

Design Readable Workspaces	2
Use Connections Effectively	7
Annotate Your Workspaces	13
Organize Workspaces with Bookmarks	16
Improve Workspaces with Bookmarks	19
Exercise_ Improving Workspace Style	25
Use Prototyping and Incremental Development	29
Exercise_ Workspace Prototyping	34
Understand Workspace Structure	40
Create Workspaces With Multiple Readers and Writers	43
Exercise_ Use Multiple Readers and Writers	52
Test Your Workspace With Partial Runs	59
Exercise_ Use Partial Runs To Test Your Workspace	64
Collaborate on Data Integration Workflows	73
Compare and Merge Workspaces	76
Exercise_ Compare and Merge Workspaces	85

Design Readable Workspaces

Learning Objectives

After completing this unit, you'll be able to:

- Explain why style is important to consider when designing workspaces.
- Compare different styles for object layout.
- Use grids and guides to assist object layout.
- Use Autolayout to organize your workspace quickly.

Best Practice

If a workspace runs to completion and produces the output you want, it can't be wrong, can it? Well, yes, it can. It's not enough just to put together a functioning workspace; it's also vital to use FME in an efficient and scalable manner.

In general terms, best practice means the best way of doing something; in other words, carrying out a task effectively and efficiently.

Despite the word 'best,' we're not presuming the ideas here will meet every need and occasion. The best description of this concept I've heard – and one that fits well here – is:

A very good practice to consider in this situation based on past experience and analysis.

Why Use Best Practice?

Best practice in FME can help a user to...

- Debug a workspace when it doesn't work the way intended
- Use FME in a project-based environment
- Create workspaces that use the correct functionality in the proper place
- Create high-performance workspaces
- Use FME Workbench in a more efficient way
- Create well-styled workspaces that are self-documenting



I learned about best practice the hard way when I had to work on someone else's workspaces. My colleague had organized the workspaces so poorly that the whole operation took me three times as long as it should have!

In this module, we'll provide a guide to the preferred design for workspaces. The correct style makes a workspace easier to interpret, particularly in the long run when the author might return to it after a period of inactivity.

An FME Workspace Style Guide

“A good-looking, well-organized workspace gives the customer the feeling that you have done quality work.”

Style is perhaps the most obvious component of FME Best Practice. You can tell at a glance when a workspace is well-styled and when it is not. As the quote above implies, a well-designed workspace demonstrates competence.

But style is more than looks; a properly designed workspace provides many benefits as it is further developed and edited in the future.

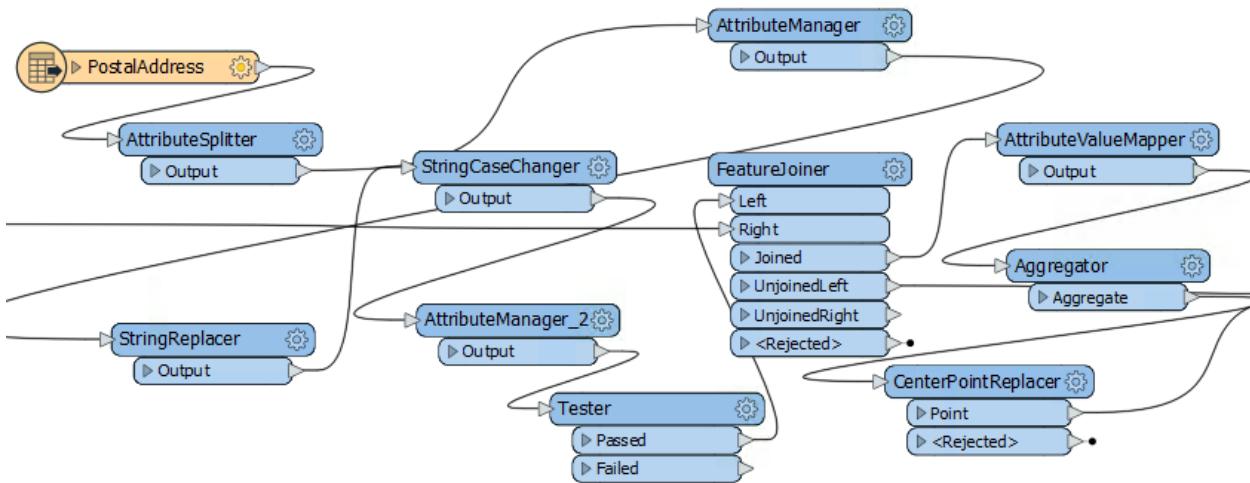
Good style makes it easier to navigate and understand an existing workspace. Design is essential when workspaces might need to be edited by other users or when you intend to make edits yourself at a later date.

Specifically, a good style can help a user to...

- Distinctly define different sections or components of a workspace
- Quickly navigate to a specified section or particular transformer
- Pass a workspace on to another user for editing
- Rename workspaces and content with a more explanatory title

Example of Poor Design

Do you need proof? Well, would you want to be given the task of editing this workspace? Can you even tell what this section does or - more importantly - why?



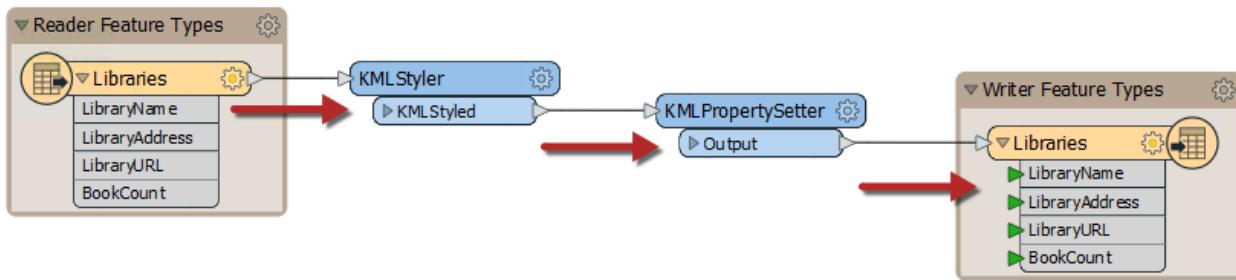
You should always use Best Practice, whether it's a small workspace, training exercise, or a large-scale project. Getting into the habit helps make your smaller projects scalable.

If you don't design a workspace well from the very start, it will become harder to make edits as you work on it.

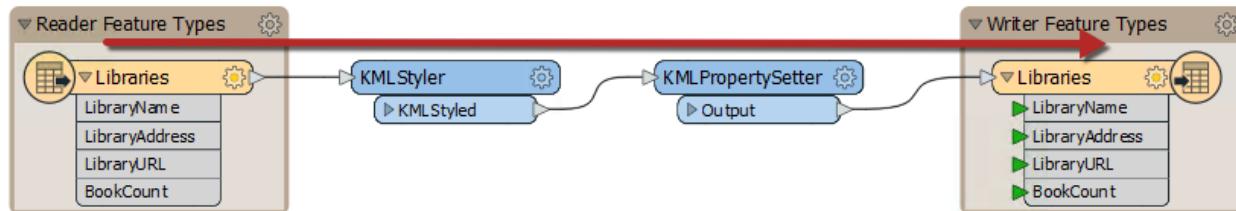
Object Layout

The positioning of workspace objects and the care taken in connecting them can make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

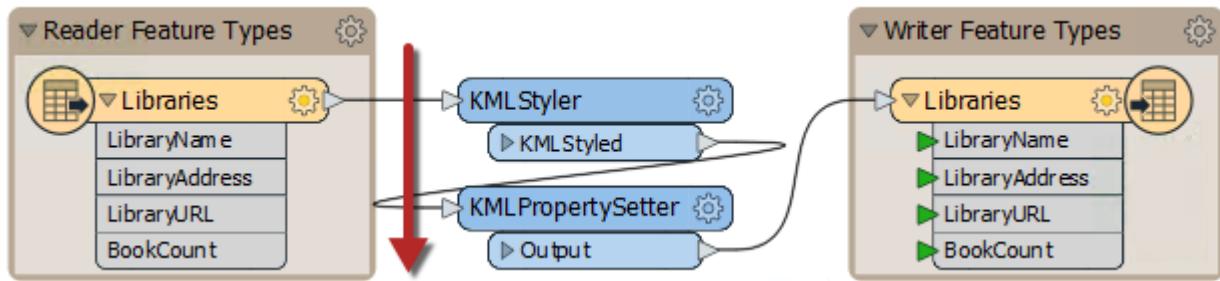
Layout methods vary from user to user. Some users like to line up objects so that all connections are horizontal:



Others prefer the tops of objects to be aligned horizontally, with angled connections:



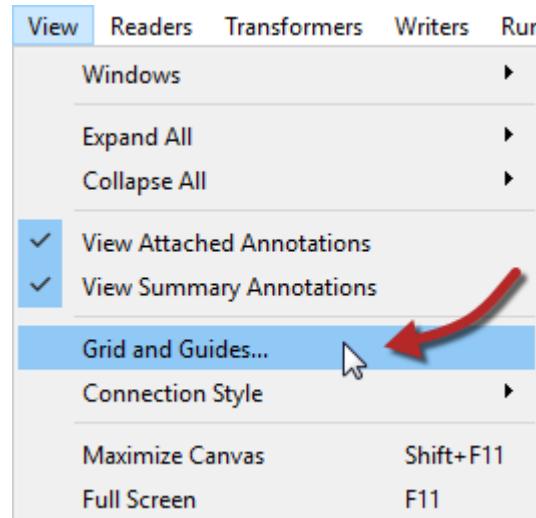
Some prefer to align object edges vertically:



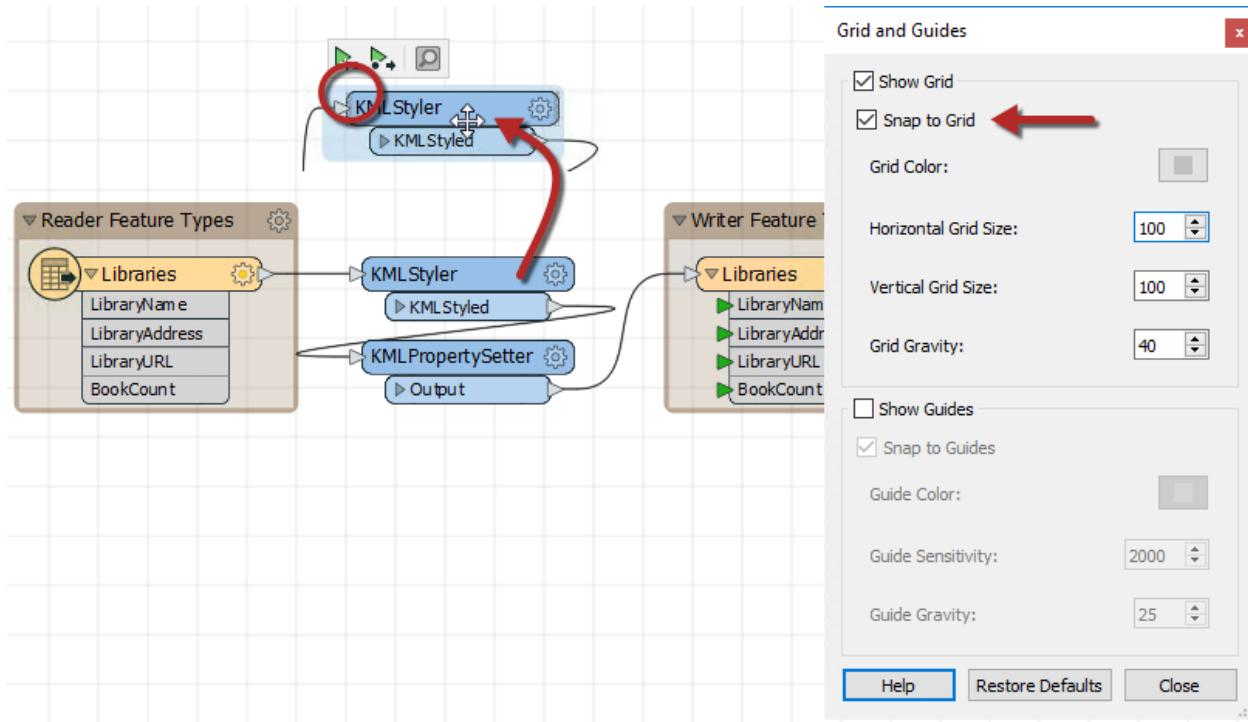
The style used is more of a personal preference than a definite rule, but consistency is important. A workspace with no apparent layout style, or an inconsistent one, does not inspire confidence in the author's abilities!

Grid and Guides

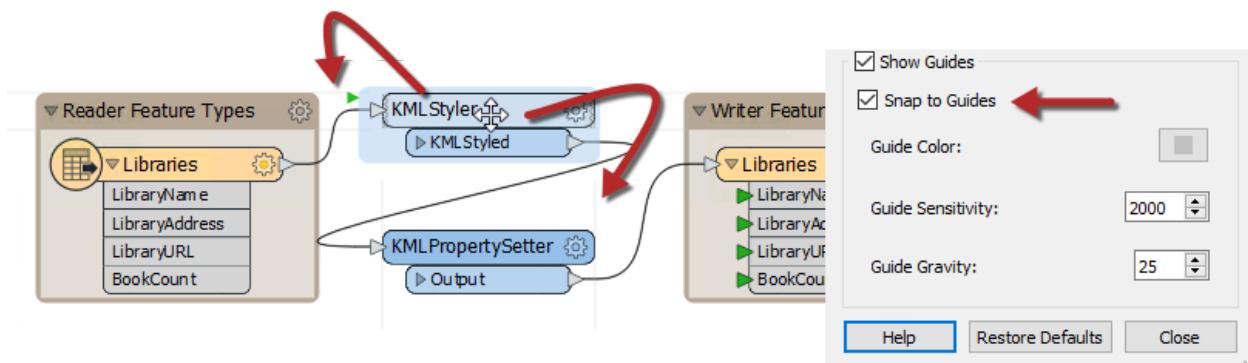
Grids and Guides are a tool to help align workspace objects neatly and tidily. You can access this functionality through View > Grid and Guides on the Workbench menu bar.



Show Grid displays a grid of lines on the Workbench canvas. Snap to Grid causes all objects – such as the KMLStyler highlighted – to snap onto the intersection of grid lines when moved. In this way, you can more easily line up objects.



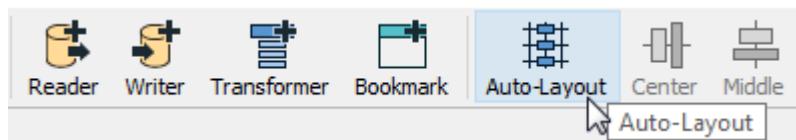
Show Guides causes guidelines to be displayed on the Workbench canvas whenever an object is moved and lines up approximately to another canvas object. Snap to Guides snaps an object onto a highlighted guideline. Guides are enabled by default.



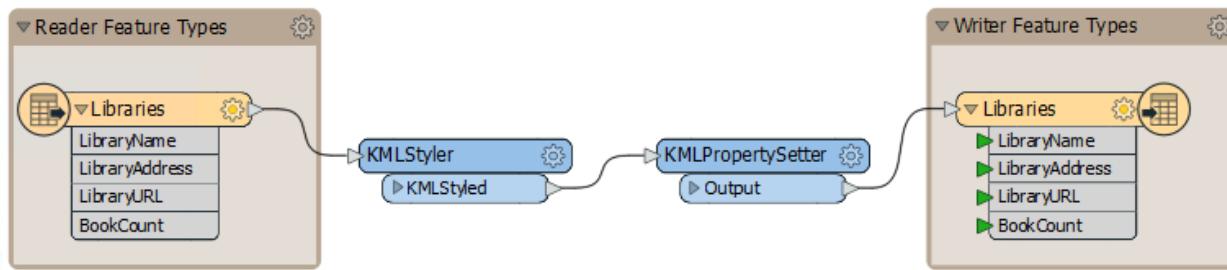
These two tools make it very simple to align workspace objects in a pleasing style.

Autolayout

The Autolayout tool appears on the toolbar of FME Workbench:



Clicking the toolbar button will layout either all of the workspace or just objects that are currently selected:



As you can see, Autolayout tends to use a horizontal pattern, with the tops of objects aligned. Therefore it's better to select groups of transformers and run the tool on them, rather than trying to lay out the entire workspace in a single action.

1 Which of the following is not a feature of a workspace using good style?

- A. Clearly defined sections or components of a workspace
- B. Quick navigation to a specified section or particular transformer
- C. Very colorful
- D. Easily passed to another user for editing
- E. Explanatory titles so the user knows what the workspace does

2 All FME users must align the tops of objects on the canvas.

- A. True
- B. False

3 Both the grid and guides are enabled by default.

- A. True
- B. False

4 Autolayout might not perfectly layout your workspace every time, but it can help you get started with tidying up your workspace.

- A. True
- B. False

Check the Quiz to Earn 150 Points

Use Connections Effectively

Learning Objectives

After completing this unit, you'll be able to:

- Understand the different connection style options.
- Explain why you should avoid overlapping connections.
- Change port order to avoid overlapping connections.
- Use junctions, hidden connections, and tunnels to tidy larger workspaces.

Connection Styles

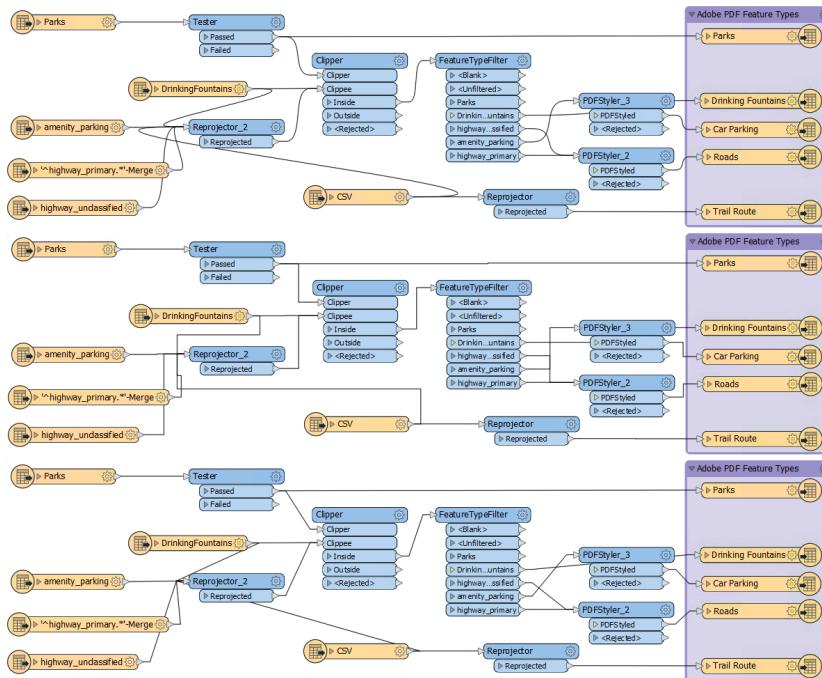
Object positioning is only part of a good layout. The other key part is the connection style.

As with the positioning of workspace objects, the care taken in connecting them can make the difference between a poorly-designed workspace and one that is visually attractive and efficient.

