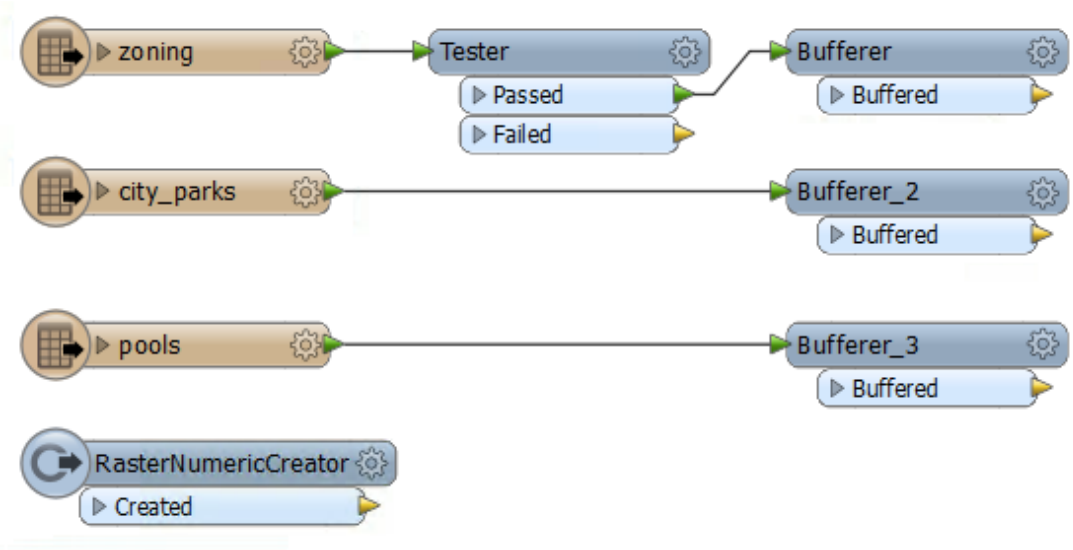
You can always connect up Inspectors

and run the workspace at this point, to test what the data looks like.

5) Add Bufferers
In all cases, we want to identify areas within 2000 feet (approximately ½ mile) of each vector feature. This can be done by buffering these features.

Add three *Bufferer* transformers. Connect one to each of the source Feature Types (or to the *Tester* output in the case of the zoning data).

Set up the parameters to create 2000 foot zones around each feature.

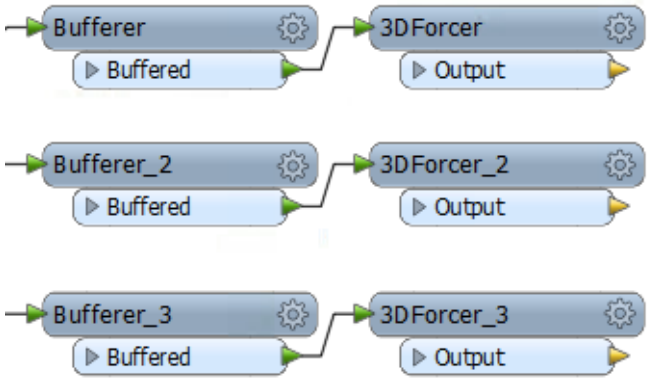


6) Add 3DForcers
Because the raster data is numeric (rather than a color image) the value of each cell as overlaid by the vector data will come from the Z coordinate of the vector feature. At the moment they are 2D features, but this can be resolved using a 3DForcer transformer.

Place three *3DForcer* transformers, each connected to one of the *Bufferers*.

Set the *3DForcer* elevation values as follows, to reflect the relative value of each data type:

Shops (Zoning)	6
Pools	3
Parks	10



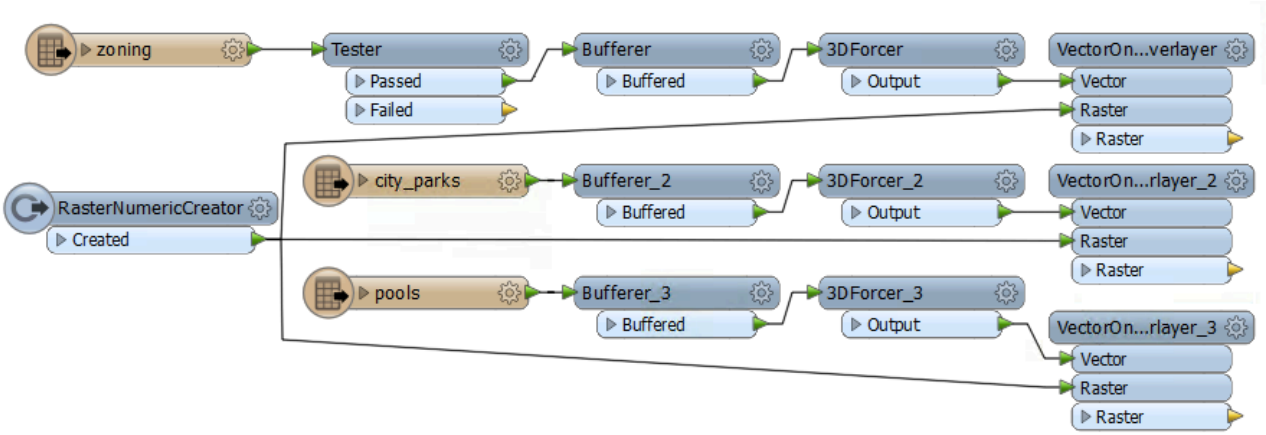
7) Add VectorOnRasterOverlayers
Now the vector data can be overlaid onto the raster created earlier. Rather than overlay it all onto a single raster, we want to create three different raster features. That way we can use the RasterExpressionEvaluator to make calculations between the three.

Place three *VectorOnRasterOverlayer* transformers.

Connect the output port from each *3DForcer* to the VECTOR port of one of the *VectorOnRasterOverlayers*.

Connect the CREATED port from the *RasterNumericCreator*, to each of the three *VectorOnRasterOverlayer*:RASTER ports.

If you've followed best practice, and not crossed any streams, your workspace might look something like this:

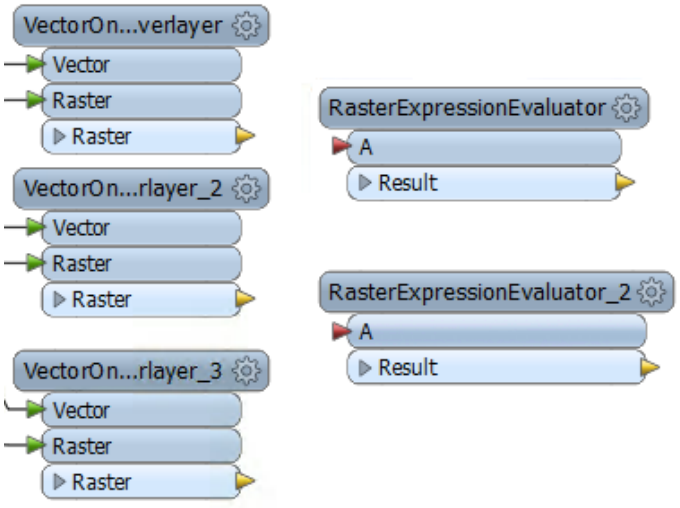


The default parameter values will be fine for these transformers.

8) Add RasterExpressionEvaluator
To apply some raster algebra to this data we'll use the *RasterExpressionEvaluator* transformer.

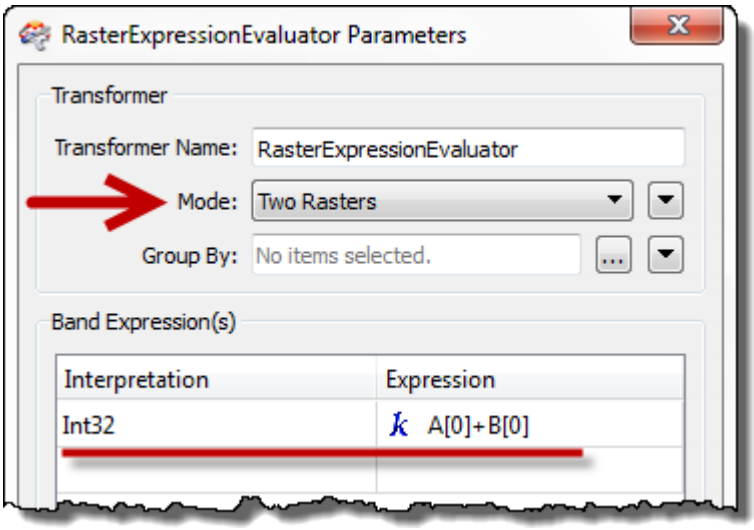
Place two *RasterExpressionEvaluator* transformers. One will be used to process the Zoning and Parks data, and the second will be used to process the Pools data against the results of the first.

