



FME® Desktop Training Workbook

FME Desktop 2013 Edition



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

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

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

Any spatial data included in this material is intended for training use only. It is not meant to accurately reflect any real life situation and may have been altered in some way from the original source data.

Much of the data used here originates from public domain data made available by the City of Austin and Travis County, Texas. The datasets can be found at:

ftp://ftp.ci.austin.tx.us/GIS-Data/Regional/coa_gis.html

Copyright

© 1994–2013 Safe Software Inc. All rights are reserved.

Revisions

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

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

fax: 604-501-9965
e-mail: services@safe.com

www.safe.com

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

Trademarks

FME® and SpatialDirect® are registered trademarks of Safe Software Inc. All brand or product names are trademarks or registered trademarks of their respective holders and should be noted as such.

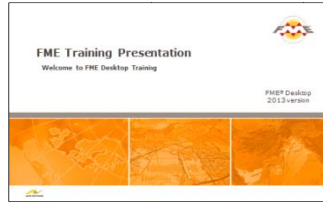
Document Information

Document Name: FME Desktop Training Workbook 2013
Version: FME Desktop 2013
Updated: January 2013

Training Exercises

FME Desktop Training Workbook	3
This Workbook	3
Scenario	3
Session 2 – Data Transformation	5
Objective	5
Detailed Steps	6
Advanced Tasks	13
Session 3 – Translation Components	14
Objective	14
Detailed Steps	14
Advanced Task 1	19
Advanced Task 2	20
Session 4 – Datasets and Feature Types	22
Objective	22
Detailed Steps	22
Advanced Tasks	25
Session 5 – Practical Transformer Use	26
Objective	26
Detailed Steps	26
Advanced Task	30
Session 6 – Best Practice	32
Objective	32
Detailed Steps	32
Advanced Tasks	36
Session 7 – Group Project	37
Task 1: Data Preparation	38
Task 2: Data Extraction	40
Task 3: Schema Definition and Schema Mapping	42
Advanced Tasks	44

FME Desktop Training Workbook



This workbook contains the exercises to be performed in conjunction with the FME Desktop training sessions.

This Workbook

Each exercise in this workbook relates to a different FME Desktop training session.

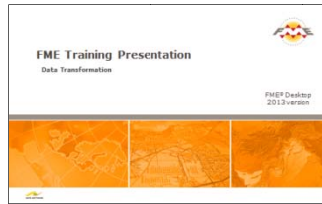
Scenario

All exercises are based on the scenario that you work as a GIS Technician at the fictional city of Interopolis.

As an FME user with the City of Interopolis, you are using Workbench to translate source data and produce output for other departments within your organization.



Session 2 – Data Transformation



This exercise involves using transformers to validate and clean up some spatial data and its associated attributes.

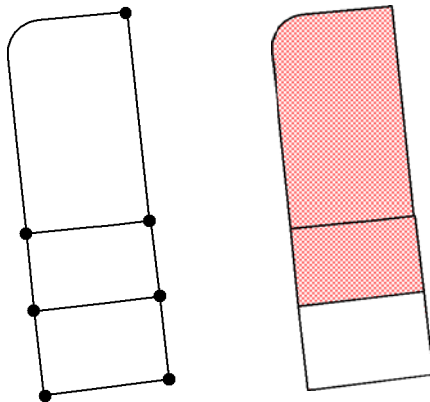
Exercise: Data Transformation	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property lot lines (MapInfo MIF); Google Earth KML
Overall Goal	Convert a set of property lines in MapInfo MIF format, to a set of polygons in Google Earth KML format; simultaneously validating geometry and attributes.
Starting Workspaces	None
Finished Workspaces	C:\FMEData\Workspaces\DesktopWorkbook\Exercise1aComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise1bComplete.fmw

Objective

In this exercise you have been given an urgent task of converting a Planning Department property lot lines dataset, and, at the same time, validating the dataset's geometry and attributes.

The validation tests to be carried out are as follows.

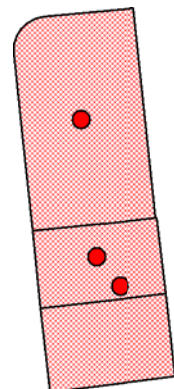
a) Test that each property can be formed into a unique polygon.



Here is a set of data shown as lines and then polygons. The bottom property boundary must have a gap that prevents a polygon being formed. This is the sort of problem we need to validate against.

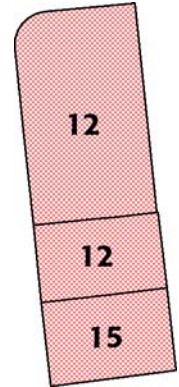
b) Test each property has a single centroid (point feature) inside of it.

The gap is fixed, but there are more problems. The upper polygon is correct, but the middle polygon has two centroids (not one), and the lower polygon has none at all (perhaps it was accidentally placed too far north?)



c) Test that each property in a block has a common Block ID number.

These three properties form a single block, but the lower property has a problem (again!). This time it's a different block ID number.



Detailed Steps

1) Inspect the Source Data

Inspecting the source data with the FME Universal Viewer before a translation should now be automatic.

Start the FME Universal Viewer and use it to open the source dataset. Query some features to become familiar with the geometry and attributes that exist.

The source data here is:

Reader Format	MapInfo MIF/MID
Reader Dataset	C:\FMEData\Data\Properties\parcel_L26.mif

2) Create a New Workspace

Start Workbench and create a workspace to translate the source data (parcel_L26) to:

Writer Format	Google Earth KML
Writer Dataset	C:\FMEData\Output\DesktopTraining\Parcels.kml

Save the workspace.

3) Place a Transformer – GeometryFilter

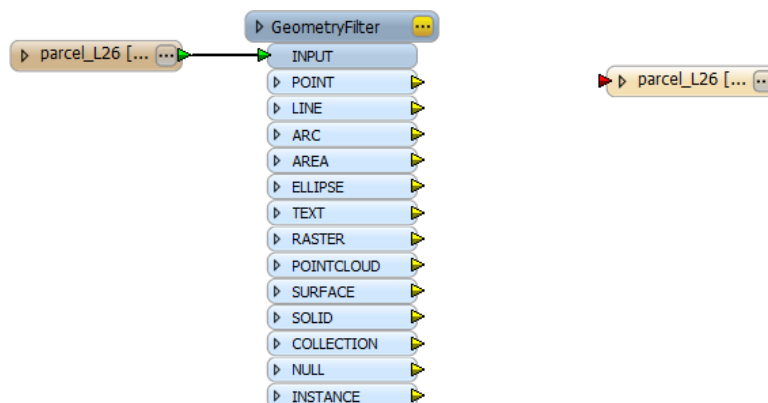
From inspecting the data you will have noticed that both line and point features occupy a single layer (feature type). However, only the line features are needed to build polygons.

In workspace authoring it's often useful to run the workspace after every step, without writing data.

So, first delete the existing connection between reader and writer feature types.

Now add a *GeometryFilter* transformer using Quick Add to place it unconnected onto the canvas.

Then connect the *GeometryFilter* to the reader feature type.



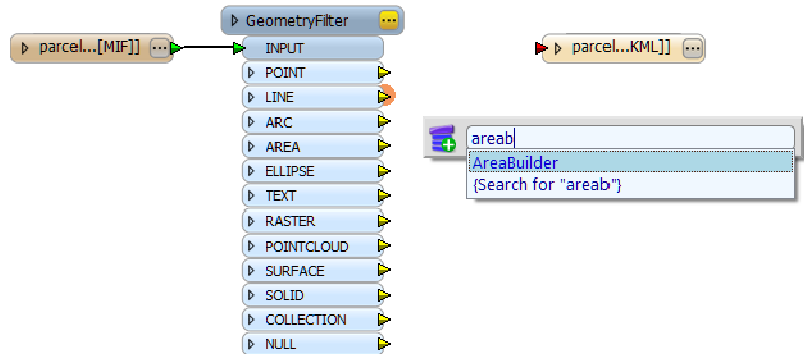
4) Place a Transformer – AreaBuilder

Add an *AreaBuilder* transformer to turn the lot linework into lot polygons.

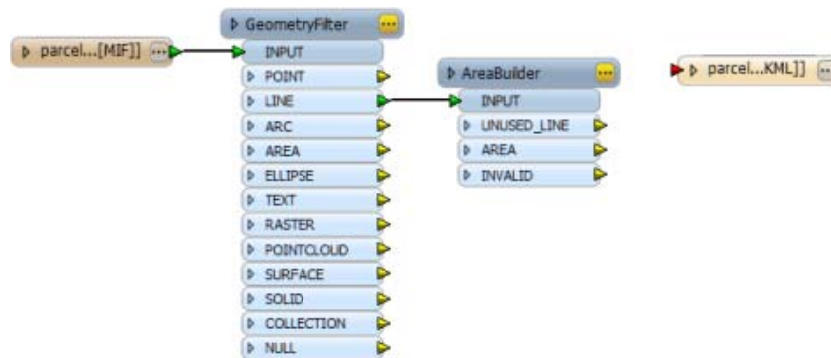
Ultimately the *GeometryFilter* transformer will be used to keep lines, but filter out points, so connect the *AreaBuilder* onto the port *GeometryFilter:LINE*.

The most efficient way to do this is with Quick-Add.

Click the output port marked LINE and start typing 'areabu' on the canvas.



The workspace will now look something like this:



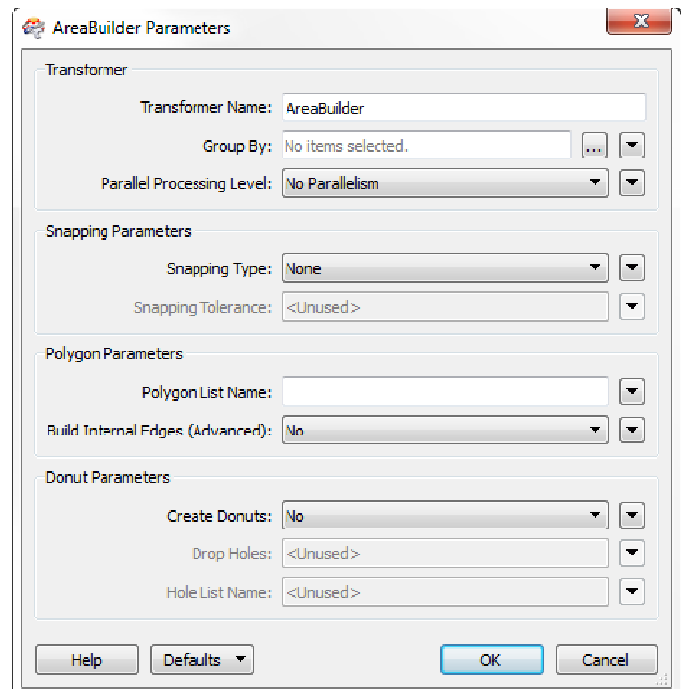
5) Check the Transformer Parameters

The yellow color of the *AreaBuilder* icon indicates some parameters need to be checked.

Open the *AreaBuilder* Parameters dialog by clicking the yellow icon.

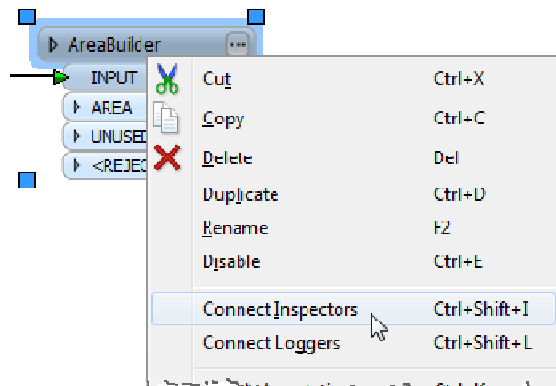
You don't need a Group-By so leave that field empty, and Create Donuts can be set to no.

The yellow icon on the *AreaBuilder* should now turn blue indicating the parameters have been checked and the workspace is ready to run.



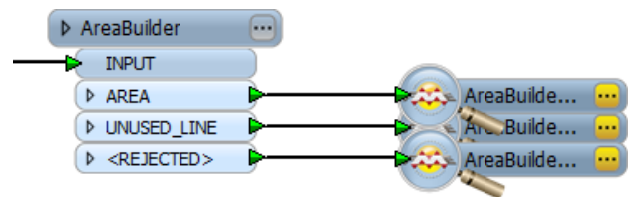
6) Place Transformers – Inspectors

At this point you can try the workspace, but sending the data to FME Universal Viewer to check the output is always a good step to take before writing the output for real.



Right-click the title of the *AreaBuilder* and select the option to **Connect Inspectors**.

Inspectors are connected and named automatically.



NB: The <REJECTED> port is for features from which polygons cannot be built. Because these are being filtered out by the GeometryFilter, there should be no such features in the output.

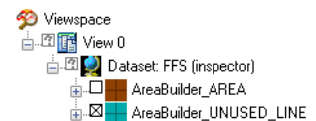
7) Run the Workspace

Click the Run button on the Workbench toolbar or choose **File > Run** from the menu bar.