Connections are the lines between objects on the workspace canvas. There are three different styles of connection that you can create in Workbench:

- Curved: The default style with curved line connections
- Squared: A style that evokes a Manhattan skyline through squared connections
- Straight: The original connection style; a straight line between two objects

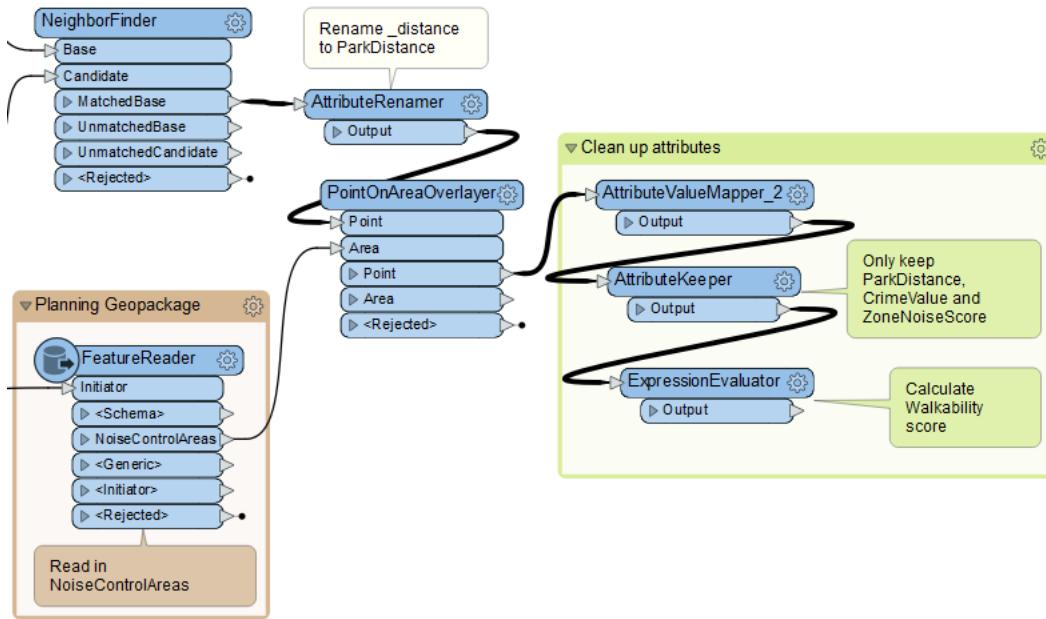
You can switch between styles in the FME Options menu (Tools > FME Options > Appearance > Canvas > Connection Path), or the shortcut Ctrl+Shift+C. This image shows a comparison of the three styles:



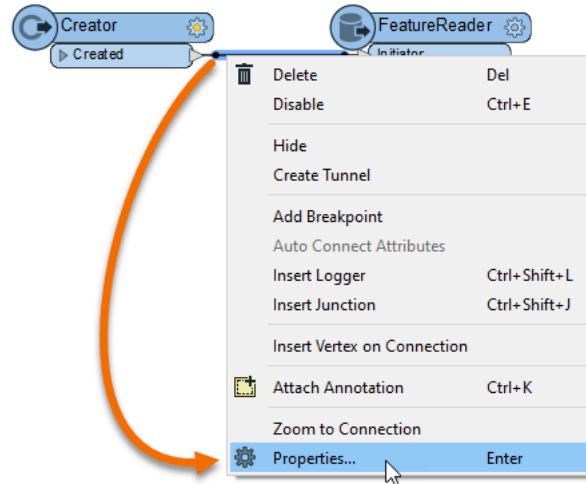
Once more, there is no right or wrong choice about which style to use; it is a personal preference. However, object layout and connection style are related; the best FME authors will vary the objects' position according to the connection style used to avoid issues like overlapping connections.

Custom Connection Styles

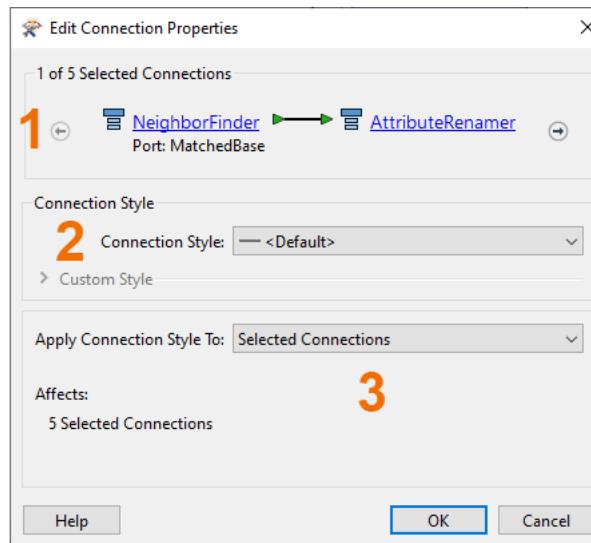
You can also use custom connection styles to highlight certain connections in your workspace and give you more control over the appearance of your connections. For example, you can color-code connections based on what data they contain or change connection thickness based on estimated data volume.



To change a connection style, select a connection line by clicking it (or Ctrl/Cmd clicking multiple lines). Then right-click a selected line and choose Properties...



From here, you can change connection styles.



1. If you have multiple connections selected, you can click the left and right arrows to change the selected connection.
2. You can click this drop-down to apply an existing connection style or create a custom one.
3. You can choose to apply the selected connection style to just this connection, all, or matching connections up and downstream.

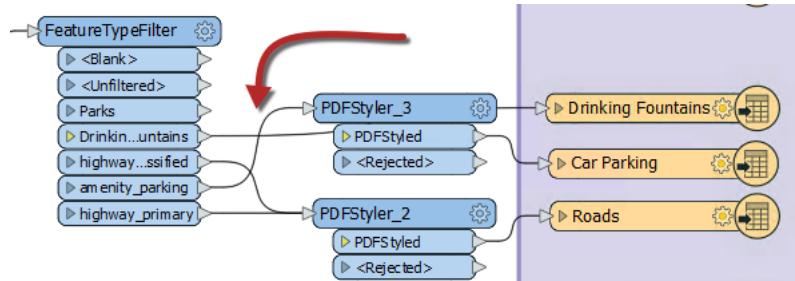
Use custom connection styles to make it easier to trace particular features through your workspace. You can use this for yourself while debugging or for others to make workspaces more readable. Just ensure you include some kind of legend or explanation for your custom styles, for example, in annotation or workspace documentation.



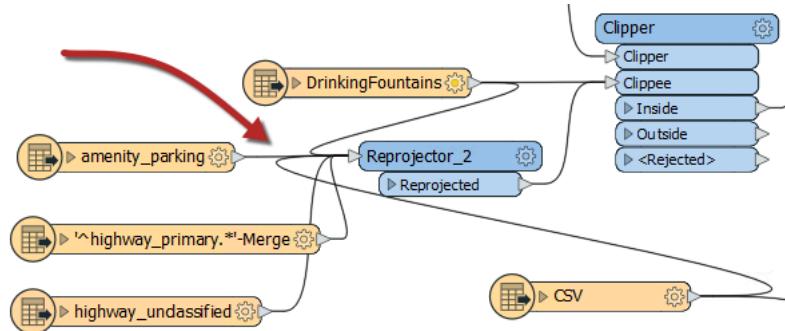
[Learn more](#) about custom connection styles.

Overlapping Connections

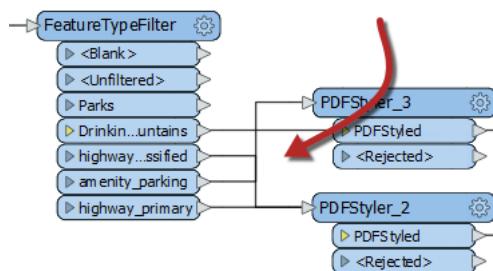
One of the most conspicuous failings of a workspace design is to have connections that cross over each other, for example like this:



The intent of a connection is compromised when it overlaps with another connection or object on the canvas. However, the choice of connection style affects the possibility of overlap occurring. For example, curved connections tend to cross over more than straight ones:



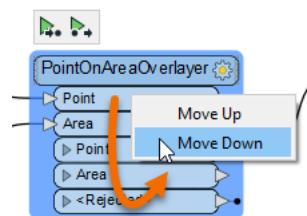
...and squared connections can sometimes cross in ways that are difficult to decipher:



Because these issues can spring up when you switch connection styles, it's wise to choose a particular connection style and layout technique and stick with it. For example, transformers could be spaced more widely in a curved connection workspace to avoid overlaps.

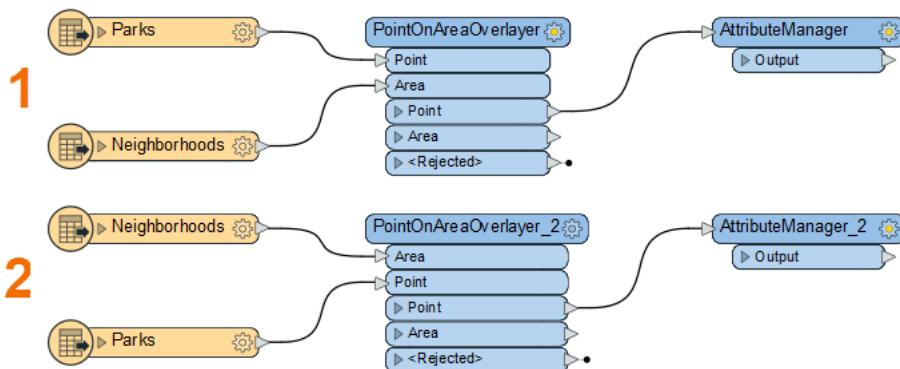
Changing Port Order

You can change port order to avoid overlapping connections by right-clicking a port and choosing Move Up or Move Down.