Because we'll use two input ports we'll need to set up the transformer parameters before we connect the transformers.



9) Set Parameters
In both *RasterExpressionEvaluator*s in turn, open the parameters dialog. Set the Mode parameter to “Two Rasters” and set the following expression:

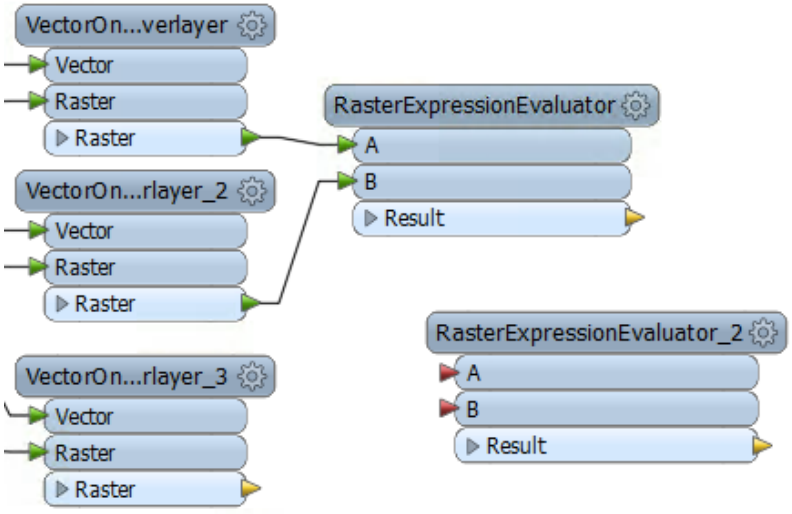
Band Interpretation INT32
Band Expression A[0] + B[0]

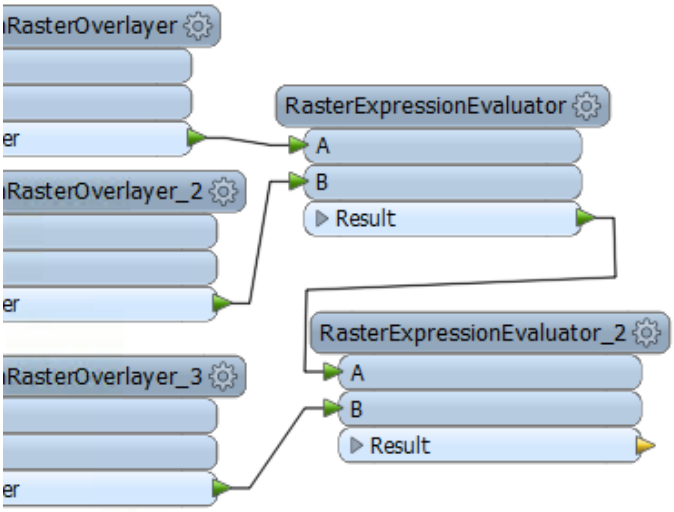


10) Connect Transformers
Both transformers should now have two input ports (A and B) and can be connected up.

Connect the output from the first *VectorOnRasterOverlayer* transformer to the first *RasterExpressionEvaluator* A port.

Connect the output from the second *VectorOnRasterOverlayer* transformer to the first *RasterExpressionEvaluator* B port.





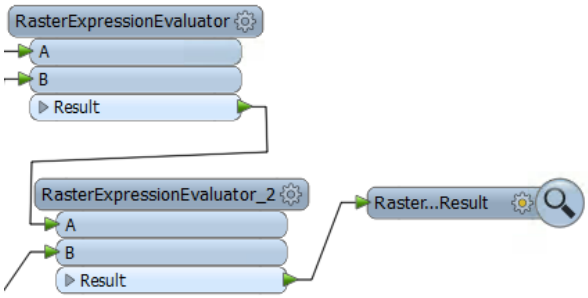
Now connect the output from the first *RasterExpressionEvaluator* to the second RasterExpressionEvaluator A port.

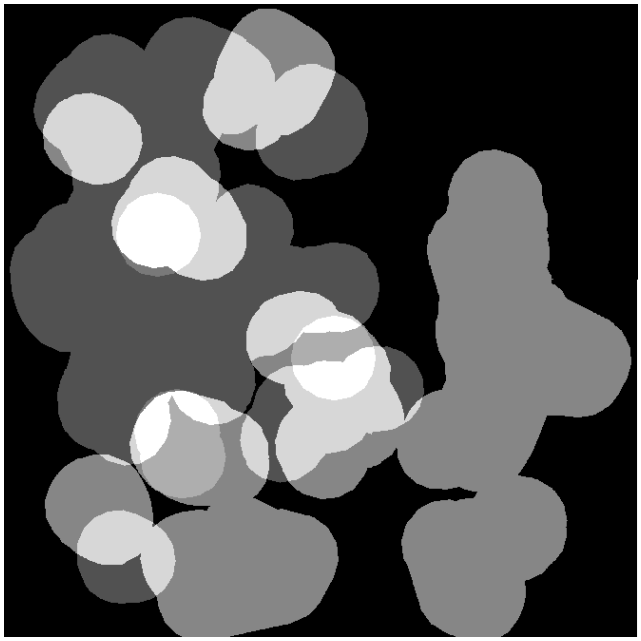
Connect the output from the third *VectorOnRasterOverlayer* to its B port.

11) Add Inspector

Finally, add an *Inspector* to the output (RESULT) port of the second *RasterExpressionEvaluator* and run the workspace.

The workspace should take about five seconds to run, and produce an output that looks like this:





Query some cells. Notice how the white highlights are 'hot-spots' where parks, shopping, and pools all come together. The highest value should be 19, which is the sum of the three *3DForcer* elevations.

Less intense colors illustrate different combinations of the three vector data types.

Advanced Tasks
If you have time, try the following challenges:

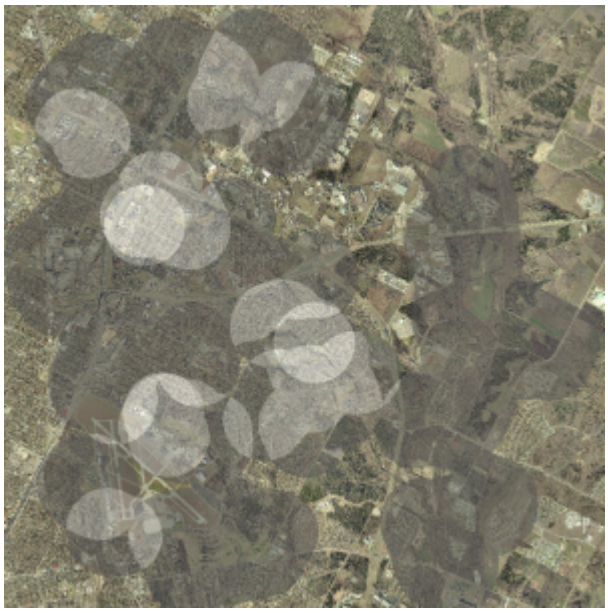
1) Set up published parameters so that a user could choose their own rankings (elevation) for the three types of vector data. The output will always look similar – and the highest ranking areas will always be those combining the three data types – but the secondary areas will depend on the numbers entered.

2) Write the data to an image format. Remember to reinterpret the data if necessary.

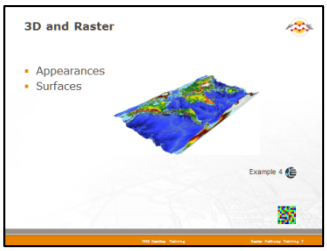
3) As a really advanced task, can you blend the results of this translation into the InteropolisCentre PNG image, using the techniques from Example 1?

As they are two raster datasets you would use the RasterMosaicker transformer (instead of the VectorOnRasterOverlayer).

The hard part will be getting the bands and interpretation to line up!



