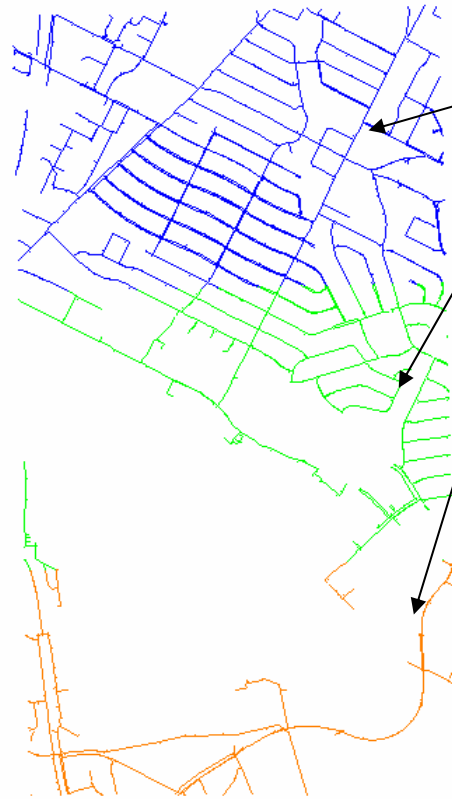
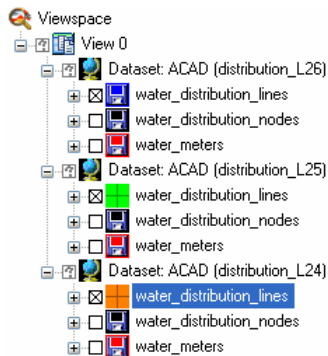




FME User Group

1. Multiple Input Files – 1 Source
2. Creating a Custom Transformer
3. *Use MRFCleaner to repair topological errors*

Multiple Input Files – 1 Source

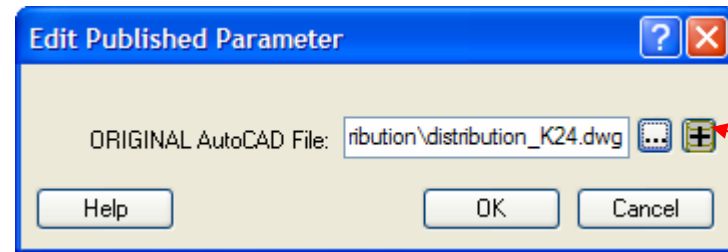
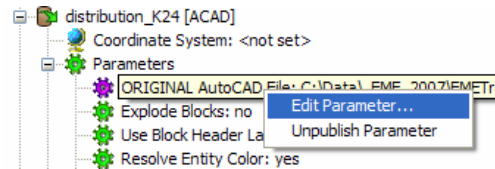


- Example: Water line network
 - Data separated into multiple tiles, separate DWGs
- Goal: read all the DWG into 1 source to avoid having 30 input sources

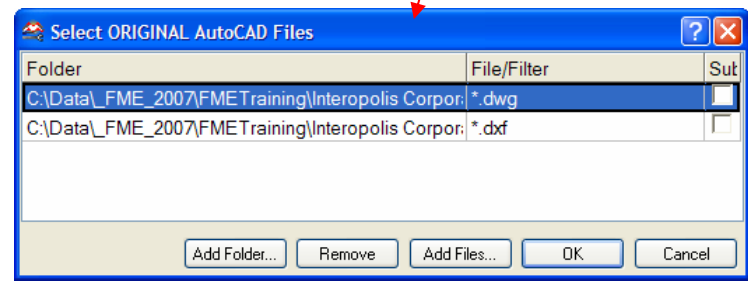
1. File Formats

ex. DWG, DGN

- In the Workspace pane, right click the data source parameter and select 'Edit Parameter'
- 2 Options
 - **Browse** tool: if all the data is in 1 folder
 - **Swizzler** tool: Can point to files or folders in different locations

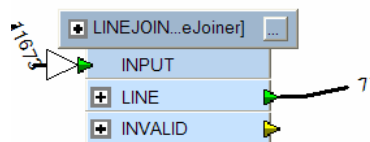


Swizzler



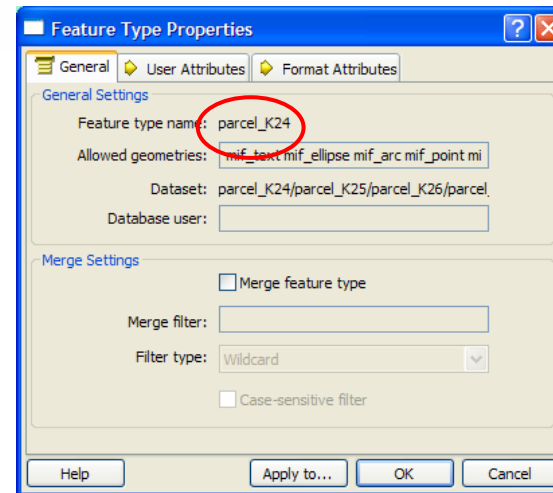
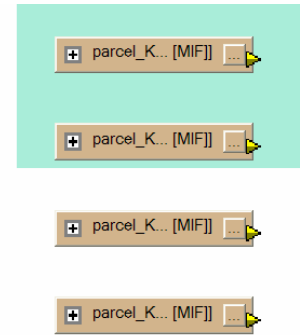
1. File Format (DWG, DGN)

