

## 2DArcReplacer

The 2DArcReplacer replaces the geometry of the feature with a two-dimensional arc whose shape is set by the parameters, which can be either constant floating point values or the values of existing attributes.

### Parameters

Each parameter may either be entered as a number, or taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Center X and Center Y

The center x and y parameters set the origin of the arc. If these values are blank, and the input features are points, the existing feature x/y values will determine the center of the arcs. If the parameter values are blank, and the input features are not points, the operation is undefined.

#### Primary Axis and Secondary Axis

The primary and secondary axis set the radii of the arc. Note that the primary axis need not be larger than the secondary, however, the rotation angle is always measured from the x axis to the primary axis. To make a circular arc, set both the primary and secondary axis to the same value.

#### Start Angle and Sweep Angle

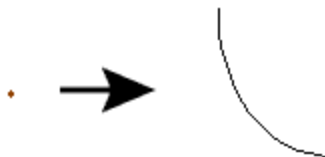
The start angle controls where the arc begins, and is measured in degrees counterclockwise from horizontal.

The sweep angle controls the duration of the arc, and is measured in degrees. The arc will run from the start angle to the start angle plus the sweep angle.

#### Rotation Angle

The rotation angle is measured in degrees counterclockwise from horizontal, and measures the rotation of the primary axis from horizontal.

## Example



### Usage Notes

If the parameters for the arc are not known and need to be calculated from a linear feature's geometry, use the ArcEstimator.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @XValue, @YValue

## **2DBoxReplacer**

Replaces the geometry of the feature with a two-dimensional box whose minimums and maximums are fixed values, or are taken from attributes in the original feature.

### **Parameters**

Min and Max X and Y Values

You can enter parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

### **Usage Notes**

To replace a feature with its bounding box in one step, use the `BoundingBoxReplacer`.

To retrieve the bounds of a feature into attributes, use the `BoundsExtractor`.

### **Transformer Category**

Manipulators

### **Transformer Type**

Feature-based

### **Technical History**

Associated FME function or factory: @XValue, @YValue, @GeometryType

## **2DEllipseReplacer**

Replaces the geometry of the feature with a two-dimensional ellipse whose shape is set by the parameters, values or the values of existing attributes.

### **Parameters**

You can enter the parameters below as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Center X and Y

The center x and y parameters set the origin of the ellipse. If these values are blank, and the input features are points, the existing feature x/y values will determine the center of the ellipses. If the parameter values are blank, and the input features are not points, the operation is undefined.

#### Primary and Secondary Axis

The primary and secondary axis set the radii of the ellipse. Note that the primary axis need not be larger than the secondary, however, the rotation angle is always measured from the x axis to the primary axis. To make a circular arc, set both the primary and secondary axis to the same value.

#### Rotation

The rotation angle is measured in degrees counterclockwise from horizontal, and measures the rotation of the primary axis from horizontal.

### **Usage Notes**

If the parameters for the ellipse are not known and need to be calculated from a linear feature's geometry, you should use the ArcEstimator.

### **Transformer Category**

Manipulators

### **Transformer Type**

Feature-based

### **Technical History**

Associated FME function or factory: @XValue, @YValue

## **2DForcer**

Removes any elevation (Z) coordinates which may (or may not) have been present on the original feature.

Some formats incur extra overhead to store three-dimensional data; in these cases, you may want to remove the elevation coordinates.

After passing through this transformer, the feature will be two-dimensional.

### **Parameters**

Not applicable.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @Dimension

## 2DGridAccumulator

Replaces the input features with a grid of two-dimensional point or polygon features having the spacing specified covering (at least) the bounding box area of all the features which enter the transformer.

### Parameters

#### Column Width and Row Height

The Column Width and Row Height parameters specify in ground units the distance between output points in the grid or the width and height of the output polygons.

**Important:** If you want to choose attribute values as the Column Width and/or Row Height, only attributes selected in the Group By field are allowed. This is because all other attributes will be dropped. It also avoids the ambiguity that would arise if features within the same group had different values in these attributes: FME would not know which value to pick.

#### Type of Grid to Create

Choose point or polygon features.

#### Column Attribute and Row Attribute

If Column Attribute or Row Attribute are specified, attributes will be added to each output tile that identify the position of that tile in the input raster. These indices are zero-based.

#### Group By

The input features may be partitioned into groups based on attribute values and one bounding box feature is output for each group. If you do not specify any Group By attributes, then all input features will be processed together and a single bounding box will be output.

### Transformer Category

Collectors

### Transformer History

This transformer has been renamed from 2DGridReplacer.

### Technical History

Associated FME function or factory: BoundingBoxFactory

## **2DGridCreator**

Creates a grid of two-dimensional point or polygon features, at the origin and using the offsets specified. Each created feature will have a row and column attribute that indicates its position in the grid.

### **Parameters**

#### Lower Left X and Y Coordinates

The X Lower Right Coordinate and Y Lower Right Coordinate parameters specify the origin for the lower-right corner of the grid as a whole.

#### Number of Columns and Rows

The Number of Columns and Number of Rows parameters specify the number of rows and columns the grid will have. This must be at least one.

#### Column Width and Row Height

The Column Width and Row Height parameters specify in ground units the distance between output points in the grid or the width and height of the output polygons.

#### Type of Grid to Create

Choose point or polygon features.

#### Row Attribute and Column Attribute

If Row Attribute or Column Attribute are specified, attributes will be added to each output tile that identify the position of that tile in the input raster. These indices are zero-based.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: CreationFactory

## 2DPointAdder

Adds a two-dimensional point as the last vertex of the feature.

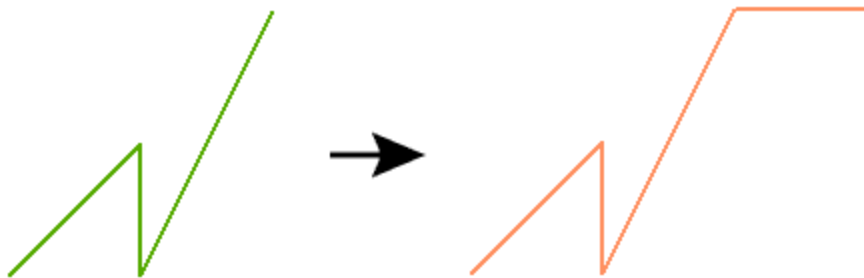
If the feature turns into a closed polygon as a result of adding the point, it will be tagged as an area feature. Otherwise, it will be tagged as a line. (However, if this was the first point added, it will be tagged as a point.)

### Parameters

X and Y Values

You can either enter constants, or choose coordinates from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Example:



### Usage Notes

If you're adding a point to close a polygon, it's easier to use the LineCloser.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Dimension, @GeometryType, @SupplyAttributes, @Tcl2

## **2DPointReplacer**

Replaces the geometry of the feature with a two-dimensional point whose ordinates are taken from attributes in the original feature. If the feature was originally a text feature, it remains a text feature but its insertion point is moved. All other features become point features.

### **Parameters**

X and Y Values

You can enter the parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

### **Usage Notes**

You can use the 2DPointAdder transformer to add additional vertices to the resulting feature.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @XValue, @YValue



### 3DAffiner

Performs 3D affine transformation on the coordinates of the feature.

An affine transformation preserves parallelism of lines and planes in geometry. That is, any lines or planes that were parallel before the transformation are parallel after the transformation. As well, if a number of points falling on a straight line or a plane are transformed, the resulting coordinates will fall on a straight line or plane in the new coordinate system.

Affine transformations include translations, rotations, scalings, and reflections.

#### Parameters

Coefficient A to L

The transformation results in the x and y coordinates being modified by:

$$x' = Ax + By + Cz + D$$

$$y' = Ex + Fy + Gz + H$$

$$z' = Ix + Jy + Kz + L$$

Coefficients <A>, <F> and <K> must be non-zero.

You can enter parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will be stroked to lines and polygons (respectively) before undergoing the transformation; otherwise, only their center points will be transformed.

#### Transformer Category

Manipulators

#### Technical History

Associated FME function or factory: @Affine

### **3DArcReplacer**

Replaces the geometry of the feature with a two-dimensional arc whose shape is set by the parameters, which can be either constant floating point values or the values of existing attributes.

#### **Parameters**

You can enter parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

##### Center X, Y and Z

The center x, y, and z parameters set the origin of the arc.

##### Primary and Secondary Axis

The primary and secondary axis set the radii of the arc. Note that the primary axis need not be larger than the secondary, however, the rotation angle is always measured from the x axis to the primary axis. To make a circular arc, set both the primary and secondary axis to the same value.

##### Start Angle

The start angle controls where the arc begins, and is measured in degrees counterclockwise from horizontal.

##### Sweep Angle

The sweep angle controls the duration of the arc, and is measured in degrees. The arc will run from the start angle to the start angle plus the sweep angle.

##### Rotation

The rotation angle is measured in degrees counterclockwise from horizontal, and measures the rotation of the primary axis from horizontal.

#### **Usage Notes**

If the parameters for the arc are not known and need to be calculated from a linear feature's geometry, the ArcEstimator should be used.

#### **Transformer Category**

Manipulators

#### **Technical History**

Associated FME function or factory: @XValue, @YValue

## **3DForcer**

Turns two-dimensional data into three-dimensional data by adding a z-value to every coordinate.

### **Parameters**

Elevation

You can enter the parameter as a floating point number, or choose it from the value of a feature attribute by selecting the attribute name from the pull-down list.

If the feature was two-dimensional, it becomes three-dimensional with a constant elevation.

If the feature was already three-dimensional, its previous elevations are wiped out and replaced with the value held in the specified attribute.

### **Usage Notes**

Note that the SurfaceDraper transformer (available in FME Professional and higher) can be used to supply interpolated elevation values to the vertices of 2D features based on a 3D grid or set of 3D features.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @ZValue

### **3DInterpolator**

Interpolates elevation values along a non-aggregated linear feature from a starting value to an ending value. The resulting feature's elevation will monotonically increase (or decrease) from the starting value to the ending value.

If the feature was two-dimensional, it becomes three-dimensional.

If the feature was three-dimensional, its previous elevations are removed and replaced.

#### **Parameters**

You can enter the parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

##### **Starting Elevation**

The elevation value for the first vertex of the input feature.

##### **Ending Elevation**

The elevation value for the last vertex of the input feature.

#### **Usage Notes**

The SurfaceDraper transformer (available in FME Professional edition and higher) can be used to supply interpolated elevation values based on a 3D grid or set of 3D features to the vertices of two-dimensional features.

#### **Geometry Handling**

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Classic, an arc is demoted to its centerpoint prior to interpolation. If the parameter is set to Enhanced, the elevation of an arc is interpolated on its defined end-points and possibly mid-point (if the arc is an arc by 3 points).

#### **FME Licensing Level**

FME Professional edition and above

#### **Transformer Category**

Manipulators

#### **Technical History**

Associated FME function or factory: @Interpolate

### **3DPointAdder**

Adds a three-dimensional point as the last vertex of the feature. If the feature turns into a closed polygon as a result of adding the point, it will be tagged as an area feature; otherwise, it will be tagged as a line. (However, if this was the first point added, it will be tagged as a point.)

#### **Parameters**

X, Y and Z Values:

You can either choose coordinates from the value of a feature attribute by selecting the attribute name from the pull-down list, or enter constants.

Note that the SurfaceDraper transformer (available in FME Professional and higher) can be used to supply interpolated elevation values to the vertices of 2D features based on a 3D grid or set of 3D features.

#### **Transformer Category**

Manipulators

#### **Technical History**

Associated FME function or factory: @XValue, @YValue, @ZValue, @GeometryType, @TCL

## **3DPointReplacer**

Replaces the geometry of the feature with a three-dimensional point whose ordinates are taken from attributes in the original feature. If the feature was originally a text feature, it remains a text feature but its insertion point is moved. All other features become point features.

### **Parameters**

X, Y and Z Values:

You can either choose coordinates from the value of a feature attribute by selecting the attribute name from the pull-down list, or enter numbers.

### **Usage Notes**

The SurfaceDraper transformer (available in FME Professional and higher) can be used to supply interpolated elevation values to the vertices of 2D features based on a 3D grid or set of 3D features.

Additional vertices can later be added to the resulting feature using the 3DPointAdder transformer.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @Dimension, @GeometryType, @SupplyAttributes, @Tcl2

## Affiner

Performs an affine transformation on the coordinates of the feature.

An affine transformation preserves lines and parallelism in geometry. That is, any lines that were parallel before the transformation are parallel after the transformation. In addition, if a number of points falling on a straight line are transformed, the resulting coordinates will fall on a straight line in the new coordinate system.

Affine transformations include translations, rotations, scalings, and reflections.

### Parameters

Coefficient A to F

The transformation results in the x and y coordinates being modified by:

$$x' = Ax + By + C$$

$$y' = Dx + Ey + F$$

Coefficients <A>, and <E> must be non-zero.

You can enter parameters as a number, or choose them from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Geometry Handling

If the Geometry Handling Advanced setting is set to Enhanced in the workspace, then arcs and ellipses will be preserved; otherwise only their center points will be transformed.

### Related Transformers

Please also see the AffineWarper, which performs warping operations on the spatial coordinates of features. The AffineWarper is used to adjust a set of observed features so they more closely match some set of reference features.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Affine

## AffineWarper

Performs warping operations on the spatial coordinates of features. It is used to adjust a set of observed features so they more closely match some set of reference features. This transformer computes an affine (scale, rotation, and offset) transformation based on **CONTROL** vector features and applies this transformation to the **OBSERVED** features to generate output, and produces good corrections when the entire set of **OBSERVED** data requires a single transformation.

Each **CONTROL** feature represents a control vector whose start point is at some location in the original **OBSERVED** data space, and whose end point is at the corresponding location in the desired output data space. The control vector represents the correction required to go from the observed vertex to the desired vertex. (Control vectors with only one point are interpreted as a requirement that this location not change from the observed dataset to the reference dataset. This is often referred to as a tie point.)

### Input Ports

Two sets of features must be routed into this transformer.

- **CONTROL**: Features that enter the **CONTROL** port represent the control features used to compute the corrections.
- **OBSERVED**: Features that enter the **OBSERVED** port are the features that will be corrected.

### Output Ports

- **CORRECTED**: The modified **OBSERVED** features are output via the **CORRECTED** port.

### Related Transformers

- The RubberSheeter transformer provides similar functionality but applies a different transformation to each **OBSERVED** vertex, depending on its distance to nearby **CONTROL** vectors. This makes the RubberSheeter more appropriate for cases when the distortions in the data are not constant.
- Please also see the *Affiner*, which performs an affine transformation on the coordinates of the feature.

### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: WarpFactory



## **AggregateFilter**

Routes features differently depending on if their geometry consists of an aggregate of several primitive geometries, or if it is a simple, single piece geometry.

---

Note: To distinguish between different *types* of geometry, use the `GeometryFilter`.

---

## **Transformer Category**

Filters

## **Technical History**

Associated FME function or factory: `TestFactory`

## Aggregator

Combines feature geometries into aggregates. One aggregate feature is output for each unique combination of values of the attributes specified in the Group By parameter.

Features are added to the aggregate being built in the order they are received by this factory. If the order of individual features within the resulting aggregate is important, you can first route the features through a Sorter.

## Parameters

### Group By

If you do not select Group By attributes, then all features fall into the same group and a single feature is output with all of the input geometry aggregated onto it. If only a single feature was in a particular group, then it is output as an aggregate with one part.

When you select Group By attributes, this transformer will aggregate the geometries of the input features, based on the selection in the Group By parameter. One feature is output for each group resulting from the Group By parameter.

### Input is Ordered by Group

If Input is Ordered by Group is set to Yes, the transformer will output all the aggregates it has accumulated every time the values of the Group By attributes change from one feature to the next.

### Attribute(s) to Sum, Attribute(s) to Average, Attributes to Average, Weighted by Area

If desired, attributes can be added to the resulting area features that contain the sum, average, or weighted (by area) average of some original attribute values. Note that non-area features will have a weight of zero in the weighted average calculation. Therefore if there are no area features, the weighted average will be infinity.

### List Name

If the optional list name is supplied, a list is made of all the attributes of each feature that is aggregated when creating the output feature. This allows later inspection of the attributes of the original features that make up the aggregate.

### Count Attribute

If a Count Attribute is entered, then an attribute with this name will be added to each output aggregate, containing the number of features that were combined to create the aggregate.

### Accumulate Attributes

If Accumulate Attributes is set to Yes, the attributes from the original features will be merged onto the output AGGREGATE features.

### Attribute Containing Geometry Name

Selecting the Attribute Containing Geometry Name will set the attribute to the name of the geometry.

### Attribute(s) to Concatenate

If desired, attributes can be concatenated so that the resulting aggregate feature maintains source attribute values under the same attribute name.

The Separator Character is used to separate elements. For example, if features with attribute values of "River Rd", "Marine Dr" and "HWY 1" for the attribute named "Road Name" respectively, and the Separator Character is ", " (a comma followed by a space) on the output aggregate feature, the attribute "Road Name" will contain "River Rd, Marine Dr, HWY 1" as a value.

### Separator Character

The separator character can be expressed as a regular character but it can also contain special characters beginning with a backslash ("\"), as specified in the following table.

If the sequence is not listed in the table, the backslash character is ignored. (For example, entering lan\es will be interpreted as lanes.) If no separator character is supplied, attributes will be concatenated without a separator.

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

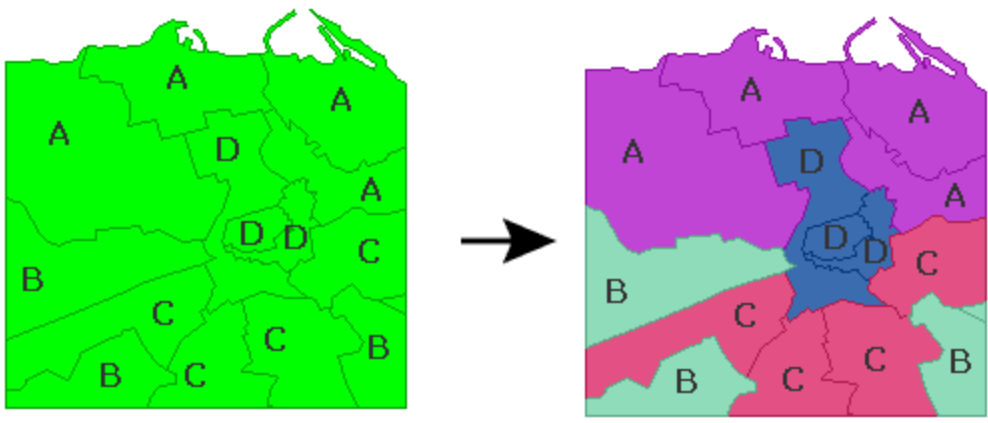
**Geometry Handling**

If the Geometry Handling advanced setting is set to Enhanced in the workspace, the aggregated geometry will preserve arcs, ellipses, and text; they will otherwise be converted into points in the aggregated geometry.

**Usage Notes**

This transformer will not dissolve adjacent area boundaries – it simply creates a collection of all that it is given. Use the Dissolver if adjacent areas are to be merged into a single area.

**Example**



**Transformer Category**

Collectors

**Technical History**

Associated FME Function or Factory: AggregateFactory

## Amalgamator

Generalizes polygonal input by connecting nearby geometries.

The Amalgamator accepts polygonal geometries – including donuts – as input, and produces triangles that join input features into connected pieces, or amalgams. Small holes are removed at the end of this process.

The strategy for generalizing polygonal geometries is as follows:


- Redirect non-polygonal geometries onto the INVALID port.
- Dissolve the input polygonal geometries to remove shared boundaries and overlapping regions.
- Densify the dissolved polygonal geometries.
- Determine the convex hull of the densified polygonal geometries.
- Overlay the convex hull against the densified polygonal geometries to compute empty regions between the polygonal geometries.
- Triangulate the empty regions. Keep only triangles that are sufficiently short in length – those that satisfy the Maximum Triangle Length parameter.
- Dissolve the triangles to form connectors.
- For each connector, check to see if it is valid. It is valid if it shares a boundary with a dissolved polygonal geometry.
- Output all features not touched by any connectors to the UNTOUCHED port.
- Dissolve valid connectors with dissolved polygonal geometries to form amalgams.
- For each amalgam, remove small holes that do not satisfy the Minimum Hole Area parameter. Remaining (large) holes are output to the HOLES port.
- For each amalgam, find all triangles that formed a part of its connectors. Output these triangles to the TRIANGLES port.
- Output all amalgams to the AMALGAMATED port.
- For attribute behavior, please see List Name and ID Attribute in the Parameters section.




### Input Port

- AREAS: Polygonal geometries, including donuts. The polygonal geometries may overlap and share boundaries. However, each geometry should be valid (that is, not self-intersecting or non-planar).

### Output Ports

- AMALGAMATED: Amalgams computed from input polygonal geometries.
- UNTOUCHED: Input polygonal geometries that are not touched by valid triangle connectors.
- HOLES: Holes in the amalgams whose areas exceed the Minimum Hole Area parameter.
- TRIANGLES: Triangles that form the valid connectors joining input polygonal geometries.
- INVALID: Non-polygonal input. Occasionally, if an unexpected condition is met, some invalid intermediate results will be posted to this port.

Input	Output
	 <p data-bbox="456 1688 639 1715">TRIANGLES (red)</p>

Input	Output
	 <p data-bbox="456 491 708 520">AMALGAMATED (green)</p>
	 <p data-bbox="456 785 610 814">HOLES (black)</p>

### Parameters

#### Group By (optional)





By specifying one or more Group By attributes, the input polygonal features will be partitioned into groups and the amalgamation process will be executed separately on each group. Within each group, all features will have the same values for the selected Group By attributes.

If no Group By attributes are selected, a single group will be formed containing all input polygonal features. By default, no Group By attributes are selected.

#### Amalgamation Mode

This parameter controls the mode of amalgamation. The Amalgamator is conceptually a binary operator that causes two nearby geometrical details to connect together. However, two geometrical details may be on the same geometry. Imagine two peninsulas protruding from the same coast line, or two different geometries, such as two neighboring islands. Therefore, a number of options are provided here to accommodate the two conceptual models:

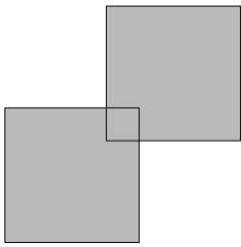
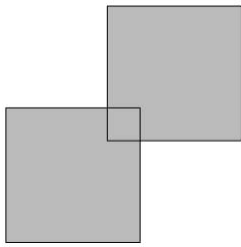
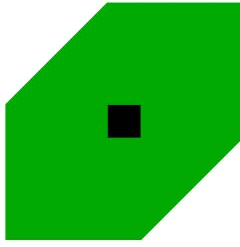
- Self Amalgamation: This mode amalgamates a polygonal geometry against itself. It would handle the two peninsulas case, but not the two neighboring islands case.
- Binary Amalgamation (default): This mode amalgamates different polygonal geometries. It would handle the two neighboring islands case, but not the two peninsulas case.
- Self, Binary Amalgamation: This mode combines the Self Amalgamation and the Binary Amalgamation. It would handle both the two peninsulas case and the two neighboring islands case.

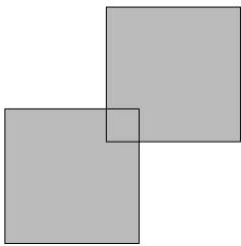
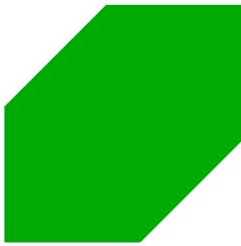
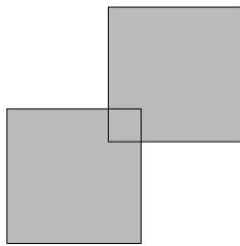
Input	Output: AMALGAMATED
	 <p data-bbox="467 594 667 621">Self Amalgamation</p>
	 <p data-bbox="467 978 695 1005">Binary Amalgamation</p>
	 <p data-bbox="467 1339 748 1367">Self, Binary Amalgamation</p>

### Dissolve Input

This parameter controls whether input polygonal features are dissolved up front. The default value is Yes.

- Yes: This value dissolves input polygonal features. The Amalgamator was designed with the assumption that input features do not overlap. This value enforces that assumption. Dissolving input up front also remedies input polygonal features that may not be overlapping, but whose shared boundaries become overlapping after the densification step due to limited precision. The overlapping due to limited precision could reduce the performance of the Amalgamator.
- No: This value is meant for advanced users who want finer control of the Amalgamator. Some users would rather not dissolve the input to improve performance because in some cases, not dissolving the input will not cause undesirable side effects.

Overlapping Input	Dissolve Input: Yes Amalgamation Mode: Binary	Dissolve Input: No Amalgamation Mode: Binary
		
Explanation	The input dissolves into one polygon, thus the Binary mode causes the one polygon not to amalgamate.	The input is not dissolved, thus the Binary mode causes the two input features to amalgamate, but because overlapping features were not dissolved, a hole results in the middle of the amalgam.

Overlapping Input	Dissolve Input: Yes Amalgamation Mode: Self	Dissolve Input: No Amalgamation Mode: Self
		
Explanation	The input dissolves into one polygon, thus the Self mode causes the one polygon to amalgamate.	The input is not dissolved, thus the Self mode causes the two input features not to amalgamate.

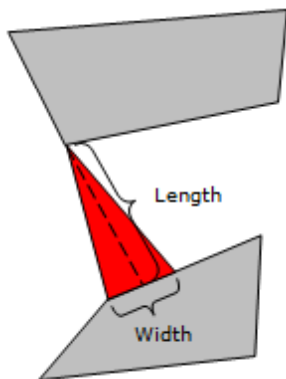
### Maximum Triangle Width

This parameter controls the widths of triangles that form the connectors. The width of the triangle is the width of its base, which is shared by a geometrical detail (see figure below). The larger its value, the wider the triangles will be. In terms of triangle count, decreasing this value generally increases (and will not decrease) the number of triangles generated.

More rigorously, after the input polygonal geometries are dissolved, extra vertices are added through a densification process. The densification interval controls the widths of triangles created. This parameter specifies the length of the densification interval.

### Maximum Triangle Length

This parameter controls the lengths of triangles that form the connectors. The length of the triangle is approximately the length of its median, which bisects the triangle's base (see diagram below). This value should not be less than the Maximum Triangle Width. The larger its value, the longer a triangle can be before being filtered out. In terms of triangle count, decreasing this value generally decreases (and will not increase) the number of triangles generated.



### Minimum Hole Area

This parameter controls which holes should be eliminated from the amalgams. The larger its value, the larger the remaining holes will be. In terms of hole count, decreasing this value generally increases (and will not decrease) the number of holes remaining in the amalgams.

### List Name (optional)

This parameter specifies the name of a list attribute for the amalgams. For each amalgam, this list will contain an entry for each input feature whose polygonal geometry shares a boundary with the amalgam. All attributes from the input feature are recorded in the list entry, except feature level attributes prefixed by **fme\_**.

### ID Attribute (optional)

This parameter specifies the name of a unique identifier for the amalgams. If specified, each amalgam will receive an ID value that is unique across groups. All triangles and holes contained in an amalgam will receive the same ID as that amalgam.

### Usage Notes

Dissolving the input is necessary to remove shared boundaries and overlapping regions, with which the Amalgamator cannot be expected to function properly. However, dissolving the input has some side effects:

- If multiple input features dissolve into a single feature, then only one set of feature attributes are kept on the dissolved feature.
- If two input geometries share a boundary – for example, two peninsulas glued together – the user might expect triangle connectors to form between the two geometries when the “Binary Amalgamation” mode is selected. However, such geometries will first be dissolved, making it much more likely that “Binary Amalgamation” will not cause triangle connectors to form between the peninsulas. To overcome this issue, please select the “Self, Binary Amalgamation” mode.

If the Maximum Triangle Length specified is less than Maximum Triangle Width, the results may be unpredictable.

### Transformer Category

Manipulators

### fmepedia

See fmepedia for additional information about this transformer.

### Related Transformers

Generalizer

### Technical History

Associated FME function or factory: AmalgamatorFactory

**Note: This transformer uses functionality from the GEOS library (<http://geos.refractions.net/>), which is distributed under the terms of the Free Software Foundation's LGPL license (<http://www.gnu.org/licenses/lgpl.html>). To comply**



**with the terms of the LGPL, Safe Software Inc. has set up a website (<http://www.safe.com/foss>) to provide further information.**

## AnchoredSnapper

Takes a series of features that match the input specification and performs snapping on the features that lie within the specified tolerance from other features that match the input specification. You can use this transformer to perform cleaning operations on data during a translation.

The difference between the AnchoredSnapper and the Snapper is that anchor features are considered to be accurate and consequently do not move.

### Output Ports

- SNAPPED: Features whose geometry is changed by the transformer.
- UNTOUCHED: Features that leave the transformer without being changed.

### Parameters

#### Group By

If you select Group By attributes, only those features with the same Group By attribute values will be snapped together.

#### Snapping Type

When Snapping Type is *End Point Snapping*:

- The transformer snaps endpoints of features that enter via the CANDIDATE port to endpoints of features that enter via the ANCHOR port. ANCHOR features are not output.
- Point features can be used as ANCHOR or CANDIDATE features, and CANDIDATE points will be snapped together (or to a linear base feature) as well.

The transformer will not alter area features.

The Add Additional Vertex parameter is enabled (see below).

When Snapping Type is *Vertex Snapping*:

- The transformer snaps vertices of features that enter via the CANDIDATE port to vertices of features that enter via the ANCHOR port. ANCHOR features are not output.
- Point features can be used as ANCHOR or CANDIDATE features, and CANDIDATE points will be snapped together (or to a linear ANCHOR feature) as well.

The transformer will alter area features.

The Add Additional Vertex parameter is ignored.

#### Snapping Tolerance

Snapping Tolerance specifies the distance that the snapping occurs between features. This distance is in ground units.

#### Add Additional Vertex

This parameter is enabled only if Snapping Type is End Point Snapping. It controls how lines are modified when they are snapped.

- NEVER: the endpoint of a line is moved when it is snapped and no additional vertex is added.
- ALWAYS: the original end point (start point) of the line becomes the second from the end (start) and a new vertex is added to complete the snap.
- FORWARD\_ONLY: a new vertex is added only when doing so creates an angle greater than 90 degrees with the original line segment. In this case, if adding the vertex would cause a less than 90-degree angle, the old end point is still moved.

### Usage Notes

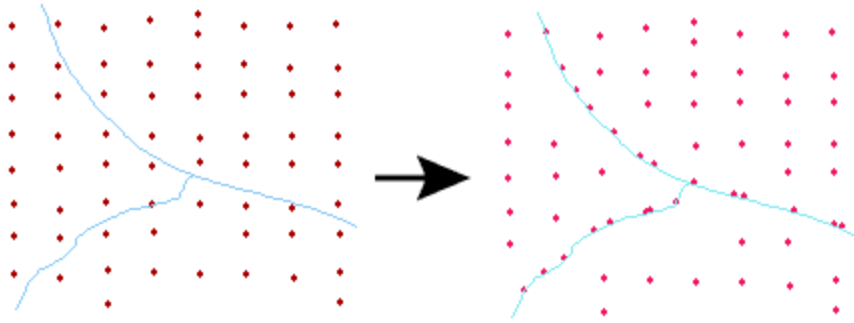
- Any feature that undergoes dimensional collapse as a result of being snapped will be logged as "degenerate" and dropped. "Dimensional collapse" refers to a line or area that becomes a point, or an area that becomes a line.

- A short cleanup step is performed after snapping. This step will remove duplicate points, and may create aggregates to preserve overlapping, directed segments.

### Geometry Handling

Note: If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, arcs are snapped as linear features, and ellipses are snapped as polygonal features; otherwise, arcs and ellipses are both snapped as point features located at their respective center points.

### Example



### FME Licensing Level

FME Professional Edition and above

### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: SnappingFactory

## AngleConverter

Converts angles of a feature's geometry and/ or attributes from one representation to another.

### Parameters

Convert Angles on

Choose from Geometry and Attributes or Attributes Only.

If geometry is being converted, then it assumed that the geometry of the input features is represented in the source angular format and will be converted to the destination angular format. For example, if a two-point line is represented in DDMSS as 1800000,223000 900000,450000, then it would be represented as 180,22.5 90.0,45.0 when converted to DECIMAL\_DEGREES.

Attributes Containing Angles

Choose from the list of attributes containing angles that are to be converted.

Source and Target Angle Type

DECIMAL\_DEGREES, RADIANS, DDDMMSS, DDDMMSSSS, NSDDEW, NSDDMSSEW and DD1234

where:

- DECIMAL\_DEGREES - The angle is represented as decimal degrees.
- RADIANS - The angle is represented as radians.
- DDDMMSS - The angle is represented as an integer in degrees, minutes, seconds. For example, 180 degrees would be 1800000, and 22.5 degrees is represented as 22 degrees and 30 minutes and encoded as 223000.
- DDDMMSSSS - The angle is represented as an integer in degrees, minutes, hundredths of a second. For example, 180 degrees would be 180000000, and 22.5 degrees is represented as 22 degrees and 30 minutes and encoded as 22300000.
- NSDDEW - The angle is represented as decimal degrees with an indicated bearing. The degrees will never exceed 90 degrees. For example, 150 degrees would be represented as S30E.
- NSDDMSSEW - The angle is represented as an integer in degrees, minutes, and seconds with a bearing indicating direction. The degrees will never exceed 90. For example 180 degrees would be S00-00-00E, and 22.5 degrees is represented as North 22 degrees and 30 minutes East and encoded as N22-30-00E.
- DD1234 - The angle is represented as decimal degrees followed by a quadrant. Quadrant 1=NE, 2=SE, 3=SW and 4=NW. The decimal degrees will never exceed 90. For example 150 degrees will be represented as 30-2.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Angle

## **AngularityCalculator**

Calculates the "angularity" of a linear or area feature. Angularity indicates the degree of curvature of a feature. The higher the value, the more curved its geometry.

A value of 0 indicates the geometry is a straight line. This mode works on linear and polygonal geometries, and all other geometries will receive a value of 0.

The calculation of this property utilizes the angle between segments and, as a result, linear and polygonal features must have more than one segment to receive a non-zero value. Arcs are stroked into lines before determining this value.

### **Parameters**

Angularity Attribute

The name of the attribute that will contain the angularity calculation.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Circularity

## AppearanceAdder

Adds appearance style(s) to the front and/or back side of surfaces. You can also set the texture coordinates of the surfaces.

### Input

- FRONT, BACK: Appearance styles coming in through the FRONT or BACK port will be applied to the front and/or back side of surfaces.
- SURFACE: Features that have a surface geometry.

### Output

- SURFACE: Features that have a surface geometry.
- INVALID\_GEOMETRY: Any SURFACE input features that do not have a surface geometry.
- EXTRA\_APPEARANCE: Extra appearance styles.
- INVALID\_APPEARANCE: If an appearance style has input geometry other than null or raster.

Note: Texture coordinates mapping is currently supported only for faces, composite surfaces, and meshes.

## Coordinate Space Terminology

In order to reduce confusion between the real coordinate space of the surface and the texture coordinate space, this transformer uses "u" and "v" instead of "x" and "y" when referencing the latter. Note that this is also reflected in the parameter names.

### Parameters

#### Group By

Note that only the first "front" and "back" appearance styles will be used in each "Group By" group.

#### Texture Mapping Type

This parameter specifies how the texture defined in an appearance style will be mapped onto the surface.

**Surface Normal:** The textures are projected onto the surfaces along their normals. For composite surfaces and meshes, each of the child parts will be treated separately, since the parts can have different normals.

**From Top View:** The textures are projected onto the surfaces along a single normal – one that is perpendicular to the x-y plane. In this mode, a composite surface is considered as one single geometry when the texture coordinates are applied.

#### Texture Offset

You can specify offset through the Texture u Offset and Texture v Offset parameters.

#### Texture Repeat Factor

Texture u Repeat Factor and Texture v Repeat Factor can be used to specify the number of times the texture is repeated in rows and columns, respectively.

#### Recursive

- No: Only the top level of the surface will be assigned an appearance.
- Yes: The top level of the surface will be assigned an appearance and all parts of the surface will inherit that appearance.

For composite surfaces, the Recursive option is always treated as Yes.

### Usage Notes

Note that this transformer only sets the appearance style(s) to the top level of the surface geometry. To set individual parts of a surface, use a Deaggregator or GeometryCoercer to get the parts before using this transformer.

### Transformer Category

Surfaces

## **Technical History**

Associated FME function or factory: @Geometry, SharedObjectFactory

## **AppearanceExtractor**

Extracts appearance style(s) from the front and/or back side of the surfaces.

The output appearance style feature can either be an attributes-only feature or a feature with raster and attributes, depending on whether a surface has textures on the requested sides.

If a surface does not have any appearance associated with it or it has a default appearance, no appearance styles are output for this surface.

If the geometry of the input feature is not a surface, the feature will be output through the INVALID port.

### **Parameters**

Side Appearance to Extract

This parameter specifies side(s) of the surface from which appearances should be extracted.

Output Unique Appearances

If Output Unique Appearance is set to Yes, a style is output only once if the appearance is shared by multiple surfaces.

Recursive

- Yes: The appearance style(s) are output from the surface and its child parts.
- No: Only the appearance style(s) are output from the top level of the surface.

### **Transformer Category**

Surfaces

### **Technical History**

Associated FME function or factory: SharedObjectFactory



## **AppearanceRemover**

Removes appearance from the front and/or back side of surface features. Removing the appearance of a surface causes that surface to inherit its appearance from its parent, if a parent surface exists.

### **Output Ports**

- **SUCCESSFUL:** All surfaces that do not have any appearances or have their appearances removed successfully are output through the SUCCESSFUL port.
- **INVALID:** If the input feature is not a surface, the feature is output unmodified through the INVALID port.

### **Parameters**

Side Appearance To Remove

This parameter specifies the surface side(s) whose appearance should be removed.

Recursive

- **Yes:** All the appearance(s) from the surfaces and their child parts are removed.
- **No:** Only the appearance(s) from the top level of the surfaces are removed.

### **Transformer Category**

Surfaces

### **Technical History**

Associated FME function or factory: SharedObjectFactory

## AppearanceStyler

Creates an appearance style that can later be applied to a surface (using the AppearanceAdder, for instance).

### Ports

- INPUT: A feature that contains either a raster or no geometry.
- OUTPUT: The set options that become attributes on that feature when is sent out through the OUTPUT port.

### Coordinate Space Terminology

In order to reduce confusion between the real coordinate space of the surface and the texture coordinate space, this transformer uses "u" and "v" instead of "x" and "y" when referencing the latter. Note that this is also reflected in the parameter names.

### Parameters

#### Appearance Name

A name that will help you remember what the appearance is for, such as "castle wall" or "house roof". Note that it does not have to be unique.

#### Diffuse Color

The most instinctive meaning of the color of an object, the essential color that is revealed under pure white light. It is perceived as the color of the object rather than a reflection of the light.

#### Ambient Color

The color that the object reflects when illuminated by color from the surrounding medium rather than direct light.

#### Specular Color

The color of the light reflected from the object through specular reflection (the type of reflection that is characteristic of light reflected from a shiny surface).

#### Emissive Color

Color of the light that the object is emitting itself.

#### Shininess

A value from 0.0 to 1.0 that specifies the shine of specular reflection, with 0.0 being completely dull and 1.0 extremely shiny.

#### Alpha

Specifies the transparency level of the appearance, with 0.0 being completely transparent and 1.0 completely opaque.

#### Texture Center u and v

Used to specify the origin of texture coordinate system. It is only used in conjunction with scaling and rotation.

#### Texture Rotation Angle

Specifies the counter-clockwise rotation angle of the texture in degrees around the texture center (from a line parallel to the u-axis, passing through the texture center).

#### Texture u and v Scaling Factor

Used to specify the amount that the texture should be scaled along the u and v axes respectively.

#### Texture u and v Shearing Factor

Used to specify the amount of shear along the u and v texture coordinate system axes, relative to the center.

#### Texture u and v Offset

Used to specify the offset applied to the texture after all the other transformations are done.

## Texture Wrap Style

Only affects the area outside the 0 to 1 U and V range. Not all texture wrapping styles are supported by each format, in which case the texture wrapping style will be defaulted to a supported style by the consumer.

- NONE means no texture wrapping style is given and behavior outside the 0 to 1 range is unspecified.
- REPEAT\_BOTH will tile the texture in both directions.
- CLAMP\_BOTH clamps both U and V to the 0 to 1 range and a constant boundary color will fill values outside this range.
- CLAMP\_U\_REPEAT\_V clamps U to the 0 to 1 range and tiles in the V direction.
- REPEAT\_U\_CLAMP\_V clamps V to the 0 to 1 range and tiles in the U direction.
- MIRROR will mirror the texture in the U and V direction.
- BORDER\_FILL will use a constant border color to fill values outside the U, V 0 to 1 range.

## Texture Border Color

This parameter is used only with the BORDER\_FILL wrapping style, and is only supported by certain formats. It specifies the color to "bleed" in the space surrounding the texture raster.

## ArcEstimator

Replaces the geometry of the feature with a two-dimensional circular arc whose shape is estimated from the first, middle, and last point of the linear feature passed in. The result is an approximation only and unless the linear feature is previously known to be circular, the result could be completely different than the original feature.

This transformer is most useful when the feature was known to have been a circular arc originally, and was stroked into a line. This transformer can then be used to turn it back into an arc.

For example, a feature containing these points:

```
2,1  
1,2  
0,2.236067977 (sqrt(5))
```

will be turned into an arc feature with this geometry:

```
0,0
```

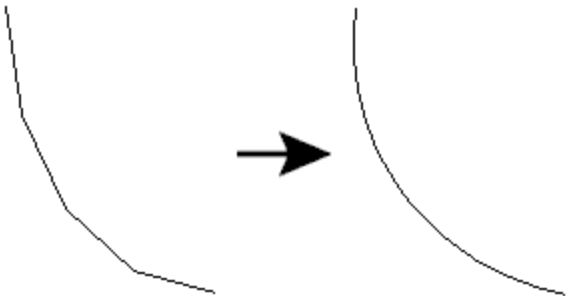
and these attributes:

```
fme_primary_axis 2.236067977  
fme_secondary_axis 2.236067977  
fme_start_angle 26  
fme_sweep_angle 63  
fme_rotation 0
```

## Usage Notes

If the parameters for the arc are already available as attributes on the feature, then use the 2DArcReplacer or 3DArcReplacer transformer.

## Example



## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: SnappingFactory, @ConvertToArc

## **ArcPropertyExtractor**

Sets the given attributes to the properties of an arc geometry. This transformer works on a single feature at a time.

### **Input**

- INPUT: Features with arc geometry.

### **Output**

- OUTPUT: Feature with given attributes set to the properties of the arc geometry.

### **Parameters**

#### Primary Radius Attribute

Attribute name that is set to the length of the primary radius for the ellipse upon which the arc is based.

#### Secondary Radius Attribute

Attribute name that is set to the length of the secondary radius for the ellipse upon which the arc is based.

#### Rotation Attribute

Attribute name that is set to the rotation of the ellipse that defines the arc. The rotation angle specifies the angle in degrees from the horizontal axis to the primary axis in a counterclockwise direction.

#### Start Angle Attribute

Attribute name that is set to the parametric angle of the start point of the arc, in degrees.

#### Sweep Angle Attribute

Attribute name that is set to the parametric sweep angle of the arc, in degrees.

#### Start X Attribute

Attribute name that is set to the x coordinate value for the arc's start point

#### Start Y Attribute

Attribute name that is set to the y coordinate value for the arc's start point

#### Start Z Attribute

Attribute name that is set to the z coordinate value for the arc's start point

#### End X Attribute

Attribute name that is set to the x coordinate value for the arc's end point

#### End Y Attribute

Attribute name that is set to the y coordinate value for the arc's end point

#### End Z Attribute

Attribute name that is set to the z coordinate value for the arc's end point

#### Center X Attribute

Attribute name that is set to the X coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

#### Center Y Attribute

Attribute name that is set to the Y coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

Center Z Attribute

Attribute name that is set to the Z coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

**Transformer Category**

Manipulators

**Transformer Type**

Feature-based

**Technical History**

Associated FME function or factory: @Geometry

## ArcPropertySetter

Sets the properties of an arc geometry to those specified. All parameters are optional; if a value is unspecified, it is left unmodified on the geometry. This transformer works on a single feature at a time.

### Input

- INPUT: Features with arc geometry.

### Output

- OUTPUT: Feature with the properties of the arc geometry modified according to the new values provided.

### Parameters

#### New Primary Axis

The new length of the primary axis for the ellipse upon which the arc is based.

#### New Secondary Axis

The new length of the secondary axis for the ellipse upon which the arc is based.

#### New Rotation

The new rotation of the ellipse that defines the arc. The rotation angle specifies the angle in degrees from the horizontal axis to the primary axis in a counterclockwise direction.

#### New Start Angle

The new parametric angle of the start point of the arc, in degrees.

#### New Sweep Angle

The new parametric sweep angle of the start arc, in degrees.

The default is 360.

#### New Start X

The new x coordinate value for the arc's start point

#### New Start Y

The new y coordinate value for the arc's start point

#### New Start Z

The new z coordinate value for the arc's start point

#### New End X

The new x coordinate value for the arc's end point

#### New End Y

The new y coordinate value for the arc's end point

#### New End Z

The new z coordinate value for the arc's end point

#### New Center X

The new X coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

#### New Center Y

The new Y coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

New Center Z

The new Z coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

**Transformer Category**

Manipulators

**Transformer Type**

Feature-based

**Technical History**

Associated FME function or factory: @Geometry



## ArcSDEGridSnapper

Simulates the ArcSDE conversion on a feature by performing ArcSDE translation, scaling, and coordinate snapping. Also removes duplicate vertices resulting from snapping multiple formerly separate vertices to the same grid point. These coordinates will also be recorded in a list attribute.

If the feature was an aggregate feature, the ArcSDE conversion is executed individually on each geometry in the aggregate feature.

### Parameters

Minimum X and Y

The false origin for x- and y-values.

Scale

A scale factor to convert to integers for x- and y-values.

Output as integer coordinates

Select to view output as either the underlying integer ArcSDE grid or the original plane coordinates

Minimum Z (if 3D)

The false origin for z-values.

Z Scale (if 3D)

A scale factor to convert to integers for z-values.

### Usage Notes

The ArcSDE simulation is executed only when all coordinates of the geometry are within the valid range defined by the ArcSDE format. Otherwise, the conversion is cancelled and an error attribute is added to the feature. Note that illegal geometries, such as areas with dangles, will not be checked for validity on the ArcSDE format.

Arcs and ellipses passed through the ArcSDEGridSnapper will be stroked into lines; this matches the behavior of the ArcSDE writer, which does not support the storage of these types of geometries.

### Transformer Category

Manipulators

### FME Licensing Level

FME Professional Edition and above

### fmepedia

See fmepedia for additional information about this transformer.

### Technical History

Associated FME function or factory: @GridSnapper

## **ArcSDEQuerier**

Performs queries on an ArcSDE spatial database. The queries can have both a spatial and a nonspatial component.

One query is issued to the ArcSDE database for each feature that enters the transformer. The results of the query are then output to the output port that matches the table name, if it exists; otherwise, results are output to the OTHER port.

If the Mode parameter is set to Delete, the results of the query are deleted from ArcSDE before they are output from the transformer. If the Mode parameter is set to Update, the features in ArcSDE matching the query feature will be updated with that feature's complete set of attributes.

The query feature defines the geometry which will be used to define the spatial component of the query, unless the search method is SDE\_NONE. In that case, only an attribute query as defined by the WHERE clause will be executed.

An attribute named `_table_name` will be added to each result feature, specifying which table the result feature came from.

An attribute named `_matched_records` will be added to each query feature, specifying how many database rows the query matched.

### **Parameters**

#### Remove Table Qualifier

If the Remove Table Qualifier parameter is set to Yes, then user names will not be included in table names when they are not required.

#### Table Name in Attribute

If the Table Name in Attribute parameter is blank, the set of tables to query is defined by the Tables parameter, which is set in the "ArcSDE Tables" panel. Alternatively, if the Table Name in Attribute parameter is not blank, the tables to query are read from the specified attribute in the input features. In this case, the tables to query should be specified as a colon-separated list.

### ***ArcSDE Spatial Query Operators***

The complete set of ArcSDE spatial query operators is supported, and each is described below:

#### SDE\_NONE

No spatial filtering is done. Any features matching the Where Clause parameter are output, and if no where clause was specified, then all features from each target table are output.

#### SDE\_ENVELOPE

The envelope of the output feature overlaps or touches the envelope of the search feature.

#### SDE\_COMMON\_POINT

The search feature shares at least one common point with the output feature.

#### SDE\_LINE\_CROSS

The search feature and the output feature have intersecting line segments.

#### SDE\_COMMON\_LINE

The search feature and the output feature share one or more common line segments.

#### SDE\_CP\_OR\_LC

The search feature and the output feature have line segments that intersect or have a common point.

#### SDE\_AI\_OR\_ET

The search feature and the output area feature edge touch (ET) or their areas intersect (AI).

#### SDE\_AREA\_INTERSECT

The search feature and the output feature's area intersect.

## SDE\_AI\_NO\_ET

The output feature and search feature have intersecting areas with no edge touching. One feature is therefore contained in the other.

## SDE\_CONTAINED\_IN

The search feature is contained in the output feature. For area features, this is clear. If search feature is a line, then a linear feature will be output when the search feature path is included in output feature. If search feature is a point, then the search feature will be one of the output features vertices.

## SDE\_CONTAINS

The output feature is contained by the search feature. If both features are linear features, then the output feature must lie on the search feature's path. Point features that lie on a search feature vertex are also output.

## SDE\_CONTAINED\_IN\_NO\_ET

The returned feature must be an area feature that does not touch or share a vertex with the search feature. The returned feature contains the search feature.

## SDE\_CONTAINS\_NO\_ET

The returned feature is contained within the search feature. The returned features cannot touch the edge of, or share a vertex with, the search feature.

## SDE\_POINT\_IN\_POLY

The returned feature contains the first point of the search feature.

## SDE\_IDENTICAL

The returned feature has the same feature type and geometry. This is used to find duplicate data.

## Search Order

The Search Order parameter controls the manner in which the search is performed. If Optimize is specified, then the SDE engine decides how to perform the search. If Attribute First is specified, then the attribute portion of the search is performed first and then the spatial component is performed on the set resulting from the attribute set. If Spatial First is specified, then the spatial search is performed first and then the attribute search is performed on the resulting set. This is useful if the Optimize setting makes the wrong choice and you want to force the search to be performed in a different order.

## Attribute Handling

- Result Attributes Only: result feature attributes are based solely on the query results.
- Keep Query Attributes if Conflict: result feature attributes are a combination of both the query results and the query feature's attributes. If there is a conflict, attribute values are taken from the query feature.
- Keep Result Attributes if Conflict: result feature attributes are a combination of both the query results and query feature's attributes. If there is a conflict, attribute values are taken from the query results.

## Geometry Handling

- Result Geometry Only: result feature geometry is taken from the query results.
- Query Geometry Only: result feature geometry is taken from the query feature.
- Aggregate Query and Result Geometry: result feature geometry is an aggregate of the geometry from the query feature followed by the geometry from the query results.

## Process Duplicates

Specifies whether or not duplicate features will be output from the transformer. If the Mode parameter is set to Update, this parameter also specifies whether duplicate updates will be performed.

## Get Spatial Relations

Specifies whether or not the relationships between the query and result geometries should be computed. Refer to the SDE30QueryFactory

factory documentation for more information (see Workbench Help > FME Functions and Factories Reference).

### **Transformer Category**

Database

### **FME Licensing Level**

FME Professional Edition and above

### **fmepedia**

See fmepedia for additional information about this transformer.

### **Technical History**

Associated FME function or factory: SDE30QueryFactory

## ArcStroker

Converts arc features into lines replacing the feature geometry with a series of edges interpolated along the arc boundary. Ellipse features are converted into polygons by interpolating edges along the elliptical boundary.

If the input geometry is a path consisting of arcs and lines, or an area whose boundary is such a path, then any arcs in the path will be stroked, also using the Number of Interpolated Edges given.

### Parameters

#### Stroke By

You can convert arc or ellipse features into lines either by specifying the Number of Interpolated Edges or the Maximum Deviation.

The Number of Interpolated Edges and the Maximum Deviation may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Number of Interpolated Edges

If this parameter is specified and set to 0, then a reasonable number of edges will be interpolated for the arc.

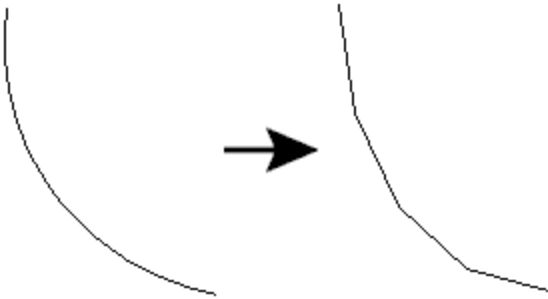
#### Maximum Deviation

If the Maximum Deviation is specified and has a value greater than 0, then arcs are converted into lines such that the maximum distance between the lines and the arcs is not greater than the value specified. If the Maximum Deviation value is greater than the primary or secondary axis of the arc or ellipse, then the converted lines will have minimum number of edges possible. If the Maximum Deviation value is smaller than or equal to 0, then the value of the Stroke Maximum Deviation advanced setting set in the workspace is used. If both values are smaller than or equal to 0, then arcs are converted into lines using the Number of Interpolated Edges value of 0.

#### Make Polygon Out of 360 Degree Arcs

If this parameter is set to Yes, then arc features with a 360-degree sweep angle are converted into polygons. Otherwise, they are converted into lines.

### Example



### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Arc

## AreaBuilder

Takes a set of topologically connected linework and creates topologically correct polygon features where the linework forms closed shapes.

### Input

- **INPUT:** The input lines must be topologically correct and must neither self-intersect nor intersect each other. They must close at their endpoints. If these conditions are met, any area features implied by the input lines are created. (You can use the *Snapper*, *Intersector*, *SelfIntersector*, and *MRF2DCleaner* to clean data that does not meet these conditions before it enters this transformer.) This transformer can also create polygons and donuts (holes/islands). Any lines that cannot be formed into polygons are joined together to create maximum length linestrings.

### Output Ports

- **UNUSED\_LINE:** Contains any lines that did not close.
- **AREA:** Contains the output polygons.

### Parameters

#### Group By

Choose which attributes are preserved by this transformer (for example, you can partition the input linear features). If you do not specify a `GROUP_BY` attribute, then all inputs will be processed together.

No attributes are carried across the `INPUT` features to the output features. However, all `OUTPUT` features are assigned the `GROUP_BY` attributes before being output.

If the `OUTPUT LINE` clause is not specified, then lines that are not part of a polygon will be deleted.

#### Polygon List Name

If you enter a Polygon List Name, a list will be created on each output feature, containing an element for each input feature which contributed to that geometry, in order of appearance.

#### Create Donuts

**Yes:** The resulting polygons will contain holes created by any other resulting polygons they completely contained. Following this, any holes that share a common edge will be dissolved together to make a larger hole.

**No:** The resulting polygons are output via the `AREA` port. Note that if you want to create donut polygons, you will need to use the `DonutBuilder`.

#### Drop Holes

**Yes:** The operation is the same as when the `Create Donuts` parameter is set to `Yes`, except that any polygons that were holes of another polygon will not be output.

#### Hole List Name

If you enter a list name in this field, then a list will be created on each output feature, containing an element for each input feature that became a hole on that geometry, in order of appearance.

### Usage Tips

- You can use one `AreaBuilder` transformer in place of the `PolygonBuilder` and `DonutBuilder`.
- To form polygons without any options for dealing with holes, use the `PolygonBuilder` transformer.

## **Geometry Handling**

1. If the Geometry Handling Advanced setting is set to Enhanced in the workspace, arcs and ellipses are accepted as linear features; they are otherwise rejected as point features.

## **Transformer Category**

Geometric Operators

## **Technical History**

Associated FME function or factory: PolygonFactory, DonutFactory

## **AreaCalculator**

Calculates the area of a polygonal object and stores the value in an attribute. The area is calculated in square map units, whatever they are.

The type of area calculation that is performed is chosen by the Type parameter.

### **Parameters**

#### Type

For the traditional projected area of features, use "Plane Area" as the type.

For 2D/2.5D geometries, the option of calculating 3D sloped areas is available by selecting "Sloped Area" as the type. This option calculates the surface area of the feature, taking height (Z) coordinates into consideration.

#### Area Attribute

The attribute that contains the area of the polygonal feature.

#### Multiplier

The multiplier parameter can be used to scale the area from being square ground units (the units of the feature's coordinates) to something else. This parameter may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

For 3D geometries, the surface area is always returned.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Area



## AreaFillColorSetter

Sets the fill color for the feature's area. Formats that support color will then render the interior feature in the set color.

**Note:** If the output format does not support fill color, this transformer will have no effect.

### Parameters

Area Color

You can edit this parameter by clicking the colored square to the right of the text field and choosing a color, or by typing values directly in the field.

The color must be specified as <red>,<green>,<blue> where each of <red>, <green>, and <blue> is a number between 0 and 1.

### Transformer Category

Infrastructure

### Related Transformers

NeighborColorSetter

RandomColorSetter

### Technical History

Associated FME function or factory: None

## AreaOnAreaOverlayer

Performs an area-on-area overlay so that all input areas are intersected against each other and resultant area features are created and output. The resultant areas have all the attributes of all the original features in which they are contained.

### Parameters

#### Group By

The default behavior is to use the entire set of features as the group. This option allows you to select attributes that define which groups to form.

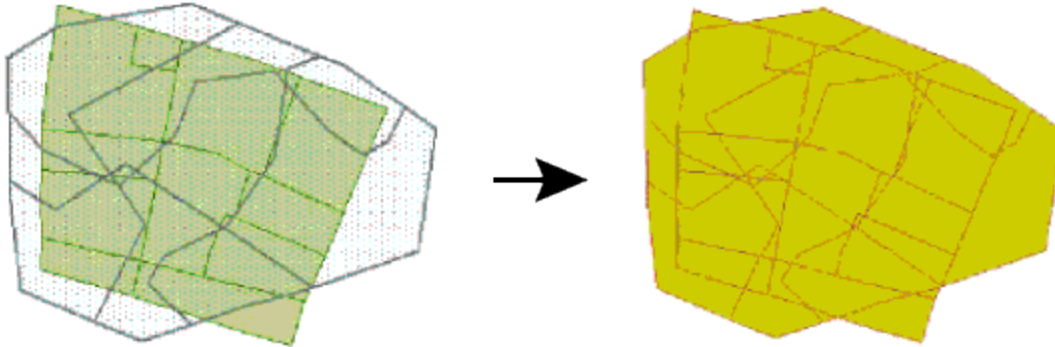
#### Overlap Count Attribute

The **Overlap Count Attribute** holds the number of features that the resultant feature overlapped, which will be at least one.

#### List Name

If a List Name is supplied, a list is created of all the attributes of each input area that overlaps the resultant feature.

### Example:



### Usage Notes

When attributes are merged between features, existing attributes are not replaced. Therefore if the polygons being overlaid have attributes with the same name, then the values will not be transferred from one to the other. You can work around this by renaming (**AttributeRenamer**), prefixing (**AttributePrefixer**), or removing (**AttributeRemover**) attributes to avoid name collisions.

### Geometry Handling

If the Geometry Handling Advanced setting is set to Enhanced in the workspace, ellipses accepted as areas are preserved as ellipses; otherwise, ellipses are stroked.

### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: OverlayFactory

## **AttributeAccumulator**

Combines feature attributes. One feature is output for each unique combination of values of the attributes specified in the Group By parameter. The output feature will have no geometry.

### **Parameters**

#### Group By

You can browse to select attributes from a list. If you do not choose any attributes, a single feature is output with all of the input attributes combined.

#### Input is Ordered by Group

If you choose Yes for this parameter, then the transformer will output all the aggregates it has accumulated so far, every time the values of the Group By attributes change from one feature to the next.

#### List Name

If an optional list name is supplied, a list is created that contains all the attributes of each feature that is combined to create the output feature.

### **Transformer Category**

Collectors

### **Technical History**

Associated FME function or factory: AggregateFactory, @RemoveGeometry

## AttributeClassifier

Tests if the contents of the source attribute are entirely of a particular character classification, and routes the feature accordingly.

### Parameters

Source Attribute

Choose the source attributes to classify.

Classification to Test

Empty strings are not considered part of any class and so will always fail.

The following character classes are available:

Character Class	Description
Alphanumeric	Any Unicode alphabet or digit character.
Alphabetic	Any Unicode alphabet character.
ASCII	Any character with a value less than \u0080 (those that are in the 7-bit ASCII range).
Boolean	The value matches <i>0</i> , <i>1</i> , <i>false</i> , <i>true</i> , <i>no</i> , or <i>yes</i> .
Control	Any Unicode control character.
Date	Any date in the form YYYYMMDD.
Digit	Any Unicode digit character. Note that this includes characters outside of the [0-9] range.
Double	A double precision number, which is: white space; a sign; a sequence of digits; a decimal point; a sequence of digits; the letter "e"; and a signed decimal exponent. Any of the fields may be omitted, except that the digits either before or after the decimal point must be present and if the "e" is present then it must be followed by the exponent number.
False	The value matches <i>0</i> , <i>false</i> , or <i>no</i> .
Graphical	Any Unicode printing character, except space.
Hexdigit	Any hexadecimal digit character ([0-9A-Fa-f]).
Integer	An integer number, defined as a collection of integer digits, optionally signed and optionally preceded by white space. If the first two characters of string are "0x" then string is expected to be in hexadecimal form; otherwise, if the first character of string is "0" then string is expected to be in octal form; otherwise, string is expected to be in decimal form.
Lowercase	Any Unicode lowercase alphabet character.
Not a Number	A floating point number equal to the special "not a number" value.
Printable	Any Unicode printing character, including space.
Punctuation	Any Unicode punctuation character.
Space	Any Unicode space character.
True	The value matches <i>1</i> , <i>true</i> , or <i>yes</i> .
Uppercase	Any uppercase alphabet character in the Unicode character set.
Wordchar	Any Unicode word character. That is any alphanumeric character, and any Unicode connector punctuation characters (e.g., underscore).

For an attribute to pass, all of its contents must belong to the specified classification.

### Transformer Category

Strings

## **Technical History**

Associated FME function or factory: OverlayFactory, TestFactory, @Tcl2 (string "is" function)

## AttributeCompressor

Compresses the values of the specified attributes.

The compressed attribute values can be decompressed using the AttributeDecompressor.

Use case scenario?

What is the output? Is it usable?

Do you always need both transformers in the same workspace?

### Input Port

- INPUT: Features with attributes.

### Output Port

- OUTPUT: List of compressed attributes.

### Parameters

Attributes to Compress

Click the browse button to select the attributes to compress.

### Usage Notes

The AttributeCompressor uses the zlib library to provide basic compression.

### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: @CompressAttributes

## AttributeCopier

Copies existing attributes to new attributes with the specified names. The existing attribute remains and a new attribute is created that has a different name, but the same value.

### Input Port

- INPUT: Features with attributes.

### Output Port

- OUTPUT: List of attributes, including any that have been copied. All copied attributes will initially appear at the bottom of the list.

### Parameters

Connect the transformer to one or more feature types and click the Transformer properties button. Two columns (*Old Attribute* and *New Attribute*) are displayed in the dialog.

### Attribute List

You can manually populate the properties dialog, or follow the steps in the example below.

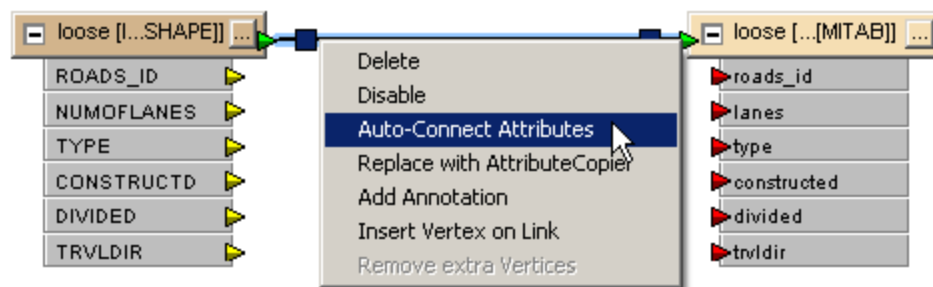
- Old Attribute: After you connect the transformer, feature type attributes will be visible in the pull-down menu. Choose the attribute to copy.
- New Attribute: Choose a new attribute name from the list, or type a new attribute name.

### Example

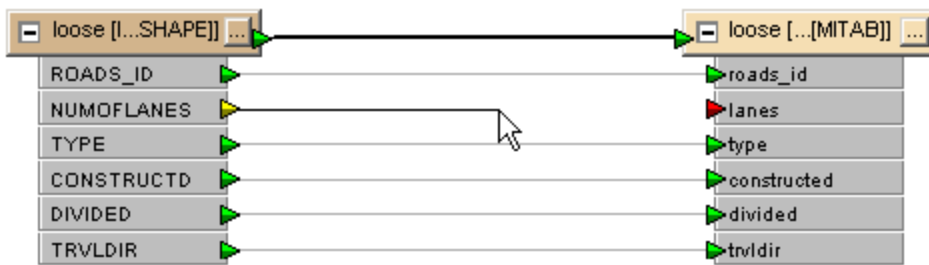
You can replace manual attribute connections with an AttributeCopier transformer. In this example, FME won't connect the attributes because either the case is different or the name is different.



If you right-click on the existing connection and choose Auto-Connect Attributes, FME will guess at the connections, choosing to connect the attributes that have lowercase names.

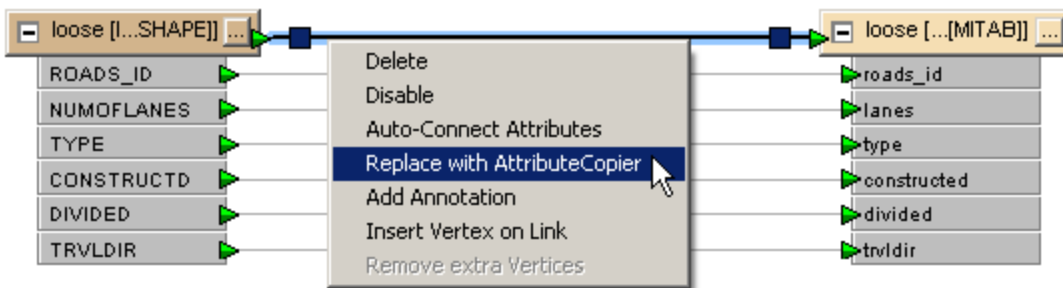


FME won't connect NUMOFLANES to lanes because the attribute names are different, so you will have to manually connect these attributes.

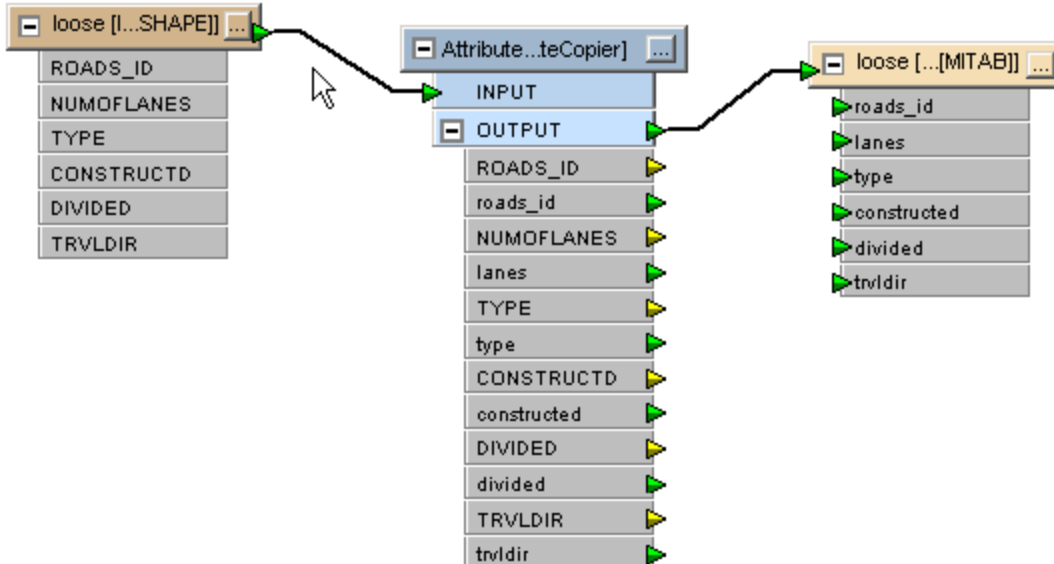


If these connections are somehow deleted, you will have to repeat these steps. However, if you insert an AttributeCopier to map the attributes, then the connections are saved.

Right-click on the connection and choose Replace with AttributeCopier:

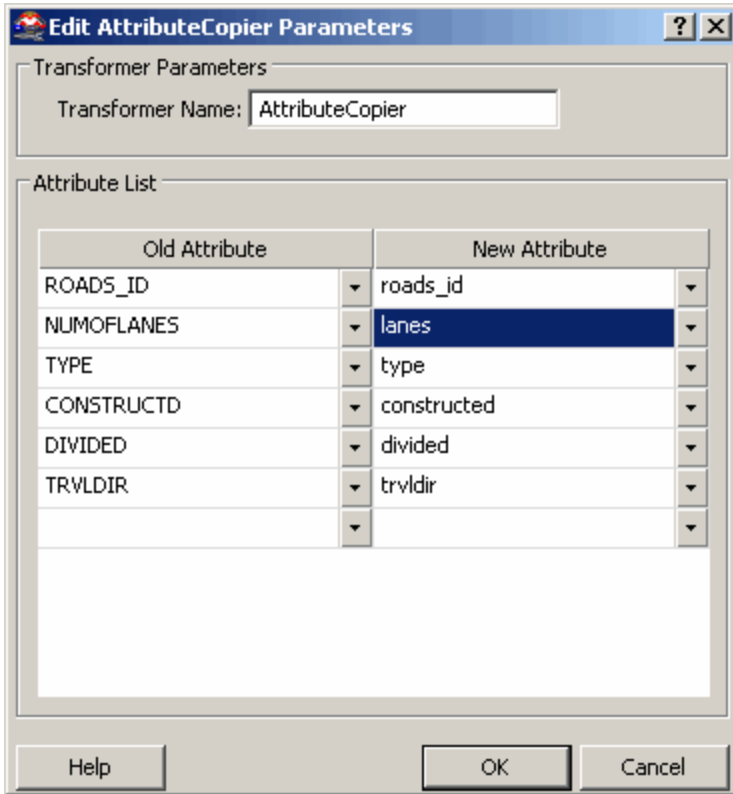


The transformer replaces the attribute connections. The new attribute names are in the list:



Click the properties button to see how the old attributes map to the new attributes.





**Transformer Category**

Infrastructure

**Technical History**

Associated FME function or factory: @CopyAttributes

## AttributeCreator

Adds a number of attributes to the feature, supplying them with constant values. Any feature that enters the transformer emerges with a new set of attributes as defined in the transformer's parameters dialog.

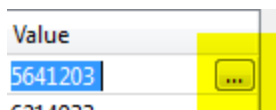
### Parameters

#### Attribute Name

Enter a new attribute name in each Attribute Name field

#### Value

Enter values associated with attribute names. You can also click the browse button in the cell to open a code editor:



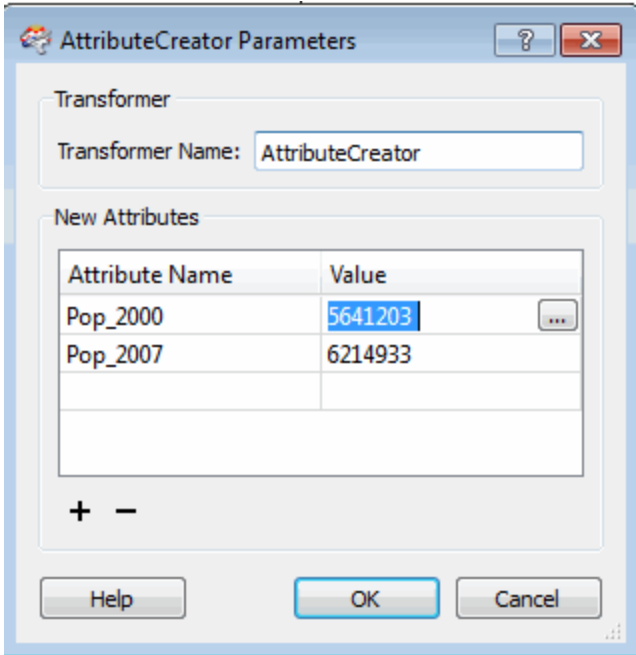
This editor is useful for entering multi-line attribute values, such as HTML or XML fragments.

The attribute value can also include any number of escape sequences, as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering *lan\es* will be interpreted as *lanes*.)

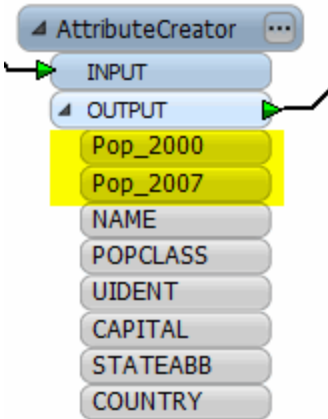
Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

### Example

In the following example, the AttributeCreator is used to add attributes and values to a dataset.



In Workbench, the new attributes are added to the transformer:



And if you output to an Inspector, the attributes and values appear in the Information pane:

NAME	Value
Pop_2000	5641203
Pop_2007	6214933

### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: @SupplyAttributes

## **AttributeDecompressor**

Decompresses the values of the specified attributes that were compressed by the AttributeCompressor.

Use case scenario?

What is the output?

Do you always need both transformers in the same workspace?

### **Input Port**

- INPUT: Features with attributes that were compressed by the AttributeCompressor.

### **Output Port**

- OUTPUT: List of decompressed attributes.

### **Parameters**

Attributes to Decompress

Click the browse button to select the attributes to Decompress.

### **Usage Notes**

The AttributeDecompressor uses the zlib library to provide basic compression.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @CompressAttributes

## AttributeDereferencer

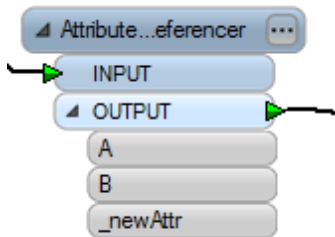
Copies the value of an attribute whose name is referenced in the source attribute. The value is added to a newly created attribute.

For example, say you have two attributes:

- Attribute A has the value **B**
- Attribute B has the value **999**

In the transformer parameters:

- Set the Source Attribute parameter to A.
- Create a new Destination Attribute called **newAttr**.



When you run the workspace, the value 999 (from Attribute B) is added to the destination attribute, as shown in the log file:

```
2010-11-27 18:13:38| 0.1| 0.0| INFORM| Attribute(string) : `A' has value `B'
2010-11-27 18:13:38| 0.1| 0.0| INFORM| Attribute(string) : `B' has value `999'
2010-11-27 18:13:38| 0.1| 0.0| INFORM| Attribute(string) : `_newAttr' has value `999'
```

## Transformer Category

Infrastructure

## Technical History

Associated FME function or factory: @CopyAttributes

## AttributeExploder

Creates a new pair of attributes (attribute name/attribute value) from each attribute on the input feature and either outputs these on a new feature or adds them as a list element to the original feature. In both cases, it is possible to either conserve or delete the original attributes and geometry.

### Parameters

#### Exploding Type

The original attributes and geometry are conserved or deleted depending on the settings for **Keep geometry** and **Keep attributes**.

When Exploding Type is set to **Features**, one feature is output for each original attribute. Each output feature will have two new attributes: one named Attribute Name Label (whose value will be the initial attribute's name) and one named Attribute Value Label (whose value will be the initial attribute's value).

When Exploding Type is set to **List**, the processed feature will have a new list attribute defined by List Name which will contain one entry (with attribute name and attribute value members, named by Attribute Name Label and Attribute Value Label) for each attribute of the feature. In order for Workbench to access these attributes (such as `_attr_list{0}._attr_name`), an AttributeExposer must be used.

#### Keep Geometry

If Keep Geometry is Yes, each output feature will have the same geometry as the given input feature. Otherwise, output features will have no geometry.

#### Keep Attributes

If Keep Attributes is Yes, the initial attributes on the processed features are preserved. Otherwise, output features will only have the attributes created by this factory.

### Example

Suppose we have an input feature with the following two attributes: .

- Name = John
- Type = Employee

If Keep Attributes is set to Yes, the parameter **Exploding Type: Feature** will produce two features with four attributes each:

*Name = John*

*Type = Employee*

*\_attr\_name = Name*

*\_attr\_value = John*

*Name = John*

*Type = Employee*

*\_attr\_name = Type*

*\_attr\_value = Employee*

The parameter **Exploding Type: List** will add an attribute list to the original feature:

*Name = John*

*Type = Employee*

*\_attr\_list{0}.\_attr\_name = Name*

*\_attr\_list{0}.\_attr\_value = John*

*\_attr\_list{1}.\_attr\_name = Type*

*\_attr\_list{1}.\_attr\_value = Employee*

### **Related Transformers**

AttributeExposer

### **FME Licensing Level**

FME Professional edition and above

### **Transformer Category**

Strings

### **Technical History**

Associated FME function or factory: AttributeFactory

## **AttributeExposer**

Exposes a series of hidden attributes, so they can be used by other transformers.

This transformer can be useful when dealing with transformers, such as the GMLFeatureReplacer or SchemaMapper, that dynamically add attributes onto features.

It is also useful if you know that features have hidden attributes (FME Attributes and the more obscure Format Attributes) that aren't shown in Workbench. You can use the AttributeExposer to "expose" these attributes.

This transformer is a more dynamic version of the Expose Attributes option in the context-menu of an input feature type.

Once the attributes are exposed, they can be used by other transformers.

The input features are not modified in any way.

## **Parameters**

Attributes to Expose

After you place the transformer, you can type attribute names or choose them from the drop-down list. The order in which they appear in the list is the order in which they will appear in the Output port of the transformer.

## **Transformer Category**

Infrastructure



## **AttributeExpressionRemover**

Removes all attributes on incoming features that match a given regular expression. This transformer can be used to remove large numbers of attributes that have common naming.

See the [StringSearcher](#) for more information on regular expressions.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @RemoveAttributes

## **AttributeFileReader**

Reads the contents of a file and stores them as the value for the specified attribute.

The filename can be taken either from the value of an attribute or from a fixed constant name. If the filename is a constant, then the same file will be read for each feature which passes through the transformer.

Note that if necessary, full pathnames can be created from existing attribute values by using the `StringConcatenator` to prepend directory names and/or append extensions prior to using this transformer. The resulting attribute can then be used to specify the source file.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @File

## **AttributeFileWriter**

Writes the contents of the specified attribute to a file.

If the file already exists, it will be overwritten.

The filename can be taken either from the value of an attribute or from a fixed constant name. If the filename is a constant, then the same file will be overwritten for each feature that enters the transformer, and only the results of the feature will be in the resultant file at the end. For that reason, if more than one feature will pass through the transformer, the filename should be taken from an attribute value which holds the name of the file to write.

Note that if necessary, full pathnames can be created from existing attribute values by using the Concatenator to prepend directory names and/or append extensions prior to using this transformer. The resulting attribute can then be used to specify the target file.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @File

## AttributeFilter

Routes features to different output ports depending on the value of an attribute. The set of possible attribute values can be entered manually, or extracted from an input source in the properties dialog.

### Input

- INPUT: The feature type that contains the attributes you want to filter.

### Output

- BLANK: If the feature's attribute has no value, the feature is output via the BLANK port.
- UNFILTERED: If the feature's attribute has a value not in the list, the feature is output via the UNFILTERED port.

### Parameters

Attribute to Filter By

When you connect the transformer to the feature type, the list of attributes will appear in a pull-down list. Choose the attribute from the list.

Possible Attribute Values

If you know the possible attribute values, you can type them here. If not, you can import them from a source dataset:

- Click the Import button to start the import wizard, and click Change Dataset.
- Choose the source dataset. Click Next.
- Choose the desired feature types. Click Next.
- Select an attribute to be scanned for values. Click Next.
- FME will scan the dataset, and the processing summary will appear in the wizard. For example:

Processing Summary	
Features scanned:	2661
Unique attribute values found:	3

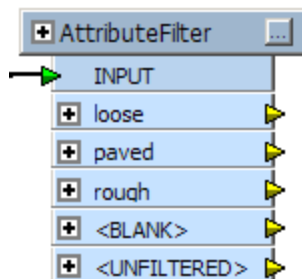
- Click Import to add the values to the AttributeFilter. The list will appear in the transformer properties dialog.

### Usage Notes

Ideally, you will want to keep the filter list fairly small; otherwise you can end up with a very long list of attributes. If FME determines that the list might be too large, it will return a warning.

### Example

You have a feature type called ROADS, and you know you have an attribute value called TYPE. You want to filter by TYPE (in this example, loose, paved, or rough). When you set up the transformer properties and click OK, you will see that the AttributeFilter now has new attributes that correspond to the filter settings:



The transformer will filter and, after you run the workspace, output the separate attributes.

### **Transformer Category**

Filters

### **Technical History**

Associated FME function or factory: @Lookup

## **AttributeKeeper**

Removes all attributes from the feature, except the ones that are selected from the attribute list.

This can be useful when features have large numbers of unnecessary attributes. You can clean up the features so that they only retain the attributes you want to work with.

### **Parameters**

Attributes to Keep

Click the browse button to select from the attribute list.

### **Usage Notes**

If the number of attributes you want to keep is greater than the number of attributes you want to remove, try the AttributeRemover.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @KeepAttributes

## **AttributePrefixer**

Adds a prefix or suffix to all attributes coming in to the transformer. This transformer is useful if you need to quickly rename all your attributes. It can be used before other transformers that may overwrite incoming attributes.

## **Transformer Category**

Manipulators

## **Technical History**

Associated FME function or factory: @Tcl2

## AttributeRangeFilter

Performs a lookup on a range-based lookup table, and routes the feature to the appropriate output port. If the input attribute does not match any of the given ranges, the feature is routed to the <UNFILTERED> port.

### Parameters

It's important to note that the ranges you create are inclusive and that the order in which they appear on the table matters.

Each range has a From value and a To value. These values are inclusive, and each feature will be matched to the first range (in the order present in the table) that contains the value of that feature's Source Attribute. If either From or To is omitted, the range is open-ended; that is, it will match any value greater than (for From) or less than (for To) what is specified.

For example:

From To Output Port

0 A

0 10 B

10 20 C

30 D

The following values will cause features to come out the given output port:

value  $\leq$  0 port A

0 < value  $\leq$  10 port B

10 < value  $\leq$  20 port C

20 < value < 30 port <UNFILTERED>

30  $\leq$  value port D

No reverse mapping is available.

### Transformer Category

Filters

### Related Transformers

AttributeValueMapper

AttributeRangeMapper

### Technical History

FME Function Used: @Lookup



## AttributeRangeMapper

Performs a lookup on a range-based lookup table and stores the resulting value, or writes the value to, a new output attribute. This transformer lets you use an input attribute to classify features by numeric ranges.

### Parameters

Note: The ranges you create are inclusive and the order in which they appear on the table matters. The ranges are tested starting with the first one in the list and progressing to the last range listed.

#### Input Attribute

The Input Attribute value is the attribute whose value is matched in the range lookup table.

The list of available input attributes are those that are exposed to this transformer, which depends on the transformers connected to it.

This is the attribute you want to classify. Open the list and select the input attribute you want to use for classifying ranges.

#### Output Attribute

The RangeMapper transformer writes to an attribute. This is the name of the new attribute that's created as the result of the classification.

#### From

Specify the start value for the range that you are classifying. This field accepts numerical input only and must be less than the value in the To field.

If a From value is not specified, an open-ended range is specified.

#### To

Specify the end value for the range that you are classifying. This field accepts numerical input only and must be greater than the value in the From field.

If a To value is not specified, an open-ended range is specified.

#### Output Value

Enter the name you want to give this range of values.

#### Default Value

If an attribute's value does not match any input ranges set in the table, the output class assigned is specified by the Default Value.

### Example

From	To	Output Value
	0	A
0	10	B
10	20	C
30		D

Default Value: DEFAULT

The following input values will cause the given output value:

Value is less than or equal to 0: D

value is less than zero and less than or equal to

value	$\leq 0$	A
-------	----------	---

0 <	value <=10	B
10 <	value < 20	C
20 <	value <30	DEFAULT
30 <=	value	D

### How to Quickly Generate Range Table Values

When you know the total number of values in a range, and you know how many groupings you want, this is a quick way to generate the range table values. Click Generate button in the Parameters dialog.

### Transformer Category

Manipulators

### Related Transformers

AttributeValueMapper

AttributeRangeFilter

### Technical History

FME Function Used: @Lookup

## **AttributeRemover**

Removes the selected attributes from the feature.

This is normally not necessary unless the feature will later be processed by a transformer that merges attributes onto it from other features. In such a case, if the other features have attributes with the same names as this feature, the attributes must first be removed for the merge to occur.

### **Parameters**

Attributes to Remove

Click the browse button to select from the attribute list.

### **Usage Notes**

The AttributeKeeper offers similar functionality, but instead of selecting the attributes you want to remove, you select the attributes you want to keep.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @RemoveAttributes

## AttributeRenamer

Renames selected attributes.

### Input Port

- INPUT: Features with attributes.

### Output Port

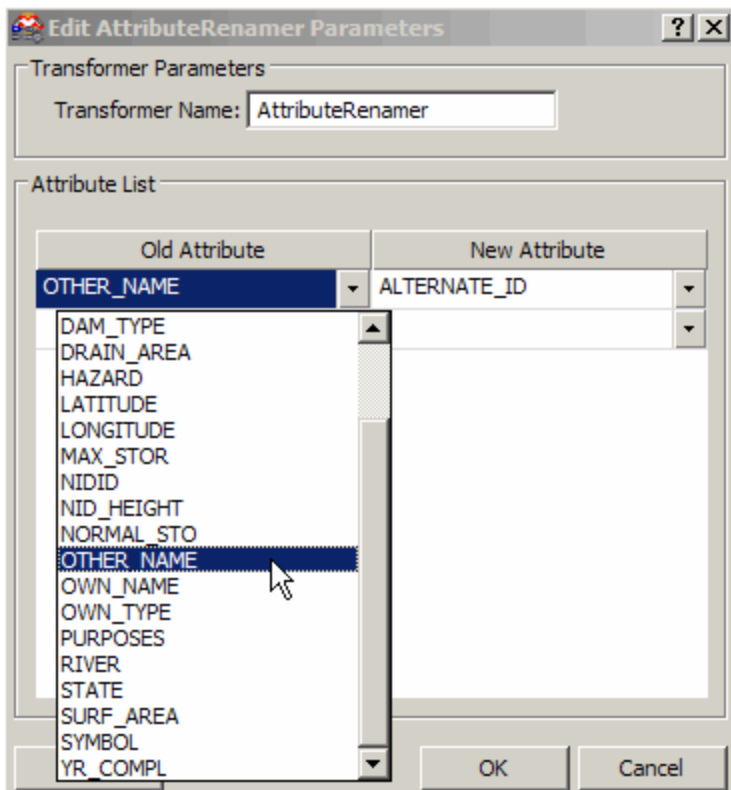
- OUTPUT: List of attributes, including any that have been renamed. All renamed attributes will initially appear at the bottom of the list.

### Parameters

Connect the transformer to one or more feature types and click the Transformer Properties button. Two columns (*Old Attribute* and *New Attribute*) are displayed in the Parameters dialog.

Use the pull-down list in the *Old Attribute* column to choose the attribute that you want to rename .

Type the new name in the *New Attribute* column, or select it from the pull-down list.



### Usage Notes

You cannot use the AttributeRenamer to rename multiple reader attributes with different names to a single writer attribute. Only the last reader attribute would actually be renamed. The transformer does not act as a filter to see if the reader attribute already exists.

14. If an attribute doesn't exist, then the new attribute is given a NULL or blank value.

### Transformer Category

Manipulators

### Related Transformers

AreaOnAreaOverlayer

LineOnAreaOverlayer

LineOnLineOverlayer

PointOnAreaOverlayer

PointOnLineOverlayer

PointOnPointOverlayer

SpatialRelator

**Technical History**

Associated FME function or factory: RenameAttributes

## **AttributeReprojector**

Reprojects attributes from one coordinate system to another.

### **Parameters**

X and Y Attributes

Choose which X and Y attributes to reproject.

Source and Destination Coordinate System

Choose from recently accessed coordinate systems, or click the browse button to access the Coordinate System Gallery.

### **Usage Notes**

This transformer does not alter the feature's coordinates – only the values of the selected X and Y attributes (if they contain coordinate values) are changed.

### **Transformer Category**

Coordinate Systems

### **Related Transformers**

GtransReprojector

### **Technical History**

Associated FME function or factory: @Reproject

## **AttributeRounder**

Rounds off attributes to the specified number of decimal places.

### **Parameters**

Source Attributes

Select the source attribute to round.

Decimal Places

Enter a number or take the value of a feature attribute by selecting the attribute name from the pull-down list.

Round-off Direction

This parameter controls how the rounding will take place.

- nearest: the number is rounded up or down to nearest value
- up: the number will always be rounded up
- down: the number will always be rounded down

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Tcl2

## AttributeSetter

Sets an existing attribute to a constant value, or to the value of an another attribute.

### Parameters

#### Attribute

Choose an attribute from the pull-down list, or enter a new attribute name.

Note: If you specify a new attribute name, it will not appear in the feature's schema. You can use an `AttributeExposer` to make the attribute visible.

#### Value

You can use a value from an existing attribute, by choosing the attribute from the pull-down list, or you can type a value in the field.

This parameter can also include any number of escape sequences, as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering `lan\es` will be interpreted as `lanes`.)

Sequence	Description
<code>\a</code>	Audible alert (bell) (0x07)
<code>\b</code>	Backspace (0x08)
<code>\f</code>	Form feed (0x0c)
<code>\n</code>	Newline (0x0a)
<code>\r</code>	Carriage return (0x0d)
<code>\t</code>	Tab (0x09)
<code>\v</code>	Vertical tab (0x0b)
<code>\\</code>	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: @Tcl2



## AttributeSplitter

Splits a selected attribute into a list attribute. Each item in the list will contain a single token split from the list.

You would use this transformer, for example, to separate an attribute that has a comma-separated value list into its component pieces.

### Parameters

#### Attribute to Split

After you connect this transformer, choose an attribute from the pull-down menu.

#### Delimiter or Format String

The delimiter character can be expressed as a single character, or it may be a special character sequence beginning with a backslash ("\"). Special character sequences are interpreted as shown below. If the sequence is not listed in the table, the backslash character is ignored. (For example, \e will be interpreted as e.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

Alternatively, instead of using a delimiter character you can provide a string in the format #s#s#s, where each number is the length of the substring you wish to extract.

#### Trim Whitespace

By default, the resulting strings have both leading and trailing (left and right) whitespace removed but you can change the setting to either left or right, or none.

#### List Name

The default list name is \_list, but you can change it to a name that is more specific to your workflow.

### Usage Notes: Accessing Individual List Elements

Lists are usually indicated in Workbench by name, followed by a pair of curly brackets

```
mylist{}
```

A specific list element contains its number between the brackets:

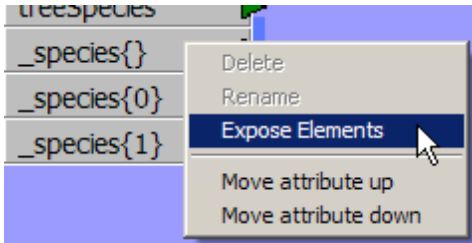
```
mylist{0}
```

Attributes of a list element are shown like this:

```
mylist{0}.myattribute
```

However, list elements generated by the AttributeSplitter contain values but no attributes.

To access specific list elements in Workbench, right-click on a list name in the attribute list and choose Expose Elements.



In the dialog that appears, type the number of elements you want to expose.

Keep in mind that list elements start counting at 0, so exposing 1 element of `mylist{}` will result in the extra attribute `mylist{0}`.

## Examples

### *Using a delimiter to split myattr into mylist:*

```
if myattr = A,B,C and the delimiter is ,
```

then the result would be:

```
mylist{0} = A
```

```
mylist{1} = B
```

```
mylist{2} = C
```

### *Using a format string to split myattr into mylist:*

```
if myattr = 20030210 and the format string is 4s2s2s
```

then the result would be:

```
mylist{0} = 2003
```

```
mylist{1} = 02
```

```
mylist{2} = 10
```

## fmepedia

See fmepedia for additional information about this transformer.

## Related Transformers

You can also use the `StringSearcher` to split apart the values of attributes using regular expression pattern matching.

## Transformer Category

Strings

## Technical History

Associated FME function or factory: `@Split`

## AttributeTrimmer

Removes leading and trailing trim characters from selected attributes.

### Parameters

Attributes to Trim

After you connect this transformer, click the Browse button and choose from the list of attributes.

Trim Type

Characters may be trimmed from the right, left, or from both sides.

Trim Characters

Trim Characters can also include any number of escape sequences, as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering *lan\es* will be interpreted as *lanes*.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

If you do not specify a trim character, then white space is removed (spaces, tabs, newlines, and carriage returns).

Allow Trim Down to Nothing

If a given attribute value contains only Trim Characters, the character that remains depends on the Trim Type: for Both it will be the middle character, for Left it will be the last character, and for Right it will be the first character.

If the Allow Trim Down to Nothing option is set to No, the output of the trimming will have at least one character remaining.

### Transformer Category

Strings

### Technical History

Associated FME function or factory: @Tcl2

## AttributeValueMapper

Looks up and assigns attribute values based on other attributes, and stores the looked-up value in a new attribute. This transformer does not modify feature types.

### Parameters

#### Source Attribute

After you place and connect this transformer, choose the attribute from the drop-down menu.

#### New Attribute Name

The attribute that will store the looked-up value. You can use the default, or type a different name.

#### Default Value

The value to be looked up in the named lookup table. The transformer will return the found value, or will be blank if none was found.

The value can include any number of escape sequences, as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering *lan\es* will be interpreted as *lanes*.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

Any feature that does not map to a listed value is given the default value instead. A blank default value means that unmapped features are given a null value and the translation will continue. (Note that in FME versions earlier than 2009, this behavior is different.)

### Entering a KEY Default Value

You can also enter a default value using the word *KEY* (uppercase). If, for example, you have the mappings:

a : 1

b : 2

c : 3

and a feature has an attribute value of "d", it won't exist in the ValueMapper.

However, if you have entered a default value of *KEY-0*, the unknown attribute value will be internally mapped to *KEY-0*. And because *KEY* is replaced by the original value (d), the transformer will return the value "d-0".

### Reverse Mapping

If checked, the direction of the lookup is reversed. That is, it will perform a lookup from replacement values to source values if operating in the forward direction, or from source value to replacement value if executing in the inverse direction.

### Import

Original attribute values can also be imported from any FME-supported source dataset.

The most common use would be to import from the same dataset as is being mapped, but you can also import the mappings from a lookup table stored in a text, CSV or Excel file, or another dataset. Click the Import button. A wizard will step you through the import procedure:

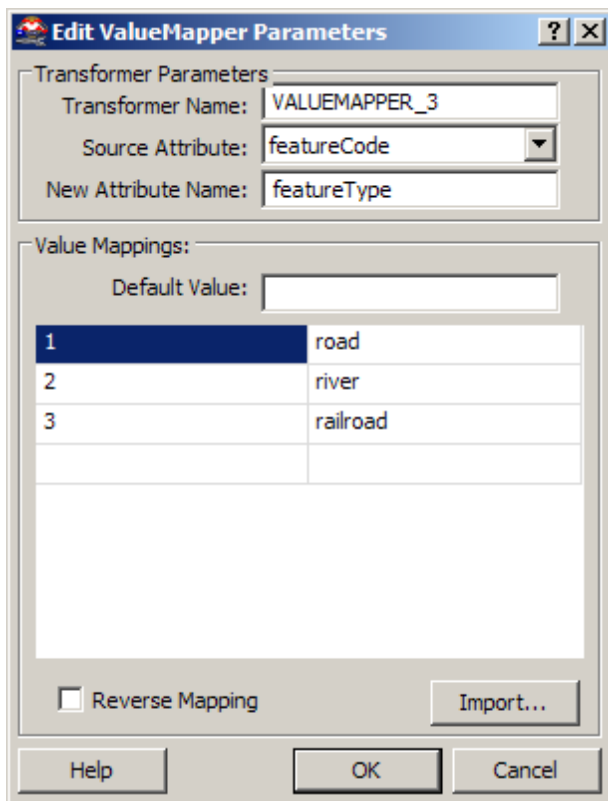
1. Select the dataset to be read. Click Next.
2. A list of feature types will appear. The features in the dataset will be scanned, and the attribute values found will be added to the attribute list in the ValueMapper. Click Next.
3. Choose the key attribute. The values of this attribute should match the attribute on features that are entering the ValueMapper. Click Next.
4. Choose the value attribute. The values of this attribute will be added to the ValueMapper. Click Next.
5. The status will show how many features were scanned, and the overall number of unique attribute values found. Click Finish to include the values in the table.

Where only one side of the mapping is available for import - for example, the source values are in the source data, but the mapped values are not - then it's recommended that you select the same source attribute for both sides of the equation. That will give you a basic 1:1 mapping which you can then edit. If different values are chosen, then the mapping may not be 1:1. Having said this, the ValueMapper can be purposely used to map several source values to a single output value.

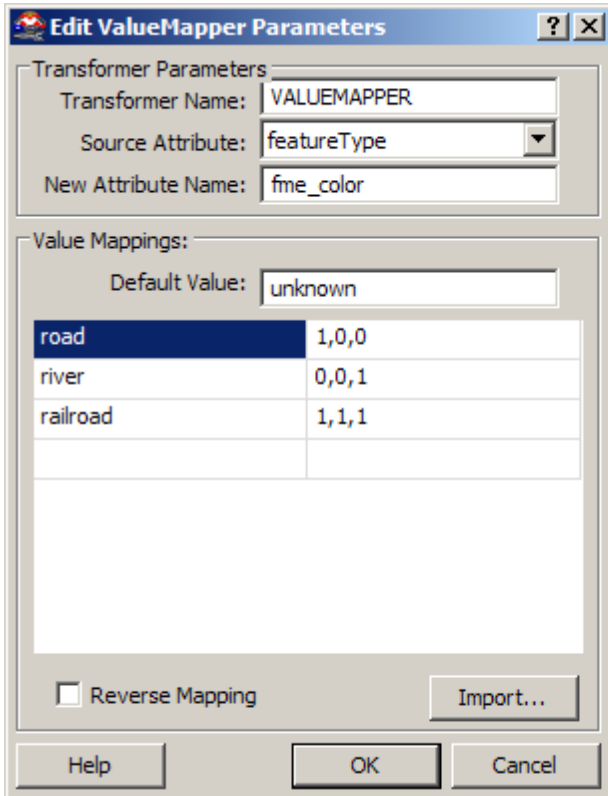
### Example

You could use the ValueMapper to map `fme_feature_type` against `fme_color` to assign a different color to each type of feature.

In this example, the ValueMapper is used to map an existing feature code to the road, river, and railroad feature types, and output a new featureType attribute.



A second ValueMapper maps the featureType attribute to a new attribute `fme_color`. The format of this attribute is `r,g,b` where each of `r`, `g`, and `b` is a number between 0 and 1.



Performs a lookup of the value of an attribute in a lookup table, and stores the looked-up value in a new attribute.

If the value cannot be found in the lookup table, the default value will be stored instead. If the default value has the special name KEY (all upper case), then the original source attribute's value will be stored in the new attribute.

FME Function Used: @Lookup

## fmepedia

See fmepedia for additional information about this transformer.

## Transformer Category

Strings

## Transformer History

This transformer was previously named the ValueMapper.

## Technical History

Associated FME function or factory: @Lookup

## BaseConverter

Converts an attribute's value from one numeric system (base) to another, putting the resulting value in a new attribute. The base is the number of unique digits, including zero, that a positional numeral system uses to represent numbers.

The most common base values are 2 (binary), 10 (decimal) and 16 (hexadecimal), and 8 (octal). The BaseConverter supports bases from 2 (binary) to 36 (hexatridecimal / sexatrigesimal).

### Parameters

#### Source Attribute

Choose the source attribute that contains the base to be converted.

#### Original Base

Enter the base of the value to be converted.

#### Destination Base

Enter the base to which the value is to be converted.

#### Output Width

The output width parameter specifies the total width of the output attribute. The value after conversion will be padded on the left with zeroes to make up the necessary width. If the width is set to 0, no padding is done. If the value is wider than the width before padding, it is not touched.

#### Destination Attribute

The name of the attribute that will contain the results, and will appear in the list of OUTPUT port attributes.

### Example

What if you want to convert a Microstation color to a KML color?

- Expose the `igds_color.red`, `igds_color.green` and `igds_color.blue` attributes.
- Add three BaseConverter transformers.
- Edit the parameters so that each transformer converts one of the color values from base 10 to base 16.
- Use a StringConcatenator to combine the three color strings.

### Usage Notes

- This transformer supports only unsigned whole numbers. It does not handle fractions or decimal points.
- This transformer supports very large numbers.
- Digits are chosen from the set below. Any lowercase digits are automatically converted to uppercase.

0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ

### Transformer Category

Calculators

### Technical History

Associated FME function or factory: @ConvertBase

## BoundingBoxAccumulator

Takes a set of point, linear, polygonal, and/or aggregate features, and creates a two-dimensional bounding box, which contains all features. The bounding box is defined as the minimum enclosing rectangle for all input features. The minimum rectangle is such that all sides of the rectangle are parallel to the x axis and the y axis.

The input features may be partitioned into groups based on attribute values using the Group By parameter, and one bounding box feature is output for each group. If the Group By parameter is not specified, then all input features will be processed together and a single bounding box will be output. If a given bounding box has zero area, it will become a line or a point.

### Group By

One bounding box feature is output for each unique combination of values of the attributes specified in Group By parameter. The bounding box is the smallest rectangle that contains all the features that were members of the group. If Group By attributes are not specified, a single feature is output that represents the bounding box of all the features.

### Calculate Topfer Index

The Topfer index is a rule of thumb that dictates the number of source and destination features a generalized map should maintain. This parameter defaults to No. If you choose Yes, the Topfer Index Attribute and Source/Destination Scale parameters are enabled.

### Topfer Index Attribute

If you set the Calculate Topfer Index parameter to Yes, an attribute containing the Topfer index will be attached to both the BOUNDING\_BOX and ORIGINAL output features.

### Source and Destination Scale

The following formulas are used to calculate the scale factors of the source and destination data:

$$N\text{-destination} = N\text{-source} * ((S\text{-source}/S\text{-destination}) ** 0.5)$$

where:

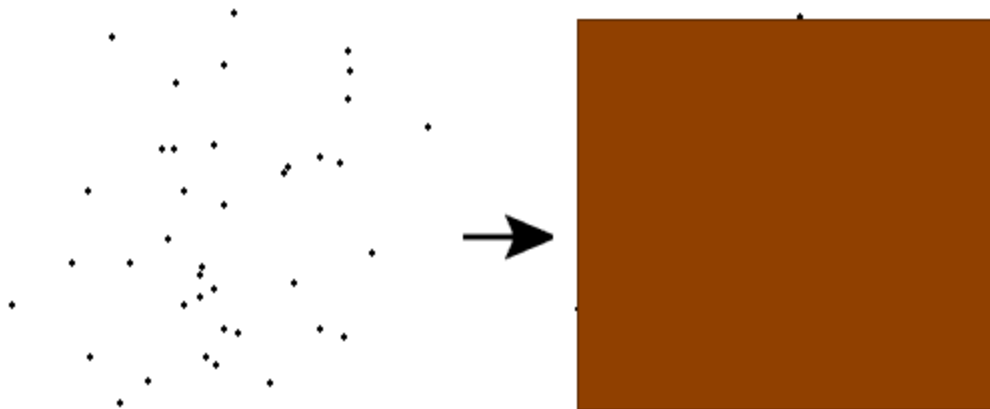
N-destination = number of features output

N-source = number of features in input

S-source = scale of source map

S-destination = scale of destination map

## Example



## Transformer Category

Collectors



### **Related Transformers**

To retrieve the bounds of a feature into attributes, use the BoundsExtractor.

To replace a feature with its bounding box, use the BoundingBoxReplacer.

### **Technical History**

Associated FME function or factory: BoundingBoxFactory

## BoundingBoxReplacer

Replaces the geometry of the feature with either its two-dimensional bounding box or its two-dimensional minimum oriented bounding box.

If a feature's bounding box has zero area, it will become a line or a point. If the feature was originally 3D, it will be set to 2D upon leaving, and the z values will be ignored and dropped.