3D and Raster

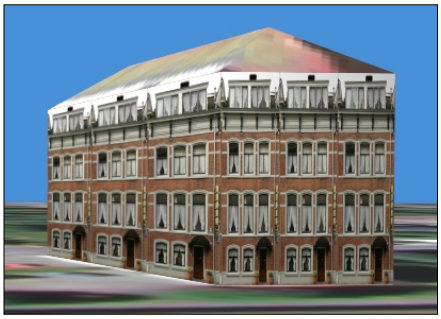


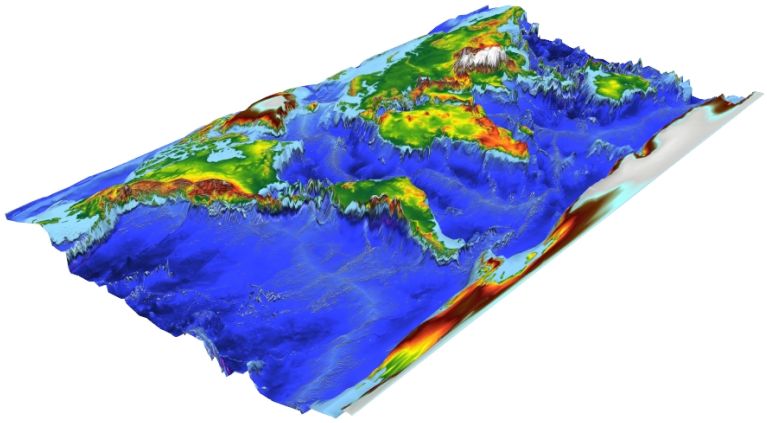
Raster data is nicely integrated into 3D tools in Workbench

Raster datasets can play a part in 3D data transformation, mainly in surface creation or as an appearance on a 3D object.

Appearances
Appearances are the use of raster features as textures on three dimensional elements such as surfaces, faces, and meshes.

Here a set of raster images are used on the different faces of a 3D element that represents a building.

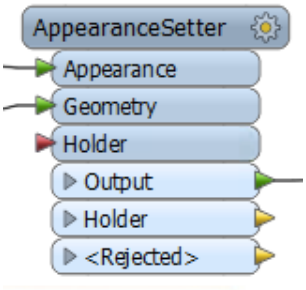




This example is a raster image of a world map, used as an appearance on a 3D DEM surface.

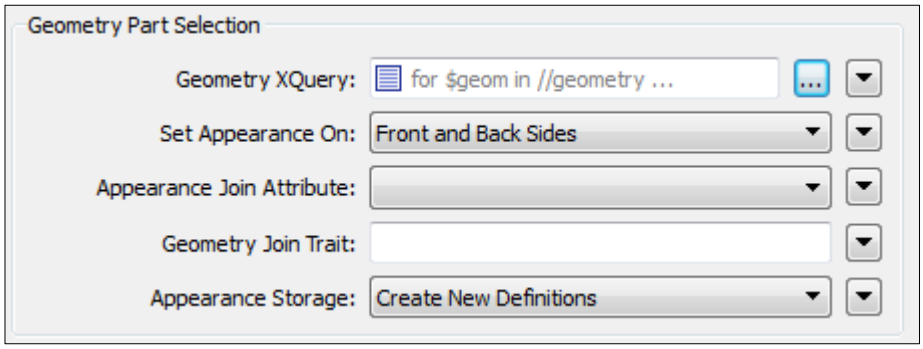
Creating Appearances in FME

The *AppearanceSetter* transformer is used to attach a raster feature as an appearance to a face or surface.



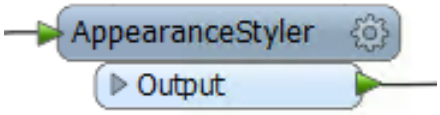
A parameter allows appearances to be added to both the front and back of a surface, so that FME can put textures on a building both inside and out.

Because a single 3D geometry can be made up of a number of different parts, XQuery and Join Attributes are provided to allow the workspace author control over which of the various parts the appearance is applied to.



The *AppearanceStyler* transformer adds style information to a raster feature that is applied when the raster is added as an appearance.

In particular it can be used to:



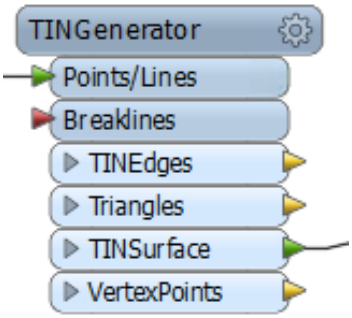
- Set a scale, offset, or shearing on an appearance
- Rotate an appearance
- Set appearance transparency
- Set different light types (Diffuse, Ambient, Specular, Emissive)

Two other appearance-related transformers are the *AppearanceRemover* and the *AppearanceExtractor*. The *AppearanceRemover* is for taking appearances off a surface feature, while the *AppearanceExtractor* is for extracting the appearances of a surface as raster features.

Surfaces
A surface is a three-dimensional continuous field. Since raster data is also a form of continuous field (rather than discrete values) it's relatively simple to transform a raster feature into a surface.

In that scenario, Z values for the surface are taken from the cell values of the raster data.

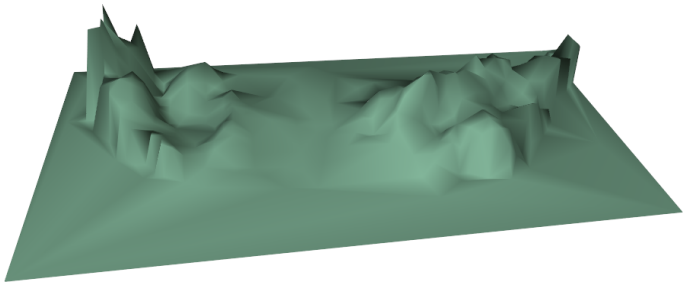
Creating Surfaces in FME
Either of the *TINGenerator* or *SurfaceModeller* transformers is capable of converting a raster dataset to a surface. The *TINGenerator* is the simpler of the two.



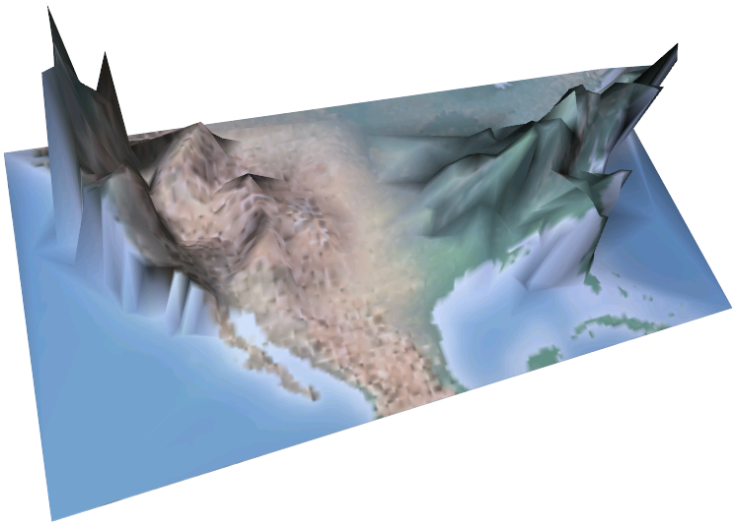
Raster features enter these transformers through the POINTS input port. There are multiple outputs possible, but 3D surfaces emerge from the TIN_SURFACE output port.

The incoming features must be a raster feature containing only a single band, so color images are not a form of raster that could be used in this scenario. Numeric data is more likely.

Here a very simple raster rainfall dataset for the US is converted to a 3D surface:



Because a surface like this can look rather drab, it's common to also add a raster dataset (such as a relief map) as an appearance:



Of course it's necessary to be able to write the data to a format that supports textured surfaces.

The simplest format is perhaps Adobe 3D PDF. Others include:

- Esri Geodatabase
- 3ds
- OBJ
- Collada
- CityGML
- SketchUp
- KML

Example 4: Raster 3D DEM Model	
Scenario	FME user; City of Interopolis, Planning Department
Data	Interopolis Centre (PNG), DEM (CDED)
Overall Goal	3D raster DEM surface with draped image
Demonstrates	Raster Surfaces, Raster Appearances
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Raster4-Complete.fmw

The goal here is to create a three-dimensional surface model of an area of land, starting out with a raster DEM and a raster appearance.

1) Start Workbench
Start Workbench. Create a translation from the CDED elevation dataset to 3D PDF format.

Reader Format Canadian Digital Elevation Data (CDED)
Reader Dataset *C:\FMEData\Data\ElevationModel\RasterDEM-250K.dem*

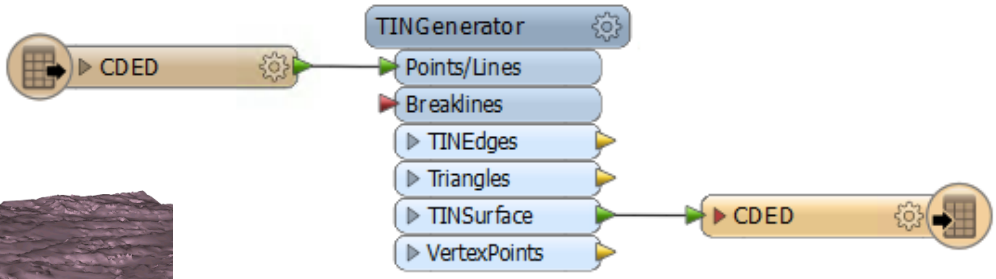
Writer Format Adobe 3D PDF
Writer Dataset *C:\FMEData\Output\DemoOutput\Surface.pdf*

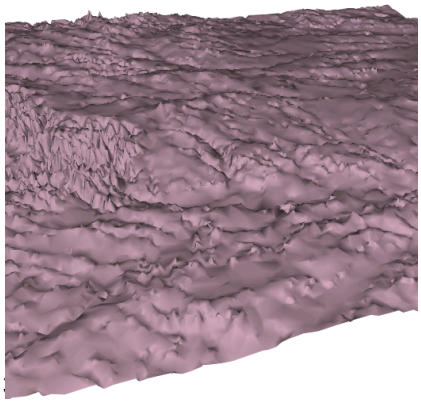
What output do you get if the workspace is run at this point? Does the log window reveal why?