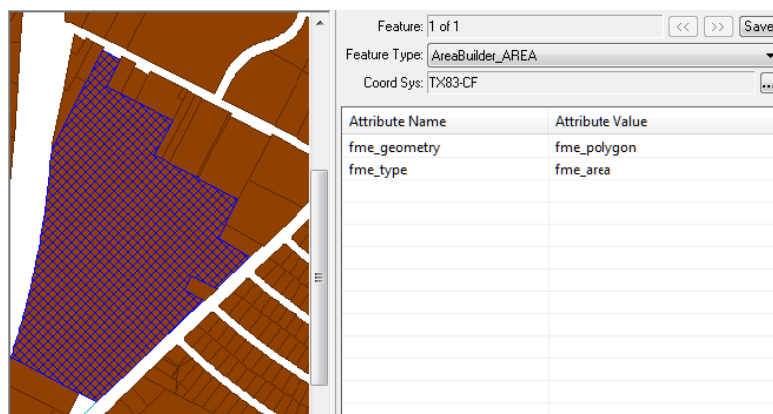
8) Check the Output

Well, polygons have been formed, but there are two things to check: geometry and attributes.

For a geometry check, turn off the polygon layer in FME Universal Viewer and check for any unused lines.



In a real-life situation you might now use **File > Save Data As** to save unused lines in a format of your choice, so that it can be sent to a data editor for fixing.



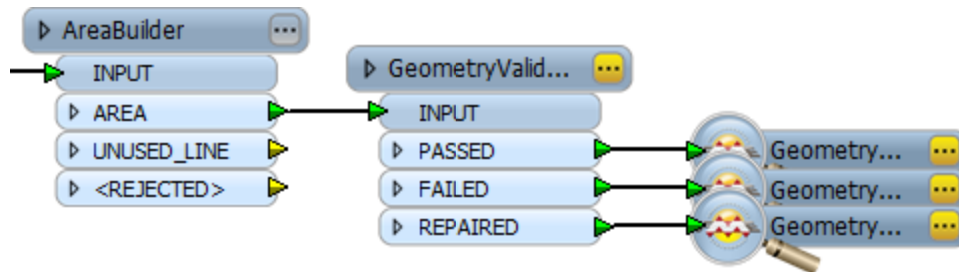
For an attribute check, turn the polygon layer back on and query a feature to see what attributes it has (if any)

9) Validate the Geometry

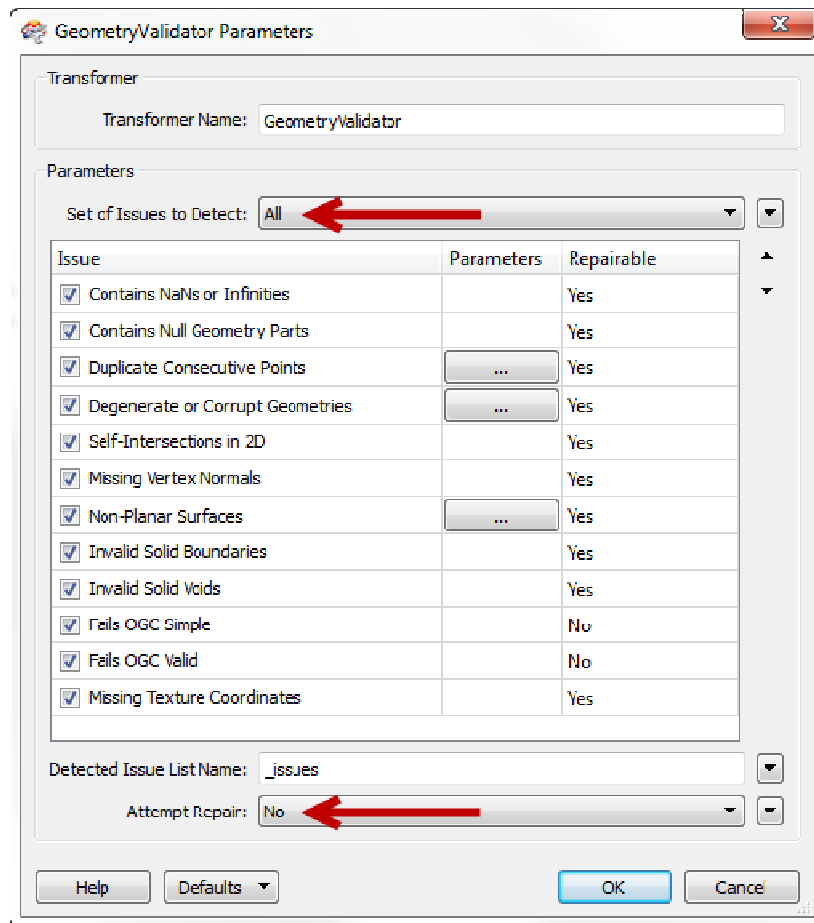
When you think about it, just because some lines have been formed into polygons it doesn't necessarily make them valid. It would be too difficult and time-consuming to check each polygon manually, so let's use a transformer to do it for us.

Locate and place a *GeometryValidator* transformer, and connect it to the AREA output port on the *AreaBuilder* transformer.

Connect the *GeometryValidator* output ports to *Inspectors* to analyze the output.



Open the *GeometryValidator* transformer parameters dialog. Set "Issues to Detect" to All. Ensure that Attempt Repair is set to No (we must validate the data, but it's not our data to repair).



10) Run the Workspace Again

Run the workspace and check for invalid features.

Even if you can't see a fault in the highlighted features, query them and check their attributes; the *GeometryValidator* transformer adds an attribute to indicate the problem(s) and which test(s) was failed .

Again, in the real world the FAILED features would probably be saved for further examination.

Attribute Name	Attribute Value
_issues{0}.issue_found	Self-Intersections in 2D
_issues{0}.location_sample.x	3128986.5
_issues{0}.location_sample.y	10088097.5
_issues{1}.issue_found	Fails OGC Simple
_issues{1}.location_sample.x	3128986.5
_issues{1}.location_sample.y	10088097.5
_issues{2}.issue_found	Fails OGC Valid
_issues{2}.location_sample.x	3128953
_issues{2}.location_sample.y	10088089
fme_geometry	fme_polygon
fme_type	fme_area

11) Place a Transformer – *PointOnAreaOverlayer*

If you looked carefully during the data inspection, then the lack of attributes is no surprise because you'd have noticed it was the polygon centroids (point features) that held the attributes, not the line features.

So the next task is to get the attributes off the point features and onto the new polygons.

The *PointOnAreaOverlayer* transformer transfers attributes from points onto surrounding areas.

Use whatever method you prefer to locate the *PointOnAreaOverlayer* transformer and place one into an empty part of the workspace.

12) Correct Connections

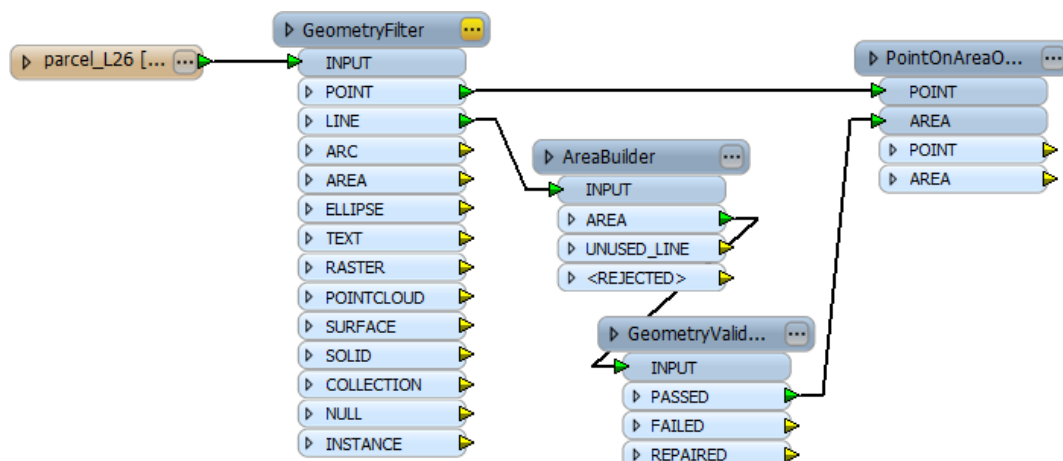
This is where you start to use transformers in parallel.

Connect *GeometryFilter:POINT* to the input port *PointOnAreaOverlayer:POINT*

Connect *GeometryValidator:PASSED* to the input port *PointOnAreaOverlayer:AREA*

Check the *PointOnAreaOverlayer* parameters (see the yellow button) and accept the defaults.

At this point your workspace will look like this:

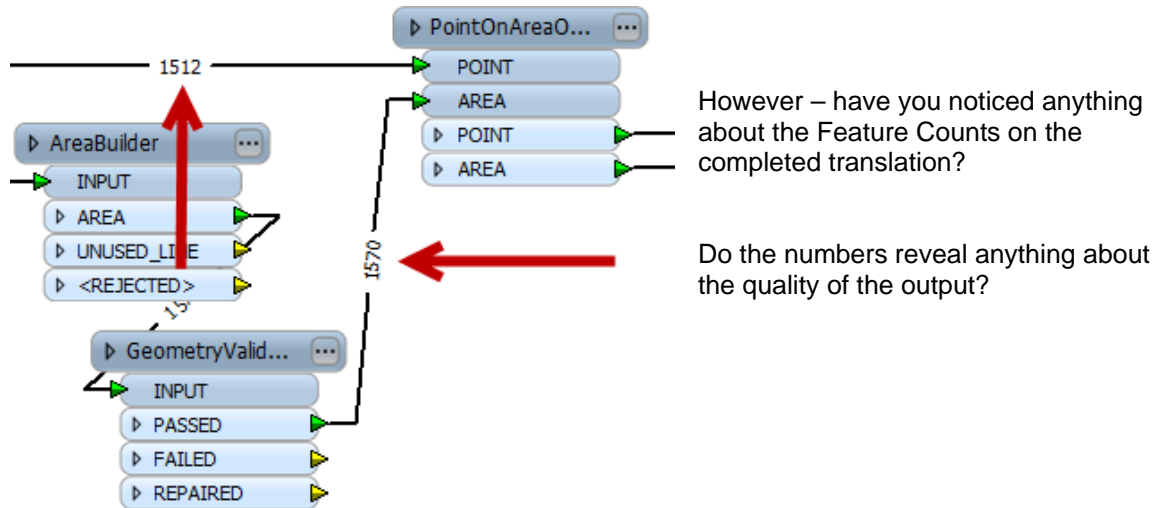


13) Run the Workspace

Connect up some Inspectors and run the workspace again.

14) Check the Output

Query a feature. That's much better. Polygons were formed with the correct attributes.



15) Place a Transformer – Tester

Well done if you noticed the problem: 1570 area features but only 1512 point features.

This means a number of area features are without a centroid, and therefore without attributes. Potentially it's worse – what if areas are overlapping and sharing the same centroid?

Let's test for polygons without a one-to-one match.

Place a *Tester* transformer on the canvas. Connect it to the *PointOnAreaOverlay:AREA* port.

See if you can figure out how to set up the *Tester* before looking at the next step!

16) Check the Transformer Parameters

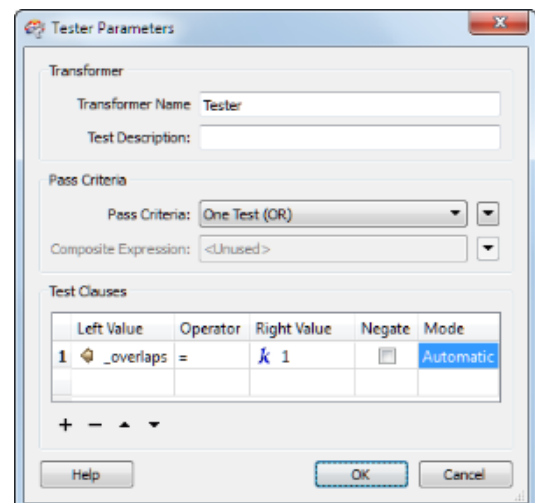
Did you figure it out?

Importantly the *PointOnAreaOverlay* adds a new attribute called *_overlaps*.

This tells you how many points were found inside each area.

Set the *Tester* properties with the required test:

_overlaps = 1



17) Identify Bad Output

Bad features (those where the number of overlaps is zero, or more than one) will emerge from the *Tester:FAILED* port. Let's run the workspace to identify bad output quickly.

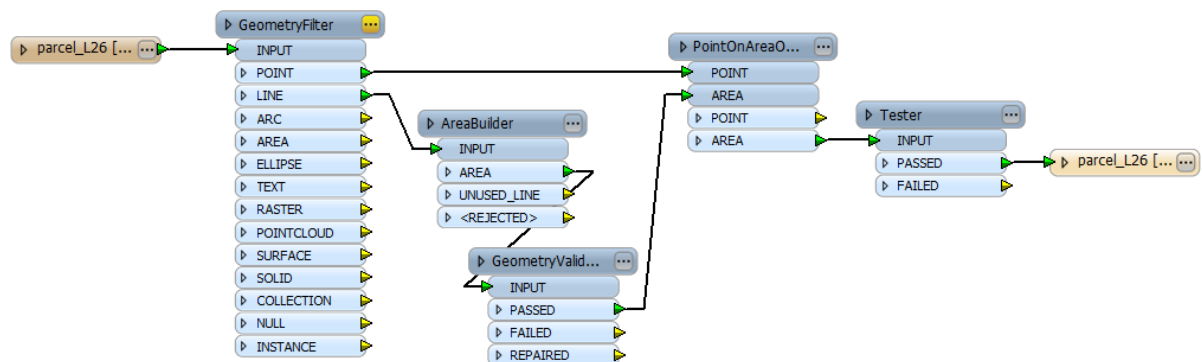
Delete any existing Inspectors then add two more; one to the port *Tester:FAILED* and another to *PointOnAreaOverlayer:POINT* (inspecting the points is necessary to see why the areas failed).