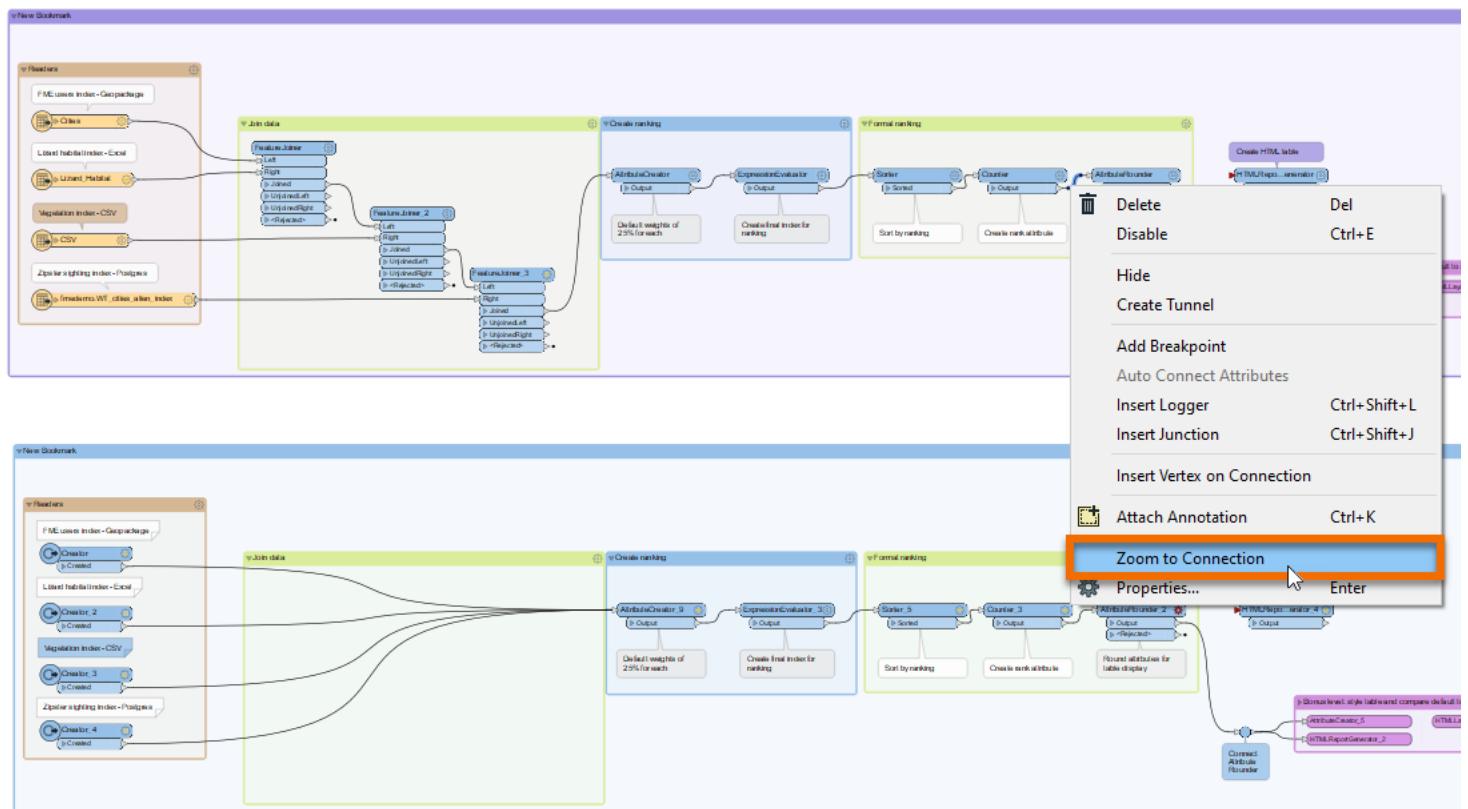
However, beware: this might throw off experienced users who expect the default port order. Laying out the workspace to avoid overlaps in the first place is ideal if you can manage it.

Compare these two options. While both manage to avoid overlapping connections, layout #1 on the top is better because it uses the default port order:



Zoom to Connection

If you work with large workspaces, you'll often find yourself scanning the zoomed-out workspace looking for a particular section. Once you find a section you are interested in, you can right-click a connection you'd like to zoom to, then click **Zoom to Connection**.

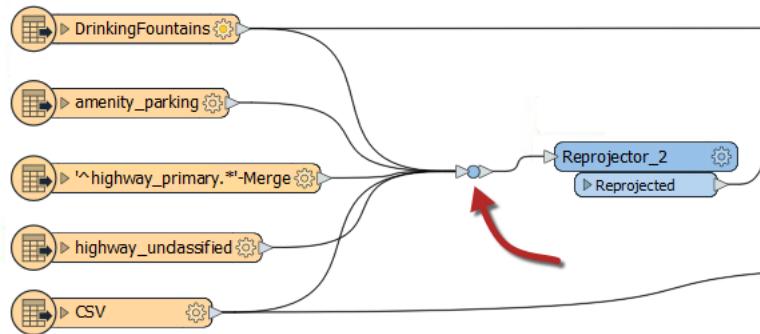


You can do this with transformers and feature types, too.



Junctions

One transformer in FME Workbench enhances the layout of objects and connections: the Junction.

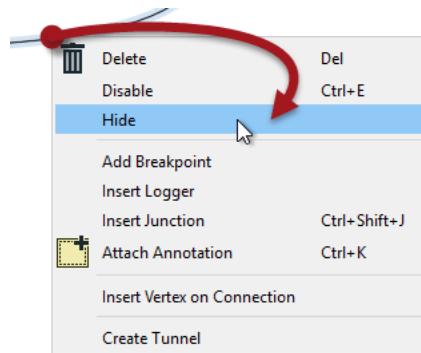


This transformer is a small, node-like object that carries out no function on the data but is used to tidy connections within a workspace - as in the above screenshot. This trait makes it an excellent tool for best practice.

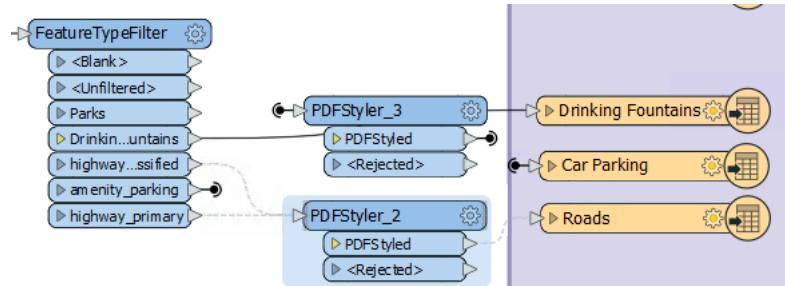
As with any other transformer, you can connect a junction to an Inspector or Logger and attach annotation objects to it. It also works with Quick Add, Drag/Connect functionality, and Feature Caching.

Hidden Connections and Tunnels

The ability to hide connections is handy for avoiding overlaps. To hide a connection, right-click on it and choose the option to Hide:



A hidden connection is represented by a 'transmitter' icon or by a greyed-out dashed line when the object at one end of the connection is selected:



You must select the object (transformer or feature type) for the connection to be visible. It is best to hide connections you consider less important and don't want the user to see by default.

The other available option is "Create Tunnel." This choice creates a hidden connection with the addition of an annotated junction transformer at each end:



A tunnel makes a hidden connection slightly more apparent, plus allows for annotation at each end. Tunnels help make connections across a large workspace without requiring the user to follow a long connection line.

Revisualizing Hidden Connections

To view hidden connections, click on an object at either end. The connection appears as a greyed-out dashed line.

To return a connection to view, right-click an object to which it is connected and choose Show Connection(s).

For more information on Tunnels and Junctions, see [this blog post](#).



1 Which of the following is not a connection style option in FME?

- A.Curved
- B.Elliptical
- C.Squared
- D.Straight

2 Which of these techniques should you not use to avoid overlapping connections?

- A.Changing port order
- B.Creating many vertices in your connection lines to draw around objects
- C.Using hidden connections
- D.Using a tunnel

3 Which of these features should you use for connections you want to deemphasize unless the user is specifically interested in them?

- A.Annotation
- B.Bookmark
- C.Hidden connection
- D.Junction
- E.Tunnel

Check the Quiz to Earn 100 Points

Annotate Your Workspaces

Learning Objectives

After completing this unit, you'll be able to:

- Explain the benefits of using annotation.
- Distinguish between user and summary annotation.
- Use annotation to explain what your workspace does and why.

Annotating Workspaces

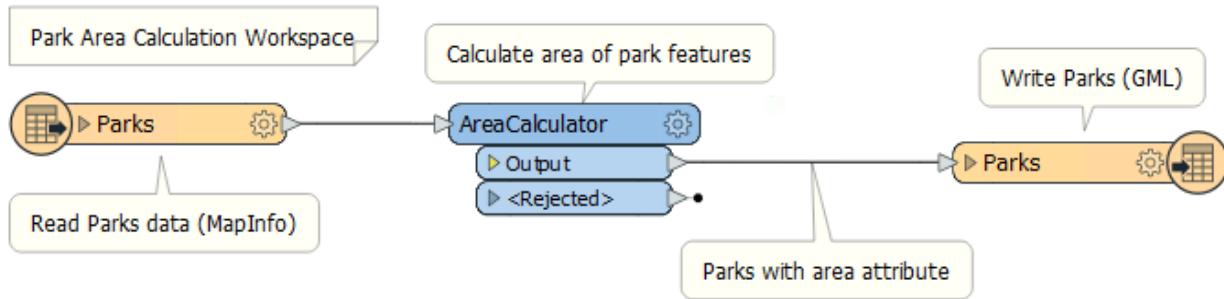
Annotation is a crucial method for a clear and understandable design.

Annotation helps other users understand what is supposed to be happening in the translation and helps the creator when returning to a workspace after a long interval (take it from me that this is especially important!)

You can apply two different types of annotation to a workspace.

User Annotation

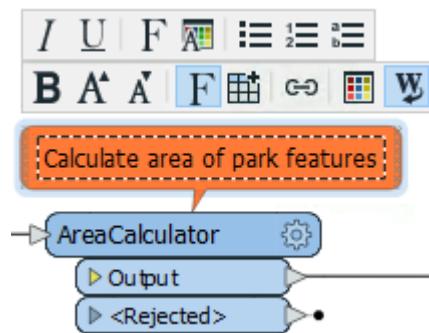
User annotation is a comment created by the user. It can be connected to a workspace object (transformer or feature type), connected to a workspace connection, or can float freely within the workspace.



To create attached user annotation, right-click a workspace object and select Add Annotation, or use the shortcut Ctrl+K when the object is selected.

To create floating user annotation, right-click the canvas, select Insert Annotation, or press Ctrl+K when nothing is selected.

When you place an annotation, you have the opportunity to change the font style, font size, and background color; plus, you can also add hyperlinks, bullet points, and tables.

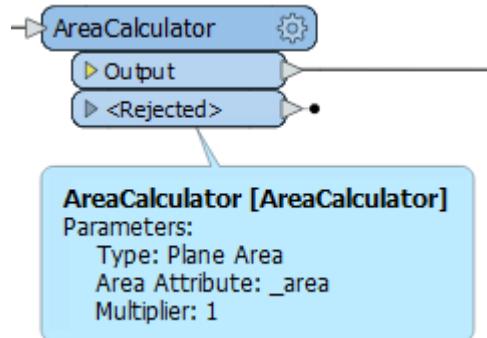


You might be asking: "Why would I use annotation when I can just rename my transformers to make their function more explicit?" This is a good question! Some FME users like to rename transformers to describe what they are doing. However, we've seen some problems with this approach. First, it's hard to fit enough description in a short name. Second, other users (or yourself in the future) will have no idea what the original transformer was and will likely have to open it to see how it is configured. In the end, we find this method doesn't save much time, but you can decide for yourself.

Summary Annotation

Summary annotation is an FME-generated comment that provides information about any object within the workspace. This item can be a source or destination feature type or a transformer.

Summary annotation is always colored blue to distinguish it from other annotation. It's always connected to the item to which it relates and cannot be detached.