- Because the 'layer' name is the same for all dwg, all the data will enter through the same feature type
- I can now easily route all the data into 1 output
- The **LineJoiner** could be used to merge together all the lines at the tile borders



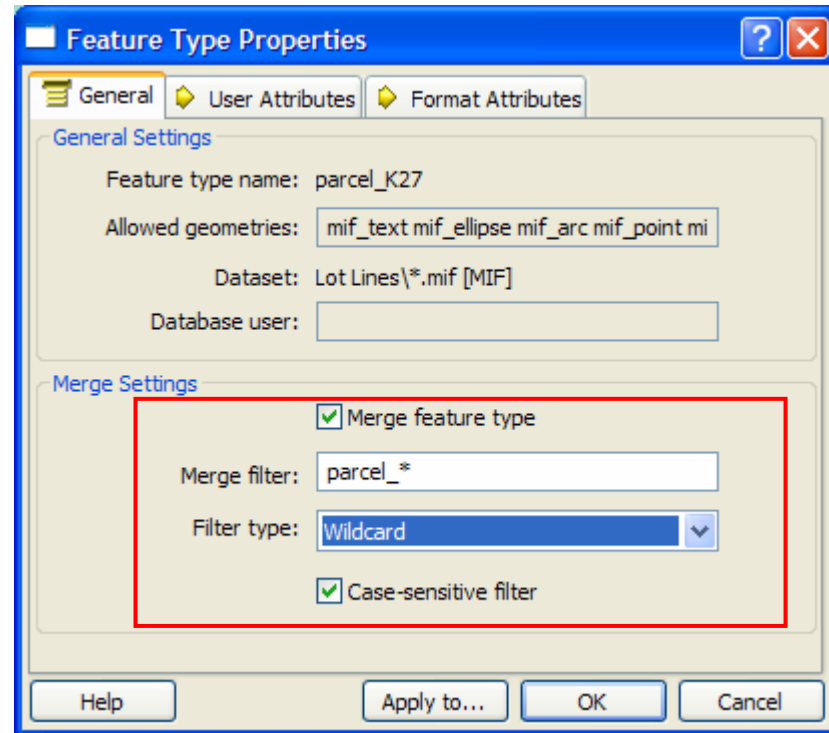
1. Folder-based Formats (SHP, MIF, TAB)

- The names of the feature types are not the same if the name of the mifs are not the same. Each individual mif becomes a feature type in FME (like each layer of a DWG)
- Ex. The mifs with the lot data are each named after the tile that they represent (parcel_K24.mif, parcel_K25.mif, parcel_K26.mif.....)



Folder-based Formats (SHP, MIF, TAB)

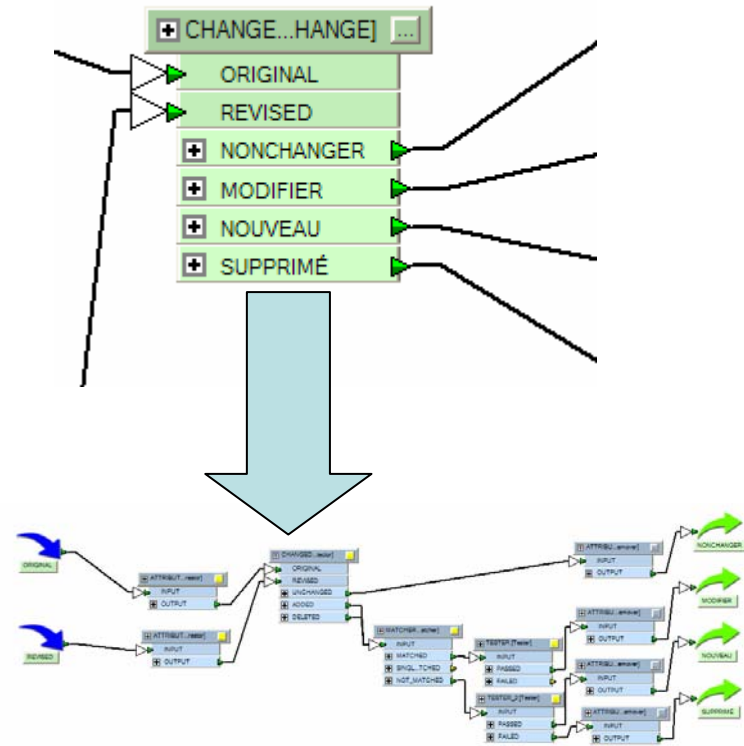
- Solution:
 - Use the **Merge Feature Type** to create a filter which will accept all mifs beginning with `parcl_*` (* is the wildcard)



2. Creating a Custom Transformer

What is a Custom Transformer?