Run the workspace and examine the bad data. How many bad features are there?

18) Make Final Connections

Now we've filtered out all of the bad data, we can make a final connection to the output dataset. Connect the output port *Tester:PASSED* to the destination feature type *OGCKML:parcel_k26*

Delete any existing Inspectors.



19) Save and Run the Workspace

Save the workspace using the **Save** button on the toolbar. Run the workspace again.

20) Check the Output

Inspect the output data to prove that it only contains polygon features with attributes.

Open it in Google Earth to make sure it is in the correct location.





Advanced Tasks

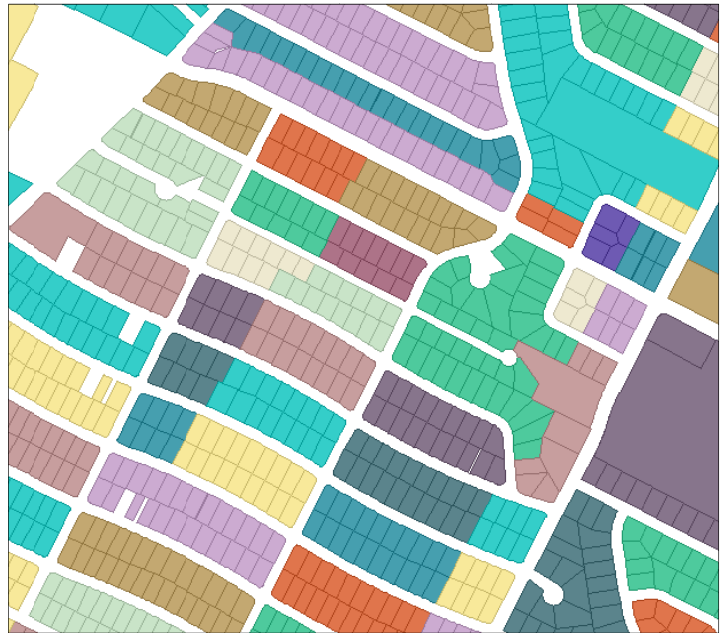
Ooops! We haven't done the final test – that each block of properties has a unique ID number

As an advanced task, devise a way to inspect the data in the FME Universal Viewer, in such a way that a user could spot wrongly numbered properties.

The simplest approach is to group like features together in some way (probably using an *Aggregator* transformer and the BLOCK_ID attribute) then applying a different color per feature.

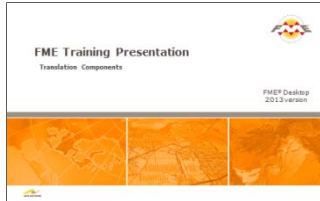
So, does every property in every block have the same ID?

Similarly, does every block have a unique ID?





Session 3 – Translation Components



Here's an exercise with tasks that gently stretch your use of formats within FME.

Exercise: Translation Components	
Scenario	FME user; City of Interopolis, Planning Department
Data	Landmarks (AutoCAD DWG), Elevation Model (CDED Raster)
Overall Goal	Compare landmarks against an elevation model to find prominent positions for potential cell phone (mobile phone) masts.
Starting Workspaces	None
Finished Workspaces	C:\FMEData\Workspaces\DesktopWorkbook\Exercise2aComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise2bComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise2cComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise2dComplete.fmw

Objective

A communications company is negotiating with the city to find locations to position cell phone (mobile phone) masts. The ideal positions are on publically owned, prominent landmarks at as high an elevation as possible.

You have been tasked with creating an FME workspace to pick out likely sites for further analysis.

In this exercise, you'll fulfill the following objectives:

- Read a source dataset of local landmarks.
- Filter out unsuitable sites such as schools.
- Overlay the data onto a raster DEM to assign elevation to each landmark.
- Select the top ten sites in terms of elevation.

Detailed Steps

1) Inspect the Source Data

Inspecting the source data with the FME Universal Viewer before a translation should now be automatic.

Start the FME Universal Viewer and use it to open the source datasets. Query some features to become familiar with the geometry and attributes that exist.

The source data here is:

Reader Format Autodesk AutoCAD DWG/DXF
Reader Dataset C:\FMEData\Data\Landmarks\Landmarks.dwg

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

2) Start Workbench

Start Workbench. Create a workspace to translate the AutoCAD landmarks data; namely:

Reader Format	Autodesk AutoCAD DWG/DXF
Reader Dataset	C:\FMEData\Data\Landmarks\Landmarks.dwg
Reader Parameters	Group Entities By: Attribute Schema
Writer Format	Adobe 3D PDF
Writer Dataset	C:\FMEData\Output\DesktopTraining\Cellmasts.pdf

3) Place a Tester

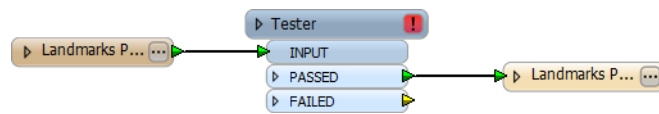
While inspecting the data you should have noticed that there is an attribute called LANAME – this represents the name of the landmark feature.

One task is to filter out landmarks which have:

- No name (i.e. no value for LANAME)
- The word “School” in the name

...as City council has determined that no cell-phone masts should be erected on school property.

Place a *Tester* transformer with the PASSED output port connected to the output. The test will be set up so that the features we want will pass the test.



Set up two tests:

- LANAME Attribute Is Empty
- LANAME Contains School

Then click the Negate option for both of these. That way we will cause these features to FAIL.

Left Value	Operator	Right Value	Negate	Mode
1 LANAME	Attribute Is Empty	<Unused>	<input checked="" type="checkbox"/>	Automatic
2 LANAME	Contains	School	<input checked="" type="checkbox"/>	Automatic

+ - ^ v Duplicate

Be sure to set up the Pass Criteria field as “All Tests (AND)”; i.e. filter where LANAME is not empty AND it doesn’t contain the word “School”.

Pass Criteria:	All Tests (AND)
Composite Expression:	<Unused>



This is a very interesting test of logic.

If the tests were not negated, then the pass criteria should be OR.

But, because the tests are negated, the pass criteria should be AND

Basically, if you negate multiple tests, you’ll want to switch from OR to AND. Can you see why this might be?

4) Run Workspace?

At this point you may wish to disconnect the writer and connect Inspectors to both *Tester* output ports, and run the workspace to make sure the *Tester* is acting as expected.

If all features emerge from the PASSED port, then check the Pass Criteria setting and try again.

5) Add Reader

The next step is to include the elevation model data that is required.

Select **Readers > Add Reader** from the menubar, and add the following reader:

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

6) Place Transformers – Reprojectors

You may have noticed that the data in the two source datasets are stored in different coordinate systems.

The Landmark source DWG file is stored in the coordinate system **LL84**.

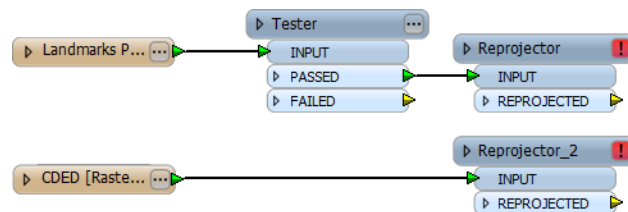
The DEM source CDED data is stored in the coordinate system **LL72-SECONDS**.

If we want to overlay the landmark features onto the raster DEM, we need to convert them to the same coordinate system. The *Reprojector* transformer allows us to do this inside a workspace.

Place two *Reprojector* transformers.

Connect one of them to the CDED Reader's single Feature Type.

Connect the other to the *Tester*:PASSED output port.

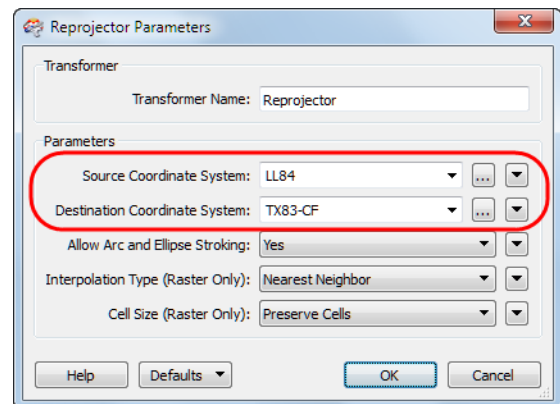


Set up these transformers to reproject the source datasets to a common TX83-CF coordinate system.

The Source and Destination Coordinate System are the key parameters to define.

The *Reprojector* parameters for the Landmark data will look like this:

Q) Could you leave Source Coordinate System as 'Read from Feature'? How would you know?



7) Run Workspace

To ensure the *Reprojector* transformers are working as expected, connect them both to *Inspector* transformers and re-run the workspace.

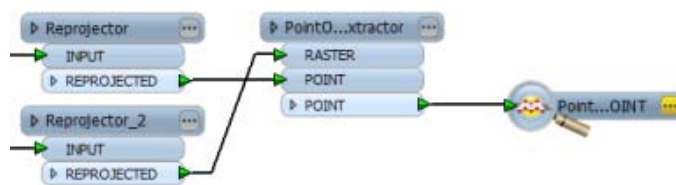
If the data is reprojected correctly, both sets of features appear in the same coordinate space.

8) Place a Transformer – *PointOnRasterValueExtractor*

To assign elevation to landmark data, we need to overlay it onto the DEM and extract a Z value.

Place a *PointOnRasterValueExtractor* transformer. Use its default parameter values.

Connect the landmark REPROJECTED data to the *PointOnRasterValueExtractor:POINT* input port, and the elevation model REPROJECTED to *PointOnRasterValueExtractor:RASTER*.



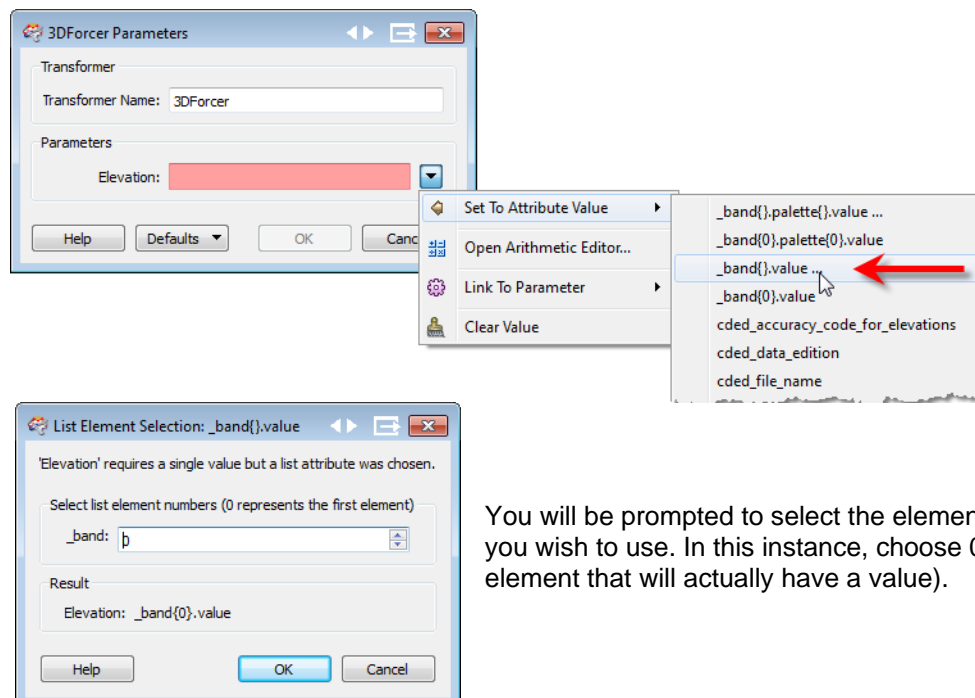
Again, connect an *Inspector* to the *PointOnRasterValueExtractor* and run the workspace to inspect what data the process has created so far. The resulting attribute from this operation is called '*_band{0}.value*'.

The attribute is called '*_band{0}.value*' because it's the first band (layer) of the raster. In fact this layer of DEM values is the only band here. A three-band color raster dataset would produce *_band{0}.value*, *_band{1}.value*, and *_band{2}.value* for the colors Red, Green, and Blue.

9) Place a Transformer – *3DForcer*

The elevation can now be used to provide the output features with a proper Z value.

Place a *3DForcer* transformer. For elevation click on Set To Attribute Value and select *_band{0}.value*:

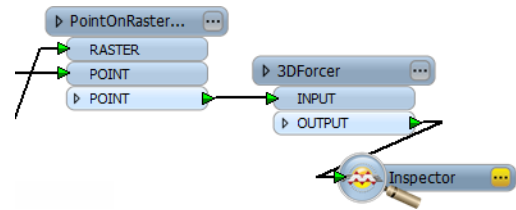


You will be prompted to select the element in the list that you wish to use. In this instance, choose 0 (the only list element that will actually have a value).