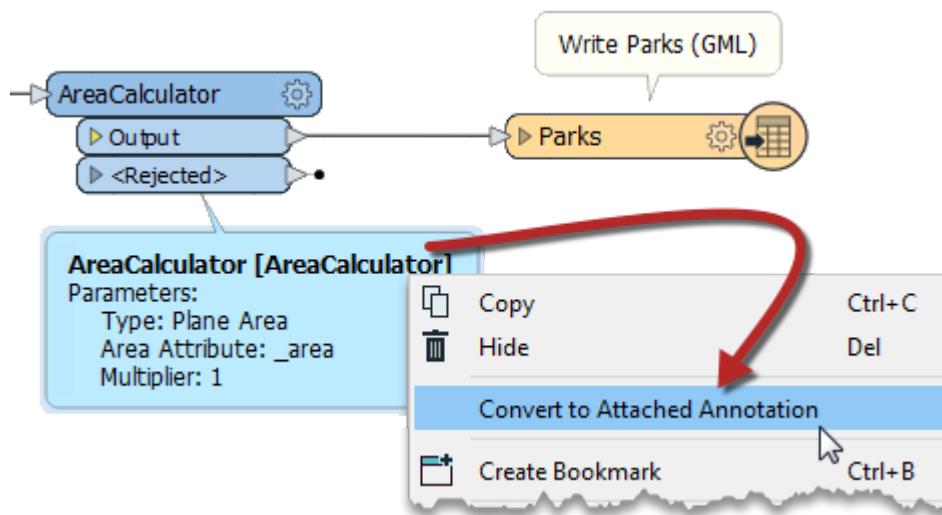


The nice thing about Summary Annotation is that it automatically updates in response to changes. That makes it very useful for checking transformer parameters (or reader/writer schemas) at a glance. It's beneficial in situations where the parameters are set through a wizard and are more awkward to check (for example, the SchemaMapper or FMEServerJobSubmitter transformers).

Which Annotation Type Should I Use?

It's a good idea to use summary annotation to show what actions you are taking and user annotation to clarify why you are taking these actions. A good example is when you are not using default values for a transformer or feature type.

You can convert a summary annotation to a user annotation by using this context menu option:



This method allows you to extract the information from a summary annotation but edit it as you would a user annotation. Note, however, that a converted summary annotation no longer updates automatically!

1 Annotating your workspace makes it easier for others (or you at a later date) to understand what the workspaces does and why.

- A.True
- B.False

2 Summary annotation should be attached to every transformer in your workspace.

- A.True
- B.False

3 User annotation should ideally be used to explain why you designed your workspace in a particular way.

- A.True
- B.False

Check the Quiz to Earn 100 Points

Organize Workspaces with Bookmarks

Learning Objectives

After completing this unit, you'll be able to:

- Identify why you should use bookmarks to divide your workspace into sections.
- Add a bookmark to your workspace.
- Resize and edit a bookmark.
- Edit bookmark properties.

Why Use Bookmarks?

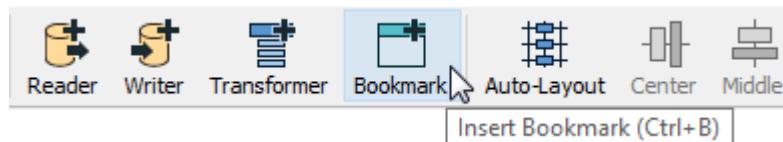
Like its real-world namesake, a bookmark is a means of putting a marker down for easy access.

With FME, the bookmark covers an area of the workspace that usually carries out a specific task, so a user can pick it out of a broader set of transformers and move to it with relative ease.

Bookmarks make your workspace easier to understand and navigate.

Adding a Bookmark

To add a bookmark, click the Bookmark icon on the toolbar.



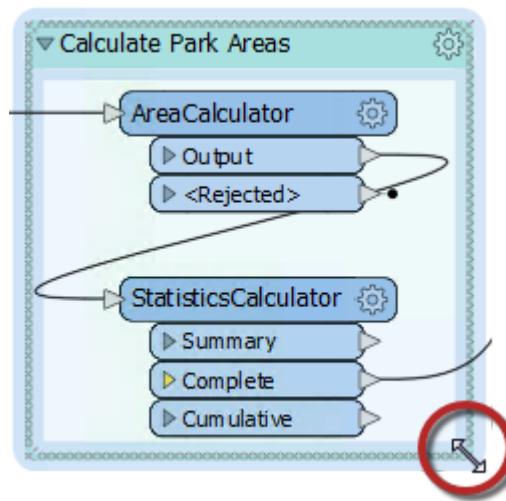
Whereas a traditional bookmark marks just a single page in a book, the FME bookmark can cover a wide area of the canvas. You can divide a single workspace into different sections by applying multiple bookmarks.



If you create a bookmark while objects are selected, the bookmark automatically expands to include those items.

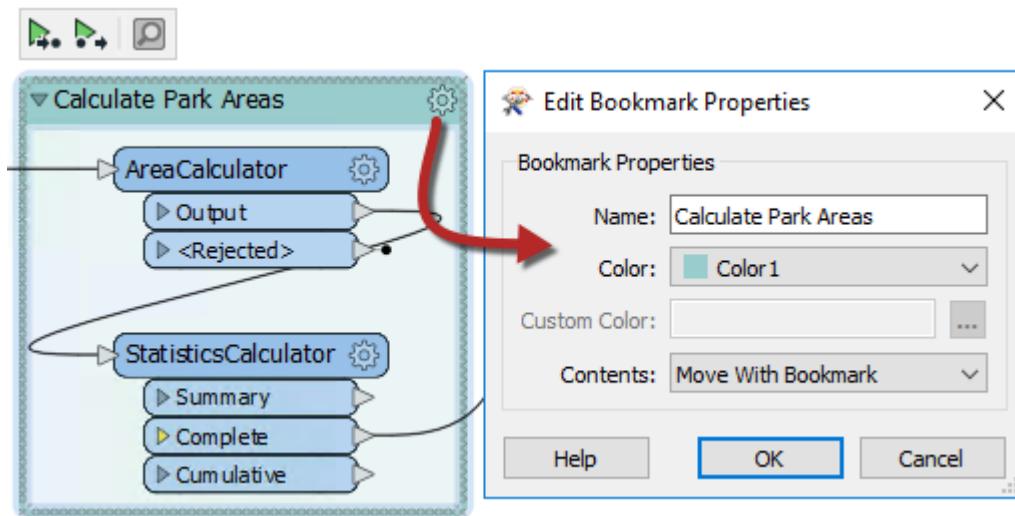
Resizing and Editing a Bookmark

To resize a bookmark, hover over a corner or edge and then drag the cursor to change the bookmark size or shape.



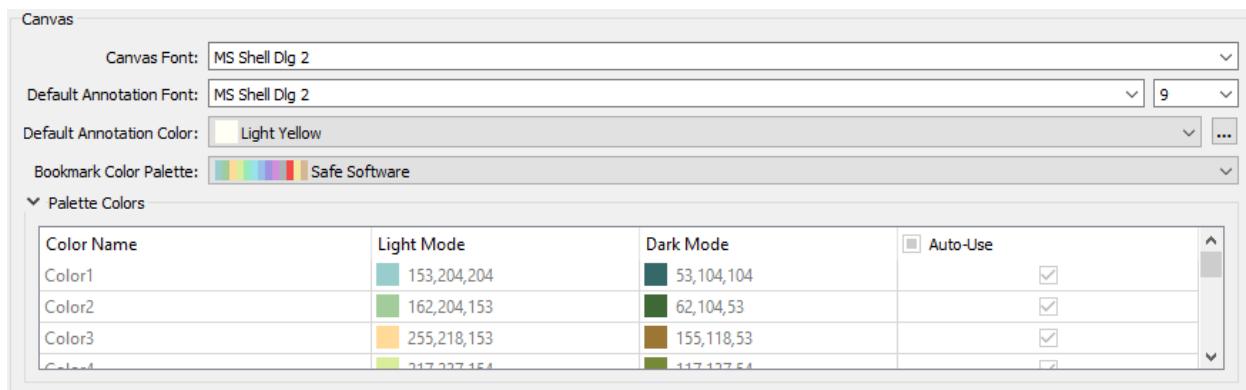
Bookmark Properties

Click the cogwheel icon on a bookmark header to open the bookmark properties dialog:

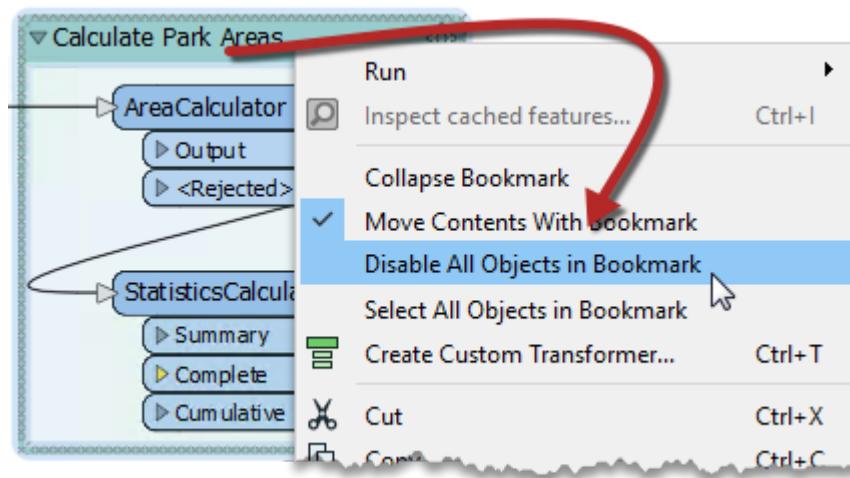


Here you can change both the name and color of the bookmark and decide whether contents will move with it (more on that later).

You can set bookmark colors to an existing color palette or use custom colors. Additionally, you can create customized palettes by going to Tools > FME Options... > Appearance:



The context (right-click) menu for a bookmark reveals options to select all objects within the bookmark or to disable all of those objects, making it useful for testing purposes:



1 Adding a bookmark with multiple objects selected will ensure the new bookmark contains the selected objects.

- A.True
- B.False

2 Bookmarks cannot have their color changed after they are added.

- A.True
- B.False

3 Bookmarks can be placed or “nested” inside one another.

- A.True
- B.False

4 Clicking a bookmark in the Navigator will take you to that section of the workspace.

- A.True
- B.False

5 Bookmarks can be used to make moving large sections of workspaces easier.

- A.True
- B.False

Check the Quiz to Earn 50 Points

Improve Workspaces with Bookmarks

Learning Objectives

After completing this unit, you'll be able to:

- Use bookmarks to plan workspace design.
- Use bookmarks to navigate and edit your workspace quickly.
- Use bookmarks to improve workspace performance.