### Parameters

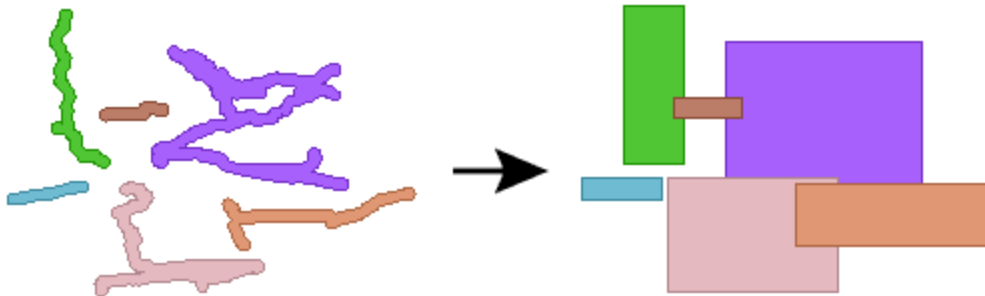
Replace With

Choose Bounding Box (which means the axis-aligned or rectilinear bounding box) or Oriented Bounding Box

Length of Shorter and Longer Side Attribute

Optionally, you can calculate the lengths of the shorter and longer sides of the chosen bounding box. They will appear as attributes in the BOX output port.

### Example



### Related Transformers

- To accumulate the bounding box of a number of features, use the BoundingBoxAccumulator.
- To retrieve the bounds of a feature into attributes, use the BoundsExtractor.
- To replace the geometry of a feature with a box whose values come from attributes, use the 2DBoxReplacer.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Bounds, @XValue, @YValue, @GeometryType

## BoundsExtractor

Extracts the minimum and maximum values of the feature's coordinates into new attributes.

It determines the extreme values of the feature in each of the x and y (and optionally z) axes, and assigns these values to the resulting attribute names.

This transformer does not change the geometry of the original feature.

### Parameters

#### Minimum and Maximum X, Y, Z Attributes

The names of the attributes to be assigned the minimum and maximum values of the feature along the X, Y, and Z axes. If the feature has only two dimensions, the Z Attributes will be set to 0. All of these parameters are optional.

### Example

As each feature enters the transformer, it has six new attributes added to it to hold the extents of the feature.

If a feature had the coordinates (1,10,100), (2,-20,150), then after leaving the transformer, it would have these new attributes with these values:

Attribute	Value
_xmin	1
_xmax	2
_ymin	-20
_ymax	10
_zmin	100
_zmax	150

### Transformer Category

Calculators

### Related Transformers

- To accumulate the bounding box of a number of features, use the `BoundingBoxAccumulator`.
- To replace a feature with its bounding box, use the `BoundingBoxReplacer`.

### Technical History

Associated FME function or factory: @Bounds

## Bufferer

Replaces the geometry of a feature with one that represents the original, padded by a specified width. If the specified width is too great for a buffered feature to be created, a null feature is output.

Each point in the output geometry will be the specified distance (specified by a Buffer Amount parameter, measured in ground units) away from the original geometry.

## Parameters

### Group By

If any attributes were specified in this parameter and any regular buffering style (CAP\_ROUND, CAP\_BUTT, CAP\_PROJECTING) is selected, then all features with the same values for the specified Group By attributes are dissolved together after to create non-overlapping areas.

### Buffer Amount

Enter a buffer amount (floating-point value) or select a feature attribute from the pull-down list.

Please note the correlation between Buffer Amount and Buffer Style below: a regular buffering style can accept both a negative and a positive buffer amount; a side buffering style can accept only a positive buffer amount.

### Interpolation Angle

This parameter, measured in degrees, controls the smoothness of curves in the resulting buffer. Vertices approximating the curve will be generated at this interval. If the value 0.0 is specified, no curve approximations are made and corners are beveled. You can also use the Interpolation Angle from a selected attribute.

### Buffer Style

There are two main buffering styles: regular buffering and side buffering.

#### **Regular Buffering: CAP\_ROUND, CAP\_BUTT, CAP\_PROJECTING**

These styles apply to all geometries, and always output area features (that is, polygon or donut features). The cap style describes the type of ending for input line features. If the input feature is an area, the particular cap style is ignored.

If you specify a Buffer Amount of 0, then the output features will not grow or shrink in size; all geometries of this feature are dissolved together. For example, a feature containing an aggregate geometry, using this transformer with a width of 0.0, will generate a dissolved version of the feature.

If you specify a negative Buffer Amount, then the output features will be shrunken. Please note that when specifying a negative buffer amount for input line features, the output will contain a null geometry.

The different types of end cap styles are described and illustrated below. Note that the Buffer Amount in all cases is .25.

- CAP\_ROUND: This style rounds line endings by a stroked arc.



- CAP\_BUTT: This style makes line endings flush to the end of the line, that is, the end of the line touches the end of the buffer.



- CAP\_PROJECTING: This style creates square line endings. The buffer extends beyond the line ending by Buffer Width.



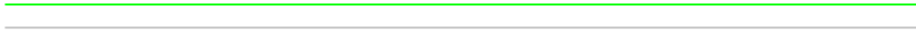
#### **Side Buffering: LEFT\_SIDE\_ONLY, RIGHT\_SIDE\_ONLY, SIDES\_ONLY**

These styles apply to line features only – features with geometries that do not contain lines will generate no output geometries. Unlike regular buffering, side buffering styles can only accept positive Buffer Amount values.

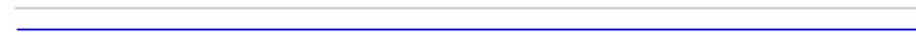
Side buffering styles will only generate valid output geometries if each input line feature does not self-intersect.

The different types of side buffering are described and illustrated below. Note that the Buffer Amount in all cases is .25.

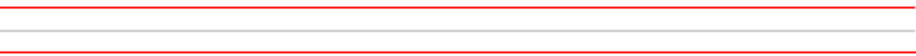
- LEFT\_SIDE\_ONLY: This style generates a line representing the left side of a regular buffer.



- RIGHT\_SIDE\_ONLY: This style generates a line representing the right side of a regular buffer.



- SIDES\_ONLY: This style generates lines representing both the left and right sides of a regular buffer.



## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will be stroked before being buffered; they are otherwise buffered as points located at their respective center points.

## fmepedia

See fmepedia for more information on this transformer.

## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: @Buffer2, DissolveFactory

**Note: This transformer uses functionality from the GEOS library (<http://geos.refractions.net/>), which is distributed under the terms of the Free Software Foundation's LGPL license (<http://www.gnu.org/licenses/lgpl.html>). To comply with the terms of the LGPL, Safe Software Inc. has set up a website (<http://www.safe.com/foss>) to provide further information.**

## **CaseChanger**

Changes the case of text attributes to UPPERCASE, lowercase, Title case, or Full Title Case.

### **Parameters**

Attribute(s) to Change

After connecting the CaseChanger in your workspace, click the Browse button. Use the checklist to choose which attributes to change, and then click the OK button.

Case Change

- UPPERCASE changes attributes to uppercase characters.
- lowercase changes attributes to lowercase characters.
- Title case changes the first character in the string to its Unicode title case variant (or to uppercase if there is no title case variant) and the rest of the string lowercase.
- Full Title Case converts the first letter of each word, rather than just the first letter in the string. Full Title Case will ignore parentheses if they start the string or follow whitespace, and will treat hyphens (-) as whitespace characters.

### **Transformer Category**

Strings

### **Technical History**

Associated FME function or factory: @RenameAttributes

## CenterLineReplacer

Replaces an area feature with its medial axis, straight skeleton, or a centerline. This transformer works best with long, narrow areas.

### Parameters

Mode

**Medial Axis:** The geometry of an area feature is replaced by its medial axis. The medial axis is the subset of the straight skeleton that does not include any edges that share a vertex with the original area.

**Straight Skeleton:** The geometry of an area feature is replaced by its straight skeleton (angular bisector network). All edges that share a vertex with the original area are removed.

Note: In either of these modes, the algorithm may take a long time to run on large input features. The efficiency is given as  $O(nm + n \log m)$  where  $n$  is the total number of vertices, and  $m$  is the number of "reflex vertices" – vertices that cause the polygon to be non-convex.

**Classic:** The geometry of an area feature is replaced with a line that threads through the center of the feature. This mode only works well with long, thin features. This mode requires the Line Tolerance parameter.

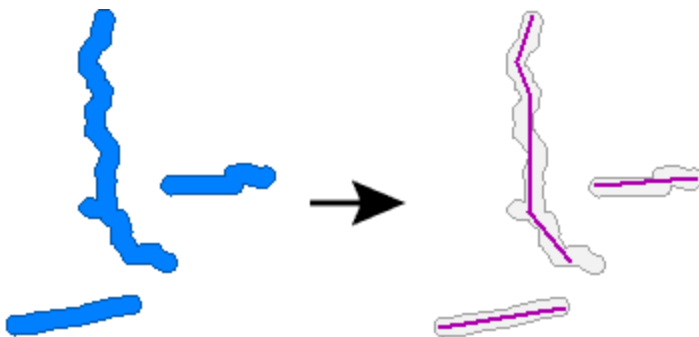
Line Tolerance

Used with Classic mode only, this parameter (measured in ground units) defines the point density on the resulting line. Lines generated have points no closer than this distance apart. It may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Usage Notes

- Unexpected output may result if input polygons self-intersect or have duplicate vertices.
- Because Z coordinates are not considered in either algorithm, all features processed by this function are forced to 2D.
- If non-area features are passed to this transformer, they will not be changed, and they are logged with a warning.

### Example



### Transformer Category

Manipulators

### Related Transformers

When combined with the Tester, this transformer enables FME to perform area generalization operations.

### Technical History

Associated FME function or factory: @ConvertToLine

***The algorithm used to calculate the angular bisector network for the Medial Axis or Straight Skeleton mode was designed by Petr Felkel and Stepan Obdrzalek and is provided as-is, with no warranties. The paper outlining this algorithm is referenced in Straight Skeleton Implementation, 1998, "SCCG 98: Proceedings of the 14th Spring Conference on Computer Graphics," pages 210-218, ISBN 80-223-0837-4 (Publisher: Comenius University, Bratislava).***

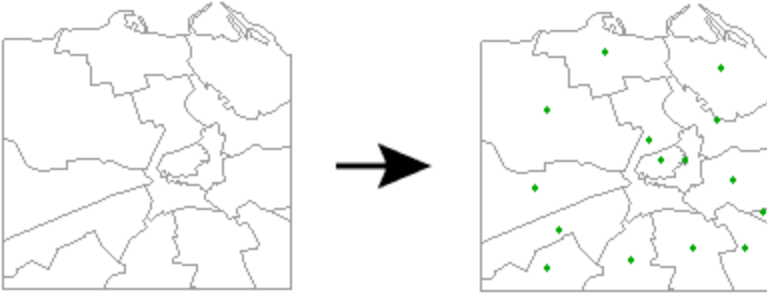


## CenterOfGravityReplacer

Replaces the geometry of the feature with a point at the center of mass of the feature. Note that, depending on the feature's shape, the resulting point may be far outside of the original feature.

### Example

The transformer calculates the exact distribution of portions of a shape; therefore, some points can actually appear outside a feature.



## Transformer Category

Manipulators

### Related Transformers

- To generate points guaranteed to be inside an area feature, use the LabelPointReplacer transformer.
- To generate points at the center of the bounding box of the feature, use the CenterPointReplacer transformer.

### External References

- Center of mass, from Wikipedia

### Technical History

Associated FME function or factory: @ConvertToPoint

## **CenterPointReplacer**

Replaces the geometry of the feature with a point that is in the center of the feature's bounding box.

### **Usage Notes**

Note that, depending on the feature's shape, the resulting point may be far outside of the original feature.

### **Transformer Category**

Manipulators

### **Related Transformers**

- To generate points guaranteed to be inside an area feature, use either the `LabelPointReplacer` or `InsidePointReplacer` transformer.
- To generate the center of gravity, use the `CenterOfGravityReplacer`.

### **Technical History**

Associated FME function or factory: `@ConvertToPoint`

## ChangeDetector

Detects changes between two sets of input features.

This transformer is often used with multiple readers, to identify changed features in the two files. It can identify all features that two input files have in common, and those which are in one file and not the other, such as the additions and the deletions.

### Input

- ORIGINAL: One set of features enters the transformer via the ORIGINAL port.
- REVISED: Another set of features enters via the REVISED port.

### Output

- UNCHANGED: An ORIGINAL feature is output via the UNCHANGED port when it is found to have either matching geometry, matching attribute values, or both, with a feature in the REVISED set.
- ADDED: A REVISED feature with no match in the ORIGINAL set is output via the ADDED port.
- DELETED: An ORIGINAL feature is output via the DELETED port when no match for it can be found in the REVISED set.

### Parameters

#### Match Geometry

This parameter controls whether 2D or 3D (or None) geometry must be the same before a match is declared.

FULL makes sure 3D, measures, and Geometry Attributes all match.

When comparing raster geometries: 2D matches the properties, 3D matches the properties and values, and FULL matches the properties, values and geometry traits.

When comparing surface and solid geometries: 2D behaves the same way as 3D, that is, z values will also be compared.

#### Lenient Geometry Matching

If Lenient Geometry Matching is set to Yes, then the order of points in area features will be ignored. Composition differences between paths and lines will be ignored. True arcs and ellipses versus their stroked polygon equivalents will be ignored in Aggregates, Polygons, Donuts, Paths, and all other multis.

This transformer does not support surfaces or solids in the input if this parameter is set to Yes.

#### Attributes to Match

Choose the attributes that must have changed before a match is declared.

#### Match all Attributes Except

Choose the specific attributes to exclude.

#### Match All Attributes

Choose Yes to compare all attributes.

#### Treat Blank Value as Different From Missing Attribute

If this parameter is set to Yes, then an attribute will be considered different for two features when one feature contains the attribute with a blank value and the other feature does not contain the attribute at all.

#### Extra Vertex Tolerance

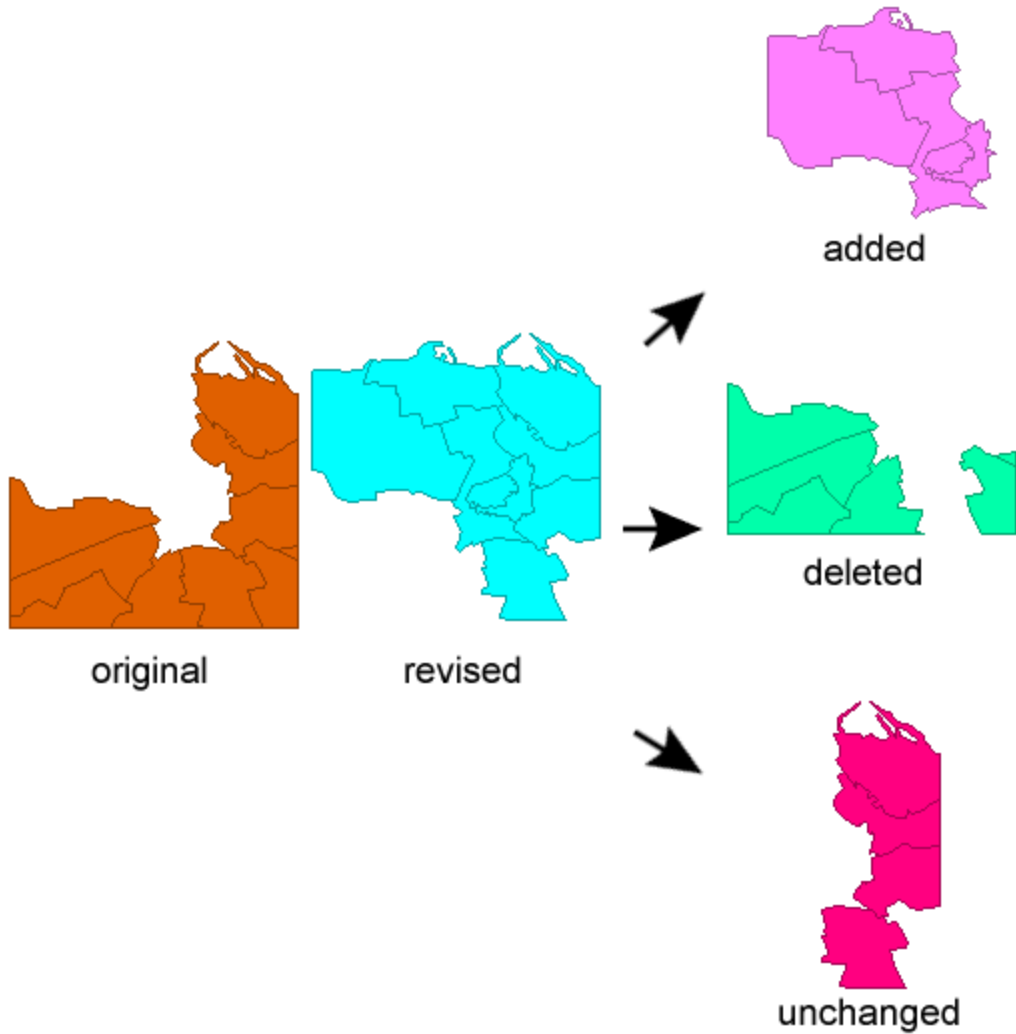
Note: This parameter has no effect when using lenient geometry matching.

When geometry is being matched, Extra Vertex Tolerance can allow for extra vertices along line segments.

A value of 0 means that no such extra vertices are permitted. A nonzero value controls how close any extra vertices must be to the line connecting the adjoining matching vertices.

For surface and solid geometries, Extra Vertex Tolerance is ignored and assumed to be 0.0.

### Example



### Related Transformers

The Matcher provides a more general approach, which may be more convenient for certain applications.

### Transformer Category

Filters

### Technical History

Associated FME function or factory: MatchingFactory

## **CharCodeExtractor**

Extracts the integral character code of the first character in the source string attribute, and adds its integer value in the character set to the feature as another attribute.

This can be used to obtain the ASCII code of any character, including non-printable ones.

### **Parameters**

Source String Attribute

The attribute that contains the character for which you would like to find the equivalent numeric code

Character Code Attribute

The name of the attribute that will store the resulting integer value.

### **Example**

For example, a control-A character as the first character of the source string attribute would result in a value of 1 being placed in the character code attribute.

### **Usage Notes**

To obtain the character equivalent of a numeric value, use the CharacterEncoder transformer.

### **Transformer Category**

Strings

### **Technical History**

Associated FME function or factory: @Tcl2

## **CharacterEncoder**

Sets the result attribute to the character whose numeric code was contained in the source code attribute (or the entered integer).

This can be used when the numeric ASCII code of any character is available, but the actual ASCII character is desired. For example, if the source attribute had the value 1, a control-A character would be placed in the resulting string.

### **Parameters**

Source Code

The numeric ASCII code of any character.

Character String Result Attribute

The name of the attribute that will store the resulting value.

### **Transformer Category**

Strings

### **Related Transformers**

To obtain the numeric code equivalent of a character, use the `CharacterCodeExtractor` transformer.

### **Technical History**

Associated FME function or factory: @Tcl2

## Chopper

Ensures that all features output have less than or equal to the specified maximum number of vertices.

This is useful if you're outputting to a format that has limitations on the number of points in lines or areas. It can also be used to crudely simplify complex objects.

All new features have the same attributes as the original feature had and are output via the CHOPPED port.

### Output Ports

- CHOPPED: Features created by chopping the original feature into smaller pieces are output here. Each feature contains less than or equal to the maximum number of vertices, but has all of the original feature's attributes.
- UNTOUCHED: All input features with less than or equal to the maximum number of vertices are output untouched here.

### Parameters

#### Maximum Vertices

Each new feature will have the specified number of vertices, except for the last one which may have fewer than the maximum.

All new features have the same attributes as the original feature had and are output via the CHOPPED port.

If this parameter is less than 4, all area features are converted to linear features and processed as such. If it is set to 1, area and line features are turned into point features.

### Usage Notes

For linear features, if the feature has more than this number of vertices, it is chopped into several smaller features. Linear features are simply broken into linestrings, so that if they were drawn atop the original feature, they would completely cover it.

When an area feature is chopped, the result will still be a number of closed areas. For area features, if the feature has more than this number of vertices, it is chopped into several smaller features with vertical and horizontal cuts. In most cases, each new feature will have at most the maximum number of vertices specified, where possible. (For some very odd Area shapes with a very small number of Maximum Vertices, this cannot always be accomplished.)

Area features are chopped in such a way as to preserve the original area of the feature. That is, the original area feature is cut into smaller areas, which if dissolved together, would match the original area. New vertices may be introduced into the area feature as it is chopped.

### Example



### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: ChoppingFactory

## **CircularityCalculator**

Calculates the "circularity" of an area feature, which indicates how elongated the feature is.

A value of 1 is returned if the feature is a circle, and a value of 0 is returned if the feature is a line.

### **Parameters**

Calculation Geometry

If the calculation geometry used is specified as CONVEX\_HULL, then the convex hull of the feature is used for the calculation; otherwise, the geometry of the feature (FEATURE\_BOUNDARY) is used.

The value returned is:  $4 * \text{PI} * \text{area} / (\text{perimeter} * \text{perimeter})$

Circularity Attribute

The name of the attribute that will contain the angularity calculation.

### **Transformer Category**

Calculators

### **fmepedia**

See fmepedia for more information on this transformer.

### **Technical History**

Associated FME function or factory: @Circularity



## ClassicSQLExecutor

Executes an arbitrary SQL statement against a database.

A wizard is used to set the connection type, SQL query, list name, and attributes.

Note: The fields that the wizard displays may depend on your database connection. You might not see all of the information listed below.

### Specify External Database

Supported databases include ODBC, Oracle, Microsoft Access, Microsoft SQL Server, PostgreSQL/PostGIS, DB2, MySQL, SQLite, and Microsoft Excel.

Note that when you connect to Microsoft SQL Server, there is also an option to check the **Use Windows Authentication** checkbox, which enables Windows Authentication for the database. If this option is selected, the Username and Password fields are ignored.

### SQL Query

Enter the SQL statement to be executed on the data source. Ctrl + Tab will enter a "tab" character.

Multiple SQL commands can be delimited by a character specified using the **FME\_SQL\_DELIMITER** keyword, embedded at the very beginning of the SQL block. The single character immediately following this keyword will be used to split the SQL which will then be sent to the database for execution. (Note: Include a space before the character.)

An individual statement may be preceded with a hyphen, indicating that errors should be ignored.

Example:

```
FME_SQL_DELIMITER ;  
SELECT * FROM TABLEA;  
SELECT * FROM TABLEB;
```

If the SQL statement resulted in a row (or rows) being returned, the attributes from the row are added to the feature. If multiple rows are returned, the attributes from those rows are held in the list specified by the <list name> parameter. The individual rows may be then exposed to be operated on directly, manipulated by any of the list processing transformers, or broken out into separate features by using the List-Exploder transformer.

Note that in either case, if the attributes retrieved from the SQL statement are to be used in later transformers, they must be manually identified in the transformers' properties. This is because the SQLExecutor does not know what attributes it will be retrieving from the database until the SQL statement is actually executed at run time.

Some databases require that quotation marks be escaped. For mixed registration table names, or Oracle tables with spaces, you must include backslashes to run a query. For example: `select * from \"MixedRegister\"`

### List Name (optional)

Enter a list name that will store the attributes.

### Add Extra Attributes

Enter any attributes you would like to add to the transformer.

Note that attribute names are case-sensitive.

### Timeout (Applies only to Microsoft SQL Server)

This parameter may only be set through Workbench's Navigator pane, since it does not appear in the transformer's wizard interface.

### Transformer Category

Database

## Technical History

Associated FME function or factory: @SQL

## Clipper

Performs a geometric clipping operation.

This transformer takes a number of clip boundaries (clippers) and a number of features to be clipped (clippees). The output is split in four groups.

### Input

- **CLIPPER** : The features routed into the transformer via the CLIPPER port identify the area against which all CLIPPEE features are processed. The CLIPPER can be any area features (polygons, donuts, or aggregate polygons/donuts). Any non-area clipping features that are encountered will be logged with a warning and discarded.
- **CLIPPEE**: Features to be clipped enter via the CLIPPEE port.

### Output

- **INSIDE**: CLIPPEE features that are completely within the CLIPPER .
- **CLIPPED\_INSIDE**: CLIPPEE features that intersect the CLIPPER feature are broken up into pieces. Those pieces that are on the inside of the CLIPPER are output via the CLIPPED\_INSIDE port. Those portions outside the clipping area are output via the CLIPPED\_OUTSIDE port. When a feature repeatedly enters and leaves the CLIPPER area, aggregates will be created of the parts that are INSIDE and OUTSIDE, and will be output via the appropriate port.
- **CLIPPED\_INSIDE (raster)**: For raster CLIPPEE features, each CLIPPER that intersects the CLIPPEE will produce output through the CLIPPED\_INSIDE port. If *Preserve Clippee Extents* is set to No, the extents of CLIPPED\_INSIDE rasters will be equal to the bounding box of the portion of the CLIPPER within the CLIPPEE and the number of rows and columns will be reduced. Otherwise, the resolution and extents of the CLIPPED\_INSIDE rasters will be identical to the those of the CLIPPEE rasters. Additionally, the remaining areas of the CLIPPEE that are disjoint from all CLIPPERS will be output through the CLIPPED\_OUTSIDE port. The resolution and extents of CLIPPED\_OUTSIDE rasters will be identical to those of the CLIPPEE rasters.
- **OUTSIDE**: CLIPPEE features completely outside of the CLIPPER are output via the OUTSIDE port.
- **INSIDE or OUTSIDE**: Point CLIPPEE features that are on the CLIPPER boundary will be output via the inside or outside port depending on the setting of the *Clippees on Clipper Boundary* parameter. Linear CLIPPEE features that are completely collinear with the CLIPPER boundary are output via the inside or outside port depending on the setting of the *Clippees on Clipper Boundary* parameter. If a linear CLIPPEE feature has a portion that is collinear with the boundary, the feature will only be broken when it crosses the CLIPPER boundary.

### Parameters

#### Group By

If Group By attributes are selected, features are only clipped by features with the same values in the group by attributes.

#### Clipper Type

- **Single Clipper**: Only a single CLIPPER feature will be used.
- **Multiple Clippers**: All CLIPPER features will be used.
- **Clippers First**: The Clipper assumes that all CLIPPER features will enter the transformer before any CLIPPEE features.

Any further CLIPPER features that arrive after the first CLIPPEE will be logged with a warning and discarded.

Merge Attributes, Merge Attribute Prefix

If the Merge Attributes parameter is enabled, any CLIPPEE that is clipped will receive the attributes from the feature that clips it. In such a case, the Merge Attribute Prefix parameter can be used to give the CLIPPER's attributes a prefix before they are added to the CLIPPEE.

### Clippees on Clipper Boundary

This parameter directs what action should be taken with clippee features that lie entirely on the boundary of the clipper.

- Treat as Inside: These features that lie on the boundary are output via the INSIDE tagged output clause.
- Treat as Outside: These features that lie on the boundary are output via the OUTSIDE tagged output clause.
- Treat as Inside and Outside: Points and line segments on the boundary are duplicated and output as both INSIDE and OUTSIDE. The default behavior is INSIDE.

### Create Aggregates

If the Create Aggregates parameter is No, then features that are clipped into multiple parts will not be aggregated, but rather each part will be output as a separate feature.

### Preserve Clippee Extents

If this parameter is set to No, the extents of CLIPPED\_INSIDE rasters will be equal to the bounding box of the portion of the CLIPPER within the CLIPPEE and the number of rows and columns will be reduced. Otherwise, the resolution and extents of the CLIPPED\_INSIDE rasters will be identical to the those of the CLIPPEE rasters.

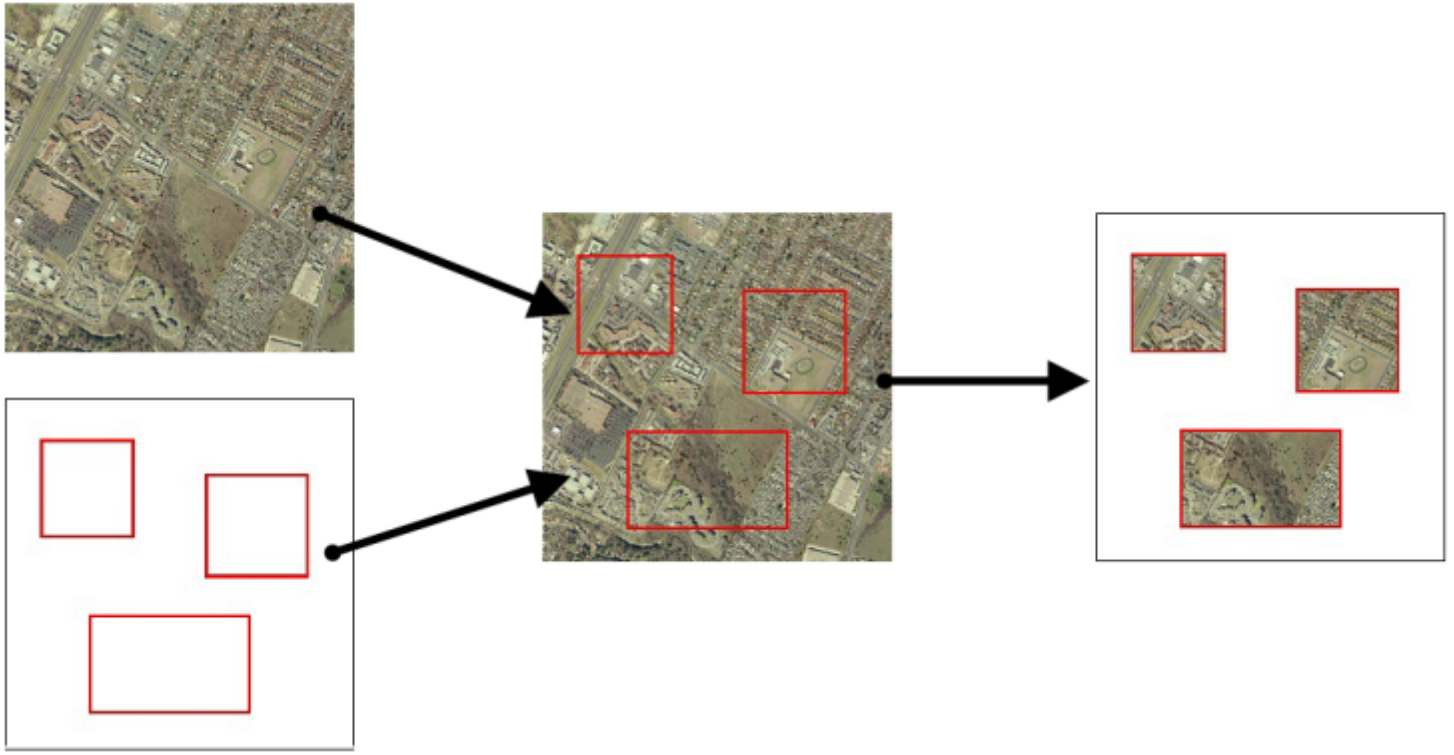
### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will be clipped without stroking. For example, the result of clipping an ellipse in half will result in an arc and a line connecting the two ends of the arc. Otherwise, all arcs and ellipses will be stroked prior to clipping.

### Usage Notes

- This transformer works with both raster and vector data.
- This transformer is unaffected by raster band and palette selection.

## Example



### Transformer Category

Geometric Operators

### fmepedia

See fmepedia for additional information about this transformer.

### Technical History

Associated FME function or factory: ClippingFactory, RasterClippingFactory

## Cloner

Makes the specified number of copies of the input features and outputs all copies through its single output port.

Feature cloning happens automatically in Workbench whenever more than one connection is made from any output port of a transformer. This transformer can be used to make a dynamic number of copies of a particular feature.

### Parameters

#### Number of Copies

Specifies how many copies will be made of the input features. The number of copies may be a constant integer, or it can be taken from the value of an attribute on the feature by selecting the attribute from the pull-down list.

If the number of copies is less than 1, then the feature will be deleted.

If the number of copies is set to 1, then the transformer acts as a "null" operation and is useful to provide a place to connect several inputs together and join them into single stream of features to be routed elsewhere.

#### Copy Number Attribute (optional)

The copy number attribute (by default, `_copynum`) will hold the copy number of each output feature. For example, if 3 copies of an input feature are made, the output copies will have 0, 1, and 2, respectively, assigned to their copy number attribute.

### Usage Notes

This transformer can also be used to add one or more constant attributes to a feature:

- Right-click the OUTPUT port and select Add Attribute.
- From the Workbench menu bar, choose Insert > Constant.

You can use the `AttributeCreator` to accomplish the same result.

### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: `TeeFactory`

## **CommonSegmentFinder**

Category: Collectors

Tests to see which of the **CANDIDATE** features have even one segment in common with any **BASE** feature.

- If a **CANDIDATE** feature does share one segment with some **BASE** feature, then it is output through the **OVERLAP** port.
- If a **CANDIDATE** does not share any segment with any **BASE**, then it is output through the **NO\_OVERLAP** port.

The **BASE** features are consumed by the transformer and are not output.

### **Technical History**

Associated FME function or factory: CommonSegmentFactory

## ContourGenerator

Generates contours from the underlying surface which is defined by the input POINTS, 3D\_LINES, and BREAKLINES.

- POINTS/LINES: The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model. This port also accepts raster features as input.
- BREAKLINES: These features are put into the model as breaklines such that triangle edges will always be along the line of the feature.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, DEMGenerator, ContourGenerator, and SurfaceModeller transformers.

- 3D\_LINES: The vertices of the 3D line are inserted into the model to define the surface. They are not used to define breaklines but rather just provide 3D data for the model from their vertices.

## Output

- CONTOURS: If specified, then the model is output as a set of contours. Each feature is 2D or 3D depending on the value specified for CONTOUR\_DIMENSION. Each output feature also has an attribute *SurfaceModel.elevation* that holds the elevation of the contour. If the output feature is 3D, then this value is the same as the z coordinate on each feature.

## Parameters

### Surface Tolerance

The surface tolerance is used when building the surface to determine which vertices to add to the surface model. Specifying a value of 0 turns off the vertex filtering so that all vertices (except those with duplicate x,y) are added to the model.

When specified, the surface tolerance is used during the construction phase of the surface model to determine whether a vertex will be added to the model, or whether it will be discarded and not added to the model.

The surface tolerance works as follows, for each vertex that is being added to the model:

- If the x,y location is outside the convex hull of the existing model, it is added to the model.
- If the x,y location is inside the existing model:
  - The difference between the z value from the existing TIN and the z value of the vertex is calculated.
  - This difference is compared to the surface model tolerance.
  - The vertex is only added to the model if the difference is greater than the surface tolerance; otherwise, the vertex is discarded.

### Output Contour Dimension

The dimension of the generated contour lines can be 2 or 3.

### Output Contour Interval

This parameter sets the z spacing between contours.

### Perturb Contours

This parameter controls whether contours can be perturbed in the z direction. The perturbed amount is 1% of the contour interval.



Yes: Features whose z-values are full multiples of the contour interval will be preserved. The z-values of these features will be perturbed so they no longer are full multiples of the contour interval.

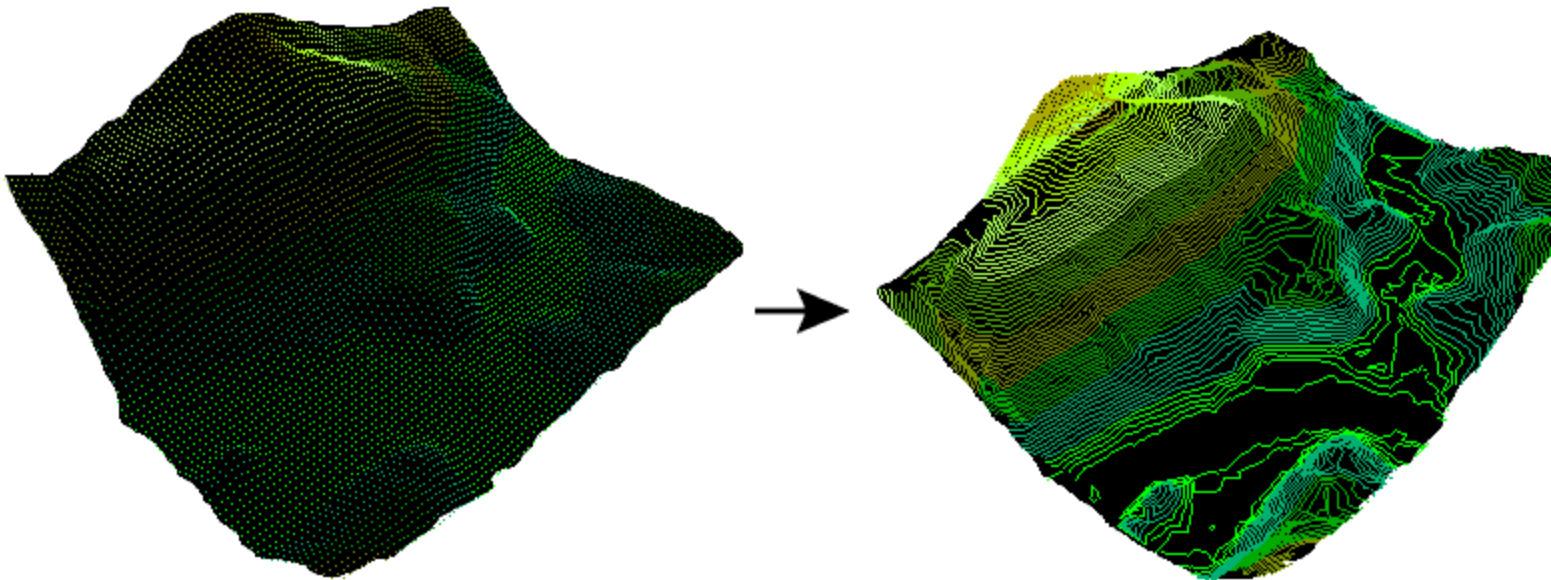
No (default): Features whose z-values are full multiples of the contour interval will be dropped.

Note that by default, input features with z-values that are full multiples of the contour interval will be skipped. For example, setting a contour interval of .5 or 1 means that no features will be used if their z-values are all whole numbers; for example, 123.0, 0, 567, 987.0, etc.

### Elevation Attribute

The Elevation Attribute specified will be added to the output contours and will hold their z value.

### Example



### FME Licensing Level

FME Professional edition and above

### Transformer Category

Surfaces

### Technical History

Associated FME function or factory: SurfaceModelFactory

## ConvexHullAccumulator

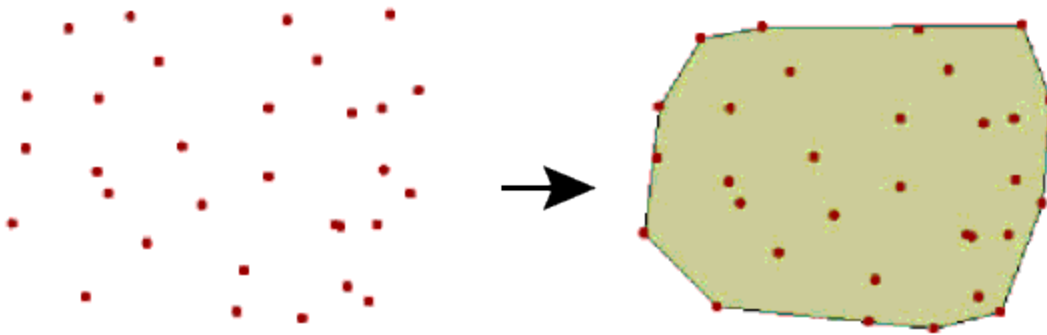
Creates convex hulls for groups of features. One convex hull feature is output for each unique combination of values of the attributes specified in the Group By parameter.

The convex hull is the smallest polygon where no interior angle is greater than 180 degrees that contains all the features that were members of the group. In lay terms, the effect is similar to tightening a rubber band around the features. Note that the hull may be a line or a point if the resulting polygon has an area of zero.

If no Group By attributes are specified, a single feature representing the convex hull of all the features is output.

Note: If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, all arcs, ellipses, and text will be stroked prior to the computation of the convex hull; otherwise, the center point of these geometries will be used during the computation of the convex hull.

### Example



### Transformer Category

Collectors

### Technical History

Associated FME function or factory: ConvexHullFactory

## ConvexHullReplacer

Note: This transformer was previously named ConvexHullCreator.

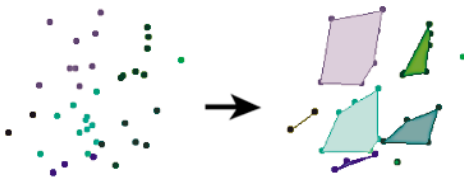
Category: Manipulators

Replaces the geometry of the feature with a polygon representing its convex hull. The convex hull is defined as the minimum enclosing convex polygon. A convex polygon is a polygon where no interior angle is greater than 180 degrees. In lay terms, the effect is similar to tightening a rubber band around the feature.

If the feature was an aggregate feature, a single convex hull is computed from all vertices of all geometries in the aggregate. If the feature was a linear feature, it will be turned into a polygonal feature. No change is made to a point feature.

Note: If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, arcs and ellipses will be stroked prior to the calculation of the convex hull; they are otherwise regarded as points located at their respective center points.

### Example



### Technical History

Associated FME function or factory: ConvexHullFactory

## ConvexityFilter

Determines whether areas, surfaces, and solids are convex or concave. A polygon is simple when it is not self-intersecting and has a non-zero area. Simple polygons are convex if every internal angle is less than or equal to 180 degrees. All other polygons are considered concave.

### Input

- INPUT: Typically these are polygon-based geometries.

### Output Ports

If the geometry of the feature is polygon-based (such as a polygon, a donut, or a box) all of the geometry's composing polygons are tested for convexity. Based on the results of this testing, the appropriate output port is determined.

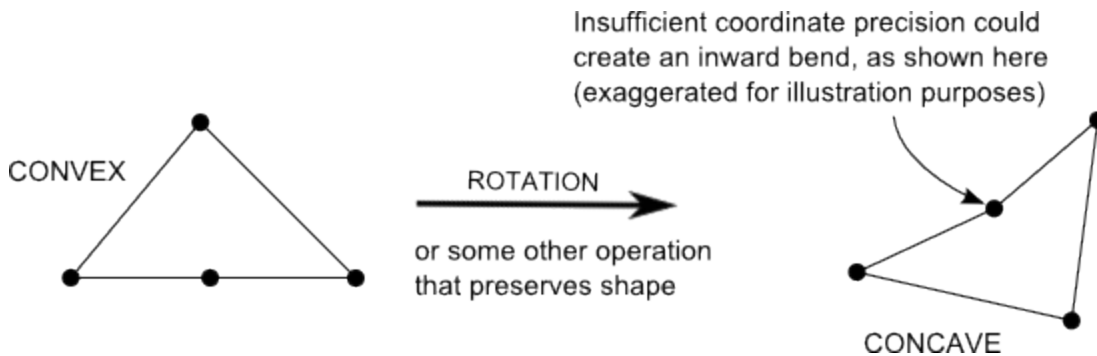
- CONVEX: If all composing polygons for the geometry of a feature are convex, the feature is written to the CONVEX output port.
- CONCAVE: If any composing polygons for the geometry of a feature are not convex, those features are written to the CONCAVE output port.
- UNDEFINED: Features with no geometry or with geometries that are not polygon-based, such as arc, raster, and text, are written to the UNDEFINED port.

### Parameters

There are no parameters for this transformer.

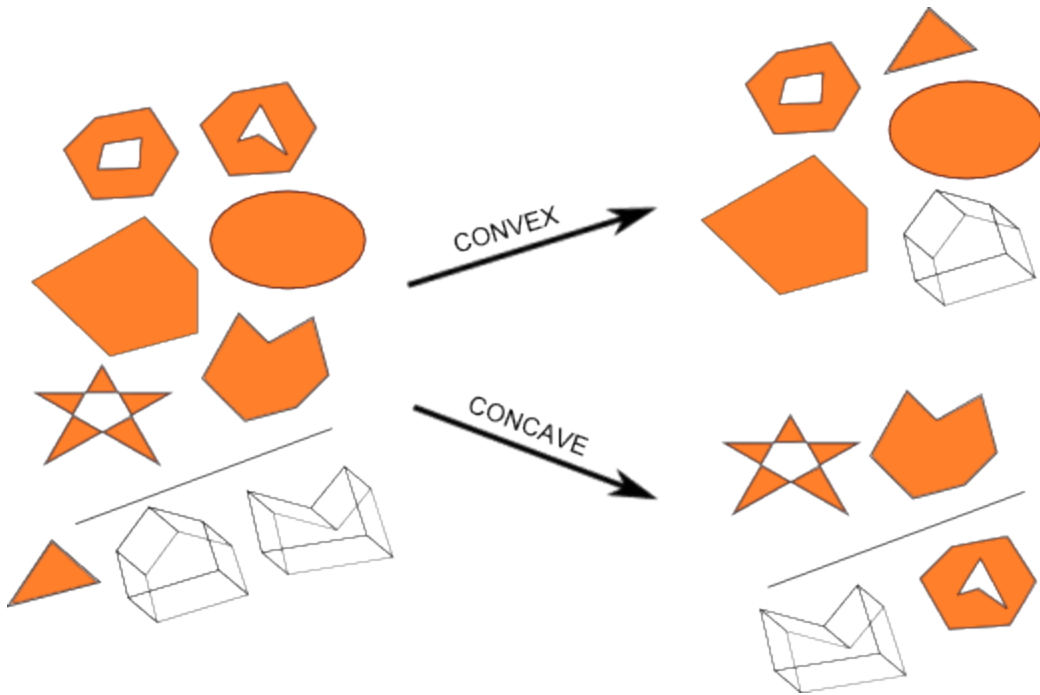
### Usage Notes

- Lines and points are considered concave.
- The test is limited by the precision of the geometry's coordinates – be careful of "invisible" bends in lines, as shown here:



### Example

This diagram shows both convex and concave examples.



### **FME Licensing Level**

FME Professional Edition and above

### **Transformer Category**

Filters

### **Related Transformers**

ConvexHullAccumulator

ConvexHullReplacer

### **Technical History**

Associated FME function or factory: @Geometry

## CoordinateConcatenator

Retrieves the value of all the feature's coordinates into an attribute, separated by the delimiter characters.

### Parameters

#### Coordinate Delimiter

This parameter can contain either an arbitrary string or a character, which is inserted between each coordinate set.

#### Coordinate Element Delimiter

This parameter can also contain either an arbitrary string or a character, which is inserted between each x/y/z element of each coordinate.

The delimiter character can be expressed as a single character, or it may be a special character sequence beginning with a backslash ("\"). Special character sequences are interpreted as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, *lan\es* will be interpreted as *lanes*.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

#### Coordinate Element(s) to Include

This parameter specifies which elements in the coordinate to output. For example, if the coordinate has a value of Z, only the z coordinate will be output.

#### Coordinate Attribute

This attribute stores the value of the feature's coordinates.

### Usage Notes

- This transformer works only for point, line, area, and donut features. Aggregate features will give invalid values.
- If Coordinate Element(s) to Include contains Z and the features contain two-dimensional data, only the x and y coordinates will be stored.

### Transformer Category

Calculators

### Technical History

Associated FME function or factory: @Tcl2, @Dimension

## **CoordinateCounter**

Stores the number of a feature's coordinates into an attribute.

### **Parameters**

Coordinate Count Attribute

The attribute that stores the number of coordinates. For multi-part features (aggregates), this is the sum of all coordinates in all parts of the feature.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @NumCoords

## **CoordinateExtractor**

Retrieves the value of the x, y, and z coordinate at the specified index into attributes. A negative index can be used to indicate the position relative to the end of the feature (-1 is the last coordinate, -2 the second last, and so on). The index can be entered as an integer, or may be taken from the value of another attribute by selecting the attribute name from the pull-down list. If the index is invalid, then the translation will be terminated.

If the feature was two-dimensional, then the Z attribute will be given the default value specified.

If the feature was a multi-part feature (aggregate), each part's coordinates are indexed sequentially.

### **Transformer Category**

Calculators

### **Transformer History**

This transformer was renamed from CoordinateFetcher.

### **Technical History**

Associated FME function or factory: @Coordinate, @Dimension



## CoordinateRemover

Removes one or more coordinates from the geometry of the feature.

### Output Port

- OUTPUT: This port contains the attribute *\_num\_removed*, which will contain the number of coordinates actually removed. (This could be less than the number specified if the feature did not have that many coordinates between the specified index and the end of the feature.)

### Parameters

#### Keep or Remove Coordinates

This parameter specifies how coordinates are chosen for removal.

- If you choose Remove, then, starting at the index specified, the transformer will remove the number of coordinates given in Number to Keep/Remove.
- If you choose Keep, then all coordinates except those specified will be removed.

#### Coordinate Index

A negative index can be used to indicate the position of the coordinate(s) to be removed relative to the end of the feature (-1 is the last coordinate, -2 the second last, and so on). The index can be entered as an integer, or may be taken from the value of another attribute by selecting its name from the pull-down list of attributes.

#### Number to Keep/Remove

If you choose Remove in the *Keep or Remove Coordinates* parameter above, then, starting at the index specified, the transformer will remove the number of coordinates that you specify in this parameter.

If you choose Keep in the *Keep or Remove Coordinates* parameter above, then all coordinates except those specified will be removed.

### Usage Notes

If you try to use this transformer on an aggregate, solid, or surface feature, the translation will terminate with an error.

If you try to use this transformer on an area feature that has holes, the results will not be useful.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Coordinate

## CoordinateRounder

Rounds off the coordinates of the feature to the specified number of decimal places. Any consecutive points that become duplicates as a result of the rounding are thinned by removing the redundant points.

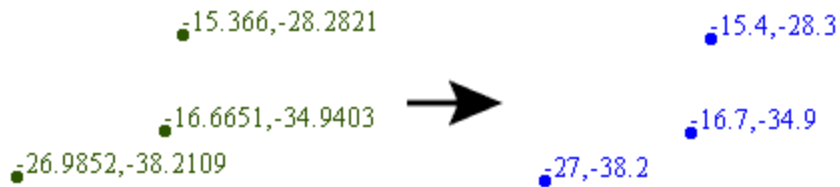
This transformer may be used to remove superfluous decimal points in the coordinates when they are destined for an ASCII output file.

Precision Values: X, Y and Z

The precision values control the number of decimal places that the coordinates are rounded to. A value of 0 causes the coordinates to be rounded to the nearest integer. A value of 1 causes rounding to the nearest tenth of a unit. Negative values are allowed. A value of -1 causes rounding to the nearest 10.

Each parameter may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Example



## Geometry Handling

If the Geometry Handling Advanced setting is set to "Enhanced" in the workspace, arcs have their start and end points rounded off, and ellipses aren't touched; otherwise, arcs and ellipses have their center points rounded off.

## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: @RoundOffCoords

## **CoordinateSwapper**

Swaps coordinate axes of the input features.

### **Parameters**

Swap Type

X <-> Y: Each feature's x-coordinates will be swapped with its y-coordinates.

X <-> Z: Each feature's x-coordinates will be swapped with its z-coordinates.

Y <-> Z: Each feature's y-coordinates will be swapped with its z-coordinates.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @Affine

## CoordinateSystemDescriptionConverter

Converts coordinate systems between FME and AutoDesk WKT, EPSG, ESRI WKT, MapInfo, OGC WKT, Oracle SRID, and PROJ.4 representations.

If, for example, you are converting from ESRI Shape to Oracle Spatial, in addition to converting the data, FME also converts the coordinate system metadata from the ESRI representation to the Oracle representation. This transformer allows you to do this manually.

### Source Coordinate System Description Attribute

The attribute that contains the source coordinate system.

### Coordinate System Description Conversion Direction

- Convert from FME Representation: Set the Representation parameter to the coordinate system type that you want to use to format the output field.
- Convert to FME Representation: Set the Representation parameter to select the coordinate system format to use when interpreting the input field.

### Representation

- Autodesk Well Known Text (WKT)
- EPSG number
- ESRI Well Known Text (WKT)
- MapInfo String
- OGC Well Known Text (WKT)
- Oracle SRID number
- PROJ.4 string

### Resulting Coordinate System Description Attribute

This is attribute that will contain the results. The default name is `_newcoordsys`.

## Usage Notes

You should use this transformer only in rare instances, much like the Reprojector. Instead, you should use the workspace Navigator view to set the source and destination coordinate systems for the translation.

## Related Transformers

This can be used in conjunction with the `CoordinateSystemFetcher` and `CoordinateSystemSetter` to use externally defined coordinate systems with FME features within the transformation environment.

## Transformer Category

Coordinate Systems

## Technical History

Associated FME function or factory: `@Reproject`

## **CoordinateSystemExtractor**

Retrieves the coordinate system of the feature into an attribute.

If the feature has no coordinate system, the attribute will be an empty string.

### **Transformer Category**

Coordinate Systems

### **Transformer History**

This transformer was renamed from CoordinateSystemFetcher.

### **Technical History**

Associated FME function or factory: @CoordSys

## **CoordinateSystemRemover**

Removes the coordinate system from all input features. This transformer does not reproject features, or otherwise modify their geometry.

To reproject features, either change the source and destination coordinate systems in the workspace tree view, or use the Reprojector transformer.

To set the coordinate system without reprojecting, use the CoordinateSystemSetter transformer.

### **Transformer Category**

Coordinate Systems

### **Technical History**

Associated FME function or factory: @Tcl2

## CoordinateSystemSetter

Tags all features with the specified coordinate system. It does not reproject features, or otherwise modify their geometry.

You should use this transformer if you know the feature belongs to a certain coordinate system, but it is not tagged. This can happen if the feature is read from a format that does not store coordinate system information.

### Parameters

Coordinate System

Click the Browse button to choose a coordinate system from the Coordinate System Gallery.

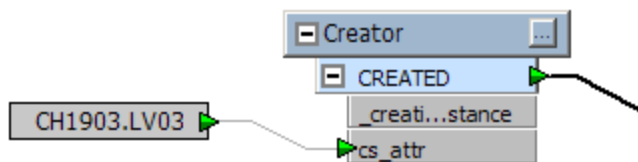
### Usage Notes

- To reproject features, either change the source and destination coordinate systems in the Navigator, or use the Reprojector transformer.
- To remove the coordinate system from features, use the CoordinateSystemRemover transformer.

### Setting a Feature's Coordinate System Based on an Attribute Value

The following example shows how to set a feature's coordinate system based on an attribute value.

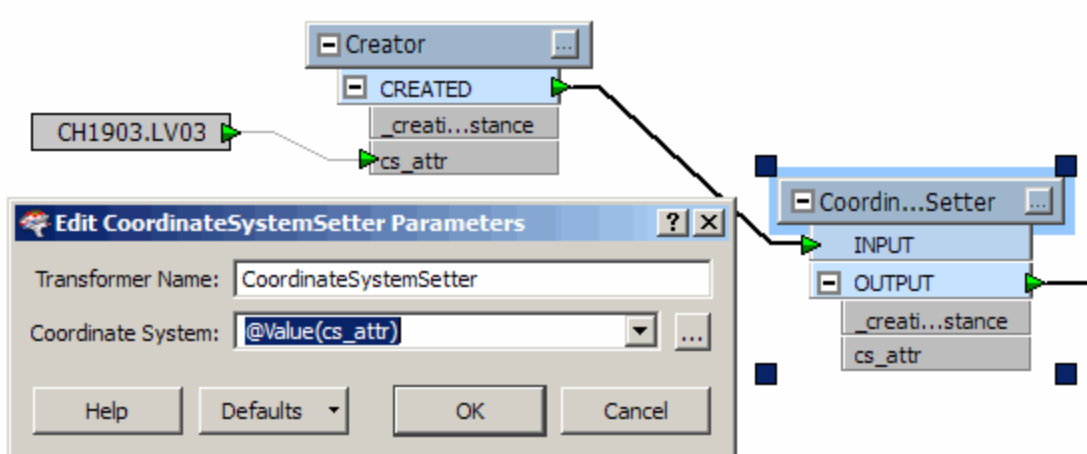
Input features should have this attribute set to the FME coordinate system name as shown in the Coordinate System Gallery (the example below uses "CH1903.LV03"). You can do this by adding an attribute, then choosing Insert > Constant to specify the coordinate system name. This example provides input features with their "cs\_attr" attributes set to "CH1903.LV03".



Right click in the CREATED port and choose Add Attribute. In this example, we called it cs\_attr.

Choose Insert > Constant.  
Enter the coordinate system name.  
Connect the constant to the attribute.

Click the Properties button in the CoordinateSystemSetter transformer, and enter `@Value(<attribute_name>)` in the Coordinate System field.



### Transformer Category

Coordinate Systems

### fmepedia

See fmepedia for additional information about this transformer.

### Technical History

Associated FME function or factory: @CoordSys



## Counter

Adds a numeric attribute to a feature and assigns a value. Each subsequent feature passing through the transformer receives an incremented value – in other words, the Counter is counting the features.

This transformer can be useful for assigning a unique, numeric ID number to a set of features, for counting the number of features, or for creating a basic histogram for values of a given attribute.

### Transformer Parameters

#### Transformer Name

This parameter is common to most transformers, and you do not always have to edit it to control workspace output. See the *Usage Notes* below for information on why you might want to edit this parameter for the Counter.

#### Count Output Attribute

The name of the attribute that will contain the results, and will appear in the list of OUTPUT port attributes.

#### Counter Name

Create separate sequences of numbers to be assigned, either by placing several Counter transformers with each having a different counter name, or by choosing an attribute whose value will be used as the counter name as each feature passes through. (In effect, using an attribute to supply the Counter name is like having a Group By option for the Counter.)

#### Count Start

Use this parameter to specify a starting value for the Counter. This is useful for applications where ranges of values have meanings in the problem domain. See *Usage Notes* below.

#### Count Scope

Specify whether the scope of this counter is Global (throughout the entire workspace) or Local (for this transformer only). Global counters with the same Counter Name will share the same counting sequence, while Local counters will each have a unique counting sequence.

## Usage Notes

### Using Multiple Counters

Using multiple Counter transformers in a workspace can produce different results. The default behavior of FME is to name all Counters the same.

When multiple Counters have the same name, they produce a single count. When multiple Counters have different names, each Counter produces a unique count starting at 1 (or the number determined by the Count Start parameter). You can rename any Counter by editing the Transformer Name parameter.

For example, there are two workspaces each with two Counters. Ten features passed through each Counter would result in the following:

#### Workspace 1

Counter Name: Counter1 - features numbered 1 - 10

Counter Name: Counter2 - features numbered 1 - 10

#### Workspace 2

Counter Name: Counter - features numbered 1 - 10

Counter Name: Counter - features numbered 11 - 20

You can use this feature as needed to produce different results.

## Transformer Category

Calculators

## **fmepedia**

See fmepedia for additional information about this transformer.

## **Technical History**

Associated FME function or factory: @Count

## **CRCCalculator**

Category: Calculators

Calculates a CRC value as directed for a feature and places the calculated CRC value into the attribute specified.

The CRC can be calculated on either "Geometry and Attributes" or "Attributes Only". In either case, the user optionally selects the attributes that are to take place in the CRC calculation. If no attributes are selected and "Geometry and Attributes" is specified, then the calculated CRC will be based only on the feature's geometry.

If no attributes are selected and "Attributes Only" is specified, then the output CRC value is 0.

### **Example**

This fmepedia example shows an advantage of using the CRCCalculator over the ChangeDetector.

### **Technical History**

Associated FME function or factory: @CRC

## Creator

Creates features using the parameters supplied, and sends them into the workspace for processing.

The Creator's interface allows you to select the desired type of geometry, then enter the coordinates and/or parameters that will create the desired object.

## Parameters

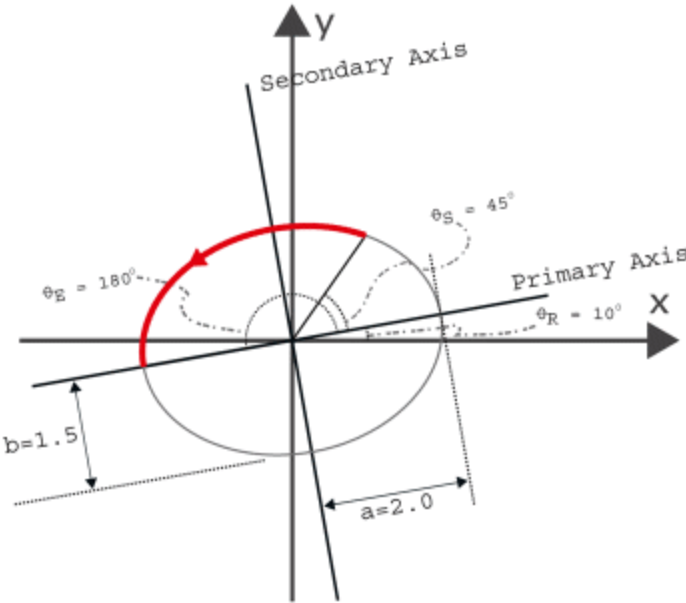
### Geometry Source

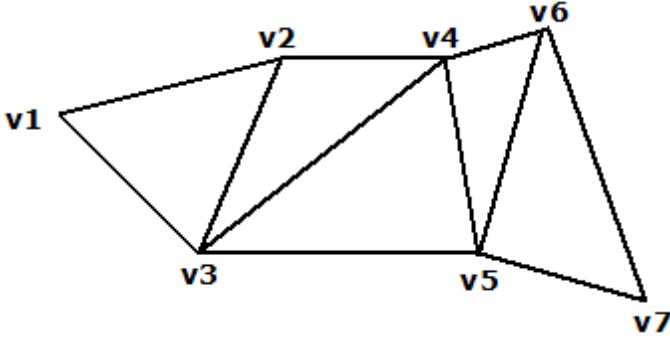
This parameter specifies how to create the geometry, using the options listed below. Your choice for this parameter determines whether other parameters are enabled or disabled.

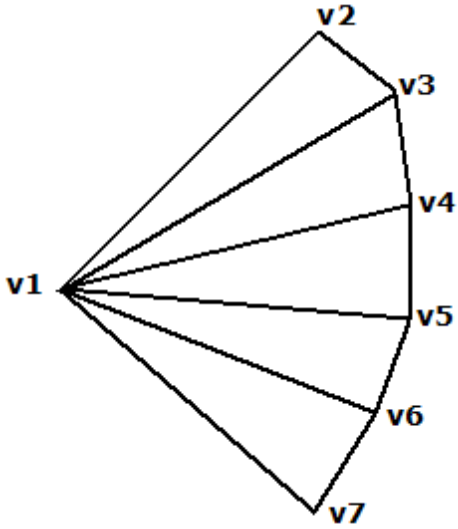
- **Geometry Object:** This option allows you to create the geometry types specified in the Geometry Object parameter below. The XML tab shows, by default, the XML representation of any object defined in the other areas of the dialog.
- **2D Coordinate List:** Creates two-dimensional geometry from the space-delimited list of x y coordinates entered in the Coordinate List parameter field.
- **3D Coordinate List:** Creates two-dimensional geometry with z-values from the space-delimited list of x y z coordinates entered in the Coordinate List parameter field.
- **2D Min/Max Box:** Creates a bounding box from bottom left and top right corner, specified as minx miny maxx maxy in the Coordinate List parameter field (for example: 0 0 10 10).

### Geometry Object

Geometry Object	Description
Null	Creates an object without geometry. For some applications, it can be useful to have such an object available; for example, when you are using a CSV writer but you want to output a header or footer.
Point	Creates a single point at the given coordinates.
Text	Creates a single text object at the given coordinates, using the text string, height and rotation.
Line	Creates a single line object using the specified coordinate pairs. If you want to create an area, use the Polygon option instead. Even if you specify the first and last coordinate pairs to be the same, the type of the object generated will still be <code>fme_line</code> .
Arc	Creates an arc using the specified parameters.  The primary axis parameter specifies the length of the primary axis, and the secondary axis parameter specifies the length of the secondary axis. The start angle parameter specifies the start angle for the arc, measured in degrees counterclockwise from horizontal. The sweep angle parameter specifies the number of degrees on the ellipse that define the arc, measured in degrees counterclockwise. The rotation parameter specifies the angle in degrees from the horizontal axis to the primary axis in a counterclockwise direction. A circle can be created by setting the primary and secondary axis to the same length and using a sweep angle of 360 degrees.

Geometry Object	Description
	 <p data-bbox="456 892 1416 951">Please note that not all transformers or output feature types work well with arcs. You may need an ArcStroker to simplify it.</p>
Ellipse	<p data-bbox="456 957 1416 1014">Creates an ellipse using the specified parameters. In order to create a circle, make sure the primary and secondary axes have the same length.</p> <p data-bbox="456 1031 1416 1121">Please note that the ellipse generated by this is an arc feature. Not all transformers or output feature types can work with arc features, so you may need an ArcStroker to simplify it.</p>
Polygon	<p data-bbox="456 1127 1416 1184">Creates a polygon feature. Works in the same way as the line option, but it will warn you if you forget to close the polygon.</p>
Box	<p data-bbox="456 1190 1416 1276">Creates a rectangular prism in 3D space. It is defined by a minimum corner and a maximum corner, but unlike a Rectangle Face, these two coordinates must not share identical x-, y-, or z-values.</p> <p data-bbox="456 1293 1416 1459">The two corner points unambiguously represent a unique rectangular prism, in which all faces are parallel to the coordinate planes. If the first point is the minimum point, then the surface normal points out from the box; otherwise, the box has been flipped inside-out and the surface normal points into the box. With conjunction of a 4x4 transformation matrix, a Box can be used to represent boxes that are not parallel to the coordinate planes. This matrix can store affine transformations.</p>
Rectangle Face	<p data-bbox="456 1465 1416 1522">Creates an optimized rectangular face representation that lies parallel on a coordinate plane (either xy-, xz-, or yz-plane).</p> <p data-bbox="456 1539 1416 1705">This face specifies its position by using two points, the minimum corner and maximum corner. Because the face must lie parallel to a coordinate plane, the corner points share a common coordinate value. For example, if the rectangular face lies on the xy-plane, the corner points share a common z-value. The surface normal of this rectangular face depends on the order of the specification of the min and max points, as described in the following table.</p>

Geometry Object	Description																					
	<table border="1" data-bbox="461 260 1411 575"> <thead> <tr> <th data-bbox="466 266 740 323">Plane to which rectangle is parallel</th> <th data-bbox="745 266 1159 323">Order of specification of (coordinates of) the corners</th> <th data-bbox="1164 266 1406 323">Direction of the surface normal</th> </tr> </thead> <tbody> <tr> <td data-bbox="466 329 740 365">XY</td> <td data-bbox="745 329 1159 365">Min-corner, max-corner</td> <td data-bbox="1164 329 1406 365">Positive Z-axis</td> </tr> <tr> <td data-bbox="466 371 740 407">YZ</td> <td data-bbox="745 371 1159 407">Min-corner, max-corner</td> <td data-bbox="1164 371 1406 407">Positive X-axis</td> </tr> <tr> <td data-bbox="466 413 740 449">XZ</td> <td data-bbox="745 413 1159 449">Min-corner, max-corner</td> <td data-bbox="1164 413 1406 449">Positive Y-axis</td> </tr> <tr> <td data-bbox="466 455 740 491">XY</td> <td data-bbox="745 455 1159 491">Max-corner, min-corner</td> <td data-bbox="1164 455 1406 491">Positive Z-axis</td> </tr> <tr> <td data-bbox="466 497 740 533">YZ</td> <td data-bbox="745 497 1159 533">Max-corner, min-corner</td> <td data-bbox="1164 497 1406 533">Positive X-axis</td> </tr> <tr> <td data-bbox="466 539 740 575">XZ</td> <td data-bbox="745 539 1159 575">Max-corner, min-corner</td> <td data-bbox="1164 539 1406 575">Positive Y-axis</td> </tr> </tbody> </table> <p data-bbox="461 596 1411 730">The surface normal determines the orientation of the rectangular face; that is, the direction in which the surface normal points indicates which side is the front. With conjunction of a 4x4 transformation matrix, a Rectangle Face can be used to represent rectangular faces that are not parallel to the coordinate planes. This matrix can store affine transformations.</p>	Plane to which rectangle is parallel	Order of specification of (coordinates of) the corners	Direction of the surface normal	XY	Min-corner, max-corner	Positive Z-axis	YZ	Min-corner, max-corner	Positive X-axis	XZ	Min-corner, max-corner	Positive Y-axis	XY	Max-corner, min-corner	Positive Z-axis	YZ	Max-corner, min-corner	Positive X-axis	XZ	Max-corner, min-corner	Positive Y-axis
Plane to which rectangle is parallel	Order of specification of (coordinates of) the corners	Direction of the surface normal																				
XY	Min-corner, max-corner	Positive Z-axis																				
YZ	Min-corner, max-corner	Positive X-axis																				
XZ	Min-corner, max-corner	Positive Y-axis																				
XY	Max-corner, min-corner	Positive Z-axis																				
YZ	Max-corner, min-corner	Positive X-axis																				
XZ	Max-corner, min-corner	Positive Y-axis																				
Triangle Strip	<p data-bbox="461 749 1214 772">Creates a triangle strip, which is a series of connected triangular faces.</p> <p data-bbox="461 793 1411 953">These faces are defined by three consecutive points in a point list. The first three vertices (labelled below as v1, v2, and v3), define the first triangular face. A new triangle is formed by connecting the next point with its two immediate predecessors. That is, every additional point vi defines a new triangular face with vertices vi-2, vi-1, and vi. For example, the second triangle is defined by v2, v3, v4, the third by v3, v4, v5, etc. The following diagram illustrates a typical Triangle Strip.</p>  <p data-bbox="461 1356 1411 1488">The orientation of the entire triangle strip is determined by the orientation of the first triangle. If the vertices of the first triangle are ordered counterclockwise, then the front of the strip is displayed; otherwise, the back of the strip is displayed. If the triangle strip has been flipped, then the front/back of the entire strip is actually the reverse of what the first triangle indicates.</p>																					
Triangle Fan	<p data-bbox="461 1505 1411 1556">Creates a triangular fan, which is a series of connected triangular faces. The fan differs from a Triangle Strip in the way that vertices define faces.</p> <p data-bbox="461 1577 1411 1709">The first three vertices (labelled below as v1, v2, and v3), define the first triangular face. A new triangle is formed by connecting the next point with its immediate predecessor and the first point of the triangle fan. That is, every additional point vi defines a new triangular face with vertices v1, vi-1, and vi. For example, the second triangle is defined by v1, v3, v4, the third by v1, v4, v5, etc. The following diagram illustrates a typical Triangle Fan.</p>																					

Geometry Object	Description
	 <p data-bbox="456 829 1416 940">The orientation of the entire triangle fan is determined by the order of vertices of any triangle within the fan (all the triangles are already oriented in the same direction). When they are ordered counterclockwise, the front is displayed; otherwise, the back is displayed.</p>
Face	<p data-bbox="456 947 1416 1003">Creates a planar area in 3D space. The planar structure can be a polygon, an ellipse or a donut.</p> <p data-bbox="456 1020 1416 1129">The orientation of a Face is determined by using the following rule: if the fingers of your right hand curl along the order of the vertices, the direction that the thumb points to is the front of the face. This thumb direction also describes the surface normal of the face, a vector that points outwards perpendicular from the area.</p> <p data-bbox="456 1146 1416 1234"><b>Curve closing method:</b> This method controls how the curve is closed. It is applicable only if the first and last coordinates entered do not match X, Y or Z coordinate values. It ensures that the coordinates of the start and end points match so that it is a valid area.</p> <p data-bbox="456 1251 1416 1308"><i>Average:</i> An additional point is added that connects the start and end point of the area. This point is computed by the average of the start and end points.</p> <p data-bbox="456 1325 1416 1360"><i>Extend:</i> The start and end points are connected with no additional points.</p> <p data-bbox="456 1377 1416 1507"><i>Extend or Average Based on Z:</i> The area is closed using the Average method if – and only if – the start and end points lie on the same coordinate plane (i.e. they share the same X, Y or Z coordinates). Otherwise, the Extend method is used to close the area.</p>
Custom	Creates an object based on an XML representation.

### Coordinate List

Creates either 2D geometry or 2D geometry with z-values, depending on your choice in the Geometry Source parameter. Enter a space-delimited list of x y (z) coordinates.

### Coordinate System

This parameter allows you to set the coordinate system on the specified feature. Select a coordinate system from the pull-down list, or click the Browse button to select from the Coordinate System Gallery. If you leave this parameter blank, the coordinate system will not be set on the feature(s) created by this transformer.

### Number to Create

This parameter specifies how many features will be created. When more than one feature is created, the attribute specified by the Creation Instance Attribute parameter will hold each feature's creation number (starting from 0).

### Create at End

This parameter specifies whether the feature should be added to the translation before any features that come from source datasets, or after.

### Creation Instance Attribute

When more than one feature is created (see the *Number to Create* parameter above), this attribute will hold each feature's creation number (starting from 0).

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: CreationFactory, @Geometry



## CSGBuilder

Creates Constructive Solid Geometry (CSG) from pairs of solid geometry features which are input through the A and B ports.

CSG is useful for representing complex solids by specifying the Boolean operations between simpler solids. For example, a wall with a window can be represented by taking the difference of the solid wall against the window. The CSGBuilder can be used to create a CSG solid that is comprised of relationships (order of Boolean operations or a hierarchy of Boolean operations) between the simpler solids.

### Input

- A and B: Pairs of solid geometry features.

### Output

Each Boolean operator is output via its corresponding port:

- UNION: Merges two objects into one.
- DIFFERENCE: Subtracts one object from another.
- INTERSECTION: The portion common to both objects.

Any features that do not have solid geometry, or extra features that come after the first feature with solid geometry in each group, are output through the UNUSED port.

### Parameters

Group By

If any Group By attributes are given, then each group will be treated independently. This allows a single transformer to operate on multiple pairs of input features.

### Related Transformers

CSG is the unevaluated model. The CSGEvaluator can be used to compute the equivalent boundary representation of the CSG solid.

### Transformer Category

3D

### FME Licensing Level

FME Professional edition and above

### Resources

See the definition of CSG at [http://en.wikipedia.org/wiki/Constructive\\_solid\\_geometry](http://en.wikipedia.org/wiki/Constructive_solid_geometry).

### Technical History

Associated FME function or factory: CSGFactory

## **CSGEvaluator**

Replaces the geometry of a feature that has CSG (Constructive Solid Geometry) by evaluating the tree of the CSG solid, effectively removing the constructive aspect of the geometry.

### **Input**

- INPUT: A feature that contains CSG. Features that do not have CSG geometry are untouched.

### **Output**

- OUTPUT: The result could be a multi-solid, a boundary representation solid, or a NULL geometry.

### **Transformer Category**

3D

### **FME Licensing Level**

FME Professional edition and above

### **Resources**

See the definition of CSG at [http://en.wikipedia.org/wiki/Constructive\\_solid\\_geometry](http://en.wikipedia.org/wiki/Constructive_solid_geometry).

### **Technical History**

Associated FME function or factory: @CSG

## CsmapReprojector

Reprojects feature coordinates from one coordinate system to another using the CS-MAP library.

### Parameters

#### Source and Destination Coordinate System

If the specified source coordinate system is "Read from feature" or blank, the input feature's coordinate system is used as the source. In this case, if the input feature doesn't have a coordinate system, this transformer only sets the coordinate system of the feature to the destination coordinate system, and the coordinates of the feature remain unchanged.

#### Vertical Handling

This parameter determines how Z values will be handled. There are three options:

- **Ignore heights and leave them unchanged.** Z values will not affect the reprojection and will not be changed. This is the traditional behavior of the Reprojector transformer when the selected reprojection engine is "FME".
- **Heights are relative to the ellipsoid(s).** Z values will be treated as ellipsoid heights with the same units as the horizontal units (or meters, for geographic coordinate systems).
- **NAD27 heights in NGVD29; NAD83 heights in NAVD88.** VERTCON will be used to convert between NAD27 (NGVD29) and NAD83 (NAVD88). Z values have the same units as the horizontal units (or meters, for geographic coordinate systems).

Note that rasters may only be reprojected in 2D (that is, with *Ignore heights and leave them unchanged*)

#### Interpolation Type (Raster Only)

The Interpolation Type affects only raster data. Cell values are interpolated in order to change the raster to the specified size.

- Nearest Neighbor is the fastest but produces the poorest image quality.
- Bilinear provides a reasonable balance of speed and quality.
- Bicubic is the slowest but produces the best image quality.
- Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Cell Size (Raster Only)

The Cell Size applies only to raster features.

- **Stretch Cells:** The cell size of the raster will be adjusted to maintain the same number of rows and columns in the reprojected raster as there were in the input raster.
- **Square Cells:** The number of rows and columns as well as the spacing will be changed to maintain approximately the same cell ground area and form square cells where the horizontal and vertical cell sizes are equal. Like the Square Cells option, Preserve Cells will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.
- **Preserve Cells:** Like the Square Cells option, this option will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.

### Dynamic Coordinate Systems

If the destination coordinate system is specified as "\_AZMEA\_" or "\_AZMED\_", each input feature is reprojected to either an equal area or equal distance projection appropriate for that feature, respectively. In general, this causes a new coordinate system to be defined for each input feature.

Each feature remembers which specific equal distance or equal area coordinate system it has, and can be safely reprojected back to a normal (non-dynamic) coordinate system.

For example:

There is an input feature representing a point on the earth in LL-WGS84 (normal lat/long).

- The point is reprojected to \_AZMED\_ via a CsmapReprojector transformer. The Source Coordinate System parameter is set to LL-WGS84 and the Destination Coordinate System parameter is set to \_AZMED\_.
- The x and y coordinates of the point are extracted into x1, y1.

- Set  $x_2 = x_1 + 1000$ , and  $y_2 = y_1$ .
- Add a vertex to the point to make the line  $(x_1, y_1) \rightarrow (x_2, y_2)$ .
- Reproject back to LL-WGS84 via a CsmmapReprojector with the Source Coordinate System parameter set to "Read from feature" and the Destination Coordinate System parameter set to LL-WGS84.

You have now changed the point into a line extending 1km east of the original point, in lat/long.

### **Usage Notes**

- This transformer works with raster and vector data.
- This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Coordinate Systems

### **Technical History**

Associated FME function or factory: @Reproject

## Curvefitter

Smooths lines derived from line segments, points or raster data, and replaces a series of line segments with the optimal combination of straight lines and embedded arc segments required to create smooth curving lines. This process provides a truer representation of real-world features and can reduce file sizes by up to 80%.

In addition to processing simple line features, Curvefitter preserves feature topology when smoothing boundaries of adjacent area features.

Before the advent of enhanced geometry, FME was forced to stroke arcs into line sections to be able to process that data. Many other GIS packages that did not support arcs also ended up forcing the stroking of arcs, the net results being a great amount of data that could be represented by arcs but isn't. The Curvefitter transformer will allow a user to replace such stroked line segments with true mathematical arc segments. It will likewise permit the replacement of surveyed line features with an arc representation.

### Parameters

#### Precision

Precision is the main transformer parameter that guides the curve fitting process. It sets the maximum deviation allowed at any point along the polyline between the original and the resulting polyline.

#### Flattening

Allows very flat curves to be represented by straight segments. Any curve that has a mid-ordinate less than this amount will be replaced by a straight segment. A typical value is 10% of the precision setting.

#### Compression Weight, Smoothness Weight, Accuracy Weight

The values of the three Weight parameters determine the importance of the three factors relative to each other. Compression is the reduction in the number of vertices. Smoothness is the tangency of consecutive segments – how close the end angle of a segment is to the start angle of the next segment. Accuracy is how closely the resulting curve overlays the original.

#### Preserve Shared Boundaries

This parameter is useful when area features that form a coverage are sent into the transformer. If it is set to No, then each feature is considered independent of all others, and if any features shared boundaries, gaps or overlaps may be introduced. If it is set to Yes, then the input data is decomposed into boundary lines, these are Curvefitted, and then the areas are reassembled. In this case, no gaps will be created. If only linear or non-adjacent area features are input into the transformer, No is the best choice.

#### Allow Arcs at Nodes

Note that it is possible for the Curvefitter to introduce overlaps between areas with shared boundaries if a very large precision value is used. The chances of this happening can be reduced if this parameter is set to No, which prevents the Curvefitter from having any arcs starting or ending at a node.

#### Flag Introduced Overlaps

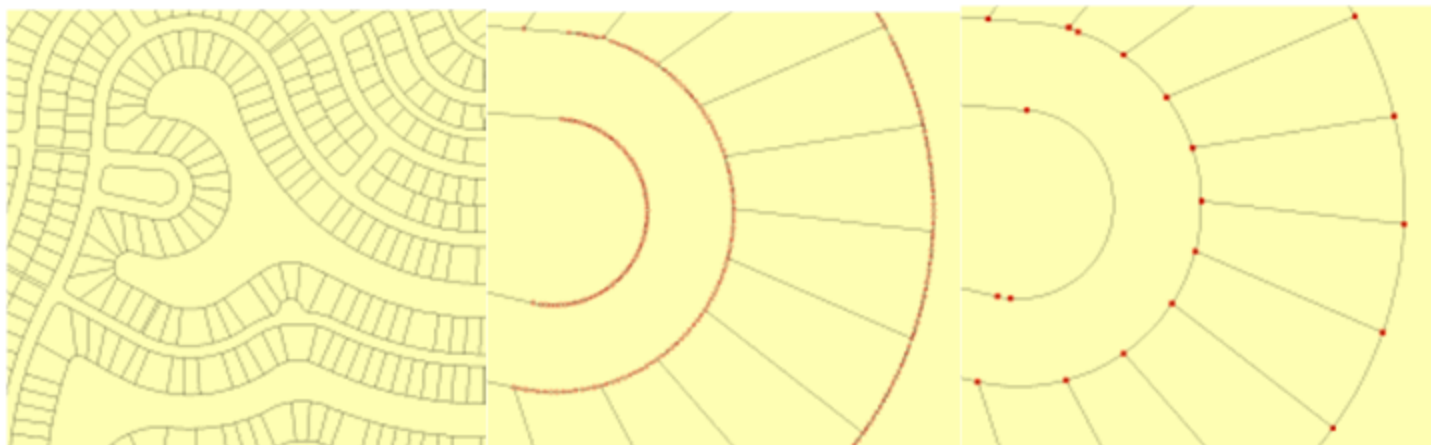
If this parameter is set to Yes, then any overlaps between features that are introduced by the Curvefitting are flagged with a point output via the OVERLAPS port.

### Curvefitter Applications and Functionality

Using the Curvefitter, you can:

- Compress file sizes by 30 - 80%
- Smooth jagged polylines
- Process adjacent area features
- Restore arcs in legacy data
- Automatically process data

## Example: Parcel Data



**Parcel Data**

**Before Curvefitter**

**After Curvefitter**

### Procedure

A subset 6.1 MB ESRI Shape file (provided by Grays Harbor County, in Washington State) was extracted and converted into four different formats: DWG file (3.8 MB), MapGuide SDF (4.5 MB), ESRI Personal Geodatabase (5.4 MB) and ArcGIS 9.2 File Geodatabase (1.89 MB).

Each file was then processed using the following Curvefitter settings: Precision 0.1 feet; Flattening 0.1; Compression Weight 1; Smoothness Weight: 1; Accuracy Weight: 1; Preserve Shared Boundaries: Yes.

### Parcel Data Test Results

Data Format	Before Curvefitter	After Curvefitter	File Size Reduction
AutoCAD DWG	3.8 MB	1.6 MB	58%
MapGuide SDF3	1.6 MB	4.5 MB	64%
ESRI Personal Geodatabase	5.4 MB	4.8 MB	11%
ESRI ArcGIS 9.2 File Geodatabase	1.89 MB	1.07 MB	77%

### External References

For more information and examples, please see:

- How to get the most from the Curvefitter
- fmpedia: Curvefitter.
- Technology brief (PDF format) on Safe Software's website.

### Transformer Category

Manipulators

### FME Licensing Level

This transformer is an extra-cost add-on to FME Professional Edition and above.

### Dependencies

The Curvefitter transformer is based on linear optimization technology licensed from TCI Corp. It is available for purchase as an add-on to premium editions of FME. Please contact sales@safe.com.

### Technical History

Associated FME function or factory: @Curvefit



## CustomTransformerLooper

Note: This transformer is considered experimental and will likely change form in the future. Dependence on this transformer should be considered only with direct cooperation with Safe Software Inc.

Allows external looping over a selected LINKED custom transformer. It iterates over the custom transformer, sending features emitted from the OUTPUT port back to the INPUT port over each iteration. The intent is to allow looping in workspaces without being subject to blocking features getting in the way.

Iteration continues until there is no "loop-back" output on a given iteration. Any output from that iteration which is emitted through the custom transformer's "final iteration" output port is deleted after all but the final iteration.

### Parameters

#### *Linked custom transformer*

The name of the custom transformer which will be used for each iteration of the loop. It is to be structured as described above.

#### *Name of loop input port*

The name of the custom transformer's "Loop input" port.

#### *Name of loop output port*

The name of the custom transformer's "Loop output" port.

#### *Name of recurring input port*

(Optional) The name of the custom transformer's "recurring input" port, if needed.

#### *Name of immediate output port*

(Optional) The name of the custom transformer's "Immediate output" port.

#### *Maximum number of iterations*

The maximum number of iterations to perform. If there are still features left to loop back after this iteration, they are written to the LOOP\_OUTPUT port. A value of "0" indicates that looping will continue until there is no output on LOOP\_OUTPUT.

#### *Additional parameters to custom transformer*

(Optional) A space-separated list of additional parameter/value pairs to be given to the custom transformer. (This is entered verbatim into the generated mapping file code and must therefore contend with FME's mapping file parser's character interpretation.)

### Transformer Category

Infrastructure



## DateFormatter

Reformats and replaces date or time strings into a new date format.

The source string can be in almost any date and/or time format. (Note, however, that the DateFormatter does not currently support dates earlier than 1902.)

Some valid examples include:

- 20091206 15:05
- 20091206150500
- December 6, 2009
- 06 December 09, 15:05
- 3:05pm

## Parameters

### Date Attributes

Choose the attributes to reformat and replace.

### Date Format

The source date can be in virtually any standard date and/or time string, which can include standard time zone mnemonics. If only a time is specified, the current date is assumed. If the string does not contain a time zone mnemonic, the local time zone is assumed. The source date can consist of zero or more specifications of the following form:

Type	Description
time	<p>time of day, which is of the form: <i>hh[:mm[:ss]] [meridian] [zone]</i> or <i>hhmm [meridian] [zone]</i>. If no meridian is specified, <i>hh</i> is interpreted on a 24-hour clock.</p> <p><b>Warning:</b> An integer less than 24 will be interpreted as an hour, with the date assumed to be today's date. Therefore, an input of "0" is taken to mean "today at midnight." If zero values are to be considered invalid, they must be filtered using a Tester before reaching the DateFormatter.</p>
date	<p>A specific month and day with optional year. The acceptable formats are <i>mm/dd[/yy]</i>; <i>monthname dd [, yy]</i>; <i>dd monthname [yy]</i>; and <i>day, dd monthname [yy]</i>. The default year is the current year. If the year is less than 100, we treat the years 00-68 as 2000-2068 and the years 69-99 as 1969-1999.</p> <p><b>Note:</b> Some older UNIX and Windows platforms cannot represent the years 38-70, so an error may result if these years are used.</p>
ISO 8601 point-in-time	<p>An ISO 8601 point-in-time specification, such as <i>CCyymmddThhmmss</i>, where <i>T</i> is the literal T, <i>CCyymmdd hhmmss</i>, or <i>CCyymmddThh:mm:ss</i>.</p>
relative time	<p>A specification relative to the current time. The format is <i>number unit</i>. Acceptable units are <i>yearfortnight</i>; <i>month</i>; <i>week</i>; <i>day</i>; <i>hour</i>; <i>minute (or min)</i>; <i>second (or sec)</i>.</p> <p>The unit can be specified as a singular or plural, as in <i>3 weeks</i>. These modifiers may also be specified: <i>tomorrow</i>; <i>yesterday</i>; <i>today</i>; <i>now</i>; <i>last</i>; <i>this</i>; <i>next</i>; <i>ago</i>.</p> <p>The actual date is calculated according to the following steps. First, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added. Next, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences and the correct date is given when going from the end of a long month to a short month. Daylight savings time correction is applied only when the relative time is specified in units of days or more, i.e., days, weeks, fortnights, months or years.</p>

## Symbols

The format specifier determines the format for the new date attribute. It can include the following symbols to substitute the date and time values taken from the source attribute:

<b>"-"</b>	Quoted strings (%Y"-"%m"-"%d)
<b>%%</b>	Insert a percent symbol (%)
<b>%a</b>	Abbreviated weekday name (Mon, Tue, etc.).
<b>%A</b>	Full weekday name (Monday, Tuesday, etc.).
<b>%b</b>	Abbreviated month name (Jan, Feb, etc.).
<b>%B</b>	Full month name.
<b>%c</b>	Locale specific date and time. The format for date and time in the default "C" locale on UNIX/Mac is "%a %b %d %H:%M:%S %Y". On Windows, this value is the locale specific long date and time, as specified in the Regional Options control panel settings.
<b>%C</b>	First two digits of the four-digit year (19 or 20).
<b>%d</b>	Day of month (01 - 31).
<b>%D</b>	Date as %m/%d/%y.
<b>%e</b>	Day of month (1 - 31), no leading zeros.
<b>%h</b>	Abbreviated month name.
<b>%H</b>	Hour in 24-hour format (00 - 23).
<b>%I</b>	Hour in 12-hour format (01 - 12).
<b>%j</b>	Day of year (001 - 366).
<b>%k</b>	Hour in 24-hour format, without leading zeros (0 - 23).
<b>%l</b>	Hour in 12-hour format, without leading zeros (1 - 12).
<b>%m</b>	Month number (01 - 12).
<b>%M</b>	Minute (00 - 59).
<b>%n</b>	Insert a newline.
<b>%p</b>	AM/PM indicator.
<b>%r</b>	Time in a locale-specific "meridian" format. The "meridian" format in the default "C" locale is "%I:%M:%S %p".
<b>%R</b>	Time as %H:%M.
<b>%s</b>	Count of seconds since the epoch, expressed as a decimal integer.
<b>%S</b>	Seconds (00 - 59).
<b>%t</b>	Insert a tab.
<b>%T</b>	Time as %H:%M:%S.
<b>%u</b>	Weekday number (Monday = 1, Sunday = 7).
<b>%U</b>	Week of year (00 - 52), Sunday is the first day of the week.
<b>%V</b>	Week of year according to ISO-8601 rules. Week 1 of a given year is the week containing 4 January.
<b>%w</b>	Weekday number (Sunday = 0, Saturday = 6).
<b>%W</b>	Week of year (00 - 52), Monday is the first day of the week.
<b>%x</b>	Locale-specific date format. The format for a date in the default "C" locale for UNIX/Mac is "%m/%d/%y". On Windows, this value is the locale-specific short date format, as specified in the Regional Options control panel settings.
<b>%X</b>	Locale-specific 24-hour time format. The format for a 24-hour time in the default "C" locale for UNIX/Mac is "%H:%M:%S". On Windows, this value is the locale-specific time format, as specified in the Regional Options control panel settings.
<b>%y</b>	Year without century (00 - 99).
<b>%Y</b>	Year with century (e.g. 1990)
<b>%Z</b>	Time zone name.

## Usage Notes

- In a numerical date such as 10/11/99, the first number is always interpreted as the month. (So the given date is October 11, not November 10.) A date such as 13/1/05 will therefore produce an error, because 13 is an invalid month.
- Each date is treated on a feature-by-feature basis; no attempt is made to determine a common format amongst all input.
- Relative Dates: The DateFormatter can accept relative dates as an input string. For example, if today is Monday, 25-Oct-2010, it can convert *next thursday* to "28-Oct-2010" or *three years ago* to "25-Oct-2007". See the fmepedia link below for workspace examples.
- If the source feature has an attribute named <sourceDateAttr>.full, as it would coming from an Oracle DATE field, this attribute will be used to obtain time-of-day information.
- If the source attribute is a 14-digit number, its format will be interpreted as YYYYMMDDHHMMSS, as formatted by the Oracle reader.
- If the source attribute is blank, the new attribute returns today's date.
- If the source date or the format string is not valid, the new attribute is returned blank.

## Transformer Category

Calculators

### fmepedia

See fmepedia for additional information about this transformer.

## Technical History

Associated FME function or factory: @Tcl2

## Deagggregator

Decomposes an aggregate feature into its components.

Each component of the original feature is output via the port corresponding to its geometry type. Each output feature has a complete, unaltered copy of the source feature's attributes. If a non-aggregate feature is input, it will be output untouched via the port corresponding to its geometry.

### Parameters

#### List Attribute

The optional List Attribute is used to associate attributes with each member of the aggregate.

If the feature has a list of attributes that are in parallel to the members of the aggregate, these can be decomposed onto the output features by specifying the List Attribute parameter.

#### Recursive

This parameter is used to deaggregate nested aggregates recursively. If Yes, the final output will contain no aggregates, only the underlying geometry components. Otherwise, any nested aggregates will not be broken apart.

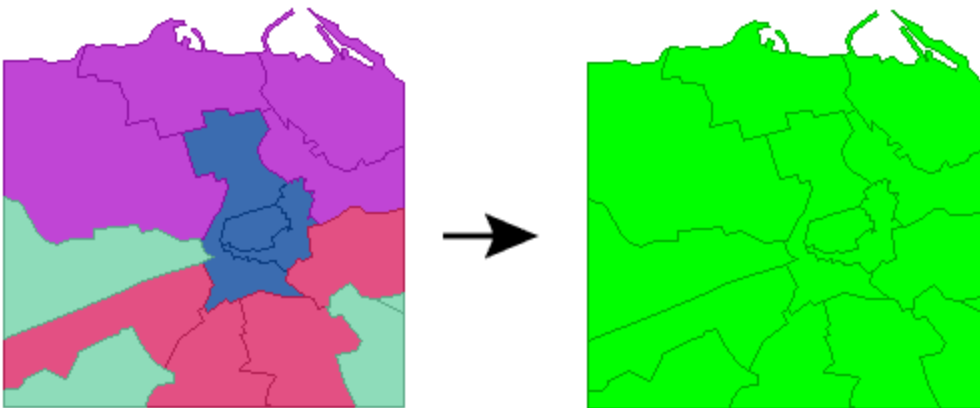
#### Part Number Attribute

If the Part Number Attribute is specified, then each deaggregated output feature is given an attribute containing the part's (deaggregated feature) index within the original aggregate feature. This clause will not apply if Recursive is set to Yes.

#### Geometry Name Attribute

For geometries that are named, supplying the Geometry Name Attribute will set the attribute to the name of the (deaggregated) geometry.

### Example



### Transformer Category

Collectors

### fmepedia

See [fmepedia](#) for additional information about this transformer.

### Technical History

Associated FME function or factory: DeaggregateFactory

## **Decelerator**

Slows down the flow of features through the workspace.

### **Parameters**

Processing Slowdown Method

Choose from Per Feature Delay or Limit Features Per Second.

Delay Per Feature (Seconds)

Each feature will be delayed by the number of seconds specified.

Maximum Features Per Second

The transformer will ensure that no more than that number of features are output each second (however, no guarantee is made that any pair of sequential features will have the same time gap between them).

### **Transformer Category**

Web Services

### **Transformer Type**

Feature-based

### **Related Transformers**

This transformer is commonly used in conjunction with the HTTPFetcher to slow down web requests and avoid overwhelming the target server.

### **Technical History**

Associated FME function or factory: @Tcl2 (after)

## DecimalDegreesCalculator

Calculates a decimal degree value from separate degrees, minutes, and seconds (DMS) values, stored in attributes.

### Parameters

Degrees, Minutes and Seconds Attributes

After connecting this transformer, choose the attribute that contains each of the degrees, minutes, and seconds attributes. The minutes and seconds values must always be positive.

If these parameters are contained in a single attribute, see Usage Notes below.

### Usage Notes

If the degrees/minutes/seconds are together in a single attribute, they can be parsed apart using the StringSearcher transformer.

For instance, if the attribute has the format:

```
-DDDMMSS.SS°
```

the following regular expression could be used in the StringSearcher to parse it into a list attribute with three elements:

```
^(-[0-9]+)([0-9][0-9])([0-9][0-9][.][0-9]*)
```

The three elements could then be exposed and used as input to this transformer.

### Transformer Category

Calculators

### Technical History

Associated FME function or factory: @Evaluate

## **DEMDistanceCalculator**

Calculates the distance between a number of input vector lines and the elevation values of a reference DEM raster, and outputs a new DEM raster per input line.

The data contained in the resulting DEM consists of the 3D distance between the line being considered and the corresponding point on the reference DEM.

### **Parameters**

Group By

Use this parameter to organize lines into groups, with each group of lines having its own reference DEM.

### **Usage Notes**

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

### **Transformer Category**

Calculators

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: DEMDistanceFactory

## DEMGenerator

Generates a Digital Elevation Model (DEM) as a regularly spaced set of output DEM points from the input POINTS/LINES, 3D\_LINES.

### Input

- **POINTS/LINES:** The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model. This port also accepts raster features as input.
- **BREAKLINES:** These features are put into the model as breaklines such that triangle edges will always be along the line of the feature.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, DEMGenerator, ContourGenerator, and SurfaceModeller transformers.

- **POINTS:** The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model. Raster features should go into the POINTS input.
- **BREAKLINES:** These features are put into the model as breaklines such that triangle edges will always be along the line of the feature.
- **3D\_LINES:** The vertices of the 3D line are inserted into the model to define the surface. They are not used to define breaklines but rather just provide 3D data for the model from their vertices.

### Output

- **DEM\_POINTS:** Returns the model as a series of evenly spaced 3D points based on the sampling rate specified.

### Parameters

#### Surface Tolerance

The surface tolerance is used when building the surface to determine which vertices to add to the surface model. Specifying a value of 0 turns off the vertex filtering so that all vertices (except those with duplicate x,y) are added to the model.

When specified, the surface tolerance is used during the construction phase of the surface model to determine whether a vertex will be added to the model, or whether it will be discarded and not added to the model.

The surface tolerance works as follows, for each vertex that is being added to the model:

- If the x,y location is outside the convex hull of the existing model, it is added to the model.
- If the x,y location is inside the existing model:
  - The difference between the z value from the existing TIN and the z value of the vertex is calculated.
  - This difference is compared to the surface model tolerance.
  - The vertex is only added to the model if the difference is greater than the surface tolerance; otherwise, the vertex is discarded.

#### Interpolation Method

This parameter defines how the z elevation for each point is determined. If CONSTANT is specified, then the z elevation of each coordinate is set to be the elevation of the closest model point. If PLANAR is set, then interpolation is used to determine the elevation. If AUTO is set, then the best method will be chosen automatically.

#### Output DEM X and Y Cell Spacing

The Output DEM X Cell Spacing and Output DEM Y Cell Spacing directives are used to specify the sampling interval for the generated points. These spacings are measured in the ground units of the input data.



Elevation Attribute

The Elevation Attribute specified will be added to the output point features and will hold their z value.

### **Usage Notes**

Use the RasterDEMGenerator if a DEM is destined to be sent to a raster format, or if further raster processing is required.

### **Transformer Category**

Surfaces

### **FME Licensing Level**

FME Professional Edition and above

### **Technical History**

Associated FME function or factory: SurfaceModelFactory

## Densifier

Adds vertices to the feature by interpolating new coordinates along its definition at some interval of distance. The interval may be along only one of the two primary axes, or it may be along the length of the line segments.

## Parameters

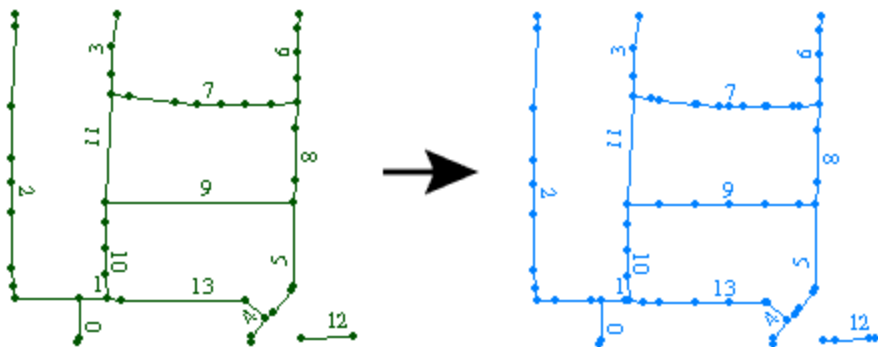
### Densification Interval

The interval parameter may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Densification Axis

This parameter is often used to densify the vertices of a feature to prepare it for reprojection. By adding vertices to long linear segments, the feature better represents the original in a different coordinate system.

## Example



## Geometry Handling

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Enhanced in the workspace, arc segments in a path are preserved and not densified. Otherwise, a path is stroked into a line prior to adding more vertices.

## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: @AddVertices

## DensityCalculator

Determines the density of a group of CANDIDATE features based on the area of a corresponding AREA feature.

### Input

- AREA: Usually a single reference feature.
- CANDIDATE: A group of CANDIDATE features.

### Output

- AREA: The original AREA features with the Density Attribute added. Its value will be the sum of the corresponding CANDIDATE features' measure (specified by the Calculate Density By parameter) divided by the area of the AREA feature.
- CANDIDATE: The original CANDIDATE features with the Density Attribute added. For all CANDIDATEs in a group, the attribute's value will be the density calculated for the first corresponding AREA feature.

### Parameters

#### Group By

The default behavior is to use the entire set of features as the group. This option allows you to select attributes that define which groups to form.

#### Calculate Density By

A sum value for all input CANDIDATEs will be calculated according to the Calculate Density By parameter. Choose Line Length, Number of Features or Area.

#### Density Attribute

On the first corresponding AREA feature, the Density Attribute will be set to the sum divided by the AREA feature's area. This same attribute value will also be applied to all the CANDIDATE features output.

### Usage Notes

- This transformer does not make any assumptions about whether or not candidate objects are inside the AREA feature. To determine this, you can use a SpatialFilter or SpatialRelator in conjunction with the DensityCalculator.
- If any additional AREA features are supplied in the same group, their Density Attribute will be set in the same manner, but this value will not be applied to the CANDIDATE features.

### Transformer Category

Calculators

### Related Transformers

SpatialRelator

### FME Licensing Level

FME Professional edition and above

### fmepedia

See fmepedia for more information on this transformer.

### Technical History

Associated FME function or factory: DensityFactory

## DGNStyler

Prepares features for output to Bentley Microstation Design V7/V8 by providing a convenient interface to set a variety of Bentley Microstation Design format-specific attributes.

### Parameters – Color

#### Color By Level

This parameter specifies that the pen color will be set by the color of the level specified on an FME Feature, rather than by the RGB color or Index color. (Note that this parameter is only supported by the V8 version of the Design writer.)

Format attribute set: `igds_color_set_bylevel`

#### Color Type

This parameter specifies format of the color to use. This applies to both the pen color and the fill color which are together used to render the feature.

#### RGB Color

This parameter specifies the pen color that will be used to render the feature. The pen color determines the color of lines, and area boundaries. To edit this parameter, click the browse button to the right of the text field.

Format attribute set: `fme_color`

Note that when this parameter is specified, the `igds_color` attribute is removed from the feature and `fme_color` is set. This is because `igds_color` overrides `fme_color` if they are both present.

#### RGB Fill Color

This parameter specifies the fill color that will be used for the feature. The fill color determines the color used within the boundary of area features. To edit this parameter, click the browse button to the right of the text field.

Format attribute set: `fme_fill_color`

Note that when this parameter is specified, the `igds_fill_color` attribute is removed from the feature and `fme_fill_color` is set. This is because `igds_fill_color` overrides `fme_fill_color` if they are both present.

#### Index Color

This parameter specifies the index of a pen color in a table of colors. The indexed color will be used for the feature. The pen color determines the color of lines, and area boundaries. This parameter can be set to integer value between 0 and 255.

Format attribute set: `igds_color`

#### Index Fill Color

This parameter specifies the index of a fill color in a table of colors. The indexed color will be used for the feature. The fill color determines the color used within the boundary of area features. This parameter can be set to integer value between 0 and 255.

Format attribute set: `igds_fill_color`

### Parameters – Cells

If this section is active, point features will be turned into cells and given a cell name, rotation, and size.

Features with other geometry types will not be affected by settings in this section.

Format attributes set: `igds_type` to `igds_cell` or `igds_shared_cell`

#### Cell Library File

This optional parameter is used to specify the name of an existing Bentley Microstation Design Cell file that will be used by the transformer as a

source for cell names. It is not used by the actual Design writer – the cell library file must be specified in the writer’s parameters and should contain the same cell names as the file specified here. Most often, the same file will be used in both the transformer and the writer. Further note that if no cells are to be used, then this parameter can be left blank.

#### Cell Name

Cell Name specifies the cell that will be placed at the point location. Click the browse button to pick the name from the set of cells defined in the cell library file.

Format attribute set: **igds\_cell\_name**

#### Cell Mode

This parameter specifies how a cell will be written. If a value of Library is specified, an **igds\_cell** element is created which is expected to reference a cell definition in the cell library file. Otherwise, if a value of Shared (V8 Only) is specified, then an **igds\_shared\_cell** element is created in the destination which does not refer to a cell definition in the cell file definition. (Note that the Shared value for this parameter is only supported by the V8 version of the Design writer.)