Once more, connect an *Inspector* and run the workspace. Querying a feature should show that each landmark feature now has a Z coordinate.

```
Geometry Type: IFMEPoint
Coordinate Dimension: 3
(3124797.95906762,10086757.286974,199)
```

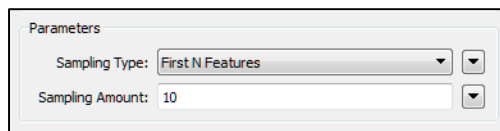
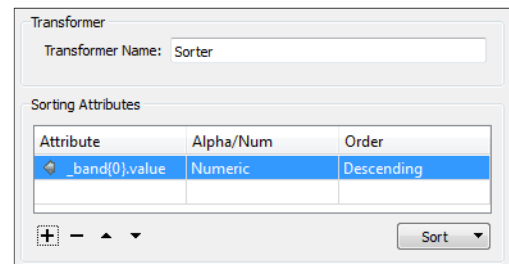
The last part of the workspace will now look something like this:



10) Place Transformers – Sorter and Sampler

Not much work left now: we just need to select the top 10 sites in terms of elevation.

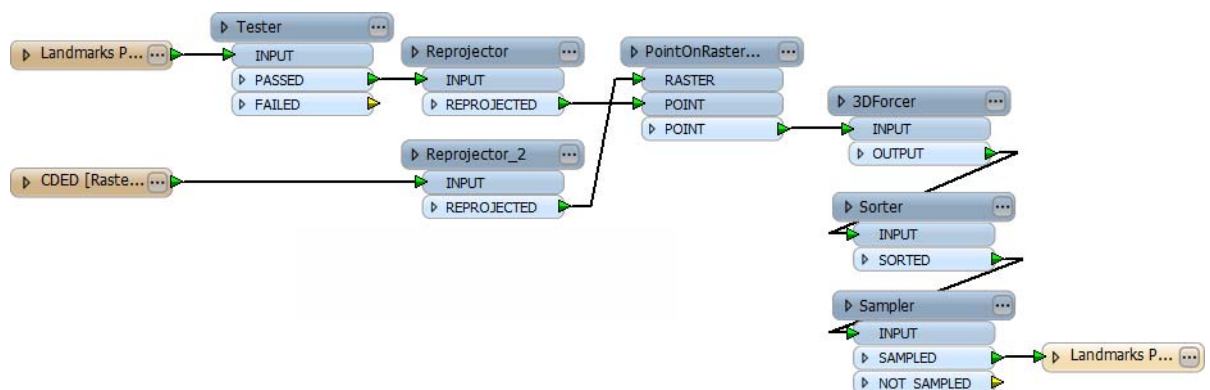
Place a *Sorter* transformer and set it up to sort features in order of elevation – highest to lowest.



Place a *Sampler* transformer and set it up to pass only the first 10 features in the workflow.

11) Reconnect Writer

Reconnect the writer, and delete any remaining *Inspectors*. The workspace should now look something like this:



12) Run Workspace

Save and run the workspace, and inspect the output.



Advanced Task 1

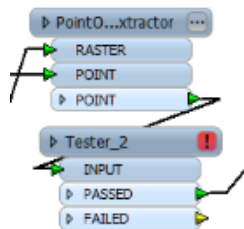
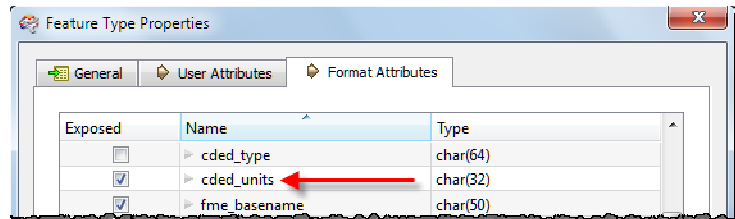
When inspecting the data you may have noticed a CDED format attribute called *cded_units*. This tells us what units the elevation values are in the data. At the moment we might assume the data (being from the US) is in feet, but this might not be the case...

By checking the value of the attribute, and converting elevation from metres to feet if necessary, we can make the workspace impervious if used on source datasets with different units.

1) **Expose Format Attribute**

Check if the format attribute *cded_units* is already exposed.

If not, click the Format Attributes tab on the CDED source properties dialog. Put a checkmark next to the format attribute to expose it.



2) **Place a Transformer – Tester**

Place a *Tester* after the *PointOnRasterValueExtractor* transformer to test if *cded_units* = feet.

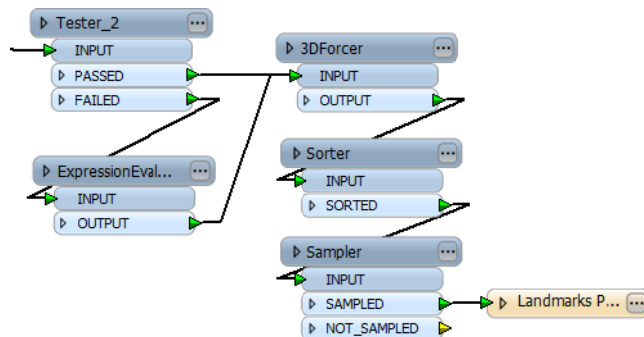
If so (PASSED) the features can proceed to the *3DForcer*.

The test clause will look like this:

Test Clauses				
Left Value	Operator	Right Value	Negate	Mode
1	cded_units	=	feet	<input type="checkbox"/> Automatic

3) **Place a Transformer – ExpressionEvaluator**

If a feature fails the test – its units are metres and not feet – then we need to convert the units.

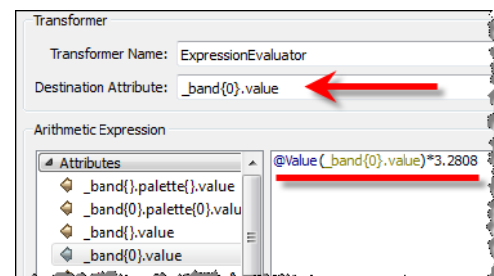


Place an *ExpressionEvaluator*.

Connect it to the *Tester:FAILED* output port, and then to *3DForcer:INPUT*.

Set up the transformer to multiply the input attribute (*_band{0}.value*) by the number of feet per metre (3.2808) and write to *_band{0}.value*.

Save and run the workspace. Check to see what difference this update has made.





Advanced Task 2

It would be useful to write a little more output than just 10 point features, to give a backdrop to the data. As a raster-supporting format, the PDF writer is ideal for this.

1) Create Writer Feature Type

Create a new writer feature type (either duplicate the existing one, or use **Writers > Add Feature Type**). Connect the reprojected CDED data to it.

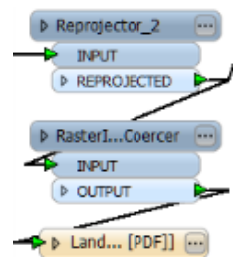
By default, writing the Raster data directly to PDF is not going to be much use, as it is a single-band of elevations not really compatible with PDF. If you run the workspace, the log has a warning that states as much:

Unsupported number of bands: 1. Only rasters with 3 or 4 bands are supported.

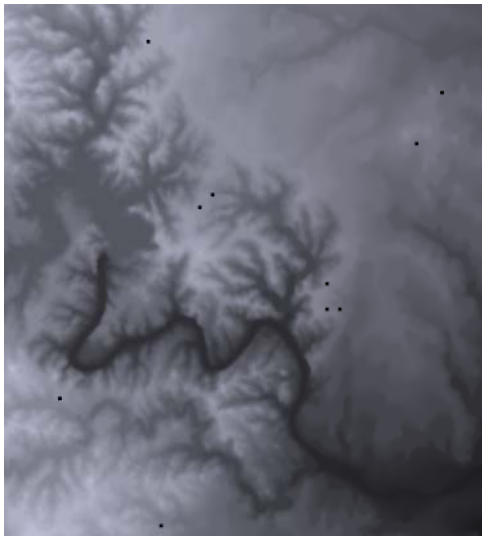
2) Place a Transformer – *RasterInterpretationCoercer*

To change the data to a color raster place a *RasterInterpretationCoercer*.

Set the interpretation type to RGB24. This means 3x8 bit bands representing red, green and blue values from 0 to 255 ($2^8 - 1$)



This, with the Conversion Options, will convert the numeric values into color, scaling the Z value between 0 and 255.



Now we have a raster backdrop to the point data

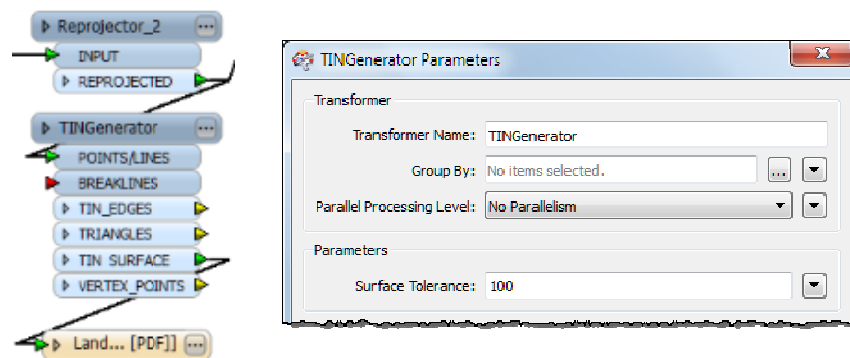
However, that solution doesn't make use of the 3D nature of PDF. We can remedy that!

3) Place a Transformer – *TINGenerator*

Replace the *RasterInterpretationCoercer* with a *TINGenerator* transformer.

Connect the reprojected raster to *TINGenerator:POINTS/LINES* and then *TINGenerator:TIN_SURFACE* to the writer feature type.

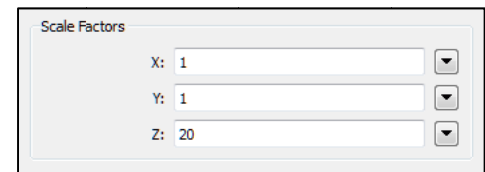
Set the *TINGenerator* Surface Tolerance parameter to 5



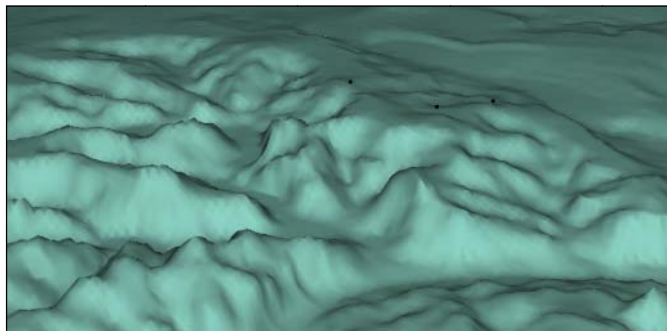
4) Place a Transformer – *Scaler*

The output from this workspace is now a bit flat (run it and check if you like). It would be helpful to exaggerate the Z scale to show the relief a little better.

Place a *Scaler* transformer AFTER the *TINGenerator* and set it to scale the Z value by a factor of 20.



Now when you run the workspace the PDF will have a full 3D backdrop to the point data.



NB: You'll have to also scale the Z values of the points if you want to see them on top of the raster backdrop (and scale the feet and metres versions differently)!



Session 4 – Datasets and Feature Types



This exercise makes use of the dataset related skills you have just learned to do a change detection process.

Exercise: Datasets and Feature Types	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property lot lines (MapInfo MIF, AutoCAD DWG)
Overall Goal	Compare developer proposals against a base dataset to detect changes
Starting Workspaces	None
Finished Workspaces	C:\FMEData\Workspaces\DesktopWorkbook\Exercise3aComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise3bComplete.fmw

Objective

When a property developer submits a planning application to the City of Interopolis, they also provide a set of new property boundaries to update the city's records.

Your current task is to set up an FME solution that will compare a set of developer DXF lot lines to the City's property map, to determine which features to upload and replace in that map.

By doing this, you'll achieve the following objectives:

- Set up a workspace to read the city's current property datasets (MIF/MID format).
- Add a new Reader to read the developer's DXF dataset.
- Clean all source data to ensure a better match.
- Use a *ChangeDetector* transformer to compare city data to the developer's new property lines.
- Add a new Writer to write the data to DGN format.

Detailed Steps**1) Open a Blank Workspace**

Start Workbench and begin with a blank workspace.

2) Add a Reader

First we'll set up the workspace to read the city's base datasets. Select **Readers > Add Reader** on the menu bar. Add a new reader to the workspace to read the following property dataset:

Reader Format MapInfo MIF/MID
Reader Datasets All MIF files in the folder C:\FMEData\Data\Properties\

Workflow Option Single Merged Feature Type



Notice how the Feature Type title is "<All>" to reflect its dynamic status.

3) Add a Reader

Now we can read the new developer data.

On the menu bar, select **Readers > Add Reader**.

Click the Browse button to browse for and select the developer's DXF dataset.