2) Add TINGenerator
One way to get a proper output is to turn the raster CDED into a surface.
Add a *TINGenerator* transformer.

Connect the source CDED Feature Type to the POINTS/LINES input port.
Connect the TIN_SURFACE output port to the PDF Writer Feature Type.

Open the *TINGenerator* parameters dialog and enter a tolerance of 5.0





Run the workspace. It should take about 30 seconds to complete, and the output will look like this in Adobe Reader:

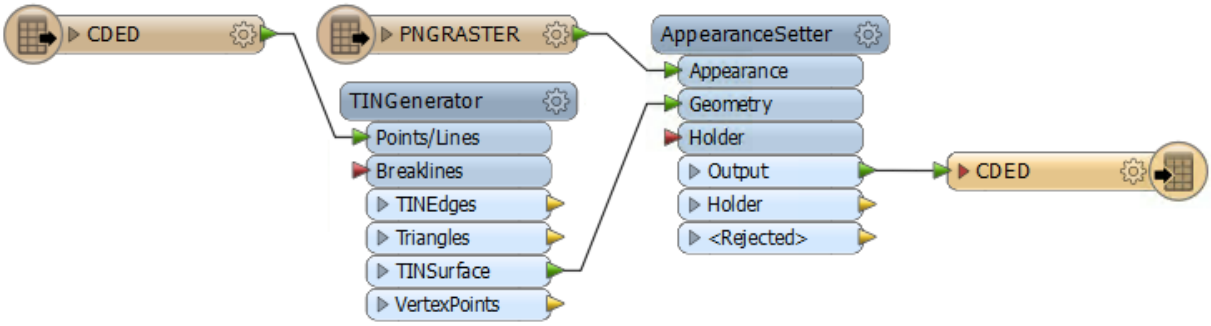
3) Add PNG Reader
Now an appearance can be added to this surface model. Back in Workbench, add a Reader to read a PNG format file:

Reader Format PNG (Portable Network Graphics)
Reader Dataset *C:\FMEData\Data\Raster\InteropolisCentre.png*

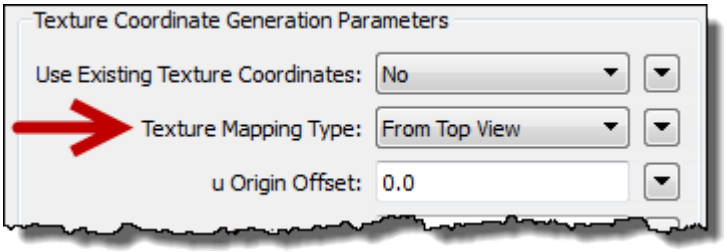
4) Add AppearanceSetter
Add an *AppearanceSetter* transformer.

The PNG Reader Feature Type should connect to the APPEARANCE input port.
The *TINGenerator*:TIN_SURFACE should be connected to the GEOMETRY input port.
The AppearanceSetter OUTPUT port should be connected to the PDF Writer Feature Type.

Delete the existing connection between TIN_SURFACE and the PDF Feature Type.
The workspace will now look like this:

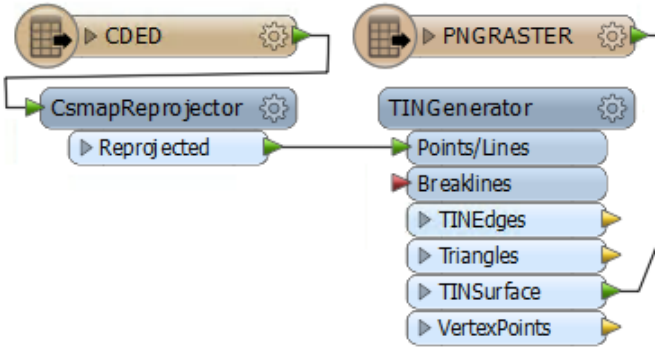


5) Set Parameters
Set the *AppearanceSetter* 'Texture Mapping Type' parameter to 'From Top View'.



6) Add Reprojector
The PNG and CDED datasets occupy different coordinate spaces, and that would be a problem.

Add a *CsmapReprojector* transformer to reproject the CDED data before it reaches the *TINGenerator*. Reproject it from its source coordinate system to TX83-CF.



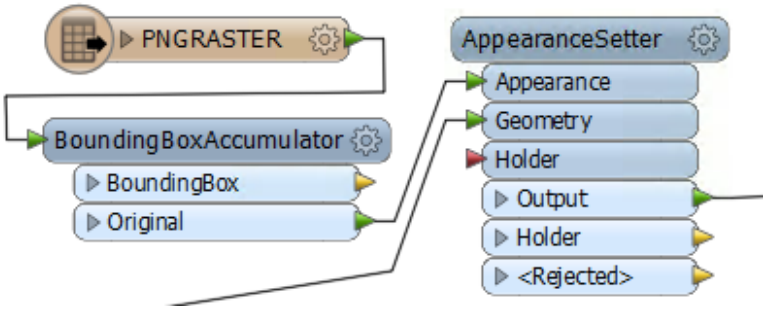
But DON’T run the workspace just yet!

7) Add BoundingBoxAccumulator
To optimize the process, we should make sure the surface only covers the same area as the raster image – otherwise the system resources are being wasted.

This can be achieved by finding the extents of the PNG data, and clipping the CDED to it.

So, add a *BoundingBoxAccumulator* transformer to the incoming PNG data.

The ORIGINAL output port should be the one connected to the *AppearanceAdder*.



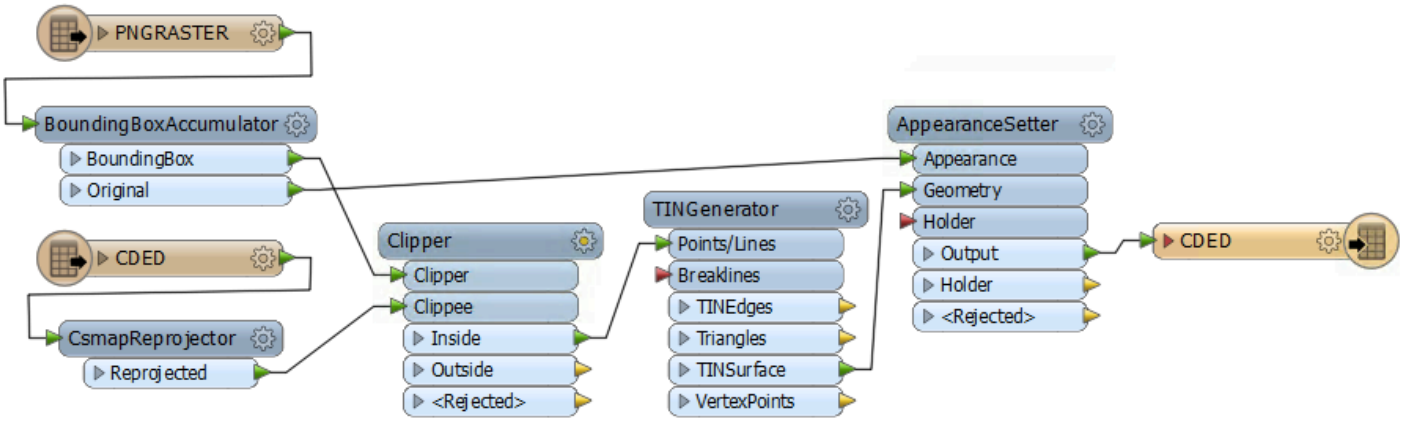
8) Add Clipper
Now add a *Clipper* transformer.