Format attribute set: **igds\_type** to **igds\_cell** or **igds\_shared\_cell**

#### Relative Graphic Cells

This parameter specifies if the graphic cells are to be created as relative graphic cells. Relative graphic cells map the cell members with the lowest level number to the current cell feature's level which becomes a base level for the relative graphic cell. All the subsequent cell members have their levels offset according to the base level. This does not apply to point or shared cells. (Note that this parameter is only supported by the V8 version of the Design writer.)

Format attribute set: **igds\_is\_graphic\_cell\_relative**

#### Cell Rotation

Cell Rotation specifies rotation of the cell (in degrees counterclockwise from horizontal). This can be set to any floating point value between -360.0 to 360.0, or be set to a value taken from an attribute.

Format attribute set: **igds\_cell\_rotation**

#### Cell Size By

Cell Size By indicates how the size of the cell will be specified. If master units are chosen, then the cell will be scaled so that its range in x, y, and z covers the specified amounts. If scale factor is chosen, then the amounts in x, y, and z specify the scaling which will be applied to the cell in each of the three axes.

#### X

X specifies the sizing amount for the x axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attributes set: **igds\_cell\_x\_scale** (if Cell Size By is Scale Factor); **igds\_cell\_size\_x** (if Cell Size By is Master Units)

#### Y

Y specifies the sizing amount for the y axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attributes set: **igds\_cell\_y\_scale** (if Cell Size By is Scale Factor); **igds\_cell\_size\_y** (if Cell Size By is Master Units)

#### Z

Z specifies the sizing amount for the z axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attribute set: **igds\_cell\_z\_scale** (if Cell Size By is Scale Factor)

### Parameters – Lines

If this section is active, then linear features will be prepared for output to Bentley Microstation Design.

Features with other geometry types will not be affected by settings in this section.

### Line Weight

Line Weight specifies an index in the range 0 to 31 which designates the width or thickness of the lines used to render a graphic element. In addition, the line weight may be set to ByLevel, which specifies that the line weight be set by the level specified on an FME Feature. Click the browse button to pick from a visual representation of the possible values.

Format attribute set: `igds_weight` and `igds_weight_set_bylevel`

### Line Style

Line Style specifies an index in the range 0 to 7 which designates the style of the lines used to render a graphic element. In addition, the line style may be set to ByLevel, which specifies that the line style be set by the level specified on an FME Feature. Click the browse button to pick from a visual representation of the possible values.

Format attribute set: `igds_style` and `igds_style_set_bylevel`

### Additional References

For more information about DWG/DXF styling:

- See the Bentley Microstation Design Reader/Writer > Feature Representation section in the *FME Readers and Writers* manual. In Workbench, select Help > FME Readers and Writers Reference.

### Transformer Category

Stylers

## **DimensionExtractor**

Returns the dimension of an attached feature as a new attribute.

### **Parameters**

Dimension Attribute

The name of the attribute. The attribute will hold either 2 or 3.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Dimension

## DirectTweeter

Sends a direct Twitter message from Workbench.

## Output Ports

Upon completion, the output feature will have several new attributes:

*\_twitter\_response*: This contains the JSON response text from the server. The value of this attribute can be explored using the JSON-Exploder or JSONExtractor transformers.

*\_twitter\_message\_id*: This contains the integer ID of the direct message.

## Parameters

### *Twitter Settings*

#### Message Text

The message text, or the name of an attribute containing the message text, can be entered in the Message Text parameter.

#### Message Recipient

Similarly, the message recipient, or the name of an attribute containing the recipient can be entered in the Message Recipient parameter. The value of this parameter can be a Twitter username or user ID number.

#### Twitter Username and Password

Enter a Twitter account username and password.

#### Include Geometry in Message Text

If this parameter is set to Yes, the #WKT hashtag will be appended to the message, along with the feature geometry, formatted as OGC Well-Known Text.

If this results in the message text being longer than 140 characters, the message text will be truncated, not the geometry. If geometry is included in the message text, the feature will be reprojected to the LL84 coordinate system before the message is sent. Features with no coordinate system are assumed to already be in the LL84 coordinate system.

### *Proxy Settings*

#### Proxy URL, Port, Username, Password, Authentication Method

These optional parameters may be set for organizations that require Internet access via an HTTP proxy server.

## Related Transformers

JSONExploder

JSONExtractor

## Transformer Category

Web Services



## Displacer

Solves proximity conflicts between features using a variant of the Nickerson displacement algorithm. This transformer is usually used after generalization.

### Input and Output Ports

The features routed into the transformer through the BASE port are geometrically frozen (cannot move).

The features routed in through the CANDIDATE port are compared against the BASE feature(s), displaced as necessary, and exit through the DISPLACED port. If no displacement occurred, they exit through the UNTOUCHED port.

Each comparison/displacement is independent of the others.

BASE features with geometries other than point, curve or area (polygon or ellipse or donut) will exit through the INVALID\_BASE port. Candidate features with geometries other than point, curve or simple area (polygon or ellipse) will exit through the INVALID\_CANDIDATE port.

The EXTRA\_BASE port holds extra BASE features as described in the Base Type parameter below.

### Parameters

#### Group By

You can use this option to narrow down which candidate features to compare with which base features.

#### Stiffness

Specifies how much the displacement at one point in the candidate feature's geometry should affect the neighboring points. A lower value means that the candidate geometry can be deformed easily, while a higher value means that it will try its best to keep its original shape.

#### Minimum Separating Distance

The Minimum Separating Distance parameter specifies the minimum separating distance between the candidate feature's geometry and the base feature's geometry after displacement.

#### Displace Endpoints

The Displace Endpoints parameter specifies whether or not to displace the endpoints of candidate features whose geometries are unclosed lines.

#### Base Type

The Base Type parameter specifies whether only a single BASE feature will be used, or whether all BASE features will be used. If Bases First is selected, then the transformer assumes that all BASE features will enter the transformer before any CANDIDATE features. Any further BASE features that arrive after the first CANDIDATE will be output through the EXTRA\_BASE port. The same goes for any BASE features after the first when in Single Base mode.

### Transformer Category

Manipulators

### Related Transformers

Generalizer

### FME Licensing Level

FME Professional edition and above

### Technical History

FME Factories Used: DisplacementFactory

## Dissolver

Dissolves area features by removing common boundaries to create larger areas.

### Input

- This transformer accepts two-dimensional polygonal features.
- Dissolved polygons are those polygons formed when shared edges between adjacent polygons are removed. The transformer assumes the input polygons are properly noded (that is, a vertex is present at each intersection point). This transformer also assumes that the polygons do not overlap each other.

### Output

- AREA: Dissolved Polygon features with specified attributes set appropriately.
- INTERIOR\_LINE: Linear features that represent the portions of the input polygons which are not part of the output dissolved polygon.
- NON\_POLYGON: Any features that are not polygons are output untouched.

### Parameters

#### Group By

The input polygonal features may be partitioned into groups for dissolving by using the Group By parameter. If this parameter is not specified, then all input features are processed together. The Group By parameter enables a single factory to perform the dissolve of several different, potentially overlapping, polygonal coverages.

By default, no attributes other than the Group By attributes are carried across from the input features to the output features. However, if any of the parameters have been set, then the output features will have the attributes of their constituent input features merged into them. In this case, the attributes of the largest constituent feature are copied to the output feature, and any different attributes found in any other constituent features are also added.

#### Overlapping Input

If Overlapping Input is set to No Overlap, then for two areas to be dissolved together, they MUST share a common boundary with identical vertices. For example, a small area entirely contained inside another area without sharing any edges will NOT be dissolved with the containing area.

If Overlapping Input is set to Overlap, then polygons being dissolved can overlap.

#### Attributes to Sum, Average, Weight Average

If desired, attributes can be added to the resulting area feature that contain the sum, average, or weighted (by area) average of some original attribute values.

#### Dissolve Member Count Attribute

The number of polygons dissolved into an output polygon is stored in the attribute identified by this parameter.

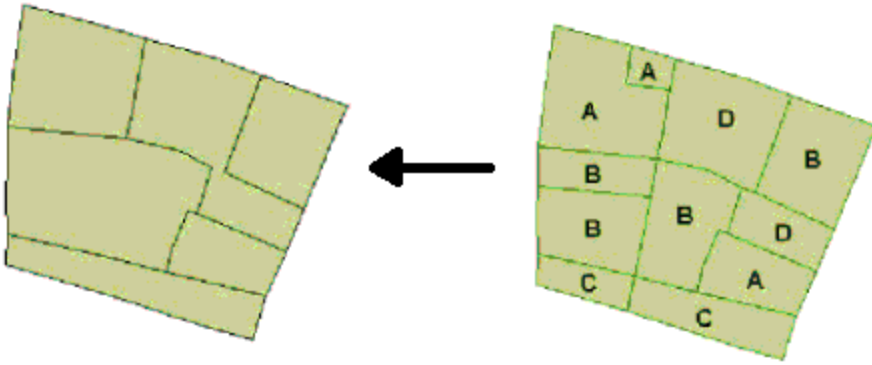
#### List Name

If the optional list name is supplied, a list is made of all the attributes of each area that is dissolved into an output area.

### Usage Notes

Aggregate polygons are deaggregated after entering the Dissolver. It is therefore possible that the number of features output will be greater than the number of features input.

## Example



## Geometry Handling

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Enhanced in the workspace, ellipses are accepted as area features; they are otherwise regarded as non-area features.

## fmepedia

See [fmepedia](#) for more information on this transformer.

## Transformer Category

Geometric Operators

## Technical History

Associated FME function or factory: PolygonDissolveFactory

## **DMS Calculator**

Calculates degrees, minutes, and seconds (DMS) from a decimal degrees value stored in an attribute.

### **Parameters**

#### Source Attribute

Choose the source attribute that contains the decimal degrees value.

#### Degrees, Minutes, Seconds Attributes

The resulting DMS values are stored in attributes that have these names.

If the decimal degrees value was negative, the resulting degrees value will be negative, but the minutes and seconds values will be positive.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Evaluate

## **DonutBridgeBuilder**

Builds connections between donut holes with the outer boundary of a donut, resulting in a polygon-equivalent representation of the input donut. A single, connected path visits the boundary and each donut hole exactly once. This action is performed on all donuts contained in an input feature.

The transformer outputs unmodified features at the INVALID port when a feature contains a donut with faulty topology (such as a hole lying outside of an outer boundary).

The generated polygon boundary is guaranteed to be non-self-intersecting if the input feature is properly oriented.

### **Parameters**

None

### **Transformer Category**

Geometric Operators

### **Technical History**

Associated FME function or factory: @Geometry

## DonutBuilder

Cuts holes in polygonal features by making polygons completely enclosed in other polygons into holes of the containing polygon.

The DonutBuilder assumes that the input area features are topologically clean and that no polygons overlap within a group.

Aggregate features are decomposed recursively to their components and non-area features will be output via the INVALID port.

### Parameters

#### Group By

Choose the attributes to group by.

#### Drop Holes

The Drop Holes parameter indicates whether or not features used to cut holes in containing features should themselves be dropped or output.

#### Hole Flag Attribute

This parameter will be added to each output feature and will contain "yes" if that feature was used to cut a hole into some containing feature, and "no" if that feature was not used as a hole.

Area features may be considered in groups based on the value of one or more attributes. Only features in the same group are then considered for hole nesting.

#### Hole List Name

If you specify a Hole List Name, a list will be created on each output donut containing an element for each input feature that became a hole, in the order that the holes appear on the donut.

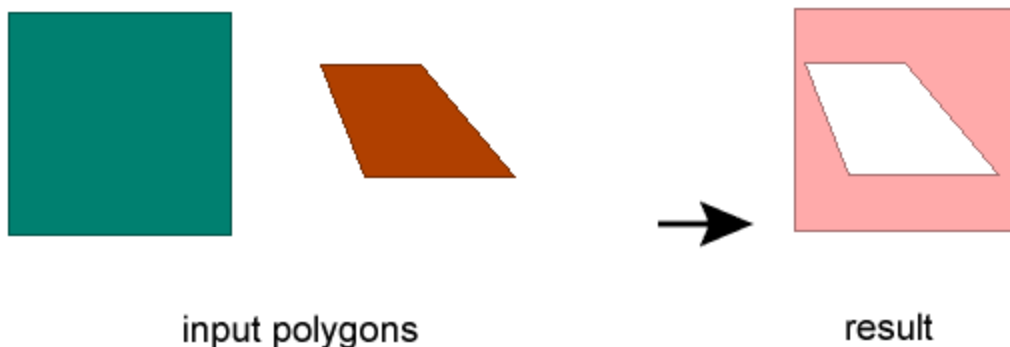
#### Preserve Internal Edges

Preserve Internal Edges (for advanced FME users) specifies that coordinate "cycles" within a polygon are allowable and will be preserved. A "cycle" is a line segment that occurs twice in the same polygon's boundary (once in each direction).

### Geometry Handling

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Enhanced in the workspace, ellipses will be handled as input polygons; they will otherwise be ignored.

### Example



### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: DonutFactory



## DonutHoleExtractor

Splits an area feature with holes into its component rings.

### Output Ports

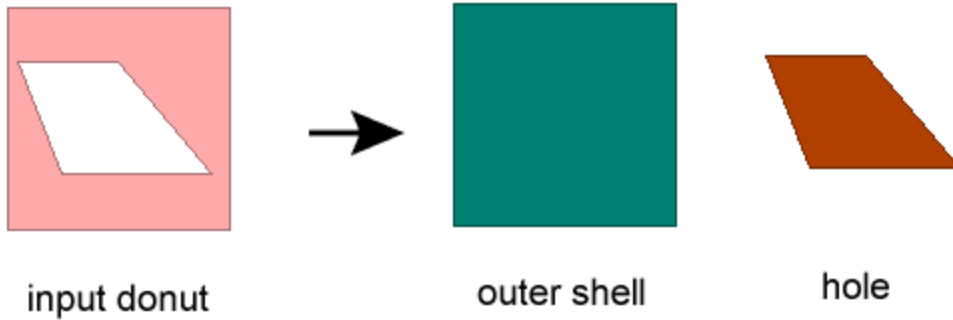
- **OUTERSHELL:** The outer boundary polygon is separated from the holes and output to this port. If an area had no holes, it is output untouched.
- **HOLE:** The individual holes are output as polygons.

Each output feature has all the attributes of the original area feature.

### Parameters

None.

### Example



### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: DonutHoleFactory



## **DuplicateCoordinateRemover**

Checks all elements of the geometry that are lines for duplicate coordinates.

Any consecutive coordinates with the same location will be reduced to a single coordinate. Only the coordinate locations are considered in comparing coordinates for equality. Measures are ignored.

### **Parameters**

Compare Z Values

If Yes, then coordinates will be considered equal only if their X, Y, and Z values all match; otherwise, only their X and Y values are considered.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @Geometry

## **DuplicateRemover**

Detects duplicate features based on the value of a key attribute.

### **Output Ports**

This transformer outputs to the DUPLICATE port any feature whose key attribute value is the same as one previously encountered. Any feature whose key value has not yet appeared is output via the UNIQUE port.

The first feature with a unique key value is output via the UNIQUE port – and subsequent features that have the same key value are output via the DUPLICATE port.

### **Parameters**

Key Attribute

Choose the feature that contains the key attribute value.

### **Usage Notes**

This is a more specific type of operation than that provided by the *Matcher*, which can match on multiple attributes as well as geometry. For such situations, this transformer is dramatically more efficient than the *Matcher*.

### **Transformer Category**

Filters

### **Technical History**

Associated FME function or factory: TestFactory, @GlobalVariable

## DWGStyler

Prepares features for output to AutoCAD DWG/DXF by providing a convenient interface to set a variety of AutoCAD DWG/DXF format-specific attributes.

### Parameters – Color

Color

This parameter specifies the pen color that will be used to render the feature. The pen color determines the color of lines and area boundaries.

To edit this parameter, click the browse button to the right of the text field.

Format attribute set: `fme_color`

Note that when this parameter is specified, the `autocad_color` attribute is removed from the feature and `fme_color` is set. This is because `autocad_color` will override `fme_color` if they are both present.

### Parameters – Block/Style Name Source

DWG/DXF Template File

This optional parameter is used to specify the name of an existing DWG or DXF file that will be used by the transformer as a source for block names, linetypes, and text shape names. It is not used by the actual DWG/DXF writer – the template file must be specified in the writer's parameters and should contain the same block names and linetypes as the file specified here. Usually the same file will be used in both the transformer and the writer.

If no blocks or linetypes will be used, this parameter can be left blank.

### Parameters – Blocks

If this section is active, point features will be turned into inserts and given a block name, rotation, and size.

Features with other geometry types are not affected by settings in this section.

Format attributes set: `autocad_entity` to `autocad_insert`

Block Name

Block Name specifies the block that will be placed at the point location. Click the browse button to pick the name from the set of blocks defined in the DWG/DXF template file.

Format attribute set: `autocad_block_name`

Block Rotation

Block Rotation specifies rotation of the block (in degrees counterclockwise from horizontal). This can be set to any floating point value between -360.0 to 360.0, or taken from an attribute.

Format attribute set: `autocad_rotation`

Block Size By

Block Size By indicates how the size of the block will be specified. If ground units are chosen, then the block will be scaled so that its range in x, y, and z covers the specified amounts. If scale factor is chosen, then the amounts in x, y, and z specify the scaling which will be applied to the block in each of the three axes.

X

X specifies the sizing amount for the x axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attribute set: `autocad_x_scale` (if Block Size By is Scale Factor); `autocad_size_x` (if Block Size By is Ground Units)

Y

Y specifies the sizing amount for the y axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attribute set: **autocad\_y\_scale** (if Block Size By is Scale Factor); **autocad\_size\_y** (if Block Size By is Ground Units)

Z

Z specifies the sizing amount for the z axis, which can be a floating point value or taken from an attribute. Its meaning is dependent on the Block Size By setting described above.

Format attribute set: **autocad\_z\_scale** (if Block Size By is Scale Factor); **autocad\_size\_z** (if Block Size By is Ground Units)

## Parameters – Lines

If this section is active, linear features will be prepared for output to AutoCAD DWG/DXF.

Features with other geometry types will not be affected by settings in this section.

### Linetype

Linetype specifies the AutoCAD linetype that will be used to render the line. Click the browse button to pick the name from the set of blocks defined in the DWG/DXF template file.

Format attribute set: **autocad\_linetype**

### Linetype Generation

Linetype Generation specifies how linetypes will be rendered at each vertex of the line. If set to Continuous, line vertices are ignored since the pattern is applied to the line when it is rendered. If set to Restart At Each Vertex, the pattern is restarted at each vertex of the line.

Format attribute set: **autocad\_linetype\_generation**

### Line Weight

Line Weight specifies the thickness of the line when it is rendered, measured in hundredths of a millimeter.

Format attribute set: **autocad\_lineweight**

### Line Scale

Line Scale specifies the amount that the line pattern will be scaled when the line is rendered.

Format attribute set: **autocad\_linetype\_scale**

### Line Elevation

Line Elevation sets a single elevation value that will be applied to all the vertices of the line. This is an efficient way to set a constant elevation (z) to an otherwise two-dimensional line.

Format attribute set: **autocad\_elevation**

## Parameters – Areas

If this section is active, then area features will be prepared for output to AutoCAD DWG/DXF.

Features with other geometry types will not be affected by settings in this section.

### Area Entity Type

Area Entity Type specifies the kind of entity and fill pattern that will be created to represent the feature when it is written to AutoCAD DWG/DXF. The choices with the corresponding AutoCAD entity type are:

Area Entity Type Option	autocad_entity
Hatch With Fill Pattern	autocad_hatch
Hatch With Gradient Pattern	autocad_hatch
MPolygon With Fill Pattern	autocad_mpolygon

Area Entity Type Option	autocad_entity
MPolygon With Gradient Pattern	autocad_mpolygon
Polygon	autocad_polygon

Format attribute set: **autocad\_entity**

#### Fill Pattern Name

Fill Pattern Name sets the name of the fill pattern that will be used to fill either the hatch or mpolygon. Click the browse button to pick the name from the set of fill patterns that ship with AutoCAD.

Format attributes set: **autocad\_hatch\_pattern\_name** (if Area Entity Type is set to Hatch With Fill Pattern); **autocad\_mpolygon\_pattern\_name** (if Area Entity Type is set to Mpolygon With Fill Pattern)

#### Pattern Scale

Pattern Scale specifies the amount that the fill pattern will be scaled when the area is rendered.

Format attributes set: **autocad\_hatch\_pattern\_scale** (if Area Entity Type is set to Hatch With Fill Pattern); **autocad\_mpolygon\_pattern\_scale** (if Area Entity Type is set to Mpolygon With Fill Pattern)

#### Gradient Name

Gradient Name sets the type of the gradient pattern (predefined as part of AutoCAD) that will be used to fill either the hatch or mpolygon.

Format attribute set: **autocad\_hatch\_gradient\_name** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_name** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

#### Gradient Fill Color 1

Gradient Fill Color 1 sets the first color to be used in the gradient pattern that will be used to fill either the hatch or mpolygon.

Format attributes set: **autocad\_hatch\_gradient\_color1** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_color1** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

#### Gradient Fill Color 2

Gradient Fill Color 2 sets the first color to be used in the gradient pattern that will be used to fill either the hatch or mpolygon.

Format attributes set: **autocad\_hatch\_gradient\_color2** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_color2** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

#### Fill Angle

Fill Angle sets the angle of the gradient pattern that will be used to fill either the hatch or mpolygon.

Format attributes set: **autocad\_hatch\_gradient\_angle** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_angle** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

#### Fill Origin X

Fill Origin X sets the x coordinate of the origin for the gradient pattern that will be used to fill either the hatch or mpolygon.

Format attributes set: **autocad\_hatch\_gradient\_origin\_point\_x** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_origin\_point\_x** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

#### Fill Origin Y

Fill Origin Y sets the y coordinate of the origin for the gradient pattern that will be used to fill either the hatch or mpolygon.

Format attributes set: **autocad\_hatch\_gradient\_origin\_point\_y** (if Area Entity Type is set to Hatch With Gradient Pattern); **autocad\_mpolygon\_gradient\_origin\_point\_y** (if Area Entity Type is set to Mpolygon With Gradient Pattern)

### Parameters – Text

If this section is active, text features will be prepared for output to AutoCAD DWG/DXF.

Note that the text string, size, and rotation is part of the text feature and cannot be set by this transformer: use the `TextPropertySetter` to adjust or set these properties.

Features with other geometry types are not affected by settings in this section.

## Text Entity Type

Text Entity Type specifies the kind of entity that will be created to represent the feature when it is written to AutoCAD DWG/DXF. The choices with the corresponding AutoCAD entity type are:

Area Entity Type Option	autocad_entity
Text	autocad_text
Multi-Text	autocad_multi_text

Format attribute set: `autocad_entity`

## TrueType Font Name

TrueType Font Name sets the name of the font that will be used to draw the text while Multi-Text is chosen for the entity type. Click the browse button to pick the font name.

Format attribute set: `autocad_mtext_string`

## Text Shape Name

Text Shape Name sets the name of the shape that will be used to draw the text while Text is chosen for the entity type. Click the browse button to pick the shape name from those defined in the DWG/DXF Template File.

Format attribute set: `autocad_shape_name`

## Text Justification

Text Justification sets the justification for text placement.

Format attribute set: `autocad_justification`

## Text Size

Text Size sets the size of the text that is placed.

Format attribute set: `fme_text_size`

## Text

Text sets the text string that will be placed.

Format attribute set: `fme_text_string`

## Additional References

For more information about DWG/DXF styling:

- See the Autodesk AutoCAD DWG/DXF Reader/Writer > Feature Representation section in the *FME Readers and Writers* manual. In Workbench, select Help > FME Readers and Writers Reference.

## Transformer Category

Stylers

## **ElevationExtractor**

Extracts the elevation of the first coordinate and assigns it to the named attribute.

### **Parameters**

Elevation Attribute

If the feature was two-dimensional, 0 will be assigned to the elevation attribute.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @ZValue

## **EllipsePropertyExtractor**

Sets the given attributes to the properties of an ellipse geometry.

### **Parameters**

#### Primary Radius Attribute

Attribute name that is set to the length of the primary radius for the ellipse upon which the arc is based.

#### Secondary Radius Attribute

Attribute name that is set to the length of the secondary radius for the ellipse upon which the arc is based.

#### Rotation Attribute

Attribute name that is set to the rotation of the ellipse that defines the arc. The rotation angle specifies the angle in degrees from the horizontal axis to the primary axis in a counterclockwise direction.

#### Start Angle Attribute

Attribute name that is set to the parametric angle of the start point of the arc, in degrees.

#### Sweep Angle Attribute

Attribute name that is set to the parametric sweep angle of the arc, in degrees.

#### Start X Attribute

Attribute name that is set to the x coordinate value for the arc's start point.

#### Start Y Attribute

Attribute name that is set to the y coordinate value for the arc's start point.

#### Start Z Attribute

Attribute name that is set to the z coordinate value for the arc's start point.

#### End X Attribute

Attribute name that is set to the x coordinate value for the arc's end point

#### End Y Attribute

Attribute name that is set to the y coordinate value for the arc's end point

#### End Z Attribute

Attribute name that is set to the z coordinate value for the arc's end point

#### Center X Attribute

Attribute name that is set to the X coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

#### Center Y Attribute

Attribute name that is set to the Y coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

#### Center Z Attribute

Attribute name that is set to the Z coordinate of the center of the ellipse. If not specified, the first coordinate of the feature is used.

## **Transformer Category**

### Manipulators



## **Technical History**

Associated FME function or factory: @Geometry

## EllipsePropertySetter

Sets the properties of an ellipse geometry.

### Parameters

Each parameter may either be entered as a number, or taken from the value of a feature attribute by selecting the attribute name from the pull-down list. All parameters are optional. If a value is unspecified, it will be left unmodified on the geometry.

#### New Primary Axis and New Secondary Axis

The primary and secondary axes set the radii of the arc. Note that the primary axis need not be larger than the secondary, however, the rotation angle is always measured from the x axis to the primary axis. To make a circular arc, set both the primary and secondary axis to the same value.

#### New Rotation Angle

The rotation angle is measured in degrees counterclockwise from horizontal, and measures the rotation of the primary axis from horizontal.

#### New Orientation

This parameter can adjust the orientation of a feature to follow the right-hand rule or the left-hand rule<sup>1</sup>.

#### Center X, Center Y, Center Z

The center x and y parameters set the origin of the arc. If these values are blank, and the input features are points, the existing feature x/y values will determine the center of the arcs. If the parameter values are blank, and the input features are not points, the operation is undefined.

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Geometry

<sup>1</sup>[http://en.wikipedia.org/wiki/Right-hand\\_rule](http://en.wikipedia.org/wiki/Right-hand_rule)

## **EnvironmentVariableFetcher**

Fetches the specified environment variable and includes it in a new attribute.

Environment variables are convenient especially if certain standard directories and parameters need to be referenced in a workspace, but the actual locations or names can vary from computer to computer.

The available variables will depend on which platform you are using. For example in Windows you can find a list of variables on your system by opening Control Panel > System > Advanced (or Advanced system settings link in Vista)> Environment Variables.

### **Parameters**

Environment Variable

Type the environment variable name (for example, PATH or TEMP).

Attribute

If the specified environment variable does not exist, a null string is put into the given attribute.

### **Usage Notes**

Within an FME workspace you could use the value retrieved to:

- Define the output location of a dataset (in conjunction with Feature Type Fanout, which is described in the Workbench help files)
- Define the location of a file for the AttributeFileWriter or AttributeFileReader
- Define the location of a workspace to run with the WorkspaceRunner
- Write the value to some type of metadata output

### **Transformer Category**

Calculators

### **Transformer History**

This transformer has been renamed from EnvironmentVariableRetriever.

### **Technical History**

Associated FME function or factory: @Tcl2

## ESRIReprojector

Reprojects feature coordinates from one coordinate system to another using the ESRI reprojection library.

This transformer always reprojects from the source coordinate system to the destination coordinate system, tagging the features with the destination coordinate system on output. Any coordinate system set on the input features is ignored.

### Parameters

#### Source Coordinate System

Click the Browse button to display the ESRI reprojection library. You can select a coordinate system, import a coordinate system, or create a new coordinate system.

#### Destination Coordinate System

Click the Browse button to display the ESRI reprojection library. You can select a coordinate system, import a coordinate system, or create a new coordinate system.

#### Geographic Transformation

A geographic transformation converts data between two geographic coordinate systems. [Click here](#) to see a PDF of valid geographic transformation names for this parameter.

**Note:** The geotransformation name is case-sensitive. If you do not enter the parameter in the correct case, FME will fail and produce an error saying it cannot locate the transformation parameter.

#### Geographic Transformation Direction

This is the geotransformation direction (either "forward" or "reverse").

#### Interpolation Type (Raster)

The Interpolation Type affects only raster data. Cell values are interpolated in order to change the raster to the specified size.

- **Nearest Neighbor** is the fastest but produces the poorest image quality.
- **Bilinear** provides a reasonable balance of speed and quality.
- **Bicubic** is the slowest but produces the best image quality.
- **Average 4** and **Average 16** have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Cell Size (Raster)

The Cell Size applies only to raster features.

- **Square Cells:** The number of rows and columns as well as the spacing will be changed to maintain approximately the same cell ground area and form square cells where the horizontal and vertical cell sizes are equal. Like the Square Cells option, Preserve Cells will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.
- **Stretch Cells:** The cell size of the raster will be adjusted to maintain the same number of rows and columns in the reprojected raster as there were in the input raster.

### Usage Notes

This transformer is not affected by raster band and palette selection.

### Category

Coordinate Systems

### Dependencies

To use this transformer, you must have ArcGIS installed on the same machine as FME.

**Licensing Level**

This transformer is not available in FME Base Edition.

**Technical History**

Associated FME function or factory: @Reproject

## ExpressionEvaluator

Evaluates an arbitrary Tcl 8.5.2 expression and returns the result in a new attribute. The operators permitted in the expressions to be evaluated are a subset of the operators permitted in C expressions. They have the same meaning and precedence as the corresponding C operators.

Expressions usually yield numeric results, such as integer or floating-point values. For example, the expression

```
8.2 + 6
```

returns 14.2.

The ExpressionEvaluator is based on the FME @Evaluate function, which in turn is based on the Tool Command Language (Tcl) expr command, the documentation of which is provided below.<sup>1</sup>

Note: It is easy to build an invalid expression, so you may have to double check that what you build makes sense.

## Operands

An expression consists of a combination of operands, operators, and parentheses. White space may be used between the operands, operators, and parentheses as it is ignored by the expression processor. Where possible, operands are interpreted as integer values. Integer values may be specified in decimal which is the normal case, in octal if the first character of the operand is 0, or in hexadecimal if the first two characters of the operand are 0x. If an operand does not have one of the integer formats given above, then it will be treated as a floating-point number if possible. Floating-point numbers may be specified in any of the ways accepted by an ANSI-compliant C compiler, except that "f", "F", "l", and "L" suffixes are not permitted in most installations. For example, all of the following are valid floating-point numbers: 2.1, 3., 6e4, 7.91e+16. If no numeric interpretation is possible, then an operand is left as a string and only a limited set of operators may be applied to it.

Operands may be specified in any of the following ways:

- As a numeric value, either integer or floating-point.
- As a value of an FME feature attribute, using standard & notation. The attribute's value is used as the operand.
- As an FME feature function, such as @Area(). The function is evaluated and the result used as the operand.
- As a mathematical function whose arguments have any of the above forms for operands, such as sin(&x). See the table below for a list of defined mathematical functions.

## Operators

The valid operators listed below are grouped in decreasing order of precedence:

Function	Description
**	Exponentiation. Valid for numeric operands only.
- + ~ !	Unary minus, unary plus, bit-wise NOT, logical NOT. None of these operands may be applied to string operands, and bit-wise NOT may be applied only to integers.
* / %	Multiply, divide, remainder. None of these operands may be applied to string operands, and remainder may be applied only to integers. The remainder always has the same sign as the divisor and an absolute value smaller than the divisor.
+ -	Add and subtract. Valid for any numeric operands.

<sup>1</sup>Tcl and its documentation is copyrighted by the Regents of the University of California, Sun Microsystems, Inc. and other parties. However, the authors have granted permission to any party to reuse and modify the code and documentation, provided the original copyright holders are acknowledged.

Function	Description
<< >>	Left and right shift. Valid for integer operands only.
< > <= >=	Boolean less, greater, less than or equal, and greater than or equal. Each operator produces 1 if the condition is true, 0 otherwise. These operators may be applied to strings as well as numeric operands, in which case string comparison is used.
== !=	Boolean equal and not equal. Each operator produces a zero/one result. Valid for all operand types.
&	Bit-wise AND. Make sure that there is a space between the operand and the operator (e.g., 5 & 3 is valid; 5&3 is not valid). Valid for integer operands only.
^	Bit-wise exclusive OR. Valid for integer operands only.
	Bit-wise OR. Valid for integer operands only.
&&	Logical AND. Produces a 1 result if both operands are non-zero, 0 otherwise. Valid for numeric operands only (integers or floating-point).
	Logical OR. Produces a 0 result if both operands are zero, 1 otherwise. Valid for numeric operands only (integers or floating-point).
x?y:z	If-then-else, as in C. If x evaluates to non-zero, then the result is the value of y. Otherwise, the result is the value of z. The x operand must have a numeric value.

See the C Manual for more details on the results produced by each operator. All binary operators group left-to-right within the same precedence level. For example, the expression:

```
4*2 < 7
```

returns 0.

## Math Functions

The ExpressionEvaluator supports the following mathematical functions in expressions. Each of these functions invokes the C math library function of the same name. Refer to the C Manual entries for the library functions for details on what they do.

Function	Description
acos(arg)	Returns the arc cosine of arg, in the range [0,pi] radians. Arg should be in the range [-1,1].
asin(arg)	Returns the arc sine of arg, in the range [-pi/2,pi/2] radians. Arg should be in the range [-1,1].
atan(arg)	Returns the arc tangent of arg, in the range [-pi/2,pi/2] radians.
atan2(x, y)	Returns the arc tangent of y/x, in the range [-pi,pi] radians. x and y cannot both be 0.
ceil(arg)	Returns the smallest integer value not less than arg.
cos(arg)	Returns the cosine of arg, measured in radians.
cosh(arg)	Returns the hyperbolic cosine of arg. If the result would cause an overflow, an error is returned.
exp(arg)	Returns the exponential of arg, defined as e**arg. If the result would cause an overflow, an error is returned.
floor(arg)	Returns the largest integer value not greater than arg.

Function	Description
fmod(x, y)	Returns the floating-point remainder of the division of x by y. If y is 0, an error is returned.
hypot(x, y)	Computes the length of the hypotenuse of a right-angled triangle ( $\sqrt{x^2+y^2}$ ).
log(arg)	Returns the natural logarithm of arg. Arg must be a positive value.
log10(arg)	Returns the base 10 logarithm of arg. Arg must be a positive value.
pow(x, y)	Computes the value of x raised to the power y. If x is negative, y must be an integer value.
sin(arg)	Returns the sine of arg, measured in radians.
sinh(arg)	Returns the hyperbolic sine of arg. If the result would cause an overflow, an error is returned.
sqrt(arg)	Returns the square root of arg. Arg must be non-negative.
tan(arg)	Returns the tangent of arg, measured in radians.
tanh(arg)	Returns the hyperbolic tangent of arg.

The ExpressionEvaluator also implements the following functions for conversion between integers and floating-point numbers:

Function	Description
abs(arg)	Returns the absolute value of arg. Arg may be either integer or floating-point, and the result is returned in the same form.
double(arg)	If arg is a floating value, returns arg. Otherwise converts arg to floating point and returns the converted value.
int(arg)	If arg is an integer value, returns arg. Otherwise converts arg to integer by truncation and returns the converted value.
round(arg)	If arg is an integer value, returns arg. Otherwise converts arg to integer by rounding and returns the converted value.

### Types, Overflows, Precision

All internal computations involving integers are done with C-type long, and all internal computations involving floating-point are done with C-type double. When converting a string to floating-point, exponent overflow is detected and results in an error. For conversion to integer from string, detection of overflow depends on the behavior of some routines in the local C library, so it should be regarded as unreliable. In any case, integer overflow and underflow are generally not reliably detected for intermediate results. Floating-point overflow and underflow are detected to the degree supported by the hardware, which is generally reliable.

Conversion among internal representations for integer, floating-point, and string operands is automatically done as needed. For arithmetical computations, integers are used until some floating-point number is introduced, after which floating-point is used. For example,

```
5 / 4
```

returns 1, while

```
5 / 4.0
```

and

```
5 / (4 + 0.0)
```

both return 1.25. Floating-point values are always returned with a "." or an "e" so that they will not look like integer values. For example,

```
20.0/5.0
```

returns "4.0", not "4".

Seventeen digits of precision are always used for floating point calculations.

### Example

fmepedia has a good example of the ExpressionEvaluator.



**Transformer Category**

Calculators

**Technical History**

Associated FME function or factory: @Evaluate

## Extender

Creates two-point extensions to linear features that extend the feature by a user-specified length.

This transformer can also output the original feature with the first and last segments stretched by a user-specified amount. Each of the created features gets a copy of all attributes of the original feature, including the feature type. Arcs that are input are converted to lines before processing.

### Output Ports

- **BEGINNING:** This transformer can create an extension of the first segment in the input feature. This feature's orientation is the same as the first segment of the input feature and its end point is the same as the input feature's start point. This new feature is output to the BEGINNING port.
- **END:** This transformer can create an extension of the last segment of the input feature. This feature's orientation is the same as the last segment of the input feature and its start point is the same as the input feature's end point. The feature holding this segment is output to the OUTPUT port.
- **STRETCHED:** This transformer can create a duplicate of the input feature except that the first and last segments are extended in their respective orientation directions. The length of these extensions is also controlled by the Extension Length parameter. In this case, the end nodes of the line are moved; no new nodes are added. The feature holding this segment is output via the STRETCHED port.

### Parameters

#### Extension Length

The extension length parameter is measured in ground units. You can enter a number, or the value can be taken from an existing feature attribute (select the attribute name from the pull-down list).

#### Segments to Average

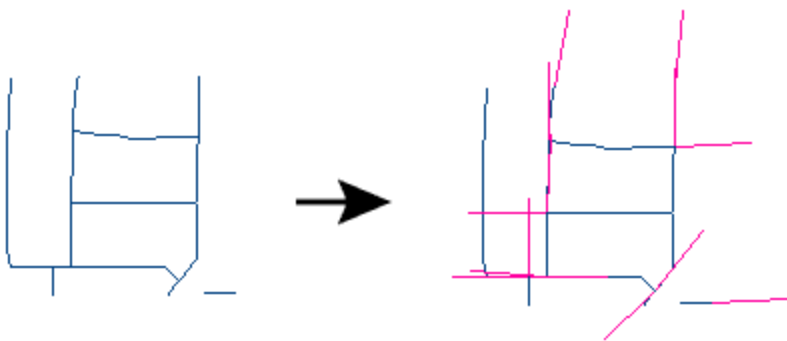
This parameter specifies the number of segments that should be considered when computing the orientation angle for the extension feature. By default, this is set to 1, which means the orientation of the extension feature matches the orientation of just one segment in the original feature. It can be set to any number of segments, in which case the orientation will be set to the average orientation of those segments. If the number of segments is larger than the number of segments available on the feature, then the entire feature orientation is averaged and used.

You can enter a number, or the value can be taken from an existing feature attribute (select the attribute name from the pull-down list).

### Geometry Handling

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Enhanced in the workspace, arcs are extended as arcs instead of being stroked. The two point line segments are output normally, but the stretched version is converted to a path with three segments: the start line, the arc itself, and the end line.

### Example



### Transformer Category

Manipulators

**Usage Notes**

This transformer can be used in combination with the Snapper to perform a simple form of data cleaning.

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: ExtensionFactory

## Extruder

Creates long, surface or solid geometries with a fixed cross-sectional profile taken from the original geometry of the feature.

### Input

- If the geometry of the input feature was a face or an area, then the geometry is replaced with a solid.
- If the geometry of the input feature was linear, then the result is a composite surface geometry.

### Parameters

#### Extrusion Input By

The amount of extrusion can be set either by height (in which case the extrusion is applied in the positive direction along the z axis), or an extrusion vector (which allows for extrusion in an arbitrary direction).

#### Extrusion Height

You can enter the extrusion height, or the value can be taken from an existing feature attribute (select the attribute name from the pull-down list).

#### Extrusion Vector

You can enter the x, y or z components of the extrusion vector, or the values can be taken from existing feature attributes (select the attribute names from the pull-down list).

### Usage Notes

This transformer has no effect on features that have a geometry other than a face, an area or a line.

### Example



### Transformer Category

3D

### Technical History

Associated FME function or factory: @GeometryType

## FaceReplacer

Replaces the geometry of a feature from donut or polygon to face. If the donut or polygon is not already three-dimensional, a 0.0 value for Z coordinates is assumed.

A face is a planar area in 3D space. The planar structure can be a raster, polygon or a donut.

For a raster, the transformer will replace the geometry with a face with the same bounding box as the original raster with a textured appearance using the original raster, as applied from the top view.

The planar area has a concept of a surface normal, a vector that points outwards perpendicular from the area. The direction of the surface normal in a face is determined by using the right-hand rule: if the fingers of your right hand curl along the order of the vertices, the direction that the thumb points to is the direction of the surface normal.

## Parameters

### Check for Planarity

If you choose No, then it does not check if the donut or polygon is planar. If you choose Yes, then you can enter either a float value or an attribute for the Planarity Tolerance Value. If the area is not planar given the specified tolerance value, then the area is not replaced by a face.

### Planarity Tolerance Value

Enter a value or choose an attribute to specify the positive (greater than or equal to 0) distance a point can be displaced in a perpendicular direction away from the plane and still be considered in plane. For example, a tolerance of 0 indicates that each point must be exactly in the plane for the area to be considered planar, while a tolerance of 3 means that no point may be more than 3 ground units away from the plane for the area to be considered planar.

## Usage Notes

- This transformer has no effect on features that have geometries other than raster, donut and polygon.
- You can use the 3DForcer to turn 2D geometry into 3D geometry.

## Transformer Category

3D

## Technical History

Associated FME function or factory: @GeometryType

## **FeatureHolder**

Stores incoming features until they have all arrived, and then releases them in their original order.

Use this transformer whenever you want to detain certain features until all input features have arrived. This ensures that no additional processing is performed until all features are released.

## **Transformer Category**

Collectors

## **Technical History**

Associated FME function or factory: SortFactory

## FeatureMerger

Moves the attributes/geometry from one feature to another feature. Features that contain the desired attributes/geometry are connected through the SUPPLIER port, and the features that will receive the attributes/geometry are connected through the REQUESTOR port.

When a Requestor finds a Supplier, then the attributes from the Supplier are merged onto the Requestor. If the Requestor already had an attribute that the Supplier also had, the Requestor's original value for that attribute is preserved. A single Supplier may be used by many Requestors.

Requestor features are joined to Supplier features when their respective Join Attributes have the same value.

### Input

- SUPPLIER: The source of new attributes for features that enter through the REQUESTOR port.
- REQUESTOR: Receives the new attributes from the features connected to the SUPPLIER port.

### Output Ports

- COMPLETE: Requestors that find a Supplier.
- INCOMPLETE: Requestors that do **not** find a Supplier.
- EMPTY: Requestors that have no value for the Join Attribute.
- REFERENCED: Suppliers that are used by at least one Requestor.
- UNREFERENCED: Suppliers that are **not** used by any Requestor.
- DUPLICATE\_SUPPLIER: Supplier features with the same value as the Reference attributes. Note that no Duplicate Suppliers will be output if a *List Name* is specified or *Process Duplicate Suppliers* is set to No.

### Parameters

#### Group By

The input features may be partitioned by the Group By parameter. If you choose any Group By attributes, then references between features will only be resolved if they share a common value for the selected attributes.

If you do not choose any Group By attributes, all features are processed together.

If you have more than one Reader, a typical use is to group by reader\_id to ensure that references are resolved within the correct set of features.

#### Merge Type

You would use this transformer if you have a group of features that you want to combine with another group of features to get either their geometry or their attributes, or both. These choices are determined through the Merge Type parameter:

- When the Merge Type parameter is **Attributes Only**, then the Suppliers contain attribution that will be joined to the attribution of the Requestor features.
- When the Merge Type parameter is **Geometry**, then the Suppliers are the features that contain the geometry. Note that the Requestor will lose its former geometry.
- When the Merge Type parameter is **Attributes and Geometry**, then the Suppliers contain both geometry and attribution that will be joined to the Requestor features. Any geometry on the Requestor will be overwritten. Attributes on the Requestor may or may not be overwritten, depending on the settings for the *Process Duplicate Suppliers* and *List Name* parameters.

**Attributes Only:** Existing attributes on a Requestor feature will never be overwritten by attributes of the same name on any Supplier features. Note that any fme\_\* attributes present on the Supplier features will not be transferred to the Requestor feature.

**Build Polygons:** If the Suppliers consist exclusively of polygon and donut polygon features, any common border segments will be removed. If the Suppliers contain at least one non-donut or non-polygon feature, then the transformer will form polygons and donuts from the Suppliers

and will join connected line segments of the Supplier features before setting the geometry of the Requestor feature. In this case, the geometry may be an aggregate if several disjoint geometries were created.

**Build Aggregates:** The transformer will create an aggregate of the geometries of the Supplier features. (If there is only one Supplier feature, then the Requestor geometry will be an aggregate with one element.)

**Build Lines from Points:** The transformer will connect the points of the Supplier features into lines. Note that any non-point features that are referenced will be ignored when building lines.

### Join Attribute

Normally the Suppliers all have unique values in their Join Attribute, and any duplicate Suppliers are ignored by the transformer. However, if the *Process Duplicate Suppliers* parameter is set to Yes, then all Suppliers that match the Requestor's Join Attribute value will be combined onto that Requestor.

### List Name

Note that if a List Name is specified, then duplicate Suppliers will automatically be allowed regardless of the parameter setting. If a List Name is specified, then any Suppliers that are combined with a Requestor will have their attributes added to the specified list on the Requestor, in addition to being merged onto the Requestor feature.

### Build Incomplete Requestors

When multiple Suppliers are associated with one Requestor, this parameter governs what happens to Requestors that cannot locate all of their Suppliers.

If this parameter is set to Yes, then the Suppliers that were found will be combined onto the Requestor and it will be output (after being changed) via the INCOMPLETE port. The Suppliers used will be output via the REFERENCED port. If this parameter is set to No, then the Requestor will be output untouched via the Incomplete port and the Suppliers will be output via the UNREFERENCED port.

### Process Duplicate Suppliers

If more than one Supplier is found for a given Requestor, and **Process Duplicate Suppliers** is NO (and a List Name is not given), then every Supplier after the first is output via the DUPLICATE\_SUPPLIER port. Only the first of the Suppliers will be matched with a Requestor. If set to Yes, then the duplicate Suppliers are all matched with the corresponding Requestor.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, the merged geometry preserves arcs, ellipses and text; otherwise, they appear as points in the merged geometry.

### Transformer Category

Collectors

### Technical History

Associated FME function or factory: ReferenceFactory



## FeatureReader

Performs queries against any FME-supported format. The queries can have both a spatial and a nonspatial component.

One query is issued to the FME-supported format for each feature that enters the transformer. The results of the query are then output.

### Input Parameters

Select the Reader format and dataset, including any reader-specific parameters

### Feature Types to Read

Feature types can either be specified from the wizard or from a chosen attribute. The name of the chosen attribute should be specified in "Query the feature types specified in the attribute below". The chosen attribute should contain a colon-separated list of feature types to be queried.

### Query Operation

Select to query the feature types specified in the previous pane, or select the feature types from an attribute.

Query the feature types specified in the attributes below

### Spatial Interaction

- Select all features
- Select only features that intersect with the TRIGGER feature's bounding box
- Select only features that have the following spatial relationship with the TRIGGER feature. Note that any spatial predicates supported natively are listed first, and will likely have better performance than the generic predicates.

INTERSECTS	Features output overlap and their boundaries intersect in one or more places with the TRIGGER feature's bounding box.
CONTAINS	Features output have their geometry entirely contained in the TRIGGER feature's bounding box.
CROSSES	Features output intersect. Or in the case of line/line, the intersection of the interiors forms a point.
DISJOINT	Features output have no common boundary or interior points with the TRIGGER feature's bounding box.
EQUALS	Features output share every point of their boundaries and interior, including any hole, with the TRIGGER feature's bounding box.
OVERLAPS	Features output overlap but their boundaries do not interact with the TRIGGER feature's bounding box.
TOUCHES	Features output share a common boundary with the TRIGGER feature's bounding box.
WITHIN	The interiors of the TRIGGER features intersect.

### Advanced Options

The TRIGGER feature defines the geometry which will be used to define the spatial component of the query, unless it does not contain any geometry. In this case, only an attribute query as defined by the WHERE clause will be executed.

### Attribute Handling Parameter Settings

- **Result Attributes Only:** The result feature attributes are based solely on the query results.
- **Keep Trigger Attributes if Conflict:** The result feature attributes are a combination of both the query results and the TRIGGER feature's attributes. If there is a conflict, attribute values are taken from the query feature.

- **Keep Result Attributes if Conflict:** The result feature attributes are a combination of both the query results and TRIGGER feature's attributes. If there is a conflict, attribute values taken from the query results.

### **Geometry Handling Parameter Settings**

- **Result Geometry Only:** The result feature geometry is taken from the query results.
- **Trigger Geometry Only:** The result feature geometry is taken from the query feature.
- **Aggregate Trigger and Result Geometry:** The result feature geometry is an aggregate of the geometry from the query feature followed by the geometry from the query results.

### **Transformer Category**

Database

### **Technical History**

Associated FME function or factory: QueryFactory

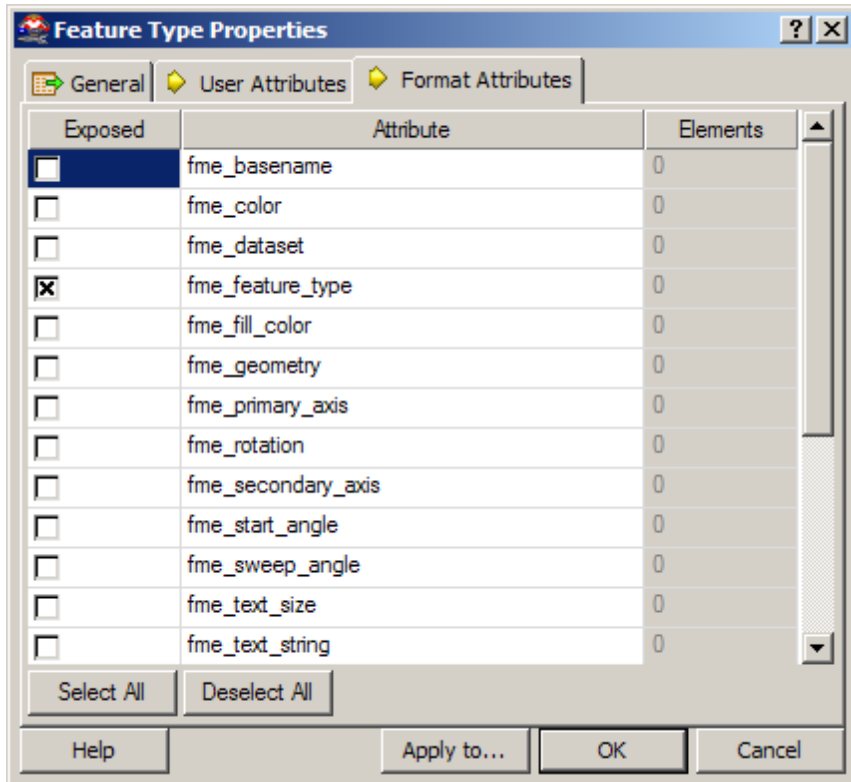
## FeatureTypeExtractor

Adds an attribute containing the original feature type of a feature.

### Usage Notes

The feature may have lost its original feature type if it was already processed through a transformer that does not preserve attributes (and you did not Group By that feature type). In this case, the attribute added will have an empty string as its value.

Exposing the `fme_feature_type` attribute on the original source feature type may be an easier and more reliable way to work with feature types. Click the Properties button on the original feature type, and choose the Format Attributes tab:



### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: @SupplyAttributes

## FeatureTypeFilter

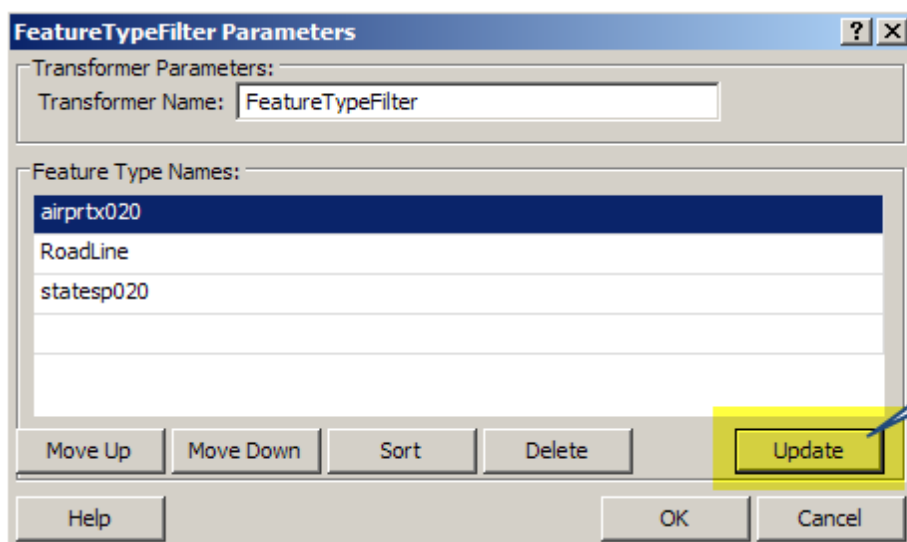
Routes input features to different output ports based on their original feature type. This transformer is especially useful when you have combined several input features for a particular operation (for example, a Clipper) but want to filter them afterwards in order to route them differently.

### Output Ports

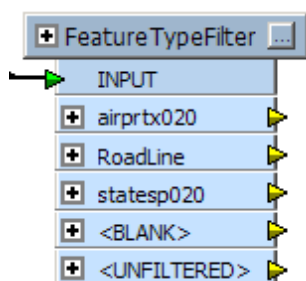
- Feature Type Name: An output port is created for every connected feature type.
- BLANK: If a feature was generated dynamically by this workspace, or has gone through another transformer that caused it to lose its feature type, then it will exit through the BLANK port.
- UNFILTERED: If a feature has a feature type that is not specified in the Feature Type Names parameter, then it will exit through the UNFILTERED port.

### Parameters

After connecting feature types to the Input port of the FeatureTypeFilter, open the transformer parameters and click the Update button at the bottom right. (You can also type the name in the Feature Type Names field.)



After you click OK, FME analyzes the workspace and determines the possible input feature types. The corresponding output ports are then created. Every time a new connection is made, FME adds an output port.



Note that when you remove a connection and update the FeatureTypeFilter, the corresponding output port is removed, as well as any connections made from that port. To avoid this, proceed with caution.

## Transformer Category

Filters

## **fmepedia**

See fmepedia for additional information about this transformer.

## **Technical History**

Associated FME function or factory: None

## **FilenamePartExtractor**

Extracts a part of a filename path and returns the result as a string.

### **Parameters**

Source Attribute

The attribute that holds the full filename and path.

Desired Filename Part

Choose the part of the filename you want extracted: Directory, Full Filename, Filename Root, Filename Extension

Target Attribute

The name of the attribute that will contain the results.

### **Usage Notes**

When parsing the path, both backslashes and forward slashes are handled as separators, regardless of operating system.

### **Example**

For example, the path

```
C:\WINNT\Profiles\user\Desktop\roads.shp
```

has a Directory of

```
C:\WINNT\Profiles\user\Desktop\
```

and a Full Filename of

```
roads.shp
```

and a Filename Root of

```
roads
```

and a Filename Extension of

```
shp
```

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @Filenamepart

## **FMEFunctionCaller**

Calls the specified FME function, optionally putting the resulting value in the Result Attribute. This is a safe way to run FME functions in Workbench, as they are used in mapping files.

See the *FME Functions and Factories* manual (Help > FME Functions and Factories Reference) for more information about FME functions.

### **Parameters**

FME Function

Specifies the FME function (*@FunctionName*) to be called, as well as any parameters it can take.

Result Attribute (optional)

The value returned by the FME function called can be specified in this result attribute.

Configuration Line (optional)

Can be specified if the underlying FME function requires it.

### **Example**

The @Reformat function requires a configuration line. To employ it within a workspace, the function would be called like this:

```
@Reformat(DestDecodeSrcEncode,BinaryKey,Company_Id)
```

and the configuration line which defines the "BinaryKey" binary structure would be specified as:

```
Reformat STRUCT_DEF BinaryKey_Part1 BigEndianInt(4,1) _Part2 BigEndianInt(4,5)
```

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function: Any

## FME ServerLogFileRetriever

Accesses the translation log for a specified FME Server-run translation. The translation log to access is identified by the job ID input parameter.

### Output Ports

The initiating feature is output via the SUCCEEDED port if the translation log was successfully retrieved for the specified job ID, and will have these attributes added to it:

- **\_log\_attr**: the contents of the translation log for the specified job

The feature will be output via the FAILED port if the translation log was not successfully retrieved for the specified job ID, and will have these attributes added to it:

- **\_job\_failure\_type**: contains the reason for the translation log retrieval failure
- **\_log\_attr**: the contents of the translation log for the specified job, obtained before an error occurred

### Parameters

#### Server Name and Port Number

These parameters identify the server that will be used to connect to the FME Server that contains a translation log of interest.

#### Username and Password

These parameters are credentials used to verify your level of access to the FME Server. Depending on your FME Server configuration, these parameters may be optional.

#### Job ID

This parameter is the job ID of the translation log that is to be requested.

#### Output Log Attribute

This parameter is the output feature attribute that the translation log is to be written to. By default, the value is **\_log\_attr**.

### Transformer Category

Workflow

### Dependencies

This transformer requires an FME Server connection.

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: ServerLogFileRetrieverFactory



## FME Server Job Submitter

Submits FME Spatial ETL jobs to be run on an FME Server. A job consists of a workspace (housed within a repository on an FME Server) together with values for each of its published parameters. This transformer submits one job per feature which enters it. Any published parameters of the server workspace will be given values as specified in the transformer, or taken from attributes of the feature which enters it.

Note: Publishing a workspace to FME Server that includes this transformer is not recommended unless the transformer is **not** set to *Wait* for results. On FME Server, when a workspace that contains this transformer is run and the workspace sends a job to the same FME Server that is running the original workspace, the original workspace may never finish (depending on the availability of FME Engines).

### Output Ports

- **SUCCEEDED:** Features that successfully submitted the requests to the server.
- **FAILED:** Features that failed to submit the requests to the server.

### Wizard Panes

#### *Connect to an FME Server*

##### Connection Type

Direct Connection: The server Host and Port number identify the server that will be used to execute the job.

Web Connection: You can also enter a URL to access FME Server using the SOAP protocol. (SOAP is a lightweight protocol intended for exchanging structured information in a distributed environment.) Note that when you enter a URL, the Port field will disappear, and the connection to FME Server will switch to the SOAP protocol. You may need to contact your System Administrator for information on host names or URLs.

##### Credentials

Username and Password: These are optional, but, depending on your configuration, you may need them in order to access the server.

Note: You can click the Test... button at the bottom of the dialog to ensure that the FME Server connection has been successfully established.

#### *Select Workspace*

Choose a repository, then select the workspace that will be run for each feature that enters the transformer.

#### *Edit Job Parameters*

Edit parameters to enable the current workspace to submit jobs to the FME Server

##### Wait for Server Job to Complete

**WARNING:** Do not set this parameter to Yes if you want to use this transformer in a workspace authored to FME Server. The FME-ServerJobSubmitter on FME Server may send a job to the same engine that is running the original workspace, resulting in a translation that can never complete. To use this transformer in a workspace authored to FME Server, ensure this parameter is set to No.

If this parameter is set to Yes, then the transformer will wait until the job is completely processed by the server before proceeding. In this case, the initiating feature is output via the SUCCEEDED port if the job successfully ran to completion, and will have these attributes added to it:

\_job\_id: the integer id the server assigned to this job

\_NumFeaturesOutput: the number features that were output by the job

\_timeFinished: the time the job finished

\_timeRequested: the time the server received the job request

\_timeStarted: the time the server started processing the job

All times are in YYYYMMDDhhmmss format.

The feature will be output via the FAILED port if the job either did not run to completion, or the server could not be contacted, and will have these attributes added to it:

`_job_failure_type`: holds one of "Connection or Server Problem" or "Translation Failed". If the latter, then the following attributes will also have values; otherwise, they are empty

`_job_id`: the integer ID the server assigned to this job

`_StatusMessage`: the error message returned from the server explaining why the request failed

`_StatusNumber`: an error number corresponding to the above message

`_timeFinished`: the time the job finished

`_timeRequested`: the time the server received the job request

`_timeStarted`: the time the server started processing the job

If this parameter is set to No, the transformer will output the initiating feature as soon as the job is submitted to the server. In this case, the initiating feature is output via the SUCCEEDED port if the request was successfully submitted, and only the `_job_id` attribute will be added to it. If the server could not be contacted or the job submission otherwise failed, then the feature is output via the FAILED port, and the `_job_failure_type` attribute will hold the "Connection or Server Problem" value.

### Job Priority

Job Priority defines the order in which the job will be executed on the server.

### Advanced (button)

The *Server Configuration Keyword* parameter defines the subsection name used by the FME Server in the configuration file `fmeengineconfig.txt` (which is described in the *FME Server Administrator's Guide*). In most cases, you will not need to change this keyword from the default that appears. If a subsection is present, it is named with a keyword by which can be referenced by client applications. Multiple subsections can be defined for various purposes.

### Buttons

Test	This button appears on the first wizard pane. Click it to confirm that the FME Server connection has been successfully established.
Help	Displays the help text for this transformer.
Back	Displays the previous pane.
Finish	Sets the transformer parameters. The transformer will use these parameters when you run the workspace.
Cancel	Closes the wizard pane and resets the parameters if they have not yet been saved.

### Dependencies

- This transformer requires an FME Server connection.

### Transformer Category

Workflow

### FME Licensing Level

FME Professional edition and above

### Transformer History

This transformer was previously named the `ServerJobSubmitter`.

### Technical History

FME Factory Used: `ServerJobSubmissionFactory`



## **FME Server Job Waiter**

Waits until submitted FME Spatial ETL jobs are completely processed by an FME Server. The list of jobs to wait for is identified by the job IDs of the input features. When a job that the transformer is waiting for is completed, it outputs the corresponding feature immediately.

### **Output Ports**

The initiating feature is output via the SUCCEEDED port if the job successfully ran to completion, and will have these attributes added to it:

- `_job_id`: the integer ID the server assigned to this job
- `_LogFileLocation`: the location of the logfile of the job
- `_result`: the translation result of the job
- `_requestKeyword`: the subsection keyword of the job
- `request`: the string request of the job that was submitted to the server
- `_StatusMessage`: the successful message returned from the server
- `_StatusNumber`: a success number corresponding to the message above
- `_NumFeaturesOutput`: the number features that were output by the job
- `_timeFinished`: the time the job finished
- `_timeRequested`: the time the server received the job request
- `_timeStarted`: the time the server started processing the job

All times are in YYYYMMDDhhmmss format.

The feature will be output via the FAILED port if the job either did not run to completion, or the server could not be contacted, and will have these attributes added to it:

- `_job_failure_type`: holds one of "Incomplete Parameters", "Connection or Server Problem" or "Translation Failed". If the latter, then the following attributes also will have values; otherwise, they are empty
- `result`: the translation result of the job
- `_requestKeyword`: the subsection keyword of the job
- `_request`: the string request of the job that was submitted to the server
- `_job_id`: the integer ID the server assigned to this job
- `_StatusMessage`: the error message returned from the server explaining why the request failed
- `_StatusNumber`: an error number corresponding to the above message
- `_timeFinished`: the time the job finished
- `_timeRequested`: the time the server received the job request
- `_timeStarted`: the time the server started processing the job

### **Parameters**

#### **Server Name and Port Number**

These parameters identify the server that will be used to execute the job.

#### **Username and Password**

These are optional, but, depending on your configuration, you may need them in order to access the server.

#### **Job ID**

The list of jobs to wait for is identified by the job IDs of the input features. When a job that it is waiting for is completed, it outputs the corresponding feature immediately.

## Polling Interval

The time interval for this transformer to wait between inquiries into the status of each job is specified by the Polling Interval. This is measured in seconds and can be entered as an integer value or an integer attribute.

Note: You should use as large value as possible for this parameter. If it is set too small, it not only impacts the resources on the client, but also on the FME Server responding to each query. For example, if the job is expected to take about 20 minutes, then it is not efficient to set the Polling Interval to a few seconds.

## Transformer Category

Workflow

## Dependencies

This transformer requires an FME Server connection.

## FME Licensing Level

FME Professional edition and above

## Transformer History

This transformer was previously named the ServerJobWaiter.

## Technical History

Associated FME function or factory: ServerJobWaitingFactory

## FME Server Workspace Runner

Submits FME Spatial ETL jobs to be run on an FME Server, and downloads the resulting data to a specified location. You can optionally upload files used for the job, and download results locally when the FME Server job is complete.

This transformer uses FME Server's REST (Representational State Transfer) capabilities.

Note: Publishing a workspace to FME Server that includes this transformer is not recommended unless the transformer is **not** set to *Download Results*. On FME Server, when a workspace that contains this transformer is run and the workspace sends a job to the same FME Server that is running the original workspace, the original workspace may never finish (depending on the availability of FME Engines).

### Output Ports

- SUCCEEDED: Features that successfully submitted the requests to the server.
- FAILED: Features that failed to submit the requests to the server.

### Wizard Panes

#### *Connect to an FME Server*

Connection Type

**Direct Connection:** The server Host and Port number identify the server that will be used to execute the job.

**Web Connection:** You can also enter a URL to access FME Server using the SOAP protocol. (SOAP is a lightweight protocol intended for exchanging structured information in a distributed environment.) Note that when you enter a URL, the Port field will disappear, and the connection to FME Server will switch to the SOAP protocol. You may need to contact your System Administrator for information on host names or URLs.

Credentials

**Username and Password:** These are optional, but, depending on your configuration, you may need them in order to access the server.

Note: You can click the Test... button at the bottom of the dialog to ensure that the FME Server connection has been successfully established.

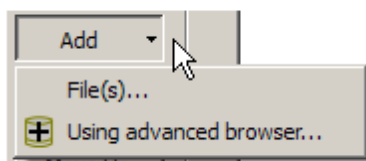
#### *Select Repository Item*

Once connected to the server, you must choose a repository and select a workspace from the list. This workspace defines the job that will be executed on the server.

#### *Select Data to Upload*

If you need to upload files to be used in this job, this transformer can use FME Server's Data Upload Service to upload the files to the FME Server. To use this feature, please ensure that the Data Upload Service is enabled on the target FME Server.

If desired, select files (or attributes that contain file paths) to upload.



Add single items using the file browser, or use the advanced browser to add multiple files and directories.

You can change the order in which the files will be processed by the transformer.

#### *Edit Published Parameters*

This page will appear if the server workspace contains any configurable parameters.

Any published parameters associated with the server workspace must be given values. These values can be taken from attributes of the feature that triggered the job to be submitted, or can be typed in (in which case they will be the same for each job that is sent via the transformer).

Note: If any of the parameters are filenames, those filenames must be valid paths on the server which will execute the job.

## Specify Job Settings and Result Processing

### Job Settings

*Job Priority:* Defines the order in which the job will be executed on the server.

### Result Processing

These parameters determine how the transformer will respond after a job is submitted.

*Do not download results:* The transformer will run without waiting for the job to complete, and will perform no additional tasks after the job is submitted.

*Download results:* If the selected FME workspace contains file path parameters, input files may be specified to be uploaded, and when the job is complete, the transformer will download the results to the path specified in the Target folder field.

*E-mail Result link when job is complete:* Optionally, the transformer may be configured to work asynchronous to the FME Server job. In this case, e-mail address(es) can be provided to notify the user when the job is complete, or taken from the value of an attribute.

### Buttons

Test	This button appears on the first wizard pane. Click it to confirm that the FME Server connection has been successfully established.
Help	Displays the help text for this transformer.
Back	Displays the previous pane.
Finish	Sets the transformer parameters. The transformer will use these parameters when you run the workspace.
Cancel	Closes the wizard pane and resets the parameters if they have not yet been saved.

### Usage Notes

- Please ensure that the Data Download Service is enabled on the target FME Server.
- If files are to be uploaded, please ensure that the Data Upload Service is enabled on the target FME Server.
- In the Navigator pane, you can specify the following transformer parameters through feature attributes: Server Host, Server Port, Username, Repository, Workspace, and Workspace Parameters.

### Transformer Category

Workflow

### Dependencies

This transformer requires an FME Server connection.

### FME Licensing Level

FME Professional edition and above

### Related Transformers

There is also a WorkspaceRunner that is used to run local workspaces.

### Technical History

Associated FME function or factory: ServerDataDownloadFactory

## Generalizer

There are four types of algorithms:

- **Generalizing algorithms** reduce the density of coordinates by removing vertices.
- **Smoothing algorithms** determine a new location for each vertex.
- **Measuring algorithms** calculate the location of points, and return a list of these points (for example, to measure the sinuosity of a feature).
- **Fitting algorithms** replace the original geometry completely, with a new feature fitted to a specified line (for example, to minimize the orthogonal distance to the original).

The algorithm that you choose determines which transformer parameters are enabled in the transformer dialog.

## Generalizing Algorithms

### *Douglas*

The Douglas algorithm will remove vertices which cause a deviation of less than the Generalization Tolerance, but the location of remaining vertices are not altered.

Corresponding parameters:

- Preserve Shared Boundaries
- Generalization Tolerance

### *Thin*

The Thin algorithm will remove vertices that are less than the Generalization Tolerance distance away from an adjacent vertex. The begin and end points are never moved, unless the entire length of the feature being thinned is less than the tolerance, in which case the feature is replaced by a point feature holding the final coordinate.

Corresponding parameters:

- Preserve Shared Boundaries
- Generalization Tolerance

### *ThinNoPoint*

The ThinNoPoint algorithm will remove vertices that are less than the Generalization Tolerance distance away from an adjacent vertex. The begin and end points are never moved, even when the entire length of the feature being thinned is less than the tolerance, in which case the feature is replaced by a linear feature connecting the first point to the last point.

Corresponding parameters:

- Preserve Shared Boundaries
- Generalization Tolerance

### *Deveau*

The Deveau algorithm removes vertices which contribute less to the overall shape of the feature, and may introduce new vertices at positions not originally in the feature as it works. The inherent behavior of the algorithm is such that it invalidates the z coordinate of the vertices, and any measures. Therefore the output features will always be 2D, and have no measures on them. It requires the Smoothness Factor parameter and the Sharpness Angle parameter to be specified.

Corresponding parameters:

- Preserve Shared Boundaries
- Generalization Tolerance



- Smoothness Factor
- Sharpness Angle

### **Wang**

The Wang algorithm will iteratively combine, eliminate and exaggerate bends until the input line feature has no bend that is smaller than the given tolerance value.

Corresponding parameters:

- Preserve Shared Boundaries
- Generalization Tolerance

## **Smoothing Algorithms**

### **McMaster**

The McMaster algorithm calculates a new location for each point by first taking the average value of the x and y coordinates of the point and a number of neighboring points. It then slides the averaged point towards the original point according to a specified displacement value. The overall effect is that each point will be pulled towards its neighboring points.

Corresponding parameters:

- Preserve Shared Boundaries
- Number of Neighbors
- Displacement Percentage

### **McMaster Weighted Distance**

The McMaster Weighted Distance algorithm performs the same operations as the McMaster algorithm only it uses inverse distance weighting to take into account the distance from each neighbor to the point being moved. The overall effect is that points further away will have less "pull" than points close by.

The Weighting Power parameter is used by the McMaster Weighted Distance algorithm only. It is used to determine the weight of each neighboring point.

**Note:** For lines, the McMaster algorithms do not change the first and last N points (where N is the number of neighbors), because they don't have enough neighbors for the averaging calculations to work with. For polygons, a wrap-around is used so each point in a polygon will be changed. In the case of adjacent polygons and the Preserve Shared Boundaries option, collinear portions of their boundaries will be smoothed together. The remaining parts of their boundaries will be smoothed as lines. This means that no wrap-around will be used for adjacent polygons.

Corresponding parameters:

- Preserve Shared Boundaries
- Number of Neighbors
- Displacement Percentage
- Weighting Power

### **NURBfit**

The NURBfit algorithm will fit lines using B-Spline curves of given polynomial degree. The resulting lines will follow these curves with given segment length. The higher the degree, the smoother the line. An example of usage is smoothing contour lines in order to remove sparks and simulate the work of a cartographic craftsman.

Corresponding parameters:

- Preserve Shared Boundaries
- Degree of Basis Polynomial
- Segment Length

## Measuring Algorithms

### *Inflection Points*

The Inflection algorithm will calculate the location of the inflection points along a line and return the list of these points. Inflection points are measures of the sinuosity of a line.

Corresponding parameters:

- Number of Neighbors

## Fitting Algorithms

### *Orthogonal Distance Regression*

This algorithm replaces the feature's geometry with a line that minimizes the orthogonal distance between it and the original geometry's points. Orthogonal distance means the shortest (perpendicular) distance between a point and a line.

Corresponding parameters:

- None

## Parameters

Each numeric parameter may be entered as a number or taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Preserve Shared Boundaries

**No:** Each feature will be treated and generalized individually without regard to its neighboring features. If the area features originally formed a coverage, there will be gaps and overlaps in the coverage. If you want the coverage to be maintained while doing area boundary generalization, choose *Yes*.

**Yes:** Coverage topology will be maintained while doing area boundary generalization. The entire coverage of area features must not overlap. If the area features overlap, then you should choose *No*, or use the *AreaOnAreaOverlayer* first to create a coverage. In some situations, you can also use the *Snapper* in VERTEX mode either before, after, or instead of this transformer.

This transformer computes topology for the coverage, generalizes the individual arcs, and then recreates the area features. This option will take longer for areas because it computes the arc/node topology, generalizes the individual arcs, and then recreates the areas.

### Generalization Tolerance

This parameter is used by all four generalizing algorithms. It is measured in ground units (units of measure of the feature coordinates).

### Smoothness Factor

This parameter is used by the *Deveau* algorithm only. It controls the number of simultaneous wedges considered when floating bands around the points in the set. The larger this value is, the more aggressive the generalization.

### Sharpness Angle

This parameter is used by the *Deveau* algorithm only. It sets the tolerance for spikes that will be blunted. Vertex points at angles less than the value given from the previous two points are not moved. The angle is measured in degrees.

### Number of Neighbors

This parameter specifies the number of neighbors to consider for each point. For example, a value of 2 specifies that the 2 points to the left of each point, the point itself, and the 2 points to the right will be considered. For the *Inflection Points* algorithm, this parameter specifies the number of neighboring points on either side that will affect the inflection calculation. A higher number has the effect of smoothing the line and may result in fewer inflection points. A value of 0 means no filtering.

### Displacement Percentage

This parameter specifies the location between the original and average points to move the point. For example, a value of 50 will place the point at the halfway point between the averaged point and the point's original location.

## Weighting Power

This parameter is used by the McMaster Weighted Distance algorithm only. It is used to determine the weight of each neighboring point.

## Degree of Basis Polynomial

This parameter specifies the degree of the polynomial used to approximate the curve.

## Segment Length

This parameter specifies the length of the output segments. If this is set to 0, then the output curve will have 10x the number of points in the input.

## Geometry Handling

If the Geometry Handling Advanced setting is set to Enhanced in the workspace, arcs and ellipses are not touched, while the location of text features are generalized/smoothed; otherwise, the result of generalizing/smoothing arcs, ellipses, and text are single points located at their respective center points.

## Usage Notes

To maintain topologies that involve other features while generalizing, consider using the SherbendGeneralizer transformer.

## Transformer Category

Manipulators

## FME Licensing Level

FME Professional edition and above

## Transformer History

This transformer replaces the AreaGeneralizer, AreaSmoother, LineGeneralizer and LineSmoother.

## Technical History

FME Function Used: @Generalize

FME Factories Used: AggregateFactory, DeaggregateFactory, DonutFactory, ReferenceFactory, PolygonFactory, TopologyFactory

## GeometryCoercer

Resets the geometry type of the feature. Depending on the feature's actual coordinates, the transformer may have no effect. This transformer is sometimes used to have area features treated as though they were linear features, either because some later processing requires lines, or the destination format represents lines different than polygons and the linear representation is desired.

### Parameters

#### Geometry Type

If you try to set the feature's geometry type to `fme_point` and the feature has more than one coordinate, then the feature's geometry type is unchanged. The feature will also be cleaned up if there are duplicate points. An exception is made for point clouds, where the `fme_point` option will produce a single multi-point feature containing all points in the point cloud.

If you try to set the feature's geometry type to `fme_polygon` and the feature has more than one coordinate, then the first coordinate and the last coordinate must be the same or the feature's geometry type will be unchanged.

If you try to set the feature's geometry type to `fme_line` or `fme_polygon` and the feature had only one coordinate, then the feature's geometry type is unchanged.

If you try to set the feature's geometry type to `fme_arc` or `fme_ellipse` and the feature had a center point, the existing center point will be used. If no center point existed, the first coordinate on the feature will be used for the center point.

If you try to set the feature's geometry type to `fme_text`, the existing geometry will be used as the text location. An exception to this occurs when the existing geometry is already of type `fme_text`, in which case the text location will be unchanged.

If you try to set the feature's geometry to `fme_composite_surface` and the geometry of the feature is not a multi-surface or brep solid, or extrusion or box or csg solid, then the feature's geometry will be unchanged. An exception for this option is when the source feature contains a mesh, in which case this option will produce a multi-surface.

If you try to set the feature's geometry to `fme_brep_solid` and the geometry of the feature is not a composite surface or multi-surface or extrusion or box or csg solid, then the feature's geometry will be unchanged.

If you try to set the feature's geometry to `fme_point_cloud` and the geometry of the feature is not a raster, multi-point or a simple aggregate made up only of points, then the feature's geometry will be unchanged.

#### Compare Z values for duplicates removal

If Compare Z values for duplicates removal is set to Yes, then two coordinates are considered as duplicates if their coordinates are equal in X, Y and Z values. Otherwise, only X and Y values are considered. If the geometry is a surface or a solid, then the geometry of the feature is turned into a polygon or a collection of polygons. In these cases, this parameter is ignored.

### fme\_geometry and fme\_type

See `fme_geometry` and `fme_type` for more information.

### Usage Notes

- When setting a feature's geometry type to `fme_arc`, `fme_ellipse`, or `fme_text`, an error will occur if the input feature does not have the attributes required to define the new geometry. If an input feature does not have all of the attributes it requires, an `AttributeCreator` can be used to add them.
- Coercing into an arc requires `fme_primary_axis`, `fme_start_angle`, and `fme_sweep_angle`, and optionally `fme_secondary_axis` and `fme_rotation`.
- Coercing into an ellipse requires `fme_primary_axis`, and optionally `fme_secondary_axis` and `fme_rotation`.
- Coercing into text requires `fme_text_string`, `fme_text_size`, and optionally `fme_rotation`.
- If the feature was a donut polygon or an aggregate, this transformer will have no effect on it.
- You can't directly use the `GeometryCoercer` to convert donut polygons to lines. If you need to do this, extract the donut parts first using the `DonutHoleExtractor`.

### Transformer Category

Manipulators

**fmepedia**

See fmepedia for additional information about this transformer.

**Technical History**

Associated FME function or factory: @GeometryType

## GeometryExtractor

Extracts the geometry of a feature according to the setting of the geometry encoding parameter.

The resulting encoded geometry is added to the feature in an attribute. This attribute can later be restored as the feature's geometry via the GeometryReplacer transformer.

This transformer is often used to make a copy of the feature's geometry into an attribute before some temporary geometry change is made, so that it can later be restored. Alternately, the geometry may be extracted to be stored in a database or other file format that cannot handle geometry, but can handle large attributes as blobs or text strings. Later, FME can read this data back and restore the geometry via the GeometryReplacer transformer.

Note: Restoring GML-encoded geometry is not supported.

## Usage Notes

To carry out a similar operation on raster data, please use the RasterExtractor transformer.

## Parameters

### Geometry Encoding

This parameter can be set to FME Binary, hexadecimal-encoded FME Binary, FME XML, OGC Well Known Text (wkt), Well Known Binary (wkb), hexadecimal-encoded Well Known Binary (wkbhex), GeoJSON, ESRI JSON, GeoRSS Simple Encoding, GML, or KML.

The most efficient and truest representation of the geometry is FME Binary, and this should be used unless some external reasons intervene. All the FME representations can accommodate all aspects of FME Enhanced Geometry, including measures and paths consisting of multiple linear segments; however, both the FME XML and Hex Encoded FME Binary representations impose some overhead in translating between the internal binary representation and the ASCII-encoded representation.

The OGC variants are useful if interaction with other OGC supporting systems is required. However, some characteristics of geometries may be lost in these modes; for instance, any path will be flattened into a single linestring.

Because OGC provides no WKT or WKB representation for "null" geometries, the resulting attribute value in these modes for a feature with no geometry will be the empty string ("").

The GeoJSON, ESRI JSON, and GeoRSS Simple Encodings may not preserve all geometry characteristics. For example, arcs will be stroked to lines, and ellipses stroked to polygons. These encodings also do not support measures.

### Destination Geometry Attribute

The name of the attribute that will hold the encoded geometry.

### OGC Version (WKT/WKB only)

You can choose between versions 1.1 and 1.2.

Version 1.2 contains two key improvements: support for measures and z-values. Note that the only measure considered when producing an OGC representation is the "default" (unnamed) measure.

### Omit XML Namespace Declarations (GeoRSS/KML/GML only)

For XML output formats (except FME XML), this parameter controls whether or not the output contains XML namespace declarations. All well-formed XML output must contain namespace declarations.

However, users who are manually inserting the transformer output into a larger XML document may wish to omit the namespace declarations.

## Transformer Category

Manipulators

## Related Transformers

GeometryReplacer

RasterExtractor

RasterReplacer

**FME Licensing Level**

FME Professional edition and above

**Transformer History**

This transformer contains the functionality of the now-deprecated OGCGeometryExtractor and XMLGeometryExtractor.

**Technical History**

Associated FME function or factory: @Geometry, @OGCGeometry, @JSONGeometry, @GeoRSSGeometry, @GMLGeometry, @KMLGeometry

## GeometryFilter

Routes a feature based on its geometry type.

Each feature that enters the transformer is output via the port corresponding to its *fme\_type*. Each output feature has a complete, unaltered copy of the source feature's attributes and geometry.

### fme\_type

Each output port corresponds to standard *fme\_type* attributes.

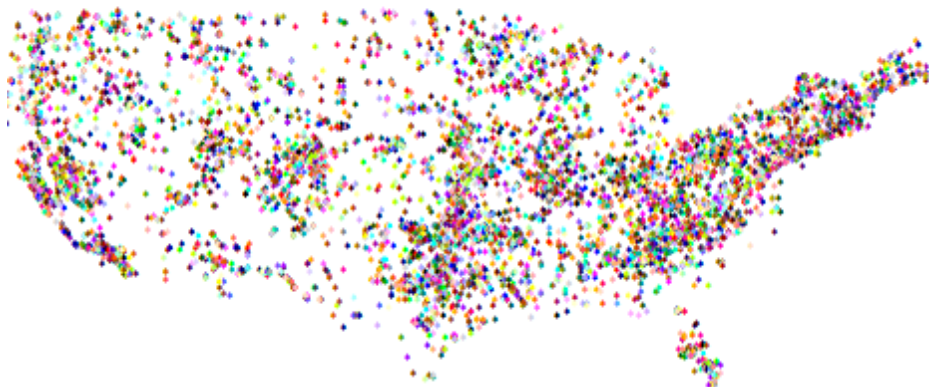
### Parameters

None.

### Example

View results by routing a feature type through a GeometryFilter to an Inspector.

If you route a feature type that contains point geometry through the GeometryFilter POINT output port to an Inspector, you will see only the *fme\_point* geometry in the viewer:



The Information pane in the Universal Viewer shows the *fme\_geometry* and *fme\_type*:

Attribute Name	Attribute Value
<i>fme_geometry</i>	<i>fme_point</i>
<i>fme_type</i>	<i>fme_point</i>
HAZARD	H
LATITUDE	45.18833
LONGITUDE	-89.755
MAX_STOR	6600
multi_reader_full_id	0
multi_reader_id	0
multi_reader_keyword	SHAPE_1
multi_reader_type	SHAPE
NID_HEIGHT	40
NIDID	WI00748
NORMAL_STO	5200

### Usage Notes

FME will sometimes automatically insert a GeometryFilter into a new workspace. Some destination formats only permit features of a specific geometry type to be written to a single feature type. For example, a Personal GeoDatabase Feature Type (ESRI Feature Class) can hold polygons or polylines, but not both.



When you read from a source dataset that permits multiple geometry types in a feature class, but write to a destination dataset that is restricted to a single geometry type per class, FME automatically creates a destination feature type for each geometry type and inserts a GeometryFilter to divide up the features on the basis of geometry. This ensures that no destination feature type receives features that it is not permitted to write.

### **Related Transformers**

Aggregate features are not handled specifically by this transformer, as several geometries may be structured as aggregates. To filter aggregates, use the AggregateFilter.

### **Transformer Category**

Filters

### **Technical History**

Associated FME function or factory: Not applicable

## **GeometryInstantiator**

Instantiates a geometry instance into a specific concrete instantiation of geometry definition.

The geometry instance's insert location and placement matrix are applied to the geometry definition for this instantiation.

### **Output Port**

- **INSTANTIATED:** Each output feature will contain a copy of the source feature's attributes. If the feature passed in is not a geometry instance, this transformer will output the feature untouched.

### **Parameters**

Deaggregate Recursively

If requested, the instance will be deaggregated recursively and each part will be output as separate features. In this case, every geometry instance will be instantiated, at every depth, as it recurses.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: DeaggregateFactory

## **GeometryNameExtractor**

Retrieves the name of the geometry, and sets it on the specified attribute.

### **Output Ports**

- HAS\_NAME: If the name exists and can be converted to UTF8, it will be supplied on the attribute as a UTF8 string and the geometry will be passed through this port.
- NO\_NAME: All other geometries will be passed through this port.

### **Parameters**

Geometry Name Attribute

The name of the new attribute that will hold the geometry name.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @Geometry

## **GeometryNameRemover**

Removes the name of the geometry. Note that component parts of aggregate geometries will remain unchanged.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @Tcl2

## **GeometryNameSetter**

Applies a name to the geometry of a feature.

The name is set on the geometry using the UTF8 encoding. If geometry previously had a name, it will be overwritten with the new value.

### **Parameters**

Geometry Name

The geometry name can be a fixed value, or you can choose an existing name from an attribute in the pull-down list.

### **Technical History**

Associated FME function or factory: @Tcl2

## GeometryOGCValidator

Evaluates the simplicity or validity of a geometry feature, and routes the feature according to the outcome of the test(s).

### Output Ports

In Valid mode, features coming out the FAILED port will have additional attributes added. These attributes describe where and why the geometry is invalid:

`_validateFailCoordX` X coordinate where the geometry is invalid

`_validateFailCoordY` Y coordinate where the geometry is invalid

`_validateFailReason` Failure case index (an integer)

`_validateFailReasonString` English description of the failure

Possible values of `_validateFailReason` and `_validateFailReasonString` are:

- 0: Undetermined Error
- 1: Repeated Point (not currently implemented)
- 2: Hole Outside Shell
- 3: Nested Holes
- 4: Disconnected Interior
- 5: Self Intersection
- 6: Ring Self Intersection
- 7: Nested Shells
- 8: Duplicated Rings
- 9: Too Few Points
- 10: Invalid Coordinate
- 11: Ring Not Closed

### Parameters

Validation Type

Validation Type can be set to either Simple or Valid:

- When set to Valid, the feature is evaluated according to the Java Topology Suite (JTS) implementation of the OGC Geometry test *isValid*.
- When set to Simple, the feature is evaluated according to the JTS implementation of the OGC Geometry test *isSimple*.

See <http://www.vividsolutions.com/jts/> for more information on JTS.

Note: Any lines, even when closed, will be treated as JTS LineStrings, never LinearRings.

### Descriptions

The meanings of Simple and Valid can be different, depending on the type of geometry:

#### **Point**

A point is always Valid and Simple.

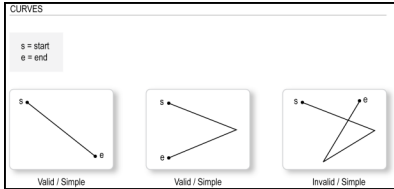
#### **Multi-point**

A multi-point is not Valid, or Simple, when it contains a duplicate point.

## Curve

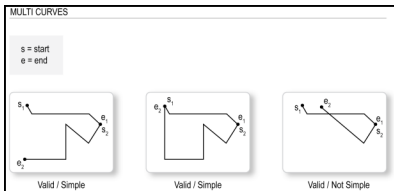
A curve is Simple and Valid if (and only if) the following statements are all true:

1. It does not contain exactly 1 unique point.
2. It contains at most 1 duplicate point where the duplicates define the endpoints of a closed curve.
3. It does not self-intersect.



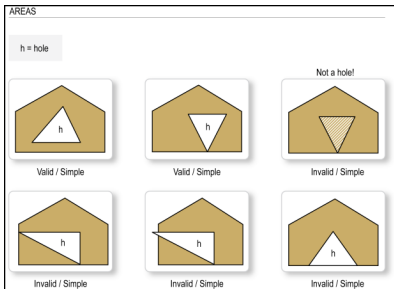
## Multicurve

A multicurve is Valid only if its components are Valid, and any intersections among its components occur at the endpoints of those components. It is Simple only if it is Valid and its components do not intersect one another.



## Area

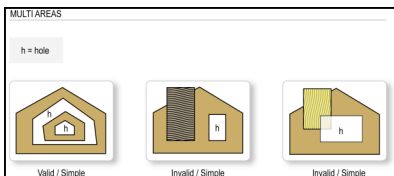
An area is Simple if its boundary rings are Simple. It is Valid if it is Simple and its boundary rings do not cross one another or split the interior into two or more pieces.



## Multiarea

A multiarea is Valid if (and only if) the following statements are true:

1. its components are Valid,
2. the component interiors do not intersect, and
3. the component boundaries do not intersect, or intersect at a finite number of points.



## **Aggregate**

An aggregate is Valid only if its components are Valid. It is Simple only if its components are Simple.

### **Technical Definitions**

The above descriptions are intended to cover common cases, and may be incomplete or contain inaccuracies.

For the original technical definitions, please refer to Section 6.1 *Geometry Object Model* of the "Common Architecture Specification" available at: <http://www.opengeospatial.org/standards/sfa/>

See these sections for definitions relating to specific geometry:

- Curve: Sections 6.1.6 *Curve*, and 6.1.7 *LineString*, *Line*, *LinearRing*
- Multicurve: Sections 6.1.8 *MultiCurve*, and 6.1.9 *MultiLineString*
- Area: Sections 6.1.10 *Surfaces*, and 6.1.11 *Polygon*, *Triangle*
- Multiarea: Sections 6.1.13 *MultiSurface* and 6.1.14 *MultiPolygon*

### **Transformer Category**

Filters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @OGCGeometry



## **GeometryRefiner**

Performs the following "refinements" on features' geometry:

- Any homogeneous IFMEAggregate will become a multi (IFMEMultiCurve, IFMEMultiArea, IFMEMultiPoint, or IFMEMultiText).
- Any IFMEAggregate or multi with only one member will be replaced by its single part.
- Any IFMEDonut with no holes will become an IFMEPolygon or IFMEEllipse.
- Any IFMEPath with only one segment will be replaced by that segment.
- Consecutive IFMELine segments within an IFMEPath will be combined.

## **Transformer Category**

Manipulators

## **Technical History**

Associated FME function or factory: @Geometry

## **GeometryRemover**

Completely removes the geometry of the feature, for example, if you want to turn spatial data into non-spatial data.

You can use the 2DPointAdder transformer afterwards to add vertices to the resulting feature.

## **Transformer Category**

Manipulators

## **Technical History**

Associated FME function or factory: @RemoveGeometry

## GeometryReplacer

Replaces the geometry of a feature according to the setting of the geometry encoding parameter. This transformer is typically used to restore geometry previously extracted into an attribute by the GeometryExtractor.

### Parameters

#### Geometry Encoding

This parameter can be set to FME Binary, hexadecimal-encoded FME Binary, FME XML, OGC Well Known Text (wkt), Well Known Binary (wkb), hexadecimal-encoded Well Known Binary (wkbhex), Parseable Encoded FME XML, GeoJSON, ESRI JSON, GeoRSS Simple Encoding, or KML.

The most efficient and truest representation of the geometry is FME Binary, and this should be used unless some external reasons intervene. All the FME representations can accommodate all aspects of FME Enhanced Geometry, including measures and paths consisting of multiple linear segments; however, both the FME XML and Hex Encoded FME Binary representations impose some overhead in translating between the internal binary representation and the ASCII-encoded representation.

The OGC variants are useful if interaction with other OGC supporting systems is required. However, some characteristics of geometries may have been lost in these modes; for instance, any path will be flattened into a single linestring. For OGC Well Known Text and OGC Well Known Binary, if measures are specified in the input attribute, they will be saved as the "default" (unnamed) measure on the generated geometry.

When converting from WKT or WKB, if the specified attribute has a blank value, the feature's geometry will be left untouched and a warning will be output. This is important to remember if a GeometryExtractor was used to generate the attribute, because that transformer produces an empty value whenever it encounters a feature with no geometry.

The Parseable Encoded FME XML option is used to take the geometry representation used by the Creator transformer and set the geometry from that.

The GeoJSON ESRI JSON and GeoRSS Simple Encodings may not preserve all geometry characteristics. For example, arcs will be stroked to lines, and ellipses stroked to polygons. GeoJSON does not support measures.

#### Source Geometry Attribute

The name of the attribute that will hold the geometry.

### Transformer Category

Manipulators

### Related Transformers

GeometryExtractor

RasterExtractor

RasterReplacer

### FME Licensing Level

FME Professional edition and above

### Transformer History

This transformer contains the functionality of the now-deprecated OGCGeometryReplacer and XMLGeometryReplacer.

### Technical History

Associated FME function or factory: @Geometry, @OGCGeometry, @JSONGeometry, @GeoRSSGeometry, @KMLGeometry

## GeometryTraitExtractor

Copies the specified geometry traits into feature attributes with the same names.

Geometry traits are similar to attributes on a feature; they are defined as any kind of user-defined data that is stored onto a geometry. Each separate geometry can hold its own specific data. This capability is useful for situations where geometries are combined or split apart within features.

### Parameters

Traits to Extract

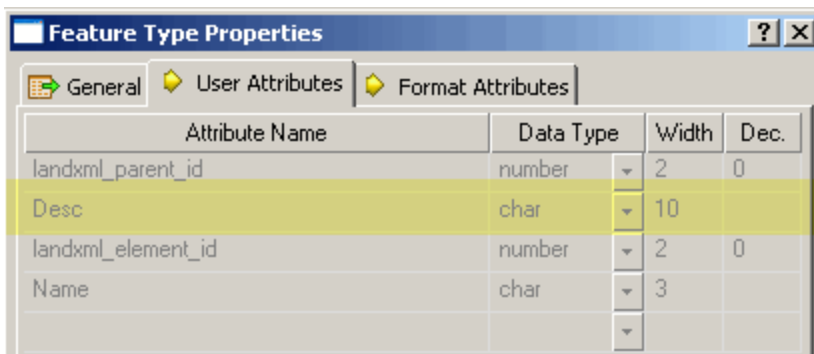
The name of the geometry traits to extract.

**Note:** By default, the transformer will overwrite any attributes that match the selected traits.

### Example

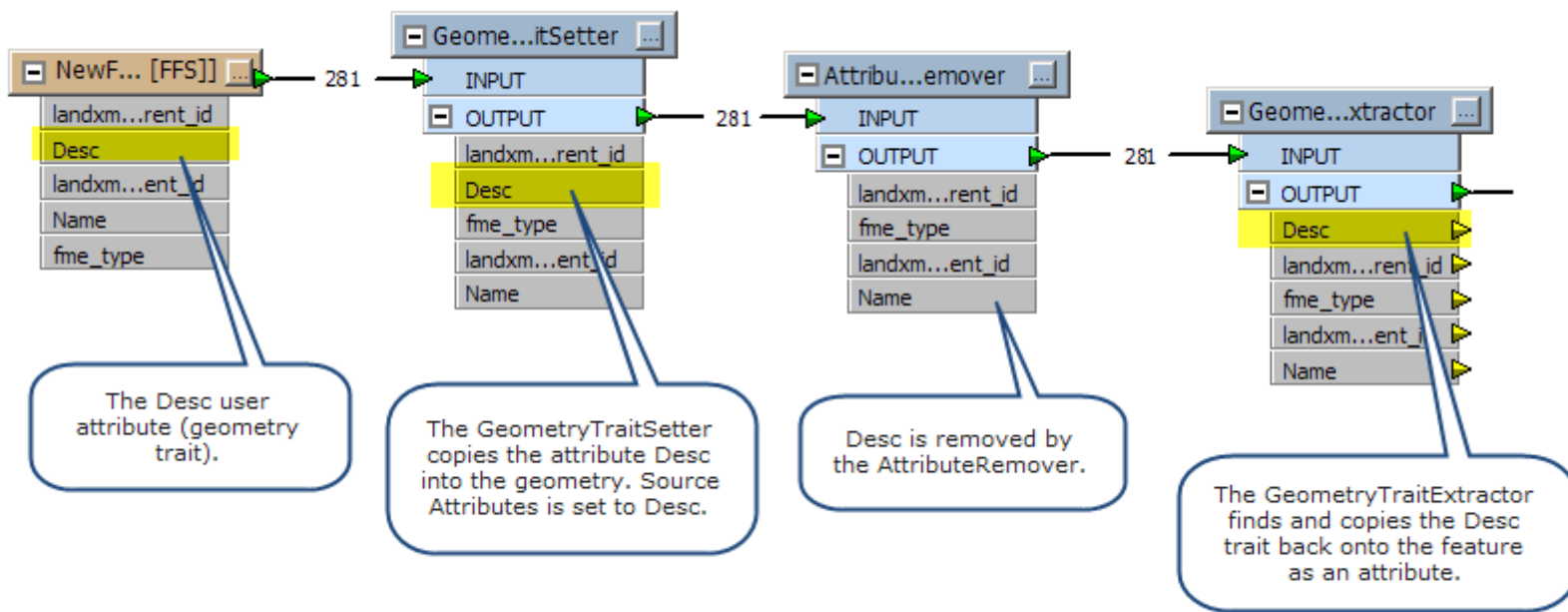
The example below shows how the GeometryTraitExtractor works in conjunction with a GeometryTraitSetter and an AttributeRemover.

- The Reader has a user attribute (geometry trait) named "Desc", as shown in the Feature Type Properties dialog.



Attribute Name	Data Type	Width	Dec.
landxml_parent_id	number	2	0
Desc	char	10	
landxml_element_id	number	2	0
Name	char	3	

And in the workspace, the transformers are set up as follows:



**Transformer Category**

Manipulators

**Transformer History**

This transformer was renamed from GeometryTraitFetcher.

**Technical History**

Associated FME function or factory: @GeometryTraits

## GeometryTraitRemover

Removes some or all traits from a feature's geometry. A geometry trait is defined as any user-defined data that is stored within a feature's geometry.

You can choose to remove traits at a geometry's top level (default), or at all levels.

### Parameters

Remove All

By default, the transformer will remove all traits at the top level of a geometry.

Filter (regex)

You can use the "Filter" option to enter a regular expression that the transformer will use as criteria to select traits for removal. Only traits whose names match this regular expression will be removed.

See the StringSearcher for more information on regular expressions.

Recursive

To remove all traits at all levels of a geometry, select Yes.

### Example

Consider a multi-surface containing a single face:

- Multi-surface contains two traits: `multisurf_id`, `type`
- Face contains two traits: `face_id`, `type`
- To remove top-level traits for the multi-surface but preserve traits for the face: Remove All = Yes; Recursive = No
- To remove just `type` from the multi surface: Remove All = No; Filter (regex) = `type`; Recursive = No
- To remove all traits at all levels: Remove All = Yes; Recursive = Yes
- To remove all the `*id` traits: Remove All + No; Filter (regex) = `id`; Recursive = Yes

### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @GeometryTraits

## **GeometryTraitSetter**

Copies attributes from a feature into geometry traits.

Geometry traits are similar to attributes on a feature; they are defined as any kind of user-defined data that is stored onto a geometry. Each separate geometry can hold its own specific data. This capability is useful for situations where geometries are combined or split apart within features.

### **Parameters**

#### Source Attributes

The new traits will have the same names as the selected source attributes.

#### Overwrite Existing Traits

By default, existing traits with the same names as the source attributes will be overwritten. If you do not want them to be overwritten, change this parameter to No.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: @GeometryTraits

## GeoRSSFeatureExtractor

Constructs GeoRSS documents from the input features and stores them in the specified attribute for the features that are output by the GeoRSS port. The constructed GeoRSS document will be created according to the specified Geometry Format and Output Format, in the character set specified by the Output Character Set parameter.

### Output Ports

- FEED: For any given GeoRSS document, the metadata for the feed.
- ENTRY: The entry features from the input GeoRSS document/feed.
- GEORSS: The features containing the GeoRSS document in the attribute specified by the GeoRSS Attribute parameter.

### Parameters

#### Flush Attribute

The GeoRSS document that is being constructed accumulates features until the value specified for the feature Flush Attribute. When this occurs, the newly received input feature is not part of the flushed document, but rather is part of the next document to be constructed.

#### GeoRSS Attribute

This attribute contains the GeoRSS document.

#### Output Format

Choose the format of the XML feed that the GeoRSS writer will produce:

- Atom: the writer will produce an Atom 1.0 feed.
- RSS: the writer will produce an RSS 2.0 feed.

#### Geometry Format

Choose the format of the output feed's geometry extensions: GML, W3C, or Simple.

#### Escape HTML Content

This parameter determines how the writer handles HTML content. Select Yes to ensure that the writer will escape HTML content before outputting it. Select No to ensure that the writer will output the content unchanged.

#### Output Character Set

The character set encoding in which the output XML feed should be written. If no character set is specified, the feed will be written in the UTF-8 character set. If an invalid character set is specified, the translation will fail.

### Transformer Category

Web Services

### FME Licensing Level

FME Professional edition and above

### fmepedia

See fmepedia for additional information about GeoRSS and FME.

### Technical History

Associated FME function or factory: GeoRSSFactory



## GeoRSSFeatureReplacer

Constructs features out of GeoRSS documents/URLs that are stored in a specified attribute of the input features. The features from the GeoRSS document/URL are output with the attributes from the original feature and are merged, if desired.

### Output Ports

- FEED: This port will output one feature per GeoRSS document that is read, and which will contain the feed metadata (as per the GeoRSS Reader).
- ENTRY: One feature will be output for each GeoRSS entry in the GeoRSS document.
- INVALID\_GEORSS: Features whose GeoRSS attribute does not contain valid GeoRSS.

### Parameters

#### *GeoRSS Settings*

##### GeoRSS Attribute

The attribute to read from.

##### GeoRSS Reader Mode

The GeoRSS reader can be run in two modes:

- In NORMAL mode, the reader will always return a feature for every feed entry that it processes.
- In UPDATE mode, the reader will only return entry features if they are new or updated.

##### Fail On Invalid GeoRSS

If this parameter is No, features whose GeoRSS attribute does not contain valid GeoRSS will be output via the INVALID\_GEORSS port.

##### Merge Attributes

The specification of the optional Merge Attributes and associated prefix results in the copying of attributes from the INPUT feature into the constructed GeoRSS features. If a merge prefix is specified, then the string prefix is added at the start of each attribute from the original feature.

##### Feature Number Attribute

Since a GeoRSS document may contain several features, the transformer allows the sequence of GeoRSS features from a particular GeoRSS document to be numbered.

##### Feature Identifier

Furthermore, each GeoRSS feature can also be tagged by a unique sequence number that identifies it as coming from the same GeoRSS document; the optional Feature Identifier parameter can be used to specify the name for this attribute.

##### Additional URL Parameters

The value of this parameter is only used if the reader is accessing a dataset which is a URL. The value should be a space-separated list of space-separated name-value pairs. The name value pairs will be added to the dataset URL. A value must be provided for each parameter name. An empty string ( "" ) can be used to provide an empty parameter value.

##### Max Feedstore Entry Age

Specifies the age in days at which entries will be deleted from a feed's database. When the reader is run in UPDATE mode, the reader will delete any entries from the database for this feed which are older than the specified value. This ensures the feed database does not become arbitrarily large. If the value of this directive is not specified, or is 0, no entries will be deleted.