Improve Workspaces with Bookmarks

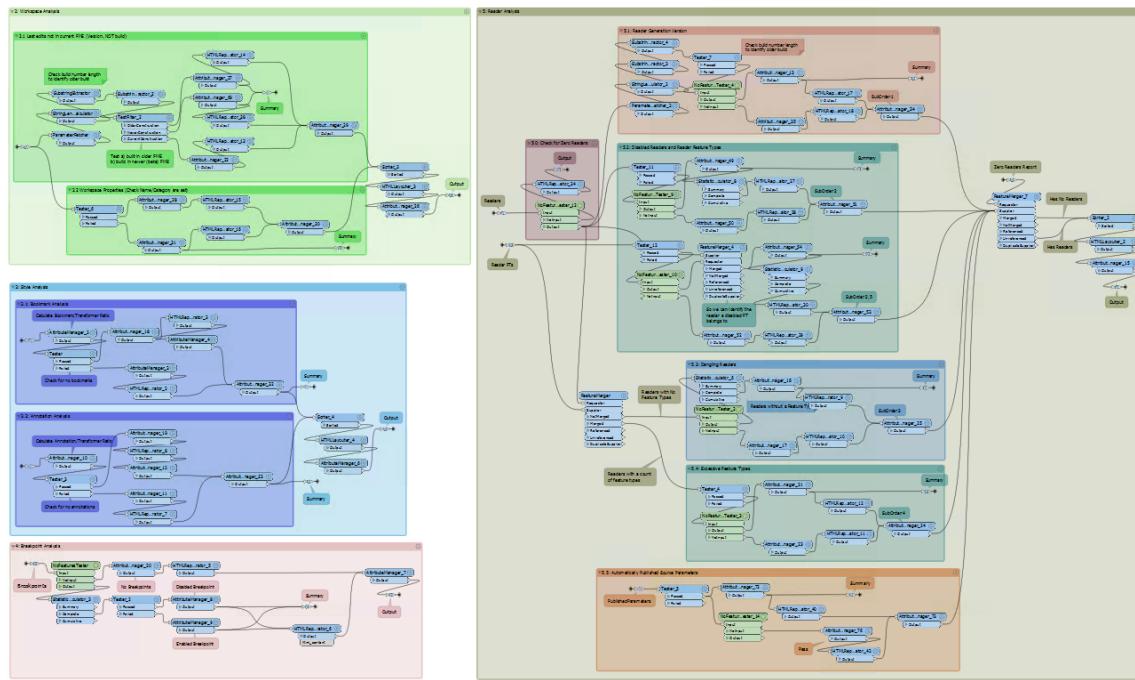
Bookmarks play an essential role in a well-styled workspace for many reasons, including these:

- Design: As a way to subdivide a workspace and manage those sections
- Access: As a marker for quick access to a specific area of a workspace
- Editing: As a means to move groups of transformers at a time
- Performance: As a means to improve workspace performance when caching data

Bookmarks for Design

A bookmark is a great way of indicating that a particular section of a workspace is for a specific purpose. By subdividing a workspace in this way, the layout is often a lot easier to follow.

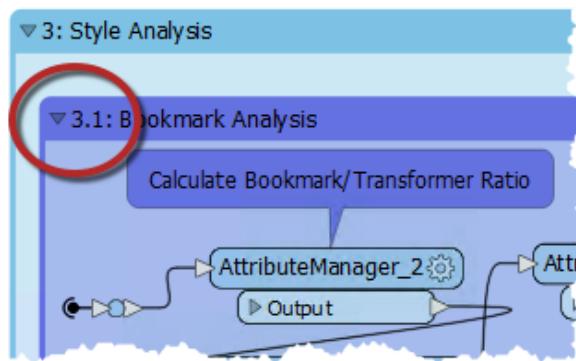
As one user has put it, bookmarks are like paragraphs for your workspace!



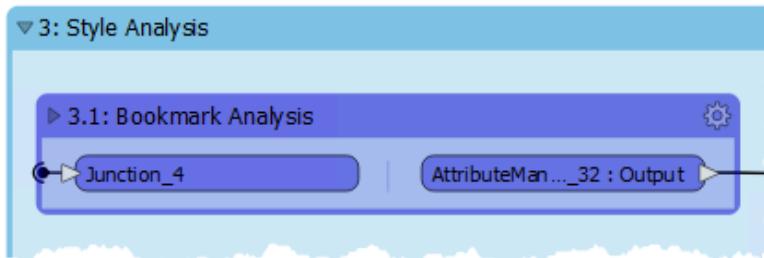
The above workspace illustrates how to mark up different sections of a workspace using bookmarks. As you can see, it's permitted to subdivide bookmarks further by nesting one bookmark inside another.

Collapsible Bookmarks

You can click the small icon in the top-left corner of a bookmark to collapse it:



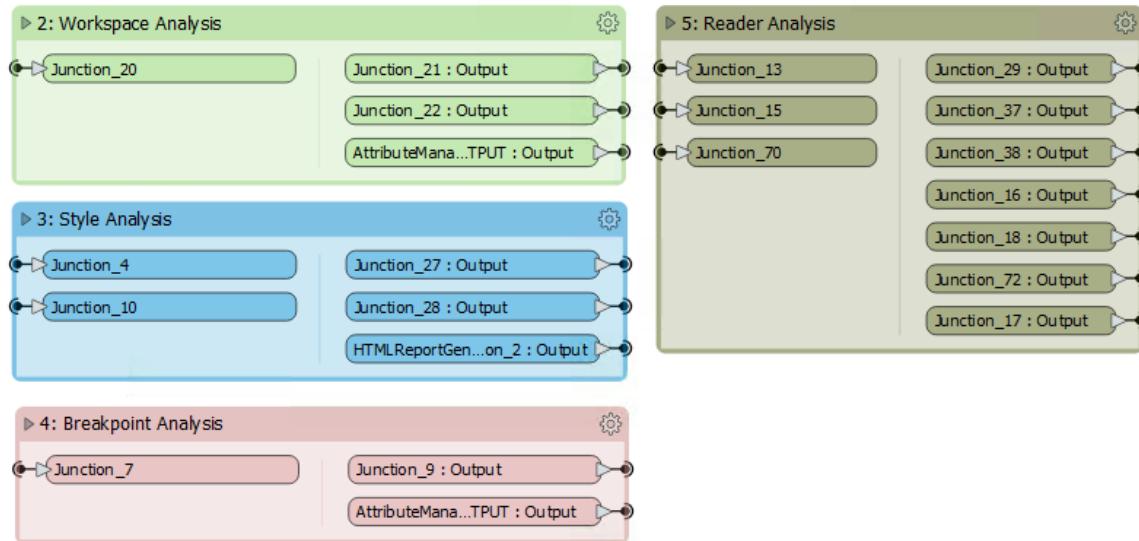
Collapsing a bookmark means it is compressed down to the size of a single transformer, displaying none of the contents except for where data enters or exits the bookmark:



Clicking the icon a second time re-opens the bookmark to its previous size.

This functionality allows you to render large sections of your workspace in a much smaller area and only open them when editing is required.

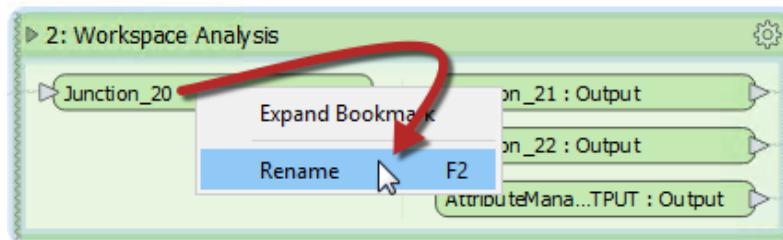
For example, the section of the workspace displayed above might become this:



Re-opening a collapsed bookmark adjusts the workspace's layout, moving other transformers or bookmarks out of the way without overlap. Re-closing the bookmark causes the opposite to occur.

For example, if you expand bookmark three (Style) in the above screenshot, then bookmarks four and five move to one side to accommodate it. When you collapse bookmark three again, the reverse occurs to give the same compact layout as before.

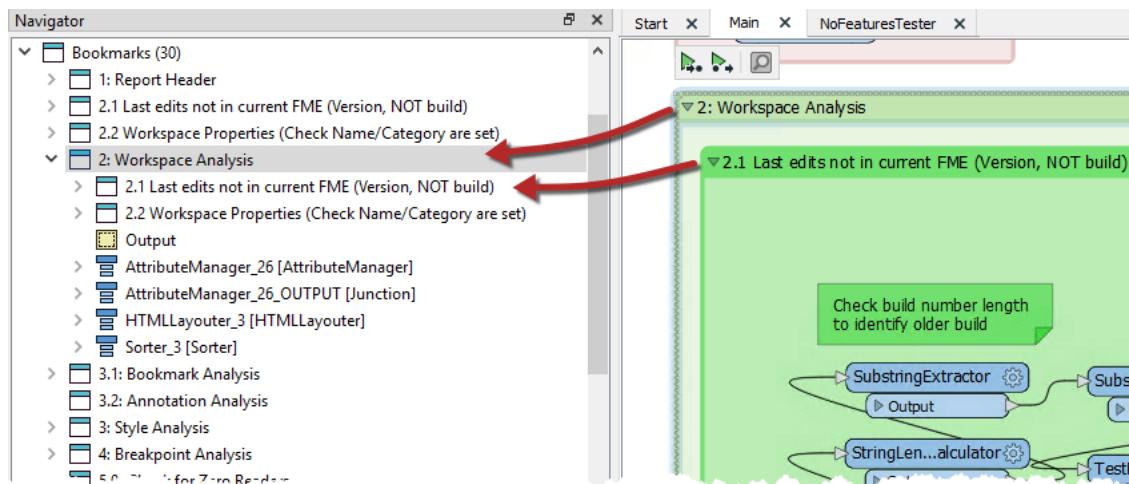
You can rename the input and output ports on collapsed bookmarks to help clarify the data entering and exiting:



You can do this by either double-clicking the object, pressing F2, or using the Rename option on the context menu.