- A transformer you create consisting of a series of transformers
- If we have a series of transformers representing a procedure we want to re-use in other workbenches or use multiple times in one
- Essentially a subroutine




2. Creating a Custom Transformer: Change Detection

1


2-5

6

Original



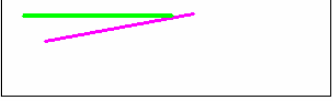
Revised



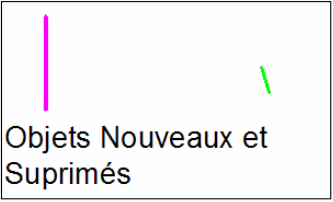
Add an attribut, **_ORIGIN**
_ORIGIN = ORIGINAL
_ORIGIN = REVISED

ChangeDetector
Matcher


Objets Modifiés



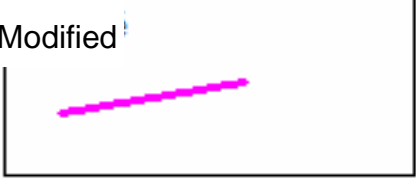
Objets Nouveaux et Suprimés




Unchanged



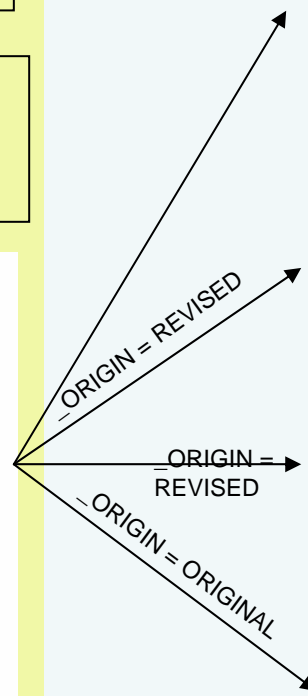
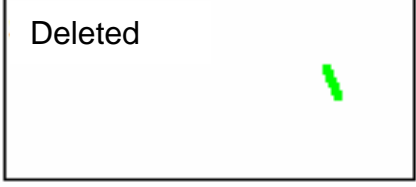
Modified