##### Feed Database Location

Specifies the filesystem directory which contains the databases that store information about the feeds that the GeoRSS reader has processed. This parameter is only read from the transformer if the reader is running in UPDATE mode. If no value is set, the FME temp directory will be used.

## ***Proxy Settings***

### Http Proxy URL

Specifies a proxy server that the reader will use when accessing a URL dataset. The port number of the proxy server can be set in the URL, or by using the Http Proxy Port parameter.

### Http Proxy Port

Specifies the port number of the proxy server indicated by the Http Proxy URL parameter. This parameter should only be used if the port number was not indicated in the Http Proxy URL parameter. This parameter is ignored if the Http Proxy URL parameter has no value.

### Http Proxy Username

Specifies the username to use when accessing a password-protected proxy server. This parameter is ignored if any of the Http Proxy URL, Http Proxy Password, or Http Proxy Authentication Method parameters have no value.

### Http Proxy Password

Specifies the password to use when accessing a password-protected proxy server. This parameter is ignored if any of the Http Proxy URL, Http Proxy Username, or Http Proxy Authentication Method parameters have no value.

### Http Proxy Authentication Method

Specifies the authentication method to use when accessing a password-protected proxy server. This parameter is ignored if any of the Http Proxy URL, Http Proxy Username, or Http Proxy Password parameters have no value.

## **Usage Notes**

This transformer works in conjunction with the GeoRSS Reader/Writer. For technical information on the GeoRSS Reader/Writer, choose FME Readers and Writers Reference from the Workbench help menu.

## **Transformer Category**

Web Services

## **FME Licensing Level**

FME Professional edition and above

## **fmepedia**

See fmepedia for additional information about GeoRSS and FME.

## **Technical History**

Associated FME function or factory: GeoRSSFactory

## **GMLFeatureExtractor**

Constructs GML2 documents from the input features and stores them in the specified attribute for the features that are output by the GML2 port. The GML2 documents written under the attribute conform to the GML SAFE schema.

### **Output Ports**

- GML2: The features that contain the GML2 document in the attribute specified by GML Attribute.

### **Parameters**

Flush Attribute

The GML document that is being constructed is accumulated until the value specified for the Flush Attribute. When this occurs, the newly received input feature is not part of the flushed feature, but rather is part of the next feature to be constructed. Values of selected attributes are transferred to the output features.

GML Attribute

The attribute of the feature containing the GML document.

### **Transformer Category**

Manipulators

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: GML2Factory

## **GMLFeatureReplacer**

Constructs features out of GML documents that are stored in an attribute of the input features. The features from the GML2 document are output with the attributes from the original feature, and merged if desired.

### **Output Ports**

- GML2: The features extracted from the GML2 document.
- INVALID\_GML: The features that triggered an error when extracting features from a GML2 document. This is a subset of the features output via the ORIGINAL output tag.

### **Parameters**

#### GML Attribute

The GML attribute to read from.

#### Fail On Invalid GML

If this parameter is set to No, features whose GML2 attribute do not contain valid GML2 will be output via this port.

#### Merge Attributes and Merge Attribute Prefix

The Merge Attributes parameter copies attributes from the input feature into the constructed GML features. If a merge prefix is specified, then the string prefix is added to the start of each attribute from the original feature.

#### GML XFMAP

The optional GML XFMAP parameter specifies the xFMaps document to used. If this is not specified, then the transformer assumes that the GML document contains features stored with the GML SAFE schema.

#### Feature Type Attribute

The name of the output GML feature type.

#### Feature Number Attribute

Since a GML2 document may contain several features, the transformer allows the sequence of GML features from a particular GML document to be numbered. This parameter can be used to specify the name of this attribute.

#### Feature Identifier

Furthermore, each GML feature can also be tagged by a unique sequence number that identifies it as coming from the same GML document. This parameter can be used to specify the name for this attribute.

### **Transformer Category**

Manipulators

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: GML2Factory

## **GOIDGenerator**

Calculates a GOID (Geographic Object Identifier) for each incoming feature, and adds it as a new attribute.

The GOID is a unique 128-bit number that incorporates the position of a feature with other numbers. The result is a unique value that may be used to distinguish features from each other. The 128-bit GOID is composed of 32 hex digits in an ASCII string. The first 16 characters correspond to the position, the next 10 to the time, the next 4 to the sequence number.

### **Parameters**

New GOID Attribute

The attribute that contains the calculated GOID.

### **Transformer Category**

Strings

### **Related Transformers**

This transformer is similar to the UUIDGenerator, which generates a completely unique ID unrelated to location for each feature.

### **Technical History**

Associated FME function or factory: @GOID

## **GridInQuestReprojector**

Reprojects feature coordinates from one coordinate system to another using the Grid InQuest reprojection library. This library allows you to use the GridInQuestReprojector to transform coordinates between ETRS89 (WGS84) and the national coordinate systems of Great Britain, Northern Ireland and the Republic of Ireland.

This transformer always reprojects from the source coordinate system to the destination coordinate system, tagging the features with the destination coordinate system on output. Any coordinate system set on the input features is ignored.

### **Parameters**

#### Source Coordinate System

The name of the source coordinate system

#### Source Vertical Datum

Choose a vertical datum from the list.

#### Destination Coordinate System

Choose the destination coordinate system.

#### Destination Vertical Datum

Choose a vertical datum from the list.

#### Interpolation Type (Raster Only)

The Interpolation Type will only have an affect on raster data. Cell values are interpolated in order to change the raster to the specified size; you have the choice of Nearest Neighbor, Bilinear or Bicubic interpolation methods. Nearest Neighbor is the fastest but produces the poorest image quality. Bilinear provides a reasonable balance of speed and quality. Bicubic is the slowest but produces the best image quality. Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Cell Size (Raster Only)

The Cell Size applies only to raster features. If Cell Size is set to Stretch Cells, the cell size of the raster will be adjusted to maintain the same number of rows and columns in the reprojected raster as there were in the input raster. If Cell Size is set to Square Cells, then the number of rows and columns as well as the spacing will be changed to maintain approximately the same cell ground area and form square cells where the horizontal and vertical cell sizes are equal. Like the Square Cells option, Preserve Cells will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Coordinate Systems

### **Technical History**

Associated FME function or factory: @Reproject

## **GtransAttributeReprojector**

Reprojects attributes holding coordinate values from one coordinate system to another using the Gtrans reprojection library (from the National Land Survey of Sweden), and the specified translation file.

### **Parameters**

X and Y Attribute

The name of the attributes containing the x and y coordinate values.

Translation File

Browse to the path containing the translation file.

The feature's coordinates are not altered by this transformer: only the values of the named X and Y attributes are changed.

### **Transformer Category**

Coordinate Systems

### **Technical History**

Associated FME function or factory: @Reproject

## GtransReprojector

Reprojects features to and from SWEREF99 using the Gtrans reprojection library (from the National Land Survey of Sweden) and the specified translation file.

### Parameters

#### Translation File

This is the translation file path.

#### Reverse Translation File (Raster)

If raster reprojection is required, a reverse translation file must also be supplied. This parameter defines the inverse translation file path.

#### Interpolation Type (Raster)

The Interpolation Type affects only raster data. Cell values are interpolated in order to change the raster to the specified size.

- **Nearest Neighbor** is the fastest but produces the poorest image quality.
- **Bilinear** provides a reasonable balance of speed and quality.
- **Bicubic** is the slowest but produces the best image quality.
- **Average 4** and **Average 16** have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Cell Size (Raster)

The Cell Size applies only to raster features.

- **Square Cells:** The number of rows and columns as well as the spacing will be changed to maintain approximately the same cell ground area and form square cells where the horizontal and vertical cell sizes are equal. Like the Square Cells option, Preserve Cells will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.
- **Stretch Cells:** The cell size of the raster will be adjusted to maintain the same number of rows and columns in the reprojected raster as there were in the input raster.

### Usage Notes

This transformer ignores the coordinate system set on input features, and does not tag output features with a coordinate system. If you want to take the output of the GtransReprojector transformer and write it to an output format that includes coordinate system metadata, you will need to specify which coordinate system to write, either by using a `CoordinateSystemSetter` transformer, or by setting the writer's `Coordinate System` parameter in the Workbench Navigator.

This transformer is unaffected by raster band and palette selection.

### Category

Coordinate Systems

### fmepedia

Gtrans information page

### Technical History

Associated FME function or factory: @Reproject



## HexDecoder

Decodes the given attribute into a new attribute, by converting its encoded hexadecimal value into a new ASCII string. Each hexadecimal couplet in the input attribute is converted into a single byte in the output string.

### Parameters

Attribute to Decode

Choose the attribute to convert.

New Decoded Attribute

The name of the new attribute. The default is `_decoded`.

Output Encoding

Set the output character encoding for the resulting attribute.

### Related Transformers

- You can use this transformer to decode a value that has been encoded from the `HexEncoder`.
- If decoding the hex string into a decimal number, you can use the `BaseConverter`.

### Transformer Category

Strings

### Technical History

Associated FME function or factory: `@Tcl2`

## HexEncoder

Encodes the selected attribute into a new attribute, by converting its ASCII value into a hexadecimal string. Each byte in the input attribute is converted into two hexadecimal characters in the output string.

This transformer is useful when you need to encode binary data into a simple character representation.

### Parameters

Attribute to Encode

Choose the attribute to encode.

New Encoded Attribute

The name of the new attribute. The default is `_encoded`.

### Related Transformers

- You can use the `HexDecoder` to reverse the effects of this transformer.
- If decoding the hex string into a decimal number, you can use the `BaseConverter`.

### Transformer Category

Strings

### Technical History

Associated FME function or factory: `@Tcl2`

## **HoleCounter**

Category: Calculators

Adds a new attribute whose value is the number of holes in the feature. If the feature is not a polygonal feature, 0 will be returned.

### **Technical History**

Associated FME function or factory: @NumHoles

## HTTPDeleter

Deletes a target by performing an HTTP DELETE operation on the specified URL, storing the results in the specified target attribute.

### Output

- The HTTP Response status code will be stored in the `_http_status_code` attribute.
- The HTTP Status Code will be logged with each request. For more information on HTTP Status Codes, please refer to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.

### Parameters

#### Target URL

The target URL to perform the operation on.

#### Target Attribute

The results of the operation will be stored in this attribute.

#### Continue on Error

If an error occurs and Continue On Error is set to Yes, the target attribute will be returned empty and the error message will be logged, but the translation will continue. However, if an error occurs and Continue On Error is set to No, the translation will fail.

#### Use encoding from HTTP Response Headers

If this parameter is set to Yes, then the HTTP Response will be examined for encoding information, and the target attribute will be tagged with this encoding. If no encoding can be found in the HTTP Response, the target attribute will be tagged as binary data. If the Use encoding from HTTP Response Headers parameter is set to No, the downloaded data will be considered to be text data in the system encoding.

#### HTTP Request Headers

This parameter can be used to provide custom headers for the HTTP Request. Each header should be entered on a single line.

#### HTTP Authentication Parameters

The optional HTTP Authentication Username, HTTP Authentication Password and HTTP Authentication Method parameters may be set for accessing a password-protected HTTP server. Both Basic and Digest access authentication methods are supported.

Note that although the HTTP basic access authentication is a mechanism designed to allow a client to provide credentials to a server on the assumption that the connection between them is trusted and secure, be aware that any credentials passed from client to server can be easily intercepted through an insecure connection.

#### Proxy Parameters

The optional Proxy URL, Proxy Port, Proxy Username, Proxy Password, and Proxy Authentication Method parameters may be set for organizations that require Internet access via an HTTP proxy server.

### Transformer Category

Web Services

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @Http

## HTTPFetcher

Fetches a target by performing an HTTP GET operation on the specified URL, storing the results in the specified target attribute.

### Output

The HTTP Response status code will be stored in the `_http_status_code` attribute.

The HTTP Status Code will be logged with each request. For more information on HTTP Status Codes, please refer to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.

### Parameters

#### Target URL

The target URL to perform the operation on.

#### Target Attribute

The results of the operation will be stored in this attribute.

#### Continue on Error

If an error occurs and Continue On Error is set to Yes, the target attribute will be returned empty and the error message will be logged, but the translation will continue. However, if an error occurs and Continue On Error is set to No, the translation will fail.

#### Use encoding from HTTP Response Headers

If this parameter is set to Yes, then the HTTP Response will be examined for encoding information, and the target attribute will be tagged with this encoding. If no encoding can be found in the HTTP Response, the target attribute will be tagged as binary data. If this parameter is set to No, the downloaded data will be considered to be text data in the system encoding.

#### HTTP Request Headers

This parameter can be used to provide custom headers for the HTTP Request. Each header should be entered on a single line.

#### HTTP Authentication Parameters

The optional HTTP Authentication Username, HTTP Authentication Password and HTTP Authentication Method parameters may be set for accessing a password-protected HTTP server. Both Basic and Digest access authentication methods are supported.

Note that although the HTTP basic access authentication is a mechanism designed to allow a client to provide credentials to a server on the assumption that the connection between them is trusted and secure, be aware that any credentials passed from client to server can be easily intercepted through an insecure connection.

#### HTTP Authentication Parameters

The optional HTTP Authentication Username, HTTP Authentication Password and HTTP Authentication Method parameters may be set for accessing a password-protected HTTP server. Both Basic and Digest access authentication methods are supported.

Note that although the HTTP basic access authentication is a mechanism designed to allow a client to provide credentials to a server on the assumption that the connection between them is trusted and secure, be aware that any credentials passed from client to server can be easily intercepted through an insecure connection.

#### Proxy Parameters

The optional Proxy URL, Proxy Port, Proxy Username, Proxy Password, and Proxy Authentication Method parameters may be set for organizations that require Internet access via an HTTP proxy server.

### Transformer Category

Web Services

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @Http

## HTTPFileUploader

Uploads a file's contents by performing an HTTP PUT or POST operation on the specified URL, storing the results in the specified target attribute.

### Output

- The HTTP Response status code will be stored in the `_http_status_code` attribute. The HTTP Response status code will be stored in the `_http_status_code` attribute. The HTTP Status Code will be logged with each request. For more information on HTTP Status Codes, please refer to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.
- The file that is to be uploaded is the value of the attribute identified by the Upload File Path parameter.

### Parameters

#### Target URL

The target URL to perform the operation on.

#### Target Attribute

The results of the operation will be stored in this attribute.

#### Upload Method

Choose PUT or POST.

#### Upload File Path

This parameter specifies the file whose contents will become the body of the HTTP request sent to the server. Select the attribute that contains the file path.

#### Upload Content Type

This parameter allows you to specify the value of the Content-Type header in the HTTP Request.

Note that this parameter takes precedence over the Content-Type provided in the HTTP Request Headers parameter, if one is provided.

#### Continue on Error

If an error occurs and Continue On Error is set to Yes, the target attribute will be returned empty and the error message will be logged, but the translation will continue. However, if an error occurs and Continue On Error is set to No, the translation will fail.

#### HTTP Request Headers

This parameter can be used to provide custom headers for the HTTP Request. Each header should be entered on a single line.

#### Use encoding from HTTP Response Headers

If this parameter is set to Yes, then the HTTP Response will be examined for encoding information, and the target attribute will be tagged with this encoding. If no encoding can be found in the HTTP Response, the target attribute will be tagged as binary data. If this parameter is set to No, the downloaded data will be considered to be text data in the system encoding.

#### HTTP Authentication Parameters

The optional HTTP Authentication Username, HTTP Authentication Password and HTTP Authentication Method parameters may be set for accessing a password-protected HTTP server. Both Basic and Digest access authentication methods are supported.

Note that although the HTTP basic access authentication is a mechanism designed to allow a client to provide credentials to a server on the assumption that the connection between them is trusted and secure, be aware that any credentials passed from client to server can be easily intercepted through an insecure connection.

## Proxy Parameters

The optional Proxy URL, Proxy Port, Proxy Username, Proxy Password, and Proxy Authentication Method parameters may be set for organizations that require Internet access via an HTTP proxy server.

## **Transformer Category**

Web Services

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: @Http



## HTTPUploader

Uploads a message by performing an HTTP PUT or POST operation on the specified URL, storing the results in the specified target attribute. The HTTP Response status code will be stored in the `_http_status_code` attribute.

### Output

- The HTTP Response status code will be stored in the `_http_status_code` attribute. The HTTP Status Code will be logged with each request. For more information on HTTP Status Codes, please refer to <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.
- The message that is to be uploaded is the value of the attribute identified by the Upload Body parameter.

### Parameters

#### Target URL

The target URL to perform the operation on.

#### Target Attribute

The results of the operation will be stored in this attribute.

#### Upload Method

Choose PUT or POST.

#### Upload Body

This parameter specifies the body of the HTTP request sent to the server. Select the attribute that contains the request data.

#### Upload Content Type

This parameter allows you to specify the value of the Content-Type header in the HTTP Request.

Note that this parameter takes precedence over the Content-Type provided in the HTTP Request Headers parameter, if one is provided.

#### Continue on Error

If an error occurs and Continue On Error is set to Yes, the target attribute will be returned empty and the error message will be logged, but the translation will continue. However, if an error occurs and Continue On Error is set to No, the translation will fail.

#### HTTP Request Headers

This parameter can be used to provide custom headers for the HTTP Request. Each header should be entered on a single line.

#### Use encoding from HTTP Response Headers

If this parameter is set to Yes, then the HTTP Response will be examined for encoding information, and the target attribute will be tagged with this encoding. If no encoding can be found in the HTTP Response, the target attribute will be tagged as binary data. If this parameter is set to No, the downloaded data will be considered to be text data in the system encoding.

#### HTTP Authentication Parameters

The optional HTTP Authentication Username, HTTP Authentication Password and HTTP Authentication Method parameters may be set for accessing a password-protected HTTP server. Both Basic and Digest access authentication methods are supported.

Note that although the HTTP basic access authentication is a mechanism designed to allow a client to provide credentials to a server on the assumption that the connection between them is trusted and secure, be aware that any credentials passed from client to server can be easily intercepted through an insecure connection.

#### Proxy Parameters

The optional Proxy URL, Proxy Port, Proxy Username, Proxy Password, and Proxy Authentication Method parameters may be set for organizations

that require Internet access via an HTTP proxy server.

**Transformer Category**

Web Services

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @Http

## **ImageFetcher**

Fetches an image by performing an HTTP GET operation on the specified URL, and then returning the resulting data as the geometry of a raster feature.

### **Parameters**

#### Image URL

Enter the URL (for example, <http://www.url.com>) or select an attribute that contains a URL.

#### Image Type

Choose the image type.

#### Continue on Error

If an error occurs and Continue On Error is set to Yes, the output feature will have no geometry and the error message will be logged, but the translation will continue.

However, if an error occurs and Continue On Error is set to No, the translation will fail.

#### Proxy URL, Proxy Port, Proxy Username, Proxy Password, Proxy Authentication Method

These parameters may be set for organizations that require Internet access via an HTTP proxy server.

### **Usage Notes**

The https protocol is not currently supported.

### **FME Licensing Level**

FME Professional edition and above

### **Transformer Category**

Web Services

### **Technical History**

Associated FME function or factory: @Http

## ImageRasterizer

Draws input point, line and polygon features onto a color raster filled with the background color. The `fme_color` attribute of the input vector features is used to generate pixel values. Features without an `fme_color` attribute will be discarded.

### Input

- INPUT: The Input port takes the vector features that will be rasterized. These features are valid only if :
  - they are 3D
  - they have an `fme_color` attribute

### Output

- RASTER: The raster drawn from a group of features.

### Parameters

#### Group By

If the Group By parameter is set to an attribute list, one raster per group will be produced.

#### *Raster Properties*

Size Specification, Number of Columns (cells), Number of Rows (cells), X Cell Spacing, Y Cell Spacing

To set the size of the output raster, specify either the dimensions or the cell size.

To set the output raster size using dimensions, set the Size Specification to RowsColumns and specify values for both the Number of Columns and Number of Rows.

To set the output raster size using cell size, set the Size Specification to CellSize and specify values for both the X Cell Spacing and Y Cell Spacing.

#### *Interpretation*

#### Interpretation Type

This parameter sets the interpretation of the output raster.

Pixel values for red, green, and blue bands will be taken from the corresponding component of a feature's `fme_color` attribute. Pixel values for gray bands will be the average of the `fme_color` components.

#### Alpha Value

This parameter sets pixel values for alpha bands.

#### *Background*

#### Background Color

The Background Color parameter sets the background color for red, green, blue, or gray bands.

Click the colored square to the right of the text field, or edit the contents of the field directly. The color must be specified as `<red>,<green>,<blue>` where each of `<red>`, `<green>`, and `<blue>` is a number between 0 and 1.

#### Background Alpha Value

The Background Alpha parameter sets the background value for any alpha bands on the raster. It must also be a number between 0 and 1.

#### Fill Background with Nodata

If the Fill Background with Nodata parameter is Yes, the background color will also be flagged as the nodata value for each raster band.

## ***Anti-Aliasing***

### Anti-Aliasing

If the Anti-Aliasing parameter is Yes, the output lines will be smoothed using an anti-aliasing algorithm.

### Tolerance

The Tolerance parameter is the maximum normalized distance from a line segment or polygon vertex to a pixel to be rendered. For example a tolerance of 1.0 will draw all pixels touched by the input vector line, while a tolerance of 0.0 will draw only those pixels where the input vector line passes directly through their center. Tolerance can only be selected when anti-aliasing is off.

## ***Ground Extents***

### Ground Extents

If the The Ground Extents parameter is set to *Use input data ground extents*, which means the extents are not explicitly specified, the output raster extents will be determined by the union of the bounding boxes of the valid input vector features. If the Ground Extents parameter is set to *Specify ground extent*, the remaining Ground Extents parameters are used to specify the extents of the output raster.

## **Transformer Category**

Rasters

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: VectorToRasterFactory

## **IndividualGeometriesFilter**

Filters aggregate features based on the type of aggregate.

### **Output Ports**

- **CONTAINS:** If the aggregate is set up such that each part is independent from the others, and should be interpreted as its own complete geometry, then the feature will be sent to the CONTAINS port.
- **NOT\_CONTAINS:** Aggregates that do not match the criteria for CONTAINS will be sent to the NOT\_CONTAINS port.
- **INVALID:** This transformer works only on aggregates, so features containing any other geometry type will be sent to the INVALID output port.

### **Example**

An example where it may be useful to know whether an aggregate contains individual geometries is when the source dataset supports the notion of multiple independent geometries per feature but the destination dataset does not.

In this case, it may be desirable to split up the aggregate into multiple separate features.

### **Transformer Category**

Filters

### **Technical History**

Associated FME function or factory: @Geometry

## **IndividualGeometriesSetter**

Provides the ability to set up an aggregate where each part is independent from the others, and its own complete geometry.

For example, if an aggregate enters and it contains two parts, a line and a polygon, it will leave containing the same two parts, but the difference will be that each part should now be interpreted as its own individual geometry. For example, a database writer that supports this concept may then write out each individual geometry to its own geometry column.

### **Output Ports**

- **AGGREGATE:** Aggregates will be sent out via the AGGREGATE port.
- **INVALID:** This transformer works only on aggregates, so features containing any other geometry type will be sent to the INVALID output port.

### **Parameters**

Set to Contain Individual Geometries

When this parameter is set to No, the aggregate will represent a single geometry comprising multiple parts

When this parameter is set to Yes, each part in the aggregate will be interpreted as its own separate geometry, independent from the other parts.

### **Transformer Category**

Infrastructure

### **Technical History**

Associated FME function or factory: @Geometry

## **InsidePointExtractor**

Category: Calculators

Adds attributes holding the coordinates of a point guaranteed to be inside the area feature. The geometry of the feature is not changed by this transformer.

If the feature is not an area, then the attributes will not be given any value.

This is useful if the location of an interior point, or centroid, is needed as attributes on an area feature. If an actual text feature is to be created at the interior point, then the `LabelPointReplacer` transformer should be used instead. If a point feature is to be created, the `Inside-PointReplacer` should be used.

Note: If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, ellipses are regarded as areas; they are otherwise regarded as points.

### **Technical History**

Associated FME function or factory: `@GeneratePoint`



## **InsidePointReplacer**

Replaces the geometry of the area feature with a point that is guaranteed to be inside the area.

### **Output Port**

- **INSIDEPOINT:** This port contains the result.
- **UNTOUCHED:** Non-area features that go through the transformer will be output here, unchanged.

### **Usage Notes**

To get the coordinates of a point inside an area, use the `InsidePointExtractor`.

To generate text labels inside of an area, use the `LabelPointReplacer`.

### **Geometry Handling**

If the Geometry Handling parameter (under Advanced Workspace Settings) ellipses are regarded as areas; they are otherwise regarded as points.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: `@GeneratePoint`

## Inspector

Sends features to the FME Data Inspector (or the FME Universal Viewer) for display. (The application that opens depends on the *Inspector Application* setting in Tools > FME Options > Workbench.)

When the workspace run is completed, any features that enter this transformer will be displayed in the FME viewer application. The Inspector also allows you to view the internal FME Feature Store (FFS) data as it is being processed at that particular moment. Depending on the placement of the Inspector, FME will open the Inspector and load the data in a new view.

All Inspector transformers used in a workspace will be displayed in the same view.

A feature type in the Inspector application will have the same name as its corresponding Inspector transformer.

## Parameters

### Group By

If any Group By attributes are specified, their values are appended to the Inspector name to create a feature type for display in the Inspector.

### Vector Parameters

#### Pen Color Override

The pen color override takes precedence over the `fme_color` attribute on the feature. It has the same effect as setting `fme_color` attribute on the feature to the selected override color for the purposes of inspection.

For point cloud features, if the cloud contains color components for each point, the Inspector will use the color specified in the component. In this case, the pen color override will not be used.

If the cloud does not contain color components for each point, the following priority is used when selecting the paint color for each point:

1. If there is a Pen Color Override selected, the inspector application will paint all points in the cloud with the selected override.
2. If the cloud does not contain color component and a Pen Color Override is not selected, the viewer application will paint all points in the clouds based on the `fme_color` attribute on the feature.
3. If the cloud does not contain color component and there is no Pen Color Override or `fme_color` attribute on the feature, the viewer application will use a color ramp based on the z range from blue for the minimum value to red for the maximum value. When the intensity component exists, the intensity value becomes the lightness of the color.

#### Area Fill Color Override

The Area Fill Color override sets the `fme_fill_color` for the feature for the purposes of inspection.

Area Fill Color Override is not applicable for any feature that does not need a fill such as point clouds, line features, and point features.

### Raster Parameters

#### Reduction Type

This parameter offers different methods to speed up the inspection of rasters:

- No Reduction: Rasters will be visualized in full detail.
- Resample: Large rasters will be resampled down to a smaller size. This is useful if only an overview (that is, not full detail) is required for viewing.
- Subset: Only a subset of each raster will be written. This subset is defined by the Subset Start Column, Subset Number of Columns, Subset Start Row, and Subset Number of Rows parameters.
- Bounding Box Only: Only the bounding box of the raster data will be written.

Subset Start Column, Subset Number of Columns (cells), Subset Start Row, Subset Number of Rows (cells)

Applicable when Subset is selected for the Reduction Type parameter. These parameters identify the position in the input raster from which the subset will be taken, as well as the size of the subset.

## Point Cloud Parameters

### Thinning Type

This parameter offers different methods to speed up the visualization of point clouds by removing points from the cloud:

- Keep Every Nth Point: Keeps every <sampling amount>th point, and discards the remaining points for each input point cloud.
- Maximum Number of Points: Sets a maximum number of points to visualize for each input point cloud.

### Amount


If Thinning Type is "Keep Every Nth Point", the amount specifies how often the points are retained. For example, an amount of 2 will result in every other point of the input cloud feature will be present in the output point cloud.

If the Thinning Type is "Maximum Number of Points", the amount specifies the maximum points in the output point cloud. For example, an amount of 100 will result in points in the input cloud being dropped so the maximum number of points in the output cloud is 100.

### Usage Notes

This is a very useful tool for debugging workspaces. Keep in mind, however, that you are viewing the internal FME data, which may not be the same as the data eventually written to your output dataset. This is because FME does not apply any format-imposed restrictions on the data.

### Shortcuts

- To attach an Inspector transformer: right-click on a selected reader feature type or transformer and choose *Connect Inspector(s)*.
- Use this toolbar button  to place an Inspector transformer in a workspace.
- To immediately open the Viewer, right-click on a reader or writer feature type and select *Inspect*.

### Redirecting Output to an Inspector

In some cases you might want to inspect output data, but you don't want to actually have to write the data to do so. For example, in a translation that applies updates to a spatial database, you will probably want to check the output before actually committing an update. The answer is the *Redirect to Inspector Application* setting.

When this setting is applied, the output from a translation is redirected away from the specified output and sent directly to the viewer specified by the *Inspector Application* setting in Tools > FME Options > Workbench.

The simplest way to turn on this feature is to select *Writers > Redirect to Inspector Application* from the Workbench menu bar. This is a toggle setting, meaning that each selection turns the setting on or off.

### Transformer History

This transformer was previously named the Visualizer.

### References

The FME Data Inspector help files contain additional information about viewing and saving raster, point cloud, and vector data.

### Transformer Category

Infrastructure

## Intersector

Computes intersections between all input features, breaking lines and polygons wherever an intersection occurs. In addition, all overlapping segments are reduced to one segment before being output.

### Output Ports

- **INTERSECTED:** The intersected features are output to this port. If the List Name parameter was specified, these features will have an attribute containing their number of overlapping input features. They will also have all attributes of the original features.
- **NODE:** The locations of every intersection are represented by point features and are output to this port. The transformer parameters modify which attributes are included on the output features.

### Parameters

#### Group By

If you select Group By attributes, only those features with the same Group By attribute values will be processed. If you do not select Group By attributes, then all features will be processed.

#### Duplicate Nodes at Each Elevation

The creation of nodes can be calculated in 3D, if requested. Constructing nodes in 3D would mean that line segments would only share a node if they shared the same Z value at the point they intersected. Constructing nodes in 2D would mean that all intersecting segments would share a common node, regardless of their respective Z values.

Consider, for example, a situation where two lines (that crossed) represented roads, where one road was an overpass above the other road. Suppose these two lines had differing elevations. If you constructed nodes in 3D, these two roads would not be linked to the same node where they crossed. Two nodes would be produced at the crossing point – each one with a different Z value. If you constructed nodes in 2D, these lines would both link to a common node, which would be present at the location where they crossed. In either the 2D or 3D case, the full dimensionality of the input is preserved in the output – 3D features are never converted to 2D. The 2D or 3D choice only indicates how the nodes are created and which lines are linked to them; it does not affect the dimension of the features that are output.

If the Duplicate Nodes at Each Elevation parameter is set to **Yes**, then whenever 3D lines intersect at differing heights, two 2D nodes will be output via the NODE port. Each node will have the same x and y coordinates, but a different node number.

If the Duplicate Nodes at Each Elevation parameter is set to **No**, then whenever 3D lines intersect at differing heights, a single 2D node is output at the intersection point.

#### Overlap Count Attribute

The Overlap Count Attribute parameter names an attribute that will be added by the transformer, containing the number of collinear input lines that overlapped the output segment.

#### Segment Count Attribute

The Segment Count Attribute parameter, if specified, names an attribute that will be added by the transformer, containing the number of segments into which the segment's original feature was divided.

If an input feature was broken into n output segments, each of those segments will have an attribute named <attribute name> which has a value of n.

#### List Name

If the optional list name is supplied, a list of all the attributes of each lines which overlapped an output segment is made. This allows later inspection of overlapping segment attributes.

#### Separate Collinear Segments

This parameter causes overlapping segments not to be merged into a single segment: one copy is output for each original feature sharing the segment. Each such segment will have the respective original feature's attributes as its main attributes, and attributes from all other collinear features will be added as a list attribute, if the list name was supplied.

When a coverage of polygons is input, the set of topologically significant lines which form their boundaries is output.

## Split Self-Intersecting Features

If the Split Self-Intersecting Features parameter is set to Yes, self-intersections in the input features are removed by splitting the feature.

No feature-to-feature comparisons are made. In this case, the value set to the Overlap Count Attribute will be the number of features that result from removing self-intersections. If the feature did not self-intersect, the attribute will be set to 1.

If the segment had several overlapping input features, the attributes of each of the input features will be added to the feature in the list identified by List Name, if one was specified. In any case, each output feature is also assigned the attributes of one of its original input features. This transformer also adds a "direction" attribute for each attribute resulting from the List Name parameter, labelling it as *same* if the geometry is oriented in the same direction, and *opposite* if the geometry is oriented in the opposite direction to the current geometry.

### Example



## Using the Intersector and ListConcatenator to Solve Problems

What if you have linear street centerlines and, at each intersection point, you'd like to know which streets come together? The output should be a set of points, each with a single string attribute containing a comma-separated set of the street names.

You can solve this by setting up a workspace that routes all the street centerlines into an Intersector. Adjust the parameters of the Intersector to supply a list name; for example, `all_streets`.

Let's assume that the input street lines had an attribute called `NAME`. Now, among other things, the `NODE` output of the Intersector will have an unqualified list on it called `all_streets{}.NAME`. This list will hold the names of all the streets that intersect at each particular point (or `NODE`) that is output.

To turn the list of `NAME`s into a single string, add a `ListConcatenator` transformer and run the `NODE` features into it. Then set up the `ListConcatenator`'s parameters so that it would put the contents of the `all_streets{}.NAME` list together, separated by commas, into the "result" attribute. Then route the output of the `ListConcatenator` to an output file, and ensure that the "result" attribute was routed to an attribute in the output. After running the translation, you will have the desired result.

Note that you could also access the individual street names by "exposing" some elements of your list (by right clicking on the attribute unqualified list name (in our example, `all_streets{}.NAME`) and saying "Expose Elements", and entering the number of elements to expose. You'd then have to do something with those elements in your translation. (The disadvantage of this approach is that you need to know ahead of time how many list elements you want to work with -- so if 3 streets intersect at the same node and you only set yourself up to handle two, you'd have to do something special to handle that.)

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will remain as arcs and ellipses; they will otherwise be stroked to lines and areas, respectively.

### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: `IntersectionFactory`

## Joiner

Category: Database

A Joiner joins attributes from an external database (most popular databases are supported) to other spatial (or non-spatial) features as they are being processed through a translation.

Queries a database to retrieve attributes associated with a feature. One or more feature attributes (foreign keys) are joined to one or more columns (primary keys) in a table in the database, and the values from the matching table row(s) are added as feature attributes.

Access MDB files, ODBC connections, Oracle 10g/8i/9i, Oracle 7 and 8 attribute tables, ESRI ArcSDE, dBase III files, comma separated value (CSV) files, PostgreSQL/PostGIS tables, Microsoft Excel (XLS) files, Microsoft SQL Server, MySQL, SQLite, and DB2 tables can all be used as the database.

It is a very powerful transformer with many performance-related settings that are worth being aware of.

A wizard is used to set the connection type, data source, connection parameters, and key field properties.

**Note:** The information panes that the wizard displays will depend on your database connection. You might not see all of the panes listed below.

### Specify External Database

Choose the location of the table that holds the values you want to add to your features.

Note that when you connect to Microsoft SQL Server, there is also an option to check the **Use Windows Authentication** checkbox, which enables Windows Authentication for the database. If this option is selected, the Username and Password fields are ignored.

### Select Database Table

The wizard displays a list of tables. Select one of the tables and the values will be retrieved and added to your features.

### Identify Database Keys

The wizard displays the table columns. Choose the ones you want to use as keys and move them to the Keys list.

### Match Keys

Database keys are shown with feature attributes. Pair each database key with its corresponding feature attribute. The key pairs will be used to relate database rows with individual features.

### Select Database Columns

The wizard displays Database Columns. Select from the list to add to the features. Values from the columns will be retrieved from the database and added to the features identified in the Match Keys screen.

### Select Relationship Type

Indicate the type of relationship between the database rows and each feature. This will describe how many rows will match to each feature and what action the Joiner will take if the expected number is not found. If the cardinality of the relation is 1 to many, then a list name must be specified to hold the attribute data retrieved from the database. Each row retrieved is added to the attribute list. Subsequent transformers can perform operations on the resulting list.

Relationship Type	Description
1:0..1	One to zero or one. A feature can match to a single record or none at all. But more than one match is not permitted and will cause an error.
1:0..1+	One to zero or more. A feature can match to any number of records or none at all. But only the first record found will be matched.
1:1	One to one. Each feature <b>MUST</b> match to one record only. Zero matches or more than one match will cause an error.
1:M	One to many. A feature can match to any number of records. Each record will be matched and its data added as a List instead of regular attributes.

Especially the 1:1 join is very strict, when in doubt, use 1:0..1+ or 1:M

### Specify Attribute Prefix

To prevent overwriting existing feature attributes, you can optionally specify a prefix that will be applied to each database column name when it is added to the feature.

### Optimize Joiner

Specify the number of rows to cache locally if you do not want to accept the default of 5000. You can optionally specify an SQL query to preload the cache.

Normally the cache is filled with records as they are matched in the database. However, the cache can be preloaded (i.e., filled with a specific set of data before matching takes place) by issuing a prefetch query. This prefetch query can select an entire table or a selected part of a table which is most likely to be matched by the feature attributes.

A prefetch query which is known to retrieve all possible matches is called an *exhaustive query*. When a match cannot be found within the cached results of an exhaustive query, it is assumed that no match exists in the database; thus the database will not be further consulted.

The **Exhaustive Query** checkbox indicates whether a prefetch is exhaustive. Even when this box is not checked, however, any prefetch query is assumed to be exhaustive when it does not contain a WHERE clause, and is of the form:

```
SELECT * from TableName
```

**Note:** When the **Exhaustive Query** checkbox is checked, or there is no WHERE clause, FME considers the prefetch to be exhaustive, and the cache size limit will be ignored. This is because the cache must contain all results from the query.

For example, a number of FME features of type "roads" require a database match. The database table (myrecords) has a field (record\_type) with a number of values; roads, highways, avenues, streets. The FME features will only ever be matched to where record\_type=roads so the overall join process would be much more efficient if the following prefetch was issued:

```
SELECT * from myrecords where record_type = 'roads'
```

### Trim Key Fields (appears with character fields)

Engaging trimming of key fields may significantly reduce performance and should only be done if key column values in the database are known to contain trailing spaces. It has no effect if an exhaustive prefetch query is used (see above for an explanation of exhaustive query).

### Specify List Name

When many database rows are related to a single feature, a name for an attribute list is required. This list will hold all the rows from the database that were related to the feature.

### Example using Joiner transformer

A cache is used by the Joiner transformer, which matches records to graphic features. When FME reads a matched database record, it will hold it in a cache. For subsequent features, this cache is checked for a match before FME checks the database. The advantage there is that database records that are matched by multiple features do not cause FME to do a database read each time because the information is already held in memory (cached). This makes the join quicker and results in less network traffic.

Here is a good example...

```
@Relate: Database query statistics for table `JOINER:MY_TABLE':
```

```
7 queries made of which 0 were sequential duplicates
```

```
and 1 hit the record cache of 3 records (14% overall cache hit)
```

Firstly this doesn't explicitly state how many records were matched, but we can make a good guess that it was four. Three features matched records, all with differing IDs, and FME added these records to the cache. The fourth matching feature didn't have to query the database because it hit one of the records held in cache. That's where the 14% comes in, by the way. There were seven queries and one of these matched a cached record ( $1/7 = 14\%$ ). So FME has automatically reduced network traffic on this query by 14%.

The sequential duplicates part, by the way, indicates how many features had identical key IDs. For example...

```
@Relate: Database query statistics for table `JOINER:MY_TABLE':
```

**7 queries made of which 3 were sequential duplicates  
and 2 hit the record cache of 2 records (71% overall cache hit)**

Here there were two hits on the cache, but also three features duplicated ( $2+3 = 5$  and  $5/7 = 71\%$ ).

So this affects performance, but what can you do to help? There are two settings that can be applied within the Joiner.

The first setting is cache size. Usually only a subset of records are cached. The cache size setting specifies how many records this subset will be. Once the cache is filled, new records can only be added by dropping existing ones. Therefore the larger the number the more records will be held in memory and the less database reads will occur.

Obviously the size of the setting will depend on how many records you have, how often they will be matched by an individual record and how much memory your system contains. At a certain point it will be more efficient to read the database regardless, if the cache is holding so many records your system runs out of memory.

**Tip:** Set a cache size that is appropriate for the size of your dataset and the number matches that are likely to be made in that cache.

A second cache-related setting is the prefetch. Instead of filling the cache with records as they are matched, the cache can be preloaded (i.e., filled with a specific set of data before matching takes place) by the user issuing a prefetch query. This prefetch query can select an entire table or a selected part of a table which is most likely to be matched by the feature attributes.

For example, a number of FME features of type 'roads' require a database match. The database table (myrecords) has a field (record\_type) with a number of values; roads, highways, avenues, streets. The FME features will only ever be matched to where record\_type=roads so the overall join process would be much more efficient if the following prefetch was issued...

```
select * from myrecords where record_type = 'roads'
```

**Tip:** Where Joiner transformers will match only on a known subset of records within a table, it will often be more efficient to prefetch that subset of records before matching takes place.

## Technical History

Associated FME function or factory: @Relate



## JSONExploder

Extracts portions of JSON (JavaScript Object Notation) formatted text into new FME features.

### Output

For each newly created feature, the attribute identified by the JSON Attribute parameter will contain JSON text referred to by the JSON query.

The `json_type` attribute will contain the JSON type of the text (object, array, string, etc) and the `json_index` parameter will contain the object key or array index of the JSON value.

### Parameters

#### JSON Attribute

The JSON Attribute parameter identifies the feature attribute which contains the JSON text.

#### JSON Query

The JSON Query parameter is the JSON query referring to the JSON values which will become new FME features.

A JSON query is a mechanism to refer to values inside JSON text. The outermost JSON value, which must be an object or an array, is always referred to by the term "json". Contained values can be referred to using JavaScript-like square bracket index operators. A value in an array can be referred to using its zero-based position in the array (for example, `json[2]` for the third element).

A value in an object can be referred to using its object key name. For example:

```
json["key"]
```

All of the values in an array or object can be referred to collectively using a wildcard index. For example:

```
json[*]
```

The query used by this transformer can have multiple expressions, separated by a '+' operator. This allows the transformer to refer to values in different areas of the JSON text. For example:

```
json["resultSet_1"][*] + json["resultSet_2"][*]
```

See below for more information on JSON queries.

#### Explode as Format

If the JSON text is in GeoJSON or ESRIJSON format, you can select "Explode as format" to "GeoJSON" or "ESRIJSON". This will inform JSONQueryFactory that GeoJSON or ESRIJSON features need to be automatically recognized and extracted from the query results.

The default format, "JSON", parses the JSON text as plain JSON. If no GeoJSON or ESRIJSON features were constructed, a warning will be issued and the text will be treated as plain JSON.

#### Load Keys as Attributes, Prefix New Attributes With

If the JSON query produces a JSON Object, the keys may be used to produce additional attributes for the new feature(s) by setting "Load Keys as Attributes" to "Yes".

If this option is selected, you can optionally provide a string with which to prefix the newly-created attributes in the "Prefix new attributes with" field. If the selected format is GeoJSON or ESRIJSON, the format-specific keys will not be added as attributes, but will instead be handled by the appropriate format parser.

If the JSON query cannot be fully evaluated, a message will be logged, and the translation will continue.

## JSON Queries

A JSON query is a simple mechanism to refer to values within a JSON document. A query is made up of one or more expressions, which are separated by a + operator. There are three types of expressions: JSON structure expressions, JSON property expressions and string literal expressions.

### *JSON Structure Expressions*

A JSON structure expression can refer to values in a JSON document. The outermost JSON element, which must be an array or an object, is always referred to by the term `json`, and this term must appear at the beginning of every JSON structure expression. The child elements can be referred to using JavaScript-like square bracket index operators. For example, if the outermost element is an array, the first element of the array can be referred to by the expression `json[0]`, the second element can be referred to by the expression `json[1]`, and so on. Likewise, if the outermost JSON element is an object, with keys "name" and "address", then the values of these keys can be referred to by the expressions `json["name"]` and `json["address"]` respectively.

These index operators can be cascaded. For example, if the outermost JSON element is an object with a key and "address" whose value is an object containing keys "city" and "province", then these values can be referred to by the expressions `json["address"]["city"]` and `json["address"]["province"]`.

In order to refer to all of the values in an array or object, a wildcard index `*` can be used. For example, if the outermost JSON element is an array, then the expression `json[*]` refers to every element in the array.

### ***JSON Property Expressions***

A property expression is a structure expression as described above, followed by a `.` (dot) operator and a property name. Currently, the only supported properties are `type` and `size`. The `type` property returns the type of the JSON value referred to by the JSON structure expression. For example, if the outermost JSON element is an array, and the first element of the array is a string, then the expression `json[0].type` would have a value of `string`. The `size` property, which can only be applied to an array, returns the number of elements in the array.

### ***String Literal Expressions***

A string literal expression is simply a quoted string value, such as `"this is a string literal expression"`.

### **FME Licensing Level**

FME Professional edition and above

### **Transformer Category**

Web Services

### **Technical History**

Associated FME function or factory: `JSONQueryFactory`

## **JSONExtractor**

Extracts portions of JSON (JavaScript Object Notation) formatted text into feature attributes.

### **Parameters**

#### JSON Attribute

The JSON Attribute parameter identifies the feature attribute which contains the JSON text.

#### Target Attribute

This parameter identifies the feature attribute into which the result of the JSON query should be put.

#### JSON Query

This parameter is the JSON query whose value will be stored in the target attribute.

A JSON query is primarily used to refer to values in some JSON text. They can also be used to extract some metadata about the JSON text, such as the type (object, array, string, etc.) of JSON value or the number of elements in an array. The outermost JSON value, which must be an object or an array, is always referred to by the term "json". Contained values can be referred to using JavaScript-like square bracket index operators.

A value in an array can be referred to using its zero-based position in the array (for example, `json[2]` for the third element).

A value in an object can be referred to using its object key name

```
json["key"]
```

All of the values in an array or object can be collectively referred to using a wildcard index

```
json[*]
```

The query used by this transformer can have multiple expressions, which are separated by a '+' operator, as well as string literal values. This allows more complex attribute values to be easily created from the JSON text. For example:

```
json["name"]["first"] + " " + json["name"]["last"]
```

The JSONExploder transformer contains more information on JSON queries.

If the JSON query cannot be fully evaluated, a message will be logged, and the feature will be output without setting the target attribute.

### **Transformer Category**

Web Services

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: JSONQueryFactory

## **KMLPropertySetter**

Sets common properties for groups of vector and raster features destined for the OGCKML Writer.

### **Parameters**

#### ***Navigation Tree***

##### Name

Name specifies the text that will be displayed in the navigation tree of a KML Viewer such as Google Earth. If the feature has a point geometry, it will also be used as the label text.

##### Summary

Summary specifies the text that will be displayed below the feature's name in the navigation tree of a KML Viewer such as Google Earth.

##### Visible

Visibility specifies whether or not the feature will be visible by default when the user first opens the dataset. If the visibility is set to No, the feature will not be displayed. However, the visibility may still be toggled in the navigation tree of a KML Viewer such as Google Earth.

#### ***Balloon***

##### Content Type

Content Type specifies the type of the feature's balloon content. If the type is Text, the contents will be XML entity-encoded when written to the KML file. If the type is HTML, the contents will not be XML entity-encoded, but preserved in such a way that the content will be displayed as valid HTML.

##### Content

Content specifies the text or HTML description to be displayed in the feature's Balloon.

##### Include Attribute Table

Include Attribute Table specifies whether or not a table of the features attributes should be included in the Balloon Content.

#### ***Geometry***

##### Geometry Type

Geometry Type specifies the type of geometry properties to apply to the feature.

##### Altitude Mode

Altitude Mode specifies how KML Viewers, such as Google Earth, should interpret the z value of features with vector geometry, or the altitude setting for features with raster geometry.

##### Raster Altitude

Raster Altitude specifies the altitude, in meters, at which the raster tile should be displayed. This option is not commonly required.

##### Raster Opacity

Raster Opacity specifies the opacity of raster tile when displayed in a KML Viewer, such as Google Earth. This value is a display directive only, and has no effect on the underlying raster geometry.

##### Extrude

Extrude specifies whether or not to connect the geometry of vectors to the ground. KML Viewers, such as Google Earth, will not display extruded geometry unless the Altitude Mode is Absolute, Relative To Ground, or Relative to Sea Floor.

##### Follow Terrain

Follow Terrain specifies whether or not the feature's vector geometry should follow the terrain when displayed in Google Earth. KML Viewers,

such as Google Earth, will not enable terrain following unless the Altitude Mode is Clamp To Ground or Clamp To Sea Floor.

### **Additional References**

For more information regarding KML:

- See the FME OGCKML Writer chapter in the *FME Readers and Writers* manual (in Workbench, select Help > FME Readers and Writers Reference)
- See the documentation for KML: <http://code.google.com/apis/kml/documentation/>

### **Transformer Category**

KML

## **KMLRegionSetter**

Sets the region-related KML attributes for a group of features destined for the OGCKML Writer.

### **Parameters**

#### ***Bounding Box***

Calculate

This parameter specifies whether or not the bounding box of the region should be computed from the feature's geometry.

If the value is Yes, the bounding box will be computed; if necessary, the calculated bounding box will be reprojected to the LL84 coordinate system. A valid coordinate system is required for the bounding box to be calculated.

Minimum X, Y/Maximum X, Y

These parameters specify the coordinates, in the LL84 coordinate system, that define the boundary of the region's bounding box.

#### ***Display Criteria***

Minimum Display Size

Minimum Pixels specifies the minimum size, in pixels, in the user's Google Earth view, that the region's bounding box must occupy before the region will become active. A value in the range of 256 to 512 is recommended.

Minimum Display Size

Maximum Pixels specifies the maximum size, in pixels, in the user's Google Earth view, that the region's bounding box may occupy before the region will be deactivated. A value of -1 (infinite) is recommended.

Minimum Fade Extent

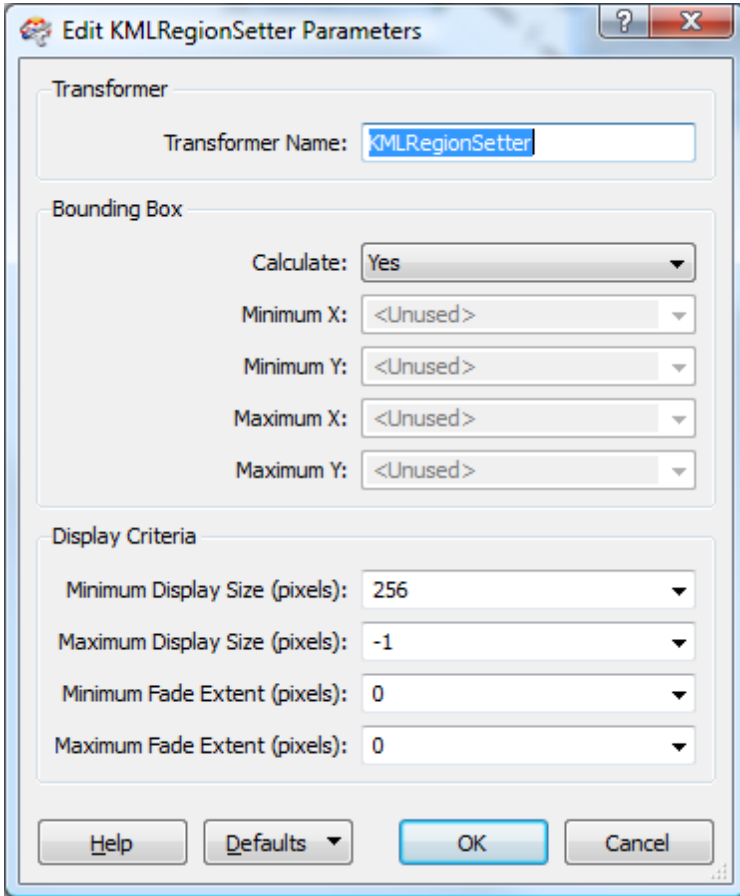
This parameter specifies an offset to Minimum Pixels, in pixels, that determines the rate at which the region will be activated, and its contents displayed. A value of 0 (immediate transition) is recommended.

Maximum Fade Extent

This parameter specifies an offset to Maximum Pixels, in pixels, that determines the rate at which the region will be deactivated, and its contents hidden. A value of 0 (immediate transition) is recommended.

### **Example**

The default settings will calculate a bounding box, and use that as the bounding box for the feature's region:



The inactive view shows that the feature is not initially displayed when you are zoomed out. The region is "inactive" and therefore the feature is not shown.



As you zoom in closer, the view meets the display criteria and the region activates; that is, the feature displays.



### Additional References

For more information about KML regions and how Level of Detail is calculated:



- See the FME OGCKML Writer chapter in the *FME Readers and Writers* manual (in Workbench, select Help > FME Readers and Writers Reference)
- See the KML documentation: <http://code.google.com/apis/kml/documentation/regions.html>

### **Transformer Category**

KML

## **KMLStyler**

Creates a common style for a group of features destined for the OGCKML writer.

### **Parameters**

#### Allow Unique Styles Per Feature

This parameter specifies whether the first feature through the transformer defines the common style that will be used for all features that pass through the transformer, or whether each feature can supply a unique style. Unique styles per feature create larger output files but allow for different settings, such as color, per feature.

### ***Color***

#### Color

This parameter specifies the pen color of the feature used to render the feature in the set color. The pen color determines the color of lines, and area boundaries. To edit this parameter, click the browse button to the right of the text field.

#### Fill Color

This parameter specifies the fill color for an area geometry on a feature used to render the interior feature in the set color. To edit this parameter, click the browse button to the right of the text field.

#### Opacity

Opacity specifies the opacity of the pen color of the feature. A value of 1.0 is fully opaque, and 0.0 is completely transparent.

#### Fill Opacity

Fill Opacity specifies the opacity of the fill color of the feature. A value of 1.0 is fully opaque, and 0.0 is completely transparent.

### ***Icon***

#### Name

Icon Name specifies either the name of an icon selected from the icon picker, or a filename/URL of a valid image file.

#### Scale

This parameter specifies the scale of the icon.

The scale is a multiplying factor, where 1.0 is unscaled. The lower scaling bound is 0.0, and there is no upper bound. If the value is exactly 0.0, Google Earth will not display the icon. The actual display size of the icon will vary according to how each Google Earth renders the dataset at various altitudes. This value has no units.

### ***Line Style***

#### Line Width

Line Width specifies the line width in pixels of line geometries and boundaries of area geometries.

### ***Label Style***

#### Scale

Label Style Scale specifies a unitless scaling factor that the KML Viewer applies to the default size of the label when rendered. Labels are rendered by KML Viewers, such as Google Earth, when the feature has both a KML name and a point geometry. The KML Writer will convert text features to KML points that have a name property. It is not possible to specify label size in terms of absolute units.

### **Example**

This screen capture shows a polygon that has had its color and fill color set by the KMLStyler.



### Additional References

For more information regarding KML styling:

- See the FME OGCKML Writer chapter in the *FME Readers and Writers* manual (in Workbench, select Help > FME Readers and Writers Reference)
- See the KML documentation: [http://code.google.com/apis/kml/documentation/kml\\_tut.html](http://code.google.com/apis/kml/documentation/kml_tut.html)

### Transformer Category

KML

## KMLTimeSetter

Sets the time-related KML attributes for a group of features destined for the OGCKML Writer.

### Parameters

#### Type

This parameter determines the type of KML time primitive that will be created. Possible options include Timestamp and Time Period.

#### Timestamp

This parameter specifies the value of the timestamp if the time type is Timestamp. The value can either be in the FME DateTime format, or the XML Schema time format.

#### Period Start

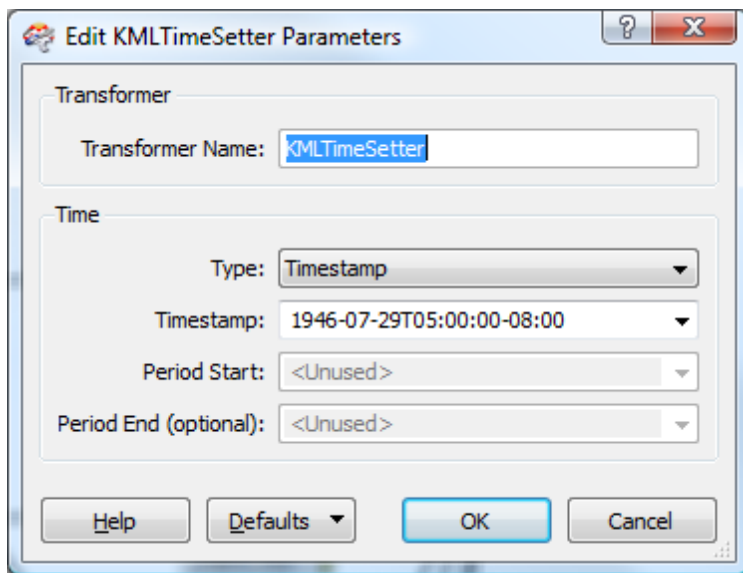
This parameter specifies the beginning of the time period if TimePeriod is the selected time type. The value can either be in the FME DateTime format, or the XML Schema time format.

#### Period End

This parameter specifies the optional end of the time period if TimePeriod is the selected time type. The value can either be in the FME DateTime format, or the XML Schema time format.

### Examples

#### *Timestamp Settings*



## Time Period Settings

Transformer

Transformer Name: KMLTimeSetter

Time

Type: Time Period

Timestamp: <Unused>

Period Start: 1946-07-29T05:00:00-08:00

Period End (optional): 1947-07-29T05:00:00-08:00

Help Defaults OK Cancel

### fmepedia

See fmepedia for additional information about FME DateTime formats.

### Additional References

For more information regarding KML time primitives:

- See the FME OGCKML Writer chapter in the *FME Readers and Writers* manual (in Workbench, select Help > FME Readers and Writers Reference)
- See the KML documentation: [http://code.google.com/apis/kml/documentation/kml\\_tut.html](http://code.google.com/apis/kml/documentation/kml_tut.html)

### Transformer Category

KML

## **KMLTourBuilder**

Generates a KML Tour from input features. The tour consists of tour stops that correspond to each input feature.

The location of the tour stop corresponds to the center point of the input feature. For features where the geometry is a line geometry, a series of tour stops will be generated, where each vertex in the line becomes a tour stop.

### **Parameters**

#### ***Tour***

##### **Tour Name**

The name of the Tour object that will be displayed in the navigation tree of a KML Viewer such as Google Earth. If the value of the Tour Name comes from an attribute, all features sharing a common attribute value will be grouped into the same tour. It is possible for the KMLTourBuilder to build several tours if there are different attribute values.

##### **Tour Duration**

The time, in seconds, for the entire tour. The duration of a tour has two components: flying time and waiting time (if wait delays are used). The flying time is calculated as the specified tour duration less the total wait times. The flying time between each tour stop will be calculated such that the velocity of the tour is approximately constant.

#### ***Transition***

##### **Type**

Smooth: The flying velocity will remain constant for the entire tour.

Bounce: The flying velocity will be gradually reduced to zero as the tour reaches each stop.

#### ***Balloon***

##### **Display**

This parameter specifies whether the feature's balloon should be popped up as the tour reaches the tour stop, and then hidden when the tour departs the tour stop.

The balloon will only display when the flight velocity has reached zero, so this option is best used with either a Wait or Pause delay.

#### ***Delay***

##### **Type**

Specifies the type of delay at each tour stop.

If the Type is set to Wait, the tour will briefly stop at each tour stop for the number of seconds specified in the delay duration. If the Type is set to Pause, the tour will stop at each tour stop until the user presses the play button on the tour control of a KML Viewer such as Google Earth.

##### **Duration**

The number of seconds to stop if the Type parameter is set to Wait.

#### ***View***

##### **Perspective**

Specifies the user's perspective for the entire tour:

- Third Person: the view will be from a point that orbits the tour stop.
- First Person: the view will be from the exact location of the tour stop.

##### **Range**

Specifies the distance, in meters, from the view point to the tour stop location in the Third Person perspective.

If the value is <calculate>, the range value will be calculated such that user's view includes the tour stop, as well as a portion of the remaining tour. The calculated range is constant for every tour stop.

#### Heading

Specifies the direction (azimuth) of the view, in degrees, relative to North. If the value is <calculate>, the heading value will be calculated such the heading for the current tour stop is in the direction of the next tour stop.

#### Tilt

Specifies the rotation, in degrees, of the view around the X axis. A value of 0 indicates that the view is aimed straight down, and a value of 90 indicates that the view is aimed toward the horizon. Values greater than 90 only apply if the View Perspective is First Person, and indicate that the view is pointed towards the sky. If the value is <calculate>, the tilt value will be calculated such the tilt for the current stop is in the direction of the next stop.

#### **Additional References**

For more information regarding KML Touring and View parameters, see the documentation for KML Tours:

<http://code.google.com/apis/kml/documentation/touring.html>

## **KMLViewSetter**

Sets the view-related KML attributes for a group of features destined for the OGCKML Writer. Creation of LookAt or Camera views are supported.

### **Parameters**

#### ***Location***

Longitude, Latitude, Altitude

These parameters specify a location associated with the View Type parameter.

Altitude Mode

Specifies how KML Viewers, such as Google Earth, should interpret the altitude value.

#### ***View***

View Type

If the View Type is LookAt, this location is the focal point of the view (where the view is looking at). In most cases, this is the view type you will choose. If you determine, for whatever reason, that it is too limiting, you can set the view type to Camera.

If the View Type is Camera, this location is the location of the camera (where the view is looking from).

Heading

Heading specifies the rotation, in degrees, of the view around the Z axis, relative to North.

Tilt

Tilt specifies the rotation, in degrees, of the view around the X axis. A value of 0 indicates that the view is aimed straight down, and a value of 90 indicates that the view is aimed toward the horizon.

Values greater than 90 only apply if the View Type is Camera, and indicate that the view is pointed towards the sky.

Roll

Roll specifies the rotation, in degrees, of the view around the Z axis, after the Heading and Tilt rotations have been applied. This value only applies to Camera views, and is generally not necessary.

Range

Range specifies the distance, in meters, that the view location is viewed from. This value only applies to LookAt view types.

### **Example**

These transformer parameters are configured for a LookAt view type.



**Edit KMLViewSetter Parameters**

Transformer

Transformer Name:

Location

Longitude:

Latitude:

Altitude Mode:

Altitude:

View

View Type:

Heading:

Tilt:

Roll:

Range:

This is the resulting view of the selected area.



## **Additional References**

For more information regarding KML:

- See the FME OGCKML Writer chapter in the *FME Readers and Writers* manual (in Workbench, select Help > FME Readers and Writers Reference)
- See the KML documentation: <http://code.google.com/apis/kml/documentation/cameras.html>

## Labeller

Interpolates labels along a linear or polygonal feature. Labels are placed at a regular perpendicular distance from the closest point on the line at some interval, and are given a rotation perpendicular to the line.

### Parameters

#### Label Attribute

The label text is taken from the attribute named in the Label Attribute parameter.

#### Prevent Label Overlaps

If this parameter is set to Yes, labels will only be generated if they don't overlap with other labels already generated by the same transformer. If it is set to No, overlapping labels will be generated.

#### Label Position

The Label Position controls where the label will be placed relative to the original line.

- If Above or Left is specified, then the points are located the <offset> distance above (or, if the line is vertical, to the left of) the line. Above or Right is identical, except if the line is vertical then label point will then be to the right of the linear feature.
- Below or Left and Below or Right work similarly with all label points being below the linear feature unless it is vertical. If it is vertical, then the label points are to the left and right, respectively.
- If Right is specified, then the label is always placed on the right side of the line.
- If Left is specified, then the label is always placed on the left side of the line.

#### Label Offset

The labels will be placed Label Offset units perpendicular to the closest point on the line, where the offset is also measured in ground units. You can either enter a number, or take the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Labels at End Points

This parameter controls whether or not labels will be placed at the end points of the line. If not, then the labels will start half the label spacing from the start of the line.

#### Label Height

This parameter controls the height of the labels, measured in ground units. You can either enter a number, or take the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Average Character Width

This parameter controls the average character width. If left as the default, 0.0, the Label Height will also be used as the average character width.

#### Label Spacing

This parameter controls the interval of the labels. A label spacing of 0 will result in a single label being placed at the midpoint of the line. You can either enter a number, or take the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Minimum Length

This parameter sets a length threshold. If a line does not meet the minimum length requirement, no labels will be generated for it.

#### Label Rotation Attribute

This is the name of the attribute which will hold the rotation of the label. This rotation is adjusted from the orientation of the line so that text oriented at the label point will be parallel to the line segment and will not be upside down or right-to-left.

All rotations are measured in degrees counterclockwise from horizontal.

#### Parallel Rotation Attribute

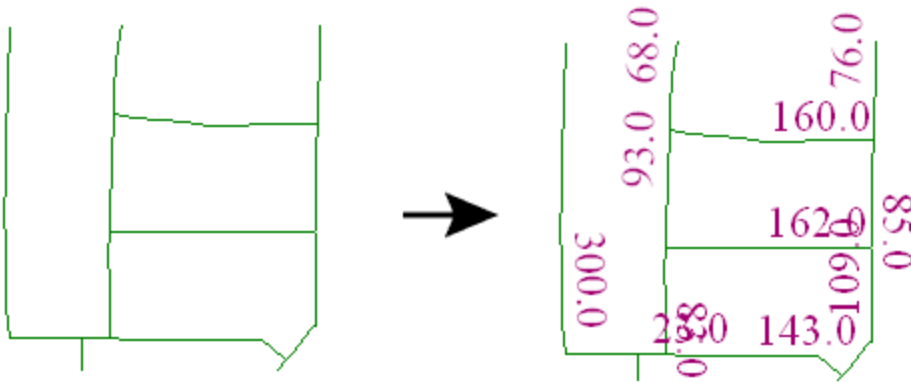
This is the name of the attribute which will hold the rotation of the line itself at the point being labeled.

All rotations are measured in degrees counterclockwise from horizontal.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses are accepted as linear and polygonal features, thus producing label point features; otherwise, they are considered non-linear, and no label point features will be produced.

### Example



### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: LabelFactory

## LabelPointReplacer

Replaces the geometry of the feature with a label point.

The text is guaranteed to be inside (in case of polygons) or on (lines and points) the original object.

### Input

Input features must have the correct geometry: point, line or area.

### Results

- If the feature was already a point, it is simply turned into a text feature at the original location.
- If the feature was a line, it is turned into a point text feature at the midpoint of the line, with a rotation parallel to the line.
- If the feature was an area feature, the resulting point is somewhere in the interior of the feature (and outside of any holes in the area).

### Parameters

Label Attribute

The name of the attribute that holds the geometry.

Label Height

The label height is measured in ground units, and may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are accepted as linear features, thus producing label point features located halfway along the arcs. Otherwise each arc will be accepted as a point located at the arc's center point and the resulting label point will be at that location.

### Example



### Transformer Category

Manipulators

### Related Transformers

To generate a point that will be used as a centroid for polygons, use an InsidePointReplacer. This saves you the overhead of having a text object.

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @GeneratePoint, PIPComponentsFactory

## **LatLongToMGRSConverter**

Calculates a Military Grid Reference System (MGRS) code based on the latitude and longitude values supplied in a feature's attributes.

The granularity of the resulting code is determined by the specified Precision which is an integer from 0 to 5 (inclusive). A precision of 5 locates a point within a 1-meter square and precision of 0 locates a point within 100-km square.

The result of the conversion is stored in the MGRS Code Attribute of the feature.

### **Parameters**

#### Ellipsoid

The ellipsoid used for the conversion. This can be any ellipsoid name supported by FME.

#### Lettering Type

The type of lettering used can be WGS84 or Bessel.

#### Precision

The granularity of the resulting code is determined by the specified Precision which is an integer from 0 to 5 (inclusive). A precision of 5 locates a point within 1-meter square and a precision of 0 locates a point within 100-km square.

#### Longitude, Latitude

You can choose these parameters from the attributes in the pull-down list, or enter the longitude (-90, 90) and latitude (-180, 180) values.

#### MGRS Code

An MGRS code used to convert to lat/long coordinates. You can choose this parameter from the list of attributes in the pull-down list, or enter an MGRS string.

### **Transformer Category**

Coordinate Systems

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @MGRS

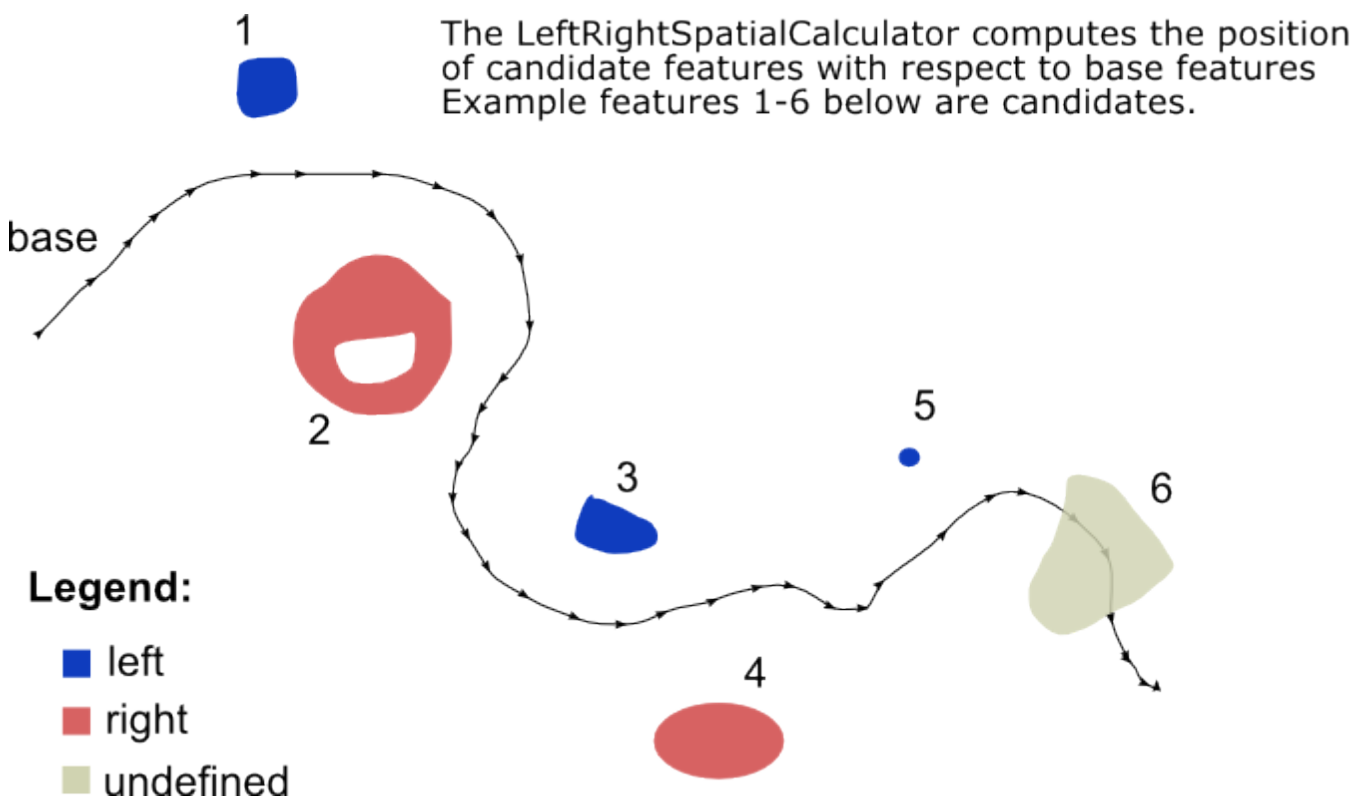
## LeftRightSpatialCalculator

Computes relative position of CANDIDATE input features relative to BASE input features.

The geometry of a CANDIDATE feature is restricted to point and area, while BASE features can only be lines.

See the example below. The BASE lines are considered to be oriented from the start point to the end point (digitization sense). A CANDIDATE feature is:

- located to the LEFT of a BASE feature if looking down the BASE line in the direction of the orientation (from start to finish);
- located to the RIGHT if the CANDIDATE is on the right of a BASE feature;
- UNDEFINED if the BASE and CANDIDATE features intersect, or if the BASE feature has a single point.



### Output Port

Before being sent out through the OUTPUT port, each CANDIDATE feature will have a list of its relative positions to each of the BASE features attached to it. Each element of the list has two pieces of information:

- a "base\_id" which identifies the base
- the computed relative "position"

BASE features are dropped after being used.

### Parameters

Use Candidate Center of Mass

If you choose Yes (default), the algorithm will use only the center of mass of passed in CANDIDATEs instead of the whole geometry.



### Use Base Closest Point

If you choose Yes, the algorithm will compute only at the point on the BASE line that is closest to the first or "center of mass" point.

### Group By

If you choose Group By attributes, CANDIDATEs are only compared against BASEs that have the same values in these attributes.

### Base Type

Define whether only a Single Base will be supplied, or whether Multiple Bases will be supplied. If you choose Bases First, all BASE features will enter the transformer first, before any CANDIDATE features.

### Base ID Attribute (Required)

Choose the BASE attribute whose value will be used to identify it in the list of relative positions attached to each output CANDIDATE.

### Relative Position List Attribute

Specify the name of the list attribute that will be added onto all output CANDIDATE features, and that will contain the relative positions.

### **Transformer Category**

Calculators

### **Technical History**

FME Factory Used: SpatialFilterFactory

## **LengthCalculator**

Calculates the length of a feature and adds it as a new attribute. The multiplier parameter can be used to scale the length from being measured in ground units (the units of the feature's coordinates) to something else.

By default, the two-dimensional length of the line is calculated, and any Z values, if present, are ignored. To have a 3D length calculated, the length dimension parameter should be set to 3, and then the elevations will be included in the length computation.

Using this transformer on a polygon will produce the polygon's perimeter.

To calculate the partial length of a feature, use the [LengthToPointCalculator](#).

To calculate the distance from the start of the line to each vertex, use the [MeasureGenerator](#).

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Length

## **LengthToPointCalculator**

Category: Linear Referencing

Calculates the length of a feature from its start until the closest spot to a point, and adds it as a new attribute. The point coordinates are taken from attributes in the original feature.

The length is calculated along the feature up until the spot on the feature that is closest to the given point.

By default, the 2-dimensional length of the line is calculated, and any Z values, if present, are ignored. To have a 3D length calculated, the length dimension parameter should be set to 3, and then the elevations will be included in the length computation.

To calculate the length of the entire feature, use the [LengthCalculator](#).

To calculate the distance from the start of the line to each vertex, use the [MeasureGenerator](#).

### **Technical History**

Associated FME function or factory: @Length

## LicenseChecker

Checks whether the license file is valid and the specified product name is licensed, based on a vendor key and vendor registration code.

### Output Ports

- PASSED: If the license file is successfully validated and the specified product name is licensed, the feature is output via the PASSED port.
- FAILED: If the license file fails to be validated or the specified product name is not licensed, the feature is output via the FAILED port.

### Parameters

#### Vendor Key

This is the custom key assigned by the vendor. This key is used together with the Vendor Registration code to generate the license file specified below.

#### Vendor Registration

The unique registration code of this vendor assigned by Safe Software Inc.

#### License File Name

The name of the license file to be validated. The license file is assumed to be located at `$FME_HOME/licenses` folder

#### Product Name

The name of the product to be checked by this transformer. If not specified, the license file will be checked for validation only.

### Example

The LicenseChecker scenario outlines the steps required to license (author/publisher) and download (end-user) a custom transformer or workspace.

### Transformer Category

Filters

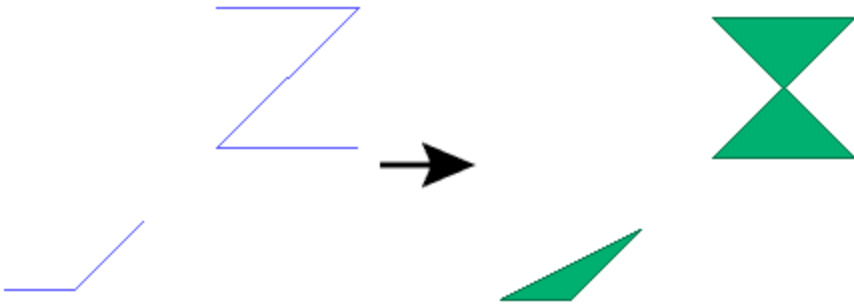
## LineCloser

Category: Manipulators

Turns input linear features into areas by adding their start point as the end point.

The result may self-intersect, so this transformer should be used with care.

### Example



### Technical History

Associated FME function or factory: @Close

## LineJoiner

Takes lines and connects them to form longer lines. Each connecting line must meet at the exact same start/end point, but otherwise they must not intersect.

Any nodes with only two lines connecting to them (sometimes called *pseudonodes*) are removed. Lines remain broken at points where three or more converge.

### Output Ports

- LINE: All linear features produced by this transformer are sent to the OUTPUT port.
- INVALID: Features with invalid geometries are sent to the INVALID port.

### Parameters

#### Group By

The Group By list of attributes separates the input data into sets which are considered independently when the node removal is done. Only attributes that are listed in the Group By list will be passed to subsequent transformers – all other attributes are removed.

#### Break Across Groups

Yes: The transformer will consider all nodes from all groups of features when deciding on topologically significant points.

No: Each group will be considered separately.

#### Preserve Original Orientation

This parameter controls whether or not lines can be reversed in order to create longer lines. If the direction of the lines is significant, then choose Yes.

#### Break Loops

This parameter indicates whether any resulting (or input) closed rings should ever be broken into two segments:

- If you choose Yes and the input data contained a set of lines that formed a ring and did not interact with any other lines, no closed ring will be formed and the two lines will be returned, broken at two arbitrary end vertices of the original input.
- If you choose No, then if a set of lines independent from any other set of lines would form a ring, they will be connected into a single linear feature whose start and end point is the same. The feature will be considered as having a geometry type of line and not polygon (however, you could use the GeometryCoercer to promote its geometry type to polygon).

#### Consider Node Elevation

This parameter specifies whether to consider the node elevation when joining lines. Nodes that share the same location but have different elevations will not be joined if this option is set to Yes.

#### Input Feature Topology

Input Feature Topology indicates the type of vertices the input features contain:

- End noded means only the start and end points will be considered for joining lines, and other vertices will be ignored.
- Vertex noded means all vertices will be considered when joining lines.

#### List Name

If a List Name is given, an attribute list is created on each output with an entry for each feature contributing to that output line, and attributes are merged as base attributes on the output features. Otherwise, no attributes besides the Group By attributes are added to the output features.

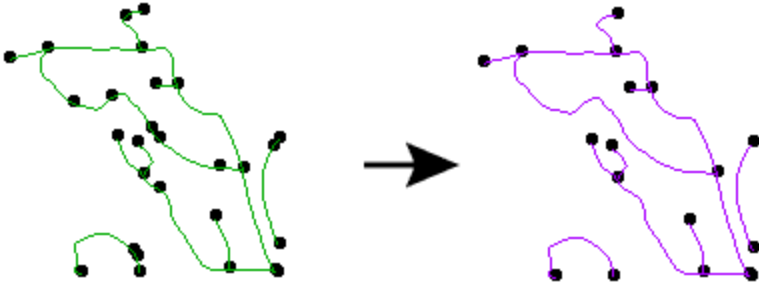
### Usage Notes

If the data might contain intersecting lines, you should run it through the Intersector before connecting this transformer.

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are preserved throughout the computation of the joined lines; otherwise, they are stroked prior to computation.

### Example



### Related Transformers

Intersector

### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: ArcFactory

## **LineMeasureExtractor**

Category: Linear Referencing

Creates a list attribute with the name given by Destination List Attribute, with its values set to those in the measure named by Source Measure Name. If no Source Measure Name is given, the default measures will be used.

### **Technical History**

Associated FME function or factory: @Geometry



## **LineMeasureSetter**

Category: Linear Referencing

Sets the measures on a line geometry to the values in the given Source Measure List Attribute. If no Destination Measure Name is specified, the default measures will be set.

### **Technical History**

Associated FME function or factory: @Geometry

## LineOnAreaOverlayer

Performs a line-on-area overlay. Each input line is split at any area boundaries it intersects.

### Parameters

#### Group By

The Group By parameter specifies a series of attributes that must match on the features before they are considered for overlaying. This can be used to ensure that the overlay occurs only on certain groups of features. Click the browse button to select from a list of attributes.

#### Overlap Count Attribute

The Overlap Count Attribute added to output *linear* features holds the number of area features that they were inside of. The Overlap Count Attribute added to output *area* features holds the number of linear features that they contained.

#### List Name

If the optional List Name is supplied, the attributes of each area containing an output line are added to that line's list, and the attributes of each line contained by an output area are added to that area's list. Note that no intersections between area features are computed.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are accepted as lines and ellipses accepted as areas preserved as ellipses; otherwise, ellipses are stroked.

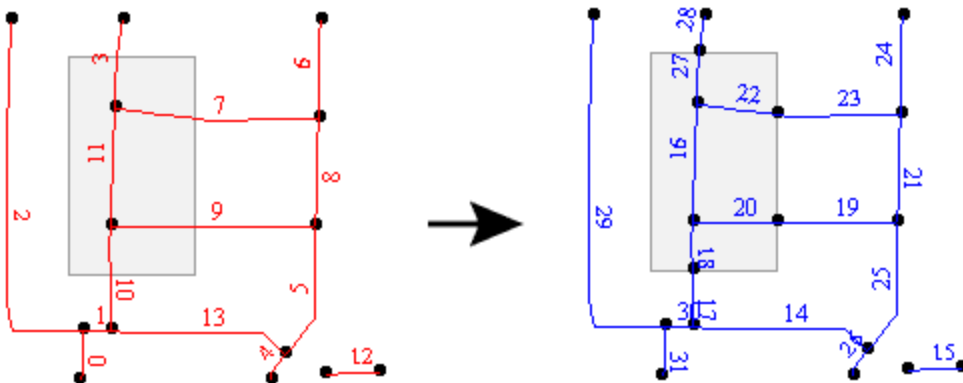
If the Advanced setting Geometry Handling is set to Classic in the workspace, the input arc features will be ignored in the output.

### Usage Notes

Each resulting piece receives the attributes of the area(s) it is contained in, and each containing area receives the attributes of the line. When attributes are merged between features, existing attributes are not replaced. Therefore, if the lines and areas being overlaid have attributes with the same name, then the values will not be transferred from one to the other.

You can avoid this problem by renaming (AttributeRenamer), prefixing (AttributePrefixer), or removing (AttributeRemover) attributes to avoid name collisions.

### Example



### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: OverlayFactory

## LineOnLineOverlayer

This transformer performs a line-on-line overlay. During the overlay, all input lines are intersected against each other and resultant line features are created and output. Intersection points are turned into point features that contain the merged list of attributes of the original intersected lines.

### Parameters

#### Group By

The Group By parameter specifies a series of attributes that must match on the features before they are considered for overlaying. This can be used to ensure that the overlay occurs only on certain groups of features. Click the browse button to select from a list of attributes.

#### Overlap Count Attribute

The Overlap Count attribute holds the number of line features that intersected the resultant feature.

#### List Name

If the optional List Name is supplied, then all output *POINTS* will include a list containing attributes of each line that intersected at that point. If the List Name is supplied AND Separate Collinear Segments is set to No, then all output *LINES* will include a list containing attributes of each collinear line that was merged. If the optional List Name is supplied, AND Separate Collinear Segments is set to Yes, then no merging will occur and so output *LINES* will not include a list.

#### Separate Collinear Segments

No: All collinear segments are reduced to one representative segment.

Yes: All collinear segments are output.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are accepted as lines; otherwise, arcs are not accepted lines.

### Usage Notes

- When attributes are merged between features, existing attributes are not replaced. Therefore if the two lines being overlaid contain attributes with the same name, then the values will not be transferred from one to the other. You can avoid this problem by renaming (AttributeRenamer), prefixing (AttributePrefixer), or removing (AttributeRemover) attributes to avoid name collisions.
- When choosing to output only one segment for each collinear group, it may be difficult to predict which of the "original" geometries and attributes are preserved. In general, the features that arrive at the transformer FIRST will be the ones that form the separate segments. Use the FeatureHolder or Sorter transformers to change the sequence of features if this is important.
- This transformer will preserve measures, based on the rules for the order of features described above.

### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: OverlayFactory

## ListBuilder

Combines attributes of the input features into a single list structure.

### Output Ports

One feature is output for each unique combination of values of the attributes specified in the Group By parameter. Features output from this transformer have no geometry.

### Parameters

#### Group By

This parameter allows you to specify the attribute values on which to join the series of input features. One feature is output for each group created by selecting a Group By attribute. If no Group By attributes are specified, then a single feature is output and the number of elements in its list is equivalent to the number of features input into the ListBuilder.

#### List Name

The features output from the ListBuilder have all of their attributes stored in the list identified by the name specified by this parameter.

### Example

Suppose this transformer is used with no group-by attributes, and three features enter it with these attributes:

Feature 0:	length = 7.3 kind = 'paved'
Feature 1:	length = 8.4 kind = 'smooth' lanes = 2
Feature 2:	length = 1.1 kind = 'rough'

then, presuming that the list name specified was *somelist*, a single feature with these attributes is output:

```
somelist{0}.length = 7.3  
somelist{0}.kind = 'paved'  
somelist{1}.length = 8.4  
somelist{1}.kind = 'smooth'  
somelist{1}.lanes = 2  
somelist{2}.length = 1.1  
somelist{2}.kind = 'rough'
```

### Transformer Category

Lists

### Technical History

Associated FME function or factory: ListFactory

## ListConcatenator

Concatenates all the values of a list into a single attribute.

### Parameters

List Attribute

Select the attribute that will supply the list containing the values you want to concatenate.

Separator Character

The separator character can be expressed either as a single character, or as a special character sequence beginning with a backslash ("\"), as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering *lan\es* will be interpreted as *lanes*.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

Destination Attribute

Allows you to specify the attribute that will contain the concatenated list values.

### Example

If this feature enters the ListConcatenator:

- `somelist{0}.length` = 7.3
- `somelist{0}.kind` = 'paved'
- `somelist{1}.length` = 8.4
- `somelist{1}.kind` = 'smooth'
- `somelist{1}.lanes` = 2
- `somelist{2}.length` = 1.1
- `somelist{2}.kind` = 'rough'

and a concatenation of `somelist{ }.kind` with a backslash (\\) separator is requested, then the resulting attribute will contain: `paved\smooth\rough`.

### Transformer Category

Lists

### Related Topics

Using the Intersector and ListConcatenator to Solve Problems

About Attribute Lists

## **Technical History**

Associated FME function or factory: @Tcl2

## ListCopier

Copies a complete attribute list, including all nested attributes, from one list name to another.

### Parameters

#### Source List Attribute

The name of the list attribute that will supply the values for the created list attribute.

#### Destination List Name

The name of a list attribute that will be created in the feature.

### Example

If this feature enters the ListCopier:

```
somelist{0}.length = 7.3  
somelist{0}.kind   = 'paved'  
somelist{1}.length = 8.4  
somelist{1}.kind   = 'smooth'  
somelist{1}.lanes  = 2  
somelist{2}.length = 1.1  
somelist{2}.kind   = 'rough'
```

and the source list name is set to somelist and the destination list name is newlist, the feature leaving the transformer will have these attributes:

```
somelist{0}.length = 7.3  
somelist{0}.kind   = 'paved'  
somelist{1}.length = 8.4  
somelist{1}.kind   = 'smooth'  
somelist{1}.lanes  = 2  
somelist{2}.length = 1.1  
somelist{2}.kind   = 'rough'  
newlist{0}.length  = 7.3  
newlist{0}.kind    = 'paved'  
newlist{1}.length  = 8.4  
newlist{1}.kind    = 'smooth'  
newlist{1}.lanes   = 2  
newlist{2}.length  = 1.1  
newlist{2}.kind    = 'rough'
```

### Transformer Category

Lists

### Technical History

Associated FME function or factory: @CopyAttributes

## ListDuplicateRemover

Removes all duplicate values from a list attribute. In the resulting list, only distinct values for the list attribute will be present.

### Parameters

List Attribute

Specifies the list attribute from which to remove duplicates.

### Usage Notes

The input list must be a valid one, beginning at index 0, and counting up without any gaps.

### Example

If this feature enters this transformer:

```
somelist{0}.kind = 'paved'  
somelist{1}.kind = 'smooth'  
somelist{2}.kind = 'smooth'  
somelist{3}.kind = 'rough'  
somelist{4}.kind = 'smooth'  
somelist{5}.kind = 'smooth'  
somelist{6}.kind = 'paved'
```

and the list name is set to "somelist{ }.kind", the feature leaving the transformer will have these attributes:

```
somelist{0}.kind = 'paved'  
somelist{1}.kind = 'smooth'  
somelist{2}.kind = 'rough'
```

Note that if there were other attributes in the list "parallel" to the attribute being operated on, these will also be moved in the list to stay in "parallel" with the key attribute.

For example, if the input feature had these attributes:

```
somelist{0}.kind = 'paved'  
somelist{0}.id = 'A3'  
somelist{1}.kind = 'smooth'  
somelist{1}.id = 'B7'  
somelist{2}.kind = 'smooth'  
somelist{2}.id = 'B8'  
somelist{3}.kind = 'rough'  
somelist{3}.id = 'C9'  
somelist{4}.kind = 'smooth'  
somelist{4}.id = 'B9'  
somelist{5}.kind = 'smooth'  
somelist{5}.id = 'B2'  
somelist{6}.kind = 'paved'  
somelist{6}.id = 'A7'
```

and the list name is set to somelist{ }.kind, the feature leaving the transformer will have these attributes:

```
somelist{0}.kind = 'paved'  
somelist{0}.id = 'A3'  
somelist{1}.kind = 'smooth'  
somelist{1}.id = 'B7'  
somelist{2}.kind = 'rough'
```



somelist{2}.id = 'C9'

### **Transformer Category**

Lists

### **Technical History**

Associated FME function or factory: @TCL

## ListElementCounter

Stores the number of member elements found in the specified list into an attribute.

### Parameters

List Attribute

The name of the list attribute whose elements are to be counted.

Element Count Attribute

The attribute that will contain the number returned by this transformer.

### Example

If this feature enters the ListElementCounter:

```
somelist{0}.length      = 7.3  
somelist{0}.kind       = 'paved'  
somelist{1}.length     = 8.4  
somelist{1}.kind       = 'smooth'  
somelist{1}.lanes      = 2  
somelist{2}.length     = 1.1  
somelist{2}.kind       = 'rough'
```

and a count of somelist{} is requested, 3 will be returned.

### Transformer Category

Lists

### Additional Resources

[More Information on Lists](#)

### Technical History

Associated FME function or factory: @NumElements

## ListExploder

Explodes each list member on each input feature out into its own feature.

Any attributes on the list are demoted to become non-list attributes of the feature output. In addition, the element number of the attributes in the original list is added to the feature. Each feature's output has a copy of the geometry from the original input feature.

### Output Ports

- LIST\_FOUND: If the list is found in the input feature, it will be processed based on Attributes To Keep and output via this port.
- NOT\_FOUND: If the list is not found in the input feature, the feature will be output via this port without being processed. In this case, the Attributes To Keep will not be applied and the transformer just passes the feature through without processing it.

### Parameters

List Attribute

The name of the list that will be exploded by this transformer.

Element Index Attribute

If the Element Index Attribute is specified, then each element feature output will be given an attribute containing the element's list position.

Attributes to Keep

Determines if the output features contain just the list element attributes, or the original input attributes as well.

**All Attributes:** The list elements that are produced in this mode will retain all the attributes of the original BASE feature. In addition, all of the attributes of each particular element in the list will be added without the list prefix.

**List Elements and Original Attributes:** The list elements that are produced in this mode will retain all the attributes of the original BASE feature except for the list attributes referred to in the LIST\_NAME clause. In addition, all of the attributes of each particular element in the list will be added without the list prefix.

**List Element Attributes:** The list elements that are produced in this mode will not retain any of the attributes of the original BASE. They will only add the attributes of each particular element in the list, without the list prefix. This mode has a significant impact on performance when the lists have large numbers of elements.

### Tip:

Attributes to Keep determines which attributes remain on the exploded features – the entire set of attributes from the source feature, or just the attributes in the List?

When you choose just the list attributes ("List Element Attributes") then you need to be aware that this will remove all existing attributes, *including format attributes*. This means your features won't have an fme\_type, and this can cause them to be written incorrectly to a destination (for example, as non-geom features).

What might be even more confusing is that you won't see the problem if you connect the ListExploder to an Inspector: the features will appear to have the correct fme\_type and fme\_geometry. This is because when features are sent to the Inspector, FME converts them to FFS format first, which recreates these attributes. The problem would only be apparent if you sent the features to a Logger transformer, where you would see that fme\_type is missing.

The solution to this problem is to either keep "All Attributes" or to manually set fme\_type after the ListExploder.

### Usage Notes

- If BASE attributes and list element attributes have the same name, the BASE attributes will be overwritten by the list element attributes.
- ListExploder will not work with an invalid list, such as one that doesn't start at entry 0. For example:

Valid List	Invalid List
mylist{0}.attr1	mylist{1}.attr1
mylist{1}.attr1	mylist{2}.attr1
mylist{2}.attr1	mylist{3}.attr1
mylist{3}.attr1	mylist{4}.attr1

## Examples

### Example 1:

If this feature enters this transformer:

```
somelist{0}.length = 7.3  
somelist{0}.kind = 'paved'  
somelist{1}.length = 8.4  
somelist{1}.kind = 'smooth'  
somelist{1}.lanes = 2  
somelist{2}.length = 1.1  
somelist{2}.kind = 'rough'  
another_attr = 'something else'
```

then, presuming that the list name specified was "somelist", the index attribute was 'element\_num', and the **Attributes to Keep** parameter was set to **All Attributes**, three features will be output:

Feature 0: length = 7.3

kind = 'paved'

element\_num = 0

somelist{0}.length = 7.3

somelist{0}.kind = 'paved'

somelist{1}.length = 8.4

somelist{1}.kind = 'smooth'

somelist{1}.lanes = 2

somelist{2}.length = 1.1

somelist{2}.kind = 'rough'

another\_attr = 'something else'

Feature 1: length = 8.4

kind = 'smooth'

lanes = 2

element\_num = 1

somelist{0}.length = 7.3

somelist{0}.kind = 'paved'

somelist{1}.length = 8.4

somelist{1}.kind = 'smooth'

somelist{1}.lanes = 2

somelist{2}.length = 1.1

somelist{2}.kind = 'rough'

another\_attr = 'something else'

Feature 2: length = 1.1

```
kind = 'rough'
element_num = 2
somelist{0}.length = 7.3
somelist{0}.length kind = 7.3'paved'
somelist{01}.kind length = 'paved'8.4
somelist{1}.length kind = 8.4'smooth'
somelist{1}.kind lanes = 'smooth'2
somelist{12}.lanes length = 21.1
somelist{2}.length kind = 1.1'rough'
somelist{2}.kind another_attr = 'roughsomething else'
```

If the **Attributes to Keep** parameter was set to **List Elements and Original Attributes**, the first returned feature would look like:

```
Feature 0: length = 7.3
kind = 'paved'
element_num = 0
another_attr = 'something else'
```

If the **Attributes to Keep** parameter was set to **List Element Attributes**, the first returned feature would look like:

```
Feature 0: length = 7.3
kind = 'paved'
element_num = 0
```

### **Example 2:**

Suppose you have a BASE feature with the following 6 attributes. (The LIST\_NAME used here would be "ID{ }".)

```
Name
Type
ID{0}.dec
ID{0}.hex
ID{1}.dec
ID{1}.hex
```

**All Attributes** will produce elements with 8 attributes:

```
Name
Type
dec
hex
ID{0}.dec
ID{0}.hex
ID{1}.dec
ID{1}.hex
```

**List Elements and Original Attributes** will produce elements with 4 attributes:

Name

Type

dec

hex

**List Element Attributes** will produce elements with 2 attributes:

dec

hex

### **Transformer Category**

Lists

### **fmepedia**

See a workspace example on fmepedia.

### **Technical History**

FME function or factory used: ElementFactory

### **Additional Resources**

- [More Information on Lists](#)

## ListHistogrammer

Computes a histogram of the values found in a list, and returns these in a new list attribute on the feature.

### Parameters

Source List Attribute

The source list from which the histogram will be computed.

Histogram List Name

The name of the new list attribute that will contain the output histogram. The new list will be sorted so that the value with the most occurrences will be first.

### Usage Notes

If the source list did not have elements, a new list will not be created and this transformer will have made no changes to the feature.

### Example

If a feature with this list enters this transformer:

```
somelist{0}.val = 'apple'  
somelist{1}.val = 'donut'  
somelist{2}.val = 'aardvark'  
somelist{3}.val = 'apple'  
somelist{4}.val = 'aardvark'  
somelist{5}.val = 'apple'
```

the resulting feature will have this list attribute:

```
_histogram{0}.value = 'apple'  
_histogram{0}.count = 3  
_histogram{1}.value = 'aardvark'  
_histogram{1}.count = 2  
_histogram{2}.value = 'donut'  
_histogram{2}.count = 1
```

### Transformer Category

Lists

### fmepedia

See [fmepedia](#) for an example use of this transformer.

### Technical History

Associated FME function or factory: @TCL

## ListIndexer

Demotes the attributes of the list element specified by the index to become main attributes of the feature.

### Parameters

#### List Attribute

Specifies the name of the list attribute the transformer will read from.

#### List Index

The index of the list element whose contents will be copied to the main attribute(s) of the output feature.

### Example

If this feature enters this transformer:

```
somelist{0}.length      = 7.3
somelist{0}.kind        = 'paved'
somelist{1}.length      = 8.4
somelist{1}.kind        = 'smooth'
somelist{1}.lanes       = 2
somelist{2}.length      = 1.1
somelist{2}.kind        = 'rough'
```

and the index is set to 2, then the feature leaving the transformer will have these attributes:

```
length      = 1.1
kind        = 'rough'
somelist{0}.length      = 7.3
somelist{0}.kind        = 'paved'
somelist{1}.length      = 8.4
somelist{1}.kind        = 'smooth'
somelist{1}.lanes       = 2
somelist{2}.length      = 1.1
somelist{2}.kind        = 'rough'
```

The index may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Transformer Category

Lists

### Technical History

FME function or factory used: @CopyAttributes

### Additional Resources

More Information on Lists



## ListKeeper

Keeps the selected list attributes from incoming features, and removes the rest.

### Parameters

Lists to Keep

After you connect this transformer, click the Browse button and select the lists to keep.

Note that if you select to keep a list, your selection will include any list attributes or nested lists. For example, if you select to keep a list called

```
list{}
```

then

```
list{}.attr or list{}.sublist{}
```

will also be kept.

### Transformer Category

Lists

### Technical History

Associated FME function or factory: @KeepAttributes

## ListPopulator

Populates a new list from a series of attributes. The attributes to be used are specified by the prefix parameter. Each attribute's index in the list is specified by everything in the attribute name after the prefix.

The **List Name** parameter specifies the name for the new list.

The geometry of each feature entering the transformer is not changed.

For example, if the prefix parameter is set to "bob," and a feature entered the transformer with the following attributes:

```
bob0 = 'cat'  
bob1 = 'dog'  
bob2 = 'rooster'  
anotherAttr = 'carrot'
```

then, presuming that the list name specified was "somelist," the feature would now look like:

```
bob0 = 'cat'  
bob1 = 'dog'  
bob2 = 'rooster'  
anotherAttr = 'carrot'  
somelist{0} = 'cat'  
somelist{1} = 'dog'  
somelist{2} = 'rooster'
```

Note that in order for the resulting list to be useful, the source attributes must begin at index 0, and count up without any gaps. For instance, if in the example above, only the **bob2** attribute was available, the resulting list would not be valid.

## Additional Resources

[More Information on Lists](#)

## Transformer Category

Lists

## Technical History

FME function or factory used: @TCL

## ListRangeExtractor

Category: Lists

Extracts the minimum and maximum values found in a list.

For example, if this feature enters this transformer:

```
somelist{0}.val = 12  
somelist{1}.val = 0.5  
somelist{2}.val = 8
```

Then the value assigned to the resulting minimum attribute would be 0.5 and the value assigned to the maximum attribute would be 12.

Note that non-numeric minima and maxima may also be extracted. For example, if this feature enters this transformer:

```
somelist{0}.val = 'apple'  
somelist{1}.val = 'donut'  
somelist{2}.val = 'aardvark'
```

Then the value assigned to the resulting minimum attribute would be 'aardvark' and the value assigned to the maximum attribute would be 'donut'.

If some values are numeric and some are not, the minimum and maximum values extracted may not be useful.

### Technical History

Associated FME function or factory: @TCL

## ListRemover

Removes the selected lists from incoming features.

### Parameters

Lists to Remove

After you connect this transformer, click the Browse button and select the lists to remove.

Note that if you select to remove a list, your selection will include any list attributes or nested lists. For example, if you select to remove a list called

```
list{}
```

then

```
list{}.attr or list{}.sublist{}
```

will also be removed.

### Additional Resources

[More Information on Lists](#)

### Transformer Category

Lists

### Technical History

Associated FME function or factory: @RemoveAttributes

## ListRenamer

Renames a list name, or the components of a list.

For example, if this feature enters this transformer:

```
somelist{0}.length = 7.3  
somelist{0}.kind = 'paved'  
somelist{1}.length = 8.4  
somelist{1}.kind = 'smooth'  
somelist{1}.lanes = 2  
somelist{2}.length = 1.1  
somelist{2}.kind = 'rough'
```

and if the 'List/Component Name' value is "somelist" and the 'Replace With' value is "newlist," then the feature leaving the transformer will have these attributes:

```
newlist{0}.length = 7.3  
newlist{0}.kind = 'paved'  
newlist{1}.length = 8.4  
newlist{1}.kind = 'smooth'  
newlist{1}.lanes = 2  
newlist{2}.length = 1.1  
newlist{2}.kind = 'rough'
```

## Additional Resources

[More Information on Lists](#)

## Transformer Category

Lists

## Technical History

Associated FME function or factory: @Tcl2

## ListSearcher

Category: Lists

Searches a list to find a value and returns the index of the value in the list. If the given string is not in the list, the returned index is -1.

For example, if this feature enters this transformer:

```
somelist{0}.length      = 7.3
somelist{0}.kind       = 'paved'
somelist{1}.length     = 8.4
somelist{1}.kind       = 'smooth'
somelist{1}.lanes      = 2
somelist{2}.length     = 1.1
somelist{2}.kind       = 'rough'
```

and the somelist{ }.kind list attribute is searched for the value smooth, then the index attribute would be set to 1.

If Search Type is "First not matching" then it will return the index of the first non-matched entry in the list.

The greater than/less than and their "or equal to" variants similarly search for the first element in the list that satisfies the criteria. Numerical comparisons are used if both the list element and the search value can be converted to floating point numbers, otherwise, string comparisons are used.

If search type is "First regular expression match" then it uses a regular expression to search for a first matching entry in the list.

Advanced Regular Expressions (AREs) are supported. Search the FME Functions and Factories on-line help for a complete description of AREs. In brief, An ARE is one or more branches, separated by '|', matching anything that matches any of the branches.

### Additional Resources

[More Information on Lists](#)

### Transformer Category

Lists

### Technical History

Associated FME function or factory: @SearchList

## ListSorter

Category: Lists

Sorts the elements of the given list. The sorting can either be alphabetic or numeric, and in either increasing or decreasing order.

For example, if an input feature had these attributes:

```
somelist{0} = '3'  
somelist{1} = '17'  
somelist{2} = '4'  
somelist{3} = '9'  
somelist{4} = '2'
```

and the list name is set to `somelist{}`, then the feature leaving the transformer will have these attributes:

```
somelist{0} = '2'  
somelist{1} = '3'  
somelist{2} = '4'  
somelist{3} = '9'  
somelist{4} = '17'
```

Note that if there were other attributes in the list "parallel" to the attribute being operated on, these will also be moved in the list to stay in "parallel" with the key attribute.

For example, if the input feature had these attributes:

```
somelist{0}.kind = 'paved'  
somelist{0}.count = '3'  
somelist{1}.kind = 'smooth'  
somelist{1}.count = '17'  
somelist{2}.kind = 'trail'  
somelist{2}.count = '4'  
somelist{3}.kind = 'rough'  
somelist{3}.count = '9'  
somelist{4}.kind = 'logging'  
somelist{4}.count = '2'
```

and the list name is set to `somelist{ }.count`, then the feature leaving the transformer will have these attributes:

```
somelist{0}.kind = 'logging'  
somelist{0}.count = '2'  
somelist{1}.kind = 'paved'  
somelist{1}.count = '3'  
somelist{2}.kind = 'trail'  
somelist{2}.count = '4'  
somelist{3}.kind = 'rough'  
somelist{3}.count = '9'  
somelist{4}.kind = 'smooth'  
somelist{4}.count = '17'
```

Lastly, if the original list was sparse, that is, it was missing some elements, the sorting operation will have the side effect of compacting the list to make all elements have consecutive indices. For example, if the input feature had these attributes:

```
somelist{0}.kind = 'paved'  
somelist{0}.count = '3'  
somelist{3}.kind = 'rough'
```

```
somelist{3}.count = '9'  
somelist{4}.kind = 'logging'  
somelist{4}.count = '2'
```

and the list name is set to `somelist{}`.count, then the feature leaving the transformer will have these attributes:

```
somelist{0}.kind = 'logging'  
somelist{0}.count = '2'  
somelist{1}.kind = 'paved'  
somelist{1}.count = '3'  
somelist{3}.kind = 'rough'  
somelist{3}.count = '9'
```

### **Additional Resources**

More Information on Lists

### **Technical History**

Associated FME function or factory: @TCL



## ListSummer

Computes the sum of all the elements of a list.