Reader Format Autodesk AutoCAD DWG/DXF
Reader Dataset C:\FMEData\Data\Developer\lotlines.dxf

Parameters Group Entities By: Attribute Schema

Workflow Option Single Merged Feature Type

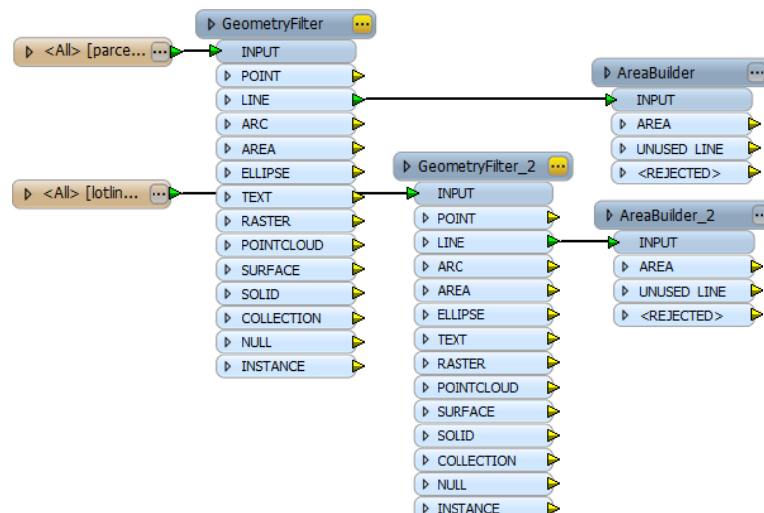
The dynamic option is used here – not because there are multiple files – but because we aren't sure what layer the developer used to store data on.

4) Place a Transformer – GeometryFilter

Change detection is going to be tricky because we won't know whether the developer turned his lines into polygons or not. The easiest solution is to convert all features into polygons.

Before we form polygons, let's make sure we're only working with line features by using a *GeometryFilter* to filter out point features before attempting to build polygons.

Place two *GeometryFilter* transformers, one connected to each reader feature type.



5) Place a Transformer – AreaBuilder

Now place two *AreaBuilder* transformers. Connect each to a *GeometryFilter:LINE* port.

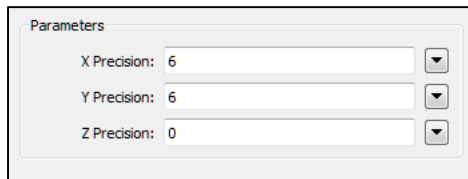
The default parameters will be fine.

6) Place a Transformer – *CoordinateRounder*

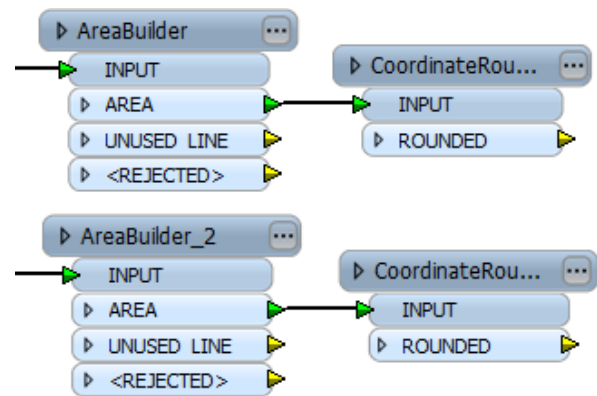
Change detection could be adversely affected if the developer's data has a different precision than City standards.

To avoid potential problems add two *CoordinateRounder* transformers.

Connect each to an *AreaBuilder:AREA* port.



Set both X and Y coordinates to be rounded to six decimal places. All data now has the same level of precision.

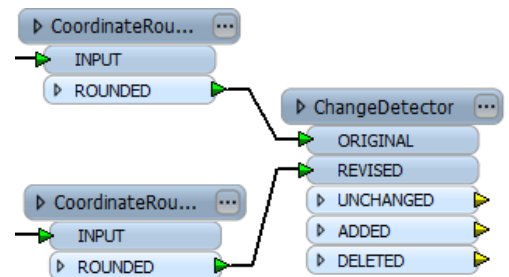


7) Place a Transformer – *ChangeDetector*

Now the data has been fully prepared, change detection can take place. Place a *ChangeDetector* transformer in the workspace.

Connect the city data
CoordinateRounder:ROUNDED
port to *ChangeDetector:ORIGINAL*

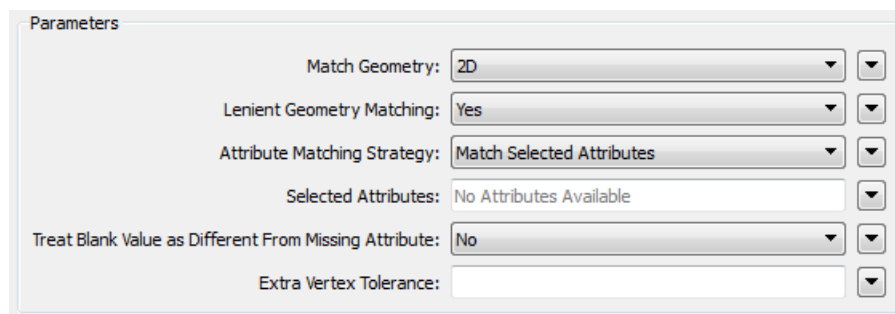
Connect the developer data
CoordinateRounder:ROUNDED port to
ChangeDetector:REVISED



8) Check Transformer Parameters

Check the *ChangeDetector* parameters to ensure the Match Geometry setting is set to '2D' and the Lenient Geometry Matching setting is set to 'Yes'. Set the Attribute Matching Strategy to 'Match Selected Attributes' – but don't select any!

Lenient Geometry Matching ensures a match when features have the same coordinates, but – for example – are in the reverse order.



9) Run Workspace

Add *Inspector* transformers to all output ports of the *ChangeDetector*.

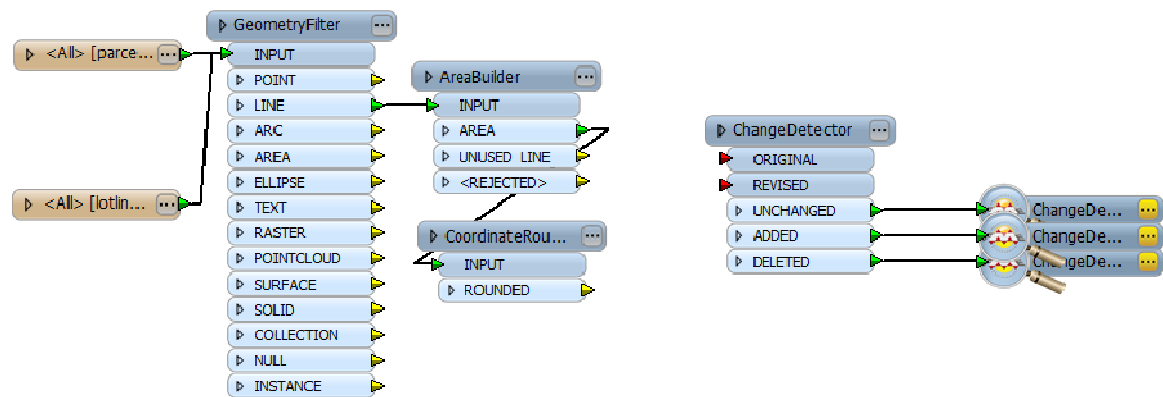
Save and run the workspace, and ensure it is producing the output you expect.



Advanced Tasks

When you look at this workspace, a number of transformers are being duplicated; there are two *GeometryFilter* transformers, two *AreaBuilder* transformers, and two *CoordinateRounder* transformers.

It might be more efficient for the workspace if you could send both city and developer data through the same transformers like this:



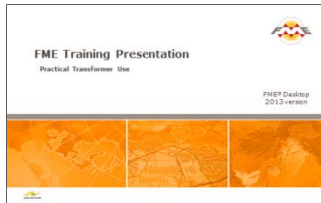
But can this be achieved? What connections would you make?

The problems to overcome:

- How can you ensure city and developer data is processed separately in the *AreaBuilder* transformer? You don't want to build polygons from half of each.
- Having merged all the data into a single stream, how can you separate it out again to connect to the ORIGINAL and REVISED ports of the *ChangeDetector*?



Session 5 – Practical Transformer Use



This exercise uses advanced transformer techniques and conditional filtering to identify specific features within the city.

Example 4: Pollution Monitoring with Lists	
Scenario	FME user; City of Interopolis, Planning Department
Data	EPA Facilities, Zipcodes (Input: GML, MIF; Output: Logger)
Overall Goal	Show the highest polluting company for a user-defined zipcode
Demonstrates	List and Parameter Handling in FME 2012
Starting Workspace	C:\FMEData\Workspaces\DesktopManual\Exercise4aBegin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopManual\Exercise4aComplete.fmw C:\FMEData\Workspaces\DesktopManual\Exercise4bComplete.fmw

Objective

The task here is to discover which of the monitored facilities in the city is creating the most pollution on any particular day, in a user-selected zipcode.

In other words, the user selects a zipcode, and FME returns the name of the highest polluter for that area. Although there are various solutions, this one will use a list in order to demonstrate list attributes.

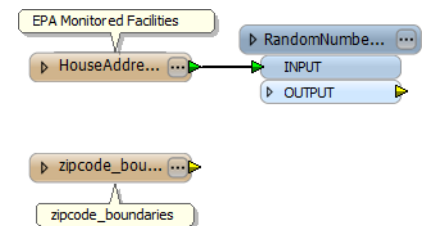
Detailed Steps

1) Start Workbench

Start Workbench and open the beginning workspace.

Notice that it reads from a GML format dataset of EPA (Environment Protection Agency) listed sites, and a MapInfo MIF dataset of zipcode boundaries. The pollution values are set as random numbers. There is no output format; the output will either be logged or sent to the FME Universal Viewer.

As always, inspect the source data to see what data you are dealing with, and check the *RandomNumberGenerator* parameters.



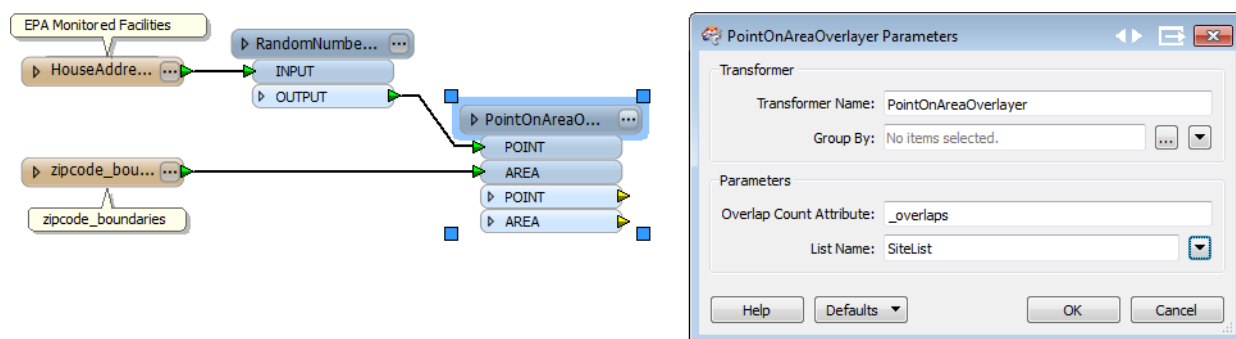
2) Add a PointOnAreaOverlayer

To discover which sites fall into which zipcodes, a *PointOnAreaOverlayer* transformer can be used. Place one of these transformers.

Connect *RandomNumberGenerator:OUTPUT* to *PointOnAreaOverlayer:POINT*

Connect *MIF:zipcode_boundaries* to *PointOnAreaOverlayer:AREA*

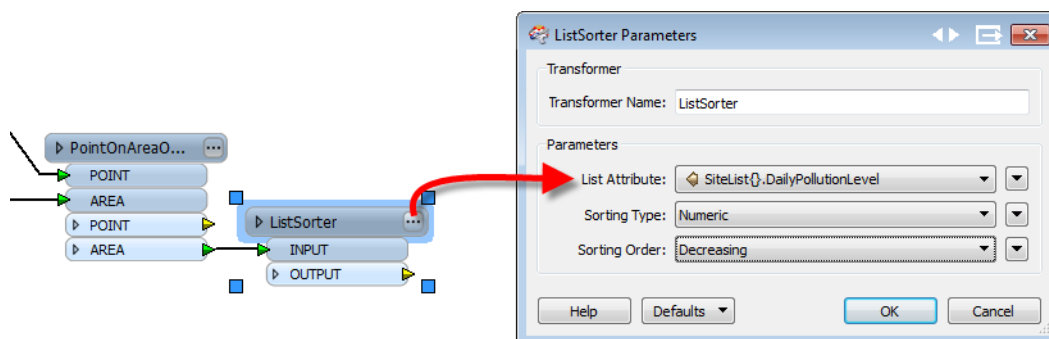
Open the properties dialog for the *PointOnAreaOverlayer* and enter a list name of SiteList



3) Add a ListSorter

Add a *ListSorter* transformer connected to the *PointOnAreaOverlayer:AREA* output port. Open the properties dialog and set it to sort on the attribute SiteList{}.DailyPollutionLevel

Set Sorting Type = Numeric, and Sorting Order = Decreasing



This will ensure the site with the highest pollution level is on top of the list.

4) Add an AttributeCreator

Add an *AttributeCreator* transformer connected to *ListSorter:OUTPUT*

This will be used to create a string reporting the highest polluter.

Open the parameters dialog, and set the Attribute Name field to HighestPolluter

In the value field, open the String Editor dialog.