Added



Deleted

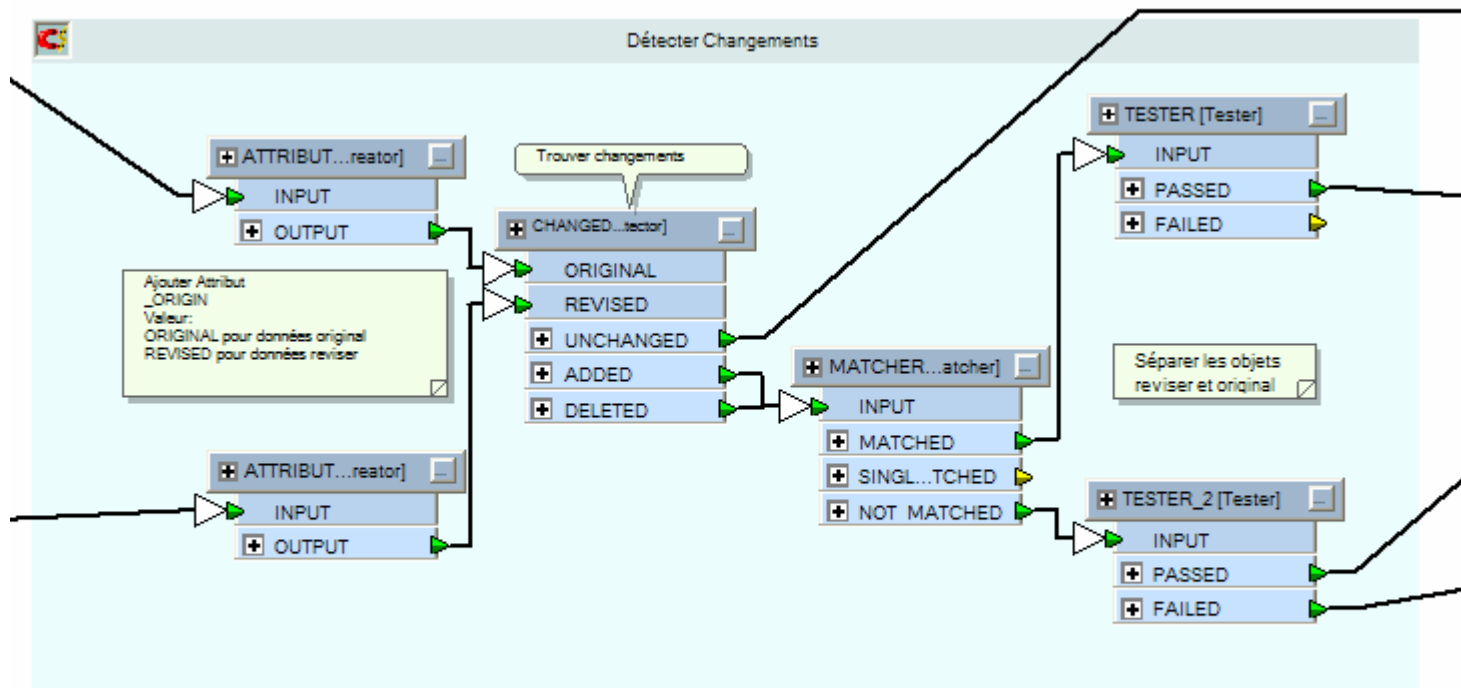


2. Creating a Custom Transformer: Change Detection

1. 2 Sources: Original and Revised
2. Add and attribute: `_ORIGIN`
 - `_ORIGIN = ORIGINAL`
 - `_ORIGIN = REVISED`
3. *ChangeDetector*: Find the changes
 - Unchanged: object exactly the same between the original and revised
 - Added: object in the Revised dataset that did not find an exact match in the Original dataset
 - Deleted: Object in the Original dataset that did not find a match in the Revised
4. *Matcher*: Match on the object's Unique ID between Added and Deleted
 - If the ID is found in both, it was a modified object
 - If the ID is not found in both, it is either a new or deleted object
5. Use the `_ORIGIN` attribute to separate the added versus deleted objects
6. 4 Results: Unchanged, Modified, Added, Deleted

2. Creating a Custom Transformer

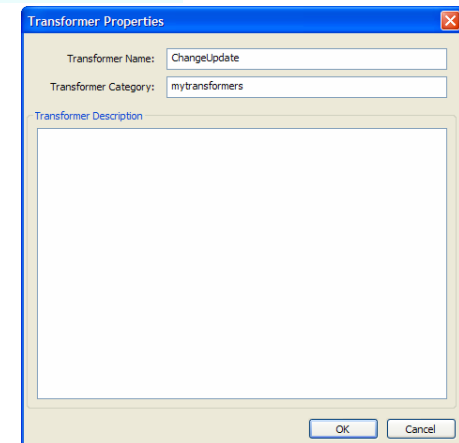
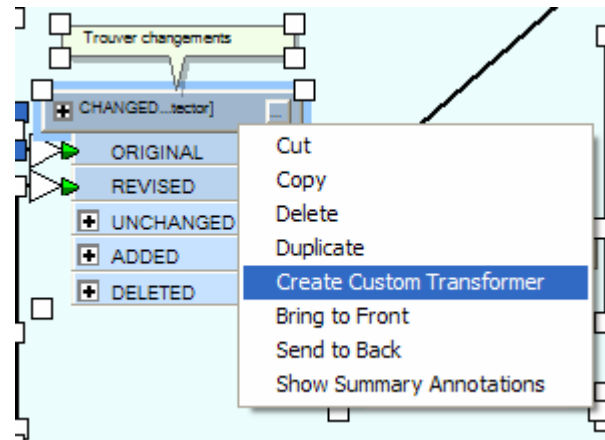
- Steps:
 1. Identify the process you want to convert into a transformer
 - In this case my procedure to determine the changes between datasets



2. Creating a Custom Transformer

2. Select the process (transformers) and convert to a custom transformer

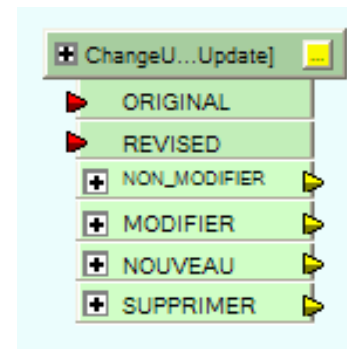
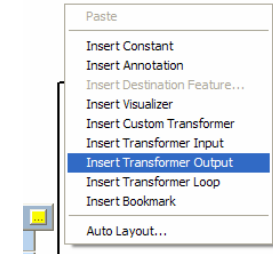
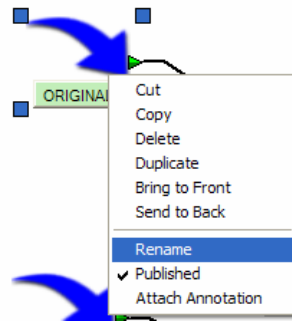
- Right-click > **Create Custom Transformer**
- Enter a **name** and **category** for the transformer