Each element must be a valid integer or floating point number. List elements that are non-numeric will be considered to have a value of 0.

For example, if this feature enters this transformer:

```
somelist{0}.val = 12  
somelist{1}.val = 0.5  
somelist{2}.val = 8  
somelist{3}.val = 'a'  
somelist{4}.val = 1
```

Then the value assigned to the resulting sum attribute would be 21.5.

## Transformer Category

Lists

## Technical History

Associated FME function or factory: @TCL

## Logger

Logs each feature to the translation log. All attributes and geometry of the feature will be output.

Limits can be placed on both the number of features that will be logged, and the number of coordinates that will be logged per feature.

### Parameters

#### Log Message

You can choose to add a message to the log file, which can be useful to identify features if you have many Logger transformers. If present, this message is output to the log file before each feature is logged.

#### Maximum Logged Coordinates

The number of coordinates to be logged. If not specified, the first 5 and the last 5 coordinates will be logged. If the feature has more than the maximum allowed coordinates, then only a set of coordinates at the beginning and end of the feature will be logged.

If you enter -1, all coordinates will be logged; if you enter 0, no coordinates will be logged.

If any other value is specified, only the first <maxCoords> coordinates will be logged.

#### Maximum Logged Features

The maximum number of features logged for this message.

If more features than the maximum to be logged enter the transformer, the features over the limit are not logged.

Note: The Workbench parameter *Max Features to Log* takes precedence over this transformer parameter.

#### Log Feature Type

The Log Feature Type text will set the feature type of the feature just prior to it being logged. If the log file is being saved, the logged features are also saved in a companion FFS file, and this setting is most useful to control the feature type of the features in this FFS file.

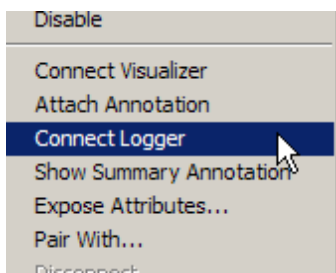
If this parameter is left blank, the feature type is constructed from the preceding transformer's name and output port. This can be useful to determine the origin of features when there are multiple Loggers in a workspace, or multiple connections made to the same Logger. For example, if this line appears in the log:

Feature Type: 'MyCreator\_CREATED'

it indicates that the feature was output through the CREATED port of a transformer named "MyCreator".

### Usage Notes

- An alternate way to see the attributes and geometries of features is to route them into an Inspector transformer, which will display them in the FME Data Inspector.
- To quickly attach Logger transformers: right-click on one or more selected reader feature types and choose *Connect Logger*.



## fmeopedia

See fmeopedia for additional information about this transformer.

**Transformer Category**

Infrastructure

**Technical History**

Associated FME function or factory: @Log

## MapInfoStyler

Prepares features for output to MapInfo MIF/MID or MapInfo TAB by providing a convenient interface to set a variety of MapInfo format-specific attributes.

### Parameters – Color

#### Color

This parameter specifies the pen color that will be used to render the feature. The pen color determines the color of lines and area boundaries.

To edit this parameter, click the browse button to the right of the text field.

Format attribute set:

- `fme_color`
- `mapinfo_symbol_color`, `mapinfo_pen_color`, `mapinfo_text_fontfgcolor`
- `mif_symbol_color`, `mif_pen_color`, `mif_text_fontfgcolor`

### Parameters – Symbols

If this section is active, point features will be turned into the specified symbol.

Features with other geometry types will not be affected by settings in this section.

#### Symbol Point Size

Symbol Point Size specifies the size of the symbol in points.

Format attribute set: `mapinfo_symbol_size`, `mif_symbol_size`

#### Symbol Type

Symbol Type specifies the type of symbol to be associated with the point.

Format attributes set:

Symbol Type Option	mapinfo_type	mif_type
TrueType Font	mapinfo_font_point	mif_font_point
MapInfo 3.0	mapinfo_point	mif_point
Custom Symbols	mapinfo_custom_point	mif_custom_point

#### TrueType Symbol

TrueType Symbol is available only if Symbol Type is TrueType Font. It specifies the number of the symbol to be associated with the point. To select a symbol, click the browse button to the right of the text field.

Format attributes set:

- `mapinfo_symbol_shape`, `mapinfo_symbol_font`
- `mif_symbol_shape`, `mif_symbol_font`

#### Symbol Font Effects

Symbol Font Effects available only if Symbol Type is TrueType Font. It specifies the display style for the symbol.

Format attribute set: `mapinfo_symbol_style`, `mif_symbol_style`

#### Symbol Rotation

Symbol Font Effects available only if Symbol Type is TrueType Font. It specifies The rotation angle for the symbol, measured in degrees counter-clockwise from horizontal.

Format attributes set: `mapinfo_symbol_angle`; `mif_symbol_angle`

## Custom Symbol File

Custom Symbol File is available only if Symbol Type is Custom Symbols. It specifies the custom symbol associated with the point. To select a symbol, click the browse button to the right of the text field.

Format attributes set: `mapinfo_symbol_file_name; mif_symbol_file_name`

## Custom Symbol Effects

Custom Symbol Effects is available only if Symbol Type is Custom Symbols. It specifies the display style for the symbol.

Format attributes set: `mapinfo_symbol_style; mif_symbol_style`

## MapInfo 3.0 Symbol

MapInfo 3.0 Symbol is available only if Symbol Type is MapInfo 3.0. It specifies the symbol to display for the point. To select a symbol, click the browse button to the right of the text field.

Format attributes set: `mapinfo_symbol_shape; mif_symbol_shape`

## Parameters – Lines/Borders

If this section is active, features containing lines will be prepared for output to MapInfo.

Features with other geometry types will not be affected by settings in this section.

### Line Width Type

Line width type specifies the how the line width will be interpreted.

### Line Width

Line width specifies the thickness of the line, when it is rendered.

Format attributes set: `mapinfo_pen_width; mif_pen_width`

### Line Pattern

Line pattern specifies how the line should be rendered. To select a pattern, click the browse button to the right of the text field.

Format attribute set: `mapinfo_pen_pattern; mif_pen_pattern`

### Line Interleaving

Line interleaving specified whether or not the line should be interleaved. You can use interleaved line styles to create the appearance of intersections for overlapping intersections and lines within a single layer.

Format attributes set: `mapinfo_pen_pattern; mif_pen_pattern`

## Parameters – Regions

If this section is active, then area and ellipse features will be prepared for output to MapInfo.

Features with other geometry types will not be affected by settings in this section.

### Fill Foreground Color

Fill foreground color specifies the foreground color when the area is filled. To edit this parameter, click the browse button to the right of the text field.

Format attributes set: `fme_fill_color; mapinfo_brush_foreground; mif_brush_foreground`

### Fill Background Color

Fill foreground color specifies the background color when the area is filled. To edit this parameter, click the browse button to the right of the text field.

Format attributes set: `mapinfo_brush_background; mif_brush_background`

## Parameters – Text

If this section is active, text features will be prepared for output to MapInfo.

Features with other geometry types will not be affected by settings in this section.

### Text Background Color

Text background color specifies the background color used when the text is drawn. To edit this parameter, click the browse button to the right of the text field.

Format attribute set: `mapinfo_text_fontbgcolor; mif_text_fontbgcolor`

### Text TrueType Font Name

Text TrueType Font Name specifies the font to use for the text. To edit this parameter, click the browse button to the right of the text field.

Format attribute set: `mapinfo_text_fontname; mif_text_fontname`

### Text Font Effects

Text font effects specifies the effects to apply to the font. To edit this parameter, click the browse button to the right of the text field.

Format attributes set:

- `mapinfo_text_fontstyle_bold; mapinfo_text_fontstyle_italic; mapinfo_text_fontstyle_underline; mapinfo_text_fontstyle_strikeout; mapinfo_text_fontstyle_outline; mapinfo_text_fontstyle_shadow; mapinfo_text_fontstyle_inverse; mapinfo_text_fontstyle_blink; mapinfo_text_fontstyle_opaque; mapinfo_text_fontstyle_halo; mapinfo_text_fontstyle_allcaps; mapinfo_text_fontstyle_expanded`
- `mif_text_fontstyle`

### Text Justification

Text justification specifies how to place the text.

Format attributes set: `mapinfo_text_justification; mif_text_justification`

### Text Line Spacing

Text line spacing specifies the space between lines of multiline text. The measure is expressed as a multiple of the text height.

Format attributes set: `mapinfo_text_spacing; mif_text_spacing`

## Additional References

For more information about DWG/DXF styling:

- See the MapInfo TAB and MapInfo MIF/MID Reader/Writer chapters (Feature Representation sections) in the *FME Readers and Writers* manual. In Workbench, select Help > FME Readers and Writers Reference.

## Transformer Category

Stylers

## Matcher

Detects features that are matches of each other. Features are declared to match when they have matching geometry, matching attribute values, or both. A list of attributes which must differ between the features may also be specified.

## Output

- All features matching another feature are output to the MATCHED port.
- A single copy of each matched feature is sent to the SINGLE\_MATCHED port.
- All non-matching features are sent to the NOT\_MATCHED port.

## Parameters

### Match Geometry

The Match Geometry parameter controls whether 2D or 3D (or NO) geometry must be the same before a match is declared. FULL makes sure 3D, measures, and Geometry Traits all match. When comparing raster geometries:

- 2D matches the properties
- 3D matches the properties and values
- FULL matches the properties, values and geometry traits.

When comparing surface and solid geometries: 2D behaves the same way as 3D, that is, Z values will also be compared.

### Attributes to Match

Controls which attributes that are part of the input feature must have the same value before a match is declared. The values from all attributes matching the regular expression are concatenated together and compared to determine a match.

### Match all Attributes Except

Instead of specifying Attributes to Match, this parameter can be used. In this case, the names and values of all attributes which are not explicitly given are concatenated to determine a match. This clause can be given with no arguments in order to force matching of all attributes.

### Match all Attributes

This clause causes similar behavior to Match all Attributes Except, except that the only fields excluded from the match are those given by Attributes that must Differ, if any.

### Attributes that must Differ

Controls which attributes that are part of the input feature must have different values before a match is declared. The values from all attributes matching the regular expression are concatenated together and compared to determine a match.

### Treat Blank Attribute Value

If this parameter is set to As Attribute Value, an attribute will be considered different for two features when one feature contains the attribute with a blank value and the other feature does not contain the attribute at all. This affects both Matching Attributes and Differing Attributes.

### Lenient Geometry Matching

If this parameter is set to yes, then the order of points in line and area features will be ignored.

Composition differences between paths and lines will be ignored. True arcs and ellipses versus their stroked polygon equivalents will be ignored in Aggregates, Polygons, Donuts, Paths, and all other multis. When comparing raster geometries, only the extents are compared.

### Interior Vertex Tolerance

This parameter determines how close together interior vertices must be in order for them to be declared a match. Note that the start and end points of features should be LESS than the parameter value. That is, if two vertices are 2 meters apart, and the parameter is set to 2, they will not match. This value is optional.

## Extra Vertex Tolerance

When geometry is being matched, Extra Vertex Tolerance can allow for extra vertices along line segments. A value of 0 means that no such extra vertices are permitted. A nonzero value controls how close any extra vertices must be to the line connecting the adjoining matching vertices.

Note: For surface and solid geometries, Interior Vertex Tolerance and Extra Vertex tolerance are ignored and assumed to be 0.0. This transformer does not support surfaces or solids in the input if the Lenient Geometry Match is set to yes.

## Match ID Output Attribute

An ID is added to each set of matched features so that it is possible to build a relationship between them if required.

## List Name on SINGLE\_MATCHED output

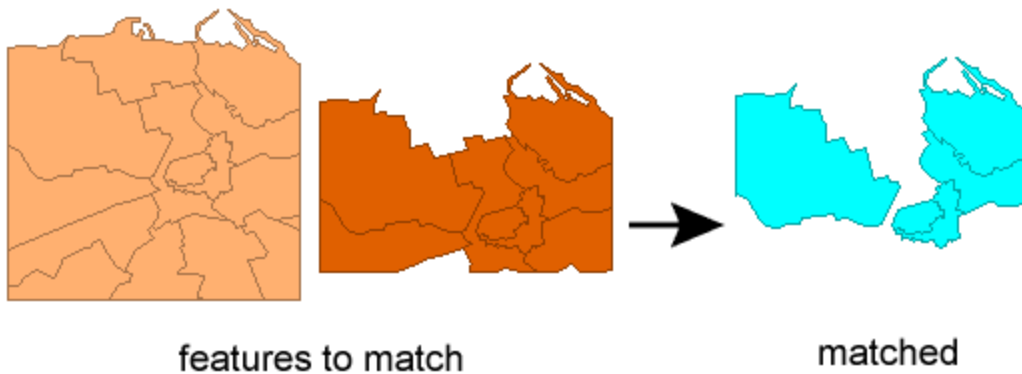
If provided, a list of all attributes from features contributing to each SINGLE\_MATCHED output will be constructed with this name.

## Related Transformers

### DuplicateRemover

The ChangeDetector provides an alternate (but less general) approach which may be more convenient for certain applications.

## Example



## Transformer Category

Filters

## Technical History

Associated FME function or factory: MatchingFactory



## MeasureExtractor

Extracts the measures of geometries that match the given type, and places them in attributes or list attributes.

### Input

- INPUT: Feature types that contain point, line arc, area geometry, or a vertex of linear or area geometry

### Parameters

The Type parameter that you choose determines which of the remaining transformer parameters become enabled.

Type

**Individual Vertex by Index** sets the value of the attribute specified in the *Destination Attribute of Point or Vertex* to the measure extracted from a point, curve or area at the vertex specified by the *Index*.

A negative index can be used to indicate the position of the vertex in which the measure is to be extracted relative to the end of the geometry (-1 is the last coordinate, -2 the second last, and so on). If the geometry of a feature is a point, then only index 0 or index -1 can be used to retrieve the measure value. Any geometry other than a point, curve or area is ignored.

**Point** sets the value of the attribute specified in the *Destination Attribute of Point or Vertex* to the measure extracted from the point.

**End Points of Arc** sets the values of attributes specified in the *Destination Start/End Point Attribute of Arc* to the measures extracted from the start and end points of the arc respectively.

**Whole Line or Whole Area** adds the list attribute specified in the *Destination List Attribute of Line or Area* and sets the list elements to the measures extracted from the line or area. A Path geometry or path boundary of an area is not supported. A PathSplitter transformer can be used to split a path to its respective segments in order to retrieve the measure(s) on each individual segment. A PathBuilder can then join the segments back into a path.

Source Measure Name

Retrieves the measures of geometries matching the given type. If the geometry is of a different type, the feature will not be modified. If a Source Measure Name is not supplied, the default measures will be used.

### Transformer Category

Linear Referencing

### Related Transformers

PathSplitter

PathBuilder

### Technical History

Associated FME function or factory: @Geometry

## MeasureGenerator

Creates a set of measures attached to the geometry of the feature, where each value is the distance from the start of the line up to that vertex, multiplied by the given Multiplier.

This transformer is often used to load useful measure values into an attribute for writing to measure-supporting formats such as Shape, Geodatabase, and SDE.

### Parameters

#### Length Dimension

The Length Dimension parameter controls how the distance will be calculated. The distance can be calculated as either a two-dimensional distance (in which case any z coordinates on the features will be ignored), or as a three-dimensional distance.

The first value will always be 0, since the length up to the very first point is 0. The last value will be the length of the entire line multiplied by the value set in the Multiplier parameter.

#### Multiplier

A Multiplier is applied to each distance before it is added to the list, to potentially scale each measurement by a fixed amount. By default, the multiplier is 1.

#### Destination Measure Name

You can enter an optional name for the resulting measures.

### Related Transformers

- To get measures as a list or a series of attributes (for example, if you need to manipulate the calculated measure values), place a MeasureGenerator followed by a MeasureExtractor and a ListConcatenator.
- To calculate the entire length of a feature, use the LengthCalculator.
- To calculate the partial length of a feature, use the LengthToPointCalculator.

### Transformer Category

Linear Referencing

### Technical History

Associated FME function or factory: @Length

## **MeasureRemover**

Removes measures from a feature's geometry.

### **Parameters**

Remove All

Yes: all measures will be removed, including measures present on components of aggregate geometries.

No: only the measure named will be removed. In this mode, the measure will not be removed from collections such as Aggregates, MultiCurves, and MultiPoints; measures will remain on the components, if they are present.

Name of Measure to Remove

Type the measure name, or choose an attribute that contains the measure name. (This can be left blank to remove the default measure.)

### **Transformer Category**

Linear Referencing

### **Feature Blocker?**

No

### **Technical History**

Associated FME function or factory: @Geometry

## MeasureSetter

Sets measure(s) on a point, line, arc, area geometry or a vertex of a linear geometry to attribute value(s) of given attribute(s) or list attribute.

For Point, Arc, Line or Area, if the geometry is of a different type than what is specified in Geometry Type, the feature will not be modified.

### Parameters

The Type parameter selection will determine which of the other parameters is enabled for this transformer.

#### Type

##### Individual Vertex by Index

Sets the measure of a point, curve, or area at the vertex specified by the Index to the value or attribute value specified in Measure Value of Point or Vertex. A negative index can be used to indicate the position of the vertex in which the measure is to be set relative to the end of the geometry (-1 is the last coordinate, -2 the second last, and so on). If the geometry of a feature is a point, then only index 0 or index -1 can be used to retrieve the measure value. Any geometry other than a point, curve or area is ignored.

##### Point

Sets the measure of the point to the value or value of the attribute specified in Measure Value of Point.

##### End Points of Arc

Sets the measure of the start and end points of an arc to the values or values of the attributes specified in Start Point Measure Value of Arc and End Point Measure Value of Arc respectively.

##### Whole Line or Whole Area

Sets the measure of a line or area to the values of the list attribute specified in Source Measure List Attribute of Line or Area. The number of elements in the list attribute must match the number of points of the line or area. A Path geometry or path boundary of an area is ignored. A PathSplitter transformer can be used to split a path to its respective segments in order to set the measure(s) on each segment separately. A PathBuilder can also be used to join the segments back into a path.

##### Destination Measure Name

If no Destination Measure Name is specified, the default measures will be set.

### Transformer Category

Linear Referencing

### Technical History

Associated FME function or factory: @Geometry

## **MeshMerger**

Merges mesh features (features with IFMEMesh geometries) into a single output mesh.

The final merged mesh is post-processed to remove duplicate vertices, texture coordinates, and vertex normals.

### **Parameters**

Merge Break Attributes

Select from the list of attributes.

The feature being created is output whenever one of the Merge Break Attributes' values changes. When this happens, the feature with the differing attribute is not added to the current output feature; instead, it begins the next feature to be output.

List Name (optional)

If supplied, produces a list of all the attributes of each feature that was merged when creating the output mesh.

### **Usage Notes**

Before using this transformer, a Deaggregator may be necessary to extract meshes stored in aggregates, multi-surfaces, or composite surfaces.

### **Transformer Category**

3D

### **Technical History**

Associated FME function or factory: ConnectionFactory

## **MGRSGeometryExtractor**

Calculates a Military Grid Reference System (MGRS) code based on the geometry of a feature.

The MGRS code for a position consists of a group of letters and numbers which include the following elements:

- grid zone designation
- the 100,000-meter square letter identification
- the grid coordinates (also referred to as rectangular coordinates); the numerical portion of the reference expressed to a desired refinement

If a non-point feature (that is, a feature with more than one point) is passed in, then the first point of the geometry will be used for the conversion.

### **Parameters**

#### Ellipsoid

The ellipsoid used for the conversion. This can be any ellipsoid name supported by FME.

#### Lettering Type

The type of lettering used can be WGS84 or Bessel.

#### Precision

The granularity of the resulting code is determined by the specified Precision which is an integer from 0 to 5 (inclusive). A precision of 5 locates a point within 1-meter square and a precision of 0 locates a point within 100-km square.

#### MGRS Code Attribute

The result of the conversion is stored in the MGRS Code Attribute of the feature.

### **Transformer Category**

Coordinate Systems

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @MGRS

## **MGRSGeometryReplacer**

This transformer is used to convert Military Grid Reference System (MGRS) code to longitude and latitude coordinates.

The geometry of an input feature will be replaced with a point at the longitude/latitude values obtained from the MGRS code.

### **Parameters**

Ellipsoid

The ellipsoid used for the conversion. This can be any ellipsoid name supported by FME.

Lettering Type

The type of lettering used can be WGS84 or Bessel.

MGRS Code

An MGRS code used to convert to lat/long coordinates. You can choose this parameter from the list of attributes in the pull-down list, or enter an MGRS string.

### **Transformer Category**

Coordinate Systems

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @MGRS

## **MGRSToLatLongConverter**

Converts Military Grid Reference System (MGRS) code to longitude and latitude coordinates.

### **Parameters**

#### Ellipsoid

The ellipsoid used for the conversion. This can be any ellipsoid name supported by FME.

#### Lettering Type

The type of lettering used can be WGS84 or Bessel.

#### MGRS Code

An MGRS code used to convert to lat/long coordinates. You can choose this parameter from the list of attributes in the pull-down list, or enter an MGRS string.

#### Longitude, Latitude Attributes

The longitude and latitude values obtained from the MGRS code will be stored in the Longitude and Latitude attributes of the feature, respectively.

### **Transformer Category**

Coordinate Systems

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @MGRS



## **MinimumAreaForcer**

Ensures that features with polygon geometry have an area that is equal to, or in excess of, the specified minimum area.

### **Ports**

ENLARGED: Features that were scaled will be output via the ENLARGED output port.

UNTOUCHED: Features that were not scaled, or have non-area geometry will be output via the UNTOUCHED port.

### **Parameters**

#### Minimum Area

This is the area, expressed in ground units, that all output features will exceed. You can also choose to use the area of a selected attribute.

Any feature with polygon geometry that is smaller than the specified area will be scaled such that the geometry's area meets the required minimum value. The center of gravity of the feature will remain unchanged.

### **Transformer Category**

Manipulators

### **Technical History**

Associated FME function or factory: TeeFactory

## **MinimumSpanningCircleReplacer**

Replaces the geometry of the feature with a polygon representing its minimum spanning circle. The minimum spanning circle is defined as the smallest circle that encloses all vertices of the passed in feature.

If the feature was an aggregate feature, a single minimum spanning circle is computed from all vertices of all geometries in the aggregate. If the feature was a linear feature, it will be turned into a polygonal feature. No change is made to a point feature.

## **Transformer Category**

Manipulators

## **FME Licensing Level**

FME Professional edition and above

## **Technical Reference**

Associated FME function or factory: @MinimumSpanningCircle

## **ModuloCounter**

Adds an attribute holding the next integer in a sequence, restarting the count at 0 whenever the sequence reaches some maximum value.

Each feature will get a number in the range from 0 to one less than the maximum added to it.

## **Transformer Category**

Calculators

## **Technical History**

Associated FME function or factory: @Count

## MRF2DCleaner

Note: The MRFCleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985. This package includes all MRFCleaner transformers.

MRF Geosystems Corporation (www.mrf.com) has produced cleaning software and made it available to FME users to apply to data as it is transformed between arbitrary input and output formats.<sup>1</sup>

The MRFCleaner repairs geometry, particularly during data migration from CAD to GIS, and is built upon the *MRFCleanFactory*, which is an integration of MRF's cleaning technology into FME. The MRFCleaner fixes geometric problems in input data such as line overshoots and undershoots within the user-specified tolerance. It is useful for multi-layer and multi-tolerance two-dimensional data cleaning. Typical applications include the correction of utility maps, parcel maps, topographic maps and resource maps as data is migrated from one system to another.

The MRFCleaner includes the following functionality:

- fuzzy tolerance
- extending lines
- weeding lines
- joining lines
- processing short elements
- removing gaps
- removing duplicates
- removing dangles
- performing conflation

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer. This allows feature data from a high-quality data source to be assigned a low tolerance and integrated with data from a lower-quality data source which would be given a larger tolerance.

Geometries such as path, polygon, donut, ellipse, elliptical arc, multi-area, multi-curve, text, and multi-text are converted to basic geometries such as point, line, path, arc or multi-point prior to the cleaning process. The cleaner understands and works with circular arcs. Input features with invalid geometries are ignored and deleted.

### Usage Tips

You can also use one more of the following transformers to perform singular MRFCleaner operations. These transformer parameters are all available as part of this MRF2DCleaner transformer, but you may wish to use separate transformers so that the operations are more easily visible in your workflow.

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## Transformer Parameters

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

If Compute Intersections is set to Yes, intersections between all input features are computed, breaking arcs and lines wherever an intersection occurs. A fuzzy intersection is also created from geometries which are within one of the tolerance distances, but do not actually touch or cross.

If Correct Undershoots is set to Yes, arcs and lines that are within the specified tolerance are extended – while maintaining line-work direction. No intersections are created while doing this. This option does not process overshoots; a combination of Compute Intersections and Delete Short Geometries can serve this purpose.

If Delete Short Geometries is set to Yes, features that have lengths smaller than the specified tolerances are deleted.

If Remove Duplicate Geometries is set to Yes, duplicated features are deleted. Features are considered to be duplicates if their geometries are within tolerance and only features with a smaller tolerance will remain after cleaning.

If Generalize Lines is set to Yes, a number of vertices of lines are removed. The number of vertices removed is controlled by a weeding tolerance of the value of (Filter Factor \* tolerance) or (Filter Factor \* value of Feature Tolerance Attribute). The latter is always used when it is valid and the Feature Tolerance attribute is specified. The larger the value of weeding tolerance, the more vertices will be removed.

The default value of Filter Factor is 1.0 and the minimum value is 0.0.

If Join Geometries is set to Yes, then singly-connected features are joined to form longer ones. A pair of linear features become candidates for joining only when the two are singly connected at a given node or end point.

If Conflate Geometries is set to Yes, then the geometry of a feature can be changed to match that of another, if the two are approximately the same to begin with.

If Remove Dangles is set to Yes, then features that have at least one free end point and have lengths smaller than (Dangle Factor \* tolerance) or (Dangle Factor \* value of Feature Tolerance Attribute) are removed. The default value of Dangle Factor is 1.0 and the minimum is 0.0.

If Clean Area Geometries is set to Yes, then area features such as polygons or donuts will be cleaned without stroking them first.

## Output Ports

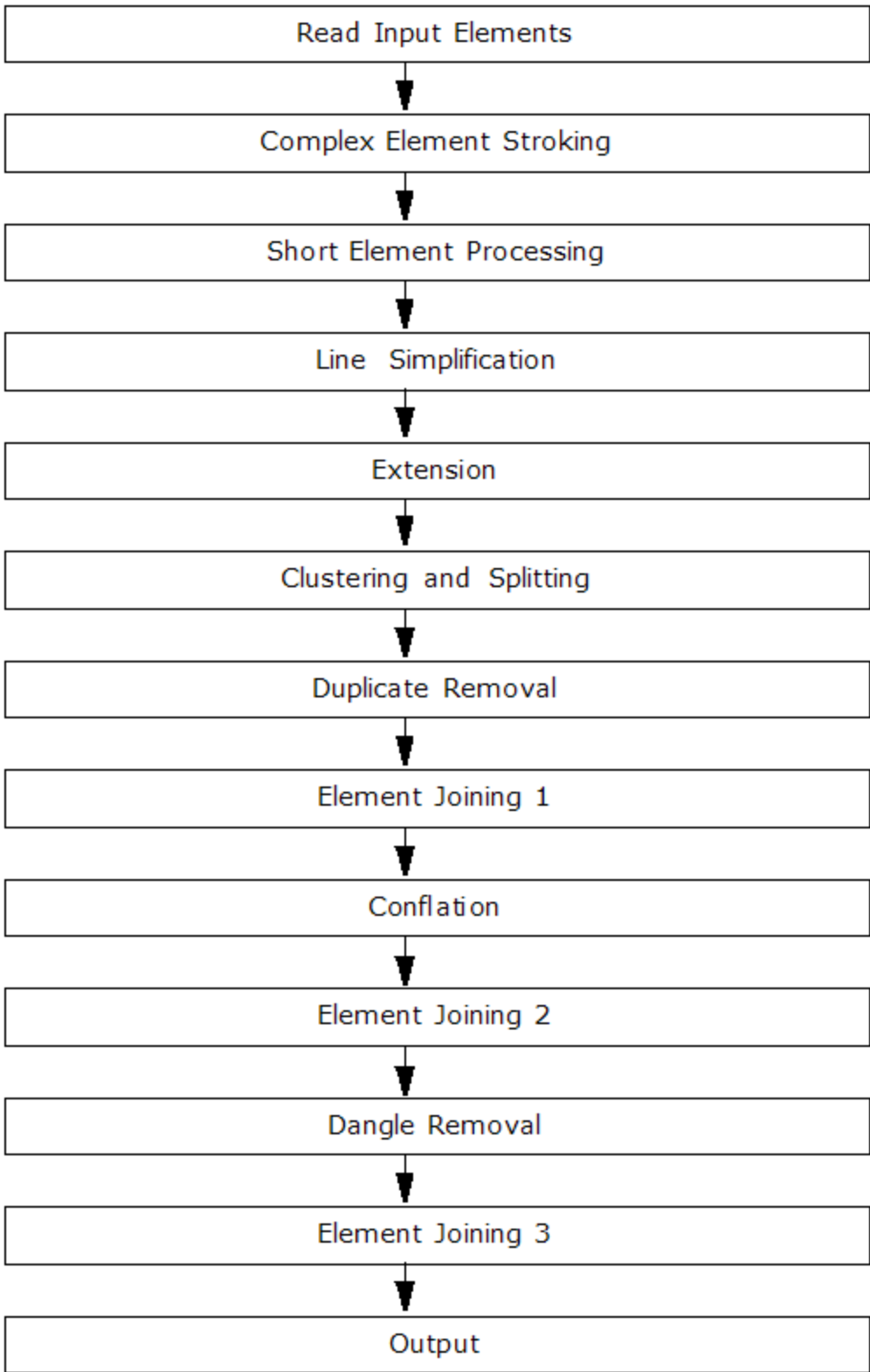
Each feature that is output through the CLEANED port has a new attribute "mrf\_clean\_status" added to specify whether the feature is modified, created, or will remain unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

Features can also be output through the FLAGGED port if any of the Remove Dangles, Delete Short Geometries and Compute True Intersections is set to Flag. Each of these features has a new attribute "mrf\_flag\_status" added to specify whether this feature is flagged as being shorter than the tolerance value ("short"), a dangling geometry ("dangle") or an intersection point ("intersection").

## Module Workflow

MRFCleaner Modules provide more detailed information on the modules in the underlying MRFCleanFactory.

This [default workflow](#) is suitable for most situations. However, using the individual modules, it is possible to create any number of customized workflows for specific projects and/or datasets (for example, in Workbench, by using a series of consecutive MRFCleaner transformers or custom transformers). It is important, however, to understand the data being processed and the desired end result.



**More Information**

- See General Processing Tips.
- MRFCleaner Sample Results

**Transformer Category**

MRF

**Technical History**

Associated FME function or factory: MRFCleanFactory

## **MRF2DConflator**

Changes the geometry of a feature to match that of another, if the two have approximately the same shape and location, and have matching end-points.

### **Output Ports**

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### **Parameters**

Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

### **Usage Notes**

This transformer performs the same operation as the MRF2DCleaner with the Conflate Geometries set to Yes and no other options selected. See the MRF2DCleaner more details.

### **FME Licensing Level**

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### **Transformer Category**

MRF

### **Related Transformers**

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

### **Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.



## MRF2DDangleRemover

Removes features that have at least one free endpoint and have lengths smaller than (Dangle Factor \* Cleaning Tolerance) or (Dangle Factor \* value of Feature Tolerance Attribute).

### Output Ports

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".
- **FLAGGED:** If Dangle Action is set to Remove Short and Flag Long, for each feature that is a dangle but is longer than this tolerance, a point, label, or circle (depending on the value of Flag Type) will be output through the FLAGGED port.

### Parameters

#### Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

#### Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

#### Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

#### Dangle Action

If Dangle Action is set to Remove Short and Flag Long, for each feature that is a dangle but is longer than this tolerance, a point, label, or circle (depending on the value of Flag Type) will be output through the FLAGGED port.

#### Flag Type and Flag Size

For each feature that is a dangle but is longer than the flag size tolerance, this parameter sets the flag type as a point, label or circle, if Dangle Action is set to Remove Short and Flag Long.

### Usage Notes

This transformer performs the same operation as the MRF2DCleaner with the Remove Dangles parameter set to Yes and no other options selected. See the MRF2DCleaner more details. Note that the circle/label flagging is an added option in this transformer.

### Related Transformers

MRF2DConflator

MRF2DDuplicateRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

**FME Licensing Level**

The MRFCleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

**Transformer Category**

MRF

**Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## **MRF2DDuplicateRemover**

Deletes duplicated features. Features are considered to be duplicates if their geometries are within tolerance and only features with a smaller tolerance will remain after cleaning.

### **Output Ports**

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### **Parameters**

Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

### **Usage Notes**

This transformer performs the same operation as the MRF2DCleaner with the Remove Duplicate Geometries set to Yes and no other options selected. See the MRF2DCleaner more details.

### **FME Licensing Level**

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### **Related Transformers**

MRF2DConflator

MRF2DDangleRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

### **Transformer Category**

MRF

### **Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## **MRF2DExtender**

Extends arcs and lines that are within the specified tolerance to correct undershoots while maintaining line-work direction. No intersections are created while doing this.

This does not process overshoots: the MRF2DCleaner with a combination of Compute Intersections and Delete Short Geometries can serve this purpose.

### **Output Ports**

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### **Parameters**

Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

### **Usage Notes**

This transformer performs the same operation as the MRF2DCleaner with the Correct Undershoots set to Yes and no other options selected. See the MRF2DCleaner more details.

### **FME Licensing Level**

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### **Related Transformers**

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

### **Transformer Category**

MRF

## Technical History

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## MRF2DGeneralizer

Removes a number of vertices from lines. The number of vertices removed is controlled by a weeding tolerance, which will be either (Filter Factor \* Cleaning Tolerance) or (Filter Factor \* value of Feature Tolerance Attribute). The latter is always used when it is valid and the Feature Tolerance attribute is specified. The larger the weeding tolerance, the more vertices will be removed.

Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### Output Ports

- CLEANED: Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### Parameters

#### Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

#### Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

#### Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

#### Filter Factor

The default value of Filter Factor is 1.0 and the minimum value is 0.0.

### Usage Notes

This transformer performs the same operation as the MRF2DCleaner with the Generalize Lines parameter set to Yes and no other options selected. See the MRF2DCleaner more details.

### FME Licensing Level

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### Related Transformers

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender

MRF2DIntersector

MRF2DJoiner

MRF2DShortGeometryRemover

### Transformer Category

MRF

## Technical History

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## MRF2DIntersector

Computes intersections between all input features, breaking arcs and lines wherever an intersection occurs. A fuzzy intersection is also created from geometries which are within one of the tolerance distances, but do not actually touch or cross.

### Output Ports

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".
- **FLAGGED:** If Intersection Action is set to Split Lines at Intersections, the input lines will be split at true intersection points into separate output lines. If it is set to Flag Intersections, then true intersection points will be output through the FLAGGED port as points, labels, or circles, depending on the Flag Type selected.

### Parameters

#### Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

#### Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

#### Intersection Action

If set to Split Lines at Intersections, intersections are computed between all input features, breaking arcs and lines wherever an intersection occurs. If set to Flag Intersections, the Flag Type and Flag Size parameters are enabled.

#### Compute Fuzzy Intersections

If set to Yes, a fuzzy intersection is also created from geometries which are within one of the tolerance distances, but do not actually touch or cross.

#### Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

#### Flag Type and Flag Size

This parameter sets the flag type for true intersection points as a point, label or circle, if Intersection Action is set to Flag Intersections.

### Usage Notes

This transformer performs the same operation as the MRF2DCleaner no options selected except Compute True Intersections and/or Compute Fuzzy Intersections. See the MRF2DCleaner more details. Note that the circle/label flagging is an added option in this transformer.

### FME Licensing Level

The MRF2DCleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### Related Transformers

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender



MRF2DGeneralizer

MRF2DJoiner

MRF2DShortGeometryRemover

**Transformer Category**

MRF

**Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## **MRF2DJoiner**

Joins connected features to form longer ones. A pair of linear features become candidates for joining only when the two are connected at a given node or end point.

### **Output Ports**

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

### **Parameters**

Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

### **Usage Notes**

This transformer performs the same operation as the MRF2DCleaner with the Join Geometries parameter set to Yes and no other options selected. See the MRF2DCleaner more details.

### **FME Licensing Level**

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### **Related Transformers**

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DShortGeometryRemover

### **Transformer Category**

MRF

### **Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## **MRF2DShortGeometryRemover**

Removes features that have lengths smaller than the specified tolerance, if Short Geometry Action is set to Remove.

### **Output Ports**

- **CLEANED:** Each feature that is output through the CLEANED port will have a new attribute "mrf\_clean\_status" added to specify whether the feature was modified, created, or unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".
- **FLAGGED:** If Short Geometry Action is set to Flag, such features will not be removed, but for each one a point, label, or circle (depending on the value of Flag Type) will be output through the FLAGGED port.

### **Parameters**

#### Group By

If selected, each group of features with the same values in the Group By attributes will be processed separately from other groups.

#### Cleaning Tolerance

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

#### Feature Tolerance Attribute

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer.

#### Short Geometry Action

You can set the Short Geometry Action to either Remove or Flag features.

#### Flag Type and Flag Size

This parameter sets the flag type as a point, label or circle, if Short Geometry Action is set to Flag.

For each feature that is smaller than the specified tolerance, this parameter sets the flag type as a point, label or circle, if Short Geometry Action is set to Flag.

### **Usage Notes**

This transformer performs the same operation as the MRF2DCleaner with the Delete Short Geometries parameter set to Yes and no other options selected. See the MRF2DCleaner more details. Note that the circle/label flagging is an added option in this transformer.

### **FME Licensing Level**

The MRF Cleaner transformers are available as an extra-cost package from Safe Software. Please contact sales@safe.com or call 604-501-9985.

### **Related Transformers**

MRF2DConflator

MRF2DDangleRemover

MRF2DDuplicateRemover

MRF2DExtender

MRF2DGeneralizer

MRF2DIntersector

MRF2DJoiner

**Transformer Category**

MRF

**Technical History**

Associated FME function or factory: MRFCleanFactory<sup>1</sup>

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

## MRF3DCleaner

Note: This is an extra-cost transformer, available from Safe Software. Please contact sales@safe.com or call 604-501-9985.

MRF Geosystems Corporation (www.mrf.com) has produced cleaning software and made it available to FME users to apply to data as it is transformed between arbitrary input and output formats.<sup>1</sup>

The MRFCleaner repairs geometry, particularly during data migration from CAD to GIS, and is built upon the *MRFCleanFactory*, which is an integration of MRF's cleaning technology into FME.

The MRFCleaner fixes geometric problems in input data such as line overshoots and undershoots within the user-specified tolerance. It is useful for multi-layer and multi-tolerance three-dimensional data cleaning. Typical applications include the correction of utility maps, parcel maps, topographic maps and resource maps as data is migrated from one system to another.

The MRFCleaner includes the following functionality:

- fuzzy tolerance
- extending lines
- weeding lines
- joining lines
- processing short elements
- removing gaps
- removing duplicates
- removing dangles
- performing conflation

The number of layers used in cleaning the data is determined by the number of different tolerance values of input features. Features that have the same tolerances are processed as being on the same layer. This allows feature data from a high-quality data source to be assigned a low tolerance and integrated with data from a lower-quality data source which would be given a larger tolerance.

Geometries such as path, polygon, donut, ellipse, elliptical arc, multi-area, multi-curve, text, and multi-text are converted to basic geometries such as point, line, path, arc or multi-point prior to the cleaning process. The cleaner understands and works with circular arcs. Input features with invalid geometries are ignored and deleted.

### Transformer Fields

Cleaning tolerance is used as the default tolerance unless the Feature Tolerance Attribute is specified and valid. The minimum tolerance allowed is 0.0.

If **Compute Intersections** is set to Yes, intersections between all input features are computed, breaking arcs and lines wherever an intersection occurs. A fuzzy intersection is also created from geometries which are within one of the tolerance distances, but do not actually touch or cross.

If **Correct Undershoots** is set to Yes, arcs and lines that are within the specified tolerance are extended – while maintaining line-work direction. No intersections are created while doing this. This option does not process overshoots; a combination of **Compute Intersections** and **Delete Short Geometries** can serve this purpose.

If **Delete Short Geometries** is set to Yes, features that have lengths smaller than the specified tolerances are deleted.

If **Remove Duplicate Geometries** is set to Yes, duplicated features are deleted. Features are considered to be duplicates if their geometries are within tolerance and only features with a smaller tolerance will remain after cleaning.

If **Generalize Lines** is set to Yes, a number of vertices of lines are removed. The number of vertices removed is controlled by a weeding tolerance of the value of (Filter Factor \* tolerance) or (Filter Factor \* value of Feature Tolerance Attribute). The latter is always used when it is valid and the Feature Tolerance attribute is specified. The larger the value of weeding tolerance, the more vertices will be removed.

The default value of **Filter Factor** is 1.0 and the minimum value is 0.0.

---

<sup>1</sup>Portions of this work are the intellectual property of the MRF Geosystems Corporation and are used under license. Copyright © 2006 MRF Geosystems Corporation. All rights reserved.

If **Join Geometries** is set to Yes, then singly-connected features are joined to form longer ones. A pair of linear features become candidates for joining only when the two are singly connected at a given node or end point.

If **Conflate Geometries** is set to Yes, then the geometry of a feature can be changed to match that of another, if the two are approximately the same to begin with.

If **Remove Dangles** is set to Yes, then features that have at least one free end point and have lengths smaller than (Dangle Factor \* tolerance) or (Dangle Factor \* value of Feature Tolerance Attribute) are removed. The default value of Dangle Factor is 1.0 and the minimum is 0.0.

If **Clean Area Geometries** is set to Yes, then area features such as polygons or donuts will be cleaned without stroking them first.

### **Output Ports**

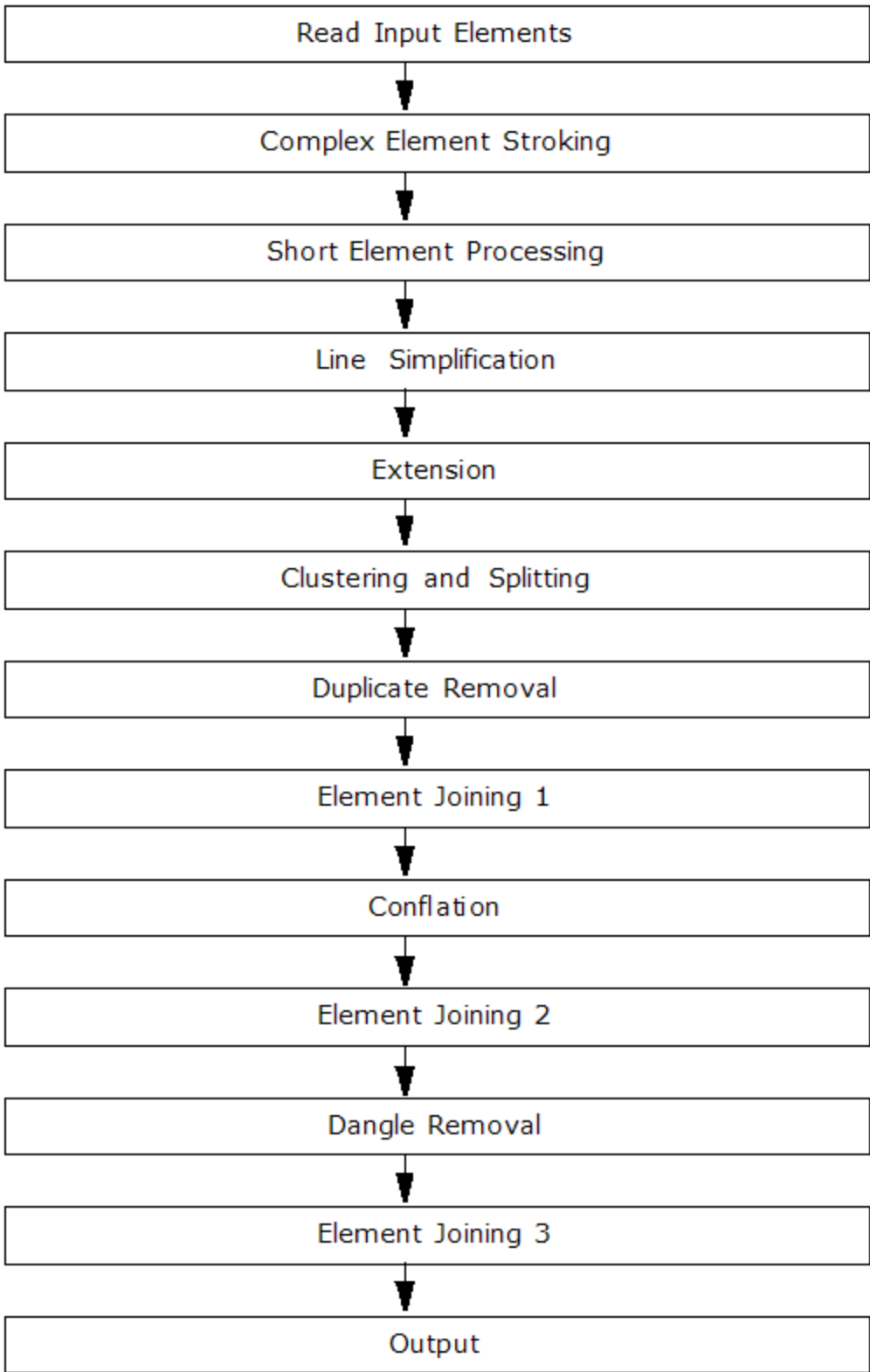
Each feature that is output through the CLEANED port has a new attribute "mrf\_clean\_status" added to specify whether the feature is modified, created, or will remain unchanged in the cleaning process. The possible values of this attribute are "Modified", "Created" and "Original".

Features can also be output through the FLAGGED port if any of the Remove Dangles, Delete Short Geometries and Compute True Intersections is set to Flag. Each of these features has a new attribute "mrf\_flag\_status" added to specify whether this feature is flagged as being shorter than the tolerance value ("short"), a dangling geometry ("dangle") or an intersection point ("intersection").

### **Module Workflow**

MRFCleaner Modules provide more detailed information on the modules in the underlying MRFCleanFactory.

This [default workflow](#) is suitable for most situations. However, using the individual modules, it is possible to create any number of customized workflows for specific projects and/or datasets (for example, in Workbench, by using a series of consecutive MRFCleaner transformers or custom transformers). It is important, however, to understand the data being processed and the desired end result.



**More Information**

- See General Processing Tips.
- MRFCleaner Sample Results

**Transformer Category**

MRF

**Technical History**

Associated FME function or factory: MRFCleanFactory



## MRFCleaner Modules

- TOLERANCE
- SHORT\_ELEMENT
- EXTEND
- INTERSECT
- DUPLICATE\_REMOVE
- JOIN
- CONFLATE
- DANGLER

### **TOLERANCE**

The TOLERANCE clause must be specified. This value serves as the default tolerance of the input features if the TOLERANCE\_ATTR clause is not specified or the features have invalid tolerance value. The minimum value allowed is 0.0.

If the TOLERANCE\_ATTR is specified and the value of the <attribute\_name> is greater than or equal to 0.0, this value is used instead of the value specified in TOLERANCE clause.

### **SIMPLIFY**

If the SIMPLIFY clause is set to "YES", the line weeding / generalization will be included as part of the cleaning process. This involves the removal of line string vertices based on a specified tolerance. This process uses a weeding tolerance of the value of (FILTERFACTOR \* TOLERANCE) or (FILTERFACTOR \* value of TOLERANCE\_ATTR). The latter is used whenever TOLERANCE\_ATTR clause is specified and its value is valid. The larger the value of the weeding tolerance, the more vertices will be removed. The default value of FILTERFACTOR is 1.0.

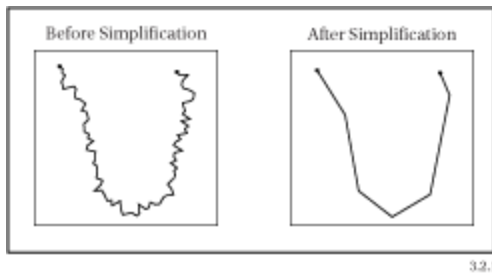


FIGURE 1 Line string before and after simplification

### **SHORT\_ELEMENT**

If the SHORT\_ELEMENT clause is set to "YES", geometries of features that have lengths smaller than the specified tolerances are deleted. Short geometries created during the cleaning process are also deleted.

### **EXTEND**

If the EXTEND clause is set to "YES", the MRF Extend module is enabled. It is useful to extend certain elements – correcting for undershoot – while maintaining line-work direction. If a feature has a free end, this module will attempt to extend it until it meets other line-work within its tolerance; no intersections are created. This module does not process overshoots; the combination INTERSECTION and DANGLER modules can be used to serve this purpose.

The EXTEND clause processes elements in the following manner:

- line-line extension
- line-arc extension
- arc-arc extension

#### **Line-line extension**

In the figure below, lines AB and CD will be extended to point E, if E is within tolerance of both AB and CD, and both B and D are free ends.

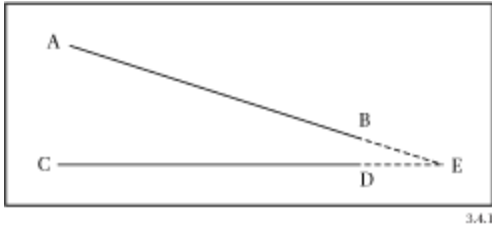


FIGURE 2 Linear extension, example 1

In Figure 3, lines AB and DE cannot be extended to point F, even though the distance BF is less than the tolerance for AB and EF is less than the tolerance for DE. This is because B is not a free end.

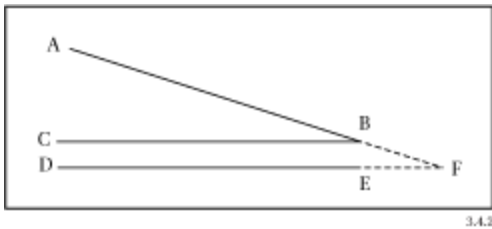


FIGURE 3 Linear extension, example 2

In Figure 4, CD has a tolerance layer larger than both the distances DE and CD. In this case, point D (rather than point C) will be extended to point E, since it has the smaller extension distance.

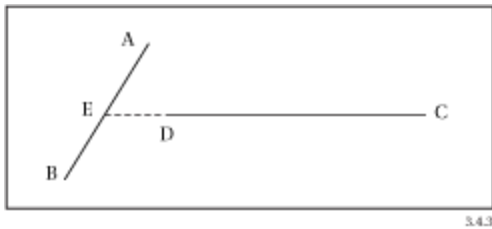


FIGURE 4 Linear extension, example 3

If a line can be extended to more than one element, it will be extended only as far as the closest one. In Figure 5, line AB will be extended to point C (not D or E).

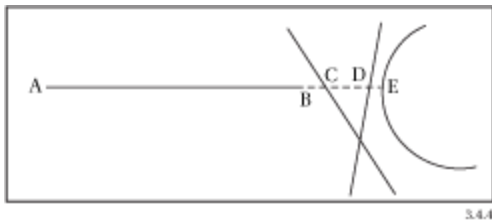
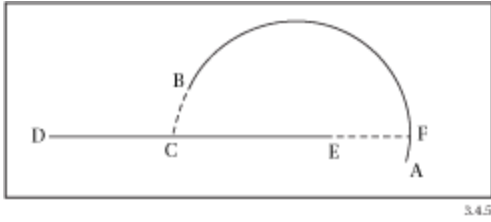


FIGURE 5 Linear extension, example 4

### Line-Arc Extension

Figure 6 shows that circular arc AB will be extended along its path to point C, if BC is less than the tolerance for AB. Also, line DE will be extended to DEF if EF is less than DE's tolerance.

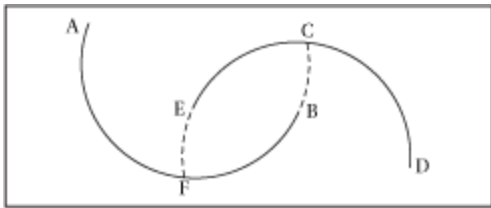


3.4.5

FIGURE 6 Line-arc extension, example 1

**Arc-Arc Extension**

Figure 7 shows that for arcs AB and DE, if BC is less than the tolerance for AB, then arc AB will be extended to C. Arc DE will be extended to F, if EF is less than the tolerance of DE.



3.4.6

FIGURE 7 Arc-arc extension, example 1

**INTERSECT**

MRF Intersect flags and/or creates true and fuzzy intersections. It has element intersection, clustering, and splitting sub-functions.

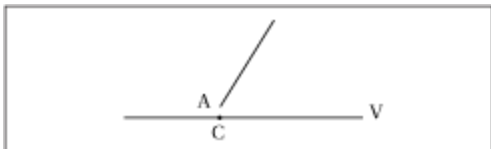
MRF Intersect recognizes two types of relationships:

**True intersections:** When two vectors cross each other, the crossover point (X) is called a true intersection



3.5.1

**Fuzzy intersections:** A fuzzy intersection can be created from elements which are within one of the tolerance distances, but do not actually touch or cross. In other words, the two lines nearly intersect, and the minimum distance from the end of one line to the other line is shorter than one of the lines' tolerances



3.5.2

If the INTERSECT clause is set to "YES", the MRF Intersect module is enabled. This module computes intersections between all input features, breaking arcs and lines wherever an intersection occurs. A fuzzy intersection is also created from geometries which are within one of the tolerance distances, but do not actually touch or cross.

If the FUZZY\_INTERSECT clause is set to "NO" then fuzzy intersections will not be created in the process.

### ***DUPLICATE\_REMOVE***

If the DUPLICATE\_REMOVE clause is set to "YES", the MRF Duplicate remover module is enabled. Features are considered to be duplicates if their geometries are within tolerance and only features with a smaller tolerance will remain after cleaning.

### ***JOIN***

If the JOIN clause is set to "YES", then singly-connected features will be joined to form longer ones. A pair of linear features become candidates for joining only when the two of them are singly connected at a given node or end point.

### ***CONFLATE***

If the CONFLATE clause is set to "YES", then the geometry of a feature can be changed match that of another, if the two are approximately the same to begin with.

### ***DANGLER***

A dangle is a geometry that has at least one free end point. If the DANGLER clause is set to "YES", MRFCleanFactory will remove dangles if their lengths are less than the  $(DANGLEFACTOR * TOLERANCE)$  or  $(DANGLEFACTOR * \text{value of } TOLERANCE\_ATTR)$ . Again the latter is always used whenever possible. The default value of DANGLEFACTOR is 1.0.

See below for a description of the other types of output clauses that are supported.

## **MRFCleaner: General Processing Tips**

There are several fundamental tips and tricks for optimizing processing. The following list identifies and briefly describes some key issues to consider when cleaning data.

### **Know your data**

The first step in any cleaning exercise is to become familiar with your source data. Information on data quality (i.e., 1m vs. 100m accuracy), data currency, and intended use is important in determining which cleaning modules and tolerances should be used. If such information is not available, a visual inspection of the design file(s) should provide insight into average line-work gap sizes, line weeding requirements, and other issues which may exist.

### **Start small**

When setting cleaning tolerances, it is always best to start small. With smaller tolerances, the software uses a smaller search radius, which reduces the number of potential element intersections to consider and increases processing speed. Also, if the bulk of the linework errors can be corrected using a small tolerance, more detail can be maintained in the dataset. One or more cleaning processes can always be repeated with larger tolerances to increase the number of errors automatically corrected.

### **Mix it up**

Depending on the source dataset, and its intended use, you may achieve better results running the individual modules with different tolerances.

## MRFCleaner Sample Results

The MRFCleanFactory uses generic algorithms to perform data cleaning and does not follow a set of predefined cases. The best way to learn the behavior of MRFCleanFactory is to construct test cases.

The following diagrams illustrate the cleaning principles. Each element is assumed to be on a unique level unless otherwise stated. Tolerances are shown at the top of each figure.

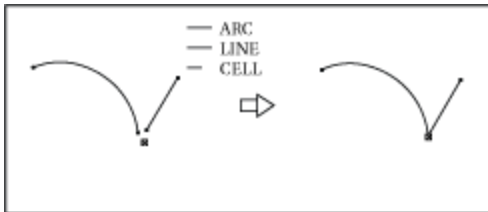


FIGURE 8 Arc and line nodes moved to cell origin

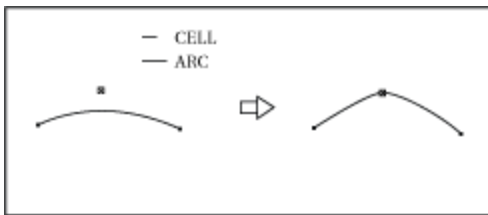


FIGURE 9 Arc broken and ends collocated to cell origin

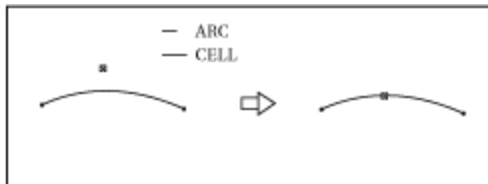


FIGURE 10 Cell moved to arc. Arc is broken.



FIGURE 11 Redundant vertices removed by line weeding.



FIGURE 12 Line with larger tolerance moved to line with smaller tolerance

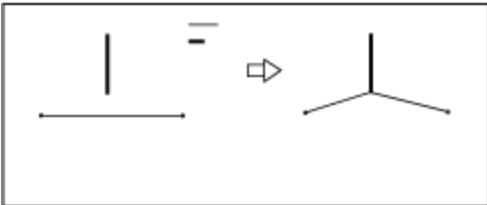


FIGURE 13 Fuzzy intersection created in linear element with larger tolerance



FIGURE 14 Fuzzy intersection created in linear element with smaller tolerance

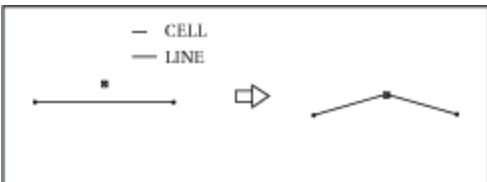


FIGURE 15 Fuzzy intersection created in linear element with larger tolerance

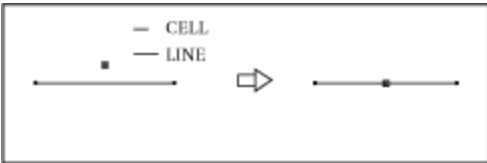


FIGURE 16 Cell with large tolerance moved to fuzzy intersection in linear element

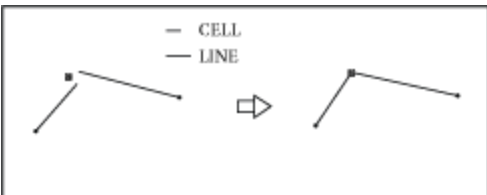


FIGURE 17 Linear elements with large tolerance moved to collocate with cell origin with smaller tolerance

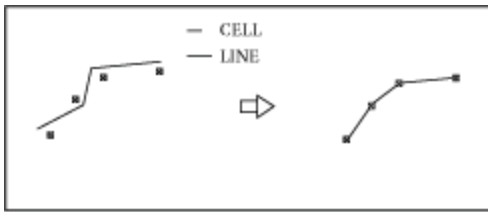


FIGURE 18 Nodes and vertices on linear element with large tolerance collocated at cell origin with smaller tolerances



## NeighborColorSetter

Assigns colors to areas in a coverage such that adjacent areas are colored differently, and the total number of colors used is kept small.

An ID and a list of neighbor IDs may be provided for each area. If these are not provided, adjacencies between areas are determined geometrically, and non-polygon geometries (including aggregates and ellipses) are simply annihilated. IDs must be non-negative integers, and neighbor IDs are provided as a comma-separated list.

Colors are output to the Color ID Attribute as integers (the first color is 0, the second is 1, etc.).

If **Set Pen and Fill Colors** is Yes, areas will be assigned colors based on a default color scheme. Alternatively, the Color ID attribute may be manipulated to provide colors for the regions (e.g., by using a PenColorSetter and AttributeFilter followed by a set of AreaColorSetters).

When the **Coloring Algorithm** is set to Simple, each area is colored with the first available color. Ideally, only a few colors will be used, but the total number of colors is not guaranteed. When the Coloring Algorithm is set to Five Color, at most five colors will be used to color all of the regions.

## Transformer Category

Geometric Operators

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: NeighborColorSetterFactory

## NeighborFinder

Finds the closest CANDIDATE feature within a specified maximum distance of each BASE feature.

### Usage Notes

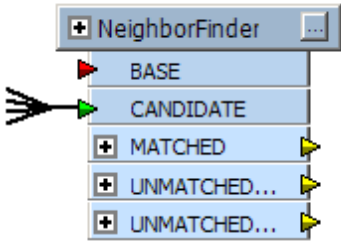
This transformer works with 2D geometries only; if an input geometry is 3D, its z-coordinate will be ignored.

This transformer has full support for points, lines, arcs, ellipses, polygons, and donuts, and has limited support for other types of geometry. Polygons, ellipses and donuts may be processed as lines or areas, depending on user selection.

### Input Ports

If a feature is routed to both the BASE and the CANDIDATE input ports, then features will be compared to themselves as they are both a BASE and CANDIDATE.

If there is only a connection made to the CANDIDATE port, but not to the BASE port (as shown below), then all CANDIDATEs will be compared with all other CANDIDATEs, but will not be compared to themselves.



### Output Ports

If there are no CANDIDATE features found to be within the maximum distance, then the BASE feature will be output unchanged via the UNMATCHED\_BASE port.

CANDIDATE features not close enough to any BASE feature are output via the UNMATCHED\_CANDIDATE port.

If a CANDIDATE feature is found, then all the attributes from the closest CANDIDATE feature are added to the BASE feature and the BASE feature is output via the MATCHED port. In addition, several other attributes are added to the BASE feature just prior to it being output via the MATCHED port:

Attributes	Description
_distance	The distance from the BASE to the matching CANDIDATE
_angle	The angle between the closest interpolated BASE point and the closest interpolated CANDIDATE point.
_closest_base_x, _closest_base_y	The coordinates of the closest interpolated point on the BASE feature relative to the CANDIDATE feature.
_closest_candidate_x, _closest_candidate_y	The coordinates of the closest interpolated point on the CANDIDATE feature relative to the BASE feature.
_candidate_angle	The angle from (_closest_candidate_x, _closest_candidate_y) to the next vertex within the CANDIDATE feature. If (_closest_candidate_x, _closest_candidate_y) equals the last vertex of the CANDIDATE feature, then candidate_angle will be the angle from the previous vertex of the CANDIDATE feature to (_closest_candidate_x, _closest_candidate_y).
_candidate_label_angle	The _candidate_angle adjusted so that if it is used as a text rotation, the text will run from left to right. This angle is guaranteed to be greater than or equal to 270 and less than 360 or greater than or equal to 0 and less than or equal to 90.

### Parameters

Maximum Distance

The maximum distance is measured in the units of coordinates of the input features.

The list contains all of the candidate features that were within the maximum distance of the base.

To get the distance from a given BASE to all CANDIDATE features, use a very large number for this parameter and specify a Close Candidate List Name.

#### Insert Vertex on BASE Feature

If Insert Vertex on BASE feature is Yes, then (`_closest_base_x`, `_closest_base_y`) will be inserted onto the BASE feature if the insertion is well-defined. For example, if a CANDIDATE polygon is contained inside a BASE polygon, insertion will not take place.

If Insert vertex on BASE feature is Yes, the `_closest_base_x`, `_closest_base_y` vertex will be inserted onto to the BASE feature as well as added as an attribute. This option only applies to Lines, Polygons, Paths, Arcs, Ellipses, and Donuts.

#### Candidates First

If set to Yes, then all CANDIDATE features must be input before any BASE features. If a CANDIDATE feature is input after a BASE feature and this option is set to Yes, the CANDIDATE feature will be ignored in all calculations.

#### Close Candidate List Name

If specified, a list will be built on the MATCHED output, consisting of all the attributes from the CANDIDATE features that were within maximum distance of the BASE feature.

#### Tested Candidate List Name

If specified, a list will be built on the MATCHED output, consisting of all the attributes from the CANDIDATE features that were within maximum distance of the bounding box of the BASE feature.

#### Treat Polygons As

- **Lines:** A polygon, donut, or ellipse will be treated as a line (that is, its boundary line) for backwards compatibility.
- **Areas:** A polygon, donut, or ellipse will be treated as an area, and any geometry that overlaps with the area will be of 0 distance away from that area.

#### Geometry Handling

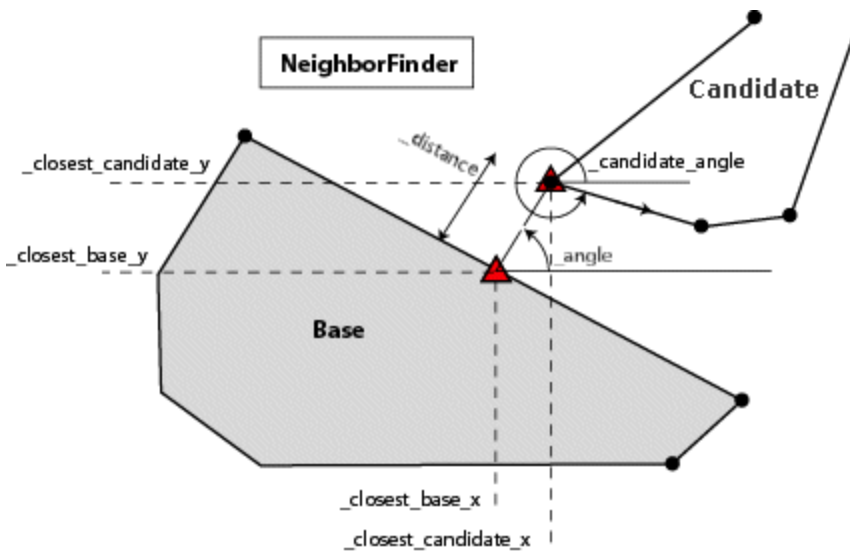
If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will have their boundaries considered in distance calculations; otherwise, their centerpoints will be used.

#### Transformer Category

Collectors

#### Example

All angles are measured in degrees counterclockwise from horizontal. Where angles are not well-defined (for example, when a CANDIDATE polygon is contained inside a BASE polygon), they are set to 0.



### FME Licensing Level

FME Professional edition and above

### fmepedia

See the NeighborFinder page in fmepedia for additional information and examples that use this transformer.

### Technical History

Associated FME function or factory: ProximityFactory

## NeighborhoodAggregator

Creates aggregates of features based on their proximity to each other. Each aggregate that is created covers approximately the neighborhood width and height (measured in feature ground units).

This transformer is used to reduce the data volume of "wallpaper" types of features that have no individual attributes. The resulting aggregates can be output to a system using many fewer records than if each feature was output by itself. For systems that support aggregates, or multi-part features, this can result in substantial performance improvements and greatly decrease storage requirements.

### Input Port

INPUT

### Output Port

NEIGHBORS

### Parameters

#### Group By

Features that leave this transformer will have only the group-by attributes present on them. Any other feature attributes are lost.

#### Neighborhood Width and Neighborhood Height

These parameters, measured in ground units, divide the input space into cells. The center of the bounding box of each input feature is used to determine the cell for the feature. Once all input features have been read, an aggregate feature is created from all features in each cell. If linear features are input, they will have pseudo nodes removed from within their cells to further reduce the number of separate entities. No such reduction is done to any polygons or donuts that enter.

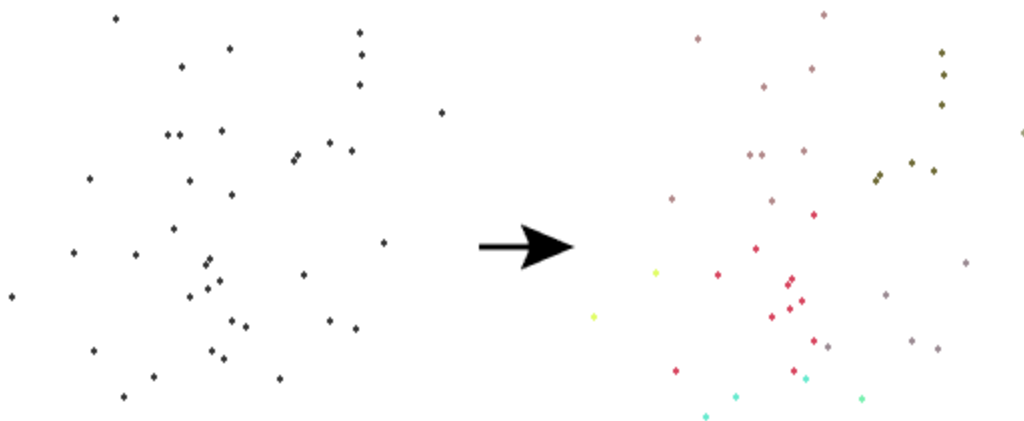
#### Minimum Neighborhood Members

When you set this parameter, neighborhoods with fewer than the specified number of features are merged with a vertical neighbor area in order to increase the number of members. You can prevent this from happening by setting the parameter to 0 (zero).

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, then arcs and ellipses will be stroked to lines and polygons (respectively) before undergoing the transformation; otherwise, only their center points will be transformed.

### Example



### Transformer Category

Collectors

### Technical History

Associated FME function or factory: TilingFactory



## NeighborPairFinder

Finds the closest two CANDIDATE features within some maximum distance of each BASE feature and some minimum separation in heading between the CANDIDATES and the BASE. The maximum distance is measured in the units of coordinates of the input features. The minimum separation angle is measured in degrees, and specifies the minimum difference in heading from the base that the two candidates must have before the second one will be used.

If none, or only one, CANDIDATE feature is found to be within the maximum distance, then the BASE feature will be output unchanged via the UNMATCHED\_BASE port.

If two CANDIDATES are found, then the BASE feature is output via the MATCHED port. In this case, the following attributes will be added to the BASE feature:

- **\_distance1, \_distance2** – The distance (in ground units) from the BASE to the matching CANDIDATE
- **\_heading1, \_heading2** – The angle between the closest interpolated BASE point and the closest interpolated CANDIDATE point.
- **\_closest\_base\_x1, \_closest\_base\_y1, \_closest\_base\_x2, \_closest\_base\_y2** – The coordinates of the closest interpolated point on the BASE feature to the closest interpolate point on the CANDIDATE feature.
- **\_closest\_candidate\_x1, \_closest\_candidate\_y1, \_closest\_candidate\_x2, \_closest\_candidate\_y2** – The coordinates of the closest interpolated point on the CANDIDATE feature to the closest interpolate point on the BASE feature.
- **\_candidate\_angle1, \_candidate\_angle2** – The angle from the closest interpolated point on the CANDIDATE feature to the next vertex within the CANDIDATE feature. (If the closest interpolated point on the CANDIDATE feature is its last vertex, then candidate\_angle will contain the angle from the previous vertex of the candidate feature to the closest interpolated point on the CANDIDATE feature.)
- **\_candidate\_label\_angle1, \_candidate\_label\_angle2** – The \_candidate\_angle adjusted so that if it is used as a text rotation, the text will run from left to right. This angle is guaranteed to be greater than or equal to 270 and less than 360, or greater than or equal to 0 and less than or equal to 90.

(The attributes ending in 1 relate to the closest CANDIDATE feature found. The attributes ending in 2 relate to the next closest CANDIDATE feature found, which has a sufficiently different heading.)

All headings are measured in degrees counterclockwise from horizontal. All distances are measured in the ground units of the features.

The CLOSEST\_VECTOR and SECOND\_CLOSEST\_VECTOR ports will have linear features output on them that connect the closest points on the base and candidate features. These ports are useful only for visualizing where the closest points were found.

The Candidate Key Attribute field will be used to select a CANDIDATE attribute from the two closest CANDIDATES that will be preserved on the BASE feature as \_key attributes.

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs and ellipses will have their boundaries considered in distance calculations; otherwise, their centerpoints will be used.

## Transformer Category

Collectors

## Technical History

Associated FME function or factory: ProximityFactory, @Tcl2

## NetworkCostCalculator

Computes and assigns the cost of the shortest path from a source object to each connected object as the Z-values or measure values of the input features.

### Output Ports

- **CONNECTED:** All lines that are connected to the SOURCE input port are output through the CONNECTED port.
- **DISCONNECTED:** The lines that are not connected to the SOURCE input port will be output through the DISCONNECTED port. If Output Optimal Cost As parameter is set to Z-Values, the dimension of the disconnected lines is set to 2D. Otherwise, disconnected lines are untouched.
- **INVALID:** There can only be one SOURCE input for each group. All other inputs and non-linear features are output through the INVALID port.

### Parameters

#### Group By

Choose the attributes to group by.

Weight Type, Forward Weight Attribute, Reverse Weight Attribute

If Weight Type is set to By Length (Forward Only) or By One Attribute, then the weight of each input line is set to the length of the line or the attribute value specified in the Forward Weight Attribute. In this case, the algorithm will only consider the original orientation of the lines when computing the cost of the shortest path.

If Weight Type is set to By Length or By Two Attributes, then the shortest path algorithm will consider both directions of the input lines. If Weight Type is set to By Two Attributes, the original orientation of the input line has the weight specified in the Forward Weight Attribute and the reversed orientation of the input line has the weight specified in the Reverse Weight Attribute. If Weight Type is set to By Length, the weight of both the original orientation and the reversed orientation of the input line is set to the length of the line.

Only linear features with non-negative weight attribute values are allowed if the Weight Type is set to By One Attribute or By Two Attributes. If a feature does not have the attribute specified in the Forward Weight Attribute or the Reverse Weight Attribute, a zero weight is used for the line.

#### Output Optimal Cost As

If this parameter is set to Z-Values, then the optimal cost for each connected node is set as the Z-value of the node. Otherwise, the optimal cost is set as the measure value of the node with the measure name specified in Measure Name.

#### Measure Name

If you leave the Measure Name empty, the default measure name will be used.

### Technical History

FME Factory Used: NetworkFactory



## NetworkFlowOrientor

Fixes the flow (direction) of each edge or linear feature in the network to fit the downstream direction to the destination node.

### Input

- LINE: Network lines
- DESTINATION: Only one destination node is allowed for each group.

### Output

- NETWORK: All connected features are output through the NETWORK port.
- UNUSED: All edges or linear features not connected to the destination node are output through the UNUSED port.
- INVALID: All non-linear features or extra destination nodes are output through the INVALID port.

### Parameters

Group By

The default behavior is to use the entire set of features as the group. This option allows you to select attributes that define which groups to form.

Direction Attribute

If an edge is reversed, the value of the attribute specified in Direction Attribute will be *opposite*. Otherwise, the value will be *same*.

### Usage Notes

The functionality works well in a non-cycle network; however, it does not always yield desired results in a cycle network. One way to fix a cycle network is to omit edges that should not be modified from the input.

### Transformer Category

Network

### Related Transformers

NetworkTopologyCalculator

ShortestPathFinder

StreamOrderCalculator

StreamPriorityCalculator

### FME Licensing Level

FME Professional edition and above

### Technical History

FME Factory Used: NetworkFactory

## **NetworkTopologyCalculator**

Finds the connected lines that belong to the same network graph.

### **Input**

- LINE: Network lines

### **Output**

- NETWORK: All connected lines are output through the NETWORK port.
- INVALID: All non-linear features are output through the INVALID port.

### **Parameters**

Group By

The default behavior is to use the entire set of features as the group. This option allows you to select attributes that define which groups to form.

Network ID Attribute

Linear input features that are connected will be assigned the same network ID in the Network ID Attribute.

### **Transformer Category**

Geometric Operators

### **Related Transformers**

NetworkFlowOrientor

ShortestPathFinder

StreamOrderCalculator

StreamPriorityCalculator

### **FME Licensing Level**

FME Professional edition and above

### **Transformer History**

This transformer has been renamed from NetworkTopologyBuilder.

### **Technical History**

FME Factory Used: NetworkFactory

## **NullAttributeReplacer**

Checks all the selected attributes and sets them to the value given in the Default Value parameter if they were null (if they had no value).

This transformer is useful when an output format requires non-null attributes.

### **Transformer Category**

Strings

### **Technical History**

Associated FME function or factory: @Tcl2

## **NumericRasterizer**

Draws input point, line and polygon features onto a numeric raster filled with the background value. The Z coordinates of the input vector features are used to generate pixel values. Features without Z coordinates will be discarded.

### **Parameters**

#### Group By

If the Group By parameter is set to an attribute list, one raster per group will be produced.

#### ***Raster Properties***

Size Specification, Number of Columns (cells), Number of Rows (cells), X Cell Spacing, Y Cell Spacing

To set the size of the output raster, specify either the dimensions or the cell size.

To set the output raster size using dimensions, set the Size Specification to *RowsColumns* and specify values for both the Number of Columns and Number of Rows.

To set the output raster size using cell size, set the Size Specification to *CellSize* and specify values for both the X Cell Spacing and Y Cell Spacing.

#### ***Interpretation***

#### Interpretation Type

Sets the interpretation of the output raster.

#### ***Background***

#### Background Value

Sets the background value of the raster.

#### Fill Background with Nodata

If this is set to Yes, the background value will also be flagged as the nodata value for the produced band.

#### ***Anti-Aliasing***

#### Anti-Aliasing

If the Anti-Aliasing parameter is Yes, the output lines will be smoothed using an anti-aliasing algorithm.

#### Tolerance

This parameter is the maximum normalized distance from a line segment or polygon vertex to a pixel to be rendered. For example a tolerance of 1.0 will draw all pixels touched by the input vector line, while a tolerance of 0.0 will draw only those pixels where the input vector line passes directly through their center. Tolerance can only be selected when anti-aliasing is off.

#### ***Ground Extents***

#### Ground Extents

If this parameter is set to *Use input data ground extents*, which means the extents are not explicitly specified, the output raster extents will be determined by the union of the bounding boxes of the valid input vector features.

If this parameter is set to *Specify ground extents*, the remaining Ground Extents parameters are used to specify the extents of the output raster.

### **Transformer Category**

Rasters

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: VectorToRasterFactory

## Offsetter

Adds offsets to the feature's coordinates.

You can enter parameter values as a number, or pick the value of a feature attribute by selecting the attribute name from the pull-down list.

The feature will be shifted by the amounts specified.

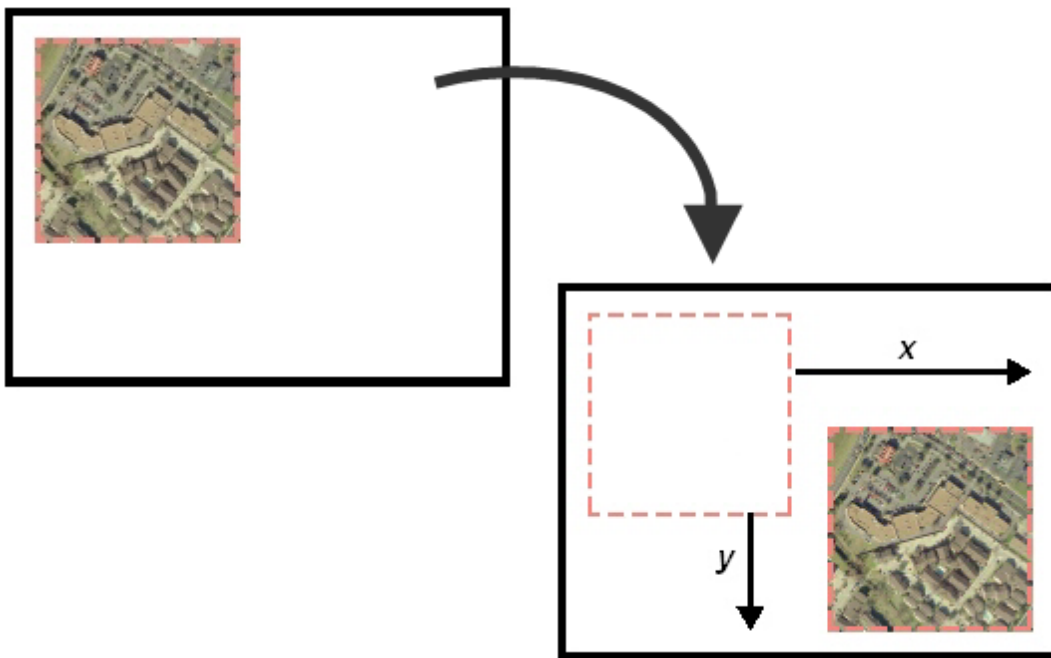
## Usage Notes

This transformer works with both raster and vector data.

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

## Examples

This example shows an image offset using the  $x$  and  $y$  parameters:



This example shows an image offset using the  $x$  and  $y$  parameters:

## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: @Offset

## OracleQuerier

Performs spatial queries against an Oracle Spatial database. The queries can have both a spatial component as well as nonspatial component.

One query is issued to the Oracle database for each feature that enters the transformer. The results of the query are then output.

### Feature Types to Read

Feature types can either be specified from the wizard table list or "Table Name in Attribute" from the navigator pane. When listed in the navigator pane, multiple tables can be specified in a colon-separated list.

### Query Operation

The query feature defines the geometry that will be used to define the spatial component of the query, unless it does not contain any geometry. In this case, only an attribute query as defined by the WHERE clause will be executed.

In the case where a spatial query is being performed, the OracleQuerier defaults to using the bounding box of the query feature as its search envelope. It is possible to tell it to use the actual geometry of the query feature to perform this search, by selecting a value of "no" for the choice **Use Bounding Box For Spatial Query**. Note, however, that Oracle sometimes does not return expected results when performing a query defined by a non-rectangular search geometry.

### Spatial Query Operators

The complete set of Oracle spatial query operators is supported and each is described below:

- **CONTAINS:** Features output have geometry that entirely contains the query feature's bounding box.
- **COVERS:** Features output have geometry that entirely contains the query feature's bounding box and whose boundaries intersect the ones of the query feature's bounding box.
- **COVEREDBY:** Features output have their geometry entirely contained in the query feature's bounding box and whose boundaries intersect the ones of the query feature's bounding box.
- **ANYINTERACT:** Features output interact with the query feature's bounding box in any of the above ways.
- **DISJOINT:** Features output have no common boundary or interior points with the query feature's bounding box.
- **EQUAL:** Features output share every point of their boundaries and interior, including any hole, with the query feature's bounding box.
- **INSIDE:** Features output have their geometry entirely contained in the query feature's bounding box.
- **OVERLAPBDYDISJOINT:** Features output overlap but their boundaries do not interact with the query feature's bounding box. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- **OVERLAPBDYINTERSECT:** Features output overlap and their boundaries intersect in one or more places with the query feature's bounding box.
- **TOUCH:** Features output share a common boundary with the query feature's bounding box.
- **NONE:** Perform a search envelope query defined by the geometry of the query feature. If the incoming feature does not contain geometry, an attribute-only query is performed.

### Attribute Handling Parameter Settings

- **Result Attributes Only:** The result feature attributes are based solely on the query results.
- **Keep Query Attributes if Conflict:** The result feature attributes are a combination of both the query results and the query feature's attributes. If there is a conflict, attribute values are taken from the query feature.
- **Keep Result Attributes if Conflict:** The result feature attributes are a combination of both the query results and query feature's attributes. If there is a conflict, attribute values taken from the query results.

### Geometry Handling Parameter Settings

- **Result Geometry Only:** The result feature geometry is taken from the query results.
- **Query Geometry Only:** The result feature geometry is taken from the query feature.

- **Aggregate Query and Result Geometry:** The result feature geometry is an aggregate of the geometry from the query feature followed by the geometry from the query results.

### **Transformer Category**

Database

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: QueryFactory



## **OrientationExtractor**

Category: Calculators

Determines the feature's orientation and returns it in the specified Orientation Attribute. If the feature has area geometry which fits the right-hand or left-hand rule, the attribute will be set to the value `right_hand_rule` or `left_hand_rule`, respectively. Otherwise the attribute will be set to `no_orientation`.

The right-hand rule requires that, on a walk of the area's coordinates, the area is on the right-hand side. Thus, outer boundaries must be clockwise and inner boundaries must be counterclockwise. The opposite is true for the left-hand rule.

Note that the orientation will always be `no_orientation` for non-area features.

### **Technical History**

Associated FME function or factory: @Orient

## Orienter

Adjusts the orientation of a polygonal feature or the direction of a linear feature.

### Parameters

Orientation Type

If the feature is an area, a **RIGHT\_HAND\_RULE** orientation causes the outer boundary to have its vertices arranged in a clockwise direction, and the holes to have their vertices in a counterclockwise direction.

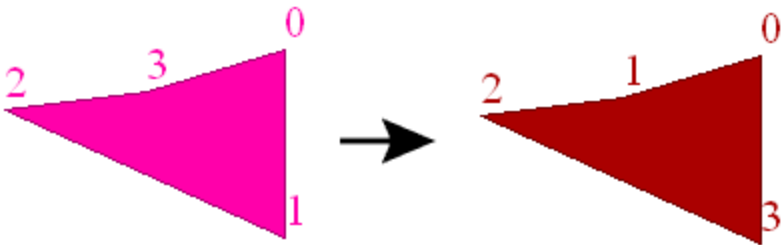
The opposite is true when **LEFT\_HAND\_RULE** is specified. When the **LEFT\_HAND\_RULE** is specified, the feature's outer boundary vertices are in a counterclockwise order and the holes have a clockwise ordering.

When **REVERSE** is specified, the feature's coordinates are flipped so that the first coordinate becomes the last one, and vice-versa. This option is intended for use with features that are not polygonal. This transformer has no effect on solid geometries. Only the **REVERSE** option is available for surface geometries.

Orientation Flag Attribute

If this attribute is specified, then the attribute will be added to the resulting features and its value is set to Yes if the orientation of the geometry has been changed. Otherwise, the value of the attribute is set to No. An aggregate or multi geometry has its orientation changed if the orientation of at least one of its members is changed.

### Example



### Transformer Category

Manipulators

### Technical History

Associated FME function or factory: @Orient

## ParameterFetcher

Adds an attribute to the feature, supplying it the value of a previously published parameter.

A parameter is a setting that you can pass values into when a translation is run. Parameters usually relate to a setting on a reader or transformer. The ParameterFetcher can be used to copy the value of one or more parameters to a corresponding number of attributes. The number of rows specified in the configuration dialog determine the number of exposed output attributes.

## Parameters

### Parameter Name

When the ParameterFetcher is connected, the Parameter Name drop-down list shows the parameters that can be fetched from the workspace. The Parameter Name also exposes these FME Server parameters:

FME\_SECURITY\_ROLES  
FME\_DATA\_REPOSITORY  
FME\_SERVER\_NAME

These parameters may not be present in the workspace but they are applicable to the FME Server environment if the workspace is published to FME Server.

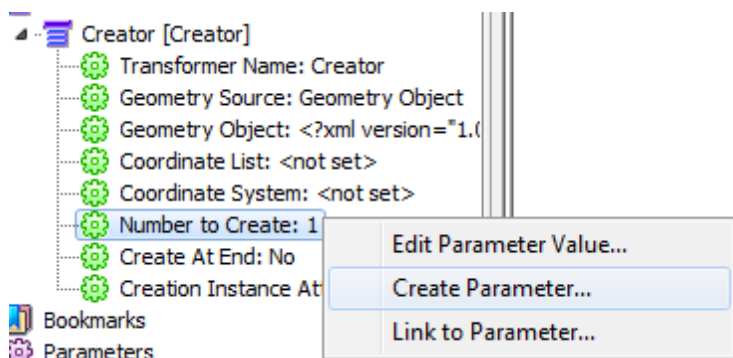
### Target Attribute

The target attribute is the name of the attribute that will be created. When you select a parameter, this field is automatically filled with an appropriate name (for example, selecting a parameter named **WIDTH** will automatically name the target attribute **\_WIDTH**). You can use the default name, or type a new name.

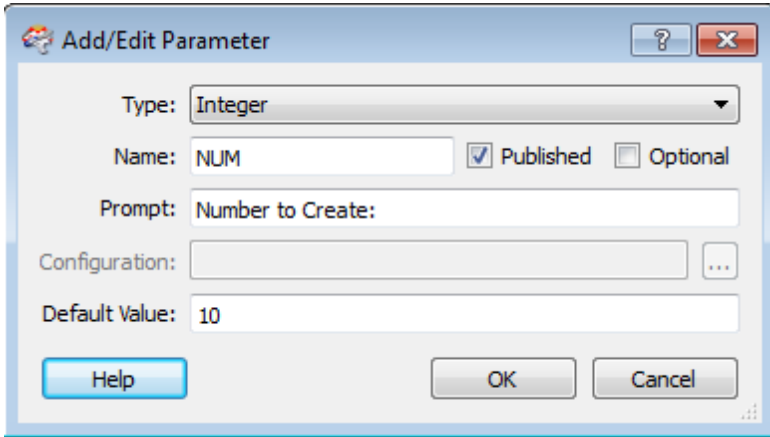
## Example

This example describes a single parameter but you can fetch multiple parameters.

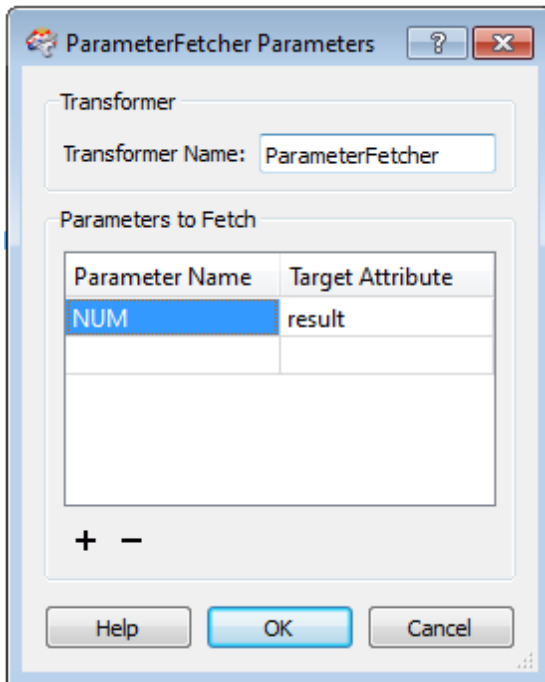
The Creator transformer shown below has a published parameter called *NUM*. To publish this parameter, place the Creator transformer in the workspace. Then right-click on Number to Create and select Create Parameter.



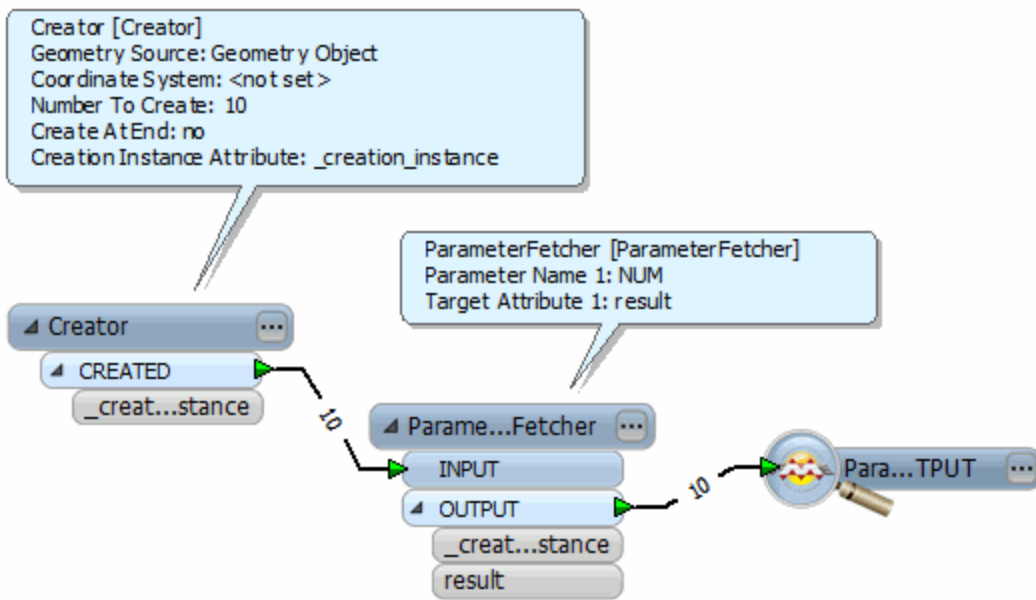
The dialog shows the Published Parameter properties, with the default value changed to 10:



Choose the *NUM* parameter in the Parameter Name field, and type a Target Attribute *result*.



When the workspace is run, the ParameterFetcher fetches the value assigned to the "NUM" parameter and stores it in the "result" attribute. Now the feature has an attribute "result" which holds the value of the "NUM" parameter.



You can see the attribute and its value in the FME Universal Viewer.

Feature:	1 of 10
Feature Type:	ParameterFetcher_OUTPUT
Coord Sys:	Unknown
Attribute Name	Attribute Value
_test_attrib	0
fme_geometry	fme_undefined
fme_type	fme_no_geom
result	10

For more information on Published Parameters, see the Workbench help topic *Adding a Parameter*.

## Transformer Category

Infrastructure

## Technical History

Associated FME function or factory: @SupplyAttributes

## **PartCounter**

Category: Manipulators

Return the number of parts in the geometry. For multis and aggregates, this is the number of parts, and for paths, this is the number of segments. Otherwise, it is one. The **Count Aggregates Recursively** field specifies whether aggregates' parts should have their parts counted.

## **Technical History**

Associated FME function or factory: @Geometry

## PathBuilder

Connects input linear (arcs and lines) features in the order they enter, forming path features.

Note: For some datasets, it may be necessary to use a Sorter to order the data correctly before it enters this transformer.

If the end point of one input segment does not match the following segment's start point, a two-point line will be inserted between these segments in the resulting path.

The feature being created is output whenever a **Connection Break Attribute** value changes. When this happens, the feature with the differing attribute is not added to the current output feature; instead, it begins the next feature to be output.

If consecutive input segments are lines, they will not be combined into a single line in the output path; they will remain as separate segments. Also, if only one segment is input for a set of Connection Break Attribute values, the output will still be a path (containing that single segment).

If the optional **List Name** is supplied, a list is made of all the attributes of each feature that was connected when creating the output path.

### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: ConnectionFactory

## **PathSplitter**

Decomposes a path feature into its component segments.

Each output feature will contain a copy of the source feature's attributes. Non-path input features will be output untouched.

The optional List Attribute is used to associate attributes with each segment of the path. If the feature has a list of attributes that are in parallel to the members of the aggregate, these can be decomposed onto the output features by specifying the List Attribute parameter.

If the Segment Number Attribute is specified, then each output feature is given an attribute containing the part's segment index within the original path feature.

Note that for an input feature to be affected, its geometry itself must be a path; that is, areas whose boundaries are paths and aggregates containing paths will be unaffected. The GeometryCoercer, DonutHoleExtractor, and Deagggregator may be used to prepare the geometries input to this transformer.

## **Transformer Category**

Manipulators

## **Technical History**

Associated FME function or factory: DeaggregateFactory



## PDFStyler

Sets the common PDF style attributes for a group of features destined for the GeoSpatial PDF Writer.

### Parameters

#### *Identity*

##### Name

Determines the name of the structure element associated with the incoming feature. Name must be set to a unique value per feature thus it is recommended this be set to an attribute containing unique string values.

##### Tooltip

Specifies the string that will be displayed when a user hovers over the feature with the mouse cursor in a PDF viewer application.

##### URL

Specifies the destination address to be resolved when a user clicks on the feature in a PDF viewer application that supports URI actions. In the common case of a URL, Adobe Acrobat Reader will open a web browser to resolve the address specified.

#### *Color*

##### Color

Specifies the pen color of the feature used to render the feature in the set color. The pen color determines the color of points, lines, arcs, area boundaries, and annotations.

Click the colored square to the right of the text field, edit the contents of the field directly. The color must be specified as r,g,b where each of r,g,b is a decimal number between 0.0 and 1.0.

##### Fill Color

Specifies the fill color for an area geometry on a feature used to render the interior feature in the set color.

The Fill Color parameter can be edited by clicking the colored square to the right of the text field, or by editing the contents of the field directly. The color must be specified as r,g,b where each of r,g,b is a decimal number between 0.0 and 1.0.

##### Opacity

Specifies the opacity of the stroking color of the feature. A value of 1.0 is fully opaque, and 0.0 is completely transparent.

##### Fill Opacity

Specifies the opacity of the fill color of the feature. A value of 1.0 is fully opaque, and 0.0 is completely transparent.

#### *Line Style*

##### Line Width

Specifies the line width in typographical points of line geometries and boundaries of area geometries. Points within half the line width perpendicular distance from the line path will be painted.

##### Line Cap Style

Specifies the cap style for the ends of lines. Valid values are:

- Butt Cap: Lines are squared off at the end and do not project past the end of the line,
- Round Cap: Semicircles with diameter equal to the line width cap the ends of lines, or
- Projecting Square Cap: Lines project past the end by a distance equal to half the line width and are squared off.

##### Line Join Style

Specifies the shape of corners between segments of paths. Valid values are:

- Miter Join: Outer edges of segments are extended until they meet,
- Round Join: Arcs with diameter equal to the line width are drawn around corners, or
- Bevel Join: Two adjacent segments are finished with butt caps, and the notch beyond the ends is filled with a triangle.

#### Line Miter Limit

Specifies a maximum on the ratio of the miter length to the line width. For example, a miter limit of 1.414 will bevel the ends of two segments meeting at an angle less than 90 degrees (the far corner will be at a distance  $\sqrt{1^2+1^2}=\sqrt{2}$  from the line).

#### Line Dash Array/Line Dash Phase

The Line Dash Array and Line Dash Phase control the pattern of dashes and gaps used to stroke lines. The elements of both the dash array and the dash phase are expressed in user space units. An empty dash array and zero phase can be used to draw a solid line.

The Line Dash Array's elements are numbers that specify the lengths of alternating dashes and gaps. The numbers must be nonnegative and not all zero. The Line Dash Phase specifies the distance into the dash pattern at which to start the dash.

When drawing paths consisting of several subpaths, the dash pattern is applied independently to each subpath.

### **Text**

#### Font

Refers to any valid TrueType font name, which you can choose from the dialog or enter directly in the parameter field. The font name is case-insensitive, and can also include optional style parameters according to the syntax:

`<fontname>[,<fontsize>][,BOLD][,ITALIC][,STRIKEOUT][,UNDERLINE]`

The [ ] enclose optional items.

#### **Usage Notes**

For more information regarding PDF styling, see the "GeoSpatial PDF Writer" in the *FME Readers and Writers* manual.

#### **Transformer Category**

Manipulators

## **PenColorSetter**

Sets the pen color of the feature.

Formats that support color will then render the feature in the set color. The pen color determines the color of points, lines, arcs, area boundaries, and annotations (labels).

### **Parameters**

Pen Color

Click the colored square to the right of the text field and choose from the color picker, or type the rgb value.

The color is formatted as  $r,g,b$ , where each of  $r$ ,  $g$ , and  $b$  is a number between 0 and 1.

### **Transformer Category**

Infrastructure

## PlanarityFilter

Filters features based on their planarity. To be planar, a geometry must have all its points situated in the same plane.

### Input

- INPUT: Typically area-based or surface-based geometries.

### Output

- PLANAR: If an area-based or surface-based geometry has all of its points situated in the same plane, its features are sent to the PLANAR output port.
- NOT\_PLANAR: If an area-based or surface-based geometry is not planar, its features are written to the NOT\_PLANAR output port.
- UNDEFINED: All features with geometry types other than areas, multi-areas, surfaces, multi-surfaces, or composite surfaces will be sent to the UNDEFINED output port.

### Parameters

If an Explicit Plane Normal is not specified, the normal vector is determined by the vertices of this geometry using Newell's method. The normal vector is then normalized so that it is a unit vector ((A, B, C) of length 1.

The planarity condition is computed by the following algorithm.

For the first point (x, y, z) of this geometry, we compute  $D' = Ax + By + Cz$ .

Then this geometry is planar if, and only if, every subsequent point (x, y, z) of the geometry gives a  $D = Ax + By + Cz$  that is within the tolerance amount of  $D'$ .

That is,  $|D - D'| \leq \text{tolerance}$ .

### Tolerance

The tolerance is the maximum distance that any point can be displaced perpendicularly away from the plane and still be considered in plane. Intuitively it is the maximum height of "peaks" and the depth of "valleys" tolerated in the geometry.

For example, a Tolerance of 0 indicates that each point (x, y, z) must be exactly on the plane for the geometry to be planar, whereas a Tolerance of 0.7 indicates that as long as points are no more than 0.7 above or below the plane, the geometry is considered planar.

### Explicit Plane Normal

By default, the Explicit Plane Normal field is set to No, which means that the coordinates must lie in the same plane, but that may be any plane. When this parameter is set to No, the fields for Normal X, Normal Y, and Normal Z are inaccessible.

If Explicit Plane Normal is set to Yes, then all coordinates must lie in a plane with the given normal vector and you must specify the Normal x, y, z parameters.

### Normal X, Y, Z

Sets the x, y, and z components of the vector.

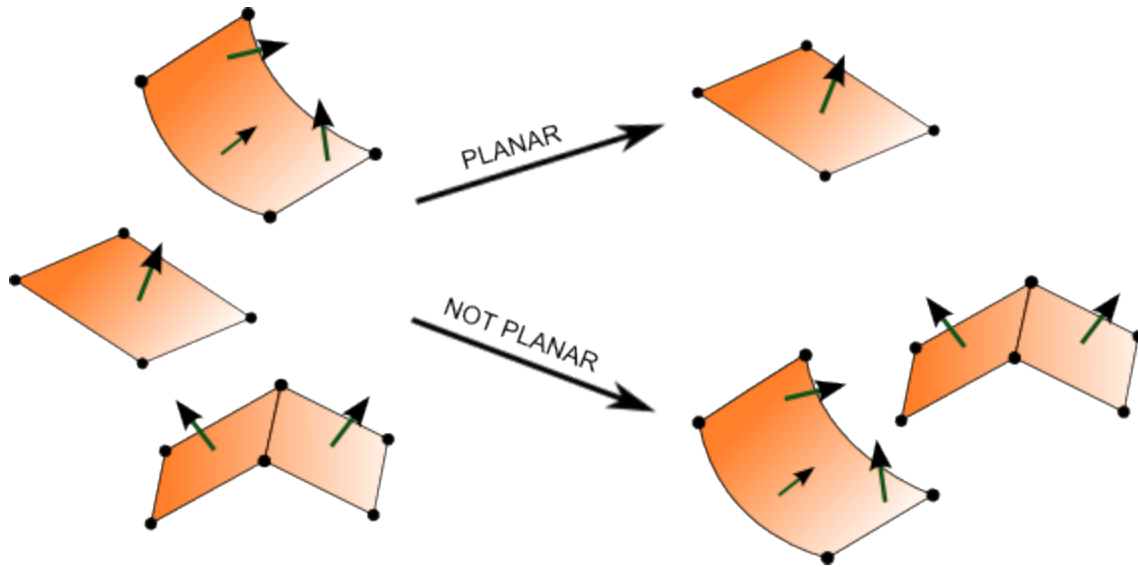
### Usage Notes

This transformer works only on the following geometries:

- areas (polygons, ellipses, and donuts)
- multi-areas
- surfaces
- multi-surfaces
- composite surfaces

## Example

These three examples illustrate one planar surface (all points are situated in the same plane), and two non-planar surfaces.



## Transformer Category

Filters

## Related Transformers

FaceReplacer

## Technical History

Associated FME function or factory: @Geometry

## **Player**

Retrieves features stored in an FME Feature Store file and outputs them into the workspace.

FME Feature Store files can be created with the Recorder transformer, by the RecorderFactory in FME, or with the FME Feature Store writer module in FME.

The optional password must be provided if the FME Feature Store file being read was produced with a password. Before the file can be read, the password must match that used to create the file.

## **Transformer Category**

Infrastructure

## **Related Transformers**

Recorder

## **Technical History**

Associated FME function or factory: RecorderFactory

## PointCloudCoercer

Decomposes all point clouds into points. This transformer is used when writing to formats that do not support point clouds.

### Input Ports

- INPUT: This transformer accepts only point-cloud features.

### Output Ports

- OUTPUT: Individual points, chunked multipoints or single multipoint features based on the selected Output Geometry

### Parameters

Output Geometry

#### *To be added*

- Chunked MultiPoint:
- Single MultiPoint: This option could result in memory errors if the point cloud is very large
- Point:

Points Per Chunk

Specifies the maximum number of points in the output multipoint.

Point Components to Preserve

#### *To be added*

Preserve Attributes

If set to Yes (default), attributes from the original feature will be preserved onto the output features.