Makes these connections:

From	**To**
BoundingBoxAccumulator:BOUNDING_BOX.....	*Clipper*:CLIPPER
Reprojector:REPROJECTED.....	*Clipper*:CLIPPEE
Clipper: INSIDE.....	*TINGenerator*:POINTS/LINES

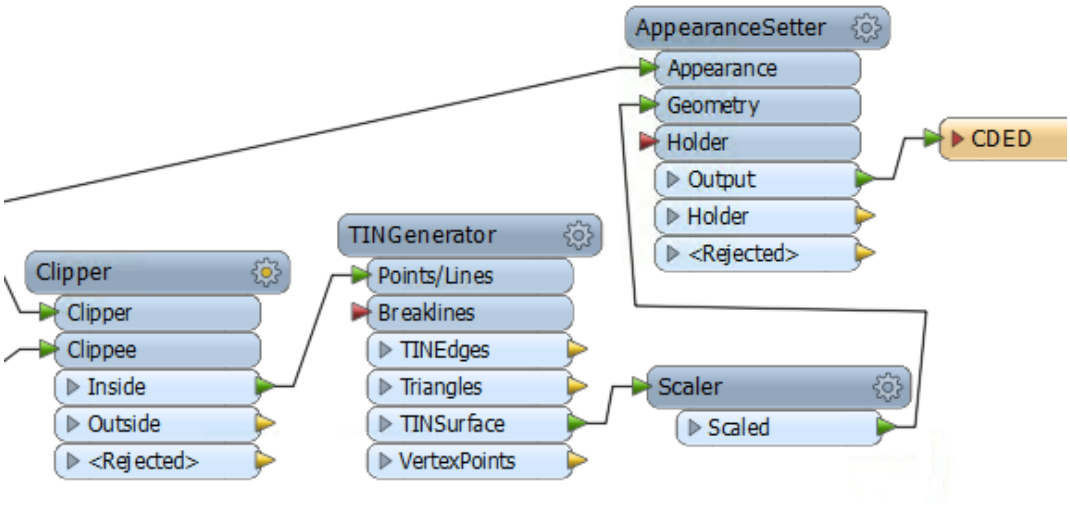
Delete the original connection from CsmapReprojector to TINGenerator.
With a little tidying, the workspace will now look like this:

Now you can save the workspace and run it. It will take about 30 seconds to run. Make sure you don't have the PDF file open already in Adobe Reader, as FME will not be able to overwrite it.

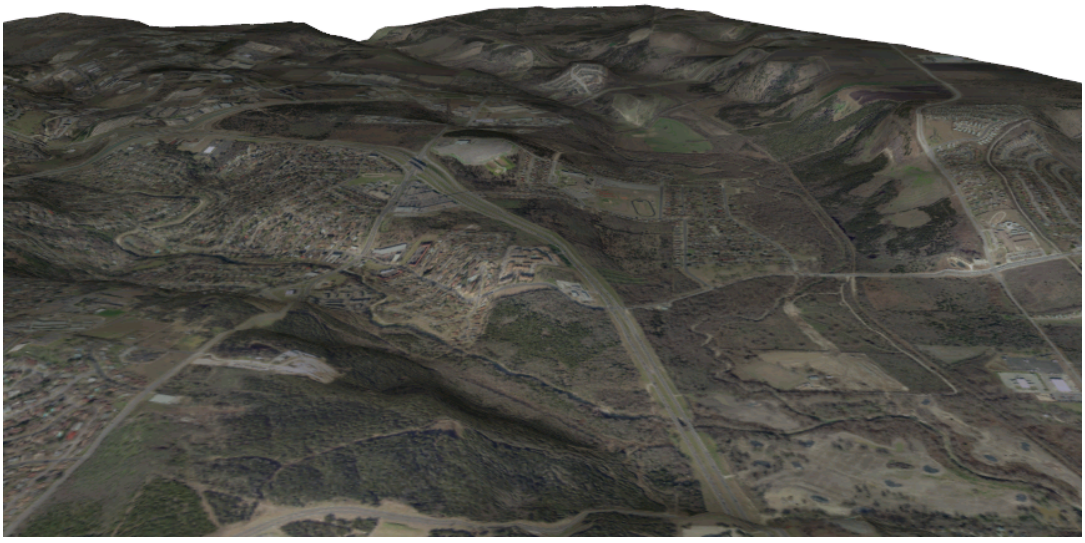


9) Add Scaler
Inspect the output PDF in Adobe Reader. It is correct but, because of the geography of the area, a little flat.

Add a *Scaler* transformer after the TINGenerator, to scale the Z values of the raster DEM. A good scale factor to use is 20.

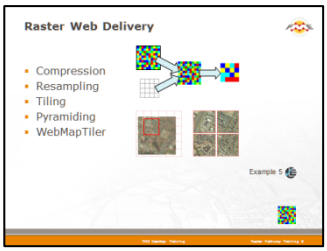


Inspect the output. It will now look like this.



If you put the Scaler before the TINGenerator, then the whole workspace could take longer to run. That's because you've exaggerated the Z scale by a factor of 20, but the TINGenerator's Surface Tolerance is still the same. You should really also scale up the tolerance parameter in that case.

Raster Web Delivery



Preparing raster datasets for web delivery is its own particular art (and science)

Raster datasets need particular preparation and processing when being used for web delivery. In general, limited network connections are a potential bottleneck, and so transfer speed can be maximized by reducing the volume of data being delivered.

With raster data there are a number of techniques for reducing data volumes:

- Compression
- Resampling
- Tiling
- Pyramiding

Compression
Compression is a technique used to reduce the size of a raster dataset. It's the process of dropping information, hopefully without reducing the quality to any noticeable extent.

Another trade-off can be performance, as reading and writing compressed data is naturally slower than handling data in a raw form.

The types of compression available depend on specific format support. Some compression types are:

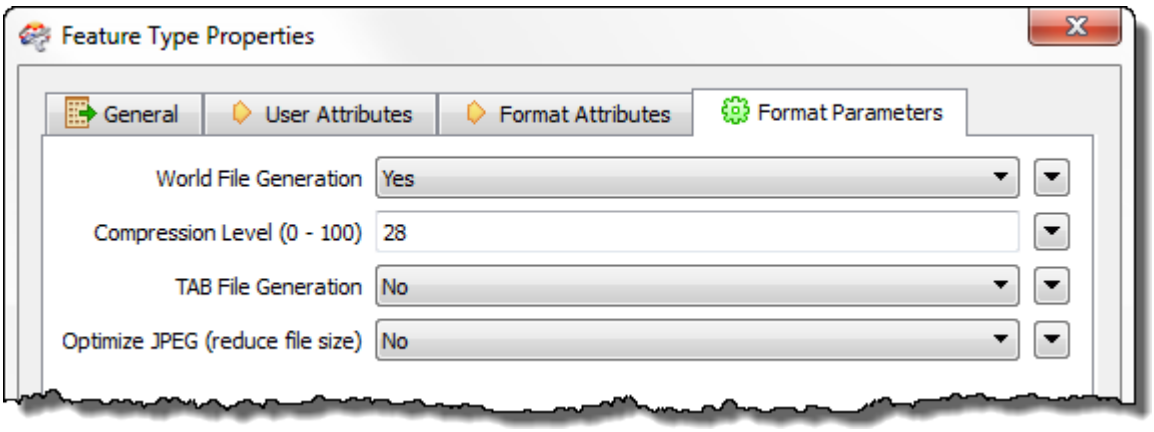
All types use different methods, which determine the amount of size reduction and the cost in terms of quality.

- DEFLATE
- JPEG-B
- JPEG-F
- LWZ
- CCIT-RLE
- LZ77

- GeoTIFF
- JPEG
- Jpeg2000
- Tiff
- Oracle Spatial GeoRaster
- Esri ArcSDE and Geodatabase
- PNG
- ERMapper compressed

Some FME-supported formats that support writing compressed raster data are:

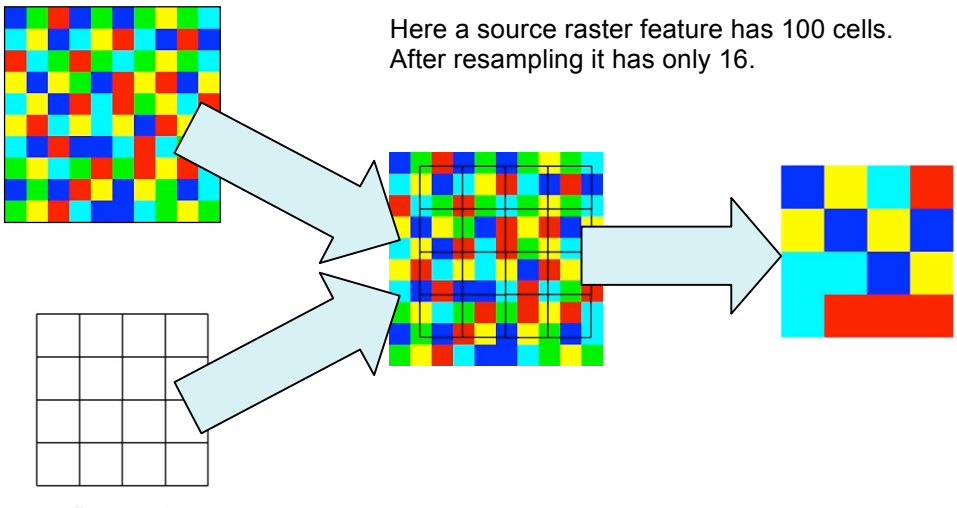
Compression is usually set by a parameter found at the feature-type level.
Here compression is being carried out using JPEG Feature Type parameters.