Bookmarks for Quick Access

The Workbench Navigator window lists all bookmarks. Each bookmark appears in order, and you can expand it to view its contents. It may include feature types, transformers, or nested bookmarks:

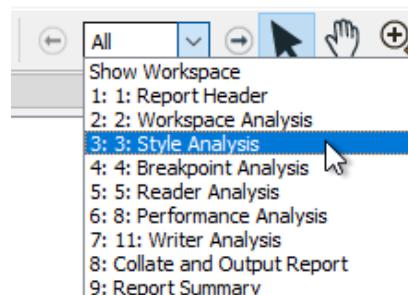


Clicking or double-clicking a bookmark in the Navigator selects that bookmark and brings it into view. Bookmarks both divide a workspace into sections visually and let you jump to different parts of that workspace.

In this way, bookmarks are like the chapter headings in a book!

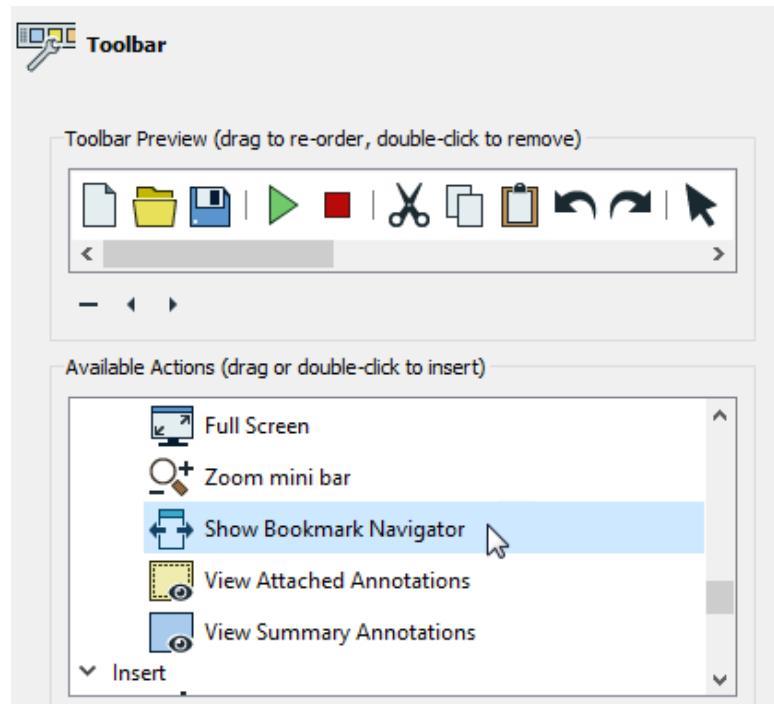
Bookmark Navigator

Bookmarks can also be navigated on the FME Workbench toolbar using the Bookmark Navigator:



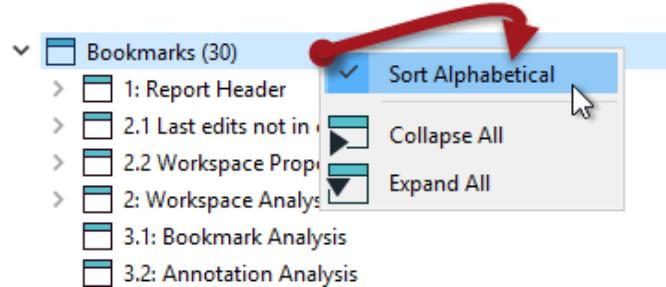
Besides being a way to access bookmarks quickly, you can use the Bookmark Navigator tool to present your workspace. By clicking the arrow button (or pressing the keyboard spacebar), you flip from bookmark to bookmark using animation in a way that would be very useful when showing the workspace as part of a presentation.

To access the functionality, you need to make sure it appears on the toolbar. You can do this by right-clicking on the toolbar and using the customize option.



The order of bookmarks in that window is alphabetical, and that might not always be the same order that you wish to present a workspace.

In that case, right-click on Bookmarks in the Navigator window and turn off the default option to "Sort Alphabetically."



Bookmarks can then be dragged up and down in the Navigator window to give the correct order. Additionally, a new option on the bookmark Properties dialog allows you to exclude specific bookmarks from the Bookmark Navigator. This view does not include nested bookmarks by default. You have to set nested bookmarks to Include for them to appear in the Bookmark Navigator.

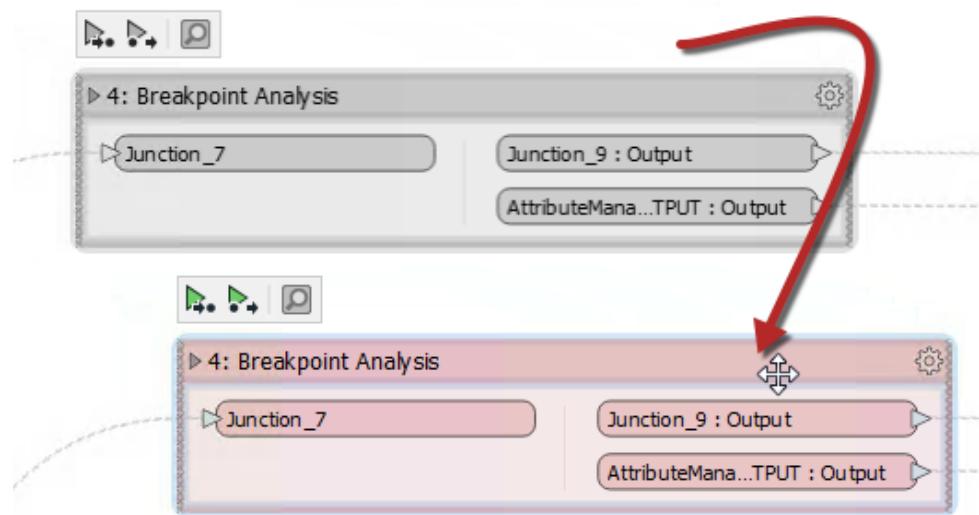


For more information, see this [article on bookmarks](#) on the Safe Software blog.

Bookmarks for Editing

Bookmarks define a section of the workspace containing several objects. When editing a workspace without bookmarks, moving objects is done by selecting the object or objects and dragging them to a new position.

However, when bookmarks divide a workspace, you can move objects by dragging the bookmark to a new position. When an object is inside the bookmark, it moves as the bookmark does.



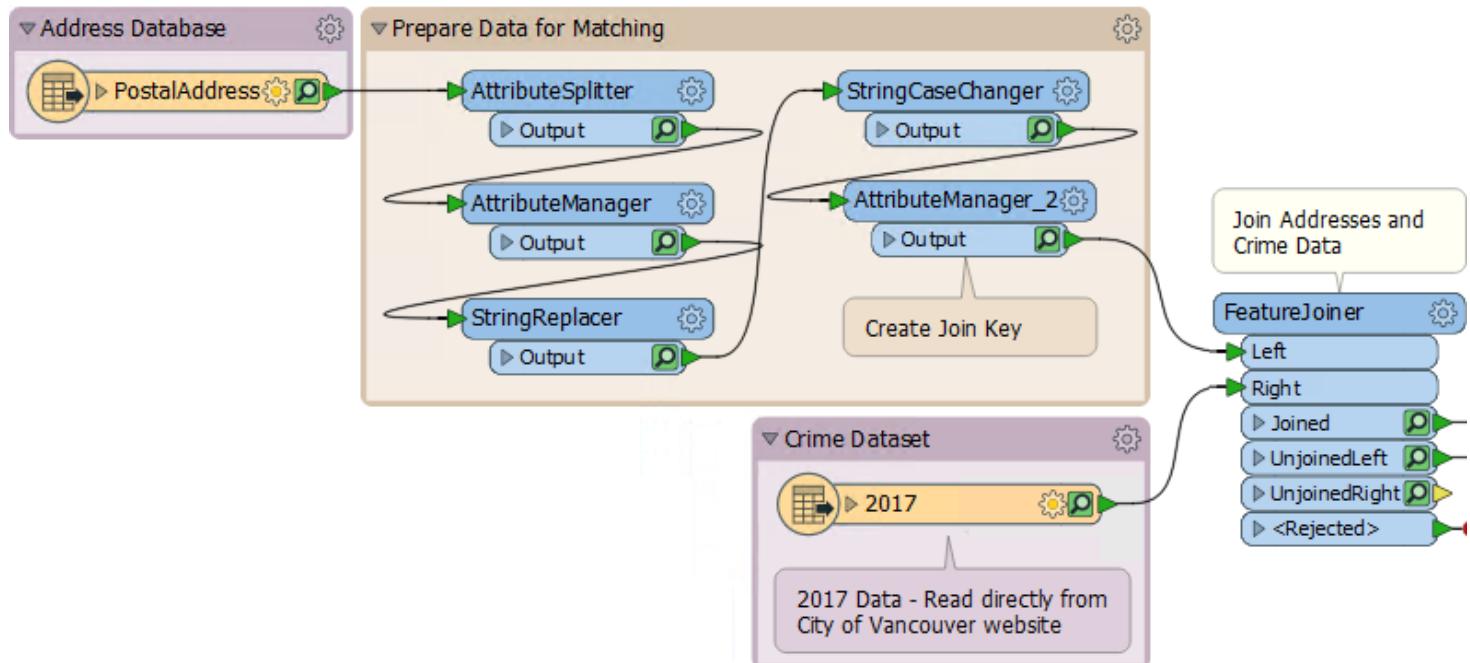
Using this technique, you can move large groups of objects around the workspace canvas to create a better layout. Objects inside a bookmark will move whether the bookmark is collapsed or expanded.



Remember that one of the bookmark properties is labeled **Contents** and has the value either "Move with Bookmark" or "Move Independently." The former allows you to move objects with the transformer; the latter causes objects to remain in position when you move a bookmark.