### Example

#### *To be added*

### Related Transformers

#### *To be added*

### Transformer Category

Point Cloud

### Technical History

Associated FME function or factory: PointCloudCoercerFactory

## PointCloudCombiner

Combines multiple geometries into a single point cloud.

### Input Ports

- INPUT: This transformer accepts all geometries. Area-based geometries will be converted to point clouds according to the spacing parameter.

### Output Ports

- OUTPUT: A single point cloud feature.

### Parameters

#### Group By

Use this parameter to organize point clouds into groups. Each group of point clouds will have its own output point cloud.

#### Accumulate Attributes

If this parameter is set to Yes, the attributes from the original features will be merged onto the output point cloud features.

#### Count Attribute

If you specify a Count Attribute parameter (text string), an attribute with this name will be added to each output feature.

The attribute contains the number of features that were combined to create the point cloud feature.

#### Spacing

This parameter controls the spacing between points in ground coordinates that will be used to generate the representative point grid used for surfaces, solids, polygons, and donuts. You can either enter a number or extract the value from a selected attribute.

### Example

*To be added*

### Related Transformers

*To be added*

### Transformer Category

Point Cloud

### Technical History

Associated FME function or factory: PointCloudSplitterFactory



## **PointCloudCreator**

Creates a new point cloud feature with the specified size and components and sends it into the workspace for processing.

### **Output Ports**

- OUTPUT: Point cloud feature

### **Parameters – Point Cloud Properties Group**

#### Size Specification

This parameter specifies how to create the points in the point cloud: either by using origin and size parameters, or extents parameters.

#### Extents: X Lower Left Coordinate and Y Lower Left Coordinate

The parameters specify the origin for the lower-left corner of the feature as a whole.

#### Origin and Size: Width and Height

These parameters define the width and height of the point cloud, in ground units.

#### Origin and Size: Average Spacing Along X and Average Spacing Along Y

These parameters specify how many points per ground unit should be in the output point cloud.

#### Extents: X Upper Right Coordinate and Y Upper Right Coordinate

These parameters specify the origin for the upper-right corner of the feature as a whole.

#### Rotation

The rotation angle is measured in degrees counterclockwise from horizontal, and measures the rotation of the primary axis from horizontal.

### **Parameters – Z Values Group**

#### Z Pattern

This parameter specifies how the points in the cloud are given Z values. The Z values will be assigned as follows:

- Flat: Every point in the cloud will have a z value specified by the minimum z.
- Sloped: The points will have z values between minimum and maximum arranged in a gradient throughout the point cloud.
- Trigonometric: The points will have z values between minimum and maximum arranged in a trigonometric pattern through the point cloud.

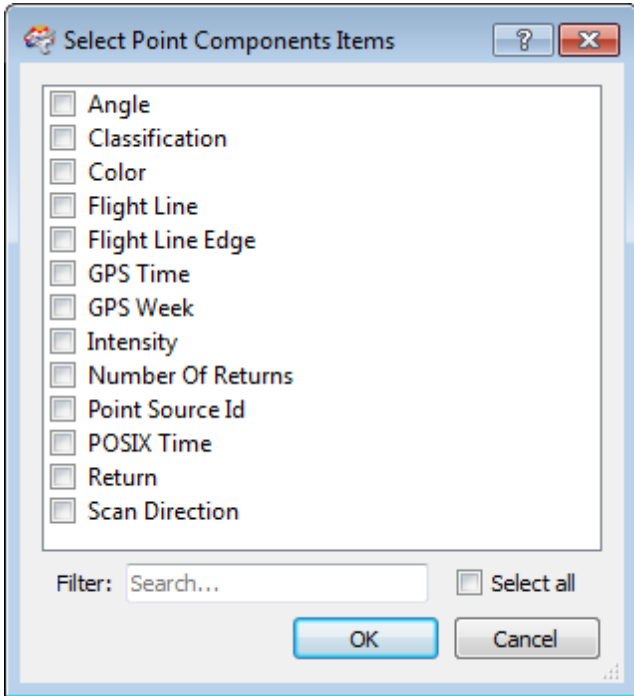
#### Z Minimum and Maximum

These parameters specify the minimum and maximum z value in the cloud.

### **Parameters – Component Values Group**

#### Point Components

Click the browse button to display the list:



Select which components should be added to each point in the cloud. Values for the points are generated as follows:

- Angle, Classification, Color, Intensity: Assigned values between their min/max by the Value Pattern parameter.
- Flight Line: starts at 1 and increases as specified by the Rows Per Flight Line parameter
- Flight Line Edge: 0 or 1 depending on if this is the last or first row with a particular flight line value
- GPS Time, Posix Time: starts at the value specified by the Start Time parameter and increase based on the Time Increment parameter
- GPS Week: starts at the value specified by Minimum GPS Week and increase in a gradient pattern through the point cloud.
- Number Of Returns - has the value specified by the Maximum Number Of Returns parameter
- Returns: starts at 1 and have values arranged in a gradient throughout the point cloud.
- Point Id: a unique number assigned by the system.
- Scan Direction: 0 or 1 depending on the current value of the flight line.

**Maximum Number Of Returns**

Specifies the maximum value for the return component and the value of the maximum returns component.

**Rows Per Flight Line**

Specifies how many rows are within a flight line.

**Start Time**

Specifies the start time for the cloud.

**Time Increment**

Specifies how long between each point in the cloud.

#### Minimum and Maximum GPS Week

Specifies the minimum and maximum values for the GPS week component.

#### Value Pattern

If the Value Pattern is:

- Single Value: the component will have a value specified by its minimum
- Checkered Pattern: the component will have values arranged in a gradient throughout the point cloud
- Checkerboard: the component will have values alternating between its min/max

#### Minimum and Maximum Angle

Specifies the minimum and maximum angle

#### Minimum and Maximum Classification

Specifies the minimum and maximum classification

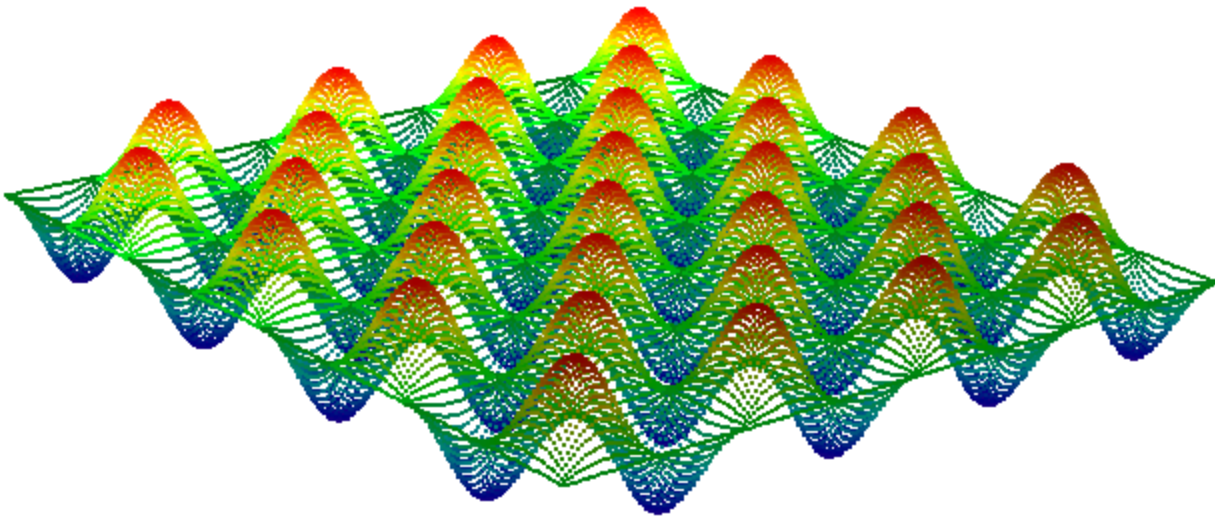
#### Minimum and Maximum Color Value

Specifies the minimum and maximum color values.

#### Minimum Intensity and Maximum Intensity

Specifies the minimum and maximum intensity.

### Example Output



### Transformer Category

Point Cloud

### fmepedia

See fmepedia for additional information about this transformer and point cloud scenarios.

### Technical History

Associated FME function or factory: PointCloudCreatorFactory

## PointCloudPropertyExtractor

Extracts the properties of a point cloud feature and exposes them as attributes. The attribute values are for reference only and may become out of date if the point cloud properties change.

The following properties are exposed:

- \_min\_x
- \_min\_y
- \_min\_z
- \_max\_x
- \_max\_y
- \_max\_z
- \_num\_of\_points

### Input Ports

INPUT: This transformer accepts only point cloud features.

### Output Ports

EXTRACTED

### Parameters

Check Existence for Components

The following exposed properties relate to the interpretation of the points. They are added if the appropriate component is selected in the Check Existence For Components parameter. The value of the attribute will be a Boolean value indicating whether or not the component is present in the cloud. The attribute name below will be prefixed by Component Attribute Prefix.

- \_intensity
- \_color
- \_classification
- \_return
- \_num\_returns
- \_angle
- \_flight\_line
- \_scan\_direction
- \_point\_id
- \_posix\_time
- \_userdata
- \_gps\_time
- \_gps\_week
- \_flight\_line\_edge

Component Attribute Prefix

The prefix to add to attributes related to point cloud components.

## Retrieve Min/Max for Components

The following exposed properties relate to the min/max for a point component. They are added if the appropriate component is selected in the Retrieve Min/Max for Components parameter.

The value of the `_min` attributes will be the minimum value for that component in the cloud. The value of the `_max` attributes will be the maximum value for that component in the cloud. The prefix specified in the Component Attribute Prefix parameter will be inserted between the `_min` and the component name. If the specified component does not exist in the cloud, the attribute will not be output.

`_min_intensity, _max_intensity`

`_min_color, _max_color`

`_min_classification, _max_classification`

`_min_return, _max_return`

`_min_num_returns, _max_num_returns`

`_min_angle, _max_angle`

`_min_flight_line, _max_flight_line`

`_min_scan_direction, _max_scan_direction`

`_min_point_id, _max_point_id`

`_min_posix_time, _max_posix_time`

`_min_user_data, _max_user_data`

`_min_gps_time, _max_gps_time`

`_min_gps_week, _max_gps_week`

`_min_flight_line_edge, _max_flight_line_edge`

## Technical History

Associated FME function or factory: @ExtractPointCloudProperties

## PointCloudSplitter

Splits a single point-cloud feature into multiple point-cloud features, each having a homogeneous value for the point component that governs the split.

### Input Ports

- **INPUT:** This transformer accepts only point-cloud features.

### Output Ports

- **SPLIT:** Multiple point-cloud features as specified by the Split By parameter.

### Parameters

Split By

**Return Number:** One point-cloud feature is output for each return number found within the input point cloud. A final point-cloud feature will be output containing all of the last return points.

**Flight Line:** One point-cloud feature is output for each flight line number found within the input point cloud.

**Classification:** One point-cloud feature is output for each classification found within the input point cloud.

Output Attribute (optional)

If specified with **Return Number**, this attribute contains the return number of all the points in the output point cloud. For the feature representing the last return, the Output Attribute will contain -1.

If specified with **Flight Line**, this attribute contains the flight number for the feature.

If specified with **Classification**, this attribute contains the classification for the feature.

### Example

### Transformer Category

Point Cloud

### Technical History

Associated FME function or factory: PointCloudSplitterFactory

## PointCloudThinner

Outputs point-cloud features that have fewer points than the original input features. This transformer is typically used to reduce data volume by arbitrarily discarding data.

### Input Ports

- INPUT: Point-cloud features.

### Output Ports

- THINNED: Point-cloud features with a reduced number of points.

### Parameters

Thinning Type

**Keep Every Nth Point:** Keeps every <sampling amount>th point, and discards the remaining points.

**Maximum Number of Points:** Sets a maximum number of points for the feature.

Amount

If Thinning Type is "Keep Every Nth Point", the amount specifies how often the points are retained. For example, an amount of 2 will result in every other point of the input cloud feature will be present in the output point cloud.

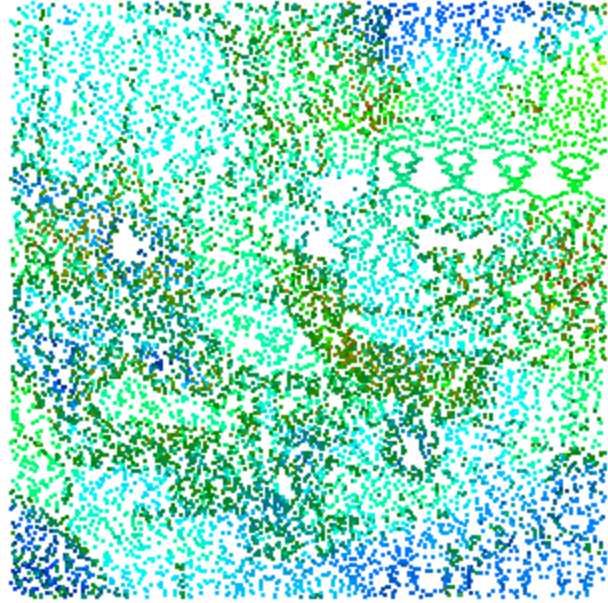
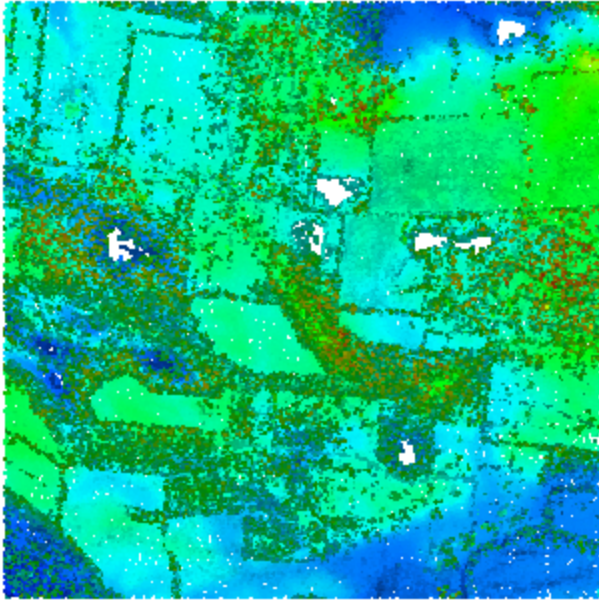
If the Thinning Type is "Maximum Number of Points", the amount specifies the maximum points in the output point cloud. For example, an amount of 100 will result in points in the input cloud being dropped so the maximum number of points in the output cloud is 100.

Note: If you enter a sampling amount of 0, all input data will be discarded no matter which Thinning Type you choose.

### Examples

Thinning Type	Amount	Result
Keep Every Nth Point	2	Every other point of the input cloud feature will be present in the output point cloud.
Maximum Number of Points	100	The output cloud will have a maximum of 100 points, arbitrarily chosen from the input.

In this example, the output point cloud on the right keeps every 50th point from the source point cloud on the left:



### **Transformer Category**

Point Cloud

### **Related Transformers**

The transformer can be used with PointCloudCombiner to make mosaics of large areas from smaller tiles.

### **fmepedia**

See fmepedia for additional information about this transformer.

### **Technical History**

Associated FME function or factory: @ThinPointCloud



## PointConnector

Connects input point features in the order they enter, forming linear or polygonal features.

For some datasets, it may be necessary to use a Sorter to order the data correctly before it enters this transformer.

The feature being created is output whenever one of the Connection Break Attributes' values changes.

When this happens, the feature with the differing attribute is not added to the current output feature; instead, it begins the next feature to be output.

If only one feature was received with a given value for the connection break attribute, it will be output as a **POINT**.

If the feature output forms a ring (begins and ends at the same point), it is output as an **AREA**.

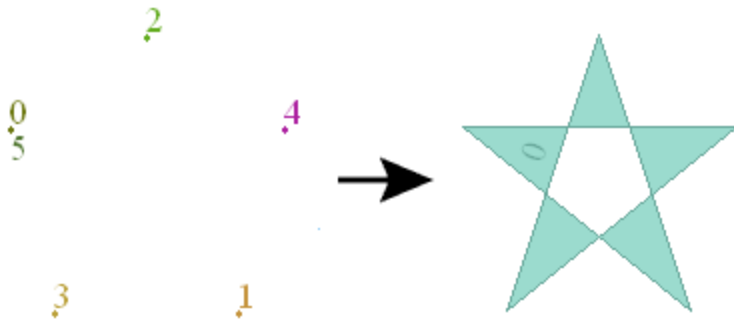
Anything else is output as a **LINE**.

If the optional List Name is supplied, a list is made of all the attributes of each point that was connected when creating the output feature. This allows later inspection of member point attributes. In addition, the feature itself will contain a merged set of attributes from all the features that were connected together, where the value for an attribute in the resulting feature comes from the first feature that had an attribute with that name that was part of the resulting feature.

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs will be allowed as input (treated as lines); they will otherwise be ignored.

## Example



## Transformer Category

Geometric Operators

## Technical History

Associated FME function or factory: ConnectionFactory

## **PointMeasureExtractor**

Category: Linear Referencing

Sets the Destination Measure Attribute to the value of the measure named by Source Measure Name. If no Source Measure Name is given, the default measure will be used.

### **Technical History**

Associated FME function or factory: @Geometry

## **PointMeasureSetter**

Category: Linear Referencing

Sets the measure on a point geometry to Measure Value. If no Destination Measure Name is specified, the default measure will be set.

### **Technical History**

Associated FME function or factory: @Geometry

## PointOnAreaOverlayer

Performs an overlay of points on areas.

### Parameters

#### Group By

The Group By parameter specifies a series of attributes that must match on the point and polygon features before the attributes will be merged. This can be used to ensure that the overlay occurs only on certain groups of features.

#### Overlap Count Attribute

The Overlap Count Attribute added to output area features holds the number of point features that they contained.

#### List Name

If the optional List Name is supplied, the attributes of each area containing an output point are added to that point's list, and the attributes of each point contained by an output area are added to that area's list.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are not accepted as points, and ellipses are accepted as areas and preserved as ellipses; otherwise, arcs are accepted as points, and ellipses are accepted as either points (located at their center points) or as stroked areas.

### Usage Notes

- Each point receives the attributes of the area(s) it is contained in, and each containing area receives the attributes of each point it contains. When attributes are merged between features, existing attributes are not replaced. Therefore, if the points and areas being overlaid have attributes with the same name, then the values will not be transferred from one to the other. You can avoid this problem by renaming (`AttributeRenamer`), prefixing (`AttributePrefixer`), or removing (`AttributeRemover`) attributes to avoid name collisions.
- Note that intersections between area features are not computed.
- If a point falls exactly on the line between polygons, FME always include points on the boundary as being in. So if a point is on the boundary of two polygons, it will be called "in".
- If you have a lot of features to process, you can improve overlay performance by also using the Clipper transformer. The Clipper also provides different options for a point that falls exactly on a line. See the Clipper page in [fmepedia](#) for information on how to use less memory using the "Clippers First" parameter.

### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: `OverlayFactory`

## PointOnLineOverlayer

Performs an overlay of points on lines. Each input line is split at its closest place to any point within the specified point tolerance.

### Parameters

#### Group By

The Group By parameter allows you to choose a series of attributes that must match on the features before the attributes will be merged. This can be used to ensure that the overlay occurs only on certain groups of features.

#### Overlap Count Attribute

The Overlap Count Attribute added to output linear features holds the number of point features that were near to it. The Overlap Count Attribute added to output point features holds the number of linear features that the point was near to

#### Point Tolerance

The Point Tolerance value is compared to the distance from the lines to the points, and the lines will be segmented if the distance is less than or equal to the Point Tolerance value. When such a match occurs, the attributes of the segmented lines are merged with the points and the attributes of the points are merged with the lines.

#### List Name

If the optional List Name is supplied, the attributes of each point used to segment an output line are added to that line's list, and the attributes of each line that a point segmented are added to that point's list.

### Usage Notes

- Each resulting point receives the attributes of the point(s) it was paired with. When attributes are merged between features, existing attributes are not replaced. Therefore if the points and areas being overlaid have attributes with the same name, then the values will not be transferred from one to the other. You can avoid this problem by renaming (AttributeRenamer), prefixing (AttributePrefixer), or removing (AttributeRemover) attributes to avoid name collisions.
- Note that intersections between linear features are not computed.

### Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are accepted only as lines, and not as points, and ellipses are accepted as neither points nor lines; otherwise, arcs and ellipses will only be accepted as points (located at their center points).

### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: OverlayFactory

## **PointOnPointOverlayer**

Performs an overlay of points on points.

### **Parameters**

#### Overlap Count Attribute

The Overlap Count Attribute added to output point features holds the number of point features to which the point was near.

#### Point Tolerance

Each input point has the attributes from any other point within the Point Tolerance distance merged onto it.

#### List Name

If the optional List Name is supplied, each output point will also have an attribute list containing the attributes from each point that is within the Point Tolerance distance.

### **Geometry Handling**

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, neither arcs nor ellipses are accepted as points; otherwise, they are accepted as points (located at their center points).

### **Usage Notes**

When attributes are merged between features, existing attributes are not replaced. Therefore, if the points and areas being overlaid have attributes with the same name, then the values will not be transferred from one to the other. You can avoid this problem by renaming (*AttributeRenamer*), prefixing (*AttributePrefixer*), or removing (*AttributeRemover*) attributes to avoid name collisions.

### **Transformer Category**

Geometric Operators

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: *OverlayFactory*

## PointOnRasterValueExtractor

Takes in a number of point features and a single reference raster. For each input point feature, a point is created after the reference raster and then output.

The data contained in the resulting point consists of the attributes from the reference raster as well as the band and palette value(s) at the location of the point.

### Parameters

#### Group By

The points may be organized into groups with the Group By parameter, with each group of points having its own reference raster.

#### Interpolation Type

This parameter determines the interpolation method to be used when retrieving raster values from points.

Nearest Neighbor interpolation is the fastest and least accurate, while Bilinear and Bicubic are slower but will weight surrounding data values into the final result. Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

The following band and palette properties are exposed:

`_band{0}.value`

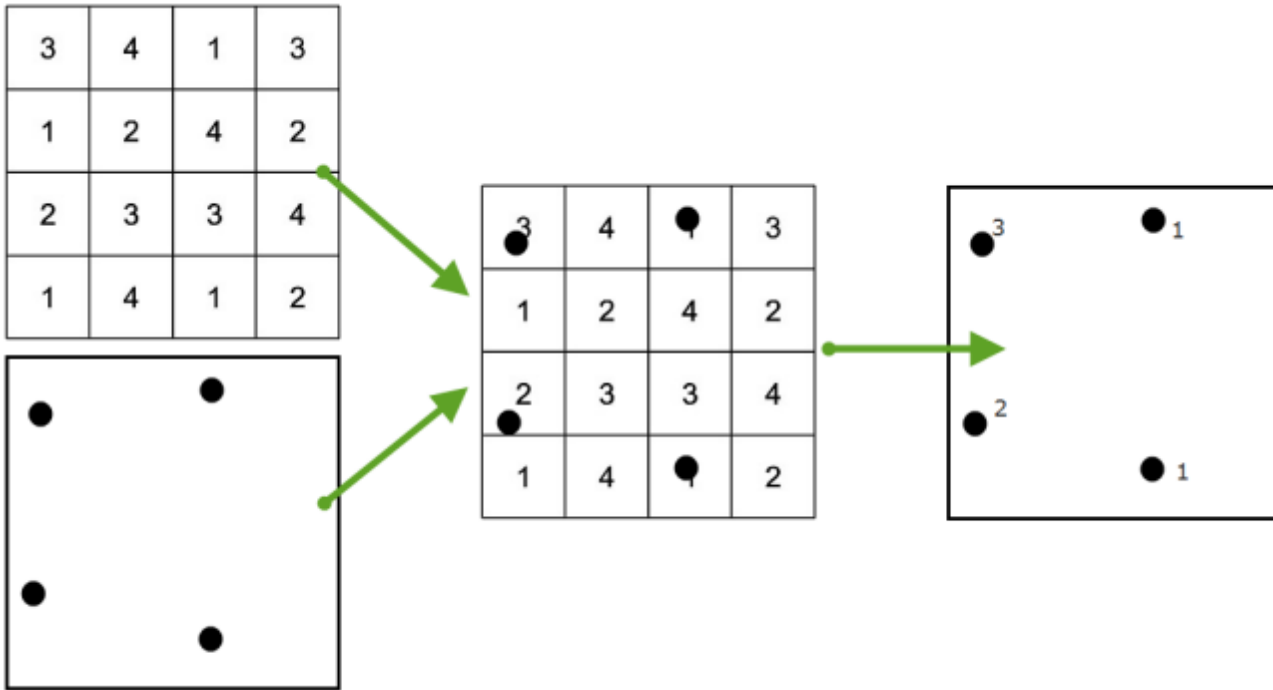
`_band{0}.palette{0}.value`

`_band{}.value`

`_band{}.palette{}.value`

Note: Wherever "{}" appears, there will be one instance of the attribute for each band or palette. The first will appear with {0}, the second with {1}, and so on.

## Example



### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Transformer History

This transformer was previously named PointOnRasterOverlayer.

### Technical History

Associated FME function or factory: VectorOnRasterOverlayFactory



## PolygonBuilder

Forms polygons from lines. The lines must be topologically correct and must not self-intersect or intersect each other, and they must close at nodes.

The Snapper, Intersector, SelfIntersector and MRF2DCleaner can be used to attempt to clean data that does not meet these conditions before it enters this transformer.

If these conditions are met, any polygons implied by the input lines are created and output via the POLYGON port. Note that no donut polygons will be created – the DonutBuilder can be used on the output of this transformer if holes are to be cut in the resulting polygons. Alternately, the AreaBuilder transformer can be used in place of this transformer to both form polygons and nest holes in them in a single operation.

Any lines that did not close are output via the UNUSED\_LINE port.

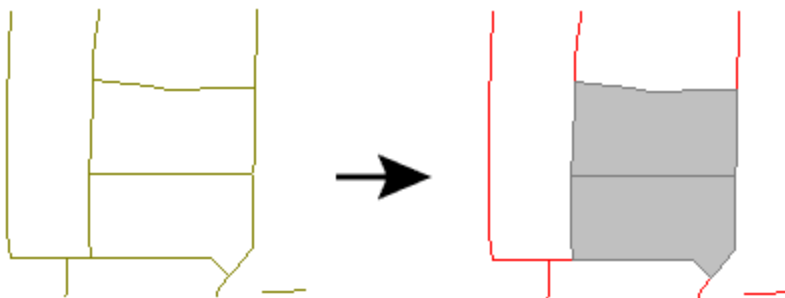
If **List Name** is given, then a list will be created on each output feature containing an element for each input feature which contributed to that geometry, in order of appearance.

**Note:** Only Group\_By attributes are preserved by this transformer, unless List Name is given. The built polygon will have the same coordinate system as the lines within the Group By – if the lines within the Group By have conflicting coordinate systems set, the resultant polygon will have no coordinate system.

**Build Internal Edges** (Advanced) specifies that coordinate "cycles" within a polygon are allowable and will be constructed; such polygons might be considered invalid by other parts of FME or by output formats. A "cycle" is a line segment that occurs twice in the same polygon's boundary (once in each direction).

**Note:** If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, arcs and ellipses are accepted as linear features; they are otherwise rejected as point features.

## Example



## Transformer Category

Geometric Operators

## Technical History

Associated FME function or factory: PolygonFactory

## ProxixGeocoder

Geocodes addresses using a Proxix Geospatial Enterprise Real-Time (GSERT) server.

### Output Ports

- **SUCCEEDED:** If a feature has been successfully geocoded, it will be output through this tag.
- **FAILED:** If the Proxix GSERT server indicates a feature could not be geocoded, it will be output through this tag.

### Attributes

Features that were successfully geocoded will contain attributes common to all datasets. For the complete list of attributes, see your Proxix representative.

### Matchcodes

Features that were not successfully geocoded will have `_px_matchcode` and `_px_error` attributes added, where `_px_error` is a human-readable explanation of the matchcode.

The PxPoint User's Guide (C API) provides full details on how to interpret match codes. This document is available through your Proxix representative. In general, the first character of each match code indicates a category of statuses:

Match Category	Meaning
U	Failed to find a match
M	Two or more matches with the same score
A	Matched a discrete address
B	A composite of multiple matching records has been created
C	No house number match, using closest range
F	No house number match, using closest range, and ignored a poorer matching street with a house number match
I	Intersection address
R	Interpolated in house number range

### Parameters

Address Line, City Line

Intersections may be geocoded with the following notation in the Address Line: "Fake St. & Imaginary St."

Single line addresses may be passed through entirely on the Address Line; there is no need to manually split up street and city components.

PxPoint URL

The URL of the Proxix PxPoint web service that you wish to use. The default URL is <https://www.proxixnetwork.com/gsert/PxPointGeocode.asmx>, which requires a Proxix GSERT account.

Proxix Username, Proxix Password

The username and password of the relevant Proxix GSERT account.

Replace With Point

If this parameter is set to Yes, features will be converted to points in the LL84 coordinate system, replacing existing geometry. The coordinates of the converted points are retrieved from the values of `_px_longitude` and `_px_latitude`. Z values will be dropped.

Reproject to Source

If Reproject to Source is set to Yes, the new points will be reprojected into the source coordinate system. If the source feature have no coordinate system, the new point will be left in LL84. The `"_px_latitude"` and `"_px_longitude"` attributes will always be LL84.

## Dependencies

Before you use this transformer, please ensure you have access to a Proxix PxPoint web service. See <http://www.proxix.com/> for more details.

## FME Licensing Level

FME Professional edition and above

## Transformer Category

Web Services

## Technical History

FME Factory Used: ProxixGeocoderFactory

## PythonCaller

Processes FME Features using a Python function or class provided by the user. The Python class or function can be written in the PythonCaller's source code editor, or it can reside in an external Python module.

Use of the PythonCaller requires the name of the Python function or class to be specified. If the Python function or class is implemented in an external Python module, the function/class name must be qualified with the module name.

In addition to the standard Python module locations, FME will also search the location of the workspace for the Python module.

### Parameters

Python symbol to use

The name of a Python Function, Class, or Object that will be used to implement the transformer.

Python source code

Python source code that is encoded as described in *Mapping File Syntax > Substituting Strings in Mapping Files*, in the FME Fundamentals manual (select Help > FME Fundamentals Reference).

FME Objects Namespace

The newer fmeobjects API is a replacement for the existing pyfme API. fmeobjects is the default. See the Python FMEObjects API documentation for more details.

### Function Implementation

A Python function should be used when the PythonCaller is intended to process a single feature at a time. The PythonCaller will call the Python function with exactly one argument: an FMEFeature object. If the Python function has more than one argument, an error will be generated by Python.

The FMEFeature provided is the same feature that will be sent out of the PythonCaller's output port.

The return value of the provided function will be ignored by the PythonCaller.

### Class Implementation

A Python class should be used when the PythonCaller is intended to process several features at a time, or generate multiple new features.

The provided class is expected to implement a method named "input", and/or a method named "close". If the class implements a constructor, it must not expect arguments in addition to `self`.

The input method will be called with exactly two arguments: `self`, and a FMEFeature object. If the method does not have exactly two arguments, an error will be generated by Python. The input method will be called once for every feature that is sent into the PythonCaller's import port.

The close method will be called with exactly one argument: `self`. If the method does not have exactly one argument, an error will be generated by Python. The close method will be called after all input features have been received by the PythonCaller.

The PythonCaller injects a method named `pyoutput` into the provided class. This method takes a FMEFeature object as its single parameter, and sends the provided feature out of the PythonCaller's output port.

### FME Objects API

pyfme, the Python wrapper to FME Objects, is available for use within the PythonCaller. The pyfme.FMEFeature class will be used most commonly, however all other classes provided by pyfme are also available. Familiarity with the pyfme module is recommended.

### FME Macros

All FME Macros, and published parameters are available via the pyfme.FME\_MacroValues dictionary. This dictionary is considered read-only. Changes made to the values within the FME\_MacroValues dictionary will not have any effect in other Transformers.

### Example Function

```
def MyExampleFn(feature):  
    feature.setAttribute("message", "Hello world")
```

If this function is implemented in the Source Code editor, it can be referenced as MyExampleFn. If the function is implemented in an external Python file named, for example, myfmestuff.py, the function should be referenced as myfmestuff.MyExampleFn

### Example Class

```
#Make sure pyfme is imported before use
import pyfme
#This is an example of a Many To One PythonCaller
# The class takes multiple input features, and outputs a single feature
class MyExampleCls(object):
    #Example constructor with no additional parameters
    def __init__(self):
        self.featureList = []

        def input(self,feature):
            #Save the feature for processing at the end
            self.featureList.append(feature)
            #The input method does not pass the feature out of the output port
```

If this class is implemented in the Source Code editor, it can be referenced as MyExampleCls. If the class is implemented in an external Python file named, for example, myfmestuff.py, the class should be referenced as myfmestuff.MyExampleCls

### Dependencies

The PythonCaller supports Python 2.3 to 2.6.

If you have not installed Python separately, the PythonCaller will use an installation of Python 2.6 that ships with FME.

### fmepedia

See Python and FME for additional information about this transformer, and using Python with FME.

### Transformer Category

Infrastructure

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: PythonFactory

## **PythonCreator**

Generates FME features using a Python object referenced by the symbol name parameter. The handler object can be a Python function, a Python class, or any Python object that understands the FME Factory protocol.

The handler object can be defined in either the transformer's code editor window, or it can be defined in an external python module. If defined in an external module, the transformer's symbol name parameter must include the module name.

For example, if the class MyFactoryHandler is defined in the source code editor, it can be referenced as ``MyFactoryHandler'`. However, if the class MyFactoryHandler is defined in `myFactories.py`, it must be referenced as ``myFactories.MyFactoryHandler'`.

### **Transformer Category**

Infrastructure

### **Dependencies**

The PythonCaller supports Python 2.3 to 2.7.

To control the Python Interpreter used by FME: from Workbench, select Tools > FME Options > Runtime. Then select the Python interpreter's dynamic library (python2x.dll on Windows, libpython2x.so\* on UNIX).

By default, FME will use an installation of Python 2.7 that ships with FME.

### **FME Licensing Level**

FME Professional edition and above

### **fmepedia**

See Python and FME for additional information about this transformer, and using Python with FME.

### **Technical History**

Associated FME function or factory: PythonFactory

## RandomColorSetter

Sets a random color for each incoming feature.

### Parameters

Color to Set

Outline	Sets the pen outline color.
Fill	Sets the area color for polygons and donuts.
Both (Offset)	Sets both colors, with the pen color slightly different than the fill color. This gives a good visual result.
Both (Exact)	Sets both the outline and fill to the exact same color.

### Transformer Category

Infrastructure

### Technical History

Associated FME function or factory: @Tcl2

## **RandomNumberGenerator**

Generates a uniformly distributed random number.

The random number is  $x$ , where  $\text{Minimum Value} \leq x \leq \text{Maximum Value}$ .

This random number will be rounded to the number of digits specified in the Decimal Places parameter.

Note that in general, when the Minimum Value and Maximum Value is specified in an exponential form (for example, "1.0e+25"), the Decimal Places refer to the number of decimal places in the exponential form.

If Minimum Value or Maximum Value have more decimal places than the Decimal Places requested for the result, the actual minimum value used will be rounded up and the actual maximum value used will be rounded down to the requested decimal places. This is to ensure that the generated random number will always fall between the Minimum Value and the Maximum Value.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Evaluate, @Tcl2



## **RasterBandAdder**

Adds a new band to a raster. The added band will have the same value in all cells, and the same raster-level properties as other bands in the raster (that is, number of rows/columns, cell spacing, cell origin, etc.).

### **Parameters**

#### Interpretation

This parameter specifies the desired interpretation for the added band, including data type and bit depth.

#### Cell Value

This parameter specifies the value that will be used for all cells in the added band.

#### Nodata Value

This parameter sets the nodata value for the band.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @AddRasterBand

## **RasterBandCombiner**

Merges multiple overlapping raster features into a single raster feature. It accepts a number of input raster features, each of which has one or more bands. The bands are removed from the input features and appended to a single output raster feature.

### **Input**

- **INPUT:** Input features must contain raster features only.

### **Parameters**

#### Group By

The rasters may be organized into groups, with each group of rasters having its own output raster.

The properties of each raster within a group, such as the number of rows and columns, must match for the processing to proceed successfully. Each raster within a group must also overlap identically.

#### Accumulate Attributes

If set to Yes, then the attributes from the original features will be merged onto the output raster features.

#### Count Attribute

If a Count Attribute is given, then an attribute with this name will be added to each output feature, containing the number of features that were combined to create the raster feature.

### **Usage Notes**

- The order of the input features and the order of the bands of the input features both determine the order of the bands in the output feature. A *Sorter* transformer may be used to enforce the order in which the features are processed.
- This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: RasterMergerFactory

## RasterBandInterpretationCoercer

Alters the underlying interpretation of the selected bands of the raster geometry on the input features, using the specified conversion options. For example, an input raster feature with a single band of interpretation UInt8 could be converted to a single band of Gray8 or UInt16 data.

### Parameters

#### Destination Interpretation Type

This parameter selects the destination interpretation along with the bit depth. Different interpretations allow for different conversion options to be used. If bands selected for conversion contain palettes, then the destination interpretation is restricted to UInt8, UInt16 or UInt32.

Convert from Color to Color, Convert to Numeric to Color, Convert Color to Numeric, Convert Numeric to Numeric

These parameters select the action to undertake when the given conversion between different types occurs. The Cast option simply uses C-style casts to convert the values. The Bounded cast option uses C-style casts to convert the values too, but also validates that the source values fit into the destination type, effectively preventing underflow and overflow; if a source value does not fit, the corresponding destination value will either be set to its type's minimum or maximum value. The Scale by data values option finds the minimum and maximum values of the source values and uses them to scale the values to the full range of the destination type. The Scale by data type option scales the source values while preserving all proportions in regard to the source and destination types' range.

#### Convert from Float to Integer

This parameter specifies the action to perform when converting from a floating-point value to an integer.

- Round: rounds to floating-point value to the nearest integer.
- Ceiling: gets the next integer which is greater than or equal to the floating-point value.
- Floor: gets the next integer which is less than or equal to the floating-point value.

### Usage Notes

- Each RasterBandInterpretationCoercer performs a conversion on the input raster. If multiple RasterBandInterpretationCoercers are used in sequence, then multiple conversions will take place. Data quality and translation performance may suffer.
- This transformer supports raster band selection. The RasterSelector can be used to modify the selection.

### Related Transformers

- The RasterInterpretationCoercer performs similar operations on the raster as a whole, such as converting 4 bands directly to RGBA.
- The RasterPaletteInterpretationCoercer performs similar conversions on palettes.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Transformer History

This transformer replaces the RasterDataTypeCoercer and RasterColorModelCoercer transformers.

### Technical History

Associated FME function or factory: @ReinterpretRaster

**RasterBandKeeper**

Removes all bands of a raster, except for those that are selected. The RasterSelector can be used to modify the selection.

If all bands in the source raster are selected, the raster will remain unchanged.

**Transformer Category**

Rasters

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @RemoveRasterBands

## RasterBandMinMaxExtractor

Extracts the band minimum and maximum values, palette minimum and maximum keys and palette minimum and maximum values of a raster feature and exposes them as attributes.

The attribute values are for reference only at one point in the workspace and may become out of date if the raster properties change.

The following band and palette properties are exposed:

```
_band{}.min  
_band{}.max  
_band{}.palette{}.keyMin  
_band{}.palette{}.keyMax  
_band{}.palette{}.valueMin  
_band{}.palette{}.valueMax
```

Note that wherever "{}" appears, there will be one instance of the attribute for each band or palette. The first will appear with {0}, the second with {1}, and so on.

## Related Transformers

This transformer only extracts band and palette minimum and maximum properties. To extract other band properties, use the RasterBandPropertiesExtractor. To extract raster geometry properties, such as the number of rows and columns, use the RasterPropertiesExtractor.

## Usage Notes

- This transformer accepts only raster features.
- Note that the any nodata values that are present on the band are ignored in the minimum and maximum value calculations.
- This transformer is unaffected by raster band and palette selection.

## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: @ProcessRasterTiles

## RasterBandNameSetter

Sets the name of selected bands on a raster.

### Input

- INPUT: Input features must contain raster features only.

### Parameters

#### Input Type

The band names may be specified in two ways depending on the parameter you choose for Input Type:

- Single Name, which enables the Band Name parameter
- List of Names, which enables the List Attribute parameter

#### Band Name

When Input Type is set to Single Name, a single band name must be specified in the Band Name parameter. This name will be set on all selected bands.

#### List Attribute

When Input Type is set to List of Names, a list attribute that contains band names must be specified. Then, the name for each selected band will be set to the value at the corresponding index in the list. The list attribute must contain at least as many elements as there are selected bands.

For example, suppose you have the following list attribute:

```
_bandnames{0} = 'red'
```

```
_bandnames{1} = 'green'
```

```
_bandnames{2} = 'blue'
```

and the raster feature has three bands, where bands 0 and 2 are selected. Then, the name of band 0 will be set to 'red', the name of band 1 will be unchanged (since it is not selected), and the name of band 2 will be set to 'green' (since 'green' is the second element in the list and band 2 is the second selected band).

### Usage Notes

- This transformer supports raster band selection: see the RasterSelector.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @RasterName

## **RasterBandNodataRemover**

Removes any existing nodata values from the selected bands of a raster feature.

If a nodata value was not present, this transformer will not perform any action on the feature.

### **Input**

INPUT: Input features must contain raster geometries only.

### **Parameters**

None.

### **Usage Notes**

This transformer supports raster band selection: see the RasterSelector.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Transformer History**

This transformer was previously named RasterNodataRemover.

### **Technical History**

Associated FME function or factory: @RasterNodata

## RasterBandNodataSetter

Identifies the nodata value on a raster feature at the band level. All selected bands on an input raster feature will receive the same specified nodata value.

If different nodata values are desired on each band, then multiple RasterBandNodataSetters must be employed using either different band selection, or by raster band splitting to separate the bands on which to set the nodata value.

### Input

- INPUT: Input features must contain raster geometries only.

### Parameters

#### Nodata Value

A numeric value specifying the nodata value to be set on the input raster. This parameter sets the nodata value and discards the original nodata value if it exists.

If the raster's bands have palettes, the function will succeed in setting a new nodata value only if the value is of a key that already exists on the band. To replace all occurrences of the original nodata value with a new value, use the RasterCellValueReplacer.

#### Replace Cell Values

This transformer also supports tagging the nodata value and optionally replacing the cell values.

### Usage Notes

- This transformer supports raster band selection: see the RasterSelector.
- This transformer does not set the nodata value on the palette level. To set the nodata value on the palettes of a raster, use the RasterPaletteNodataSetter.
- This transformer does not support RGB and RGBA color models.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @RasterNodata



## **RasterBandOrderer**

Specifies the order of bands in a raster. Bands are reordered according to the input band indices.

### **Parameters**

#### **Band List**

This parameter is a list of band indices separated by spaces. It specifies the desired order of the bands. Indices are zero-based, so the first band is at index 0.

Bands in the original raster that are not specified in the string will be appended after all the specified bands, in their original order. Note that bands may not be included in the list more than once.

For example, given an RGBA input raster with four bands and a band list of "3 0 2", the resulting raster would have bands ARBG.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @OrderRasterBands

## RasterBandPropertiesExtractor

Extracts the band and palette properties of a raster feature and exposes them as attributes.

The following band and palette properties are exposed:

```
_band{}.band_name  
_band{}.band_interpretation  
_band{}.band_bit_depth  
_band{}.band_interleaving  
_band{}.band_num_tile_rows  
_band{}.band_num_tile_columns  
_band{}.band_nodata  
_band{}.band_num_palettes  
_band{}.palette{}.palette_name  
_band{}.palette{}.palette_key_interpretation  
_band{}.palette{}.palette_value_interpretation  
_band{}.palette{}.palette_bit_depth
```

Wherever "{}" appears, there will be one instance of the attribute for each band or palette. The first will appear with {0}, the second with {1}, and so on.

### Input

- INPUT: Input features must contain raster geometries only.

### Parameters

None.

### Usage Notes

- This transformer extracts only band and palette properties. To extract raster geometry properties, such as the number of rows and columns, use the `RasterPropertiesExtractor`.
- This transformer is unaffected by raster band and palette selection.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @RasterProperties

**RasterBandRemover**

Removes the selected band(s) of a raster. To modify the selection, see the RasterSelector.

If the source raster has no selected bands, the raster will remain unchanged.

**Transformer Category**

Rasters

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @RemoveRasterBands

## RasterBandSeparator

Separates the bands and palettes from each input raster feature into one or more output raster features based on the number of input bands and palettes.

### Input

- **INPUT:** Input features must contain raster features only.

### Parameters

Split By

**Band and Palette:** The bands are split in the same way as the Band Only case, and additionally bands are constrained to having only one palette per band. Thus each output raster feature will have no more than one band and no more than one palette.

**Band Only:** Each band on the input raster feature will be placed onto a unique output raster feature. Thus each output raster feature will have no more than one band, but each band may have multiple palettes.

**Palette:** Each palette on each of the input raster feature bands is placed on a unique band on the raster. Only one raster feature will be produced per input feature. Each output raster feature may have multiple bands but will have no more than one palette per band.

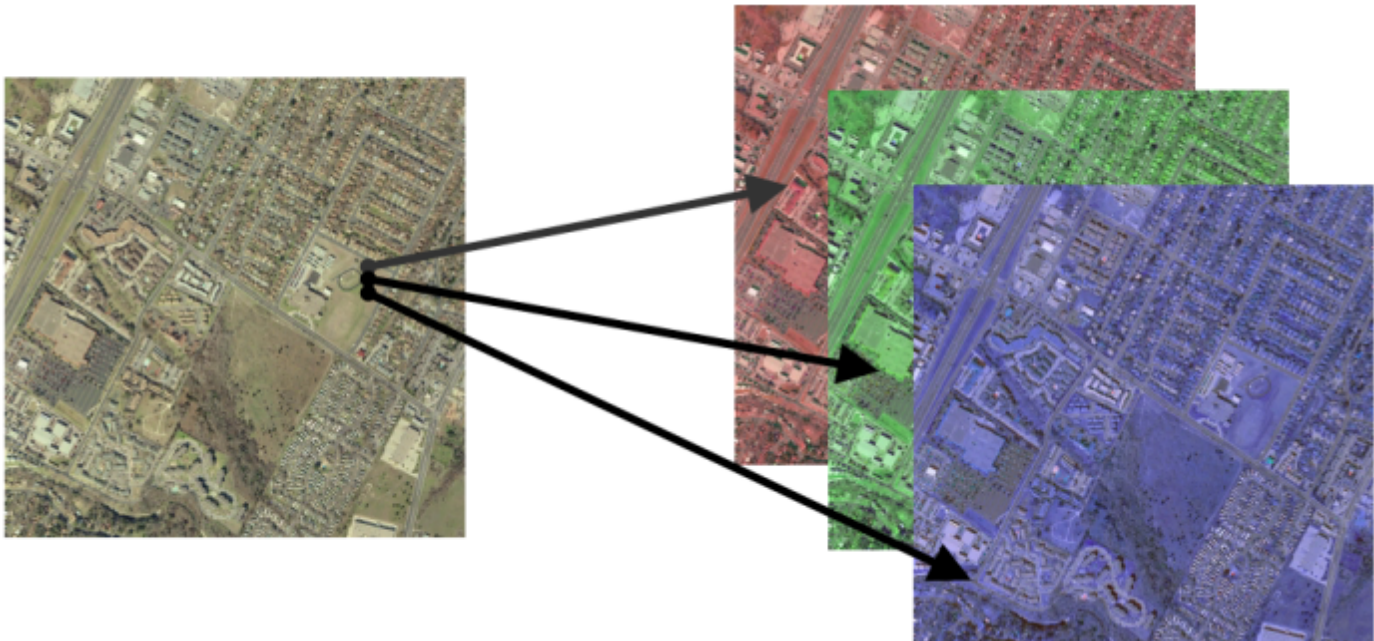
Raster Index Attribute, Band Index Attribute, Palette Index Attribute

The attributes are added to the output raster to specify which raster/band/palette it originated from. To disable this functionality, delete the default text from the parameter(s). This mode only works with *Band Only* and *Band and Palette* Split By options.

### Usage Notes

- The order of the input features and their bands and palettes will determine the order of the output features. A Sorter transformer may be used to enforce the order in which the features are processed.
- This transformer is unaffected by raster band and palette selection.

### Example



### Transformer Category

Rasters

**Transformer History**

This transformer was previously known as the RasterBandSplitter.

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: RasterSplitterFactory

## RasterCellCoercer

Decomposes all input numeric raster features into individual points or polygons. One vector feature is output for each cell in the raster.

### Parameters

#### Output Cell Geometry

Points: one point feature will be created for each of the raster's cells, positioned at the cell's origin.

Polygons: polygon features will be output, each one covering one of the raster's cells.

#### Extract Nodata Values

No: point features will not be output for nodata cells in the raster.

Yes: a point feature will be output for each nodata cell.

#### Preserve Attributes

Yes: each point will retain the attributes from the input raster.

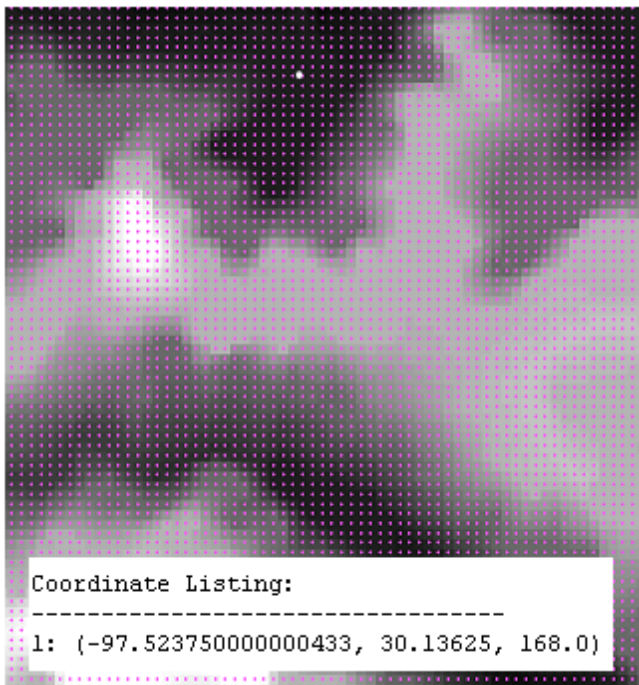
#### Raster ID Attribute, Band ID Attribute

Each point or polygon receives a raster ID and a band ID which correspond to the raster and band from which the vector feature originated.

### Usage Notes

This transformer will only operate on selected raster bands. Each selected band must not contain a palette.

### Example



### fmepedia

See fmepedia for additional information about this transformer.

### Transformer Category

Rasters

**FME Licensing Level**

FME Professional edition and above

**Transformer History**

This transformer was previously known as the RasterToPointCoercer, RasterPointExtractor, or GridPointExtractor.

**Technical History**

Associated FME function or factory: @ReplaceRasterCellValues

## **RasterCellOriginSetter**

Sets the raster's cell origin.

### **Input**

- INPUT: Input features must contain raster geometries only.

### **Parameters**

X Cell Origin, Y Cell Origin

These parameters specify whether each cell's data point is at the lower-left or center (or somewhere else) within the cell. Enter a number between 0.0 and 1.0, or choose the value from an attribute in the pull-down list.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @RasterCellOrigin



## RasterCellValueCalculator

Performs an arithmetic operation on a pair of rasters.

The first selected band of raster A is combined with the first selected band of raster B, the second selected band of raster A is combined with the second selected band of raster B, and so on.

### Input

Input features have the following restrictions:

- All input features must have raster geometry.
- Paired rasters must have the same number of rows and columns.
- Paired rasters must have the same number of selected bands.
- Paired bands must both have the same nodata value, or they both must have no nodata value.
- Bands may not contain a palette.

### Output Ports

- **OUTPUT:** Unselected bands are appended to the output raster, unchanged. The unselected bands of raster A are appended first, followed by the unselected bands of raster B.

### Parameters

#### Group By

To perform a calculation on more than one pair of rasters, a set of Group By attributes must be specified. Each group must contain one A raster and one B raster.

Group By attributes are always added to the output feature.

#### Operation

This parameter sets the operation that will be performed:

+	add
-	subtract
*	multiply
/	divide
Minimum Maximum Average	takes the minimum, maximum or average of A and B

For example, if you select the plus sign (+), the two input rasters A and B will be added together (and therefore, the output raster will be A+B).

#### Preserve Interpretation

If Preserve Interpretation is set to Yes, each output band will have the same interpretation as its input bands when the input bands share the same interpretation. If the input bands have different interpretations, or Preserve Interpretation is set to No, the interpretation of each output band will be automatically determined.

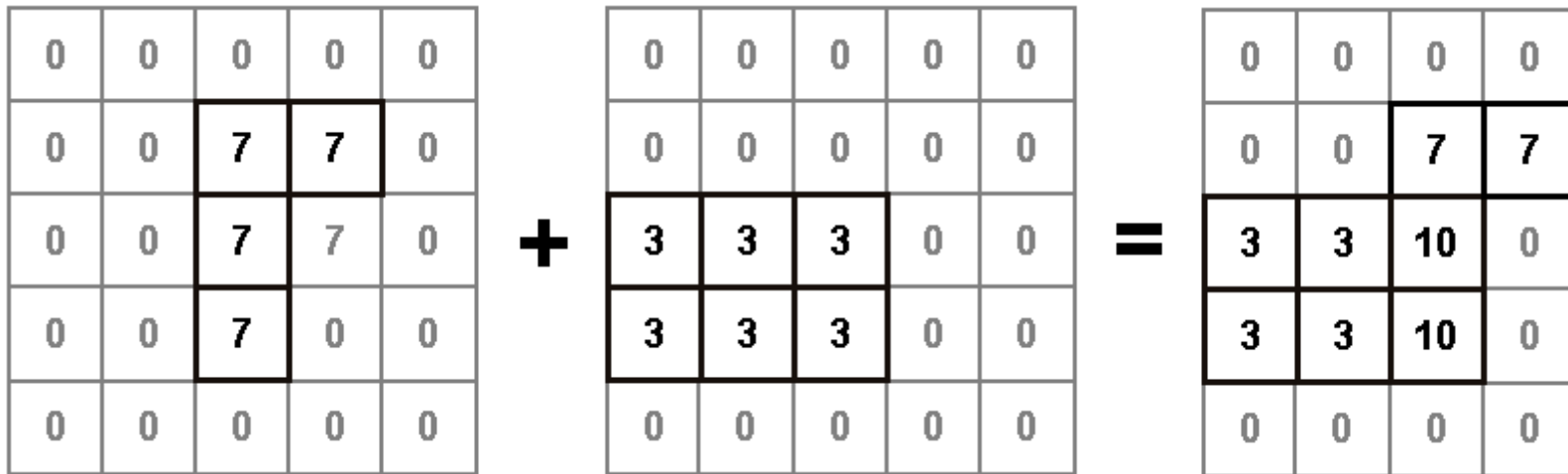
Note that when converting between different data types, a Bounded Cast is used. As a result, when a calculated value does not fit in the destination interpretation, the corresponding destination value will either be set to the minimum or maximum value possible in the destination data type.

#### Accumulate Attributes

If Accumulate Attributes is set to Yes, then the attributes from the original features will be merged onto the output feature. Group By attributes

are always added to the output feature.

### Example



### Related Transformers

- You can modify band selection with the RasterSelector.
- Nodata values can be set with the RasterBandNodataSetter or removed with the RasterBandNodataRemover.
- Palettes may be resolved using the RasterPaletteResolver or removed using the RasterPaletteRemover.
- You can use the RasterSingularCellValueCalculator to operate on a raster and a scalar.

### FME Licensing Level

FME Professional edition and above

### Technical History

FME Factories used: RasterEvaluationFactory, RasterSplitterFactory, RasterMergerFactory, TestFactory, SortFactory

FME Functions used: @RasterProperties, @Tcl2, @SupplyAttributes, @RemoveAttributes

## RasterCellValueReplacer

Replaces a range of values in the source raster with a new single value.

### Parameters

Replace Values  $\geq$ , Replace Values  $\leq$

Replace Values  $\geq$  specifies a range bounded by a minimum value, where all values greater than or equal to the supplied value will be replaced by the replacement value.

Replace Values  $\leq$  specifies a range bounded by a maximum value, where all values less than or equal to the supplied value will be replaced by the replacement value.

If both parameters are specified, both are applied. If the  $\geq$  value is less than the  $\leq$  value, then the bounds define a range of values bounded on both ends that will be replaced. For example,  $\geq 100$  and  $\leq 200$  replace with -999. Alternatively, if the  $\geq$  value is greater than the  $\leq$  value, then the bounds define two distinct ranges or everything outside the range provided. For example,  $\geq 200$  and  $\leq 100$  with -999 replaces all values in both the range  $\geq 200$  and the range  $\leq 100$ , and leaves the values 101-199 unchanged.

To specify a single value to be replaced, set both Replace Values parameters to the same value to be replaced.

To specify that all values except a single value should be replaced, supply two distinct ranges above and below the exempted value. For example, to replace all values except 100 with 0, specify  $\geq 101$  and  $\leq 99$ .

### New Value

Specifies the new value that will replace the specified range.

### Replace Nodata

Specifies whether nodata values will be replaced. If set to Yes, nodata values will be replaced just as any other values. If set to No, then nodata values will not be replaced, even if they fall within the specified bounds.

### Usage Notes

- This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.
- RGB and RGBA color models are not supported by this transformer.
- Rasters that contain bands with palettes are not supported by this transformer.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @ReplaceRasterCellValues

## RasterCellValueRounder

Rounds off raster cell values.

Note that floating point decimal values usually cannot be exactly represented. For example, the value 727.27 may actually be represented as 727.27002 when stored as a 32-bit floating point value. This property of floating point numbers may cause unexpected results when writing to text-based formats such as ESRIASCIIGRID. To avoid this, consider coercing floating point bands to real64 before using this transformer.

### Parameters

#### Decimal Places

Controls the number of decimal places to which the cell values are rounded.

A value of 0 causes the cell values to be rounded to the nearest integer. A value of 1 causes rounding to the nearest tenth. Negative values are allowed. A value of -1 causes rounding to the nearest 10.

#### Round-off Direction

Controls how the rounding will take place:

- Up: The cell values will always be rounded up.
- Down: The cell values will always be rounded down.
- Nearest (default): The cell values will be rounded up or down to the nearest value.

### Usage Notes

This transformer supports raster band selection.

The RasterSelector can be used to modify selection.

### Transformer Category

Rasters

### Technical History

Associated FME function or factory: RasterEvaluationFactory, RasterSplitterFactory, RasterMergerFactory, TestFactory, @RasterProperties, @Tcl2, @RemoveAttributes

## **RasterCheckpointter**

Sets a checkpoint in the raster processing which forces previous processing to occur immediately. Once complete, it saves the current state to disk.

### **Parameters**

None.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @CheckpointRaster

## **RasterConsumer**

Requests the tile(s) from the raster geometry but no actual operations are performed on the tile(s).

### **Input**

- INPUT: All input features must have raster geometry.

### **Parameters**

Tile Request Order

Tile: The raster data is read one tile at a time until all tiles are consumed. Several bands may be read for a given tile.

Band: The raster data is read one band at a time until all bands are consumed.

Tile Size

Use source tile size: The default tile size reported by the band will be used when tiles or bands are consumed.

Specify source tile size: The raster will be consumed using the specified Number of Tile Rows and Number of Tile Columns.

Number of Tile Rows, Number of Tile Columns

In conjunction with *Specify source tile size*, specify the number of rows and columns.

### **Usage Notes**

- Tiles may be used to partition each band into manageable chunks. Some raster sources may perform better using different patterns.
- This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @ProcessRasterTiles

## RasterDEMGenerator

Generates a Digital Elevation Model (DEM) represented as Raster from the input POINTS, BREAKLINES, and 3D\_LINES.

### Parameters

#### Surface Tolerance

The surface tolerance is used when building the surface to determine which vertices to add to the surface model. Specifying a value of 0 turns off the vertex filtering so that all vertices (except those with duplicate x,y) are added to the model.

When specified, the surface tolerance is used during the construction phase of the surface model to determine whether a vertex will be added to the model, or whether it will be discarded and not added to the model.

The surface tolerance works as follows, for each vertex that is being added to the model:

- If the x,y location is outside the convex hull of the existing model, it is added to the model.
- If the x,y location is inside the existing model:
  - The difference between the z value from the existing TIN and the z value of the vertex is calculated.
  - This difference is compared to the surface model tolerance.
  - The vertex is only added to the model if the difference is greater than the surface tolerance; otherwise, the vertex is discarded.

#### Interpolation Method

This parameter defines how the elevation for each cell is determined:

AUTO: The best method will be chosen automatically.

CONSTANT: The elevation of each cell is set to be the elevation of the closest model point.

PLANAR: Interpolation is used to determine the elevation of the center of the cell.

#### Output DEM X Cell Spacing, Output DEM Y Cell Spacing

These parameters are used to specify the sampling interval for the generated cells in the raster. These spacings are measured in the ground units of the input data.

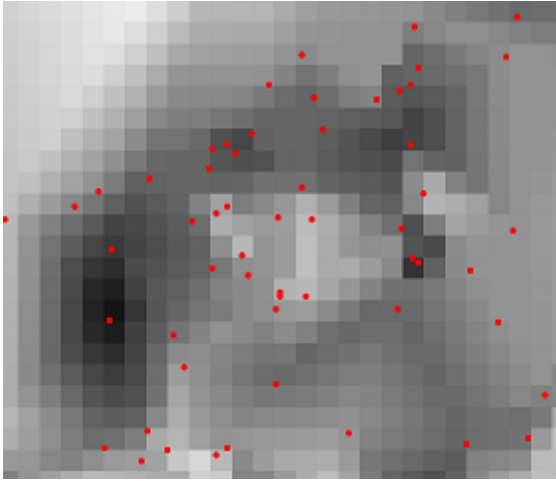
#### Output DEM Nodata Value

This parameter is used to fill the raster cell values that fall outside of the surface's bounding box. If this parameter is not used, NAN values will be used instead of nodata. A nodata value is specified by default and its use is highly recommended in order to produce consistent raster data.

### Usage Notes

- If a set of individual 3D points comprising a DEM is required, then the DEMGenerator should be used.
- This transformer is unaffected by raster band and palette selection.

## Example



### **Transformer Category**

Surfaces

### **FME Licensing Level**

FME Professional edition and above

### **Transformer History**

This transformer was previously named DEMGridGenerator.

### **Technical History**

Associated FME function or factory: SurfaceModelFactory



## RasterExpressionEvaluator

Evaluates expressions on each cell in a raster, such as algebraic operations or conditional statements.

### Input

Features are input through ports A and B. The cardinality of the input is required to be one of the following cases:

- one or more As, no Bs
- one A, one or more Bs

When both A and B features are provided, the single A input will be paired with each B input.

Note the following restrictions on input features:

- Input features must have raster geometry.
- All paired rasters must have the same number of rows and columns.
- Either all bands used in the same expression must have the same nodata value, or all bands used in the same expression must have no nodata value.
- No band may have a palette.

### Output

- **RESULT:** The output of the transformer will be a single raster feature per input pair. The output rasters will have n bands, where n is the number of interpretation/expression pairs, specified through the Interpretation List and Expression List parameters.

### Parameters

#### Group By

If any Group By attributes are given, then each group will be treated independently. This allows a single transformer to operate on multiple pairs of As and Bs. Note that this parameter is not applicable when only A input is provided; in that case, each raster is considered individually and there are no groupings.

#### Band Interpretation List

This parameter accepts a semicolon-delimited list of the interpretation for each output band. Valid interpretations are INT8, INT16, INT32, INT64, UINT8, UINT16, UINT32, UINT64, REAL32, REAL64, GRAY8, GRAY16, RED8, RED16, GREEN8, GREEN16, BLUE8, BLUE16, ALPHA8, ALPHA16.

Additionally, the interpretation for a band may be left unspecified. In this case, the output interpretation will be whatever data type was used to perform the operation. If the parameter is left entirely blank, all bands will have an automatically determined interpretation.

#### Band Expression List

This parameter accepts a semicolon-delimited list of expressions describing how to calculate each output band. A simple expression might be something like "(A[0] + B[0])/2", which calculates the average of the first band of input A and input B. Note that this parameter is case-sensitive.

See the *FME Functions and Factories* manual (Help > FME Functions and Factories Reference) for a detailed description of the structure of an expression, including the possible operands, operators, and functions.

As an example, if the input for Band Interpretation List was "REAL64; ; UINT32" and the input for Band Expression List was "A[0] + B[0]; A[0] / 2; A[1] \* 2", then the output would be a raster with a Real64 band, a band whose type was automatically determined, and a UInt32 band. The cell values in these bands would be calculated using the specified expressions.

When pairs of inputs are being operated on (i.e., the expression references both input A and B), the output feature will have all the attributes from both feature A and B. If the same attribute exists on both input features, then the attribute value from feature B will be preferred. Similarly, the output raster will have the properties of raster B. When only a single input is being operated on (i.e., the expression only references input A), the feature attributes and raster properties will remain unchanged.

### Transformer Category

Rasters

**FME Licensing Level**

FME Professional edition and above

**fmepedia**

See fmepedia for additional information about this transformer.

**Technical History**

FME Factory Used: RasterEvaluationFactory

## **RasterExtentsCoercer**

Replaces the geometry of input raster features with a polygon covering the extents of the raster.

### **Parameters**

Extents Type

Raster Extents: The polygon covers the entire extents of the raster. If the raster is rotated, the rotated corners of the raster are used for the polygon.

Data MBR Extents and Data Extents: These modes examine the data in the raster to more accurately determine the extents. A cell is considered to be nodata when, for each selected band, the value for that cell is equal to that band's nodata value. If any cell value is not equal to that band's nodata value, the cell will be considered data. In these modes, this transformer will only operate on selected raster bands, and each selected band is required to have a nodata value.

In Data MBR Extents mode, the polygon will be an axis-aligned bounding rectangle that covers all data cells in the raster. This is more computationally expensive than finding the Raster Extents.

In Data Extents mode, the output geometry will be either a single polygon or an aggregate of polygons that exactly cover only the data cells in the raster. This is more computationally expensive than finding the Data MBR Extents.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

FME Factories used: RasterSegmentationFactory, AggregateFactory, TestFactory

FME Functions used: @ProcessRasterTiles, @ApplyRasterRotation, @RasterProperties, @Tcl2, @SupplyAttributes, @RemoveAttributes, @RemoveGeometry, @GeometryType

## **RasterExtractor**

Serializes the geometry of the feature into the Raster Blob Attribute based on the selected writer format.

### **Parameters**

#### Raster Format

This is the format name of the writer.

#### Raster Format Name Attribute

This parameter will preserve the specified writer format value for future use.

#### Raster Blob Attribute

When writing the Raster Blob Attribute, you may need to adjust the attribute type to an unbounded data type to avoid truncation of the blob.

Please choose an appropriate attribute type depending on the destination format. For example, an appropriate attribute type for SQL Server is "image".

### **Usage Notes**

- The RasterReplacer may be used to do the reverse operation and convert the encoded blob back to the original data.
- Alternatively if the raster is not required as a geometry on the feature for future processing, the AttributeFileWriter transformer can be used to dump the Raster Blob Attribute directly to a file.
- To carry out a similar operation on vector data, please use the GeometryExtractor transformer.

### **Transformer Category**

Rasters

### **Related Transformers**

GeometryExtractor

GeometryReplacer

RasterReplacer

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

FME Factory Used: @RasterGeometry

## **RasterGCPExtractor**

Extracts the coordinate system and the GCP(s) from the raster feature and exposes them as attributes.

### **Input**

This transformer accepts only raster features.

### **Output**

The output attribute of this transformer is encoded as follows:

pixel line x y z;pixel line x y z etc.

The following raster properties are exposed:

\_gcp\_coordsys

\_gcp\_value

### **Parameters**

None.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @RasterGCP

## RasterGCPSetter

Sets the Ground Control Points (GCP) on a raster with the specified Column (pixel), Row (line), X Coordinate, Y Coordinate and Z Coordinate.

If the raster has already set GCPs, the old GCPs will be overwritten.

### Parameters

Source Coordinate System

This parameter indicates the coordinate system that would be effective if the Ground Control Points were applied. This is not necessarily equivalent to the coordinate system of the feature.

GCP Value

A string value specifying the GCP(s). Each GCP is separated with a semicolon and each value of a 5-value GCP is separated by a space.

### To set a GCP

- Set the source coordinate system.
- In GCP Value, you must enter 5 values, separated with a space: Column(pixel) Row(line) X-Coordinate Y-Coordinate Z-Coordinate. For example,

```
1 10 50 -20 0
```

- To set multiple GCPs, you can use a semicolon to separate each GCP. For example,

```
1 10 50 -2.5 0; 2 4 6 8 10; 1 2 3 4 5
```

### Usage Notes

This transformer is unaffected by raster band and palette selection.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @RasterGCP

## RasterGeoreferencer

Georeferences a raster using the specified parameters.

### Parameters

The Parameter Type determines which parameters are required in this transformer.

#### **Parameter Type**

*Point And Angle:* X Upper Left Coordinate, Y Upper Left Coordinate, X Cell Size, Y Cell Size and Rotation will be used.

In this case, the raster's origin will be set to the upper left coordinate and X Cell Size, Y Cell Size and Rotation are stored as input, untouched.

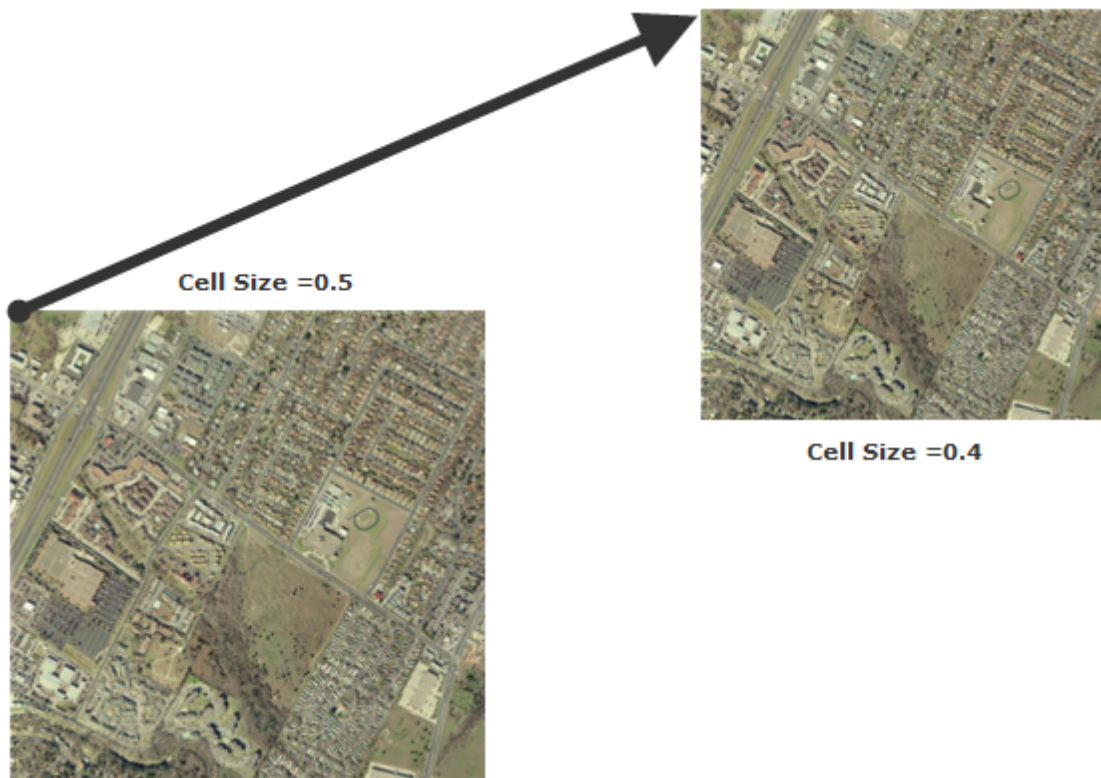
*Extents:* X Upper Left Coordinate, Y Upper Left Coordinate, X Upper Right Coordinate, Y Upper Right Coordinate, X Lower Right Coordinate, Y Lower Right Coordinate, X Lower Left Coordinate and Y Lower Left Coordinate will be used.

In this case, all input coordinates are validated to be unique, to form a rectangle which can be rotated, and to be clockwise. The rotation and spacing are then calculated from these coordinates and stored. The raster's origin will be set to the upper left coordinate.

### Usage Notes

- If the raster is already georeferenced, this will overwrite the old georeferencing information.
- This transformer is unaffected by raster band and palette selection.

### Example



### Transformer Category

Rasters

**FME Licensing Level**

FME Professional edition and above

**Technical History**

FME Factory Used: @GeoreferenceRaster



## RasterInterpretationCoercer

Alters the underlying interpretation of the bands of the raster geometry on the input features, using the specified conversion options.

For example, an input raster feature with three bands of interpretation (UInt16, Gray8, and Real64) could be converted to a raster feature with three bands of interpretation (Red8, Green8, and Blue8) or four bands of interpretation (Red16, Green16, Blue16, and Alpha16) in a single operation.

### Parameters

#### Interpretation

Destination Interpretation Type

This parameter selects the destination interpretation along with the bit depth. Different interpretations allow for different conversion options to be used. If any bands on the raster contain palettes, then the destination interpretation is restricted to UInt8, UInt16 or UInt32.

#### Conversion Options

RGBA to RGB

When converting to an interpretation with multiple components, such as RGB and RGBA, the raster will be expected to have exactly 1, 3, or 4 bands. If only one band is supplied, it will be cloned and converted to the appropriate interpretation. When converting to a single color or numeric interpretation, multiple input bands will be averaged into a single one.

The RGBA to RGB parameter selects the action to perform when converting four bands representing an RGBA raster to three bands representing an RGB raster.

- *Drop the alpha band* discards the alpha band.
- *Apply the alpha band* multiplies all RGB values with their corresponding, normalized alpha value.

RGB to RGBA

This parameter selects the action to perform when converting three bands representing an RGB raster to four bands representing an RGBA raster.

- *Create opaque alpha band* adds a new alpha band which has the maximum value for the data type in all cells.
- *Create alpha band from nodata* creates a new band which has the maximum value for the data type only in data cells. A cell is considered to be nodata when, for each selected band, the value for that cell is equal to that band's nodata value. If any cell value is not equal to that band's nodata value, the cell will be considered data. Note that when this option is selected, it is required that all input bands have a nodata value.

Convert from Color to Color, Convert from Numeric to Color, Convert Color to Numeric, and Convert Numeric to Numeric

These parameters select the action to undertake when the given conversion between different types occurs.

- *Cast* is very efficient and is lossless if the data values are contained within the destination data range. If outside the range, the Cast option allows overflow and rolls over the data values at the min and max of the destination data range, thus all values are contained in the destination range but if they rolled over may not reflect the source values very well.
- *Bounded cast* improves the basic Cast by also validating that the source values fit into the destination type, effectively preventing underflow and overflow; if a source value does not fit, the corresponding destination value will either be set to the minimum or maximum value possible in the destination data type.
- *Scale by data values* finds the minimum and maximum values of the source values and uses them to scale the values to the full range of the destination type.
- *Scale by data type* scales the source data range directly to the destination data range regardless of where the data values lie.

Convert from Float to Integer

This parameter specifies the action to perform when converting from a floating-point value to an integer.

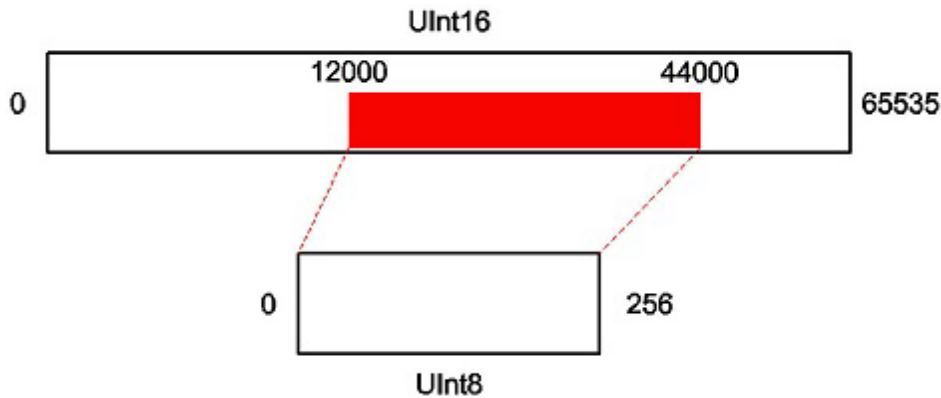
- *Round* rounds to floating-point value to the nearest integer.
- *Ceiling* gets the next integer which is greater than or equal to the floating-point value.
- *Floor* gets the next integer which is less than or equal to the floating-point value.

## Usage Notes

Each RasterInterpretationCoercer performs a conversion on the input raster. If multiple RasterInterpretationCoercers are used in sequence, then multiple conversions will take place; data quality and translation performance may suffer.

This transformer is unaffected by raster band and palette selection.

## Example



## Related Transformers

- The RasterBandInterpretationCoercer performs similar conversions on individual bands.
- The RasterPaletteInterpretationCoercer performs similar conversions on palettes.

## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Transformer History

This transformer replaces the RasterDataTypeCoercer and RasterColorModelCoercer transformers.

## Technical History

Associated FME function or factory: @ReinterpretRaster

## RasterMosaicker

This transformer mosaics multiple raster features into a single raster feature.

The transformer accepts a number of input raster features, each of which has one or more bands. All the raster features have the same number of selected bands, which, in turn, have the same number of selected palettes. Each selected band from each input raster feature will be removed, mosaicked together, and then appended to a single output raster feature. As a result, the output raster feature will also have the same number of selected bands and palettes.

### Input Features

- This transformer accepts only raster features.
- The order of the input features determines the order of how the bands overlap in the output feature. You can use a Sorter transformer to enforce the order in which the features are processed.
- When there are no selected palettes or Merge Palettes is No, each band in a set must have the same band interpretation and nodata value.
- When there are selected palettes and Merge Palettes is Yes, each band must have one selected palette, and all palettes must have the same value interpretation.

### Output Ports

- OUTPUT: The output feature created by mosaicking the input features.

### Parameters

#### Group By

The rasters may be organized into groups with the Group By parameter, with each group of rasters having its own output raster.

#### Snapping Type

The Snapping Type is used if the rasters are not perfectly aligned. The first input feature defines the reference grid. The following rasters can be Resampled to this grid. For better performance, they can also be Offset to match the grid.

#### Interpolation Type

If the input rasters do not line up correctly or if they have different spacings, this transformer will use the selected Interpolation Type to snap and/or resample the input rasters. Nearest Neighbor is the fastest but produces the poorest image quality. Bicubic is the slowest but produces the best image quality. Bilinear provides a reasonable intermediate option. Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Nodata Overwrites Real Data

If Nodata Overwrites Real Data is set to Yes, any nodata overlapping the real data values will overwrite the real data values.

#### Composite Using Alpha Band

If Composite Using Alpha Band is set to Yes, rasters will be expected to have an alpha band selected and cannot contain any palettes. Rasters will be blended according to the alpha values with the rasters they overlap instead of just being copied over them. This parameter overrides the Nodata Overwrites Data parameter because nodata values are considered completely transparent.

#### Merge Palettes

This parameter specifies how palettes will be treated when present.

When set to Yes, selected palettes in each input band set will be merged to create a single palette for the output band. When set to No, selected palettes in each input band set will be accumulated on the output band without modification.

#### Accumulate Attributes

If Accumulate Attributes is set to Yes, then the attributes from the original features will be merged onto the output raster features.

## Count Attribute

If a Count Attribute is given, then an attribute with this name will be added to each output feature, containing the number of features that were combined to create the raster feature.

## Usage Notes

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

## Example



## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: RasterMosaicFactory

## **RasterNumericCreator**

Creates a feature with a raster of the specified size with a numeric value and sends it into the workspace for processing. It is useful for creating a very large image with a user-specified width and height.

### **Parameters**

#### ***Raster Properties***

##### Rows/Columns or Extents

When specifying the size of the output raster, either the desired dimensions or the desired extents can be provided. The choice you make here will determine which additional parameters are enabled in the transformer.

To set the output raster size using dimensions, set this parameter to RowsColumns.

To set the output raster size using extents, set this parameter to Extents.

##### Number of Columns and Number of Rows

These parameters specify the number of columns and rows the raster will have. This must be at least one.

##### X Cell Origin and Y Cell Origin

The X Cell Origin and Y Cell Origin parameters specify the origin for each cell. This can be used to specify whether each cell's data point is at the lower-left or center (or somewhere else) within the cell.

##### X Cell Spacing and Y Cell Spacing

These parameters specify the spacing between cell elements. This must be greater than zero.

##### X Upper Left Coordinate and Y Upper Left Coordinate

The parameters specify the origin for the upper-left corner of the raster as a whole.

##### X Lower Right Coordinate and Y Lower Right Coordinate

These parameters specify the origin for the lower-right corner of the raster as a whole.

##### Rotation

This parameter specifies the rotation of the raster.

### ***Interpretation***

#### Interpretation

This parameter sets the type of data stored at each cell in the raster and number of bits used for that type.

### ***Data Values***

#### Min. Numeric Value and Max. Numeric Value

The parameters specify the range of values each cell in the raster can take.

#### Nodata Numeric Value

The parameter specifies the nodata value (0-1) for this raster.

#### Data Pattern

This parameter specifies which type of raster should be created:

- Single Value means each cell in the raster will be set to the maximum raster value.
- Checkered Pattern means alternating cells in the raster will be set to values between the minimum and maximum raster values.

- Checkerboard means alternating blocks of data in the raster will be set to the minimum and maximum raster values, in a standard eight-by-eight checkerboard pattern. The sizes of these blocks of data are variable, and dependent on the number of rows and columns specified. If the number of rows and columns are not multiples of eight, the remaining data will be set to either nodata values or the minimum value.

**Transformer Category**

Rasters

**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: RasterCreationFactory

## **RasterPaletteAdder**

Creates a palette from an attribute, and adds this palette to all selected bands on a raster.

Selected bands are required to have an interpretation of UINT8, UINT16, or UINT32. Note that palette entries will be discarded if they do not fit within the interpretation of a selected band. For example, when adding a palette to a UINT8 band, all keys that are greater than 255 will be dropped.

### **Parameters**

Palette Attribute

The attribute from which the palette will be read.

### **Format of Palette Attributes**

See Format of Palette Attributes in the RasterPaletteExtractor.

### **Usage Notes**

This transformer supports raster band selection. Selected bands are required to have an interpretation of UInt8, UInt16, or UInt32. The RasterSelector can be used to modify selection.

### **Related Transformers**

This transformer may be used in combination with the AttributeFileWriter to read a palette from a file. The AttributeFileWriter can read a palette file into an attribute, and this transformer can then create a palette from that attribute.

The RasterPaletteExtractor may be used to do the reverse operation and create a palette from a string attribute.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @RasterPaletteAttributes

## RasterPaletteExtractor

Creates a string representation of an existing palette and saves it to an attribute.

### Parameters

Palette Attribute

The attribute to which the palette will be written.

### Format of Palette Attributes

The general format of a palette attribute is as follows:

```
<value interpretation> [<string length>]
<key 0> <value 0>
<key 1> <value 1>
...
<key n> <value n>
```

The first line of the palette must contain the value interpretation. Valid values for the value interpretation are RGBA32, RGB24, RGBA64, RGB48, GRAY8, GRAY16, and STRING.

When an interpretation of STRING is specified, the first line may optionally specify the maximum string length of the palette values. This value must be a positive integer. If no string length is explicitly specified, a default of 32 will be assumed.

RGBA and RGB palette values consist of comma-delimited strings of integers between 0 and the maximum value of the datatype. For example, a valid RGBA32 value would be 64,128,255,255, and a valid RGB48 value would be 16384,32768,65535.

GRAY palette values consist of a single integer between 0 and the maximum value of the datatype.

STRING palette values may consist of any arbitrary text, except for the newline character.

All lines after the first are key-value pairs. Palette keys must be organized in ascending order, but they are not required to be contiguous. For example, you can have palette entries for keys 0, 2, and 4, but not 1 or 3. All missing palette entries are assumed to look up to 0 or an equivalent value, such as 0,0,0 for RGB or an empty string for string palettes.

### Palette Examples

This is an example of a color palette:

```
RGB24
0 0,49,190
1 50,255,50
2 172,0,255
3 255,0,0
```

And this is an example of a string palette:

```
STRING 10
0 Water
1 Forest
2 Commercial
3 Urban
```



## **Usage Notes**

This transformer supports raster band and palette selection. Exactly one palette must be selected on each input raster feature. The RasterSelector can be used to modify selection.

## **Related Transformers**

This transformer may be used in combination with the AttributeFileWriter to write a palette to a file. This transformer can create an attribute from a palette, and then the AttributeFileWriter can be used to write that attribute to a file.

The RasterPaletteAdder may be used to do the reverse operation and create a palette from a string attribute.

## **Transformer Category**

Rasters

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: @RasterPaletteAttributes

## **RasterPaletteGenerator**

Generates a palette out of the selected band(s) of a raster. The output raster will have the selected band(s) replaced by a new band with a palette.

### **Palette Key Interpretation Type**

This parameter allows the choice of the interpretation of the keys in the resultant palette and supports the three possible palette key interpretations: UInt8, UInt16, and UInt32.

### **Maximum Number of Palette Entries**

This parameter allows the optional specification of a maximum number of palette entries. If left as the default blank value, the number of unique keys in the bands specified by the interpretation in Palette Key Interpretation Type will determine the number of palette entries.

For example, if Palette Key Interpretation Type is set to UInt8 and Maximum Number of Palette Entries to 87, the generated palette will only have UInt8 keys and a maximum of 87 unique entries.

This operation may be lossy if an explicit maximum number of entries is used that is less than the number of unique band entries. In this case, the data values in the palette may not capture all the original band data values.

## **Usage Notes**

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

## **Transformer Category**

Rasters

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: @GenerateRasterPalettes

## RasterPaletteInterpretationCoercer

Alters the underlying interpretation of the palettes of the raster geometry on the input features, using the specified conversion options.

For example, an input raster feature with a single band with a single palette of interpretation RGB24 could be converted to a single band with a single palette of RGB64 or String data.

### Parameters

#### Destination Interpretation Type

Selects the destination interpretation along with the bit depth. Different interpretations allow for different conversion options to be used.

#### RGBA to RGB

Selects the action to perform when converting an RGBA palette to an RGB palette. The Drop the alpha band option simply discards the alpha component from the palette. The Apply the alpha band option multiplies all RGB values with their corresponding, normalized alpha value.

#### RGB to RGBA

Selects the action to perform when converting an RGB palette to an RGBA palette:

- *Create opaque alpha* adds a new alpha component which has the maximum value for the data type in all palette entries.
- *Create alpha component from nodata* adds a new alpha component which has the maximum value for the data type in all palette entries except the nodata entry. Note that when this option is selected, all selected input bands have a nodata value.

#### Convert from Color to Color

Selects the action to undertake when a conversion between different color types occurs.

- *Cast*: Uses C-style casts to convert the values.
- *Bounded cast*: Uses C-style casts to convert the values too, but also validates that the source values fit into the destination type, effectively preventing underflow and overflow; if a source value does not fit, the corresponding destination value will either be set to its type's minimum or maximum value.
- *Scale by data values*: Finds the minimum and maximum values of the source values and uses them to scale the values to the full range of the destination type.
- *Scale by data type*: Scales the source values while preserving all proportions in regard to the source and destination types' range.

#### Convert from Float to Integer

Specifies the action to perform when converting from a floating-point value to an integer.

- *Round* rounds to floating-point value to the nearest integer.
- *Ceiling* gets the next integer which is greater than or equal to the floating-point value.
- *Floor* gets the next integer which is less than or equal to the floating-point value.

### Usage Notes

Each RasterPaletteInterpretationCoercer performs a conversion on the input raster. If multiple RasterPaletteInterpretationCoercers are used in sequence, then multiple conversions will take place; data quality and translation performance may suffer.

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

### Related Transformers

The RasterInterpretationCoercer performs similar operations on the raster as a whole, such as converting 4 bands directly to RGBA.

The RasterBandInterpretationCoercer performs similar conversions on individual bands.

### Transformer Category

Rasters

**FME Licensing Level**

FME Professional edition and above

**Transformer History**

This transformer replaces the RasterDataTypeCoercer and RasterColorModelCoercer transformers.

**Technical History**

Associated FME function or factory: @ReinterpretRaster

## **RasterPaletteNodataSetter**

Identifies the nodata value on a raster feature at the palette level.

The transformer will succeed in setting the specified nodata value only if the input raster band(s) have at least one palette and a nodata key has already been set on the band.

Input features must contain raster geometries only.

### **Usage Notes**

- RGB and RGBA color models are not supported by this transformer.
- This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

### **Related Transformers**

This transformer does not set the nodata value on the band level. To set the nodata value on the bands of a raster, use the RasterBandNodataSetter.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @RasterNodata

## **RasterPaletteRemover**

Removes the selected palette(s) of a raster. If the source band has no palettes, the raster will remain unchanged.

### **Parameters**

None.

### **Usage Notes**

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @RemoveRasterPalettes

## **RasterPaletteResolver**

Resolves the palettes of the selected bands of the input raster features by using the band cell values to look up the corresponding palette values, which then replace the original band cell values in the raster.

The output raster will have no palettes on the selected bands and the band data values will reflect the original the palette values.

If the selected band has more than one palette to resolve, the first palette will be resolved into the original band and subsequent palettes will be resolved into clones of the original band, which will be added to the raster. If the palette value interpretation is a color model, there may be multiple bands generated on the raster for each palette. Palette resolving is not possible for palettes containing String values.

For example, one band with a single RGB24 palette will become three bands: a RED8 band, a GREEN8 band, and a BLUE8 band, each without a palette.

Once processing is complete, the palette will not be present on the remaining bands.

### **Usage Notes**

This transformer supports raster band selection only. Either all or none of the palettes must be resolved for each band. The RasterSelector can be used to modify selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @ResolveRasterPalettes

## RasterPropertiesExtractor

Extracts the geometry properties of a raster feature and exposes them as attributes.

The following geometry properties are exposed:

`_num_bands`  
`_num_rows`  
`_num_columns`  
`_spacing_x`  
`_spacing_y`  
`_origin_x`  
`_origin_y`  
`_rotation`  
`_min_x`  
`_min_y`  
`_max_x`  
`_max_y`  
`_cell_origin_x`  
`_cell_origin_y`  
`_upper_left_x`  
`_upper_left_y`  
`_upper_right_x`  
`_upper_right_y`  
`_lower_right_x`  
`_lower_right_y`  
`_lower_left_x`  
`_lower_left_y`

### Parameters

None.

### Usage Notes

- Input features must contain raster geometries only.
- This transformer is unaffected by raster band and palette selection.

### Transformer Category

Rasters

### Related Transformers

This transformer does not extract band or palette properties such as interpretation and nodata. To extract the band and palette properties of a raster, use the RasterBandPropertiesExtractor.



**FME Licensing Level**

FME Professional edition and above

**Technical History**

Associated FME function or factory: @RasterProperties

## RasterPyramider

Creates a series of pyramid levels for each input raster feature by specifying either the smallest pyramid level size or the number of pyramid levels to generate. Pyramid levels are created by resampling input rasters to various different resolutions.

### Output Ports

PYRAMIDS

Output raster features that comprise the levels of the pyramid generated for each input feature.

### Parameters

Smallest Level Size

If you choose Smallest Level Size, the Number of Columns and Rows in the Smallest Level fields are enabled.

Number of Levels

If you choose Number of Levels, then the Number of Levels field is enabled.

Number of Columns and Rows in the Smallest Level

If the Number of Columns and Rows in the Smallest Level is specified, then the smallest pyramid level generated will have the specified size. Each subsequent level will increase the number of rows and columns by a factor of two, until the size of the input raster is reached.

Number of Levels

If the Number of Levels is specified, then the largest pyramid level generated will have half the number of rows and columns of the input raster. Each subsequent level will decrease the number of rows and columns by a factor of two, until the specified number of levels has been generated.

Force Level Sizes to be Powers of Two

If *Force Level Sizes to be Powers of Two* is set to Yes, then the number of rows and columns in all generated levels will be powers of 2. When the smallest pyramid level size is specified, the number of rows and columns in the smallest level will actually be the smallest powers of 2 that are greater than or equal to the specified values. When the number of levels is specified, the number of rows and columns in the largest pyramid level will be the greatest powers of 2 that are less than or equal to the number in the input raster.

Note that pyramid levels will not be generated if either the number of rows or the number of columns is less than 2.

Interpolation Type

Cell values are interpolated in order to change the raster to the specified sizes.

Interpolation methods:

- Nearest Neighbor: fastest but produces the poorest image quality
- Bilinear: provides a reasonable balance of speed and quality
- Bicubic: slowest but produces the best image quality
- Average 4: performs similar to Bilinear and useful for numeric rasters such as DEMs
- Average 16: performs similar to Bilinear and useful for numeric rasters such as DEMs

Raster Index Attribute

If a Raster Index Attribute is specified, an attribute will be added to each output feature that identifies which raster it was created from. This index is zero-based, so all pyramid levels created from the first input raster will have a value of 0, all pyramid levels created from the second input raster will have a value of 1, etc.

Pyramid Level Attribute

If a Pyramid Level Attribute is specified, an attribute will be added to each output feature that identifies its level in the pyramid. The input raster

is considered to be the base of the pyramid (level 0), so the largest level that is output for a given input raster will have a value of 1, the second largest will have a value of 2, etc.

#### Number of Pyramid Levels Attribute

If a Number of Pyramid Levels Attribute is specified, an attribute will be added to each output raster indicating the number of levels in the pyramid to which it belongs

#### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

#### **Transformer Category**

Rasters

#### **Licensing Level**

FME Professional edition

#### **Technical History**

Associated FME function or factory: RasterPyramidFactory

## **RasterReader**

Reads and outputs raster features from the specified format and dataset.

Each feature that enters creates another reader.

The attributes from the incoming feature are added to the read raster features. If they both have the same name, the incoming feature's attributes will prevail, except if they start with `fme_` (in which case the raster's attributes prevail).

Features leaving the transformer will have the `fme_feature_type` and `fme_basename` attributes as read from the raster reader.

## **Parameters**

Raster Reader Type

Select the format of the raster reader.

Dataset Location

Typically the dataset is taken from an attribute in the incoming feature.

Continue on Reader Error

If an error occurs and this parameter is set to No, the translation will fail.

If an error occurs and this parameter is set to Yes, an error message will be logged, but the translation will continue. In this case, the original query feature will be output with no geometry and the last error message issued by the reader stored in the `_reader_error` attribute. Note that the `_reader_error` attribute will not be added to the schema by this transformer; this can be done using an `AttributeExposer` if desired.

## **Transformer Category**

Rasters

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: `QueryFactory`

## **RasterReplacer**

Replaces the geometry of the feature with the geometry held in the Raster Blob Attribute. The blob is decoded according to the selected raster format.

### **Parameters**

Format Selection Mode

Determines whether the writer format will be explicitly selected, or selected through an attribute.

Raster Format, Raster Format Attribute

The format used to deserialize the blob can be selected in the writer format or obtained through the Writer Format Attribute depending on the Format Selection Mode.

Raster Blob Attribute

Alternatively if the raster is not required as a geometry on the feature for future processing, you can use the AttributeFileWriter transformer to dump the Raster Blob Attribute directly to a file.

### **Usage Notes**

Note that this transformer only works on features with raster geometry. Use the GeometryReplacer to replace non-raster geometries.

### **Related Transformers**

This transformer is typically used to restore a raster previously extracted into an attribute by the RasterExtractor.

GeometryExtractor

GeometryReplacer

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

FME Factory Used: @RasterGeometry

## **RasterResampler**

Resamples an input raster using the desired dimensions, the desired cell size in ground units, or a percentage of the size.

### **Parameters**

#### ***Raster Size***

##### Size Specification

To resize the input raster by dimensions, select RowsColumns.

To resize the input raster by cell size, select CellSize.

To resize the input raster by percentage, select Percentage.

##### Number of Columns/Number of Rows

Enter values when RowsColumns is selected.

##### X Cell Spacing/Y Cell Spacing

Enter values when CellSize is selected.

##### Percentage

Specify a percentage. For example, if the input raster is 1000 rows by 800 columns and this is set to 50 percent, the resulting raster will be 500 rows by 400 columns.

#### ***Interpolation***

##### Interpolation Type

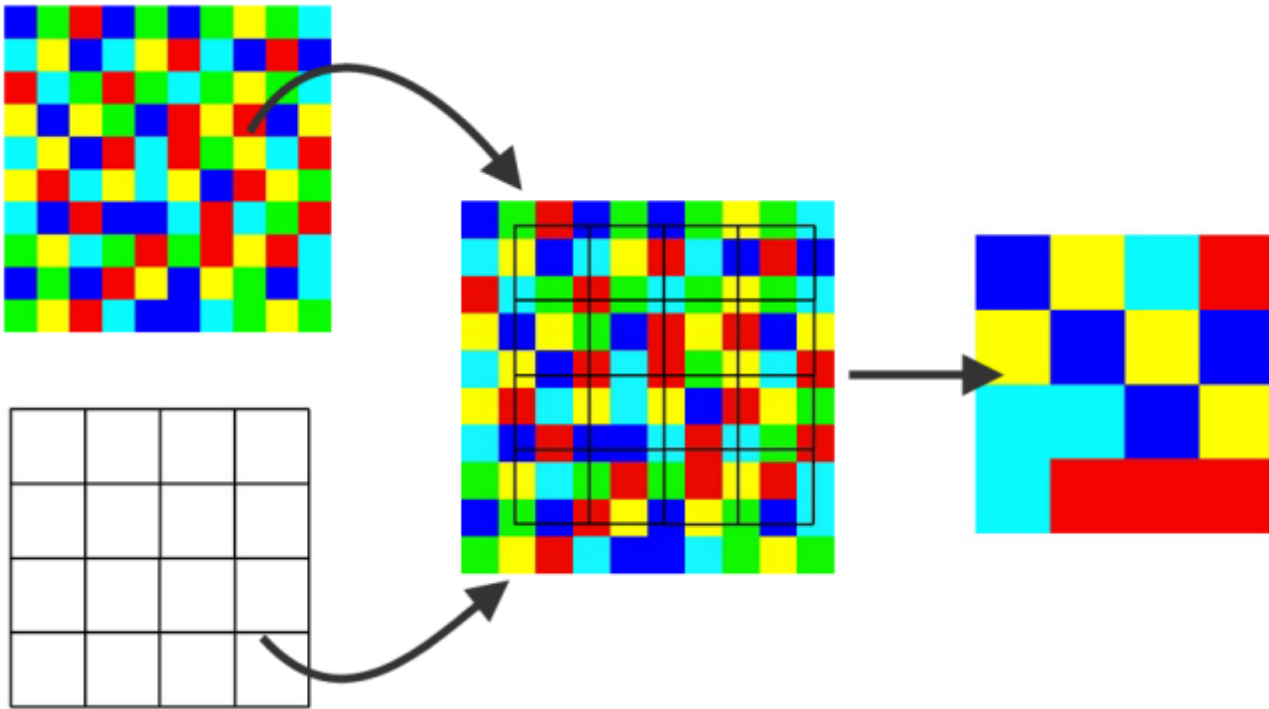
Cell values are interpolated in order to change the raster to the specified size. You can choose from these interpolation methods.

- Nearest Neighbor is the fastest but produces the poorest image quality.
- Bilinear provides a reasonable intermediate option.
- Bicubic is the slowest but produces the best image quality.
- Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

## Example



## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: @ResampleRaster

## **RasterRGBCreator**

Creates a feature with a raster of the specified size with an RGB value and sends it into the workspace for processing.

### **Parameters**

#### ***Raster Properties***

##### Rows/Columns or Extents

When specifying the size of the output raster, either the desired dimensions or the desired extents can be provided. The choice you make here will determine which additional parameters are enabled in the transformer.

To set the output raster size using dimensions, set this parameter to RowsColumns.

To set the output raster size using extents, set this parameter to Extents.

##### Number of Columns and Number of Rows

These parameters specify the number of columns and rows the raster will have. This must be at least one.

##### X Cell Origin and Y Cell Origin

The X Cell Origin and Y Cell Origin parameters specify the origin for each cell. This can be used to specify whether each cell's data point is at the lower-left or center (or somewhere else) within the cell.

##### X Cell Spacing and Y Cell Spacing

These parameters specify the spacing between cell elements. This must be greater than zero.

##### X Upper Left Coordinate and Y Upper Left Coordinate

The parameters specify the origin for the upper-left corner of the raster as a whole.

##### X Lower Right Coordinate and Y Lower Right Coordinate

These parameters specify the origin for the lower-right corner of the raster as a whole.

##### Rotation

This parameter specifies the rotation of the raster.

### ***Interpretation***

#### Create Palette

Specifies whether this raster will contain a palette.

#### Band Interpretation

Specifies the data type each cell will contain if the raster does not contain a palette.

#### Palette Key Interpretation

Specifies the key data type for each palette entry if the raster contains a palette.

#### Palette Value Interpretation

The Palette Color Type parameter specifies the value data type for each palette entry if the raster contains a palette.

### ***Data Values***

#### Min. Color Value, Max. Color Value

These parameters specify the range of values each cell in the raster can take. For grayscale color models (Gray8 and Gray16), only the red component value will be used.



## Nodata Color Value

This parameter specifies the nodata value for this raster. For grayscale color models (Gray8 and Gray16), only the red component value will be used.

Note: The Color Value parameters can be edited by clicking the colored square to the right of the text field, or by editing the contents of the field directly. The color must be specified as <red>,<green>,<blue> where each of <red>, <green>, and <blue> is a number between 0 and 1.

## Min. Alpha Value, Max. Alpha Value

The parameters specify the range of values for the alpha channel (0-1) that each cell in the raster can take.

## Nodata Alpha Value

This parameter specifies the alpha channel value (0-1) for the nodata value for this raster.

## Data Pattern

This parameter specifies which type of raster should be created:

- Single Value means each cell in the raster will be set to the maximum raster value.
- Checkered Pattern means alternating cells in the raster will be set to values between the minimum and maximum raster values.
- Checkerboard means alternating blocks of data in the raster will be set to the minimum and maximum raster values, in a standard eight-by-eight checkerboard pattern. The sizes of these blocks of data are variable, and dependent on the number of rows and columns specified. If the number of rows and columns are not multiples of eight, the remaining data will be set to either nodata values or the minimum value.

## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Technical History

FME Factory Used: RasterCreationFactory

## **RasterRotationApplier**

Applies the raster rotation angle on the input raster properties to the rest of the raster properties and data values.

The expected input is a raster with a non-zero rotation angle and the expected output is a rotated raster with a rotation angle of 0.0. It is expected that the input raster properties will be modified to conform the output raster properties for a raster rotated by the given angle.

Applying a rotation angle is primarily done for compatibility with other processing and writers that cannot handle a rotation angle.

### **Parameters**

#### Interpolation Type

Cell values are interpolated in order to change the raster to the specified sizes.

Interpolation methods:

- Nearest Neighbor: fastest but produces the poorest image quality
- Bilinear: provides a reasonable balance of speed and quality
- Bicubic: slowest but produces the best image quality
- Average 4: performs similar to Bilinear and useful for numeric rasters such as DEMs
- Average 16: performs similar to Bilinear and useful for numeric rasters such as DEMs

### **Usage Notes**

- This transformer is unaffected by raster band and palette selection.
- It is suggested that the input raster also contain a nodata value since applying the rotation often has the effect of adding nodata areas around the corners of the rotated raster. These nodata areas will be filled with 0 or black values in the absence of an input raster nodata value.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @ApplyRasterRotation

## RasterSelector

Selects specific bands and palettes of a raster for subsequent transformer operations.

The bands and palettes are selected using the band and palette indices, specified in a string. The string may either be specified explicitly or through an attribute. The format of the string is B P (separated with a space), where B is the band index and P is the palette index of the band and palette to be operated on. Indices are zero-based, so the first band or palette is at index 0.

### Parameters

#### Band and Palette List

The function will only accept alphanumeric characters and valid symbols in the code string. The code string accepts the symbols ",", ";", and ":".

Multiple palettes for a band can be specified, delimited by commas.

Multiple band-palette pairs can be specified delimited by semicolons.

The keyword ALL can be used in place of band and palette numbers to select all bands or all palettes on a certain band. Specific palettes cannot be selected on ALL bands.