Resampling
As already noted, a raster dataset consists of a grid of cells, each of which has a discrete value.

Resampling is a process that changes raster resolution. A new grid of cells is generated and overlaid on the original dataset, with cell values for the new grid taken from the existing cells.

The new grid may consist of larger cells, in which case the new raster would be coarser than the original, and thereby be a smaller size. The trade-off is lessened precision, since cells with different values might be merged into a single cell with just one value.

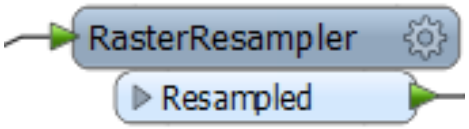


Here a source raster feature has 100 cells.
After resampling it has only 16.

Resampling Methods
Several different resampling methods exist to transfer values from the original grid to the new one.

- **Nearest Neighbor** sampling simply means that a new cell takes the value of whatever old cell is the closest; or – to be more precise – the new cell takes the value of whatever existing cell whose origin is closest to the new cell's origin. Importantly, only one cell provides a value; other surrounding cells have no bearing on the result. The above example shows Nearest Neighbor resampling.
- **Bilinear** sampling is when a new cell takes a number of surrounding cells and interpolates a new value according to the linear distance to each.
- **Bicubic** sampling is when a continuous surface is interpolated and new cell values extracted from that.
- **Average4 and Average16** sampling is when each new cell value is an average of the current 4 or 16 cells that surround it. These methods are best for use with DEM data.

Resampling Data in FME
Raster resampling in FME is carried out with the *RasterResampler* transformer.



The new size (after resampling) can be defined by either:

- The number of rows and columns
- The size of the new cells
- A percentage of the original raster resolution

Note that Compression and Resampling are two entirely different processes, and that Compression techniques are not always based around Resampling. ECW and JPEG2000, for example, use wavelet transformations for compressing datasets.

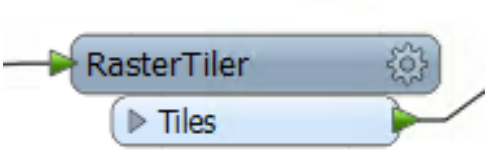
Tiling

Tiling is when source data is divided into any number of square or rectangular outputs, based on a specific user-defined tile size.

Because data delivered via the web can be individual tiles, rather than the entire dataset, the volume of data being transferred is significantly reduced. The trade-offs are artificial boundaries, plus a pause when a new tile needs to be downloaded.

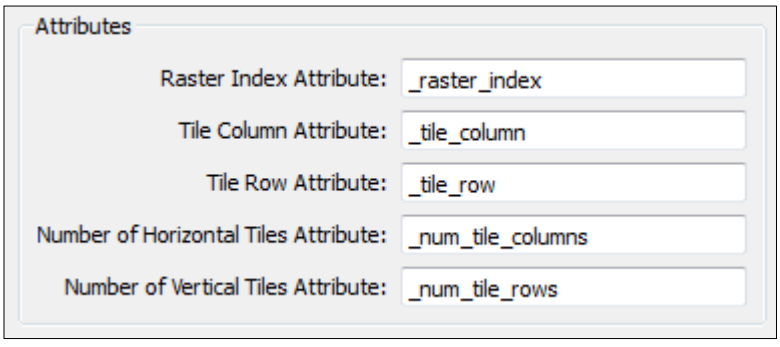
Tiling Data in FME

Tiling in FME is done with the *RasterTiler* transformer. Unlike the *Clipper*, this transformer needs no pre-defined boundaries.



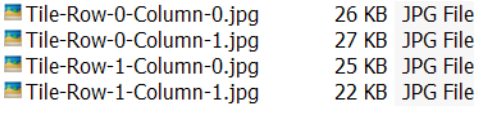
Tiling can be defined by either tile size or number of tiles. If a raster had 10 meter cells, 1000 columns and 1000 rows; with 5000m by 5000m specified as the tile size, one would get 4 tiles or 4 new raster features as output.

There is also an option to decide whether edge tiles should be truncated at the exact edge of the data, or whether they will be padded with 'nodata' to the full extent of the tile size.



The transformer adds a number of attributes to identify each raster tile, and these could be used in a Fanout to label each output dataset with the appropriate row and column ID.

Using such a Fanout, the files output from a *Tiler* transformer may be named like this:



Notice how the file size remains fairly constant for

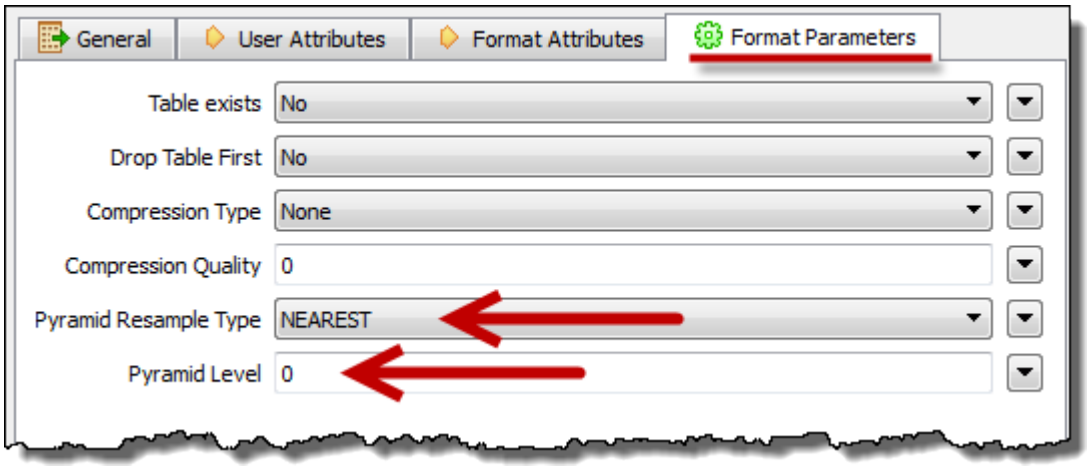
each different tile.

Pyramiding
Pyramids are lower resolution views of a dataset. When writing a raster dataset, several pyramids can be created at various lower resolutions to be used in the place of the original raster when only a snapshot or overview of the data is required.

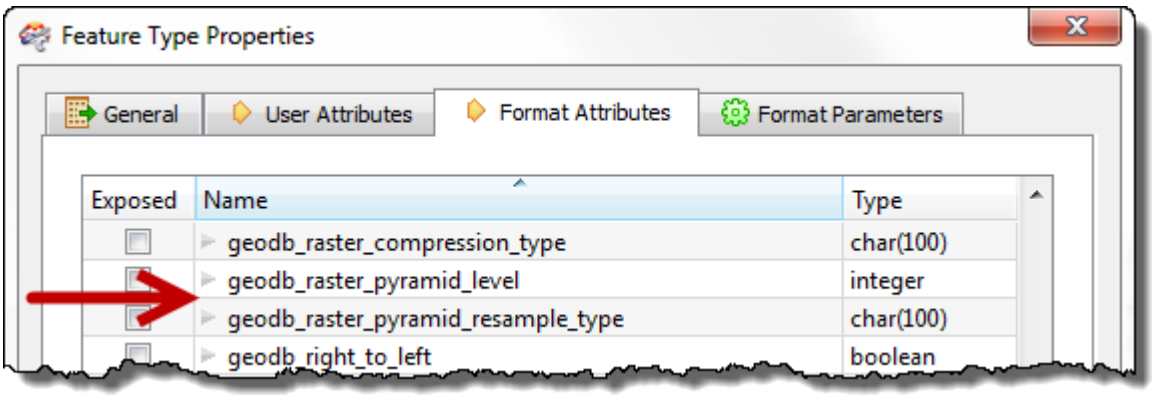
An example of when a pyramid is typically employed is when a raster viewer zooms out leaving a smaller raster with less detail. For performance reasons, the smaller raster is often rendered using a cached pyramid instead of resampling the image to a lower resolution at the time of the zoom out request.

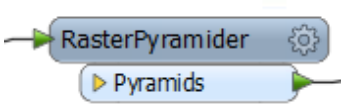
Pyramiding (by definition) re-samples raster at the same time, but users often also tile the data too, so that there are more tiles for the higher resolution images, and fewer tiles for the lower resolution ones, resulting in (more or less) even performance regardless of the view scale.