Enter

Constant: The highest polluter in zipcode<space>

Attribute Value: zipcode

Constant: <space>is:<space>

Attribute Value: SiteList{}.FacilityName

When prompted for the list element number, select 0 (the default) to pick the top of the list.

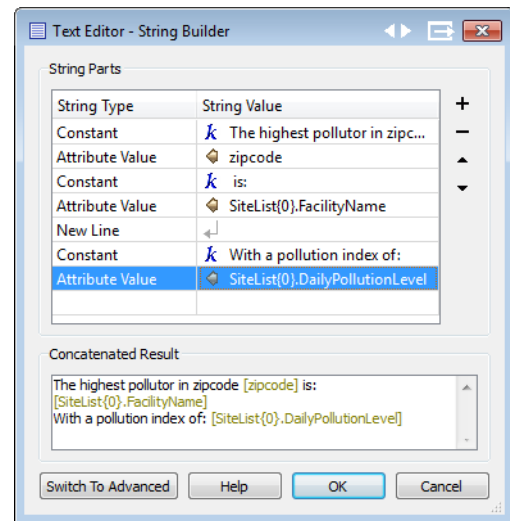
New Line:

Constant: With a pollution index of:<space>

Attribute Value: SiteList{}.DailyPollutionLevel

Again, when prompted for the list element number, select 0 to pick the top of the list.

The String Builder should now look like this:



5) Add a Tester

Now a *Tester* must be used to ensure only the chosen zipcode is output.

Add a *Tester* transformer connected to

AttributeCreator:OUTPUT

Open the Parameters dialog.

Under the Test Clauses section, set:

Left Value: Attribute Value > zipcode

Operator: =

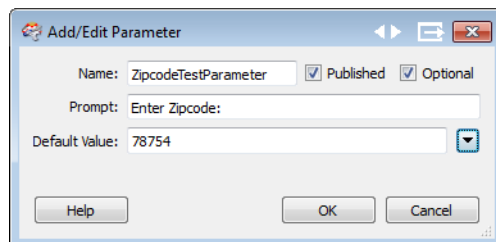
Right Value: Parameter > Create User Parameter

When prompted, create a new parameter as follows:

Name: ZipcodeTestParameter

Prompt: Enter Zipcode

Default Value: 78754



NB: Valid zipcode numbers are:

- 78724
- 78751
- 78752
- 78753
- 78754

Click **OK** to accept this dialog.

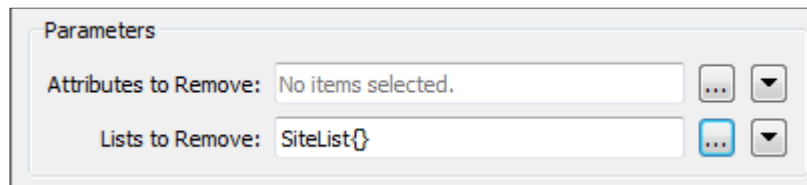
The final test clause should now look like this:

Left Value	Operator	Right Value	Negate	Mode
1	zipcode	=	\$(ZipcodeTestParameter)	<input type="checkbox"/> Automatic

6) Place ListRemover

Use Quick Add to place a *ListRemover*. Notice how the transformer actually placed is an *AttributeRemover*. The *ListRemover* has been merged into this in FME2012.

Connect it to the *Tester:PASSED* output port, open the parameters dialog, and set it up to remove the list attribute SiteList{}



Parameters

Attributes to Remove: No items selected. [...]

Lists to Remove: SiteList{} [...]

7) Run Workspace

Connect a *Logger* transformer to the *AttributeRemover:OUTPUT* port.

Run the workspace (use Prompt+Run if you wish to set Zipcode to something other than the default).

Once complete you should find an attribute in the log similar to:

```
INFORM|Attribute(encoded: utf-8): 'HighestPolluter' has value
`The highest polluter in zipcode 78754 is: IMAGE MICROSYSTEMS INC
With a pollution index of: 9.36'
```

Remember, because the pollution data is based on a random number, you will get a different result each time you run the workspace.



Advanced Task

As an advanced task, why not try writing this information to some form of output.

You might choose to write the data to a spatial format and use the HighestPollutor string as a label or piece of annotation.

Alternatively, you could create a text file containing a list of the ten highest polluters – and even format the text file content using HTML strings. This would be perfect for use with FME Server.

1) Add Writer

Add a writer using Writers > Add Writer.

Writer Format	Text File
Writer Dataset	C:\FMEData\Output\DesktopTraining\Pollution.html

Add the created feature type to replace the Logger transformer.

2) Update Message

Now update the message to HTML. Open the AttributeCreator and first change the attribute name to text_line_data – this way it will get written to the Text File writer.

Open the text editor for the string expression, and switch to the advanced dialog.

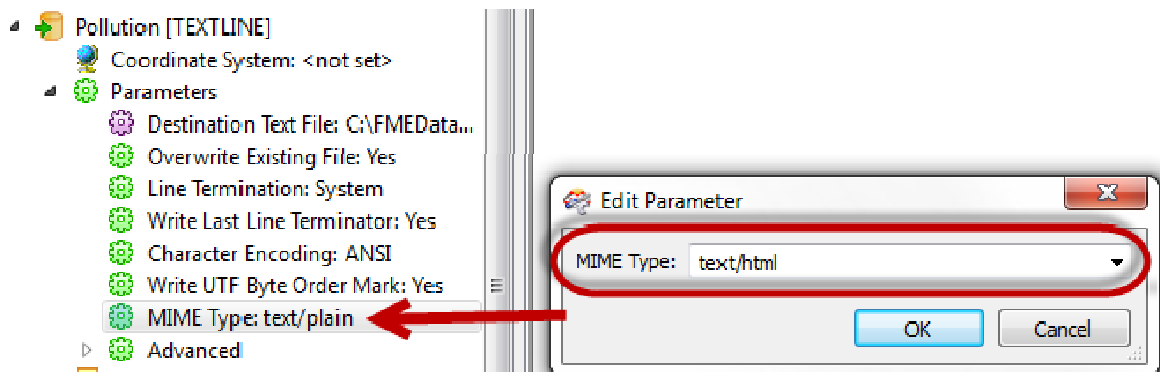
Update the message to be formatted as HTML, for example:

```
<!DOCTYPE html>
<HTML>
<h1>Pollution Report</h1>
<p>The highest polluter in zipcode @Value(zipcode) is:
@Value(SiteList{0}.FacilityName)</p>
<p>With a pollution index of: @Value(SiteList{0}.DailyPollutionLevel)</p>
</HTML>
```

Click OK twice to accept the changes and close the dialog.

3) Set Mime Type

Set the output MIME type parameter in the Navigator window. Ensure it is set to text/html

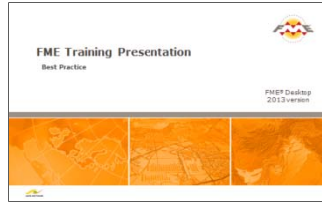


4) Run on FME Server

If you have FME Server, publish the workspace as a Data Streaming service and run it online.



Session 6 – Best Practice



This exercise covers a range of functionality from different course modules; of course the aim is to use best practice ideas while carrying it out.

Exercise: Best Practice	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property lot lines (MapInfo MIF); Google Earth KML
Overall Goal	Convert a set of property lines in MapInfo MIF format, to a set of polygons in Google Earth KML format; simultaneously cleaning geometry and attributes.
Starting Workspace	C:\FMEData\Workspaces\DesktopWorkbook\Exercise5aBegin.fmw
Finished Workspace	C:\FMEData\Workspaces\DesktopWorkbook\Exercise5aComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise5bComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise5bComplete.fmw

Objective

This exercise continues the exercise from Session 2 (Data Transformation), which converted a property lot lines dataset, while validating the dataset's geometry and attributes.

Unfortunately there were just too many invalid features for the output to be useful.

Because this is an urgent requirement, the requestor cannot wait for the data to be edited manually. Being impressed with your past work, she asks for you to fix the data using FME.

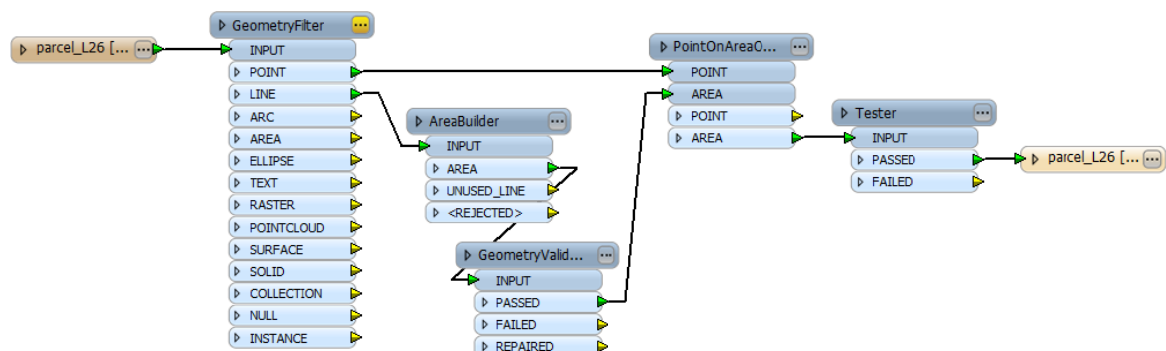
By doing this you fulfill the following objectives:

- Clean up bad linework to avoid non-closing polygons
- Create a new set of polygon centroids to ensure there is one per polygon
- Create data representing city blocks, and assign a unique ID to all properties on that block
- Write blocks, lots and centroids to separate output feature types

Detailed Steps

1) Start Workbench

Start Workbench (if necessary) and open the starting workspace.



2) Set Up Schema

Let's make the first task to tidy the reader and writer schemas.

Add two new writer feature types (either by **Writers > Add Feature Type**) or by duplicating the existing writer type.

Rename the three writer feature types as Blocks, Lots, and Centroids.

The Lots feature type should have just one user attribute; PARCEL_ID.

The Blocks feature type should have just one user attribute; BLOCK_ID

The Centroids feature type should have just one user attribute; PARCEL_ID



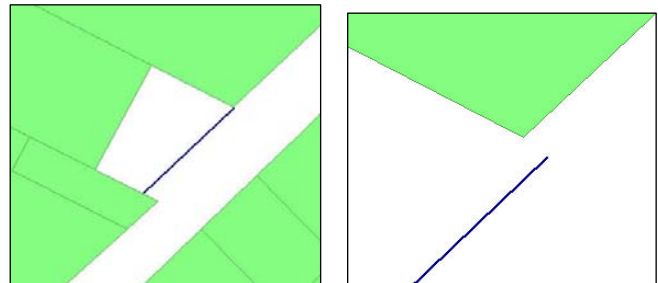
On the reader side, hide all attributes except BLOCK_ID and PARCEL_ID

3) Check Problem Linework

Let's check the sort of problems that can be found in the data.

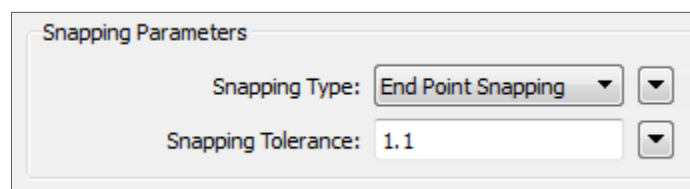
Add Inspectors to both *AreaBuilder* output ports. Run the workspace.

Examine the unused lines layer. Most are unrelated to any polygon, but two in particular seem to be part of polygons that are not being properly formed.



4) Fix Problem Linework

The most obvious solution to a non-continuous line is to snap it into place. FME 2013 has snapping capability built into the *AreaBuilder* transformer, for this very occasion.



Open the *AreaBuilder* parameters dialog. Set Snapping Type to End Point Snapping.

Experiment with the snapping tolerance setting to find the smallest possible tolerance that fixes the two problem polygons (remember, you can use FME Universal Viewer to measure the gaps). As a guide, try 1.1 (feet).

5) Locate Bad Geometry

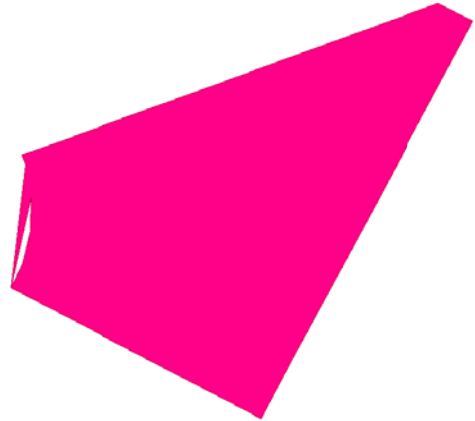
At the moment the *GeometryValidator* is merely locating bad data. However it also has the ability to fix the problems it finds.

Connect Inspector transformers to the *GeometryValidator* and run the translation to see what features it reports as bad (FAILED).

One is fairly obvious, whereas the other is less obvious. You'd need to examine the coordinates individually to find the problem.

Now open the *GeometryValidator* parameters dialog and see if any of the reported issues are listed as "Repairable".