### Example

all bands	ALL
all bands and all palettes	ALL ALL
first palette of the first band of a raster	0 0
first palette of the first band and first palette of the third band	0 0;2 0
the first three bands of the raster (without their palettes)	0;1;2
the first three bands of the raster (with their palettes)	0 ALL;1 ALL;2 ALL

### Usage Notes

- This function overrides any existing selection on the raster. All unspecified bands and palettes on a raster after selection are treated as unselected for future operations. Selection is not considered by writers, so use the RasterBandSeparator and filtering to write subsets of the bands and palettes as individual features.
- It is possible to select bands without selecting the attached palettes.
- It is NOT possible to select palettes without selecting the owning band.
- Specific palettes cannot be selected on ALL bands; for example, ALL 2.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @SelectRaster

## RasterSingularCellValueCalculator

Performs an arithmetic operation on two operands: the cell values of a raster and a numeric value. The numeric value may either be a constant or an attribute.

### Input

Input features have the following restrictions:

- All input features must have raster geometry.
- Bands may not contain a palette.

### Parameters

#### Operation

This parameter sets the operation that will be performed:

+	add
-	subtract
*	multiply
/	divide

For example, if you select the plus sign (+), the two input rasters A and B will be added together (and therefore, the output raster will be A+B).

#### Operand Order

This parameter specifies the order of the operands. This parameter only impacts non-commutative operations such as subtraction.

#### Preserve Interpretation

If this parameter is set to Yes, each output band will have the same interpretation as the corresponding input band. If Preserve Interpretation is set to No, the interpretation of each output band will be automatically determined.

Note that when converting between different data types, a Bounded Cast is used. As a result, when a calculated value does not fit in the specified destination interpretation, the corresponding destination value will either be set to the minimum or maximum value possible in the destination data type.

### Usage Notes

You can modify band selection with the RasterSelector.

### Related Transformers

- Nodata values can be set with the RasterBandNodataSetter or removed with the RasterBandNodataRemover.
- Palettes may be resolved using the RasterPaletteResolver or removed using the RasterPaletteRemover.
- You can use the RasterCellValueCalculator to operate on a pair of rasters.

### Technical History

FME Factories used: RasterEvaluationFactory, RasterSplitterFactory, RasterMergerFactory, TestFactory

FME Functions used: @RasterProperties, @Tcl2, @RemoveAttributes

## **RasterSubsetter**

Reduces a raster to a subset of its original size. This is essentially a clipping operation using pixel bounds instead of ground coordinates.

### **Parameters**

Start Column, Start Row

These parameters specify the position in the input raster from which the subset will be taken.

Number of Columns, Number of Rows

These parameters specify the size of the subset in cells to be taken from the raster. They do not include any padding values. These values must be greater than 0 for the input to be valid.

Padding Parameters

The padding parameters specify how many rows and columns of padding in cells should be placed around the subset portion of the raster.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

### **Transformer Category**

Rasters

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: @SubsetRaster

## **RasterTiler**

Splits each input raster into a series of tiles by specifying either a tile size or a number of tiles.

### **Parameters**

#### Tile Size or Number of Tiles

Choose how the input raster should be split.

#### Number of Columns, Number of Rows

These parameters specify the size of the subset in cells to be taken from the raster. They do not include any padding values. These values must be greater than 0 for the input to be valid.

#### Number of Horizontal Tiles, Number of Vertical Tiles

These parameters specify the number of tiles into which the input rasters will be split. These values must be greater than 0.

#### Force Equal Sized Tiles

This parameter controls behavior when the raster size is not a multiple of the requested tile size or number of tiles.

No: Tiles on the right or bottom edge of the raster may be smaller than other tiles.

Yes: All tiles will be the same size.

Tiles that go off the edge of the raster will be padded with the nodata value if one is set on the band; if no nodata value is set, RGB rasters will have an alpha band added to identify the padding regions.

For example, suppose you have a raster of size 1000 columns and a selected column tile size of 256. When this option is No, three tiles of 256 columns and one tile of 232 columns will be generated. When this option is Yes, four tiles of 256 columns will be generated, where the last tile contains 24 columns of padding.

#### Raster Index Attribute

An attribute will be added to each output tile that identifies which raster it was created from. This index is zero-based, so all tiles created from the first input raster will have a value of 0, all tiles created from the second input raster will have a value of 1, and so on.

#### Tile Column Attribute, Tile Row Attribute

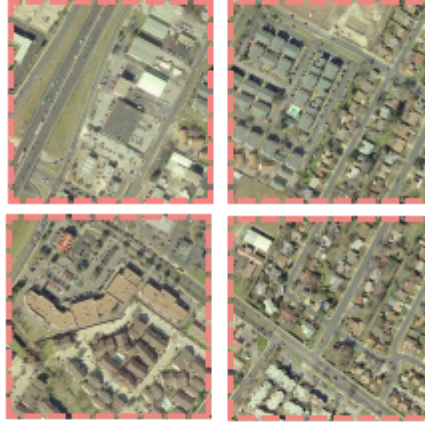
Attributes will be added to each output tile that identify the position of that tile in the input raster. These indices are zero-based.

Tile row 0, tile column 0 corresponds to the upper-left tile.

### **Usage Notes**

This transformer is unaffected by raster band and palette selection.

## Example



## Transformer Category

Rasters

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: RasterSubsetFactory

## Recorder

Saves a copy of all the features that enter to a disk file.

This disk file can later be "played back" in the subsequent workspace by using the Player, or viewed using the FME Universal Viewer.

### Parameters

#### Output Feature Store File

Specifies the output .ffs file.

#### Recording Mode

- If the mode is RECORD\_PASS\_THROUGH, the transformer will record each feature it receives and immediately pass it through to the rest of the FME for further processing.
- If the mode is RECORD, the transformer will record each feature it receives but wait until the end of translation before it sends all recorded features through to the rest of the FME for processing. This method is useful because if one of the features causes the FME translation to fail, the feature file will be complete and usable, whereas if a failure occurs when using the RECORD\_PASS\_THROUGH mode, the feature file may not be usable.

#### Compression Level

Indicates how much compression will be applied to the output recording. The higher the number, the more compression, the smaller the output file, and the more CPU that will be expended to make it.

#### Output Byte Order

Specifies how the bytes of the features will be laid out in the disk file. If the file is only going to be used on the same platform where it was produced, the NATIVE choice is the best. Otherwise, it can be forced to use BIG or LITTLE endian.

#### Store Scanned Schema

When this option is set to Yes, schemas will be recorded and stored in the output FFS file. If it is set to No (default), schemas will not be passed along to the output file.

#### Password

Provides a level of password protection for the created file. Before the file can be read, the password will have to be reentered.

#### Feature Type Attribute

Sets the feature type for the feature inside the FFS file. If no attribute is given, a default feature type is provided.

### Transformer Category

Infrastructure

### Related Transformers

Player

### Technical History

Associated FME function or factory: RecorderFactory



## ReprojectAngleCalculator

Converts a given angle from one coordinate system to another. The transformer uses the first coordinate of the passed-in feature and returns the angle of the specified line at the determined angle in the destination coordinate system. It is useful for converting data such as text rotations that are established in many systems in the destination coordinate system units.

To calculate the new angle, the transformer uses the:

- input length
- input angle
- input point
- source (reader) coordinate system
- destination (writer) coordinate system

and then calculates a second point in the source coordinate system by taking the first point and advancing length units in the angle heading. It then reprojects both points to determine the new angle.

### Usage Notes

This function assumes all coordinate systems are Cartesian.

### Parameters

#### Original Line Length

Distance specified in source coordinate system coordinates. You should enter a length that is small enough to avoid distortion, but large enough to get meaningful results (for example, 1 m might be a good choice, although a range of 1 cm to 100 m would be acceptable). If your source system is in degrees, you will have to adjust the length to produce measurable results.

#### Original Line Angle

Angle of distance in source coordinate system. The angle is measured counterclockwise from horizontal and is specified in degrees.

#### Source and Destination Coordinate Systems

The name of the reader and writer coordinate systems.

#### Reprojected Angle Attribute

The calculated value is put into the attribute specified by the Reprojected Angle Attribute parameter.

### Transformer Category

Coordinate Systems

### Technical History

Associated FME function or factory: @ReprojectAngle

## **ReprojectLengthCalculator**

Converts a given distance from one coordinate system to another. The transformer calculates the reprojected length of a line starting at the first coordinate in the feature, with the given length and angle. Both the length and angle parameters are specified in the source coordinate system.

This transformer is useful for converting data such as text heights and widths that are measured in many systems in the destination coordinate system units.

### **Parameters**

Original Line Length

Distance specified in source coordinate system coordinates.

Original Line Angle

Angle of distance in source coordinate system. The angle is measured counter clockwise from horizontal and is specified in degrees.

Source and Destination Coordinate Systems

The name of the reader and writer coordinate systems.

Reprojected Length Attribute

The calculated value is put into the attribute specified by the Reprojected Length Attribute parameter.

### **Transformer Category**

Coordinate Systems

### **Technical History**

Associated FME function or factory: @ReprojectLength

## Reprojector

Reprojects feature coordinates from one coordinate system to another.

You should use this transformer only in rare instances. It is better to use the workspace Navigator view to set the source and destination coordinate systems for the translation.

### Parameters

#### Source and Destination Coordinate System

If the feature had a coordinate system, that system is used as the source of the reprojection, and the Source Coordinate System parameter to this transformer is ignored.

If the feature did not have a source coordinate system, and the Source Coordinate System parameter was unset, then the transformer only sets the coordinate system of the feature to the destination coordinate system, and the coordinates of the feature remain unchanged.

Note: If the source coordinate system is not fixed and may change from feature to feature, and the features themselves have been tagged with a coordinate system from the reader that produced them, then a single Reprojector may still be able to be used. In such a case, both the source and destination coordinate system can be set to the same value – the destination coordinate system – and the desired behavior will be accomplished.

#### Allow Arc and Ellipse Stroking

If this parameter is set to No, then only the center points of the arcs or ellipses will be reprojected. Otherwise, if the feature came from a reader which supported this option, the arcs are converted to lines and ellipses to polygons and then reprojected. This option has no effect on advanced geometry arcs and ellipses.

#### Interpolation Type (Raster)

The Interpolation Type affects only raster data. Cell values are interpolated in order to change the raster to the specified size.

- **Nearest Neighbor** is the fastest but produces the poorest image quality.
- **Bilinear** provides a reasonable balance of speed and quality.
- **Bicubic** is the slowest but produces the best image quality.
- **Average 4** and **Average 16** have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Cell Size (Raster)

The Cell Size applies only to raster features.

- **Stretch Cells:** The cell size of the raster will be adjusted to maintain the same number of rows and columns in the reprojected raster as there were in the input raster.
- **Square Cells:** The number of rows and columns as well as the spacing will be changed to maintain approximately the same cell ground area and form square cells where the horizontal and vertical cell sizes are equal. Like the Square Cells option, Preserve Cells will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.
- **Preserve Cells:** Like the Square Cells option, this option will change both the number of rows and columns and the spacing to maintain cell ground area, but will also try to preserve the original cell aspect ratio, taking into account any warping caused by the reprojection.

### Dynamic Coordinate Systems

If the destination coordinate system is specified as "\_AZMEA\_" or "\_AZMED\_", each input feature is reprojected to either an equal area or equal distance projection appropriate for that feature, respectively. In general, this causes a new coordinate system to be defined for each input feature.

Each feature remembers which specific equal distance or equal area coordinate system it has, and can be safely reprojected back to a normal (non-dynamic) coordinate system.

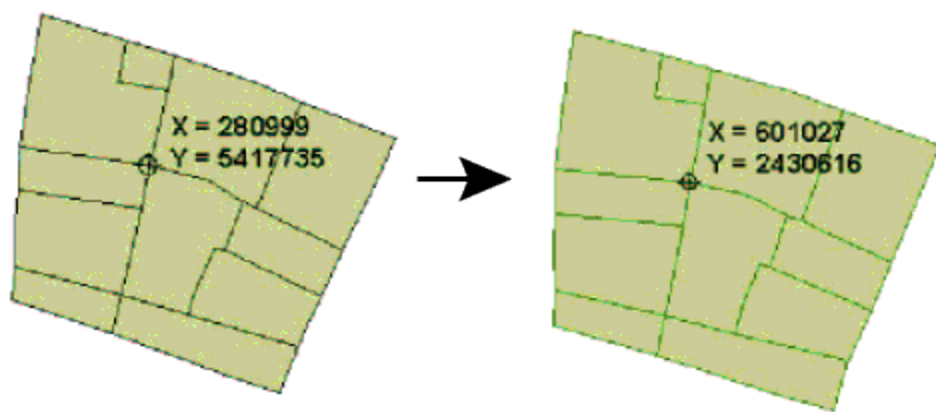
As an example:

1. There is an input feature representing a point on the earth in LL-WGS84 (normal lat/long).
2. The point is reprojected to \_AZMED\_ via a Reprojector with source LL-WGS84 and destination \_AZMED\_.
3. The x and y coordinates of the point are extracted into x1, y1.
4. Set  $x2 = x1 + 1000$ , and  $y2 = y1$ .
5. Add a vertex to the point to make the line  $(x1, y1) \rightarrow (x2, y2)$ .
6. Reproject back to LL-WGS84 via a Reprojector with source \_AZMED\_ and destination LL-WGS84.

Note that the source is IGNORED here. We just chose \_AZMED\_ to help remember what is going on. Sometimes people prefer to set the source and destination both to LL-WGS84.

7. Now we have changed our point into a line extending 1km east of the original point, in lat/long.

### Example



### Usage Notes

- This transformer works with both raster and vector data.
- This transformer is unaffected by raster band and palette selection.

### Transformer Category

Coordinate Systems

### Technical History

Associated FME function or factory: @Reproject

## Rotator

Rotates features in a counterclockwise direction about the specified point by the Rotation Angle parameter (measured in degrees).

The rotation is around the Z axis, and does not impact any elevations that may be present on the feature.

### Parameters

Each of the parameters may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

#### Rotation Angle

Specifies the angle that the feature will be rotated, measured in degrees counterclockwise.

#### X Origin, Y Origin

The x- and y-coordinates about which features are rotated. If not specified, the features will be rotated about the origin. These parameters do not apply when rotating raster geometries.

---

Note: Raster geometries do not use the supplied origin for rotation. Rasters always rotate around the upper left corner, which is the implied origin.

---

### Transformer Category

Manipulators

### Technical History

FME Factory Used: @Rotate2D

## RubberSheeter

Performs warping operations on the spatial coordinates of features. It is used to adjust a set of observed features so they more closely match some set of reference features. This transformer applies a different transformation to each **OBSERVED** vertex, depending on its distance to nearby **CONTROL** vectors. It produces good corrections when the distortions in the **OBSERVED** data are not constant.

### Input Features

Two sets of features must be routed into this transformer:

- Features that enter through the **CONTROL** port represent the control features used to compute the corrections.
- Features that enter through the **OBSERVED** port are the features that will be corrected.

Each **CONTROL** feature represents a control vector whose start point is at some location in the original **OBSERVED** data space, and whose end point is at the corresponding location in the desired output data space. The control vector represents the correction required to go from the observed vertex to the desired vertex. (Control vectors with only one point are interpreted as a requirement that this location not change from the observed dataset to the reference dataset. This is often referred to as a tie point.)

### Output Ports

The modified **OBSERVED** features are output via the **CORRECTED** port.

### Parameters

#### Distance Exponent

Specifies how the strength of a correction will be affected by its distance from an observed point. A value of 2 will cause the strength to be decreased proportional to the square of the distance.

#### Max Distance

Indicates the influence of control vectors. Any control vector start point farther than the specified distance from the point being operated on will have no effect in the correction computation. If Max Distance is not specified (or is 0), then all control vectors will be used for correcting every point.

#### Max Number of Influencing Vectors

Indicates that only the closest given number of vectors will have an effect on any point being warped. If not specified (or if 0), then all control vectors will be used for correcting every point.

### Comparison to the AffineWarper

The **AffineWarper** transformer provides similar functionality but computes an affine (scale, rotation, and offset) transformation based on **CONTROL** vector features and applies this transformation to the **OBSERVED** features to generate output. This makes the **AffineWarper** more appropriate for cases when the entire set of **OBSERVED** data requires a single transformation.

### Geometry Handling

If the **Geometry Handling Advanced** setting is set to **Enhanced** in the workspace, then arcs and ellipses entering via the **OBSERVED** port are stroked prior to performing the transformation. Otherwise, only the center point of arcs and ellipses are used in the transformation.

### Case Study

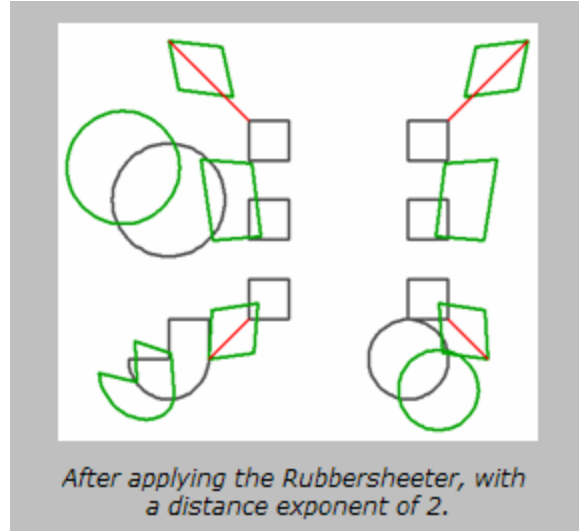
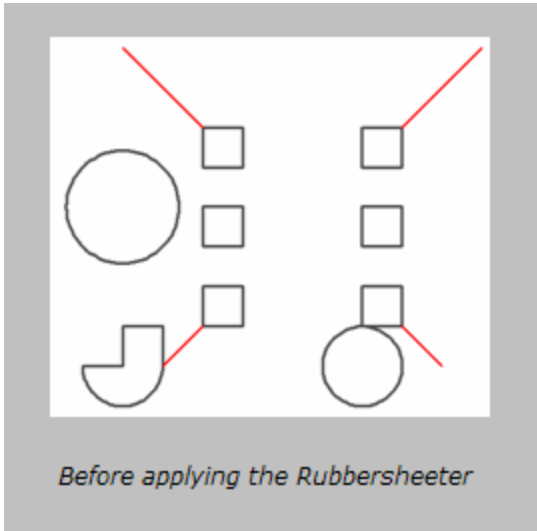
[Click here](#) to read how FME's Rubbersheetter was used to combine Manukau City Council's existing parcel-dependent cadastral data with a newer "survey accurate" national digital cadastre.

### Common Questions

*I have a line feature only some of whose points fall inside the Max Distance. How will it be warped?* Only vertex points that lie inside the Max Distance will get warped. Therefore, part of your line will be warped, part of it will not.

Why doesn't FME warp the entire feature when one of its points falls inside the Max Distance? Because this could cause topological networks to become broken. Working on a point-by-point basis, connections will never be broken because the common point on the connecting feature will also get warped (even if the rest of that feature doesn't).

### Example



### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: WarpFactory

## Sampler

Preserves either a total number of features or a sampling of features, depending on the Sampling Type selection. The remainder of the input features are discarded.

This transformer is typically used during testing to reduce data volume by arbitrarily discarding data.

## Parameters

### Group By

When specified, this parameter affects the behavior of the two sampling types. For example, say the Sampler receives a set of 15 features. Using the Group By parameter, the Sampler breaks the features into 3 groups.

- Group 1 has 3 features
- Group 2 has 5 features
- Group 3 has 7 features

This table shows how sampling amount and type affect the Group By results:

Sampling Amount	Sampling Type	Result
4	Every Nth Feature	<ul style="list-style-type: none"><li>• 0 features from Group 1 will be sampled</li><li>• 1 feature from Group 2 will be sampled: the 4th feature</li><li>• 1 feature from Group 3 will be sampled: the 4th feature</li></ul>
10	First N Features	<ul style="list-style-type: none"><li>• 3 features from Group 1 will be sampled</li><li>• 4 features from Group 2 will be sampled: the first 4 features</li><li>• 4 features from Group 3 will be sampled: the first 4 features</li></ul>

### Sampling Amount

The Sampling Amount is the number of features to keep: either a total number of features, or a sampling of features.

If you enter 1, no data will be discarded: the same number of features will be the same for input and output.

If you enter 0, all input data will be discarded.

### Sampling Type

Sampling Type determines whether to keep a set number of features, or a sampling of features.

## Example

Sampling Amount	Sampling Type	Result
2	Every Nth Feature	Every second feature will be kept.
10	First N Features	Only the first 10 features will be kept. All subsequent features will be consumed by the transformer.

## Transformer Category

Filters



## **Technical History**

Associated FME function or factory: SamplingFactory

## Scaler

The Scaler scales objects to make them bigger or smaller.

A separate multiplier must be set for each of the X, Y, and Z axes. For two-dimensional features, the Z multiplier is ignored.

Each parameter may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

If Text Scaling is set to "Location and Size", then any text features will have their height scaled in addition to their location.

This transformer supports raster band and palette selection. The RasterSelector can be used to modify selection.

Note: This transformer simply multiplies all coordinate values with the scale factors specified by the user and that if the object that's being scaled is not on or near the 0,0 point of the coordinate system, using the Scaler will also introduce a shift to your data.

## Usage Notes

This transformer works with both raster and vector data.

## Transformer Category

Manipulators

## Technical History

Associated FME function or factory: @Scale

## SchemaMapper

Converts the existing schema (data model) of features to a new structure, based on mappings defined in an external lookup table.

This technique is very useful when the mappings are potentially complex or when they need to be maintained by someone who is not familiar with FME. Using an external lookup table to define these mappings simplifies the completion of these tasks.

The schema mapping lookup table, used by the SchemaMapper transformer, defines a series of conditions that are to be met (filters), and a series of actions that will be executed when the conditions are met.

The lookup table may come in different formats such as: a comma-separated or plain text file; a spreadsheet (Excel or Google); or a database such as Oracle, PostGRES, Informix, SQL Server, etc.

If an incoming feature matches the rules in any row of the table, the potential actions of the transformer on that feature are listed by type as follows:

### Map Feature Types:

Feature types – as defined by `fme_feature_type` – are mapped from their existing value, to one which defines the new feature type. For example:

Roads → Center Lines

### Map Attributes:

One or more attributes on the feature are mapped by renaming their existing attribute name (s) to a new one. For example:

Name\_of\_Road → RoadName

### Set New Attributes:

A new attribute is created whose name and value are defined in a lookup table.

### Transformer Ports

Features that have an action carried out on them are output via the MAPPED port. Otherwise, they are output via the UNMAPPED port.

In addition to the feature type, the geometry of each feature is left untouched.

### SchemaMapper Uses

You can use the SchemaMapper for:

**Domain Mapping:** where attributes values can be remapped according to a well-defined domain or lookup table. For example:

- Primary Route US Highway
- Secondary Route Interstate Highway
- Primary Route County Road

**Dynamic Translations:** where workspaces are created to handle any data structure

**Automated Schema Mapping:** where the manual connections between source and destination schemas are done automatically using an external lookup table.

Note: A schema mapping table (`domainSchema.csv`) is often derived from a database metadata document such as ESRI's XML database schema description, which can be exported from ArcCatalog for any selected geodatabase (Export > XML Workspace Document > Schema Only).

### Example: Using the SchemaMapper transformer

The following example describes a typical scenario

## Feature Type Mapping

Using an external lookup table (.csv file), you can map feature types to simplify a schema by mapping the old feature type to a new feature type:

Old Feature Type	New Feature Type
River	Water
Lake	Water
Canal	Water
Road	Transportation
Railway	Transportation
Airport	Transportation

## Attribute Mapping

Using an external lookup table, you can map attributes to new names:

Old Attribute	New Attribute
River_Name	WaterName
River_Alt_Name	AltWaterName
River_Country	WaterCountry
River_Length	WaterSize
River_Owner	WaterOwner

## Filters

You can use filters to set your mapping rules. For example, if you want to map an "old pipe type" to a "new pipe type" You can set the rule (filter) according to the size of the pipes:

Old Pipe Type	Pipe Size Attribute	Pipe Size	New Pipe Type
Gas	PipeSize	6	GasSmall
Gas	PipeSize	12	GasMedium
Gas	PipeSize	24	GasLarge
Water	Diameter	6	WaterSmall
Water	Diameter	12	WaterMedium
Water	Diameter	24	WaterLarge
Sewage	PipeDiam	6	SewageSmall
Sewage	PipeDiam	12	SewageMedium
Sewage	PipeDiam	24	SewageLarge

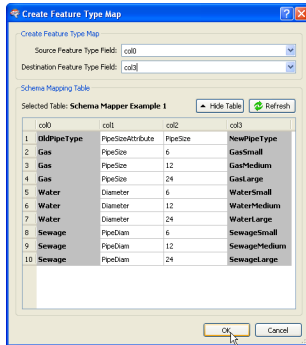
## Example Using Feature Type Mapping and Filtering

You can use the SchemaMapper wizard to define the filters and the type of mapping that you need. You accomplish this task by creating actions to be executed on the source schema.

To open the SchemaMapper wizard, click the Properties button of the transformer. The SchemaMapper Parameters dialog opens.

1. Specify the format and location of the schema mapping table (lookup table) to be used. You may want to edit format parameters but you can use the defaults that Workbench provides. Click Next.
2. If a dataset has more than one table, specify the table to be used. Click Next.
3. On the Create Actions pane, click Add and select the action type. For this example, we'll select Feature Type Map.

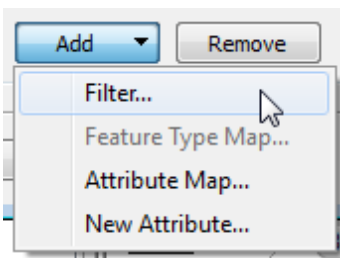
- Select the Source and Destination Feature Type Fields. You can show/hide the table columns, and refresh the contents.



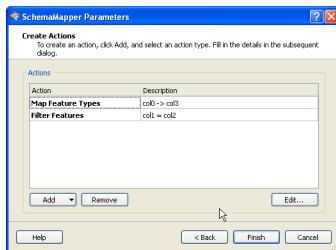
- Click OK to return to the Create Actions pane, which now shows the Feature Type Map action you just set.

Note: To transform your source schema to a new schema, you can create more than one action depending on the mapping rules that you want to apply to the existing schema.

- To add another action, select from the Add drop-down menu. In this example, we'll select Filter to display the Create Filter dialog:



- Select the table columns in the Attribute Name and Value Fields. By default, Blank Attribute Values are ignored but you can choose to include them.
- Click OK to return to the Create Actions dialog. The Action column now shows two actions: Map Feature Types and Filter Features:



- Click Finish to set the SchemaMapper parameters.

## Usage Notes

This transformer usually requires implementation help from our Professional Services department.

## fmepedia

- For a good basic introduction to the SchemaMapper, see [Crouching Schema Hidden Dragon](#).
- This [fmepedia](#) page contains descriptions and examples.
- One of our user conference workshops contained a lot of additional information about the SchemaMapper: see from page 22.

## Transformer Category

Database

**FME Licensing Level**

FME Professional edition and above

## SecondOrderConformer

Performs a second-order conformal transformation on the feature's geometry. Depending on the input geometry, a 2D or 3D transformation is performed. Transformed features are passed to the TRANSFORMED port.

The transformation results in the x and y coordinates being modified by:

$$x' = C + E(x-A) - F(y-B) + G((x-A)^2 - (y-B)^2) - 2H(x-A)(y-B)$$

$$y' = D + E(y-B) + F(x-A) + H((x-A)^2 - (y-B)^2) + 2G(x-A)(y-B)$$

$$z' = z + I$$

This transformation is non-linear. Thus, to maintain the integrity of the data, we only perform transformations on the points defining geometries. For example, a 4-point quadrilateral polygon will remain a 4-point polygon after transformation, although a pure transformation would convert its lines into curves. If curve approximates from lines are desired, use a Densifier before this transformer.

Due to the non-linear nature of the transformation, true 3D geometries (such as Box and EnclosedSurface) are not supported by this transformer. All true-3D geometries are passed to the INVALID port.

Each of the parameters may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull down list.

### Transformer Category

Manipulators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @SecondOrderConformal

## SectorGenerator

Point features are input with the following data contained in the chosen attributes: site name, sector name (optional), azimuth, and radius. (Alternatively, a universal radius can be specified, instead of a field name.) Within points having the same site name, points have their geometry changed to polygons approximating sector shapes. The center point of each sector will be the average of all input points for that site. The start and end angles of each sector will be the average of the sector's azimuth (defined in degrees clockwise from north), and the azimuth(s) of the adjacent sector(s). If multiple sectors within a site have the same azimuth, their shapes will be identical (and created as though there were only one sector with the given azimuth in the site).

If a point enters the SectorGenerator and is farther from other points in its site than the specified Maximum Distance Between Site Points, it will be output on the DISTANT\_POINTS port.

Input points not containing all required attributes, or having invalid values in the azimuth (which must be nonnegative and less than 360) or radius (which must be positive) will be output on the INCOMPLETE\_POINTS port.

Input features with non-point geometry are output on the ILLEGAL\_GEOM port.

If the AZIMUTH\_LINES port is used, then for each input point a line will be generated with one endpoint at the original point coordinates and the other endpoint a radius-length away in the direction of the azimuth.

If the SITE\_POINTS port is used, then for each input site a point will be generated which is the average of all input points for the site.

If a Sector Name Attribute is specified, then each feature entering the SectorGenerator will be checked to see if its sector name is already in use within its site. If it is, then it will be output on the EXTRA\_POINTS port.

If any Group By attributes are given, then each group will be treated independently, as though each group were in a different dataset and were processed by a separate SectorGenerator.

If it is known that each site's input features occur consecutively in the input, select Yes for Input Clustered by Site Name in order to conserve the use of resources. (Processing will occur on each site as it is input, instead of waiting for all features before starting.) This option cannot be chosen if any Group By attributes are specified.

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, the sectors are output as rich geometry polygons with arcs or ellipses. Otherwise, the sectors are output as polygons that consist of only points and no arc segments.

## Transformer Category

Surfaces

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: SectorFactory



## SelfIntersector

Checks each feature and removes self-intersections.

### Parameters

Segment Count Attribute

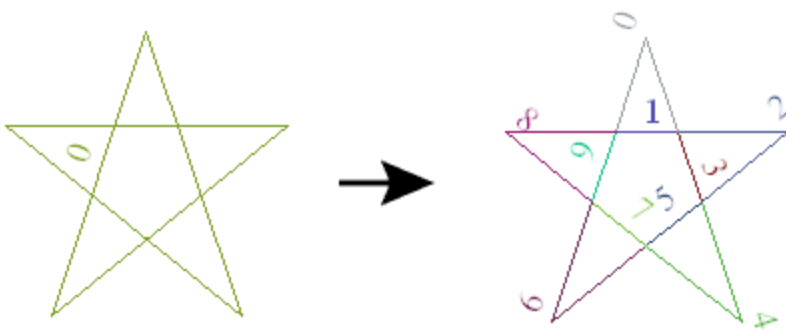
If a linear feature self-intersects, it is split into separate features, one per non-intersecting piece. Each resulting feature will have the total number of pieces created from the original feature added as an attribute.

If an area feature is self-intersected and in so doing it results in more than one area, the resulting areas are gathered into an aggregate. This behavior may change in future releases.

### Geometry Handling

If the Geometry Handling advanced setting is set to "Enhanced" in the workspace, arcs that exist in the input (as arcs or ellipses) will be preserved as arcs in the output; they will otherwise be stroked to lines before intersections are processed.

### Example



### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: IntersectionFactory

## SherbendGeneralizer

Uses the Sherbend algorithm to simplify lines by reducing unnecessary details based on the analysis of the line's bends.

Sherbend is a constraint-based algorithm that preserves the spatial relationship of the lines and points in the input data. The Sherbend algorithm iteratively generalizes the bends in a line by using the *Diameter* parameter to select bends for generalization. The generalization process may eliminate, reduce, or combine bends, while resolving conflicts.

The strategy for generalizing bends in a line is as follows:

- Calculate the area of a reference circle whose diameter is specified by the *Diameter* parameter.
- For each line, determine the locations of the bends.
- For each bend, calculate its circumference. Next, construct a circle with the same circumference. Finally, determine the adjusted area of the bend, which is taken to be 75% of the area of that circle.
- For each bend, generalize the bend if its adjusted area is below the area of the reference circle and spatial constraints are met.
- Repeat the above steps until there are no more bends to generalize.

### Input Ports

- **LINES:** Input lines for generalization. They are assumed to not self-intersect or intersect with another line or point.
- **POINTS:** Input points for the sidedness constraint. When the "Sidedness" constraint is enabled, these points will prevent a bend generalization if that would change the spatial relationship between the bend and the points.

### Output Ports

- **LINES:** The generalized lines.
- **CONFLICTS:** Bends that, if generalized, would violate the selected constraint.
- **INVALID:** Invalid input features will be output to the INVALID port.

### Parameters

#### Group By

Only lines and points in the same group are subject to constraint checking. If no group is specified, all lines and points are placed in the same group.

#### Diameter

This parameter specifies the diameter of the reference circle described at the beginning of this documentation, which roughly describes the width of a bend below which the bend will be generalized. Different lines can have different diameters specified as an attribute. The bigger the diameter, the more likely bends will be generalized.

#### Spatial Constraints

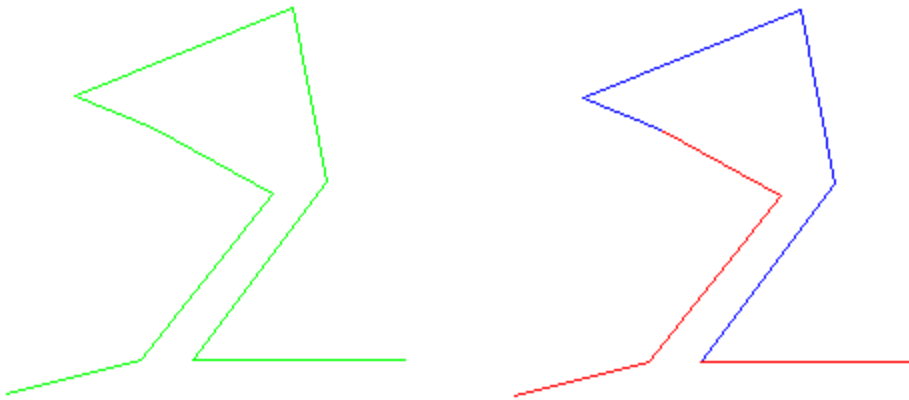
Enables spatial constraints, which are only applied to lines and points in the same group.

- *None:* constraints will not be applied.
- *Self Intersection* prevents a line from intersecting with itself, with the assumption that input lines do not self-intersect when entering SherbendGeneralizer.
- *Self, Line-Line Intersection* prevents a line from intersecting with itself or another line, with the assumption that no input line self-intersects or intersects another line when entering SherbendGeneralizer.
- *Self, Line-Line Intersection, Sidedness*, in addition to maintaining non-intersecting lines, maintains the relative positioning of all lines and points. For example, if a line is entirely on the right side of another line, that line will remain entirely on the right side of that other line after the generalization process.

In this diagram, the blue bend cannot be generalized as it would violate the "Sidedness" constraint:



In this diagram, the blue bend cannot be generalized as it would violate the "Self Intersection" constraint:



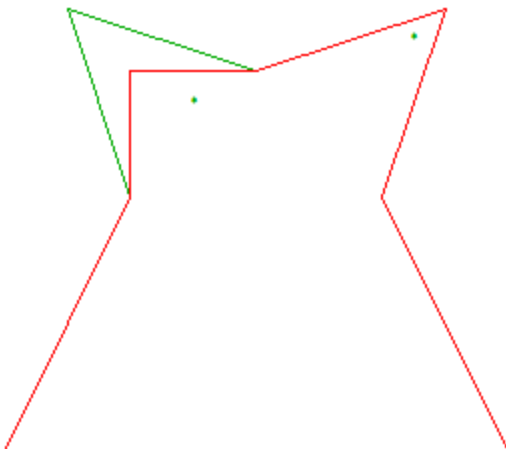
### Preserve Endpoints in Closed Lines

This parameter, if set to *No*, will re-order (rotate) the coordinate list of each closed line in an attempt to improve the quality of generalization. To preserve juncture connectivity, the transformer must ensure that the starting and end coordinates of every line are kept stationary. Therefore, if it is important to keep the positions of the first and last coordinates in a closed line (perhaps because they are on a juncture), this parameter should be set to *Yes*.

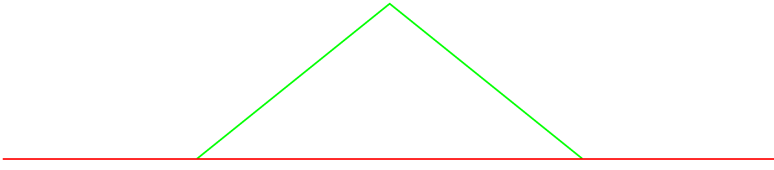
If this parameter is set to *Yes*, the endpoints of a line will not be moved. This behavior allows the preservation of juncture connectivity.

### Examples

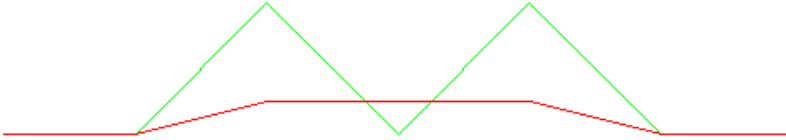
In this example, a bend is reduced (green = input, red = output):



In this example, a bend is eliminated:



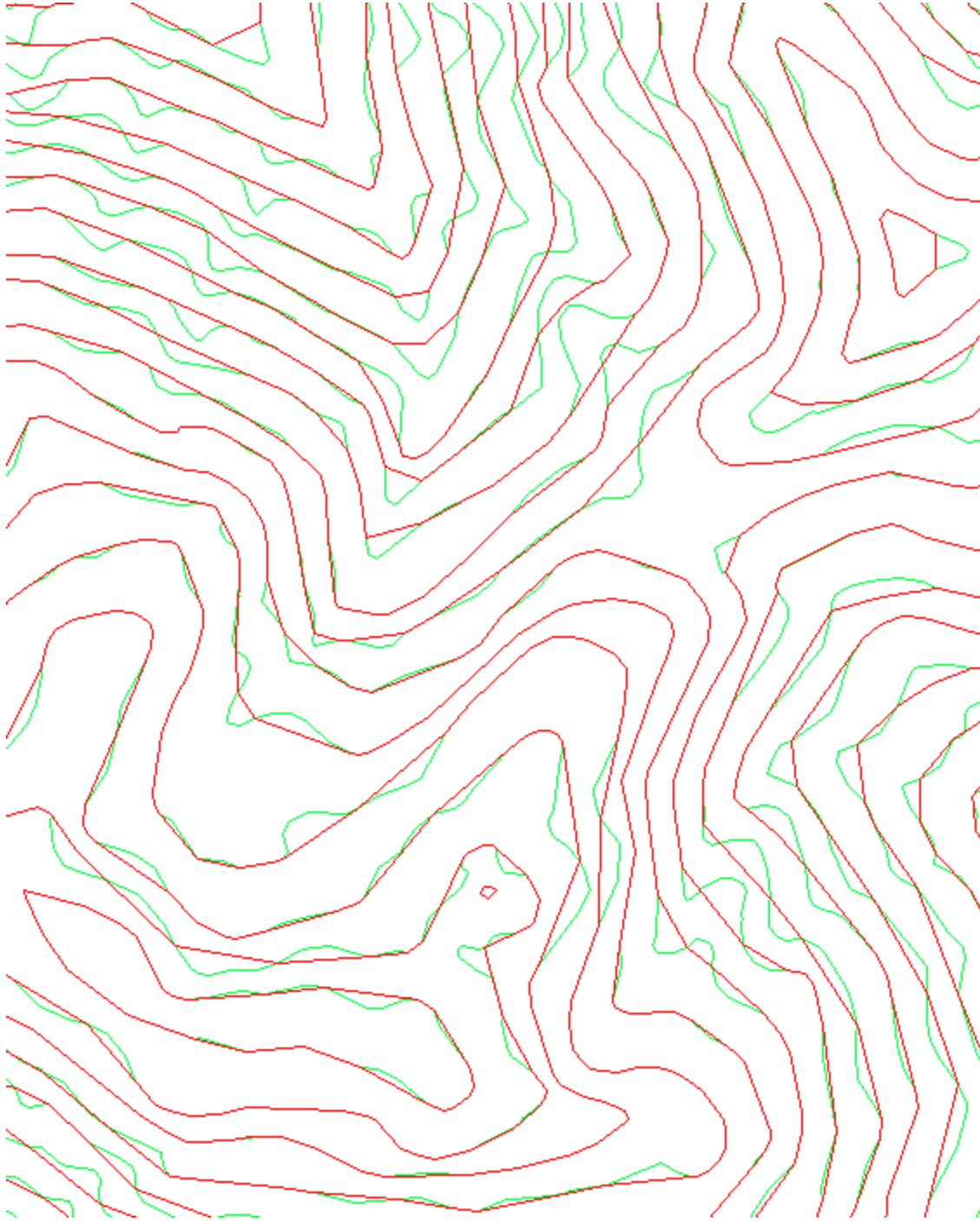
In this example, three bends are combined into one:



The following diagram illustrates the generalization process on a single line in a real-world dataset:



This example illustrates the generalization process on a set of contours:



### **Additional Information**

The aim of line generalization is to reduce the details on a line for representation at a smaller scale. While the well-known Douglas-Peucker algorithm is good at reducing the number of points in a line, it is not very good at preserving the shape or the spatial relationship of the line relative to other entities. In comparison, the Sherbend algorithm is well suited for the generalization of natural features (contours, lakes, rivers, wooded areas, etc.) because it preserves the general shape and, if spatial constraints are enabled, also the spatial relationship between the input entities. The Douglas-Peucker algorithm with a small tolerance is often used before or after Sherbend to further reduce the number of points to further fulfill the goals of generalization.

### **Performance and Usage Notes**

- The Sherbend algorithm iteratively detects and generalizes bends, and then detects and resolves spatial conflicts. The generalized lines from one iteration are passed to the next iteration until the lines cannot be generalized further. Due to this iterative process, the algorithm is time-intensive, which is a tradeoff to improved accuracy and quality of generalization.
- Constraint checking is a highly time-intensive operation. Disabling constraints should greatly improve the performance.
- To generalize each feature independently, consider using the Generalizer transformer.

### **External Resources**

See also: <http://www.fmepedia.com/index.php/SherbendGeneralizer>

### **Transformer Category**

Manipulators

## ShortestPathFinder

Computes the shortest path from a source node to a destination node in a network based on the length of the input or the weight of edges.

### Input

SOURCE: The source node in a network.

DESTINATION: The destination node in a network.

There can only be one SOURCE port and one DESTINATION port for each group.

### Output

All non-linear features and extra nodes are output through the INVALID port.

If found, the features that are part of the shortest path are output through the PATH port in an order such that the start point of the first feature is the source node and the end point of the last feature output is the destination node. If a reversed line is used in the shortest path, the value of the attribute specified in Direction Attribute will be "opposite". Otherwise, the value will be "same".

All other linear features that are not used as part of the shortest path are output through the UNUSED port.

### Parameters

#### Group By

The default behavior is to use the entire set of features as the group. This option allows you to select attributes that define which groups to form.

#### Weight Type: By Length or By One Attribute

If Weight Type is set to By Length or By One Attribute, then the weight of each input line is set to the length of the line or the attribute value specified in the Forward Weight Attribute. In this case, the algorithm will only consider the original orientation of the lines when finding the shortest path.

#### Weight Type: By Two Attributes

If Weight Type is set to By Two Attributes, then the shortest path algorithm will consider both directions of the input lines. The original orientation of the input line has the weight specified in the Forward Weight Attribute and the reversed orientation of the input line has the weight specified in the Reverse Weight Attribute.

#### Forward Weight Attribute

This parameter is used in conjunction with By One Attribute and By Two Attributes. See above.

#### Reverse Weight Attribute

This parameter is used in conjunction with By Two Attributes. See above.

#### Direction Attribute

This attribute holds the value of the shortest path.

### Usage Notes

Only linear features with non-negative weight attribute values are allowed if the Weight Type is set to By One Attribute or By Two Attributes. If a feature does not have the attribute specified in the Forward Weight Attribute or the Reverse Weight Attribute, a zero weight is used for the line.

### Transformer Category

Network

### Related Transformers

NetworkFlowOrientor

NetworkTopologyCalculator

StreamOrderCalculator

StreamPriorityCalculator

**FME Licensing Level**

FME Professional edition and above

**Technical History**

FME Factory Used: NetworkFactory



## Snapper

Brings end points or vertex points of features together if they are within a certain distance of each other and (optionally) if they have one or more attributes in common.

The difference between the AnchoredSnapper and the Snapper is that anchor features are considered to be accurate and consequently do not move.

## Output Ports

- SNAPPED: Features whose geometry is changed by the transformer.
- UNTOUCHED: Features that leave the transformer without being changed.

## Parameters

### Group By

If Group By attributes are selected, features are only snapped to other features with the same values in the group by attributes.

### Snapping Type

When this parameter is set to *End Point Snapping*, the transformer:

- Snaps end points of lines together if they are within the **<tolerance>** distance of each other.
- Snaps point features together (or snaps them to a linear feature).

Area features will not be altered by the transformer when run in this mode.

When two features are snapped together, the feature that entered the factory most recently is the one that is modified.

When this parameter is set to *Vertex Snapping*, the transformer does the following:

- It performs vertex-to-vertex snapping for vertices that are within **<tolerance>** of each other.
- Snaps point features together (or snaps them to a linear feature).

Area features are altered by this operation as its vertices are snapped.

When two features are snapped, the feature that entered the factory most recently is the one that is modified.

### Snapping Tolerance

Snapping Tolerance specifies the distance that the snapping occurs between features. This distance is in ground units.

### Add Additional Vertex

This parameter is enabled only if Snapping Type is End Point Snapping. It controls how lines are modified when they are snapped.

- NEVER: the endpoint of a line is moved when it is snapped and no additional vertex is added.
- ALWAYS: the original end point (start point) of the line becomes the second from the end (start) and a new vertex is added to complete the snap.
- FORWARD\_ONLY: a new vertex is added only when doing so creates an angle greater than 90 degrees with the original line segment. In this case, if adding the vertex would cause a less than 90-degree angle, the old end point is still moved.

### Save Short Lines

Any features entering the transformer whose length is less than or equal to the tolerance will be treated specially: they will be output as UNTOUCHED, and other features (but not other short features) will be able to snap to them. If the option is not selected, features like this will collapse to a single point and will be dropped.

## Geometry Handling

If the Advanced setting Geometry Handling is set to Enhanced in the workspace, arcs are snapped as linear features, and ellipses are not snapped; otherwise, arcs and ellipses are both snapped as point features located at their respective center points.

## Usage Notes

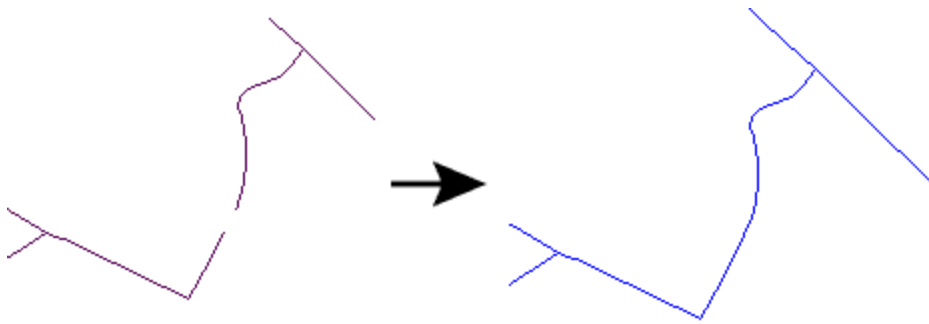
A short cleanup step is performed after snapping. This step will remove duplicate points, and may create aggregates to preserve overlapping, directed segments.

Any feature that undergoes dimensional collapse as a result of being snapped will be logged as "degenerate" and dropped. Dimensional collapse refers to a line or area that becomes a point, or an area that becomes a line.

## Related Transformers

- The Snapper seems to snap to the first suitable candidate that is found and then ignores the other possible candidates. If this is not what you're looking for, the CoordinateRounder may be useful instead.
- You can clean up a dataset by using a Snapper before trying more complex actions, such as building polygons. Other transformers that are often used in this context are the Extender and Intersector.
- The AnchoredSnapper transformer provides slightly different functionality by identifying a set of features which will not be moved and will be snapped to by another set of features.

## Example



## Transformer Category

Geometric Operators

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: SnappingFactory

## Snipper

Shortens the geometry of a line feature by snipping off specified distances, indices or measure values from the ends. It operates on features with simple line geometry and polygons without holes.

The parameters specify a starting and ending point for the snipping. After execution, the feature's geometry will be a line representing the portion of the original line between those two positions, inclusive of the endpoints. New coordinates will be generated at the beginning or end of the line, if there are not already coordinates at exactly the specified positions. If the line contains three-dimensional coordinates, the Z value at each endpoint will be interpolated linearly from the original feature's coordinates between which the endpoint exists.

### Parameters

#### *Mode*

The mode you choose determines which parameters are available and how they are interpreted. Therefore, only the mode parameters are listed here.

In all modes, features containing geometry other than polygons or lines (for example, points and donut polygons) will be passed through this transformer untouched.

#### Snipping Mode

**Distance (Value) or Distance (Percentage):** The amount to snip from the beginning and end of the line can be specified as either a measurement in ground units or a percentage of the line's entire length, starting from the first coordinate. The placeholder value "-1" may be used to specify an ending position corresponding to the original line's final vertex. Each of these parameters may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

If the measurement mode is 3D, and the linear feature has Z coordinates, all measurements will be taken as a distance from one vertex to the next in 3D space. Otherwise, only the X and Y coordinates will be considered, and measurements will be planar distances between vertices.

**Measure (Relative to Start Point):** The values specified in the Starting Location and Ending Location parameters refer to the absolute difference in the measure value of the start point of the original line and the start or end points of the resulting line respectively. These parameters must have values greater than or equal to zero. If Measure Name field is blank, the default measure values are used. Otherwise, the measure values with the name specified in Measure Name field is used instead.

**Measure (Value):** The values specified in Starting Location and Ending Location parameters refer to the measure values at the start and end points of the resulting line. If Measure Name field is blank, the default measure values are used. Otherwise, the measure values with the name specified in Measure Name field is used instead.

**Vertex:** The vertices from the original line which are to form the first and last vertices of the resulting line are specified as a numeric index, with "0" being the first vertex of the line. Negative numbers measure vertices relative to the last point in the line, with the value "-1" used to specify the vertex which is the last vertex of the line's geometry, "-2" the second last vertex, and so on. Each of these parameters may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

### Geometry Handling

If the Geometry Handling parameter (under Advanced Workspace Settings) is set to Enhanced in the workspace, arcs and ellipses can be snipped; they are otherwise left untouched.

### Usage Notes

The Measurement Mode parameter does not have any effect on two-dimensional lines, or when the Mode is Vertex.

### Transformer Category

Linear Referencing

### Transformer History

This transformer replaces the DistanceSnipper and VertexSnipper.

### Technical History

Associated FME function or factory: @Snip, @Tcl2



## **Sorter**

Sorts features by a selected attribute's value. The features leave the transformer in the order specified, and are output through the SORTED port.

This transformer tries to balance speed and memory usage by storing some features in memory and some in a temporary disk file.

### **Parameters**

#### Sorting Attributes

After you place and connect the transformer, choose the attribute(s) to sort.

Each attribute can be sorted alphabetically or numerically and can be sorted either in ascending or descending order. By default, the sort direction is in ascending order. Alphabetic ordering will sort according to ASCII ordering rules.

Attributes are sorted according to their position in the list. You can reorder the list using the Move Up and Move Down buttons.

If an attribute name is specified but is not present on a feature, the sorting algorithm will consider that feature to have a null or zero value for that attribute.

### **Related Transformers**

ListSorter

### **Transformer Category**

Collectors

### **Technical History**

Associated FME function or factory: SortFactory

## SpatialFilter

Filters features based on spatial relationships. Each input CANDIDATE feature is compared against all BASE features, based on the given spatial tests to meet. Features that pass any test are output through the PASSED port; all other features are output through the FAILED port.

**Note:** If you need to determine the relationships between the BASE and CANDIDATE features, or if you have many BASE features, you should use the `SpatialRelator` transformer. To determine relationships based on proximity, use the `NeighborFinder` transformer.

For example, if you have a group of features that you want to check against a polygon to isolate those that are inside, the polygon feature would be directed into the BASE port and the group of features would be directed into the CANDIDATE port. The PREDICATE parameter would be CONTAINS. Those features that are inside the polygon would come out the PASSED output port and those which were not inside would come out the FAILED output port.

### Input

- BASE: Features that will be compared.
- CANDIDATE: Features that will be used to compare with BASE candidates.

### Output

- PASSED: CANDIDATE features where at least one of the specified predicates was TRUE with at least one of the BASE features.
- FAILED: CANDIDATE features where all of the specified predicates were FALSE with all of the base features.

### Parameters

#### Group By

If Group By attributes are specified, candidates are only compared against bases with the same values in these attributes.

#### Tests to Perform

Defines which tests to perform.

#### Use Bounding Box

Defines whether the tests are performed using the features' true coordinates or its bounding boxes.

#### Base Type

Defines whether a single base or multiple bases will be given. If this parameter is set to **Bases First**, the SpatialFilter assumes that all BASE features enter before any CANDIDATE features.

#### Merge Attributes

Defines whether attribute merging will take place. If this is set to **yes**, every CANDIDATE that matches a BASE receives that BASE's attributes. The result is an operation known as a Spatial Join.

#### Attribute Prefix

Defines a prefix to add onto all attributes that are merged from BASEs to CANDIDATEs.

#### Predicate Attribute

Specifies an attribute that will be added onto all output PASSED features, which will contain the name of the spatial test that the feature passed.

#### Pass Criteria

This parameter specifies whether a candidate must have a predicate match against all bases or against at least one base.

## Curve Boundary Rule

This attribute specifies how to determine the boundary of curve and multicurve geometries. The **Default Rule** is that any curve endpoints that occur an odd number of times in the geometry as a whole will be considered its boundary – that is, a linear loop (a line whose start point equals its endpoint) will not have any boundary. The other rule specifies that the curve's or multicurve's boundary is the set of all its endpoints.

## Usage Notes

See Spatial Relations Defined for more information on spatial predicates and an illustration of spatial relationships.

## Geometry Handling

If the Geometry Handling advanced setting is set to Enhanced in the workspace, arcs and ellipses are stroked prior to the testing of any relationship predicates – they are otherwise regarded as points located at their respective center points.

## Transformer Category

Filters

## Licensing Level

FME Base Edition

## Technical History

Associated FME function or factory: SpatialFilterFactory

**Note: This transformer uses functionality from the GEOS library (<http://geos.refractions.net/>), which is distributed under the terms of the Free Software Foundation's LGPL license (<http://www.gnu.org/licenses/lgpl.html>). To comply with the terms of the LGPL, Safe Software Inc. has set up a website (<http://www.safe.com/foss>) to provide further information.**

## SpatialRelator

Determines topological (spatial) relationships between sets of features. It tags – but does not otherwise change – features when they have certain relationships, such as touches, overlays, intersects, and so forth. Use this transformer when you need to determine the relationships between features or if you have many BASE features. If you only need to determine if the features are related and you only have a few BASE features, the SpatialFilter transformer is more efficient.

All BASE features are output through the OUTPUT port, with a new list attribute appended. Each input CANDIDATE feature is compared against the BASE features, based on the spatial tests specified in the Tests to Perform parameter. When one of the comparisons is true, an entry is added to the BASE's list attribute as follows:

```
<LIST_NAME>{i}.de9im = [DE9IM string]
<LIST_NAME>{i}.pass{0} = [true PREDICATE 1]
<LIST_NAME>{i}.pass{1} = [true PREDICATE 2]
...
<LIST_NAME>{i}.pass{n} = [true PREDICATE n+1]
```

Additionally, all attributes of the matching CANDIDATE will be added to the list.

As well, each BASE receives the attributes of the CANDIDATES that passed the relationship, resulting in an operation referred to as a Spatial Join. When attributes are merged down from CANDIDATE to BASE features, existing attributes are not replaced. Therefore if the CANDIDATES and BASEs have attributes with the same name, then the values will not be transferred down. This can be worked around by renaming (AttributeRenamer), prefixing (AttributePrefixer), or removing (AttributeRemover) attributes to avoid name collisions.

## Input

- BASE: Features that will be compared and then output with a tagged list describing the relationships.
- CANDIDATE: Features that will be used to compare with BASE candidates, but not output.

## Output

- OUTPUT: These are the BASE features with the new attributes added. One list entry is made for each of the candidates that has at least one matching predicate.

## Parameters

### Group By

This parameter is used to indicate that only BASEs and CANDIDATEs that have the same value for certain attributes should be compared. That is, if Group By attributes are specified, candidates are only compared to bases that have the same values in these attributes.

### Tests to Perform

This parameter lists the spatial predicates that will be used for comparisons between the base and candidate features. These can either be selected directly from the list, or taken from attribute values.

In addition to the predefined predicates, you may also test relationships using arbitrary 9-character masks. Such masks consist of the rows of a Dimensionally Extended 9 Intersection Matrix. Note that in order use these masks with the SpatialRelator, you must assign them to an attribute, and include the value of that attribute in the "Tests to perform" clause (you cannot specify them directly).

For more information about predicates, see [Spatial Relations Defined](#).

### List Name

This parameter specifies the name of the list attribute that will be added to the BASE features.

### Related Candidate Count Attribute

This attribute specifies the name of an attribute that will be added to each base, which stores the number of candidates with which the base had at least one true relationship.



## Attributes that Must Differ

This attribute controls which attributes must have different values before a match is declared.

## Curve Boundary Rule

This parameter specifies how to determine the boundary of curve and multicurve geometries. The Default Rule is that any curve endpoints that occur an odd number of times in the geometry as a whole, will be considered its boundary – that is, a linear loop (a line whose start point equals its endpoint) will not have any boundary. The other rule specifies that the curve or multicurve's boundary is the set of all its endpoints.

## Calculate Cardinality of Intersections

If this parameter is specified, then for each candidate that matches a base, three attributes will be added to the corresponding list entry: `card_point`, `card_line`, and `card_area`. These count the number of points, lines, and areas that comprise the intersection of the base and candidate. For instance, a point is counted if two polygons touch at a vertex, a line is counted if they touch at an edge, and an area is counted if they overlap.

## Usage Notes

See Spatial Relations Defined for more information on spatial predicates and an illustration of spatial relationships.

## Geometry Handling

If the Geometry Handling Advanced setting is set to Enhanced in the workspace, arcs, and ellipses are stroked prior to the testing of any relationship predicates; they are otherwise regarded as points located at their respective center points.

## Transformer Category

Calculators

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: `SpatialRelationshipFactory`

**Note: This transformer uses functionality from the GEOS library (<http://geos.refractorions.net/>), which is distributed under the terms of the Free Software Foundation's LGPL license (<http://www.gnu.org/licenses/lgpl.html>). To comply with the terms of the LGPL, Safe Software Inc. has set up a website (<http://www.safe.com/foss>) to provide further information.**

## Spatial Relations Defined

This topic includes detailed information on how spatial predicates are defined:

- Background
- Boundary, Interior, and Exterior
- String Representations of Intersection Matrices
- Definitions of Predicate Attributes

[Click here to see an example that illustrates different spatial relationships.](#)

## Spatial Predicates

### Background

The definitions of each Predicate attribute are given in the tables included under this heading. This section also gives specific definitions of boundaries, exteriors, and interiors as they apply to specific feature types and explains the concept of an intersection matrix.

Each feature – whether it's a point, line, or polygon – has a definition of an INTERIOR, BOUNDARY, and EXTERIOR. The EXTERIOR is everything that is not on the BOUNDARY or the INTERIOR.

### Boundary, Interior, and Exterior

BOUNDARY	Points	Empty set.
	Lines	<p><b>CURVE_BOUNDARY_RULE ENDPOINTS_MOD2</b></p> <p>The boundary is the set of all endpoints that occur an odd number of times. For a simple linear feature (that is, not closed), the boundary is comprised of the start and end points, unless the line is closed (the start and end are the same point), in which case the boundary is the empty set. (This is the default if CURVE_BOUNDARY_RULE is unspecified.)</p> <p><b>CURVE_BOUNDARY_RULE ENDPOINTS_ALL</b></p> <p>The boundary is the set of all endpoints, regardless of the number of times they occur in the geometry.</p>
	Polygons	The border of a polygon, including the border of the holes.
INTERIOR	Points	The point location.
	Lines	The entire line except its boundary as determined above.
	Polygons	The inner surface of the polygon.

### Dimensionally Extended 9 Intersection Matrix

The comparison of two features produces a 3 x 3 matrix known as the Dimensionally Extended 9 Intersection Matrix (DE-9IM), shown below:

		candidate		
		interior	boundary	exterior
base	interior	$\chi_0$	$\chi_1$	$\chi_2$
	boundary	$\chi_3$	$\chi_4$	$\chi_5$
	exterior	$\chi_6$	$\chi_7$	$\chi_8$

The value of each element of the matrix indicates the dimension of the geometry produced by intersecting the given parts of the two features. The dimension is one of the following:

- 1 there is no interaction
- 0 the intersection produces a point
- 1 the intersection produces a line
- 2 the intersection produces a surface

So for instance, if  $C_1$  is 1, then the intersection of the base's interior with the candidate's boundary produces a line. This could occur when both features are polygons and they overlap.

Each of the predicates can be defined in terms of what the intersection matrix of the two features must look like. For this, use a pattern matrix. Each element of the pattern matrix can be one of the following:

- \* the value of this element may be anything (-1, 0, 1, or 2)
- T the value of this element must be 0, 1, or 2
- F the value of this element must be -1
- 0 the value of this element must be 0
- 1 the value of this element must be 1
- 2 the value of this element must be 2

The pattern matrix for the *disjoint* predicate is:

F	F	*
F	F	*
*	*	*

This means that neither feature's interior or boundary may interact with the other's interior or boundary.

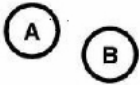

### String Representations of Intersection Matrices

Any intersection matrix can be represented as a 9-character string. To generate the string representation of a particular intersection matrix, take each element starting from the top-left, going right-to-left for each row.

For example, the string representation of the intersection matrix for the *disjoint* predicate (as seen above) is "FF\*FF\*\*\*".

### Definitions of Predicate Attributes




Each of the supported predicates is described below, along with some associated examples and pattern matrices. Note that the examples are not exhaustive: there may be entirely different situations where a given predicate is true. In the examples, the base is labeled "A" and the candidate is labeled "B".

Predicate	Example	Description	Pattern Matrix									
<b>INTERSECTS</b>		The two features are not disjoint, as defined below.										
<b>DISJOINT</b>		The boundaries and interiors do not intersect.	<table border="1"> <tr> <td>F</td> <td>F</td> <td>*</td> </tr> <tr> <td>F</td> <td>F</td> <td>*</td> </tr> <tr> <td>*</td> <td>*</td> <td>*</td> </tr> </table>	F	F	*	F	F	*	*	*	*
F	F	*										
F	F	*										
*	*	*										
<b>EQUALS</b>		The features have the same boundary and the same interior.	<table border="1"> <tr> <td>T</td> <td>*</td> <td>F</td> </tr> <tr> <td>*</td> <td>*</td> <td>F</td> </tr> <tr> <td>F</td> <td>F</td> <td>*</td> </tr> </table>	T	*	F	*	*	F	F	F	*
T	*	F										
*	*	F										
F	F	*										

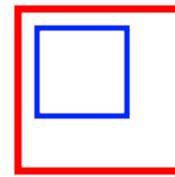
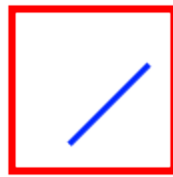
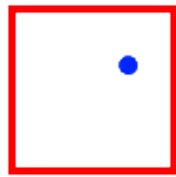
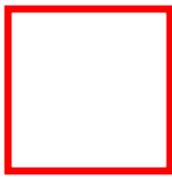
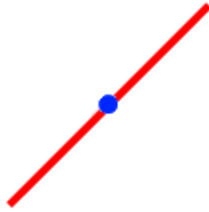
Predicate	Example	Description	Pattern Matrix									
<b>TOUCHES</b>		The boundaries may intersect or one boundary may intersect the other interior.	<table border="1"> <tr><td>F</td><td>T</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table> <p>or</p>	F	T	*	*	*	*	*	*	*
		F	T	*								
		*	*	*								
*	*	*										
The interiors do not touch.	<table border="1"> <tr><td>F</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>T</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table> <p>or</p>	F	*	*	*	T	*	*	*	*		
F	*	*										
*	T	*										
*	*	*										
Undefined for point/point.	<table border="1"> <tr><td>F</td><td>*</td><td>*</td></tr> <tr><td>T</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table>	F	*	*	T	*	*	*	*	*		
F	*	*										
T	*	*										
*	*	*										
<b>CROSSES</b>		The interiors intersect and the base's interior intersects the candidate's exterior. Or in the case of line/line, the intersection of the interiors forms a point.	<table border="1"> <tr><td>T</td><td>*</td><td>T</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table> <p>or</p>	T	*	T	*	*	*	*	*	*
		T	*	T								
*	*	*										
*	*	*										
Undefined for point/point or area/area	<table border="1"> <tr><td>Q</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table> <p>(for two lines)</p>	Q	*	*	*	*	*	*	*	*		
Q	*	*										
*	*	*										
*	*	*										
<b>OVERLAPS</b>		The interiors intersect, but neither feature is contained by the other, nor are the features equal.	<table border="1"> <tr><td>T</td><td>*</td><td>T</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>T</td><td>*</td><td>*</td></tr> </table> <p>or</p>	T	*	T	*	*	*	T	*	*
		T	*	T								
*	*	*										
T	*	*										
Undefined for point/line, point/area, or line/area.	<table border="1"> <tr><td>I</td><td>*</td><td>T</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>T</td><td>*</td><td>*</td></tr> </table> <p>(for two lines)</p>	I	*	T	*	*	*	T	*	*		
I	*	T										
*	*	*										
T	*	*										
<b>CONTAINS</b>		The interiors intersect and no part of the candidate intersects the base's exterior. (inverse of WITHIN)	<table border="1"> <tr><td>T</td><td>*</td><td>*</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> <tr><td>F</td><td>F</td><td>*</td></tr> </table>	T	*	*	*	*	*	F	F	*
T	*	*										
*	*	*										
F	F	*										
<b>WITHIN</b>		The interiors intersect and no part of the base intersects the candidate's exterior. (inverse of CONTAINS)	<table border="1"> <tr><td>T</td><td>*</td><td>F</td></tr> <tr><td>*</td><td>*</td><td>F</td></tr> <tr><td>*</td><td>*</td><td>*</td></tr> </table>	T	*	F	*	*	F	*	*	*
T	*	F										
*	*	F										
*	*	*										

<b>Predicate</b>	<b>Example</b>	<b>Description</b>	<b>Pattern Matrix</b>
<b>&lt;DE-9IM string&gt;</b>		In addition to the predicates listed above, spatial relations can also be specified using a string representation of an intersection matrix.	

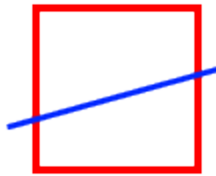
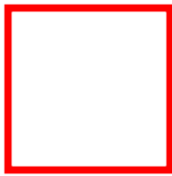
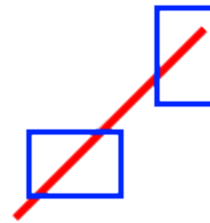
### Spatial Relationship Examples

Spatial Relations	Base	Candidates		
				

Contains



Crosses



Equals







## SpikeRemover

Cleans up feature geometries by removing spikes in 2D.

The transformer looks at every pair of line segments made up of three consecutive distinct points. If the angle (in degrees) between two line segments is less than or equal to the specified maximum angle, then the middle point is a spike and is removed.

If the **Maximum Spike Length** is specified, then the transformer will skip line segments longer than this length; otherwise, all line segments are considered.

If the geometry of a feature is a path, the transformer removes spikes between consecutive path segments as well. For a polygon or donut, if the start/end point is a spike, then it is also removed. The end result is still a polygon/donut. Any polygons, donuts, paths or lines that are part of a collection of geometry will also be processed.

The transformer will also remove any duplicate points.

The transformer is not effective when the line contains many deviations other than spikes. In such cases, it is recommended to first clean up the features using the Generalizer transformer with Douglas-Poiker algorithm.

Features that are cleaned up will be output through the CHANGED port. Duplicate points and spikes will be output through the FLAGGED port. Any untouched features will be output through the UNCHANGED port.

### Example



### Transformer Category

Geometric Operators

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: @SpikeRemoverFactory

## SQLCreator

Generates FME features from the results of a SQL query against a database. One FME feature is created for each row of the results of the SQL Query.

### Parameters

#### Reader Format and Dataset

Select the Reader format and dataset, including any format-specific parameters.

#### Coord. System

You can leave the default, or use a selection from the Coordinate System Gallery.

#### SQL Statement

Specify the SQL query using the text editor.

#### Attributes to Expose

Enter the names of attributes to expose on the features created by the query. The attributes will be output in the same sequence as specified in the list.

**Note:** By default, the attributes of the resulting features are hidden. You can specify which attributes to expose by entering the attribute names. Click the browse button next to the Attributes to Expose parameter. You can also use a SQL statement to populate the list by pressing "Populate from SQL Query..." and entering a SQL query. The columns from the first matching feature will be used to populate the attributes list.

### Transformer Category

Database

### Technical History

Associated FME function or factory: QueryFactory

## SQLExecutor

Performs SQL queries against a database.

One query is issued to the database for each feature that enters the transformer. The results of the query are then output through the RESULTS port.

### Input Ports

- TRIGGER: Features that trigger a SQL query to be executed.

### Output Ports

- RESULTS: Features that result from the SQL queries.
- TRIGGER: Input TRIGGER features with an additional attribute (`_matched_records`) that contains the number of features generated as the result of SQL query triggered by that feature.

### Parameters

Reader Format and Dataset

Select the Reader format and dataset, including any format-specific parameters.

Coord. System

You can leave the default, or use a selection from the Coordinate System Gallery.

SQL Statement

Specify the SQL query using the text editor.

Attributes to Expose

Enter the names of attributes to expose on the features created by the query. The attributes will be output in the same sequence as specified in the list.

**Note:** By default, the attributes of the resulting features are hidden. Specify which attributes to expose by entering the attribute names. Click the browse button next to the Attributes to Expose parameter. You can also use a SQL statement to populate the list by pressing "Populate from SQL Query..." and entering a SQL query. The columns from the first matching feature will be used to populate the attributes list.

Combine Attribute Parameter

Result Attributes Only: The result feature attributes consist solely of the query results.

Keep Trigger Attributes if Conflict: The result feature attributes are a combination of both the query results and the trigger feature's attributes. If there is a conflict, attribute values are taken from the trigger feature.

Keep Result Attributes if Conflict: The result feature attributes are a combination of both the query results and trigger feature's attributes. If there is a conflict, attribute values are taken from the query results.

### Transformer Category

Database

### Technical History

Associated FME function or factory: QueryFactory

## SQLQuerier

Performs SQL queries against a database. One query is issued to the database for each feature that enters the transformer. The results of the query are then output.

### Parameters

#### Reader Format and Dataset

Select the Reader format and dataset, including any format-specific parameters.

#### Coord. System

You can leave the default, or use a selection from the Coordinate System Gallery.

#### SQL Statement

Specify the SQL query using the text editor, or choose an attribute that contains an SQL query.

#### Attributes to Expose

Enter the names of attributes to expose on the features created by the query. The attributes will be output in the same sequence as specified in the list.

**Note:** By default, the attributes of the resulting features are hidden. You can specify which attributes to expose by entering the attribute names. Click the browse button next to the Attributes to Expose parameter. You can also use a SQL statement to populate the list by pressing "Populate from SQL Query..." and entering a SQL query. The columns from the first matching feature will be used to populate the attributes list.

### Transformer Category

Database

### Technical History

Associated FME function or factory: QueryFactory

## StatisticsCalculator

Calculates statistics based on a designated attribute of the incoming features.

If a feature does not contain an attribute with the specified name, or this attribute does not contain a valid number, then it will be treated as having an attribute containing a null string. Numbers that begin with '0' will be treated as octal values. Numbers that begin with '0x' will be treated as hexadecimal values.

### Parameters

#### Group By

If Group By attributes are chosen, statistics will be calculated independently within each group of features. This can be used to create a pivot-table-like analysis of values in a data stream.

#### Attribute to Analyze

The list of attributes is created when you connect the transformer to an incoming feature. Choose an attribute from the pull-down list.

#### Pass-Through Features

Yes: The INPUT features will all be passed through the STATISTICS output, with all the statistics attributes added onto them.

No: A single new feature will be output containing the statistics attributes, and the original features are consumed. (If features are grouped, the latter will emit a single feature for each group encountered in the input data.)

If no input is received, no features will be output, regardless of the setting for this parameter.

#### Attribute Statistics

Each of the following statistics will be output in the respective attribute, if one is given:

- Minimum: The numerical minimum, unless at least one value is non-numeric, in which case this will be the lexical minimum.
- Maximum: The numerical maximum, unless at least one value is non-numeric, in which case this will be the lexical maximum.
- Median: The middle value when the values are listed in order if the number of values is odd, or the average of the two middle values if the number of values is even. If there is at least one non-numeric input value, then the list is sorted lexically, and the first of the two middle values is taken as the median if the number of values is even.
- Total Count: The number of values that entered the transformer.
- Numeric Count: The number of numeric values that entered the transformer
- Range: Equal to the maximum minus the minimum, or a blank string if any value is not numeric.
- Sum: The sum of all numeric values, or a blank string if there were no numeric values.
- Mean: The sum of all numeric values divided by the number of numeric values, or a blank string if there were no numeric values.
- Standard Deviation: The standard deviation of all the numeric values, as measured by the "nonbiased" or "n-1" method, or a blank string if there were zero or one numeric values. If the data values are large, the standard deviation calculation may fail. In this case, a warning will be logged and the returned standard deviation will be -1.
- Mode: The most frequent of all the values. If the dataset is bimodal (two or more values occur with the highest frequency) one of the values will be returned randomly.

### Example

The StatisticsCalculator transformer can generate statistics for groups of features rather than all features. This effectively adds the ability to create pivot tables in FME similar to the pivot tables in Excel.

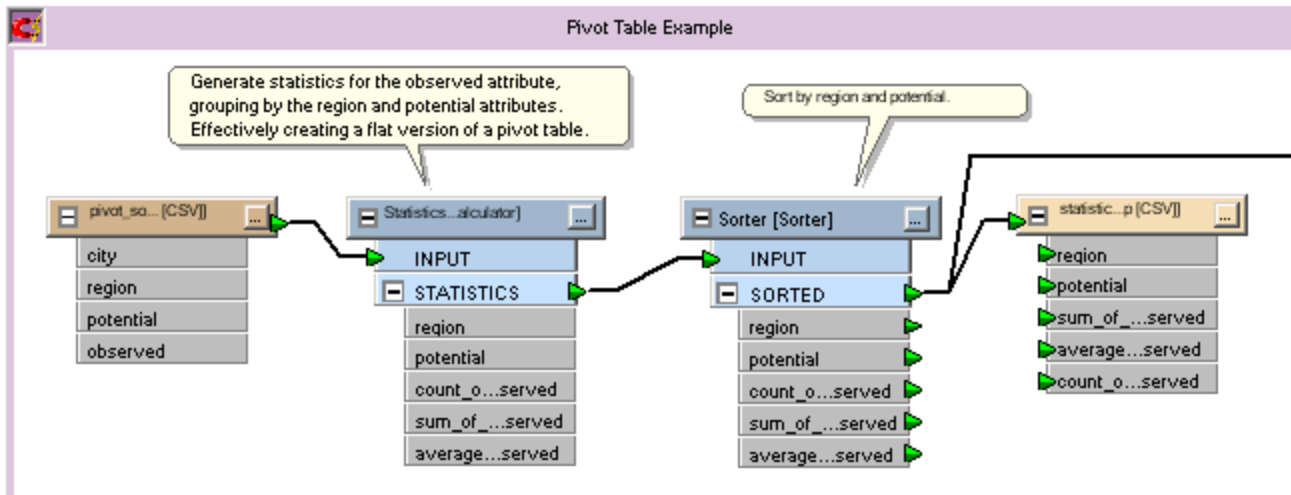
### Source Table and Excel Pivot Table

Fictitious data generated in Excel was exported it to a CSV file for use in Workbench. A simple pivot table was also created in Excel to show what we want to produce from FME; basically we want to summarize observed values based on region and potential.

city	region	potential	observed	Values				
				region	potential	Sum of observed	Average of observed	Count of observed
Surrey	Vancouver	high	3.4	Cariboo	medium	7.4	7.4	1
Vancouver	Vancouver	high	1.2	Kootenay	high	9.9	4.95	2
Prince George	Cariboo	medium	7.4		medium	0.9	0.9	1
Richmond	Vancouver	medium	4.8	Northwest	low	9.3	9.3	1
Nelson	Kootenay	high	1.4		medium	1.1	1.1	1
Ferrie	Kootenay	high	8.5	Vancouver	high	4.6	2.3	2
Terrace	Northwest	low	9.3		medium	4.8	4.8	1
Prince Rupert	Northwest	medium	1.1					
Golden	Kootenay	medium	0.9					

### FME Pivot Table

The workspace shown below uses the StatisticsCalculator transformer to create statistics for the observed attribute by first grouping features by region and potential. Then the new statistics features are sorted by region and potential, and output to a CSV file. The resulting CSV file has all of the same attributes/fields as the Excel pivot table.



The table written by FME and viewed in Excel resembles the Excel pivot table:

region	potential	sum_of_observed	average_of_observed	count_of_observed
Cariboo	medium	7.4	7.4	1
Kootenay	high	9.9	4.95	2
Kootenay	medium	0.9	0.9	1
Northwest	low	9.3	9.3	1
Northwest	medium	1.1	1.1	1
Vancouver	high	4.6	2.3	2
Vancouver	medium	4.8	4.8	1

You can also use the WebCharter transformer to chart the data.

**Transformer Category**

Calculators

**Technical History**

FME Factory Used: @Python, SortFactory

## StreamOrderCalculator

Computes the Strahler order and/or Horton order of streams in a river network.

This recursive algorithm processes vector river networks for Strahler stream order values. The algorithm requires the vector network to be topologically correct to successfully process. The network must be a center-lined network where each arc (sometimes referred to as an edge) must be joined at their node (sometimes referred to as a junction). No left and right banks or lake side shores should be present.

### Input

This transformer only takes linear features and one destination node per group.

### Output

- **NETWORK:** All river streams connected to the destination node are output through the NETWORK port with the Strahler order and/or Horton value assigned to the attribute(s) specified in the Strahler Order Attribute and/or Horton Order Attribute.
- **UNUSED:** All river streams that are not connected to the destination node are output through the UNUSED port.
- **CYCLE:** If any cycle exists, stream order is not computed and all lines are output through the UNUSED port. All nodes in which the cycles occur are output through the CYCLE port.
- **INVALID:** All non-linear features and extra destination nodes are output through the INVALID port.

### Parameters

Fix Flow Direction of Input

You can choose to fix the direction of the streams to fit the downstream direction to the destination node by setting this parameter to Yes.

Detect Cycles

You can choose to detect cycles by setting Detect Cycles to Yes. This is useful to make sure that no cycles exist in the network prior to fixing flow direction or computing stream order.

If any cycle exists, stream order is not computed and all lines are output through the UNUSED port. All nodes in which the cycles occur are output through the CYCLE port.

Stream Order Type = Strahler

The **Strahler** order of the streams is calculated as follows:

- When two or more streams with the same Strahler orders join, the outflow streams are assigned this Strahler order plus 1.
- When two or more streams with different Strahler orders join, the outflow streams are assigned the maximum Strahler order.

If Stream Order Type is set to **Horton**, then Strahler order is calculated internally before Horton order can be computed. Horton ordering is based on the idea of a main stream. It starts with finding out the main stream that flows to the sink node; the Horton order of the arcs in this main stream will be the maximum Strahler order of these arcs. This process continues for each of the remaining tributaries until all streams have been assigned a Horton order.

Stream Order Type = Horton

Users can specify the rules by which main streams are determined. At each junction in the network, the main stream of the incoming streams is selected based on the following rules:

- If Horton Class Attribute is specified, always match the value of this attribute of the incoming streams to the main stream determined previously. If there is only one stream that matches, it is part of the current main stream.
- Otherwise, pick the stream with the highest value based the following formula:

$$\text{priorityValue} = (\text{Horton Priority Weight}) * (\text{Horton Priority Attribute value}) + (\text{Horton Angle Weight}) * (\text{angle deviation between})$$



this stream and the previous main stream) + (Horton Length Weight) \* (ratio of longest length to a source node)

The angle between this stream and the previous stream is normalized to a value between 0 and 1. The value is 1 if the incoming stream is 0 degrees (straight) away from the previous main stream. If the stream is 180 degrees (exact opposite direction) away from the previous main stream, then the value is 0. If the angle between the streams is 45 or 90 or 135 degrees, the value is computed as 0.75, 0.5 and 0.25 respectively.

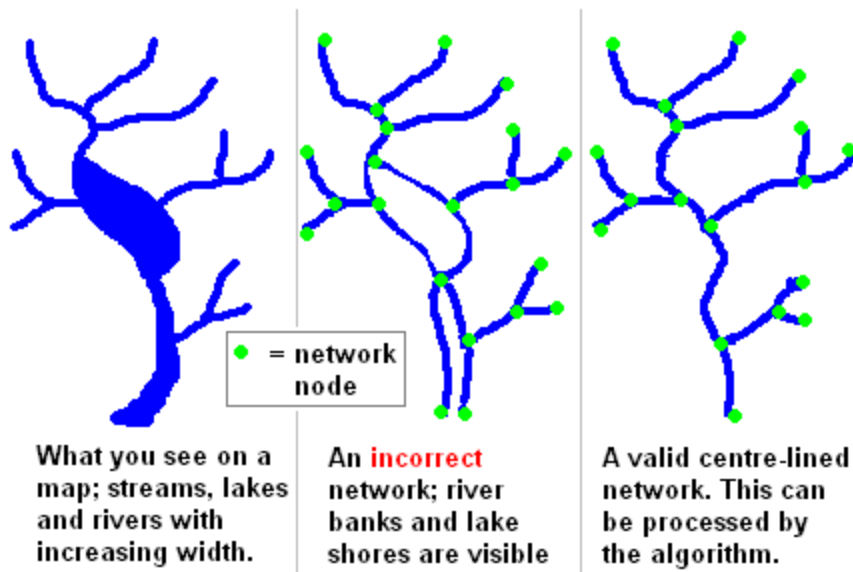
The ratio of the longest length to a source node for a stream is calculated as follows:

ratio = (longest length of this stream to a source node) / (sum of all longest lengths of the incoming arcs to a source node)

If none of the Horton Priority Weight, Horton Angle Weight and Horton Length Weight are specified, then by default, a main stream is determined by the longest branch. All these weights must be real values greater than or equal to zero.

### Example

The image below demonstrates a map representation of a river network, an invalid network where lake and river bank sides have been digitally captured and a valid, topologically correct, center-lined river network which the algorithm can process.



If the network is "broken" (arcs not connecting) then the output will be incorrect. The algorithm would treat the disconnected catchment as a separate river system, so it is important to check the connectivity of the river network before attempting to compute Strahler order values.

### Transformer Category

Network

### Related Transformers

NetworkFlowOrientor

NetworkTopologyCalculator

ShortestPathFinder

StreamPriorityCalculator

### FME Licensing Level

FME Professional edition and above

### Technical History

FME Factory Used: NetworkFactory



## StreamPriorityCalculator

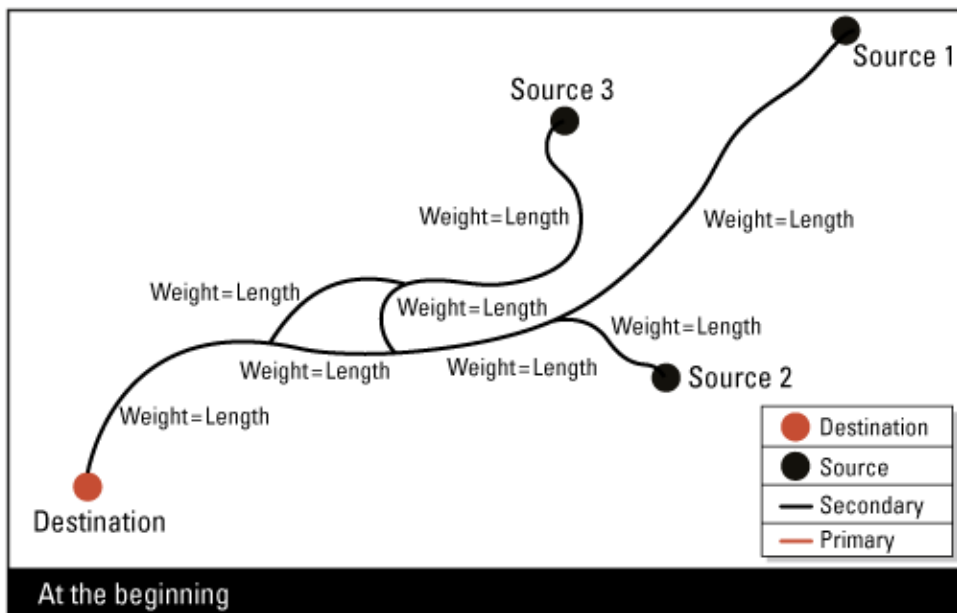
Calculates the primary and secondary streams of multiple stream networks. The key to determining the priority is the shortest path algorithm using multiple iterations within a network graph.

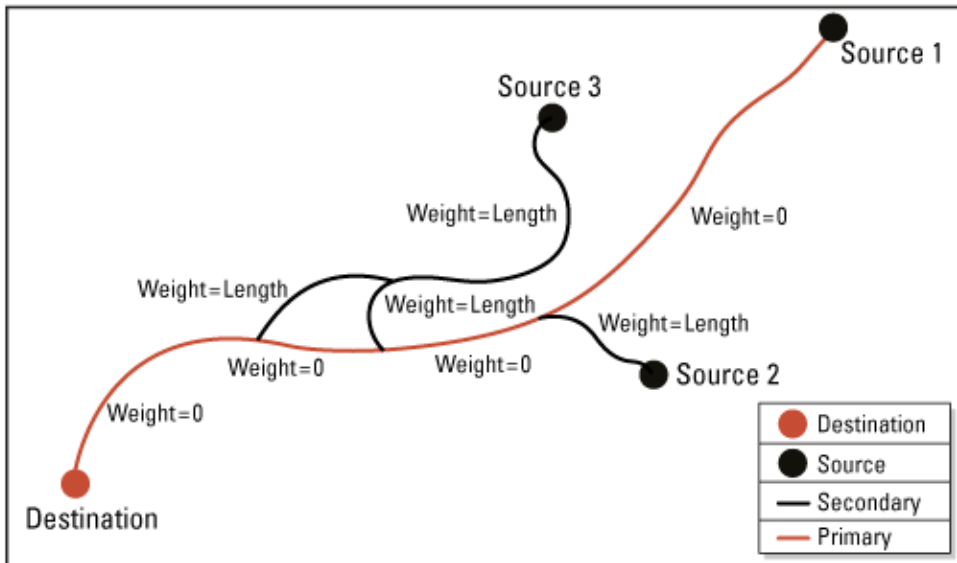
This attribute defines, for each source junction of the network, a unique path (the shortest path) to reach the destination junction. All flow lines included in a path (from the sources to the destination) will have a stream priority attribute set to 1 (primary); all others are set to 2 (secondary).

Before using this transformer, you will need to specify the weights on the network lines in the source data by specifying the Forward Weight Attribute and optionally Reverse Weight Attribute if the network graph is non-oriented.

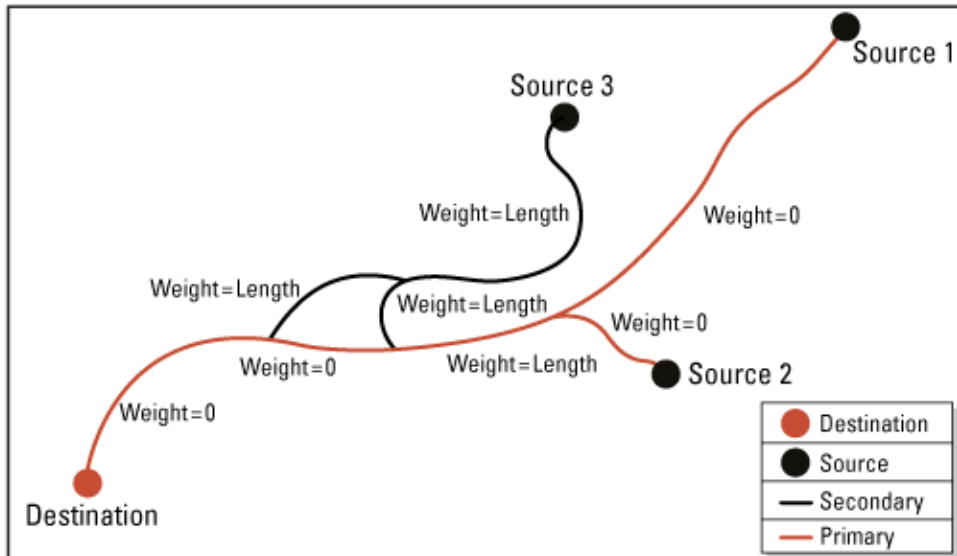
A weight is a property of a network line typically used to represent a cost for traversing across a network line. An example of a line weight is the length of the line. In a shortest path analysis, you would choose this weight if you wanted the resulting path to be of the shortest length. For line features, 2 weights can be used: one along the digitized direction of the line feature (the forward weight) and one against the digitized direction of the line feature (the reverse weight). The digitized direction of a line feature refers to the order of the vertices.

The goal is to flag the loops (cycles) in the network in order to highlight the primary network lines.

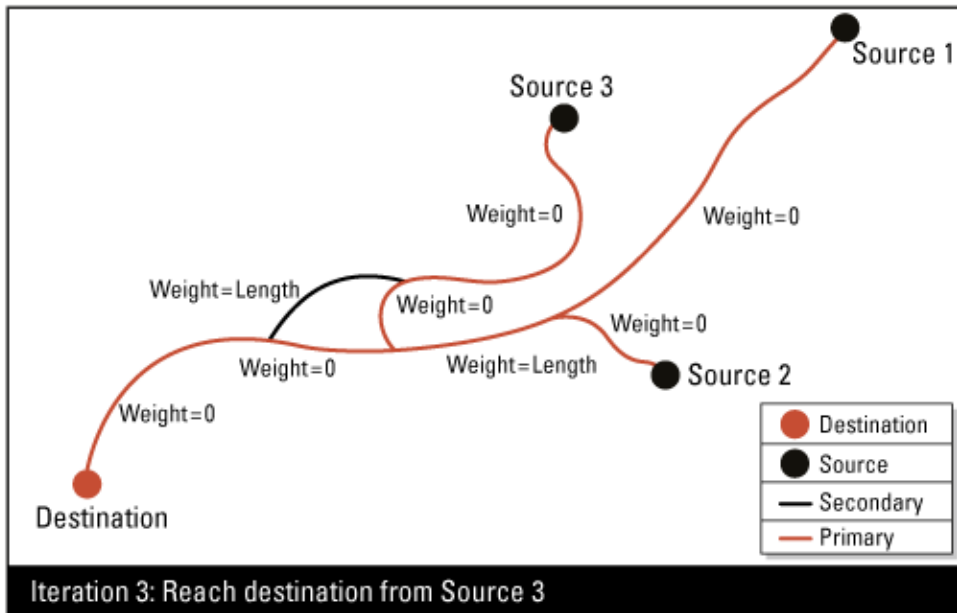




Iteration 1: Reach destination from Source 1 (*furthest source from destination*)



Iteration 2: Reach destination from Source 2 (*nearest source from destination*)



## Input

- LINE: Network lines with cycles (loops).
- DESTINATION: Destination nodes, located on an end point (leaf) of the network graph. All other end points on the network graph are considered sources.

## Output

- NETWORK: Streams that are connected will be assigned the same network ID attribute.
- EXTRA\_DESTINATION: Only one destination node is allowed per network. Any extra destination nodes found are output through the EXTRA\_DESTINATION port.
- INVALID: Destination nodes that are not found on any network are output through the INVALID port. All non-linear features are also output through the INVALID port.

## Parameters

### Forward Weight Attribute

An attribute name on the network lines that contains a weight along the digitized direction. This parameter is required to apply the shortest path algorithm.

### Reverse Weight Attribute

An attribute name on the network lines that contains a weight against the digitized direction. This parameter is required to apply the shortest path algorithm if the digitized direction of network lines is significant. By example, the digitized direction can represent a downstream flow direction for a hydrographical network. If the digitized direction is not significant for a network graph, a user can provide the same attribute of the Forward Weight Attribute parameter.

### Network ID Attribute

Streams that are connected will be assigned the same network ID in the Network ID Attribute. All streams will be assigned a stream priority value in the Stream Priority Attribute with either -1, 1 or 2. Streams that are not connected to the destination nodes will be assigned level priority value of -1. Meanwhile, primary or secondary streams will be assigned level priority value of 1 or 2 respectively.

### Stream Priority Attribute

An attribute name that will store the stream priority value (1 for primary or 2 for secondary) for output network lines.

## Expected Output

- Network lines with a stream priority attribute set to 1 (primary) or 2 (secondary). If it's not possible to determine the priority (if there isn't a destination located on network graph) a stream priority attribute is set to -1. The network lines have also a Graph Identifier attribute. All network lines in a same graph will have the same value in this attribute.
- Unused destination (if the destination is not located on an end point of the network graph)

## Usage Examples

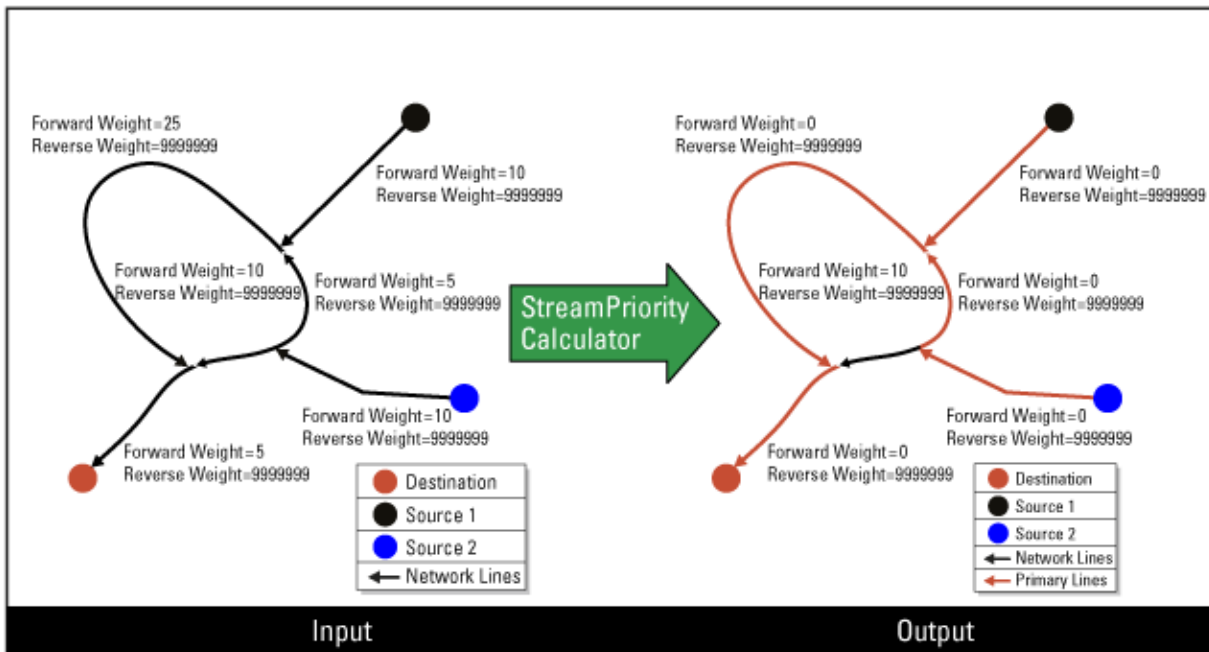
This transformer can be used on network linear flow lines. There are two ways to determine the stream priority attribute:

- To calculate the stream priority attribute for the oriented network lines: for these lines, the digitized direction represents a downstream flow direction.
- To calculate the stream priority attribute for the non-oriented network lines: for these lines, the digitized direction is not significant.

### Calculating the Stream Priority for Oriented Network Lines

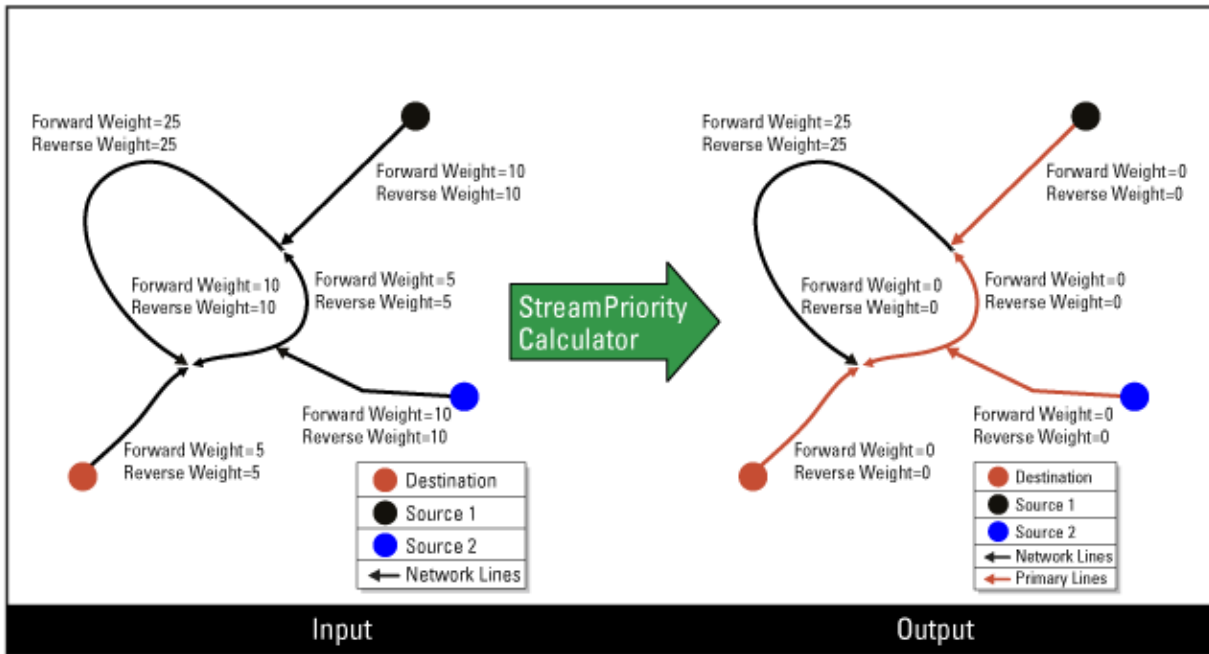
When the network lines are oriented, the shortest path should not go against the digitized direction. So initially the weight along the digitized direction (the forward weight) is the length, and the weight against the digitized direction (reverse weight) is a bigger value.

Note that the reverse weight is optional, and usually not required.



### Calculating the Stream Priority for Non-Oriented Network Lines

When the network lines are not oriented, the digitized direction of network lines is not significant. So the weight along the digitized direction (forward weight) and the weight against the digitized direction (reverse weight) are the same. In this case, you can use the same attribute corresponding to the length for both weight parameters. In this way, the loops are removed for the primary network lines (stream priority=1) and you can apply other algorithm to modify the digitized direction. This is how you can make network lines primary (stream priority=1) where the digitized direction represents a downstream flow direction.



### Transformer Category

Network

### Related Transformers

NetworkFlowOrientor

NetworkTopologyCalculator

ShortestPathFinder

StreamOrderCalculator

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: NetworkFactory

## StringConcatenator

Concatenates the values of any number of attributes and constants, and stores the result in a new attribute. The StringConcatenator complements Workbench's fanout capability by allowing you to fan out by more than one attribute simultaneously.

### Parameters

#### Destination Attribute

This is the new attribute that will contain the results. Use the default name, or type a new name.

#### Attributes

Select attribute names from the list and click Add.

#### Concatenated Items

The selected list of attribute names and constants. You can modify this list by reordering the items or removing them.

#### Constants

Constants are user-defined characters or strings that are the same for every feature. For example, you might want to specify a comma to separate the items in the concatenated list.

### Special Characters

Special characters can be concatenated:

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

### Related Transformers

ListConcatenator

CoordinateConcatenator

### Transformer Category

Strings

### fmepedia

See fmepedia for additional information about this transformer.

### Transformer History

This transformer was previously called the Concatenator.

### Technical History

Associated FME function or factory: @Tcl2



## StringFormatter

Reformats the data held in each specified attribute according to the TCL *format* command, which is similar to the C *printf* function. Attribute values can be formatted into strings, characters or numbers. Numeric output may be decimal (floating point or integer), octal, or hexadecimal; leading zeroes are removed from attribute values before performing numeric formatting.

The syntax for the format string is:

**[flags][width][precision]<type>**

**Note:** A preceding percent sign (%) is not part of the format string.

### Flag Characters:

One or more of the following flag characters can be specified:

- 0** The value should be padded with zeroes instead of spaces.
- The value should be padded on the right-hand side. (The default is to pad on the left.)
- +** The numeric value should always be signed.

### Field Width:

The optional field width is given as a decimal number. If the value has fewer characters than the field width, it will be padded with spaces.

### Field Precision:

The optional precision is given as a period followed by a decimal number.

### Attribute Type:

One of the following attribute types must be specified. The attribute value will be converted to the given type, if possible.

- d** Integer
- e** Decimal number (scientific notation: -d.ddde+dd)
- f** Decimal number (floating-point notation: -ddd.ddd)
- o** Octal number
- x** Hexadecimal number
- s** String

### Examples:

If the source attribute contains the value **spuds** the following format strings will give the following results:

Format	Result
8s	spuds
-08s	spuds000

If the source attribute contains the value **12345.6789** the following format strings will give the following results:

Format	Result
e	1.234568e+004
.2f	12345.68
015s	0000012345.6789

If the source attribute contains the value **1234** the following format strings will give the following results:

Format	Result
x	4d2
4.4f	1234.0000
+06d	+01234

### Transformer Category

Strings

### Technical History

FME Factory Used: @Tcl2

## **StringLengthCalculator**

Category: Strings

Calculates the length of the string in Source Attribute. The value is put into the String Length Attribute.

Note that in non-ASCII character sets, the length is the number of actual characters (some or all of which could be multi-byte) in the string, which may not match the number of bytes used to store the string.

### **Technical History**

Associated FME function or factory: @Tcl2

## **StringPadder**

Pads the selected attributes with spaces, either on the right or left side. For each attribute specified, if the attribute's length is less than Pad Width, spaces will be added to the given side to bring it up to that length.

### **Parameters**

Attributes

Choose the attribute(s) to pad.

Pad Width

Pad Width is the desired minimum length.

Side to Pad

This sets the padding to the Right or the Left.

### **Example**

Original value: "hello"

Pad Width: 10

Side to Pad: Left

Resulting value: " hello"

### **Transformer Category**

Strings

### **Technical History**

FME Factory Used: @Tcl2

## StringPairReplacer

Replaces characters in the value contained in the source attribute based on the replacement key-value pairs. The replacement pairs parameter is a list of

`key value key value ... .`

Each instance of a key in the source string will be replaced with its corresponding value. If case-sensitive is no, then matching is done without regard to case differences. Both the keys and values may have multiple characters. Replacement is done in an ordered manner, so the key appearing first in the list will be checked first, and so on.

For example, if the source attribute's value was:

`bobby`

and the replacement pairs were:

`b s o a`

the result will contain:

`sassy`

Note that the replacement pairs are separated by spaces. If either of the strings contains a space, it must be escaped with a \. For example, if the source attribute's value was:

`billy bob`

and the replacement pairs were:

`y\ b a`

the result will contain:

`billaob`

To substitute a substring specified using regular expressions, use the StringReplacer transformer.

## Transformer Category

Strings

## Technical History

Associated FME function or factory: @Tcl2

## StringReplacer

Category: Strings

Replaces substrings matching a string or regular expression in the string contained in the source attribute.

The **Attributes** parameter specifies which attributes will have substrings replaced.

The **Text to Find** parameter specifies the substring that will be replaced.

The **Replacement Text** parameter specifies the substring that will replace instances of the replacement substring.

If the replacement text contains a '&' or '\0', then it is replaced in the substitution with the portion of string that matched the regular expression. If replacement text contains '\#', where # is a digit between 1 and 9, then it is replaced in the substitution with the portion of string that matched the n-th parenthesized subexpression of the regular expression.

Special character sequences can be used in both the **Text to Find** and **Replacement Text** fields. These characters are interpreted as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, \e will be interpreted as e.)

Sequence	Description
\a	Audible alert (bell) (0x07)
\b	Backspace (0x08)
\f	Form feed (0x0c)
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

The **Use Regular Expressions** parameter specifies whether the Text to Find parameter is a plain string or a regular expression.