Pyramiding Data in FME
FME can create native pyramids for formats such as Oracle Spatial GeoRaster, and Esri Geodatabase Raster. This operation can be done with Feature Type parameters:



…or format attributes.

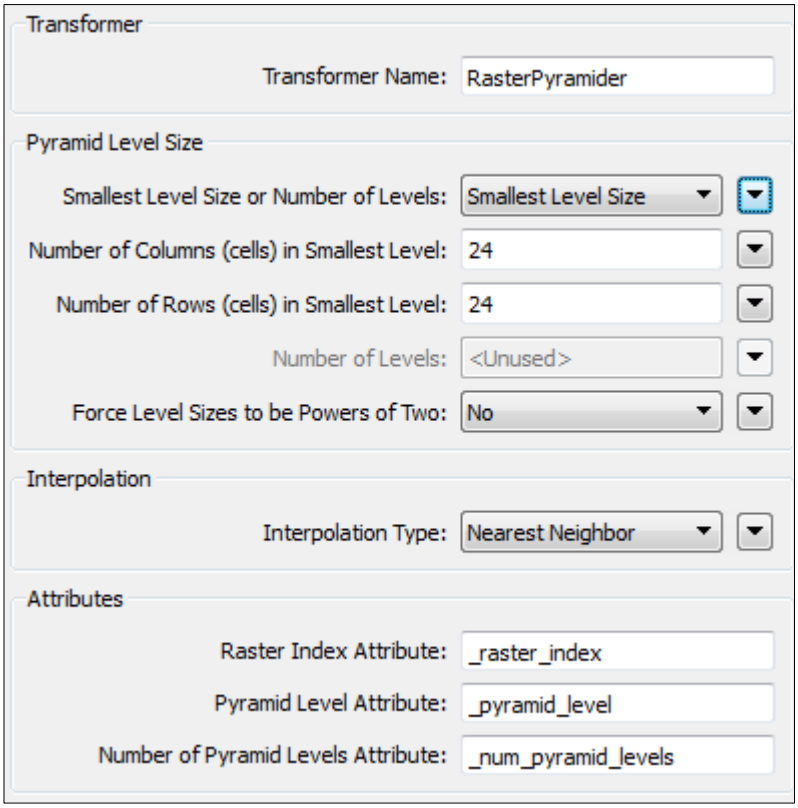




For general pyramiding there is a *RasterPyramider* transformer that will create a set of pyramids for any raster data.

With the *RasterPyramider*, pyramids can be defined by either the size of the smallest level, or the number of levels of pyramiding.

Again there are parameters for interpolation type, and for attributes to be added to the data.



Using a Fanout on the pyramid level attribute would result in a set of files like this:

Pyramid-Level-0.jpg	73 KB	JPG File
Pyramid-Level-1.jpg	40 KB	JPG File
Pyramid-Level-2.jpg	15 KB	JPG File
Pyramid-Level-3.jpg	6 KB	JPG File
Pyramid-Level-4.jpg	2 KB	JPG File
Pyramid-Level-5.jpg	1 KB	JPG File
Pyramid-Level-6.jpg	1 KB	JPG File

Notice how the file size decreases the higher the level (and the lower the resolution) of the pyramid.

WebMapTiler

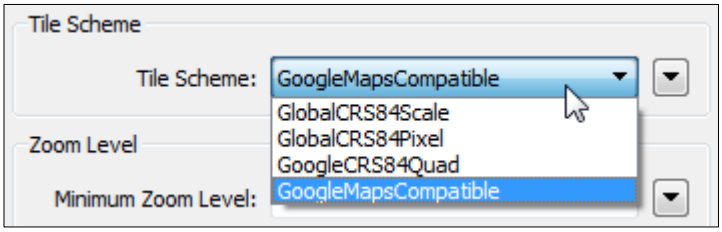
The *WebMapTiler* is a way to tile, pyramid, and resample data; all in one transformer. It's designed to render raster data into a series of pyramided tiles for use in web mapping tools such as Bing Maps, Google Maps, or a Web Map Tile Service (WMTS) system.

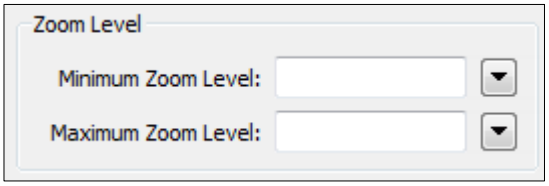
It's important to realize that all the operations carried out by this transformer relate to a predefined system of grids, not user-defined ones.

Parameters

The key parameters are the type of scheme and a minimum and maximum zoom level.

There are four available schemes. Google and Bing Maps both use the "GoogleMapsCompatible" scheme. The other schemes are for use in a WMTS mapping system.



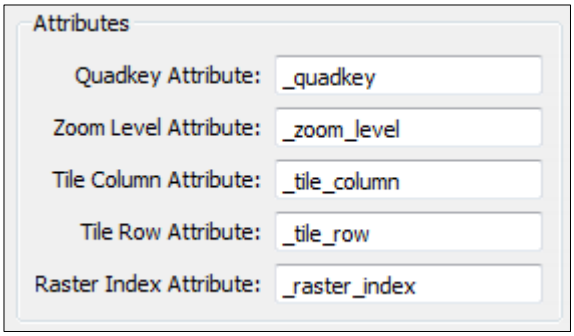


Zoom level numbers vary depending on the scheme used. Bing Maps (for example) allows values from 1 to 23, each of which is a predefined scale with its own map width, ground resolution, and map scale.

The zoom levels are directly related to the predefined sets of tiles for each zoom level; the *WebMapTiler* will automatically generate tiles that use these predefined boundaries, tile widths, scales, and image resolutions.

The attributes added to the data therefore return values that describe which tile the data falls in.

Bing Maps uses the Quadkey attribute to describe this information. Google uses a combination of Zoom Level, Tile Row, and Tile Column.

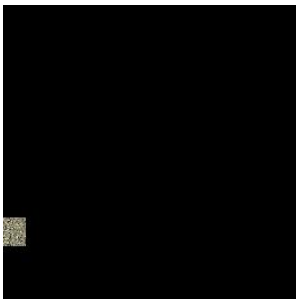


Input

Data entering a *WebMapTiler* is usually a single raster feature, although multiple features can be handled. To be sure of a correct results the data could be merged together with a mosaicking process before the data enters the *WebMapTiler*.

Nodata Areas

Because the tile boundaries are fixed, it is possible that the incoming raster data covers only a small part of a particular tile.



For that reason it's advisable to write the output data with an RGBA interpretation. That way the 'nodata' area surrounding the data will be transparent on output, and any maps underlying it will show up.

If the incoming raster features don't have an alpha band, a *RasterBandAdder* can be used to create one.

Be sure not to change the order of features as they are output by the WebMapTiler transformer. Doing this can severely compromise the performance of the Writer.

Output Formats

When the data is to be viewed in an internet browser, the output format of the raster features should be one that is compatible with this use. PNG format is a good choice because it is recognized by browsers, and also supports transparency.

Output

Using a Fanout, here is the same set of tiled data created for use in Google, and in Bing Maps:

Level-3-Row-1-Column-4.jpg	2 KB	JPG File	102.jpg	2 KB	JPG File
Level-4-Row-3-Column-9.jpg	2 KB	JPG File	1023.jpg	2 KB	JPG File
Level-5-Row-7-Column-18.jpg	2 KB	JPG File	10232.jpg	2 KB	JPG File
Level-6-Row-15-Column-36.jpg	2 KB	JPG File	102322.jpg	2 KB	JPG File
Level-6-Row-15-Column-37.jpg	2 KB	JPG File	102323.jpg	2 KB	JPG File
Level-7-Row-31-Column-73.jpg	2 KB	JPG File	1023223.jpg	2 KB	JPG File
Level-7-Row-31-Column-74.jpg	2 KB	JPG File	1023232.jpg	2 KB	JPG File

The data for use in Google is fanned out with a Level-Row-Column combination; the data for Bing Maps is fanned out by Quadkey

Both systems create a unique identifier for every possible tile. There should be no two files with the same values for Level-Row-Column combination, or for quadkey.

The quadkey value for Bing Maps is interesting because the length of the quadkey denotes the tile zoom level; i.e. quadkey 10232 contains 5 digits and so will be a level 5 tile. Also, the quadkey of a tile starts with the quadkey of its parent (the tile above it). So (above) tile 102 is the parent of tile 1023, which is the parent of 10232.