2. Creating a Custom Transformer

3. Modifie the inputs and outputs

- Inputs
- Outputs
- Review in the 'Main' tab

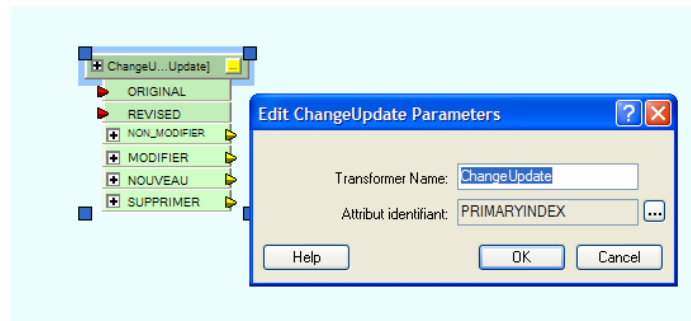
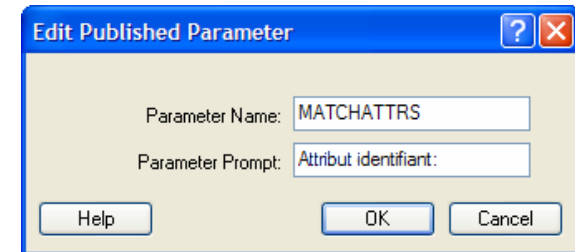
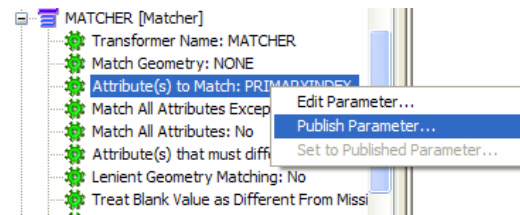


Custom Transformer in the *Main* tab

2. Creating a Custom Transformer

4. Publish parameters

- Renders the transformer more general
- For example the key attribute in my dataset was called PRIMARYINDEX. I use this in the Matcher transformer. If i want to use this tool on other datasets, my key field may not be called PRIMARYINDEX
- If i publish the parameter, my user can now change the key field anytime they use the transformer.