If the **Use Regular Expressions** parameter is set to Yes, Advanced Regular Expressions (AREs) are supported. Consult the *FME Functions and Factories* manual for a complete description of AREs. In brief, An ARE is one or more branches, separated by '|', matching anything that matches any of the branches.

A brief summary of the special characters and their meanings is:

- | separates "branches" (or choices)
- \* a sequence of 0 or more matches of what precedes it
- + a sequence of 1 or more matches of what precedes it
- ? a sequence of 0 or 1 matches of what precedes it
- . matches any single character
- ^ matches the start of the value
- \$ matches the end of the value
- [ ] enclose a set of character choices
- ( ) enclose a "subexpression" -- whatever matches each subexpression is placed into the `matched_parts{}` list attribute
- a any character can be listed to be matched

The **Case Sensitive** parameter specifies whether or not the substring matching will be case-sensitive.

### Examples:

In this example, a pure substitution of text is made without any use of regular expression functionality. This is the simplest kind of substring replacement.

Source String: Bobby  
Text to Find: obb  
Replacement Text: ill  
Use Regular Expression: no  
Case Sensitive: yes  
Result: Billy

In this example, a pattern matching zero or more 'b' characters is replaced with nothing.

Source String: Bobby  
Text to Find: b\*  
Replacement Text:  
Use Regular Expression: yes  
Case Sensitive: yes  
Result: Boy

In this example, a pattern matching zero or more 'b' characters followed by a y is duplicated in the result (prepended by hyphens)

Source String: Bobby  
Text to Find: b\*y  
Replacement Text: --\\0-\\0  
Use Regular Expression: yes  
Case Sensitive: yes  
Result: Bo--bby-bby

See the `StringSearcher` transformer help for additional regular expression examples.

To replace pairs of substrings, use the `StringPairReplacer` transformer.

To search for regular expression matches in a string without doing any replacement, use the `StringSearcher` transformer.

### **Technical History**

Associated FME function or factory: @Tcl2 (string map and reesub functions)

## StringSearcher

Note: This transformer was previously named *Grepper*. It has been renamed to more accurately describe its function.

Category: Strings

Performs a regular expression match on the value of the specified attribute. If the attribute matches the pattern, the feature is output via the MATCHED port, and the portion of the original attribute's value that matched the regular expression is stored in the `_matched_characters` attribute (and optionally matching pieces of the expressed are stored in the `_matched_parts{}` list). Otherwise, it is output via the FAILED port.

This transformer takes its name and inspiration from the UNIX utility **grep**, which searches for patterns in text files.

To use this transformer to parse out portions of an input attribute, "subexpressions" within the regular expression are used. Subexpressions are enclosed in parentheses ( ), and the portion of the input text that matched that subexpression is stored as an entry in the `_matched_parts{}` list. The elements of this list can then be exposed to Workbench by right clicking on it and indicating the number of elements to expose for later use. See below for some examples.

Advanced Regular Expressions (AREs) are supported. For a complete description of AREs, see *Syntax of Tcl Regular Expressions* in the *FME Functions and Factories* manual.

In brief, an ARE is one or more branches, separated by |, matching anything that matches any of the branches.

A brief summary of the special characters and their meanings is:

- | separates "branches" (or choices)
- \* a sequence of 0 or more matches of what precedes it
- + a sequence of 1 or more matches of what precedes it
- ? a sequence of 0 or 1 matches of what precedes it
- . matches any single character
- ^ matches the start of the value
- \$ matches the end of the value
- [ ] enclose a set of character choices
- ( ) enclose a "subexpression" – whatever matches each subexpression is placed into the `_matched_parts{}` list attribute
- a any character can be listed to be matched

Examples:

- ^A matches any value starting with an A
- ^[0-9] matches any value starting with a digit
- ^[0-9]+\$ matches any value consisting exclusively of digits
- ^(beef|chicken)\$ matches values of either "beef" or "chicken"
- ^([0-9]\*) ([0-9]\*)\$ matches two integer numbers separated by a space, and puts the first number into `_matched_parts{0}`, and the second into `_matched_parts{1}`
- ^N([0-9][0-9])[.][0-9][0-9].[0-9][0-9] matches N23.45.11, and puts 23 into `_matched_parts{0}`, 45 into `_matched_parts{1}`, and 11 into `_matched_parts{2}`

The Regular Expression field can also include any number of escape sequences, as specified in the following table. If the sequence is not listed in the table, the backslash character is ignored. (For example, entering `lan\es` will be interpreted as `lanes`.)

Sequence	Description
<code>\a</code>	Audible alert (bell) (0x07)
<code>\b</code>	Backspace (0x08)
<code>\f</code>	Form feed (0x0c)



<b>Sequence</b>	<b>Description</b>
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\t	Tab (0x09)
\v	Vertical tab (0x0b)
\\	Backslash  (Note: A single backslash entered at the end of a value will be converted to an escaped backslash.)

Note that the matches can be either case-sensitive or case-insensitive, depending on how the transformer is configured.

To replace substrings matching a regular expression in a string, use the `StringReplacer` transformer.

### **Technical History**

Associated FME function or factory: `TestFactory`, `@Tcl2` (regexp procedure)

## **SubstringExtractor**

Category: Strings

Extracts a substring from the source attribute. The substring is taken from the range of characters specified. Character indexes start at 0 for the first character. A negative index is used to indicate the position relative to the end of the string (-1 is the last character, -2 the second last, and so on). The index can also be taken from the value of another attribute.

If the last index is greater than or equal to the length of the string then it is treated as if it were the end of the string. If the first index is greater than the last index then an empty string is placed into the result attribute.

Each of the index parameters may either be entered as a number, or can be taken from the value of a feature attribute by selecting the attribute name from the pull-down list.

Examples:

- To trim off the first character only, use a start of 1 and an end of -1
- To trim off the last character only, use a start of 0 and an end of -2
- To extract the second and third characters in the string, use a start of 1 and an end of 2

### **Technical History**

Associated FME function or factory: @Tcl2 (string range function)

## **SummaryReporter**

Writes a summary report of features that enter to a disk file. Features are sorted prior to being summarized.

The summary report is a human-readable text file which can be read by any text editor. This summary report can be used to compare and quickly find differences between sets of features.

The summary report includes fields such as Feature Type, 3D Length, 2D Length, X Min, X Max, Y Min, Y Max, Z Min, Z Max, Number of Coordinates, Coordinate System, and Attribute Values.

The report title is written at the top of the file.

The **Suppress Empty Report** parameter controls whether an empty file is created when no features enter.

The **Append to Existing Report** parameter controls whether the new report overwrites or is appended to existing files.

The **Write Report in UTF-8** parameter controls the encoding of the output text file. When set to Yes, the output text file is encoded in UTF-8. Otherwise, it is written in the current system encoding.

## **Transformer Category**

Infrastructure

## **FME Licensing Level**

FME Professional edition and above

## **Technical History**

Associated FME function or factory: ReportFactory

## SurfaceDraper

Interpolates z coordinates and adds them to DRAPED\_FEATURES based on the underlying surface defined by the POINTS/LINES, and BREAK-LINES.

If the incoming DRAPE\_FEATURES are 2D, then the z elevation is added. If the incoming DRAPE\_FEATURES are 3D, then the z coordinates are replaced with new ones.

### Input Ports

- **POINTS/LINES:** The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model. This port also accepts raster features as input.
- **BREAKLINES:** These features are put into the model as breaklines such that triangle edges will always be along the line of the feature.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, DEMGenerator, ContourGenerator, and SurfaceModeller transformers.

- **DRAPE\_FEATURES:** Features input through this port are output via the **DRAPED** port with their z value set to the values from the surface model. It is not a requirement that the input feature be 2D. If the input feature is 3D, then the z dimension is simply replaced with **that** from the model.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, SurfaceModeller transformers.

### Output Ports

DRAPED\_FEATURES

### Parameters

Group By

Surface Tolerance

Drape Method

This parameter determines if z elevations are only added where there is a vertex on the input DRAPE\_FEATURES or if coordinates should be added wherever the underlying model determines a change in slope.

VERTEX: the output DRAPED\_FEATURES will have the same number of vertices as they had when they were input.

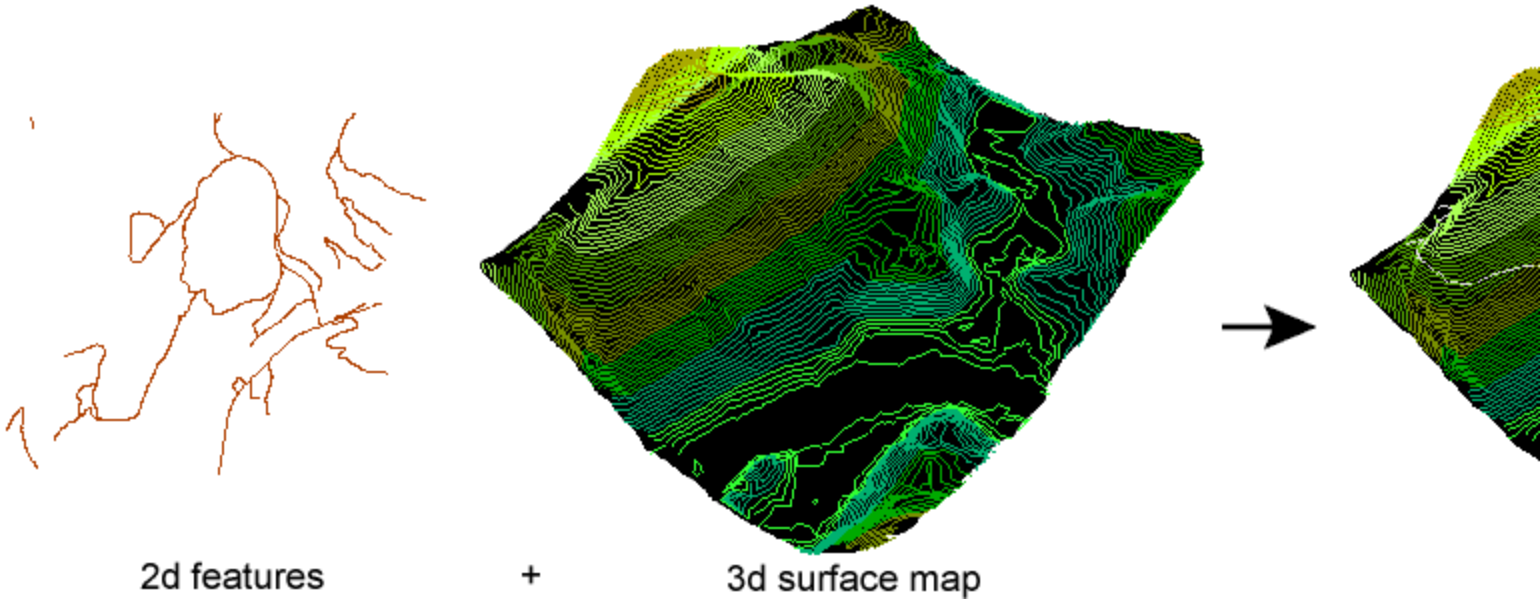
MODEL: additional vertices are added at each slope change.

Interpolation Method

The Interpolation Type parameter defines how the z elevation is determined. If CONSTANT is specified, then the z elevation of each coordinate is set to be the elevation of the closest model point. If PLANAR is set, then interpolation is used to determine the elevation. If AUTO is set, then the best method will be chosen automatically.

The underlying surface is built using the z tolerance as specified by the Tolerance parameter. This should be set to the larger of the accuracy of the elevation data being input and the accuracy of the output elevation that is required.

## Example



2d features

+

3d surface map

### Transformer Category

Surfaces

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: SurfaceModelFactory

## SurfaceModeller

Builds and queries surfaces, drapes features or changes the surface representation between Digital Elevation Model (DEM), Triangulated Irregular Network (TIN), and CONTOURS. You can go from any representation to any other representation.

When you need to extract several things from a defined surface, this transformer is more efficient than, for example, using separate transformers to build contours and a TIN.

### Input Ports

- **POINTS/LINES:** The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model. This port also accepts raster features as input.
- **BREAKLINES:** These features are put into the model as breaklines such that triangle edges will always be along the line of the feature.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, DEMGenerator, ContourGenerator, and SurfaceModeller transformers.

- **DRAPE\_FEATURES:** Features input through this port are output via the **DRAPED** port with their z value set to the values from the surface model. It is not a requirement that the input feature be 2D. If the input feature is 3D, then the z dimension is simply replaced with **that** from the model.

This is snippet text. Any changes here are also reflected in the SurfaceDraper, SurfaceModeller transformers.

### Output Ports

- **CONTOURS:** If specified, then the model is output as a set of contours through this clause. Each feature is 2D or 3D depending on the value specified for **CONTOUR\_DIMENSION**. Each output feature also has an attribute `SurfaceModel.elevation` that holds the elevation of the contour. If the output feature is 3D, then this value is the same as the z coordinate on each feature.
- **DEM\_POINTS:** Returns the model as a series of evenly spaced 3D points based on the sampling rate specified.
- **DEM\_RASTER:** Returns the model as a single raster feature consisting of evenly spaced 3D points arranged by rows and columns. The spacing between rows and columns is based on the sample spacing specified. Alternatively, the number of rows and columns can be specified to establish the size of the grid, instead of the sample spacing. Only one or the other may be specified.
- **DRAPED\_FEATURES:** Features input via the **DRAPE\_FEATURES** port are output here with the z dimension set to values from the model.
- **TIN\_EDGES:** Outputs all the edges which define the underlying Delaunay Triangulation. Each feature output on this clause has the attributes `SurfaceModel.vertex1_id` and `SurfaceModel.vertex2_id`, which identifies the ID of the vertex to which it was connected.
- **TRIANGLES:** Returns the Triangles that define the Delaunay Triangulation. Each feature output on this tag has these attributes: `SurfaceModel.vertex1_id`, `SurfaceModel.vertex2_id`, and `SurfaceModel.vertex3_id`, which identify the IDs of the vertices; `SurfaceModel.slope`, which is the slope of the triangles in degrees; `SurfaceModel.aspect`, which is aspect angle in degrees measured from the x-axis in a counter-clockwise direction; `SurfaceModel.percentageSlope`, which is the percentage slope.
- **TIN\_SURFACE:** Returns the Triangles that define the Delaunay Triangulation as one unified mesh geometry.

- VERTEX\_POINTS: Outputs each vertex of the underlying surface model triangulation. Each vertex is tagged with the unique identifier *SurfaceModel.vertex\_id*.
- VORONOI\_DIAGRAM: Outputs the dual of the Delaunay Triangulation called the Voronoi Diagram. For each vertex of the Delaunay Triangulation, a Voronoi polygon feature is returned. The Voronoi polygon has the attributes *SurfaceModel.vertex\_id*, identifying the ID that it represents and *SurfaceModel.elevation* which gives the elevation of the point and hence the Voronoi polygon.

## Parameters

### Group By

This option allows you to select attributes that define which groups to form. The default behavior is to use the entire set of features as the group.

### Surface Tolerance

The surface tolerance is used when building the surface to determine which vertices to add to the surface model. Specifying a value of 0 turns off the vertex filtering so that all vertices (except those with duplicate x,y) are added to the model.

When specified, the surface tolerance is used during the construction phase of the surface model to determine whether a vertex will be added to the model, or whether it will be discarded and not added to the model.

The surface tolerance works as follows, for each vertex that is being added to the model:

- If the x,y location is outside the convex hull of the existing model, it is added to the model.
- If the x,y location is inside the existing model:
  - The difference between the z value from the existing TIN and the z value of the vertex is calculated.
  - This difference is compared to the surface model tolerance.
  - The vertex is only added to the model if the difference is greater than the surface tolerance; otherwise, the vertex is discarded.

### Drape Method

Any features that are input to the transformer through this parameter will be returned with their attribution unchanged; however, they will now be 3D, having their z value set according to the value of Interpolation Method.

If Drape Method is VERTEX, the feature will be returned with the same number of vertices that it was input with, and with z values added to each vertex on the feature.

If Drape Method is MODEL, the feature will have other coordinates added to it, forcing it to exactly follow the underlying surface model.

### Interpolation Method

This parameter is used if the output of the model is requested as DEM\_POINTS or if draping features are input to the model.

- CONSTANT: The z value of each output point is set to that of the closest vertex of the underlying model.
- PLANAR: Planar interpolation is used to determine the z value for each output point. If a point is not over a surface triangle and PLANAR is specified, the output z value will be set to NAN.
- AUTO: The algorithm will attempt to give the best possible result for each point. In the current version of AUTO, the planar method is used if the point is over a surface triangle and the constant method is used otherwise.

### Output Contour Dimension

The value specified for this parameter is also used to specify whether the output contours are to be 2D or 3D. The default is 2.

### Output Contour Interval

If the output port CONTOURS is specified, then Output Contour Interval is used to specify the elevation separation of the output CONTOURS. The

Contour Interval has a default value of 500.0.

### Perturb Contours

This parameter controls whether contours can be perturbed in the z direction. The perturbed amount is 1% of the contour interval.

Yes: Features whose z-values are full multiples of the contour interval will be preserved. The z-values of these features will be perturbed so they no longer are full multiples of the contour interval.

No (default): Features whose z-values are full multiples of the contour interval will be dropped.

Note that by default, input features with z-values that are full multiples of the contour interval will be skipped. For example, setting a contour interval of .5 or 1 means that no features will be used if their z-values are all whole numbers; for example, 123.0, 0, 567, 987.0, etc.

### Output DEM X and Y Spacing

If the output tag DEM\_POINTS is specified, then SAMPLE\_SPACING is used to specify spacing of the output DEM POINTS. If SAMPLE\_SPACING is not specified, then 1 is used.

### Output DEM Nodata Value (Optional, Raster Only)

This parameter is used to fill the raster cell values that fall outside of the surface's bounding box. If this is not specified, NaN values will be used instead. Use of this clause is highly recommended in order to produce consistent raster data with nodata values rather than NaN values.

### Maximum Point Distance Factor

This parameter specifies the maximum distance between points allowed when calculating contours. The smaller the distance, the more selective the contour calculation. This factor is useful for cases where contours are crossing water boundaries for example. Specifying a value of 0.0 (default) turns off this selection processing.

### Elevation Attribute

You can change the name of the elevation attribute or use the default.

### Use Surface Model File

This transformer also allows you to save the surface model in a set of datafiles for later use. This is useful if you need to build a large surface model as part of multiple runs or as part of one run but want to use it again later. The saved surface models are not portable from one machine type to another but are intended to share a surface model as part of a production stream.

### Model Base File Name

To create a persistent surface model you simply toggle Use Surface Model File to Yes and then specify the full path of the base file using a suffix of \*.fsm. The surface model will then create as many files as needed to store the entire surface model that is being constructed.

### Surface Model Update Mode

To create a new model set the Surface Model Update Mode to OVERWRITE. In this mode it will not use any existing base surface model that is found but instead will overwrite it. To add more data to an existing model then change the setting to APPEND and then data will be added to an existing model. If the base surface model doesn't exist then it will be created. To use an existing surface model you simply don't route any features to the BREAKLINES, POINTS, or 3D\_LINES.

To use an existing Surface Model when you are not adding more data to it, you must set the Surface Model Update Mode to APPEND.

### Estimated # of Vertices in Model

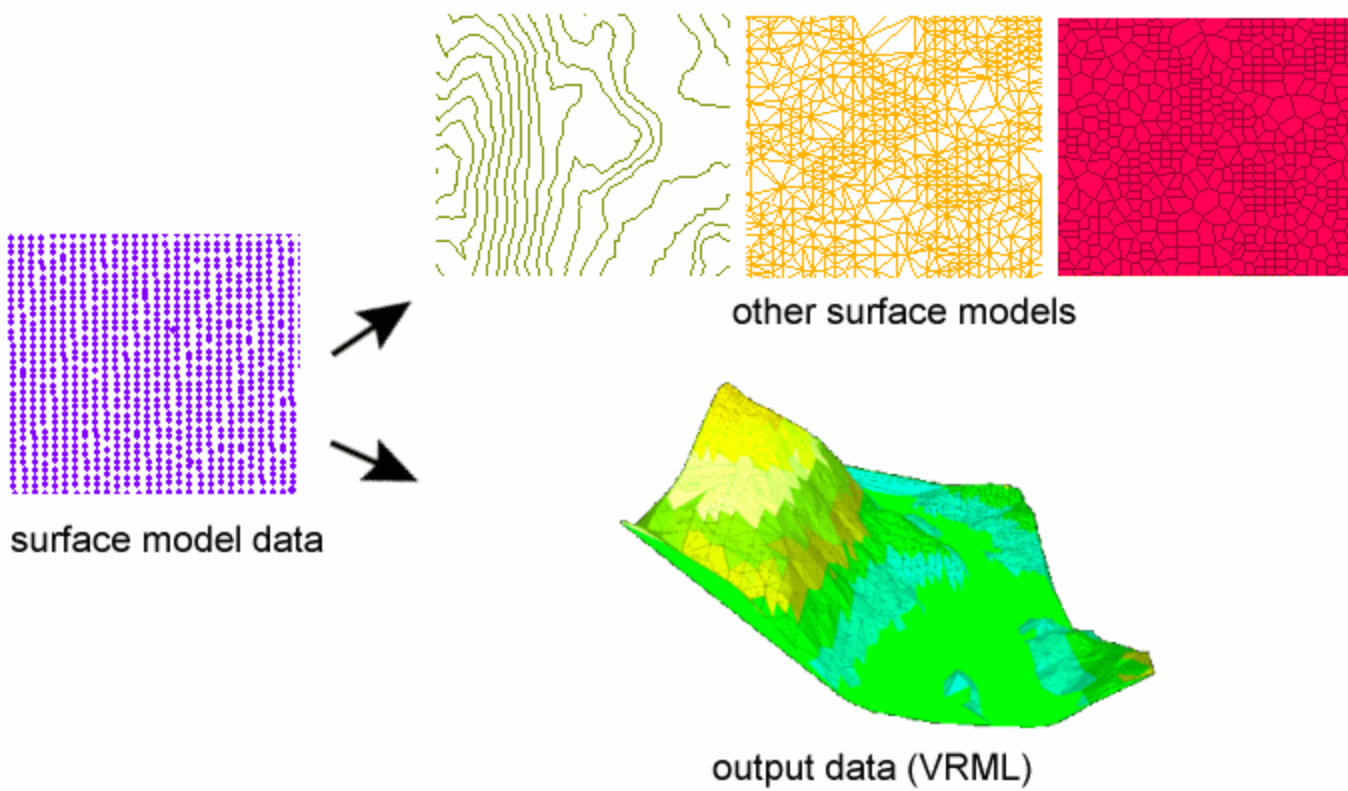
The Estimated number of vertices in the final model is strongly suggested when you are building a large model in multiple runs, as this is used by the model construction for optimization purposes.

### XMin, YMin, XMax, YMax

When building large surface models across multiple FME uses, the values specified for XMin YMin, XMax, and YMax must be specified; otherwise the surface model will use the bounding box of the first run for construction.



## Example



## Usage Notes

It is important to note that if the only thing you want to do with an existing surface model is output information, then you must route a feature into the surface model. The easiest way to do this is to simply use the Creator transformer and route a single NULL Geometry feature into the DRAPE\_FEATURES input port.

## Transformer Category

Surfaces

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: SurfaceModelFactory

## **SurfaceReverser**

Reverse surfaces and solids.

On surfaces, it will reorder the coordinates of the surface such that the normal of the output surface is the opposite of the input surface. Vertex normals that exist on the surface will be also be reversed.

On solids, it will reverse the underlying surfaces, in effect causing the solid to be turned inside-out.

### **Parameters**

Orientation Flag Attribute

If the Orientation Flag attribute is specified, then the attribute will be added to the resulting features and its value is set to Yes if the orientation of the geometry has been changed. Otherwise, the value of the attribute is set to No.

### **Transformer Category**

3D

### **Technical History**

Associated FME function or factory: @Orient

## **SurfaceSplitter**

Splits a double-sided input surface geometry into two single-sided surfaces – one equal to the front side of the input surface and one equal to the back side of the input surface.

### **Output Ports**

FRONT/BACK: This transformer splits a double-sided input surface geometry into two single-sided surfaces – one equal to the front side of the input surface and one equal to the back side of the input surface – through the FRONT and BACK ports, respectively.

The surface output through the BACK port will be a single-sided surface whose front is equivalent to the back side of the original geometry. If a single-sided geometry is input, it will be output via the corresponding output port, converted to a front-sided surface in the case of a back-sided surface.

INVALID: Zero-sided surfaces will be treated as invalid. Unprocessed geometries are output through the INVALID port.

### **Parameters**

None.

### **Transformer Category**

Surfaces

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: SurfaceSplitterFactory

## **SystemCaller**

Runs a program and waits for it to exit before continuing the translation.

The command-line parameter must start with the name of a program (.exe) or batch file (.bat) that is either fully specified or is in the PATH. It can contain additional arguments as well, which are passed as command-line arguments to the program or batch script.

If an Exit Code Attribute is specified, it will be set to the executed program's exit code: usually 0 for success and nonzero for failure.

## **Transformer Category**

Infrastructure

## **Technical History**

Associated FME function or factory: @System

## TCLCaller

Runs a Tool Command Language (Tcl) command and assigns its return value to an attribute.

The Tcl command can operate on the feature's geometry and/or attributes, using any of the built-in Tcl functions provided by the Tcl language, as well as any of the FME-provided Tcl facilities.

See the Tcl language reference manual ([www.tcl.tk](http://www.tcl.tk)) and the FME Tcl function in the *FME Functions and Factories* manual for details of the capabilities.

Common Tcl examples usage include:

- Trim spaces from an attribute:

```
FME_SetAttribute trimmedAttribute [string trim [FME_GetAttribute originalAttribute]]
```

- Replace all non-numeric characters with spaces in an attribute:

```
FME_SetAttribute anAttribute [regsub -all {[^0-9]} [FME_GetAttribute anAttribute)] {}]
```

Note that in this case, the return value is the number of substitutions actually made.

- Match a regular expression against an attribute:

```
regexp {^[A-Za-z]*$} [FME_GetAttribute anAttribute]
```

This regular expression tests if the entire value of the attribute consists only of alphabetic characters.

Note that when matching regular expressions, the return value will be 1 if the expression matched, and 0 otherwise.

The recommended way to manipulate feature attributes is through the functions that are provided for this purpose:

```
FME_GetAttribute attrName  
FME_SetAttribute attrName newVal  
FME_CopyAttribute destAttrName srcAttrName  
FME_RenameAttribute destAttrName srcAttrName  
FME_UnsetAttribute attrName1 [attrName2 attrName3 ...]
```

A Tcl source file can optionally be provided. The file will be read before the Tcl command is executed. This can be used to reference Tcl functions kept in a common external file.

Additional code may also be provided in the Source Code editor. This code will be compiled once at the beginning of the translation and so offers a more efficient option than providing extensive code in the Tcl Expression parameter.

## Transformer Category

Infrastructure

## Technical History

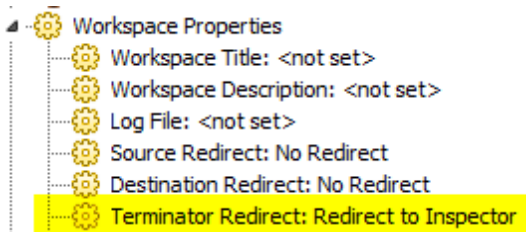
Associated FME function or factory: @Tcl2

## Terminator

Stops a translation when it detects detect non-valid situations or input data conditions that should not exist.

When a feature is directed to this transformer, the translation immediately stops and an error message is displayed (Transaction Terminated). To redirect the Terminator to an Inspector instead of stopping the translation, you can set this option in the Workspace Properties in the Navigator.

## Terminator Redirect



When a feature is directed to this transformer, Workbench immediately stops and displays an error message (Transaction Terminated). If you are programming or debugging, you would usually disable the connections to the Terminators and then add an Inspector to verify if the features are real errors. In production mode, you have to re-enable these connections and delete the Inspector.

This option allows you to automatically redirect the features that enter the Terminator to a Visualizer, without having to modify the workspace. When this option is activated, all the features that enter a Terminator are redirected to an Inspector and the translation continues without stopping. A message is added to the log file to indicate that some features were redirected to a Visualizer.

## Transformer Category

Infrastructure

## Technical History

Associated FME function or factory: @Abort

## Tester

Evaluates one or more tests on a feature, and routes the feature according to the outcome of the test(s). The tests can consist of any FME-allowed operands.

## Ports

- **PASSED:** If the test(s) pass, the feature is output via the PASSED port.
- **FAILED:** If the test(s) fail, the feature is output via the FAILED port.

Note that you can combine several tests into a single Tester transformer, and the features can be routed to PASS if only one test passes, or if ALL tests pass.

## Parameters

### Pass Criteria

The Pass Criteria defines how multiple clauses are interpreted in the final classification of the incoming feature.

You can choose one of three test scenarios:

Scenario	Pass Criteria	Description																
One test is required for the input feature to be classified as PASSED.	One Test (OR)	In this case, as long as one of the test clauses is true, then the feature is PASSED. This is an OR test (test1 OR test2 OR test3). If any one is true, then the result is true.																
All tests are required for the input feature to be classified as PASSED.	All Tests (AND)	This is stricter than One Test (OR) because all tests must pass in order for the result to be true (test1 AND test2 AND test3).																
Create your own test expression.  This is useful when you need fine-grained control over what you want the Tester to evaluate. If you select this mode, a Test Expression field appears.	Composite Test	If, for example, you want to check whether the value of an attribute is between 5 and 10, or equals 99, you can set up three test clauses:  Clause 1 : $x > 5$ Clause 2 : $x < 10$ Clause 3 : $x = 99$  (where $x$ is the selected attribute in the Left Value field): <table border="1"><thead><tr><th></th><th>Left Value</th><th>Operator</th><th>Right Value</th></tr></thead><tbody><tr><td>1</td><td>value</td><td>&gt;</td><td>5</td></tr><tr><td>2</td><td>value</td><td>&lt;</td><td>10</td></tr><tr><td>3</td><td>value</td><td>=</td><td>99</td></tr></tbody></table> To correctly get the desired results, you require that clause 1 AND clause 2 be true (between 5 and 10), OR clause 3 is true (equals 99).  In this case, choosing One Test or All Tests modes will not satisfy the test requirement. You can, however, choose Composite Test and enter the following expression in the Test Expression field:  $((1 \text{ AND } 2) \text{ OR } 3)$  The numbers above correspond to the test clauses defined in the 'Test Clauses' table. When read, the composite expression above states that 'Clause 1 AND Clause 2 must be satisfied, OR Clause 3 must be satisfied'.		Left Value	Operator	Right Value	1	value	>	5	2	value	<	10	3	value	=	99
	Left Value	Operator	Right Value															
1	value	>	5															
2	value	<	10															
3	value	=	99															

### Comparison Mode

By default, the Comparison Mode is set to *Automatic (compare as numbers if possible)*. This means the Tester will first try to convert the

operands to numbers. If it is successful, it will compare them as numbers. If it is still not successful, it will treat operands as strings.

**Alphanumeric Strings:** Say you have a string that is labeled "4E5". If you choose *Automatic (compare as numbers if possible)*, it is possible that it will be treated as a number. If you want it treated as a text string, set the Comparison Mode to *String (always compare as text strings)* – this ensures that the Tester will always treat the operands as strings, and will compare them as strings.

### Test Clauses

The **Value** columns may be a literal constant, an attribute name preceded by the value-of operator (&), or an attribute value function. If it is an attribute value function, the function will be executed on the current feature and the result will be used for the test.

The operands can consist of:

- An ampersand followed by the name of a feature attribute: this takes the "value-of" the attribute and uses that as the operand for the test: &<attribute name>

You can pick the attribute name from the pull-down list in the Tester's grid, or you can type it with the leading &.

- Any FME function: @FunctionName(argument1,argument2,...)

The function must be typed in directly into the Tester's grid, and must follow the FME syntax for functions. Choose *Help > FME Functions and Factories* to see a complete list of functions that can be called.

- Any constant value: anything not starting with an ampersand (&) or at-sign (@) is considered to be a constant: anyConstant

You can type the constant in the Tester's grid.

The grid dialog for the Tester makes it easy to specify any of the above inputs to a test or set of tests.

### Operators

The Operator column is one of: =, !=, <, >, <=, >=, In, Between, Like, Matches Regex, Contains, Begins With, Ends With.

Unique operators are described in more detail here:

Operator	Description	Example
In	A list of values in which you are testing for a certain value. The Right Value is a comma-delimited list of values, or a range.	X=5, is X In 1,2,3 (no = FAIL) X=5, is X In 3-7 (yes = PASSED)
Between	Whether a value is between a minimum and maximum. The Right Value consists of two comma-separated values.	X=5, is X Between 2,3 (no = FAIL) X=5, is X Between 1,10 (yes = PASSED)
Like	Allows you to use a wildcard query. Note that wildcard queries use the percentage symbol (%), not an asterisk (*).	X=abcd, is X Like %bc% (yes = PASSED)
Matches Regex (Advanced FME)	Does a value match a regular expression?	X=abcd, is X Matches Regex .*bc.* (yes = PASSED)
Contains	Does the Right Value appear in the Left Value?	X=abcd, is X Contains bc (yes = PASSED)
Begins With	Does a string begin with this...?	X Begins With a (yes = PASSED) X=abcd, is X Begins With b (no = FAILED)
Ends With	Does a string end with this...?	X=abcd, is X Ends With d (yes = PASSED) X=abcd, is X Ends With b (no = FAILED)

Usually the left value is an attribute, the right value is a constant; however, right and left values can be either constants or attributes.

For example,

X=abcd, is 'abcde' Contains X (answer yes = PASSED)

X=5, Y=1,2,3,4 is X In Y (answer no = FAIL)

### Examples

Comparison Mode is set to Automatic:



@Area() < 100

&numLanes > 2

Comparison Mode is set to String:

"Joe" = "Jerry"

## Usage Notes

At run-time, the Tester invokes numeric comparisons or string comparisons, as greater than and less than, that have different meanings depending on the type of operands. If both arguments may be converted to numbers, then numeric comparisons will be used; otherwise, string comparisons will be used.

If multiple Tests or String Tests are present, the test will only pass if all of the Tests are true, unless the Pass Criteria is One Test (OR). By default, the Pass Criteria used between multiple tests is All Tests (AND).

The String Comparison Mode forces string comparison of the equation, and the Automatic Comparison Mode (which is the default) indicates that a basic evaluation of the tests is performed.

## Testing for TEST

You cannot directly test for the value of "TEST". However, you can perform the test by following the steps below:

- Use an AttributeCreator to add a new attribute, and set its value to TEST.
- Use the value of the new attribute to test against.

[Click here to see a workspace example.](#)

## Transformer Category

Filters

## fmepedia

See fmepedia for additional information about this transformer.

## Transformer History

This transformer replaced the AttributeTester and GenericTester transformers.

## Technical History

Associated FME function or factory: TestFactory

## **TextAdder**

Category: Manipulators

Sets the feature's geometry to text with the previous geometry as the text location.

## **Technical History**

Associated FME function or factory: @Geometry

## **TextLocationExtractor**

Category: Manipulators

Sets a text feature's geometry to the location of the text.

### **Technical History**

Associated FME function or factory: @Geometry

## **TextPropertyExtractor**

Category: Manipulators

Sets the given attributes to the properties of a text geometry.

### **Technical History**

Associated FME function or factory: @Geometry

## **TextPropertySetter**

Category: Manipulators

Sets the properties of a text geometry to the specified properties. All parameters are optional; if a value is unspecified, it will be left unmodified on the geometry.

### **Technical History**

Associated FME function or factory: @Geometry

## TextStroker

Takes as input a text string, rotation, height and width multiplier, and outputs aggregates that describe the outline of the text. If these values are not supplied, then the FME generic text attributes `fme_text_string`, `fme_rotation` and `fme_text_size` are used respectively.

The font name can refer to any valid truetype font, and is case-insensitive. The font name can also include optional style specifiers according to the syntax:

```
<fontname>[,SIZE][,BOLD][,ITALIC][,<character set index within font>]
```

The [ ] enclose optional items. The character set index, which is an integer starting at 0, is typically not used.

Some example font specifications include:

Arial	Regular Arial
Arial,16	Regular Arial, 16 pt
Arial,ITALIC	Italic Arial
Arial,BOLD	Bold Arial
Arial,BOLD, ITALIC	Bold and Italic Arial
Arial,16,BOLD,ITALIC	Bold and Italic Arial, 16 pt

The fontname can also be taken from the value of an attribute. The value is taken as the full font specifier and can include the style information as well.

The font height and padding are measured in ground units. The font width multiplier multiplies the 'natural' width for the font, based on the height. A multiplier of two will give characters at twice their normal width.

The Output geometry type parameter specifies whether the output features should be polygons or lines.

Stroked text is generated via the stroked output, and any rejected features (i.e., points with no text attribute) are generated via the untouched output.

## Transformer Category

Manipulators

## FME Licensing Level

FME Professional edition and above

## Technical History

Associated FME function or factory: TextStrokerFactory

## **TextureCoordinateSetter**

Assigns texture coordinates to surfaces.

### **Parameters**

Texture Mapping Type

If you choose Surface Normal, then the texture coordinates are assigned to the sloped surfaces.

For composite surfaces, each surface will be treated separately since the parts can have different slopes. If you choose From Top View, the texture coordinates are assigned to the surfaces as if the surfaces are flat on the ground (that is, only X and Y coordinates are considered). In this mode, a composite surface is considered as one single geometry when the texture coordinates are applied.

Width Scale Factor and Height Scale Factor

Use these parameters to specify the number of times the texture is repeated in rows and columns, respectively.

Horizontal Offset and Vertical Offset

Use these parameters to specify horizontal and vertical offset.

### **Usage Notes**

- This transformer works only with surfaces, not including triangle strips and triangle fans.
- Multi-surfaces are not yet supported.

### **Transformer Category**

Calculators

### **Technical History**

Associated FME function or factory: @Geometry

## Tiler

Chops the input features into a series of tiles.

Features that span multiple tiles will be clipped into multiple features. Features that lie on the boundary between tiles will be output once in each tile. If this is not the desired behavior a DuplicateRemover transformer can be used.

This transformer works with both raster and vector data.

### Parameters

#### Tile Width/Tile Height

The Tile Width and Tile Height parameters specify the size of each tiles in ground units.

Note: If the source is a raster, does not have a coordinate system and is not georeferenced, then the raster extents and spacing are adjusted to default values. The horizontal and vertical spacing are set to one column and one row, respectively, and the origin is set to (0,0).

**IMPORTANT:** If attributes are used for the Tile Width or Tile Height parameters, the value will be taken from the first feature to enter the transformer. An error will occur if this feature does not have these attributes.

#### Column Attribute/Row Attribute

Each feature output from the Tiler will have a row and a column attribute added, specifying the zero-based row and column that the feature fell into. Row 0, Column 0 corresponds to the tile in the bottom-left corner.

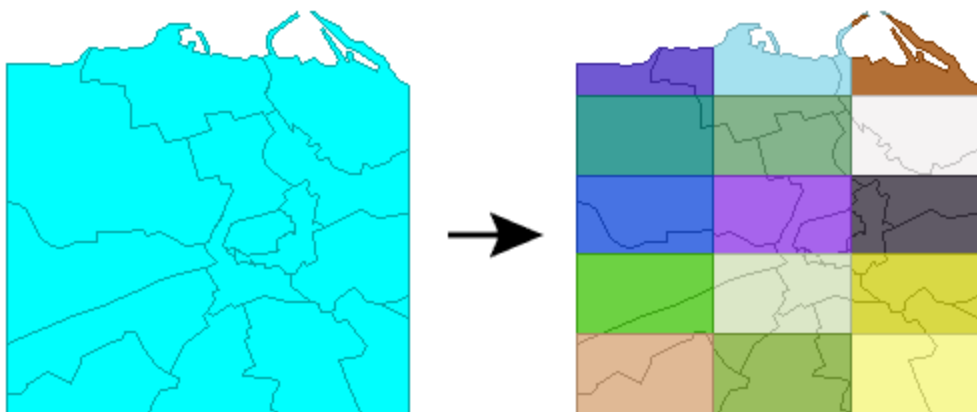
#### Mode Type

- Quick: Computes intersections in the input only.
- Complete: Computes intersections in both the input and the output.

### Geometry Handling

If the Geometry Handling Advanced setting is set to "Enhanced" in the workspace, ellipses can be tiled without stroking; otherwise, all ellipses will be stroked prior to being tiled.

### Example



### Transformer Category

Geometric Operators



## **Technical History**

Associated FME function or factory: ClippingFactory, BoundingBoxFactory

## TimeStamper

Adds a time stamp to a feature as a new attribute. The format of the time stamp is set as a parameter of the transformer.

Conversion specifiers are introduced by a ^ character and are replaced in the format as follows:

^a	The abbreviated weekday name according to the current locale.
^A	The full weekday name according to the current locale.
^b	The abbreviated month name according to the current locale.
^B	The full month name according to the current locale.
^c	The preferred date and time representation for the current locale.
^d	The day of the month as a decimal number ranging from 00 to 31. <div style="border: 1px solid black; padding: 5px; background-color: #f0f0f0;">Note that leading zeros will be added when the day of the month is less than 10. If you want to suppress the leading zero, enter ^#d.</div>
^H	The hour as a decimal number using a 24-hour clock ranging from 00 to 23.
^I	The hour as a decimal number using a 12-hour clock ranging from 01 to 12.
^j	The day of the year as a decimal number ranging from 001 to 366.
^m	The month as a decimal number ranging from 00 to 12. <div style="border: 1px solid black; padding: 5px; background-color: #f0f0f0;">Note that leading zeros will be added when the month is less than 10. If you want to suppress the leading zero, enter ^#m.</div>
^M	The minute as a decimal number.
^p	Either a.m. or p.m. according to the given time value, or the corresponding strings for the current locale.
^S	The second as a decimal number.
^s	Seconds from epoch.
^U	The week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week.
^W	The week number of the current year as a decimal number, starting with the first Monday as the first day of the first week.
^w	The day of the week as a decimal, with Sunday being 0.
^x	The preferred date representation for the current locale without the time.
^X	The preferred time representation for the current locale without the date.
^y	The year as a decimal number without a century ranging from 00 to 99.
^Y	The year as a decimal number including the century.
^Z	The time zone or name or abbreviation.

## Transformer Category

Strings

## Technical History

Associated FME function or factory: @Timestamp

## TINGenerator

Generates a Delaunay surface model and outputs the defining Triangulated Irregular Network (TIN). The TIN is output as both Triangles and TIN edges.

### Input

- **POINTS:** The vertices from the 3D points are used to define the surface model. Each vertex of the input feature is used to define the underlying model. This tag also has the ability to accept FME grid features. Points are extracted from a grid, and the vertices of these 3D points are used to define the surface model.
- **BREAKLINES:** These features are added into the model as breaklines, such that triangle edges will always be along the line of the feature.
- **3D\_LINES:** The vertices of the 3D line are inserted into the model to define the surface. They are not used to define breaklines but rather just provide 3D data for the model from their vertices.

### Output

- **TIN\_EDGES:** Outputs all the edges that define the underlying Delaunay Triangulation. Each feature output on this clause has the attributes *SurfaceModel.vertex1\_id* and *SurfaceModel.vertex2\_id*, which identify the ID of the vertex to which it was connected.
- **TRIANGLES:** Returns the Triangles that define the Delaunay Triangulation. Each feature output on this tag has the following attributes:
  - *SurfaceModel.vertex1\_id*, *SurfaceModel.vertex2\_id*, and *SurfaceModel.vertex3\_id* which identify the IDs of the vertices
  - *SurfaceModel.slope*, which is the slope of the triangles in degrees
  - *SurfaceModel.aspect*, which is the aspect angle in degrees measured from the x-axis in a counter-clockwise direction
  - *SurfaceModel.percentageSlope*, which is the percentage slope
- **TIN\_SURFACE:** Returns the Triangles that define the Delaunay Triangulation as one unified mesh geometry.
- **VERTEX\_POINTS:** Outputs each vertex of the underlying surface model triangulation. Each triangle is tagged with the unique identifier *SurfaceModel.vertex\_id*. This clause differs from the VERTEX\_POINTS output clause as it only outputs those point features that were accepted through the POINTS input clause (that is, those points that have attributes with the attributes maintained). Any points that are introduced into the model through breaklines processing will not be output on this clause.

### Parameters

#### Group By

This option allows you to select attributes that define which groups to form. The default behavior is to use the entire set of features as the group.

#### Surface Tolerance

The surface tolerance is used when building the surface to determine which vertices to add to the surface model. Specifying a value of 0 turns off the vertex filtering so that all vertices (except those with duplicate x,y) are added to the model.

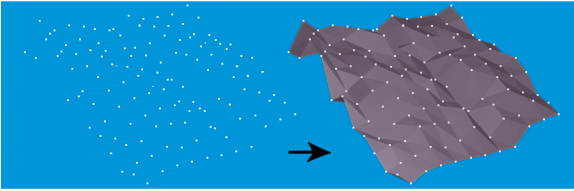
When specified, the surface tolerance is used during the construction phase of the surface model to determine whether a vertex will be added to the model, or whether it will be discarded and not added to the model.

The surface tolerance works as follows, for each vertex that is being added to the model:

- If the x,y location is outside the convex hull of the existing model, it is added to the model.
- If the x,y location is inside the existing model:

- The difference between the z value from the existing TIN and the z value of the vertex is calculated.
- This difference is compared to the surface model tolerance.
- The vertex is only added to the model if the difference is greater than the surface tolerance; otherwise, the vertex is discarded.

### Example



### FME Licensing Level

FME Professional edition and above

### Transformer Category

Surfaces

### Technical History

Associated FME function or factory: SurfaceModelFactory

## TopologyBuilder

Computes topology on input point, line, and/or area features.

This transformer is typically used to determine topological relationships to aid in decision making in later transformers.

This transformer does not assume that all input data is clean and noded properly unless Assume Clean Data is set to Yes. It takes any data and constructs the resulting topology after computing any intersections that are present in the input data.

### Output Ports

The TopologyBuilder outputs the significant NODEs and LINEs with attributes describing their topological relationships. AREAs are output with information about the LINEs that form them.

Note: Unlike most transformers, you cannot name these attributes.

Outputs	Output Feature Properties
NODE	NODENUM NODEANGLE{ All topologically significant nodes are output. They have the attributes NODE_NUMBER_ATTR and ANGLE_PREFIX_ATTR specified as directed by the factory clauses.
LINE	ARC_ID RIGHT/LEFT_POLY RIGHT/LEFT_EDGE FROM/TO_NODE POLYLIST All topologically significant lines are output. They have the attributes FROM_NODE_ATTR, TO_NODE_ATTR, RIGHT_POLY_ATTR, LEFT_POLY_ATTR and POLYGONS_ATTR specified as directed by the factory clauses.
POLYGON	POLY_ID PERIMETER AREA ARCLIST The polygonal entities constructed are output here. These have a list of the LINEs which make up the polygon along with the actual geometry.
UNIVERSE_POLYGON	<none> This is the polygon that represents everything not covered by any of the input polygons. This feature consists of a list of the defining LINE identifiers that make it up but does not have any associated geometry.

### Parameters

#### Group By

The input features may be grouped into separate topology sets based on attribute values. All attributes are carried across from the INPUT features to the output features.

#### Maximum Coords Per Line

The number indicates the maximum length to output any line. If any line contains more than this number of coordinates, it will be broken into pieces which are output separately, each with their own line IDs, and correctly noded.

#### Unify Attributes From Overlapping Input

If set to Yes, the transformer enters a mode where no collinear lines or overlapping points are output at all, whether they came from source linear features or from the borders of source area features or input points, or calculated as intersection points. In this mode, all output lines or

points which were overlapping with at least one direct input will contain a list attribute with information about each input with which it was overlapping. This keyword sets the fieldname of the list attribute to contain all the attributes (that do not start with "fme\_") from all of the input lines or points that were overlapping with the final output line or point.

A side effect of this option is that only arcs that form part of a polygon boundary will be considered in the calculation of LEFT\_EDGE\_ATTR and RIGHT\_EDGE\_ATTR. (All arcs originating only from line input will have their own ID supplied as their left edge ID, and the negation of this as their right edge ID.)

### Provide All Bounding Arcs on Output Polygons

If this is yes, the universe polygon returned will have complete geometry on it. If it is no, the universe polygon will take less time to create because it will have no geometry, but will have links to the arcs that form the boundary of the polygon. The default is no.

### Propagate All Attributes From Input

If Propagate All Attributes From Input is set to Yes, attribute lists are added to each output feature, composed of attributes from the relevant input features. For each node, this will be a list of lines and a list of polygons touching the node; for lines, there will be a list of nodes and a list of polygons; and for polygons, a list of nodes and a list of lines. The base names for the lists will be `_nodes`, `_lines`, and `_polygons`.

Note that for these lists to be manipulated in Workbench, they must first be exposed using an AttributeExposer. Be sure to include "{}" in the exposed attribute names, e.g.: `"_nodes{}"`.

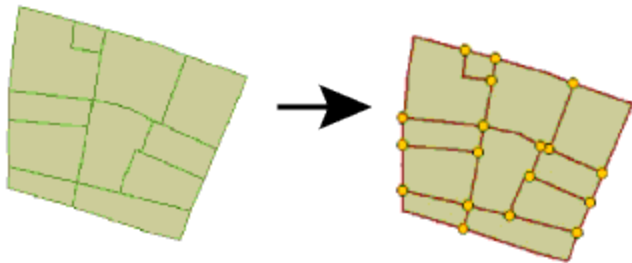
### Preserve Internal Edges (Advanced)

Preserve Internal Edges (Advanced) specifies that coordinate "cycles" within a polygon are allowable and will be preserved. A "cycle" is an edge that occurs twice in the same polygon's boundary (once in each direction); the edge's ID will appear twice in that polygon's edge list, positive in one instance and negative in the other.

### Assume Clean Data (Advanced)

Assume Clean Data (Advanced) specifies that the input is topologically clean and noded properly and therefore no intersection will be performed. Otherwise, the intersections of the data are computed prior to constructing the topology.

### Example



### Transformer Category

Geometric Operators

### Technical History

Associated FME function or factory: TopologyFactory

## TransporterReceiver

Used in conjunction with TransporterSender.

Provides a mechanism to send features to another FME workspace running in a different process which may be located on the same or a different machine. To transport features from one FME process to another, use the TransporterSender to send features to a TransporterReceiver transformer. The TransporterSender is in the source FME workspace and the TransporterReceiver is in the receiving workspace.

Here are the rules when setting up a feature stream between two FME workspaces using the TransporterReceiver and TransporterSender transformers.

1. One workspace is deemed to be the "establisher" of the transport stream. If there are multiple transport streams between workspaces, then one workspace must be the establisher of all the streams, and the other will connect to those transport streams. This is regardless of whether the individual transporters are sending or receiving data. The workspace that is the establisher must be started before the workspace with the transporters that are connecting. A transporter cannot successfully connect to a transport stream that is not established.
2. For each transport channel, there must be a different port that is used for each server machine.
3. There can be an arbitrary number of transporters used within a workspace.

The **Transfer Mode** parameter specifies if the transformer should expect the features' data to be sent over the TCP/IP connection or to be saved to disk with the file name sent over the TCP/IP connection. (Note that a TCP/IP connection is required for both modes.)

The **Initiation Sequence** parameter specifies which of the Transporter endpoints is responsible for establishing the Transport stream and which is responsible for connecting to it. The workspace that establishes the workspace must be up and running before the connector, hence if there are multiple transport streams between two workspaces, then this must be consistent.

The **Establishing Host** parameter is only needed by the workspace doing the "CONNECT"ing and only when the workspaces are running on different machines.

We recognize that one of the big uses of this transformer will be local to a machine in order to partition large workspaces into smaller components that each have their own address space.

The **Port or Service to Use** parameter can specify either a port number or a service that is to be used. A default value is specified.

### Transformer Category

Infrastructure

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: OverlayFactory

## TransporterSender

Used in conjunction with TransporterReceiver.

Provides a mechanism to send features to another FME workspace running in a different process which may be located on the same or a different machine. To transport features from one FME process to another, use the TransporterSender to send features to a TransporterReceiver transformer. The TransporterSender is in the source FME workspace and the TransporterReceiver is in the receiving workspace.

Here are the rules when setting up a feature stream between two FME workspaces using the TransporterReceiver and TransporterSender transformers.

1. One workspace is deemed to be the "establisher" of the transport stream. If there are multiple transport streams between workspaces then one workspace must be the establisher of all the streams, and the other will connect to those transport streams. This is regardless of whether the individual transporters are sending or receiving data. The workspace that is the establisher must be started before the workspace with the transporter's that are connecting. A transporter cannot successfully connect to a transport stream that is not established.
2. For each transport channel there must be a different port that is used for each server machine.
3. There can be an arbitrary number of transporters used within a workspace.

The **Transfer Mode** parameter specifies if you want to transfer data over a TCP/IP connection or via files saved to a shared disk (both require a TCP/IP connection). If TCP/IP is selected, all feature data will be transferred over the network connection. If FILE is selected, the feature data will be saved to disk and the location of the file will be sent over the TCP/IP connection.

The **Initiation Sequence** parameter specifies which of the Transporter endpoints is responsible for establishing the Transport stream and which is responsible for connecting to it. The workspace that establishes the workspace must be up and running before the connector hence if there are multiple transport streams between two workspaces then this must be consistent.

The **Establishing Host** parameter is only needed by the workspace doing the "CONNECT"ing and only when the workspaces are running on different machines.

We recognize that one of the big uses of this will be local to a machine in order to partition large workspaces into smaller components that each have their own address space.

The **Port or Service to Use** parameter can specify either a port number or a service that is to be used. A default value is specified.

The **File Name** parameter is only used when **Transfer Mode** is set to FILE. It specifies the base file name to use. Each file that is created will have a -<integer> appended to the end of the name to keep all file unique. Example: if the File Name is "features.fft" then the first file created will be features-0.fft and the second features-1.fft and so on. The full path of the file will be passed to the receiver and the receiver must be able to access the file from the same path (drive letter included).

The **Features to save per file** is only used when **Transfer Mode** is set to FILE. It specifies how many features to write to a file before send the location of the file to the receiver.

### Transformer Category

Infrastructure

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: TransporterFactory



## **Triangulator**

Breaks an input geometry into triangular units or a mesh.

For 2D geometries, the triangulation is performed with respect to the X-Y plane.

For 3D geometries, such as faces, the triangulation is performed with respect to the normal direction of each surface.

Areas and circular segments will be converted into a linear equivalent in this transformer.

### **Output Ports**

- Triangulated features are output through the TRIANGLES port
- Triangulated mesh geometries are output through the TIN\_SURFACE port.
- Unprocessed geometries are output through the UNTOUCHED port.

### **Transformer Category**

Infrastructure

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: TriangulationFactory

## **Tweeter**

Sends a Twitter status update from Workbench.

### **Output Ports**

Upon completion, the output feature will have several new attributes:

*\_twitter\_response*: Contains the JSON response text from the server. The value of this attribute can be explored using the JSONExploder or JSONExtractor transformers.

*\_twitter\_status\_id*: Contains the integer ID of the status update.

*\_twitter\_status\_truncated*: Contains a Boolean string indicating whether or not the status update had to be trimmed to 140 characters.

### **Parameters**

#### ***Twitter Settings***

##### Twitter Status

The Twitter Status parameter can be used to enter a status update directly, or to enter the name of an attribute which contains the status update.

##### Twitter Username and Password

Enter a Twitter account username and password.

##### Include Geometry in Status Update

If this parameter is set to Yes, the #WKT hashtag will be appended to the status update, along with the feature geometry, formatted as OGC Well Known Text.

If this results in the status update being longer than 140 characters, the text of the status update will be truncated, not the geometry. If geometry is included in the status update, the feature will be reprojected to the LL84 coordinate system before the status update is sent. Features with no coordinate system are assumed to already be in the LL84 coordinate system.

##### Proxy URL, Proxy Port, Proxy Username, Proxy Password, Proxy Authentication Method

These optional parameters may be set for organizations that require Internet access via an HTTP proxy server.

### **Related Transformers**

JSONExploder

JSONExtractor

### **Transformer Category**

Web Services

## **TweetSearcher**

Runs a search for Twitter™ entries that contain the given query.

### **Output Ports**

Upon completion, the transformer will output a new feature for each status update. Each output feature will contain several attributes:

*\_tweet\_id*: The id number of the search result.

*\_tweet\_user*: The name of the Twitter user who created the search result.

*\_tweet\_text*: The text of the search result.

*\_tweet\_created\_at*: The date and time of the status update. This will be formatted in the standard FME date/time format: YYYYMMDDHHMMSS

*\_tweet\_search\_result*: The JSON text of the search result. This can be further examined via the JSONExploder and JSONExtractor transformers.

### **Parameters**

#### ***Twitter Settings***

##### Twitter Search Query

Enter a search query directly, or select the name of an attribute that contains the query.

##### Maximum Number of Results

This parameter may be used to specify an upper bound on the number of search results returned, and therefore the maximum number of features output from the transformer.

##### Search for Geometry in Status Update

If this parameter is set to Yes, the transformer will search each status update for OGC Well Known Text, and if found, will set the feature geometry accordingly. Note that the transformer will only search for geometry in status updates containing the #WKT hashtag.

#### ***Proxy Settings***

##### Proxy URL, Port, Username, Password, Authentication Method

These optional parameters may be set for organizations that require Internet access via an HTTP proxy server.

### **Related Transformers**

JSONExploder

JSONExtractor

### **Transformer Category**

Web Services

## TwitterStatusFetcher

Retrieves the Twitter status updates for a particular user.

### Output Ports

Upon completion, the transformer will output a new feature for each status update. Each output feature will contain several attributes:

*\_tweet\_id*: The id number of the status update.

*\_tweet\_status*: The text of the status update.

*\_tweet\_created\_at*: The date and time of the status update. This will be formatted in the standard FME date/time format: YYYYMMDDHHMMSS

*\_tweet\_search\_result*: The JSON text of the status update. This can be further examined via the JSONExploder and JSONExtractor transformers.

### Parameters

#### *Twitter Settings*

Twitter Username to Fetch

Enter the user whose updates are to be retrieved. You can also choose an attribute that contains the username.

Twitter Username and Password

If the transformer is being used to fetch the status updates from a protected user, a valid Twitter username and password must be entered.

Search for Geometry in Status Update

If this parameter is set to Yes, the transformer will search each status update for OGC Well Known Text, and if found, will set the feature geometry accordingly. Note that the transformer will only search for geometry in status updates containing the #WKT hashtag.

#### *Proxy Settings*

Proxy URL, Port, Username, Password, Authentication Method

These optional parameters may be set for organizations that require Internet access via an HTTP proxy server.

### Related Transformers

JSONExploder

JSONExtractor

### Transformer Category

Web Services

## **UUIDGenerator**

Category: Strings

Calculates a UUID (Universally Unique Identifier) for each incoming feature, and adds it as a new attribute. The UUID is expressed as a string consisting of 8 hexadecimal digits, each followed by a hyphen, then three groups of 4 hexadecimal digits, each followed by a hyphen, then 12 hexadecimal digits. It is 36 bytes in size. UUIDs look like:

**7672aac8-fa0b-464c-b0b8-3efa9ae9cd86**

This transformer is similar to the GOIDGenerator, which generates a unique ID for each feature partially based on that feature's geometry.

### **Technical History**

Associated FME function or factory: @UUID

## **URLDecoder**

Decodes a string from its URL-encoded form and stores the result in an attribute.

### **Parameters**

String to Decode

The string to be decoded, or the name of the attribute containing the string.

Destination Attribute

This attribute will store the encoded string.

### **Transformer Category**

Web Services

### **Technical History**

Associated FME function or factory: @Http

## **URLEncoder**

Converts a string value to its URL-encoded form and stores the result in an attribute.

### **Parameters**

String to Encode

The string to be encoded, or the name of the attribute containing the string.

Destination Attribute

This attribute will store the encoded string.

### **Transformer Category**

Web Services

### **Technical History**

Associated FME function or factory: @Http

## **VariableRetriever**

Category: Infrastructure

---

Note: Previously called the VariableRasterClassifier

---

Reads the specified variable and puts its value into the specified attribute.

The variable must have been previously set using the VariableSetter transformer.

The 'Variable Scope' parameter specifies whether the scope of this variable is Global or Local. Globally-scoped variables can be accessed by a VariableRetriever anywhere in the workspace, whereas locally-scoped transformers can only be accessed within the custom transformer they are created in.

### **Technical History**

Associated FME function or factory: @GlobalVariable



## **VariableSetter**

Category: Infrastructure

Creates and sets the specified variable to the specified value.

The variable can later be read back into an attribute using the `VariableRetriever` transformer.

The 'Variable Scope' parameter specifies whether the scope of this variable is Global or Local. Globally-scoped variables can be accessed by a `VariableRetriever` anywhere in the workspace, whereas locally-scoped transformers can only be accessed within the custom transformer they are created in.

## **Technical History**

Associated FME function or factory: `@GlobalVariable`

## VectorOnRasterOverlayer

Overlays vector features onto a single raster feature by drawing them onto the resulting output raster. The properties of the output raster are identical to that of the input raster.

### Input

- **VECTOR:** The vector features which will be rasterized onto the resultant raster. The `fme_color` attribute of the input vector features is used to generate pixel values for color bands. Pixel values for red, green, and blue bands will be taken from the corresponding component of a feature's `fme_color` attribute. Pixel values for gray bands will be the average of the `fme_color` components. Pixel values for alpha bands may be specified through the Alpha Value parameter. The Z coordinates of the input vector features are used to generate pixel values for numeric bands.
- **RASTER:** The feature to use as the background raster of the resultant raster. This must be a raster feature or an error will occur. Using this input is optional.

### Output

- **RASTER:** The raster drawn from a group of features.

### Parameters

#### Group By

If the Group By parameter is set to an attribute list, one raster per group will be produced.

#### Composite Using Alpha Band

If Composite Using Alpha Band is set to Yes, rasters will be expected to have an alpha band selected. Vector features will then be blended with the underlying raster according to their alpha values, rather than just overwriting the underlying raster.

#### Anti-Aliasing

If this parameter is Yes, the output lines will be smoothed using an anti-aliasing algorithm.

#### Tolerance

The Tolerance parameter is the maximum normalized distance from a line segment or polygon vertex to a pixel to be rendered. For example a tolerance of 1.0 will draw all pixels touched by the input vector line, while a tolerance of 0.0 will draw only those pixels where the input vector line passes directly through their center. Tolerance can only be selected when anti-aliasing is off.

### Usage Notes

This transformer supports raster band selection. The `RasterSelector` can be used to modify selection.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: `VectorToRasterFactory`

## VoronoiCellGenerator

Generates a Voronoi diagram that represents the closest points around point locations. The diagram is such that the Voronoi cells only radiate out from the cell at a distance specified by the radius parameter.

Point features representing cell center point locations are input with the following data contained in the chosen attributes:

- site name – This is the identifier of the cell site, and the center of each Voronoi.
- azimuth – This is angle that the cell points. If there are multiple sectors for a cell, then this angle will form the center angle for the sector polygon.
- sector name (optional)
- radius – This is the distance around each cell that the Voronoi diagram will radiate. (Alternatively, a universal radius can be specified, instead of a field name.)

For multiple points having the same site name, sector shaped polygons are generated. The center point of each sector will be the average of all input points for that site. The start and end angles of each sector will be the average of the sector's azimuth (defined in degrees clockwise from north), and the azimuth(s) of the adjacent sector(s). If multiple sectors within a site have the same azimuth, their shapes will be identical (and created as though there were only one sector with the given azimuth in the site).

These sectors are output via the SITE\_SECTORS port. The SITE\_SECTORS may overlap. For sectors that overlap, the transformer also truncates sectors so that only those portions of the sector which are closest to the cell are output via the CELLS port.

If a point enters the VoronoiCellGenerator and is farther from other points in its site than the specified **Maximum Distance Between Site Points**, it will be output on the DISTANT\_POINTS port.

Input points not containing all required attributes, or having invalid values in the azimuth (which must be nonnegative and less than 360) or radius (which must be positive) will be output on the INCOMPLETE\_POINTS port.

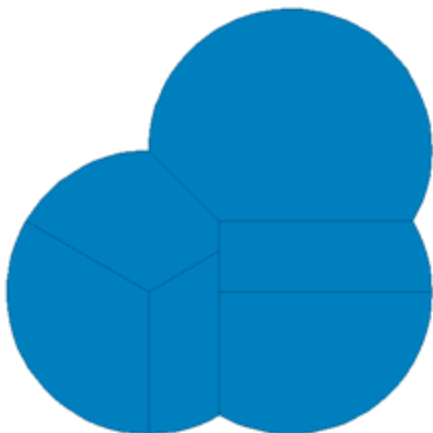
Input features with non-point geometry are output on the ILLEGAL\_GEOM port.

If the SITE\_POINTS port is used, then for each input site, a point will be generated which is the average of all input points for the site.

If a **Sector Name** parameter is specified, then each feature entering the VoronoiCellGenerator will be checked to see if its sector name is already in use within its site. If it is, then it will be output on the EXTRA\_POINTS port.

**Ensure Topological Correctness:** If this parameter is set to Yes, the output will form a well-noded coverage, as would be expected by the TopologyBuilder, NeighborColorSetter, and similar transformers. If it is set to No, there may be slight overlaps or gaps between sectors, making the output unsuitable for some geometric operations.

## Example



## FME Licensing Level

FME Professional edition and above

**Transformer Category**

Surfaces

**Technical History**

Associated FME function or factory: SectorFactory

## VoronoiDiagrammer

Generates a Voronoi diagram or Thiessen polygon from the input POINTS.

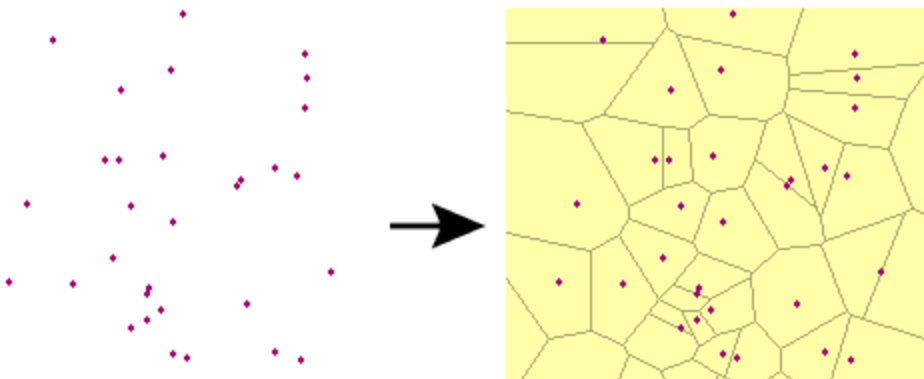
A Voronoi diagram is a set of polygons that represent proximity information about a set of input points. Each polygon in the diagram defines the area of space that is closest to a particular input point.

The extent of the Voronoi diagram is guaranteed to cover the bounding box of all the input POINT features.

Features input to the optional MINIMUM\_VORONOI\_EXTENT port will expand the extent of the resulting diagram to include their bounding box. This provides a way of extending the bounds of the diagram well past the extent of the input points.

If a Voronoi diagram is to be made from points with elevations, and you want to add additional breakline and tolerance options, then you should use the SurfaceModeller transformer.

### Example



### FME Licensing Level

FME Professional edition and above

### Transformer Category

Surfaces

### Technical History

Associated FME function or factory: SurfaceModelFactory

## WebCharter

Creates a URL that can be used to obtain a chart of the specified data as a PNG image from the Google Chart API, as documented at <http://code.google.com/apis/chart>.

Note: Google, and Google Chart API are trademarks of Google, Inc. Use of the Google Chart API is subject to the Terms of Service for the API.

One URL is created for each feature that enters the transformer. The data for the chart is taken from the list specified, and each element of the list must be numeric. Only one data series is currently supported (i.e., there is no ability to overlay two numeric sets of data on the same chart).

The transformer supports a subset of the Google chart types: Line charts, Bar Charts, and Pie Charts:

- If a Line chart is chosen, then each list value is used as a y value, and the list element numbers are used as the x values. If data labels are provided in a list with a parallel structure to the data list, these will be placed below the x axis. The number of labels can also be limited in this case as well. Optional chart ranges can be used to fix the y axis range, otherwise, the range will be adjusted to go between the min and max values in the dataset.
- If a Bar chart is chosen, then a bar is created for each list value. If data labels are provided in a list with a parallel structure to the data list, these will be used to label the bars. Optional chart ranges can be used to fix the bar ranges, otherwise, the range will be adjusted to go between the min and max values in the dataset (which means one bar will have a 0 length). The bars can be set to run horizontally or vertically, depending on the chart type chosen.
- If a Pie chart is chosen, one slice of the pie will be created for each list value. If data labels are provided in a list with a parallel structure to the data list, these will be used to label the slices. Flat or 3d pie charts can be created, depending on the chart type chosen.

The **Chart Height and Width** control the size, in pixels, of the image that the Google Chart API will return. Note that the chart API will truncate the chart if the size is not large enough, or does not have the correct aspect ratio. In particular, labelled pie charts typically must be wider than they are high. Further note that according to the the Chart API documentation, the number of pixels must not exceed 300,000.

A **Chart Title** may optionally be specified, it must not contain any special characters that cannot appear in a URL. As well, if a two-line title is desired, it can be created by placing a | character as the line separator.

An optional **Data Color** can be specified and will be used as the color for the data portion of the resulting chart. The parameter can be edited by clicking the colored square to the right of the text field, or by editing the contents of the field directly. The color must be specified as <red>,<green>,<blue> where each of <red>, <green>, and <blue> is a number between 0 and 1.

The **Chart Data Encoding** parameter controls which mechanism is used to convert the numeric data to be charted into a Google Chart API representation. The Simple method scales the data into 62 discrete values, while the Extended method provides a resolution of 4096 different values. Since the Extended method uses a . in its representation, some software (notably some builds of Google Earth) may not be able to correctly handle the URLs it generates, and so the Simple method is preferable unless the greater resolution is required.

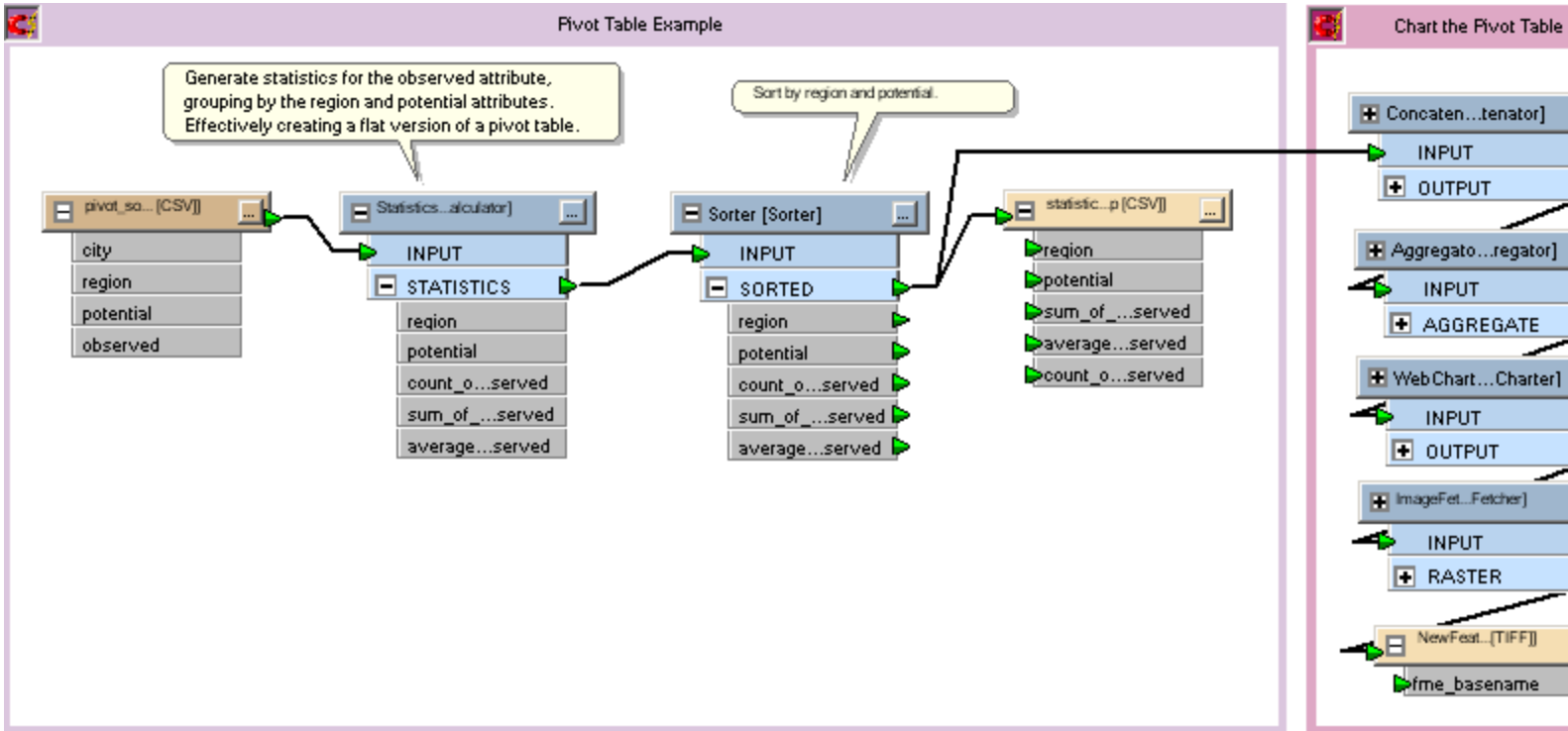
Additional Google Chart API directives may be appended onto the URL by subsequent transformers such as the StringConcatenator to add additional styling and color customization, as permitted by the API.

The ImageFetcher transformer can be used to retrieve the chart produced by the URL and convert it into a raster feature for further processing and output. Another potential use of the URL, when writing KML, is to attach dynamic charts to placemarks by including it as part of the kml\_description.

### Example: Creating an FME Chart from a Pivot Table

Following the example started in the StatisticsCalculator, you can take the pivot table a bit further and generate a chart using the WebCharter transformer. This provides us with the basis for some interesting summarization and reporting tools.

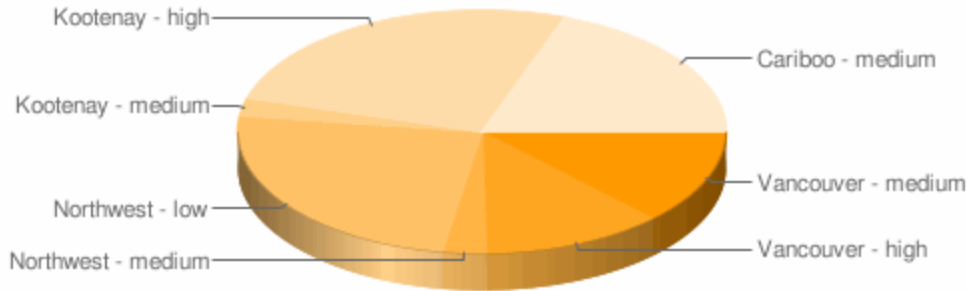
The workspace uses a WebCharter transformer to create a chart of a pivot table.



The chart created from the FME generated pivot table.

(Note that the data is fictitious.)

Sum of Observed Mineral values grouped by Region and Potential Impact



Another good example is shown on this [fmepedia](#) page.

### Transformer Category

Web Services

### Technical History

Associated FME function or factory: @Tcl2

## WebMapTiler

Creates a series of image tiles that can be utilized by Microsoft Virtual Earth. This is done by resampling rasters to various different resolutions and then splitting them into tiles. This transformer produces similar output to the MSR MapCruncher utility.

### Parameters

#### Minimum and Maximum Zoom Level

These parameters specify the zoom levels for which tiles will be generated.

Level 1 is the lowest level of detail; at that level, the entire world map is 512 x 512 pixels. Each level increases the number of rows and columns by a factor of two: level 2 is 1024 x 1024, level 3 is 2048 x 2048, etc.

Level 23 is the highest level of detail. If Minimum Zoom Level is unspecified, the minimum zoom level will default to 1. Note that tiles will not be generated if the input raster covers less than 1 row and 1 column at a particular zoom level, regardless of the minimum zoom level value. If Maximum Zoom Level is unspecified, the maximum zoom level will be the smallest zoom level such that the resampled raster has more rows or columns than the original raster.

#### Interpolation Type

Cell values are interpolated in order to change the raster to the specified size. You can choose from these interpolation methods.

- Nearest Neighbor is the fastest but produces the poorest image quality.
- Bilinear provides a reasonable balance of speed and quality.
- Bicubic is the slowest but produces the best image quality.
- Average 4 and Average 16 have a performance similar to Bilinear and are useful for numeric rasters such as DEMs.

#### Quadkey, Zoom Level, Tile Column, and Tile Row Attributes

If specified, these are attributes that will be added to output features. Quadkeys are used by Bing Maps to uniquely identify a single tile at a particular level of detail. The zoom level, tile column, and tile row attributes are an alternative way to uniquely identify a tile which can be used by Google Maps. Generally, when writing out the rasters generated by this transformer, one would fanout the destination feature type on the quadkey for Bing Maps or a combination of the zoom level, tile column, and tile row for Google Maps. The PNGRASTER writer is recommended for the best results.

#### Raster Index Attribute

If this attribute is specified, an attribute will be added to each output feature that identifies which raster it was created from. This index is zero-based, so all tiles created from the first input raster will have a value of 0, all tiles created from the second input raster will have a value of 1, etc.

### Usage Notes

- This transformer accepts only features that have raster geometry and is unaffected by raster band and/or palette subselection.
- Additionally, this transformer only accepts features that are in the EPSG:3785, EPSG:900913, or SPHERICAL\_MERCATOR coordinate systems. All features must be reprojected into one of these coordinate systems prior to entering this transformer.

### Transformer Category

Rasters

### FME Licensing Level

FME Professional edition and above

### Transformer History

This transformer was previously named the VirtualEarthTiler.



## Technical History

Associated FME function or factory: VirtualEarthTileFactory

## WhiteStarLeaseBuilder

Posts a query to a WhiteStar Legal2Map™ WebServices (WS3) server to obtain points or polygons matching a list of legal land descriptions.

A list attribute ("Request Description Attribute") on the input feature provides the set of textual property descriptions which form the request query. The submitted query will result in a set of points representing corresponding well locations, or a set of polygons representing the area described by the request, depending on the value of the "Resulting Geometry Type" selector.

If the list attribute contains a single element, and that element is an integer, it will be assumed to be the transaction ID of a previous query. In this case, a request will be performed to retrieve the results of the query corresponding to the given transaction ID.

The query is sent to the Web Service host using "basic" username/password authentication over an HTTPS connection. An optional HTTP proxy server may also be specified.

A feature is emitted for each point or polygon returned from the query. Aside from the geometry, each of these features contains attributes for a legal property description ("Result Description Attribute") and a unique numeric identifier for the transaction just completed ("Result Transaction ID Attribute").

Note that queries involving deferred results are not currently supported, and may result in an error.

### Example

The query for well locations described by the queries "6 25S 21E 1 SE SE SE NW" and "6 10S 16W 2 C SE NE SW" is formed as follows:

1. The input feature contains the attribute request{0} and request{1} with the values:  
request{0} = "6 25S 21E 1 SE SE SE NW"  
request{1} = "6 10S 16W 2 C SE NE SW"
2. "Request Description Attribute" is set to "request{"
3. "Result Description Attribute" is set to "\_description"
4. "Transaction ID attribute" is set to "\_trans\_id"
5. "Resulting Geometry Type" is set to "Point".
6. Web service host, username, and password are entered for the target server.

Execution of this query results in the following two features:

```
=====
IFMEPoint (-95.052404999999993,37.900077000000003)
_description -> ` 6 25S 21E 1 SESESENW'
_trans_id -> ` 871'
fme_geometry -> ` fme_point'
fme_type -> ` fme_point'
=====
IFMEPoint (-99.076954999999998,39.209764999999997)
_description -> ` 6 10S 16W 2 CSENESEW'
_trans_id -> ` 871'
fme_geometry -> ` fme_point'
fme_type -> ` fme_point'
=====
```

## Transformer Category

Web Services

## **Technical History**

Associated FME function or factory: @Http

## WorkspaceRunner

Runs another FME Workbench workspace on the local computer by spawning a new FME process. This transformer is useful for batch processing, especially in conjunction with the Directory and File Reader.

### Input Port

This transformer runs the specified workspace for each feature that enters through the INPUT port. Any published parameters of the specified workspace will be given values as specified in the transformer, or taken from attributes of the feature which enters it.

### Output Ports

- **SUCCEEDED:** If the Wait for Job to Complete parameter is set to Yes, then the initiating feature is output through this port if the job successfully completed. If the Wait for Job to Complete parameter is set to No, the initiating feature is output through this port if the request was successfully submitted.
- **FAILED:** Only if the FME could not be spawned off will the feature be output via the FAILED port, and the `_failure_message` attribute will hold the reason for the failure.

### Parameters

FME Workspace

Browse to select the additional workspace to run.

Wait for Job to Complete

If this parameter is set to Yes, then the transformer will wait until the workspace has finished running. In this case, the initiating feature is output via the SUCCEEDED port if the job successfully ran to completion.

The initiating feature will be output via the FAILED port if the workspace did not run to completion, and will have a `_failure_message` attribute added to it that contains the error message returned from the FME that ran the workspace.

If this parameter is set to No, the transformer will output the initiating feature as soon as an FME has been spawned off to do the translation. In this case, the initiating feature is output via the SUCCEEDED port if the request was successfully submitted.

### Usage Notes

*Publishing to FME Server:* Publishing a workspace that includes this transformer is not recommended. The transformer will try to start an FME outside of FME Server to run the workspace, and this FME will require an additional license.

### Transformer Category

Workflow

### Related Transformers

There is also an FMEServerWorkspaceRunner that will submit jobs to be run on an FME Server.

### fmepedia

See fmepedia for additional information about this transformer.

### Technical History

Associated FME function or factory: @Tcl2

## XMLFeatureMapper

Constructs features from XML documents via xfMaps.

The XMLFeatureMapper uses a set of rules to map XML data into FME features. These mapping rules are defined in xfMap documents. See the XML reader documentation for more information regarding the xfMap.

### Output Ports

- **MAPPED:** These are the features that are extracted from the XML source document via the specified xfMap.
- **INVALID:** Features with and invalid input document or xfMap document are output at this port.

### Parameters: Configuration

Configuration Type

**Single xfMap file/Embedded xfMap script/Attribute with xfMap script:** The xfMap document may be specified as a single xfMap file, embedded within the transformer, or as the value of a feature attribute.

**Multiple xfMaps files:** Multiple xfMap files may be specified to map the same source XML document.

**XRS file:** An XRS (XML Reader Switch) File allows the XMLFeatureMapper to automatically configure itself to read "known" XML datasets without the need to specify in advance the appropriate xfMap(s). If no xfMap and no XRS are specified for this transformer, then the default XRS document is used. The default XRS document is named xrs.xml, and it is located in the xml/xrs subdirectory of the FME installation directory. For more information regarding the XRS, see the XML Reader documentation in *Workbench Help > FME Readers and Writers Reference*.

### Parameters: Optional

XML Source Name

Specifies the name for the attribute whose default value is the file path of the XML document or an empty string if the XML document was specified wholly as an attribute value; the XML Source Value parameter can be used to override these default values.

Feature Count Attribute

Sets the name for the attribute that enumerates the features mapped per XML document.

Feature Type Attribute

Sets the name for the attribute to store the feature type for the mapped features.

### Transformer Category

XML

### FME Licensing Level

FME Professional edition and above

### Technical History

Associated FME function or factory: XFMapFactory

## XMLFormatter

Provides various options for formatting and cleaning up XML documents.

### Input Ports

- INPUT: Input features that contain the information of the XML documents.

### Output Ports

- PASSED: If a feature is successfully formatted, it will be output through this port.
- INVALID: If a feature does not have a well-formed XML document, it will be output through this port.

### Parameters

XML Input

Select from the pull-down list to enable the selection's corresponding parameter:

- XML Text: Click to open an editor.
- Attribute with XML Text: Choose the attribute that contains XML Text.
- XML Filename: Browse to the XML file.

### Formatting Options

Formatting Type

Select the desired formatting of the XML input:

- None: No formatting is performed
- Pretty-Print XML: XML elements will be formatted by adding indentations and new lines for improved readability. Any white spaces between the start tag and end tag will be preserved.
- Linearize: All XML contents will be put on a single line.

Collapse Empty Elements

When set to Yes, this parameter creates an empty tag for elements that have no content between the Start and End Tag.

For example,

```
<example property="empty"> <example>
```

will be collapsed into

```
<example property="empty"/>
```

### XML Clean-up

Clean up Redundant Namespace Declarations

Remove redundant and extraneous namespace declarations. For example, the following XML document that has redundant namespace declarations:

```
<root>
<f:element1 xmlns:f="http://www.w3schools.com/example">
<f:element2 xmlns:f=" http://www.w3schools.com/example"> some text </f:element2>
<f:element3 xmlns:f=" http://www.w3schools.com/example"> some text </f:element3>
</f:element1>
```

```
</root>
```

Selecting Yes for this parameter will return the following results:

```
<root>  
<f:element1 xmlns:f="http://www.w3schools.com/example">  
<f:element2> some text </f:element2>  
<f:element3> some text </f:element3>  
</f:element1>  
</root>
```

Remove Embedded xsi:schemalocation

When set to Yes, this parameter removes all embedded xsi:schemalocation attribute from all elements that are not the root element.

### XML Output

Attribute to contain XML output/XML Output File

The XML features that have been successfully processed can be output to a feature attribute by specifying an attribute name in the Attribute to contain XML Output, or to a file by specifying the path to the file in XML Output File parameter.

Error and Warning List Name

Features with at least one warning or error will be output through the FAILED port with a new list attribute added to the features.

If the default `_xml_error` is the list name, the elements of the list attribute contain the following:

Elements of List Attribute	Description
<code>_xml_error{}.type</code>	WARNING, ERROR or FATAL ERROR
<code>_xml_error{}.file</code>	the file where the warning or error occurs
<code>_xml_error{}.line</code>	the line where the warning or error occurs
<code>_xml_error{}.col</code>	the column where the warning or error occurs
<code>_xml_error{}.desc</code>	the details about the warning or error

### Transformer Category

XML

### Technical History

FME Factory Used: XMLFormatterFactory

## XMLFragmenter

Maps elements from an XML document into XML fragments.

This transformer may be used to decompose large XML documents into parts, where these parts may be further operated on via downstream XML, XQuery, XSLT or generic text processing transformers.

### Output Ports

- **FRAGMENTS:** Each fragment is output as a separate FME feature via the FRAGMENTS port. Each feature from the port will have an *xml\_fragment* attribute holding the fragment. The fragment is a valid XML document that may be further processed via subsequent XML-based and/or XQuery-based transformers.

Two additional attributes are added to the FRAGMENTS features:

- *xml\_matched\_element* – records the element that was matched. This attribute can be used to identify which element matched the expression, if the last component of the matched expression is a wildcard character (\*).
- *xml\_id* – holds an ID for that element. This attribute is not guaranteed to be globally unique, but it will be unique only in the context of the input document.

### Parameters: XML Source

XML Source Type: XML File/Attribute with XML Document

The XML source type is either an XML file or a feature attribute whose value is the entire XML document.

### Parameters: Feature Paths

Feature Paths Type: Embedded expressions/Attribute with expressions

This parameter specifies which fragments to map. The Feature Paths are whitespace-separated xfmMap match expressions. For more information, see the *FME Readers/Writers* manual: *XML (Extensible Markup Language) Reader/Writer > xfmMap*.

Embedded expressions : Type directly in the text box or click the browse button to display the editor.

Attribute with expressions: Choose a feature attribute.

### Example

```
<dc:metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:subject>Utah</dc:subject>
  <dc:subject>boundaries</dc:subject>
  <dc:subject>County</dc:subject>
  <dc:subject>Administrative</dc:subject>
  <dc:subject>geoscientificInformation</dc:subject>
  <dc:description>This data set represents county boundaries in Utah at 1:24,000 scale.</dc:description>
  <dc:date>2004-04-20T00:00:00.000</dc:date>
  <dc:type>dataset</dc:type>
  <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">{42AE2814-FCC1-4BC2-BAF4-CA3E55514-997}</dc:identifier>
  <dc:language>en</dc:language>
  <dc:spatial>
    <dc:Box name="Geographic" projection="EPSG:4326" xmlns:dcm-
    iBox="http://dublincore.org/documents/2000/07/11/dcmi-box/">
      <dc:Box:northlimit units="decimal degrees">42.01</dc:Box:northlimit>
      <dc:Box:eastlimit units="decimal degrees">-109.21</dc:Box:eastlimit>
      <dc:Box:southlimit units="decimal degrees">36.98</dc:Box:southlimit>
      <dc:Box:westlimit units="decimal degrees">-114.1</dc:Box:westlimit>
    </dc:Box:Box>
  </dc:spatial>
  <dc:rights></dc:rights>
</dc:metadata>
```

These are a few Feature Paths xfmMap expressions targeting the above <dc:metadata> input document:



* "dc:subject"	Extracts every <dc:subject> into an XML fragment.
* "dc:spatial/dcmiBox:Box"	Extracts the <dcmiBox:Box> fragment, but <dc:spatial> must be the parent.
* "dcmiBox:Box/*"	Extracts every child of <dcmiBox:Box> into fragments, 4 fragments feature corresponding to <dcmiBox:northlimit>, <dcmiBox:eastlimit>, <dcmiBox:southlimit> and <dcmiBox:westlimit> are output.
* "dc:subject dc:spatial/dcmiBox:Box dcmiBox:Box/*"	The three previous matched expressions combined, each separated by whitespace.

**Parameters: Flatten Options**

**Flatten Fragment Into Attributes**

Setting this parameter to Yes will enable the Flatten Options field that allows the children of the matched elements to become attributes/attribute lists on the features produced.

There is a brief documentation on "Flatten Options" XML Text Edit box with default values to the options available. Users can change each option accordingly. For more information, see the *FME Readers/Writers* manual: *XML (Extensible Markup Language) Reader/Writer > xfMap > Feature Mapping Rules > Structure Element*.

**Example**

Given the same XML input as above, and Feature Paths xfMap expression is set to "dcmiBox:Box" with the default options in "Flatten Options" will produce the following feature:

```

+++++
Feature Type: `XMLFragmenter_FRAGMENTS'
Attribute(string) : `_xml_source_' has value `C:\Users\rc\Desktop\PRS\28758 -
xmlfragmenter\11291\SourceDataset\degree_csw_get_record_anytext_response.xml'
Attribute(encoded: utf-16): `eastlimit' has value `-109.21'
Attribute(encoded: utf-16): `eastlimit.units' has value `decimal degrees'
Attribute(string) : `fme_type' has value `fme_no_geom'
Attribute(encoded: utf-16): `northlimit' has value `42.01'
Attribute(encoded: utf-16): `northlimit.units' has value `decimal degrees'
Attribute(encoded: utf-16): `southlimit' has value `36.98'
Attribute(encoded: utf-16): `southlimit.units' has value `decimal degrees'
Attribute(encoded: utf-16): `westlimit' has value `-114.1'
Attribute(encoded: utf-16): `westlimit.units' has value `decimal degrees'
Attribute(encoded: utf-16): `xml_fragment' has value `?<?xml version="1.0" encoding="UTF-
16"?><dcmiBox:Box name="Geographic" projection="EPSG:4326"
xmlns:dcmiBox="http://dublincore.org/documents/2000/07/11/dcmi-box/">
<dcmiBox:northlimit units="decimal degrees">42.01</dcmiBox:northlimit>
<dcmiBox:eastlimit units="decimal degrees">-109.21</dcmiBox:eastlimit>
<dcmiBox:southlimit units="decimal degrees">36.98</dcmiBox:southlimit>
<dcmiBox:westlimit units="decimal degrees">-114.1</dcmiBox:westlimit>
</dcmiBox:Box>'
Attribute(encoded: utf-16): `xml_id' has value `id-Box-1.2.1.11.1'
Attribute(encoded: utf-16): `xml_matched_element' has value `Box'
Attribute(string) : `xml_type' has value `xml_no_geom'
Geometry Type: Unknown (0)
=====

```

**Transformer Category**

XML

**Technical History**

FME Factory Used: XFMapFactory

## XMLNamespaceDeclarer

Declares missing namespaces in XML documents by matching prefixes from another sample XML file whose namespaces are fully declared.

### Input Ports

- INPUT: Input features that contain the information of the XML documents.

### Output Ports

- PASSED: If the XML namespaces of the input document are fixed and well-formed, it will be output through this port.
- INVALID: If a feature does not have a well-formed XML document, it will be output through this port.

### Parameters

#### XML Input

XML Input

Select from the pull-down list to enable the selection's corresponding parameter:

- Attribute Specifying XML Text/Attribute With XML Text: Once you connect the transformer, choose the attribute that contains XML text.
- XML Filename/XML File: Browse to an XML file.
- XML Text/XML Text: Opens an XML text editor.

#### Sample XML File

The sample XML file that will be used as a reference for declaring missing namespaces in the XML input documents. If the sample file has more than one namespace with the same prefix but different URI, then the first one will always be used.

### Example

```
<root>
<f:element1>
<f:element2> some text </f:element2>
<f:element3> some text </f:element3>
</f:element1>
</root>
```

and the sample XML file:

```
<root>
<f:element1 xmlns:f="http://www.w3schools.com/example">
<f:element2 xmlns:f=" http://www.w3schools.com/example"> some text </f:element2>
<f:element3 xmlns:f=" http://www.w3schools.com/example"> some text </f:element3>
</f:element1>
</root>
```

the output will look like this:

```
<root>
<f:element1 xmlns:f="http://www.w3schools.com/example">
<f:element2> some text </f:element2>
<f:element3> some text </f:element3>
</f:element1>
</root>
```

#### XML Output

XML Output Type

Attribute to contain XML output/XML Output File: The XML features that have been successfully processed can be output to a feature attribute

by specifying an attribute name in the Attribute to contain XML Output, or to a file by specifying the path to the file in XML Output File parameter.

Error and Warning List Name: Features with at least one warning or error will be output through the FAILED port with a new list attribute added to the features. If the default `_xml_error` is the list name, the elements of the list attribute contain the following:

Elements of List Attribute	Description
<code>_xml_error{ }.type</code>	WARNING, ERROR or FATAL ERROR
<code>_xml_error{ }.file</code>	the file where the warning or error occurs
<code>_xml_error{ }.line</code>	the line where the warning or error occurs
<code>_xml_error{ }.col</code>	the column where the warning or error occurs
<code>_xml_error{ }.desc</code>	the details about the warning or error

### Transformer Category

XML

### Technical History

FME Factory Used: XMLFormatterFactory

## XMLTemplater

Populates an XML template with feature attribute values.

This transformer is a simpler interface to the XQueryFactory than the other XQuery transformers. The XML template can be entered directly, using the Template Expression parameter, loaded from a file using the Template File parameter, or loaded from an attribute using the Template Attribute parameter.

An XML template is actually an XQuery expression, and thus any XQuery functions may be used. In particular the `fme:get-attribute`, `fme:get-xml-attribute`, `fme:get-xml-list-attribute` and `fme:get-xml-list-attribute` functions can be used to retrieve attribute values from a feature. For example, the following template populates an XML fragment with the value of the id attribute.

```
<road>
  <id>{fme:get-attribute("id")}</id>
</road>
```

Note that when using XQuery in this fashion, all function calls have to be enclosed in braces `{}`. To include braces in the output, they must be doubled. For example, the template `'{{0}}'` produces `'{0}'` as its result.

Note that to enclose the value of an XQuery function inside braces, the braces have to be tripled, as in this example: `'{{{fme:get-attribute("id")}}}'`.

The following template is a more complex example. It populates a fragment of CSW metadata XML with FME feature attribute values.

```
<ogc:PropertyIsLike xmlns:ogc="http://www.opengis.net/ogc"
  wildcard="{fme:get-attribute("_wildcard")}"
  singleChar="{fme:get-attribute("_singlechar")}"
  escape="{fme:get-attribute("_escapechar")}">

  <ogc:PropertyName>{fme:get-attribute("_property_name")}</ogc:PropertyName>
  <ogc:Literal>{fme:get-attribute("_literal")}</ogc:Literal>
</ogc:PropertyIsLike>
```

When specifying an XML template through the Template Expression parameter or the Template File parameter, the transformer will verify that all referenced feature attributes are present in an incoming feature. If attributes are missing (not exposed) from input features, the transformer will be highlighted red as incomplete. When this situation occurs, the transformer's Summary Annotation will indicate the missing attributes the XML template is referencing.

This additional validation behavior can be overridden by setting the parameter Validate Attribute Names to No.

## XQuery Functions

FME provides several functions that you can use within XQuery scripts. These functions allow XQuery scripts to access and manipulate feature attribute values. Currently there are no functions that allow the manipulation of feature geometry.

See XQuery in FME.

## Transformer Category

XML

## Technical History

FME Factory Used: XQueryFactory

## XMLValidator

Validates the syntax or schema of an XML file or text. There are different ways to specify the XML source to be validated:

- entering the XML text in the XML Text field,
- specifying the attribute that contains the XML text in the *Attribute with XML Text* parameter,
- specifying the attribute that contains the path to XML File, or
- picking the XML file in the XML Filename parameter.

### Parameters

#### XML Input

The choice that you make in this field enables its corresponding parameter.

- XML Text: Click to open an editor.
- Attribute with XML Text: Choose the attribute that contains XML Text.
- XML Filename: Browse to the XML file or choose the attribute that contains path to XML file.

#### Validation Type

If this parameter is set to None, all features will be output through the PASSED port.

If the Validation Type is set to Syntax Only or Syntax and Schema, all features that pass the syntax only or syntax and schema validation will be output through the PASSED port.

#### Error and Warning List Name

Features with at least one warning or error will be output through the FAILED port with a new list attribute added to the features. If the default *\_xml\_error* is the list name, the elements of the list attribute contain the following:

Elements of List Attribute	Description
<i>_xml_error</i> {}.type	WARNING, ERROR or FATAL ERROR
<i>_xml_error</i> {}.file	the file where the warning or error occurs
<i>_xml_error</i> {}.line	the line where the warning or error occurs
<i>_xml_error</i> {}.col	the column where the warning or error occurs
<i>_xml_error</i> {}.desc	the details about the warning or error

#### Schema Location

Optionally, users can specify the schema location if the 'Validation Type' is set to 'Syntax and Schema'. It can be specified either as an attribute that contains the schema location or selected using the file browser. However, it is not guaranteed that the schema specified is used in the validation if the schema referenced by the XML file can be located in the default XML Schema location in FME.

### Transformer Category

XML

### Technical History

FME Factory Used: XMLValidationFactory

## XQueryExploder

Uses XQuery expression to extract portions of XML text into new FME features.

### Parameters

#### XQuery Type

XQuery Input

This parameter identifies the type of the XQuery to be executed. The possible values each correspond to ways of specifying the XQuery. It can be

- directly specified – XQuery expression,
- an attribute on a feature – Attribute specifying an XQuery, or
- a path to a file – XQuery file.

#### XML Source

XML Input, XML Attribute, XML File

This parameter identifies either an attribute that contains an XML document (*XML Attribute*), or specifies a file that contains an XML document (*XML File*). You can set this parameter to *None (file is specified in query)* if the XQuery parameter above refers to an XML file. If this parameter is set, the context document for the query will be set to the value of the parameter (either as a file or a string, as appropriate).

Remove Source XML Attribute?

If the XML document is loaded from an attribute, the Remove Source XML Attribute parameter can be set to remove the XML document after the query has been processed.

#### Results

Write XML Header?

The Write XML Header parameter specifies whether or not the XML header should be written into the results of the XQuery. Note that for UNICODE files, the Byte Order Mark (BOM) is not written, and should be added by an additional process if desired.

Result Attribute

This parameter determines the attribute to which the XQuery results will be written. It is used only when the XML document is an attribute.

For every element in the result set, a feature will be output the QUERY\_RESULTS port with the value of the result set to the attribute specified by this parameter.

Query Tag Attribute Name

If this attribute is set, the text of the query which produced each result will be written to the attribute specified. The default is *\_source\_query*.

### XQuery Functions

FME provides several functions that can be used within XQuery scripts. These functions allow XQuery scripts to access and manipulate feature attribute values. Currently, there are no functions that allow the manipulation of feature geometry.

See XQuery in FME.

### Examples

XQuery Examples

### Transformer Category

XML

### FME Licensing Level

FME Professional edition and above

## **Technical History**

Associated FME function or factory: XQueryFactory

## XQueryExtractor

Uses XQuery expressions to extract portions of XML text into feature attributes.

### Parameters

#### XQuery Type

XQuery Input

This parameter identifies the type of the XQuery to be executed. The possible values each correspond to ways of specifying the XQuery. It can be

- directly specified – XQuery expression
- an attribute on a feature – Attribute specifying an XQuery
- a path to a file – XQuery file

#### XML Source

XML Input

This parameter identifies either an attribute that contains an XML document (*XML Attribute*), or specifies a file that contains an XML document (*XML File*). You can set this parameter to *None (file is specified in query)* if the XQuery parameter above refers to an XML file. If this parameter is set, the context document for the query will be set to the value of the parameter (either as a file or a string, as appropriate).

Remove Source XML Attribute?

If the XML document is loaded from an attribute, the Remove Source XML Attribute parameter can be set to remove the XML document after the query has been processed.

#### Results

Write XML Header?

The Write XML Header parameter specifies whether or not the XML header should be written into the results of the XQuery. Note that for UNICODE files, the Byte Order Mark (BOM) is not written, and should be added by an additional process if desired.

Return Value, Separator Character(s), Result Attribute

The **Result Attribute** parameter determines which attribute the XQuery results will be written to.

If the **Return Value** is set to *Separated Values*, the results will be written out as a delimited string, with the separator character determined by the value set for **Separator Character(s)**. If the **Return Value** is set to *Single value*, the results will be concatenated.

### XQuery Functions

FME provides several functions that can be used within XQuery scripts. These functions allow XQuery scripts to access and manipulate feature attribute values. Currently, there are no functions that allow the manipulation of feature geometry.

See XQuery in FME.

### Examples

XQuery Examples

#### Transformer Category

XML

#### FME Licensing Level

FME Professional edition and above

#### Transformer Category

Surfaces



## **Technical History**

Associated FME function or factory: XQueryFactory

## **XQueryUpdater**

Provides updates to an XML document using XQuery Update expressions.

### **Parameters**

#### ***XQuery Type***

XQuery Input

This parameter identifies the type of the XQuery to be executed. The possible values each correspond to ways of specifying the XQuery. It can be

- directly specified – XQuery expression
- an attribute on a feature – Attribute specifying an XQuery
- a path to a file – XQuery file

#### ***XML Source***

XML Input, XML Attribute, XML File

This parameter identifies either an attribute that contains an XML document (*XML Attribute*), or specifies a file that contains an XML document (*XML File*). You can set this parameter to *None (file is specified in query)* if the XQuery parameter above refers to an XML file. If this parameter is set, the context document for the query will be set to the value of the parameter (either as a file or a string, as appropriate).

### **Results**

Result Attribute

The parameter determines which attribute the XQuery results will be written to. It is used only when the XML document is an attribute.

For every element in the result set, a feature will be output the QUERY\_RESULTS port with the value of the result set to the attribute specified by this parameter.

When the query operates directly on a file, the update will occur on the file itself, and no features will be produced. If this is not the desired result, you can read the file using an AttributeFileReader transformer and pass that feature into the XQueryUpdater transformer.

### **XQuery Functions**

FME provides several functions that can be used within XQuery scripts. These functions allow XQuery scripts to access and manipulate feature attribute values. Currently, there are no functions that allow the manipulation of feature geometry.

See XQuery in FME.

### **Examples**

XQuery Examples

### **Transformer Category**

XML

### **FME Licensing Level**

FME Professional edition and above

### **Technical History**

Associated FME function or factory: XQueryFactory

## **XSLTProcessor**

The XSLTProcessor uses an XSL (eXtensible Stylesheet Language) stylesheet to convert an XML document.

**Note:** It is assumed that you have created (or have access to) an XSLT stylesheet that is applicable to your XML input source.

### **Output Ports**

- **TRANSFORMED:** Each transformed document is output via this port. The transformed document may be either embedded in an attribute or written to a file.
- **SKIPPED:** These features did not transform due to errors (for example, an incorrect file path).

### **Parameters**

#### ***XML Source***

XML Source Type, XML File, XML Attribute

The XML source type is either an XML file or a feature attribute whose value is the entire XML document.

#### ***Configuration***

Configuration Type

The stylesheet document is either an XML file, embedded in the transformer, or in a feature attribute whose value is the entire stylesheet document

#### ***Results***

Results Type

The results from applying the stylesheet to the input document can be either stored in an attribute value or written to a file.

### **Transformer Category**

XML

### **FME Licensing Level**

FME Professional edition and above

### **Transformer Category**

Surfaces

### **Technical History**

Associated FME function or factory: XSLTFactory