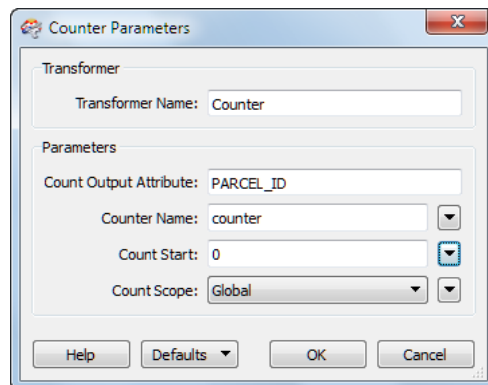
If so, set Attempt Repair to Yes and re-run the workspace. Check to ensure these features emerge from the REPAIRED port and are now correct, and make sure the REPAIRED port is also connected to the *PointOnAreaOverlayer*



6) Create New Parcel IDs

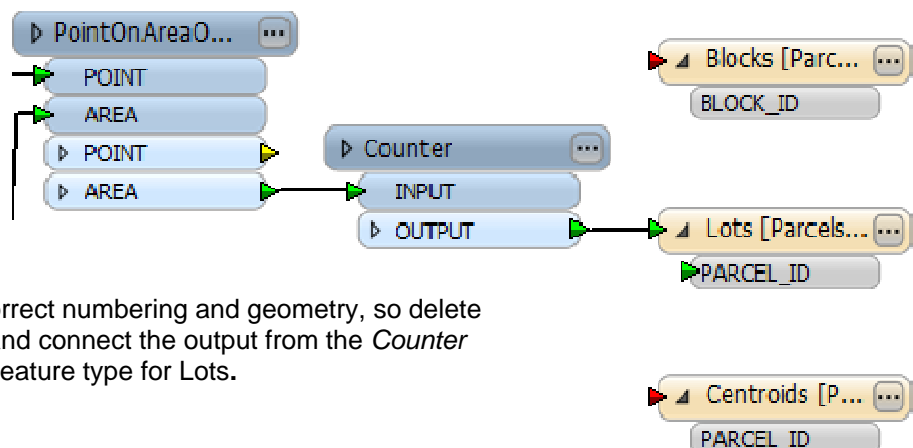
Having tidied up the linework, we can now tidy up the ID attributes.

Let's create new values for *PARCEL_ID* to ensure a correct match between parcel and point.



Place a *Counter* transformer connected to the *PointOnAreaOverlayer*.AREA output port.

Set the Count Output Attribute to *PARCEL_ID* so it overwrites the existing values.



The parcels have correct numbering and geometry, so delete the existing Tester and connect the output from the *Counter* into the destination feature type for Lots.

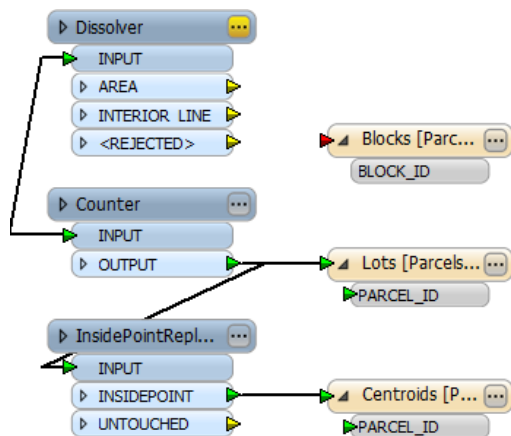
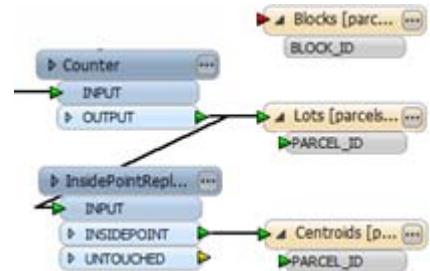
7) Create Centroids

We now need to create a new centroid – a point guaranteed to be inside each polygon. The *CenterPointReplacer* may be adequate – but we can't be sure that all the points it generates will be inside the correct polygon, so place an *InsidePointReplacer* transformer.

Create a second connection from the *Counter*:*OUTPUT* port and into the *InsidePointReplacer*.

This will turn each polygon feature into a new centroid.

Connect the *InsidePointReplacer*:*INSIDEPOINT* port into the writer feature type for *OGCKML:Centroids*



8) Create City Blocks

The final task is to create a new set of features to represent city blocks and to give them a suitable ID number.

A city block is comprised of a number of adjacent lot polygons, so the first task is to remove all the inner boundaries between adjacent lots to produce a city block outline.

Place a *Dissolver* transformer connected either before or after the *Counter*.

Route the *Dissolver* output port labelled *AREA* to an *Inspector*, delete or disable any other existing *Inspectors* and run the workspace again.

9) Inspect City Block Output

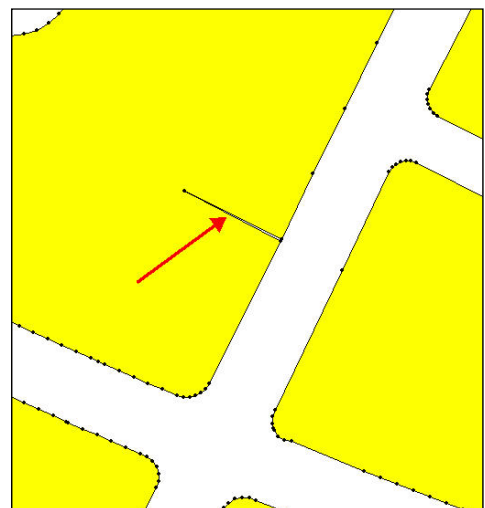
Inspect the City Block output in the FME Universal Viewer. Can you see any problems?

In fact there may be one or two pieces of bad geometry; spikes in the data.

Can you locate them with the FME Universal Viewer?

If not, they may not exist! The problems may have been cleared up by the *Snapper* depending on the tolerance you used.

If you do see these problems, do you know why they still occurred, even though we already used the *GeometryValidator*?

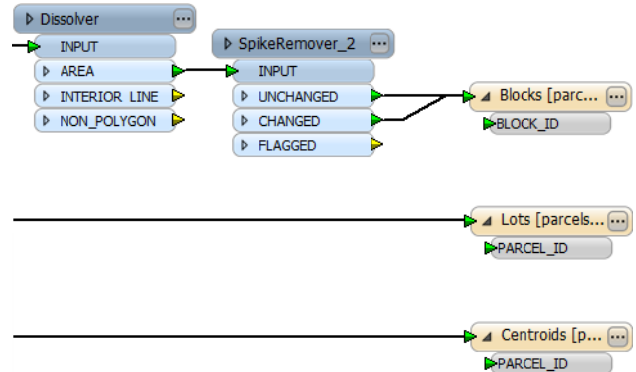


10) Remove Spikes

Place a *SpikeRemover* transformer, this time after the *Dissolver*.

Add *Inspectors* and experiment with the *SpikeRemover* parameters until you have removed the spike (or spikes).

Try to make the spike angle and, optionally, length parameters as small as possible to avoid removing linework that is already correct.



When happy, connect both UNCHANGED and CHANGED output ports to the Blocks Feature Type.

11) Create New Block IDs

One of the reasons for doing this process was to create new values for BLOCK_ID, so place a further *Counter* transformer between the *SpikeRemover* and the *Blocks* Feature Type.

Set the Count Output Attribute to BLOCK_ID so it overwrites the existing values.

Questions: What difference does it make having two *Counter* transformers in one workspace? How may it cause problems? What parameters could you change to avoid any issues?

Run the workspace and examine the output using FME Universal Viewer.



Advanced Tasks

There's one more vital task that should be done here. Can you see it?

Yes – the blocks have a BLOCK_ID value, but the same value needs to be applied to each of the lot parcels within that block. Currently the BLOCK_ID on each lot does not match the new value.

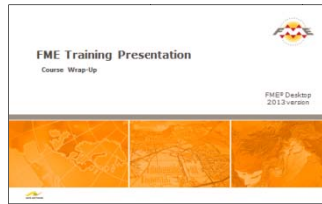
First, find a way to transfer the BLOCK_ID values onto the correct lot parcels. You need to consider whether this can be done using attributes or if it must be done spatially.

Second, use your knowledge of FME Best Practice to tidy up the workspace.

This may include the following:

- Annotating transformers to describe what they do, or adding general annotation.
- Moving sets of items and bookmarking them to form distinct groups.
- Renaming the workspace to provide a better illustration of what it does.
- Filling in other workspace properties.
- Performance Tuning the workspace
- Packaging the workspace and data as a template.

Session 7 – Group Project



This is an extra group project to carry out if you have the time and opportunity.

This additional (optional) exercise simulates a complete project.

There are three key tasks to the project. You may wish to do all three yourself, or your training instructor may wish to divide the class into three teams, each responsible for a part of the project.

Even if you aren't able to do the project, it will be worthwhile practice to look through the workspaces and other resources, to understand how it works.

As proficient FME users, for this exercise you'll receive fewer instructions and will have to use your own FME skills to identify and develop a solution for each task. *If you do get stuck, and have really tried to work it out yourself, then don't hesitate to ask your instructor for assistance.*

Exercise: Bridge Inventory Project	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads (GPS/CSV format), Railroad (SDL format), Transit (SDF format), Hydrography (MIF/MID format)
Overall Goal	To create an inventory of transit bridges in the City of Interopolis

The city engineer is going into the field to inspect transit bridges. There are no existing maps or datasets and so you have been asked to create the preliminary dataset.

In brief, the overall project is to create an inventory of transit bridges for the city of Interopolis. A bridge is assumed to exist wherever a transit route crosses a hydrographic feature.

The three tasks are:

- Data Preparation
 - Set up the data ready for extraction. This will include processing raw GPS data and reprojecting data into the correct coordinate systems.
- Data Extraction
 - Extract the location of bridges from the prepared data. This will include identifying intersecting features and testing whether they are a true bridge location.
- Schema Definition and Schema Mapping
 - Define the schema of the output data. This will include processing incoming data to match both the required schema and symbology.



Task 1: Data Preparation

Exercise: Bridge Inventory Project	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads (GPS/CSV format), Railroad (SDL format), Transit (SDF format), Hydrography (MIF/MID format)
Overall Goal	To create an inventory of transit bridges in the City of Interopolis
This Step	Data preparation
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopWorkbook\Exercise6aComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise6aBRIDGEINVENTORYPREP.fds

The task here is to create a workspace that prepares the source datasets in readiness for their transformation and data extraction. You don't need to do the part that identifies bridges, just ensure data is all in the same coordinate system and ready for transformation.

The source datasets are:

Reader Format	Autodesk MapGuide SDL
Reader Dataset	C:\FMEData\Data\Railroads\railroad.sdl
Reader Format	Autodesk MapGuide Enterprise SDF
Reader Dataset	C:\FMEData\Data\Transit\Transit.sdf
Parameters	Tables:metrorail
Reader Format	MapInfo MIF/MID
Reader Dataset	C:\FMEData\Data\Hydrography\HydrographyLine.mif
Reader Format	Comma Separated Value (CSV)
Reader Dataset	C:\FMEData\Data\GPS\major_roads.csv

Suggested Steps

To give you some assistance, here are some suggested steps.

Overall most of the data is already suitable for processing; only the roads data – which is still in a raw GPS format – will require much preparation work.

As usual, inspect the data in the FME Universal Viewer to see what it looks like.

To tie this part of the project into the other two tasks, it's suggested that you create the workspace and then turn it into a Custom Format. If you aren't familiar with custom formats in FME, then ask your instructor for assistance when you get to that step.

Once this is complete you can pass the custom format to the schema creation team, who will be able to use it as the first stage in their workspace, and to the data extraction team who will want to use it to test their transformations with.

1) Create Workspace

The first step should be to create a basic workspace to read the data.

With multiple formats it's probably easier to start with an empty canvas and simply add as many readers as are required to read all the source datasets.

2) Add Writer

Once readers have been added a writer is required.

If the intention is to turn this into a Custom Format then the format shouldn't matter, but FFS would be a good choice in either case.

So add an FFS format writer and duplicate all reader feature types on the writer.

3) Process GPS

Now the GPS data needs processing. It is a plain text file with X,Y coordinates in different fields.

A simple combination of *2DPointReplacer* and *PointConnector* transformers will achieve this goal.

4) Add Attribute Data

You'll notice the GPS data has no attributes. These are stored in a Microsoft Access Database

Reader Format	Microsoft Access
Reader Dataset	C:\FMEData\Data\GPS\road_attrs.mdb

The next step should be to connect these attributes onto the correct road features with a *Joiner*.

5) Update major roads schema

Once attributes have been added to the GPS data, they should also be defined in the writer schema, so add them there and remove any existing attributes.

6) Create Custom Format

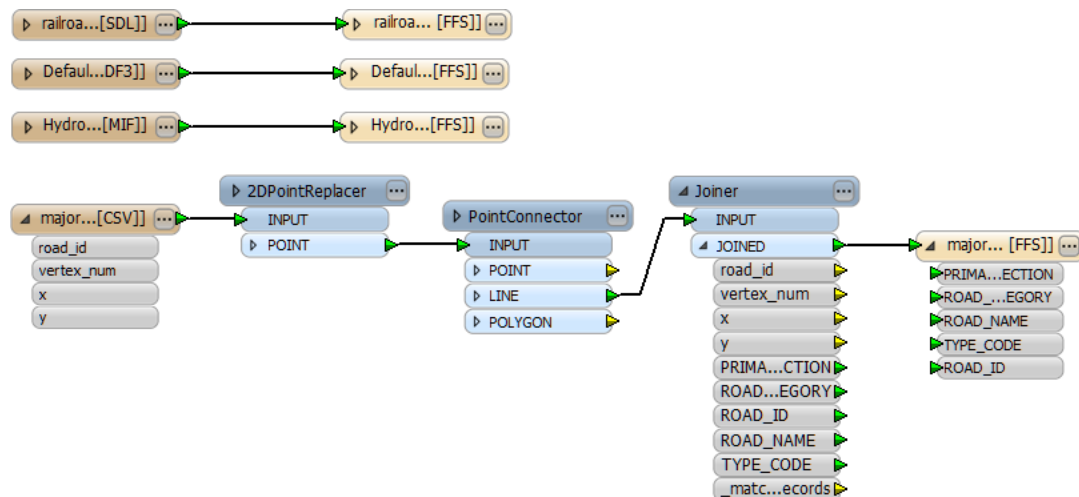
Now a Custom Format can be created.