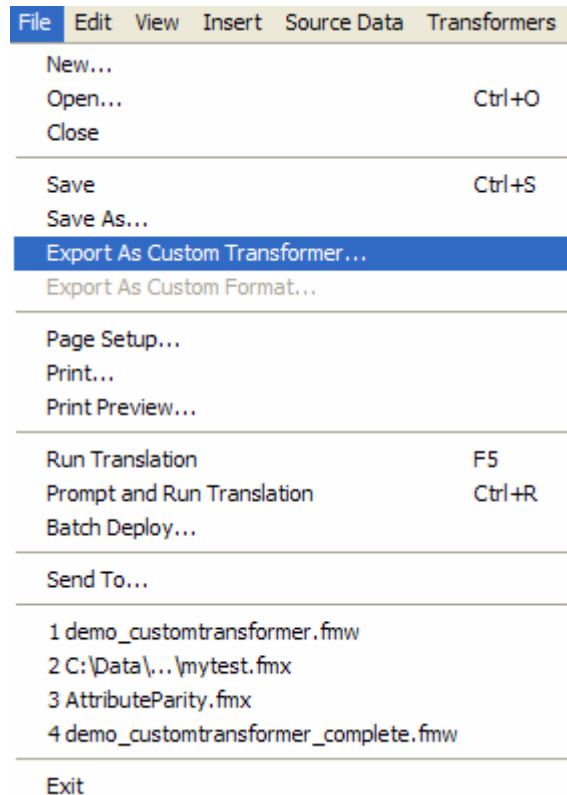


Published parameters

2. Creating a Custom Transformer

5. Export the Custom Transformer

- FME places it in:
 - ...\\My Documents\FME\Transformers\
 - Extension .fmx



2. Creating a Custom Transformer

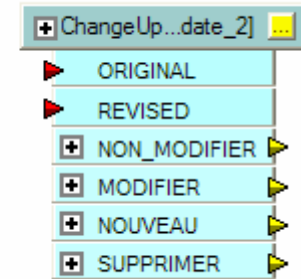
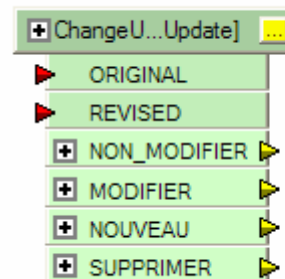
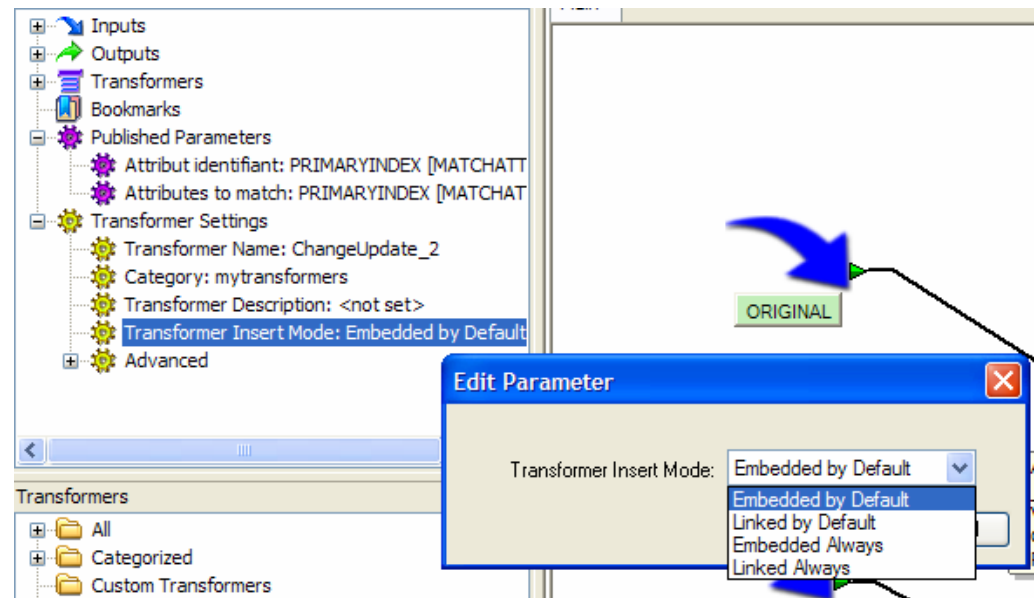
6. Specify whether the transformer should be **embedded** or **linked** in the Workbench

Embedded: The definition of the transformer is copied into the Workbench using it.

Transformer is green

Linked: The workbench uses the definition in the FMX file directly. If you send someone the workbench you must also provide the transformer.

Transformer is pale blue

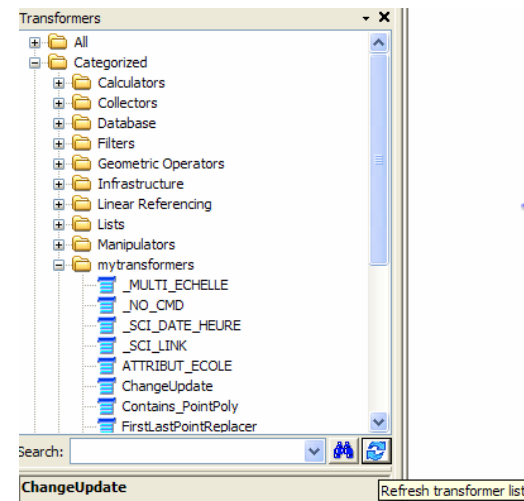
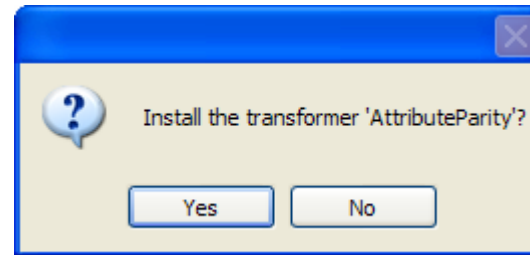


2. Creating a Custom Transformer

7. Install on users' computers

- 1) Provide the user with the fmx file and double click. The workbench will automatically copy this to the \My Documents\FME\Transformers\ directory or 2) copy the definition to a shared resource

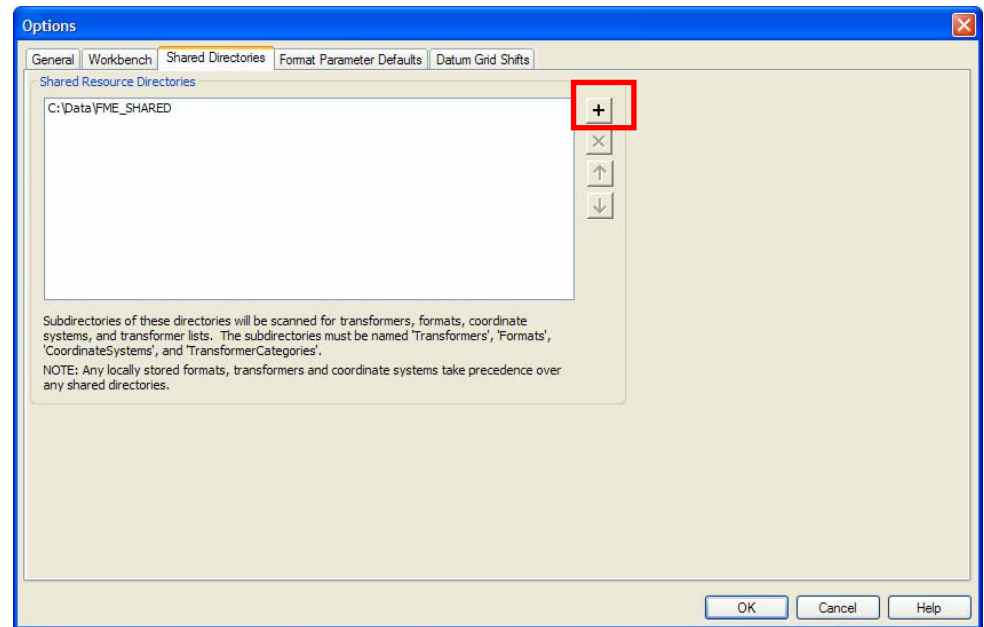
Refresh the transformer list.