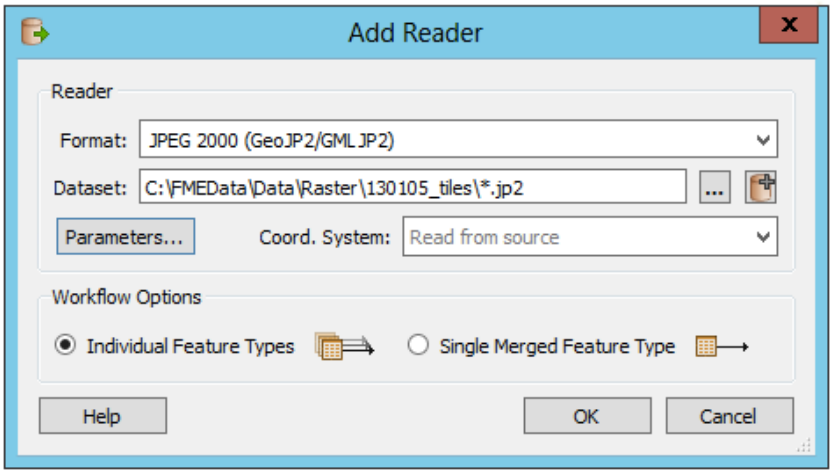
Example 5: Raster Tiling for Bing Maps	
Scenario	FME user; City of Interopolis, Planning Department
Data	Interopolis Tiles (JPEG2000)
Overall Goal	Create tiles suitable for use in Bing Maps
Demonstrates	Mosaicking, Reprojection, Tiling, Resampling, Pyramiding
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\Raster5a-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\Raster5b-Complete.fmw

The overall goal here is to turn a set of source data into tiles that can be used directly in Bing Maps. The *WebMapTiler* transformer is the key transformer to use for such a task.

1) Start Workbench
Start Workbench with an empty workspace. Add a Reader to read MrSID format files.

Reader Format JPEG 2000 (Joint Photographic Experts Group 2000)
Reader Dataset *C:\FMEData\Data\Raster\130105_tiles*.jp2*

Select all the jp2 files in the specified folder



2) Add Writer
Now add a Writer to the workspace:

Writer Format PNG (Portable Network Graphics)
Writer Dataset *C:\FMEData\Output\DemoOutput*

Connect the Reader and Writer Feature Types.

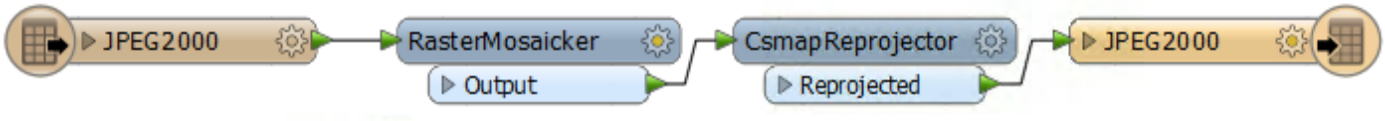
3) Add RasterMosaicker
The incoming tiles need to be merged together for the *WebMapTiler* to be able to do its job. Add a *RasterMosaicker* transformer to mosaic the incoming data.

The default parameters are adequate for this project.

4) Add Reprojector
Because the *WebMapTiler* only accepts features in specific coordinate systems, the mosaicked data will need to be reprojected.

Add a *CsmapReprojector* transformer. Reproject the data as follows:

Source Coordinate System	LL84
Destination Coordinate System	SPHERICAL_MERCATOR
Interpolation Type	Nearest Neighbor
Cell Size	Preserve Cells

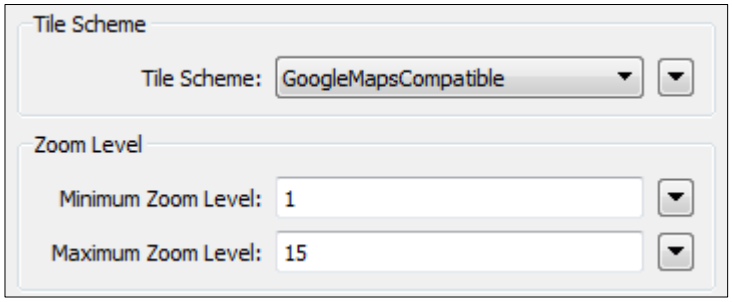


Ms. Analyst says…

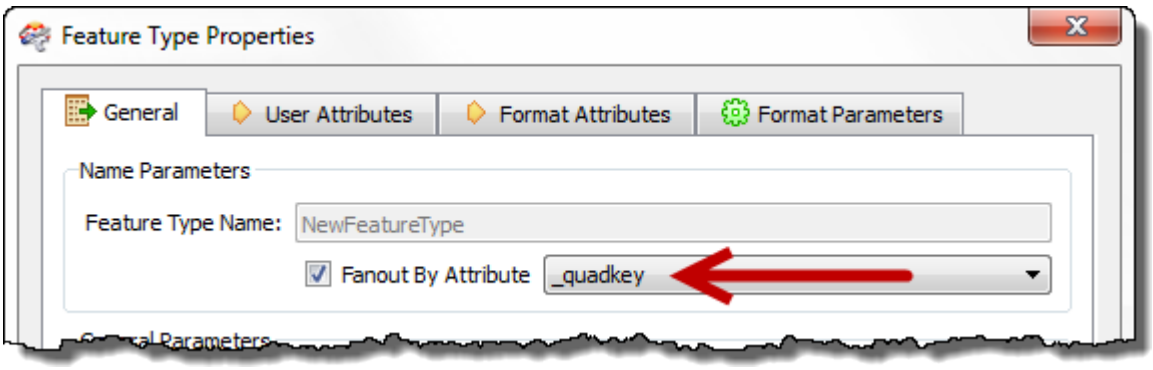
"Because reprojection skews and stretches raster features, images might get black 'nodata' borders which might not be removed by Mosaicking. To avoid this problem, be sure to mosaic raster tiles into a single feature, BEFORE reprojecting them."

5) Add WebMapTiler
Now add a *WebMapTiler* transformer after the Reprojector. Set:

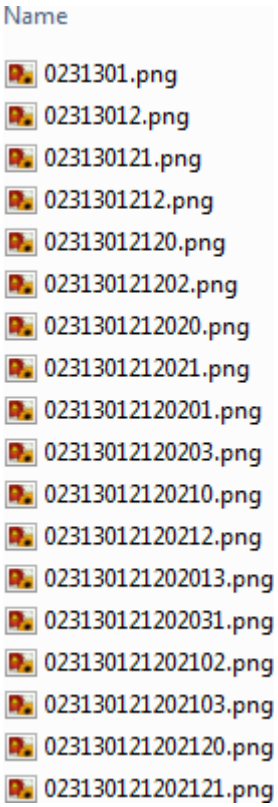
Tile Scheme	GoogleMapsCompatible
Minimum Zoom Level	1
Maximum Zoom Level	15
Interpolation Type	Nearest Neighbor



6) Set Fanout
Finally, open the properties dialog for the Writer Feature Type, and change the Fanout from fme_basename to _quadkey.



7) Save and Run Workspace
Save and then run the workspace.
The translation will take approximately five minutes to complete.



The result should be 18 tiles; the smallest tile is level 7, the largest level 15.

Levels 1 to 6 won't be present (even though specified) because the ground resolution (meters/pixel) would be smaller than a single pixel. For example, a zoom level 1 tile is over 78,000m per pixel, and this set of data is only 1,800m in width.

Notice how levels from 13 upwards have multiple tiles per level. That's because the tile boundaries for these levels cut through the source data, so the data is divided between these different tiles.

Inspect each tile in the FME Data Inspector. You'll find that the number of rows and columns remains constant, but the cell spacing gets smaller, the higher the level.

Advanced Task
If you have time, try to get data suitable for Google Maps.

This time you should use an *AttributeCreator* (or *StringConcatenator*) transformer to create a unique key from a combination of level, row, and column attributes; and then use that in place of _quadkey in the Feature Type Fanout.

Session Review

This session was all about raster data and FME.

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

Theory

- A raster dataset can be composed of one or more ***Bands*** (effectively layers) including an alpha (transparency) band
- ***Interpretation*** represents the structural composition of a raster feature, including data type and bit depth
- ***Raster Algebra*** is a set of techniques that carry out calculations and processing on individual cells
- Raster data can be used in 3D as a ***Surface*** or ***Appearance*** on a surface
- ***Raster web delivery*** is all about reducing data transfer volumes

FME Skills

- The ability to restructure raster datasets, and manage bands
- The ability to convert raster data from one interpretation to another (and understand when it is correct to do so)
- The ability to apply algebra to raster features
- The ability to create a 3D surface from a numeric raster dataset, and to apply an image raster dataset as an appearance
- The ability to create map tiles suitable for use in Bing Maps or Google Maps