Save the workspace and then choose **File > Export as Custom Format** from the menubar.

7) Update/Check Permitted Geometries

In the custom format, open all of the writer feature types and check the allowed geometry settings. They should all be set to *fme_line*, since these are all line features.

Your custom format could now look very much like this:





Task 2: Data Extraction

Exercise: Bridge Inventory Project	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads (GPS/CSV format), Railroad (SDL format), Transit (SDF format), Hydrography (MIF/MID format)
Overall Goal	To create an inventory of transit bridges in the City of Interopolis
This Step	Data Extraction
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopWorkbook\Exercise6bComplete.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise6bBridgeInventoryExtract.fmx

The task here is to create a workspace that extracts bridge locations by intersecting transit routes with hydrography lines and saving the intersection points

The datasets are the same as in Task 1

Suggested Steps

To give you some assistance, here are some suggested steps.

As usual, inspect the data in the FME Universal Viewer to see what it looks like.

Because the source data – particularly the roads – is being prepared by Team 1, you will need to either create your own test data to use, or use that data in its raw state. The best method would probably be to create a solution using Railroad and Transit Data and try the solution on the roads when they are available.

To tie this part of the project into the other two tasks, it's suggested that you create the workspace and then turn it into a Custom Transformer.

Once this is complete you can pass the custom transformer to the schema creation team, who will be able to use it as the second (transformation) stage in their workspace.

1) Create Workspace

The first step should be to create a basic workspace to read the data.

With multiple formats – and no need for a writer of any sort – it's probably easier to start with an empty canvas and simply add as many readers as are required to read all the source datasets

2) Process Data

An *Intersector* transformer is one way of calculating nodes at intersection points.

3) Identify Transit/Hydrography Nodes

The key problem to overcome is how to identify which nodes are the result of a transit/hydrography intersection, and not a transit/transit intersection.

The easiest way is to add a flag – probably an attribute – identifying the original features. When the output possesses both a transit and hydrography flag then it is one of the nodes to be kept.

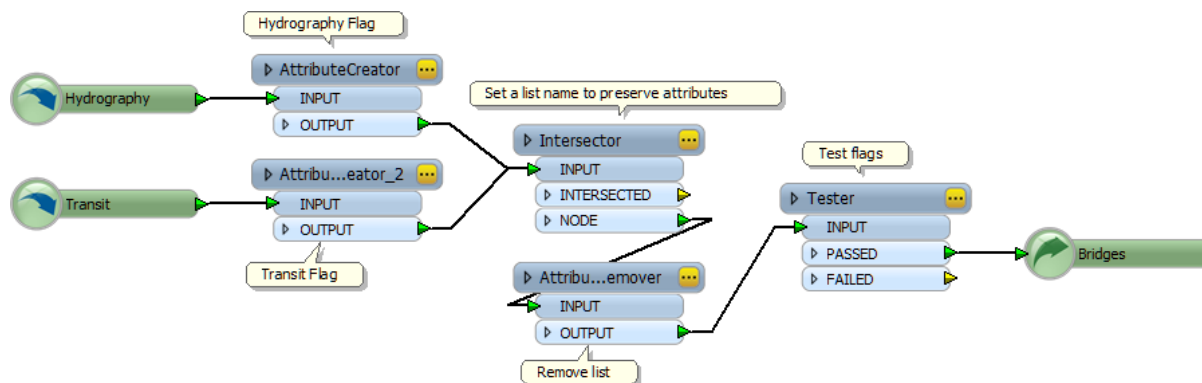
4) Create Custom Transformer

Now a Custom Transformer can be created.

Select the transformers in the workspace, right-click and choose Create Custom Transformer.

Export the custom format as an *fm*x file so that it can be passed to the schema creation team.

Your custom transformer could look very much like this:





Task 3: Schema Definition and Schema Mapping

Exercise: Bridge Inventory Project	
Scenario	FME user; City of Interopolis, Planning Department
Data	Roads (GPS/CSV format), Railroad (SDL format), Transit (SDF format), Hydrography (MIF/MID format)
Overall Goal	To create an inventory of transit bridges in the City of Interopolis
This Step	Schema Definition and Schema Mapping
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\DesktopWorkbook\Exercise6cTest.fmw C:\FMEData\Workspaces\DesktopWorkbook\Exercise6cComplete.fmw

The task here is to create a workspace that reads incoming source data (from Team 1's Custom Format), calculates bridge locations (from Team 2's Custom Transformer) and writes them to the correctly-formatted output.

The datasets are the same as in Task 1

The output might be one of two formats. You must produce a workspace that allows the user to choose which format they wish to write to, and then resymbolize the data accordingly.

The two formats are:

- Google Earth KML
- Autodesk AutoCAD DWG/DXF

The required symbology is:

	Color	Thickness/Weight	Symbol
Roads	Red (1,0,0)	Thin	n/a
Railroads	Magenta (1,0,1)	Thin	n/a
Metrorail	Orange (1,0.66,0)	Thin	n/a
Hydrography	Blue (0,0,1)	Medium	n/a
Bridges	Red (1,0,0)	n/a	Any

Suggested Steps

To give you some assistance, here are some suggested steps.

As usual, inspect the data in the FME Universal Viewer to see what it looks like.

Because the source data – particularly the roads – and bridge locations are being prepared by Teams 1 and 2, you will need to either create your own test data to use, or find other raw data that can be used to prototype a solution.

Once Teams 1 and 2 are complete, you can plug their Custom Format and Custom Transformer into your workspace to complete the project.

1) Create Workspace

The first step should be to create a basic workspace with some test data.

Since the source data is still undergoing preparation, it might be easier to use *Creator* transformers to create some simple line and point features.

2) Add Writer

Add a Generic Writer and create feature types

3) Add Styler Transformers

Add a series of styler transformers for one format (*KMLStyler* or *DWGStyler*) to create the proper symbology.

4) Add More Styler Transformers

Now add a series of styler transformers for the other format (*KMLStyler* or *DWGStyler*).

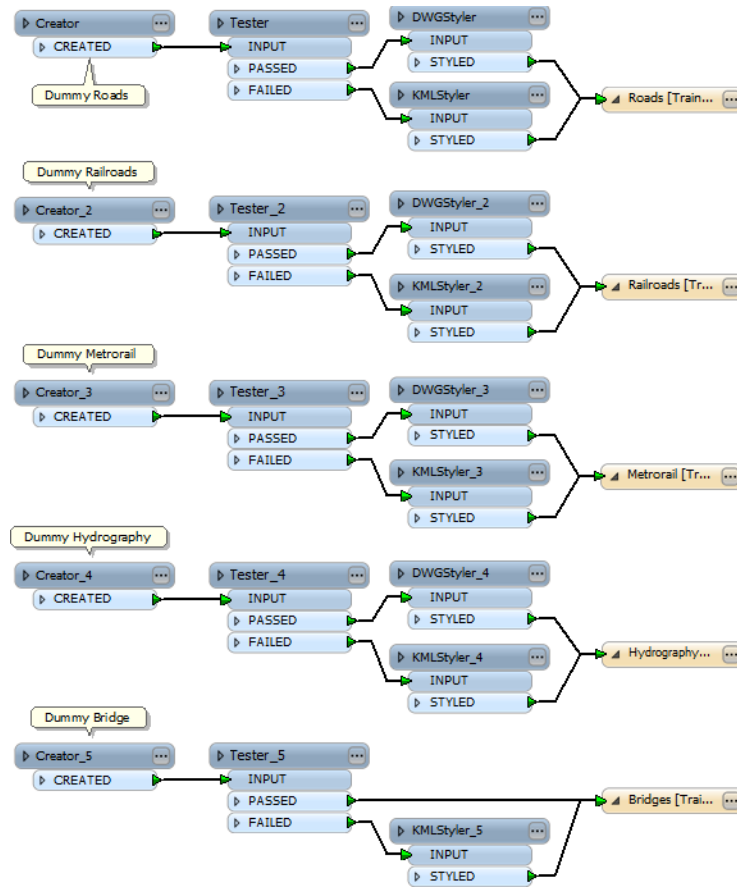
5) Add Published Parameter

Add a published parameter to ask the user which format to write.

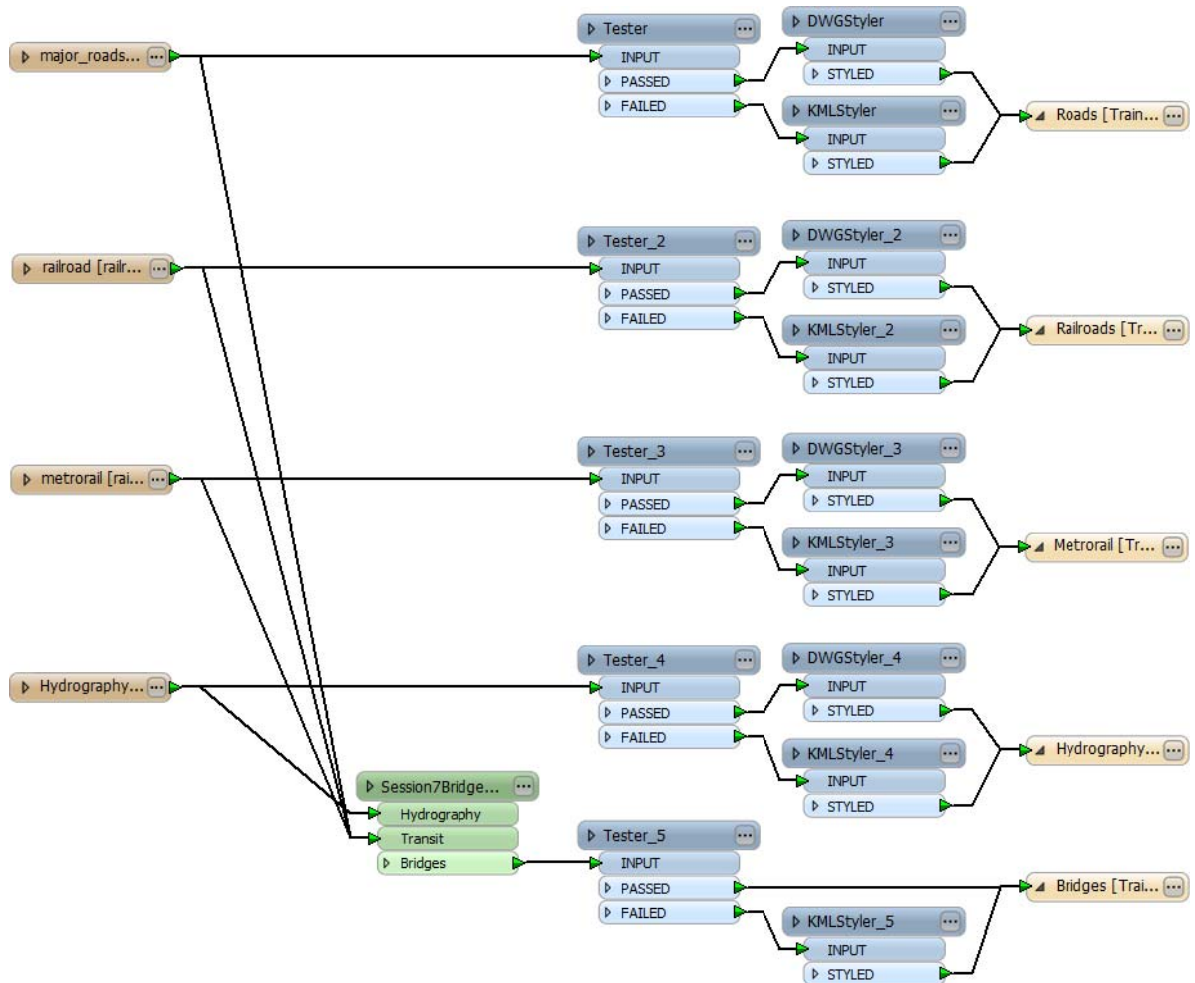
6) Map Schema

Route incoming features to the various stylers depending on the user defined format

The test workspace may now look like this, ready to insert the parts from Teams 1 and 2.



The final workspace – with parts 1 and 2 integrated – may look something like this:



Advanced Tasks

If you have time to take the whole concept further, then there are a number of potential improvements:

- Record the type of transit feature that crosses each bridge (road, rail, etc)
- Record the name or ID of the transit feature that crosses each bridge
- Read in bus routes from the Transit dataset, and record which routes cross each bridge
- Buffer the bridge location and use that to clip an area of aerial imagery around the bridge