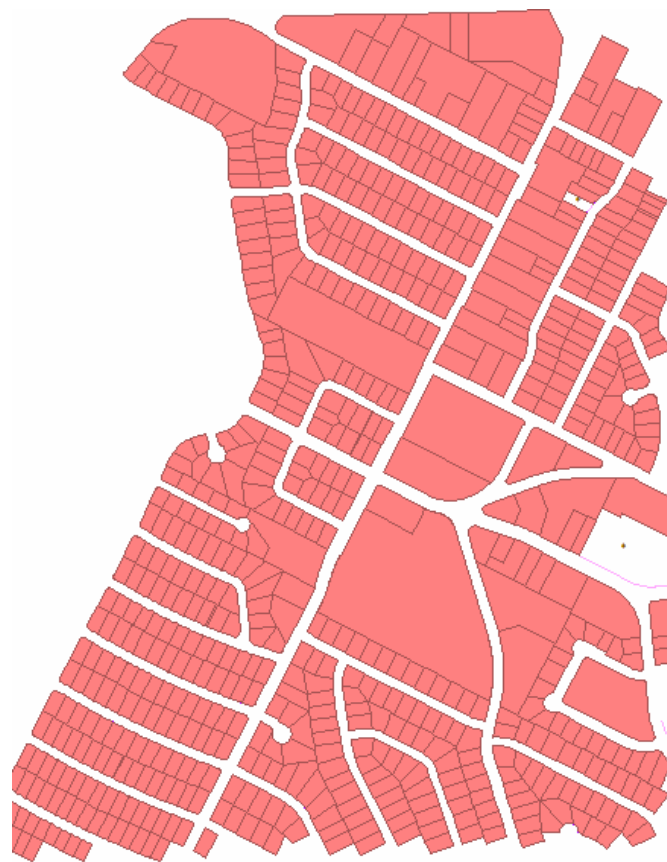
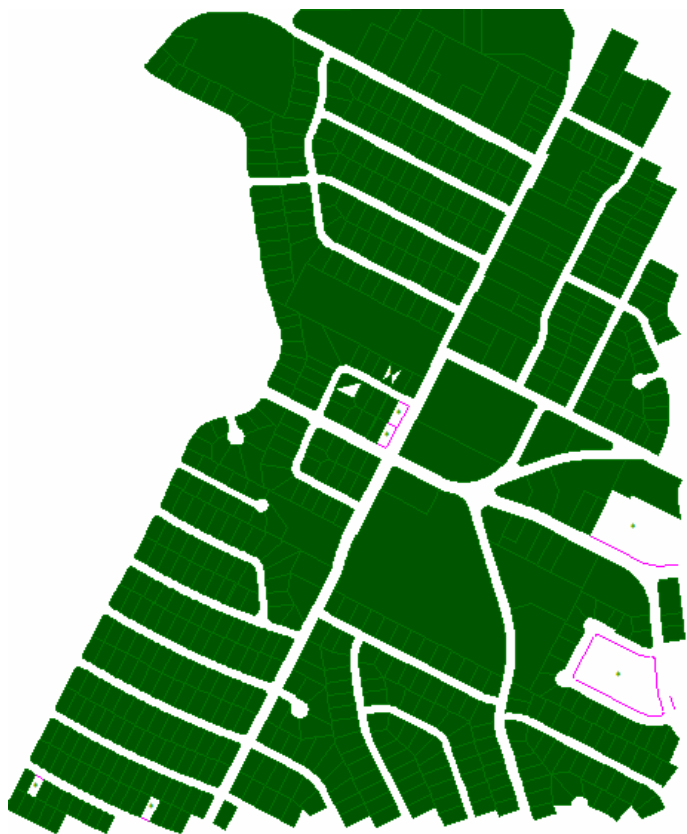
2. Creating a Custom Transformer

How to share a resource

1. In Windows Explorer create a directory
2. In the Workbench
 1. Tools > Options
 2. Add the directory with the + button (ex c:\myfme)
3. Copy the Custom Transformers to the \Transformers\ folder

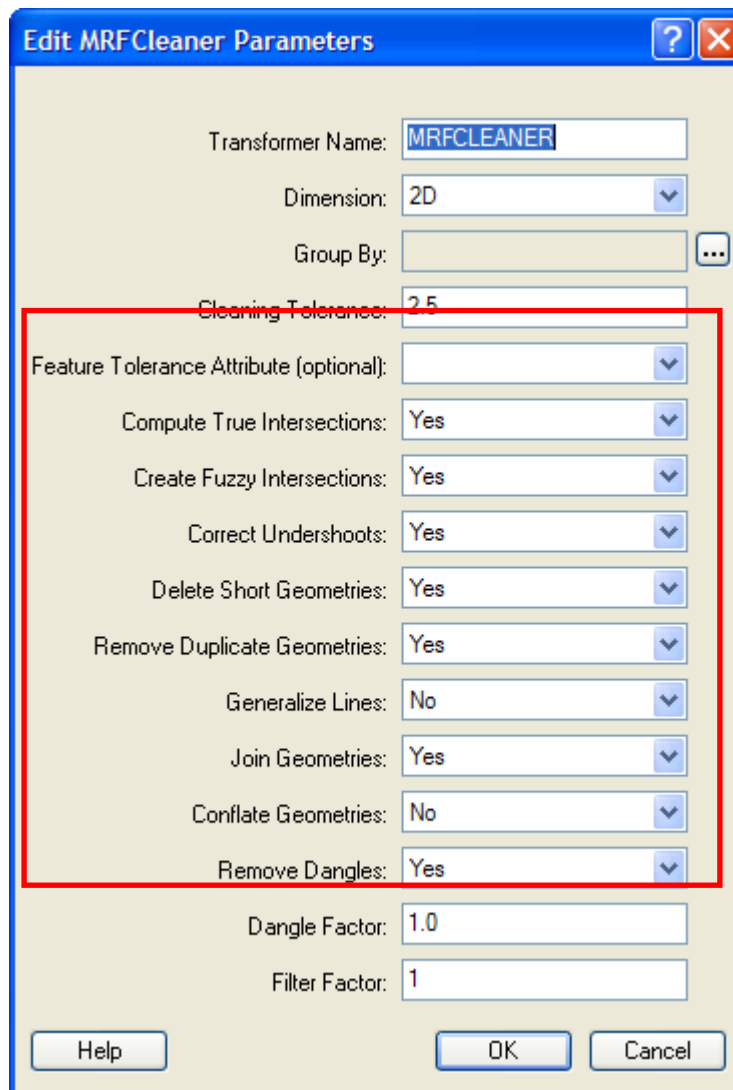


3. Cleaning topological errors with MRFCleaner



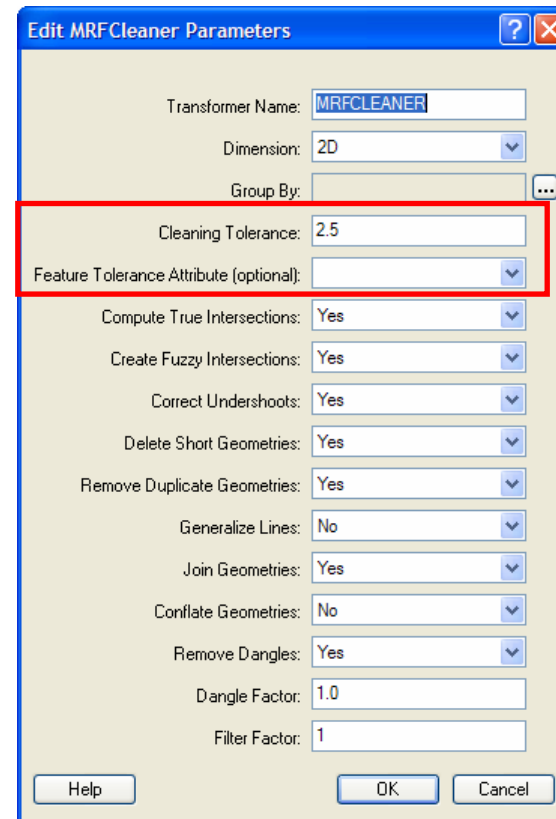
3. Cleaning topological errors with *MRF Cleaner*

- 1. Choose the cleaning operations you want to perform in 1 transformer: the *MRF Cleaner*



3. *Cleaning topological errors with MRFCleaner*

- Choose a global cleaning tolerance for all the objects or choose an attribute
 - Larger tolerance objects will move toward smaller tolerance objects



3. *Cleaning topological errors with MRFCleaner*

- To manipulate the order of operations, you could chain several *MRFCleaner* transformers
- *MRFCleaner will convert polygons to lines*
 - Polygons would have to be rebuilt

3. *Cleaning topological errors with MRFCleaner*

- To get a better understanding of each of the cleaning functions and the order that the steps are taken, see the the following sections under the help for this transformer: [MRFCleaner Modules](#) et [default workflow](#)

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](#)