Bookmarks for Performance

When a workspace is run with Feature Caching turned on, then features are cached at every transformer. As you can imagine, in larger workspaces this leads to a lot of data being cached, sometimes unnecessarily:

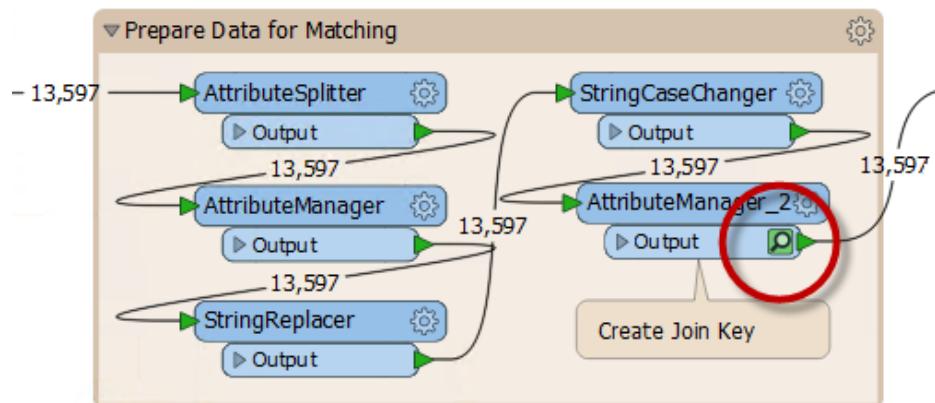


Notice in the above screenshot that every transformer in the Prepare Data for Matching bookmark is being cached.

However, when a bookmark is collapsed, then caching only occurs on the bookmark output objects:



This feature means that data is cached only for the final transformer in the bookmark, saving considerable time and resources:



You don't want to put a workspace into production when caching is turned on, regardless of whether your bookmarks are collapsed. This technique is only recommended for use in the design, authoring, and testing phases of workspace creation.

Additionally, you are limited to storing a few hundred caches per workspace due to operating system file naming restrictions, so collapsing bookmarks can help you avoid hitting that number in large workspaces.

1 Which of the following is not a good reason to use bookmarks?

- A. Access: As a marker for quick access to a specific section of a workspace
- B. Design: As a way to subdivide a workspace and manage those sections
- C. Editing: As a means to move groups of transformers at a time
- D. Performance: As a means to improve workspace performance when caching data
- E. Stability: As a way to reduce the likelihood of errors when running the workspace

Check the Quiz to Earn 100 Points

Exercise: Improving Workspace Style

Learning Objectives

After completing this unit, you'll be able to:

- Rearrange transformers into a logical layout that groups those carrying out a single task.
- Use annotations to clarify the processes taking place in a workspace.
- Use bookmarks to turn a single workspace into defined sections.
- Avoid poor design choices like overlapping connections.

Resources

- [Starting workspace](#)
- [Complete workspace \(example\)](#).

Unit Content

You have been assigned to a project to calculate the "walkability" of each address in the city of Vancouver.

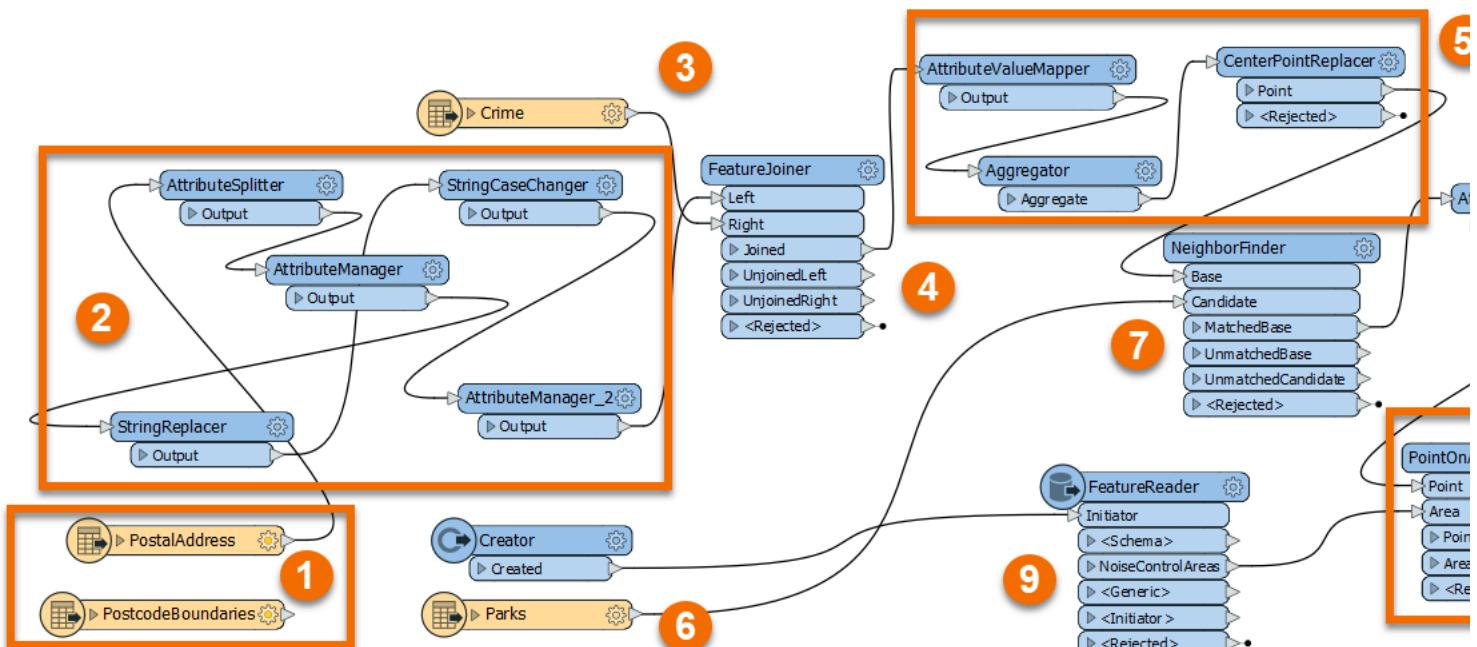
Your colleague wasn't aware of FME style best practices when they gave us the workspace, which made working with it a bit challenging. We need to present our workspace, so we want it to look neat, organized, and well-documented.

1) Start Workbench

Start FME Workbench (2022.0 or later) and open the [starting workspace](#).

2) Examine Workspace

The workspace is pretty unorganized. Let's examine it in sections to figure out how it works.

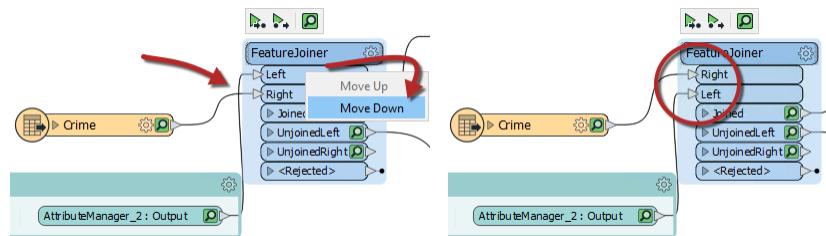


1. PostalAddress and PostcodeBoundaries read from Addresses.gdb.
2. Transformers clean up attributes from the PostalAddress feature type to create a separate Number and Street attribute. Then the last two digits of the Number are being replaced by XX to create an attribute that will be the Join Key for joining the crime data.
3. Crime reads from Crime.csv. This dataset stores the street number for each crime incident with XX as the last two digits to protect anonymity.
4. The FeatureJoiner joins PostalAddress and the Crime data based on the Join Key attribute from section two and the Block attribute from Crime.
5. These transformers give the crime Type attribute a number based on severity and then calculate the total CrimeValue for each address block. Then the CenterPointReplacer extracts only one point if there are multiple crime incidents in the same location.
6. The workspace reads the Parks MapInfo TAB file. It will use this data to measure the walking distance from addresses to parks.
7. The NeighborFinder determines the park closest to each address.
8. The AttributeRenamer renames the _distance attribute from the NeighborFinder to ParkDistance.
9. The Creator and the FeatureReader read a Planning Restrictions OGC Geopackage, including noise restriction areas from the NoiseControlAreas layer.
10. The PointOnAreaOverlayer joins the point data containing the crime, distance to park, and addresses with the NoiseControlAreas polygons. This joined data assigns the noise restrictions to any overlapping points. The AttributeValueMapper is used to assign the zone with a score, creating the attribute NoiseZoneScore. This new attribute will reflect that addresses in noise-restricted areas are more walkable.

3) Rearrange Transformers

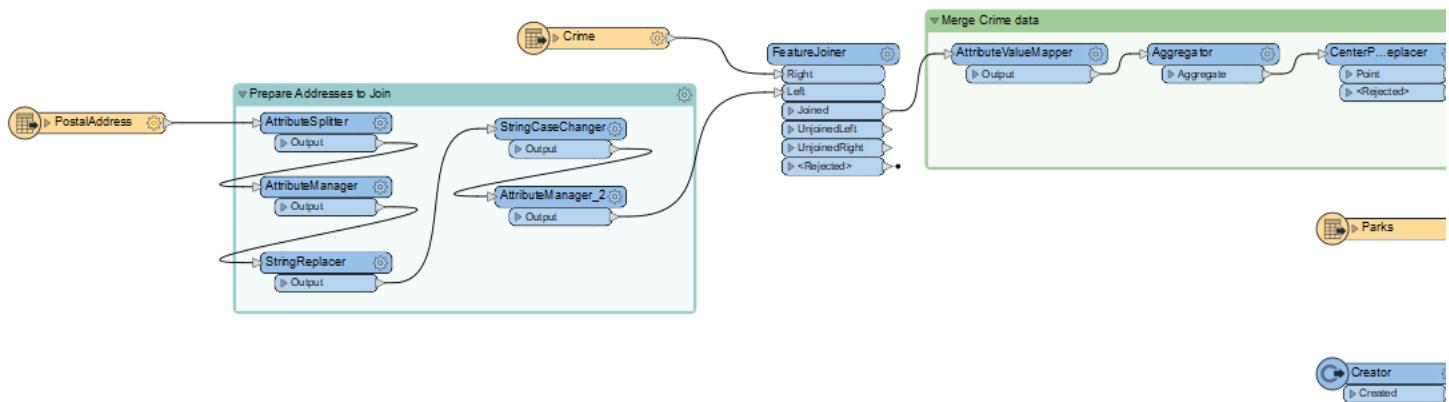
Firstly, let's clean up the transformers. Move the transformers around so that there are no overlapping connections.

For the FeatureJoiner, you could move the Crime reader below the Prepare Addresses to Join bookmark, or you can reorder the FeatureJoiner ports. Right-click on the Left input port, and select Move Down. Now the two connection lines are not crossing:

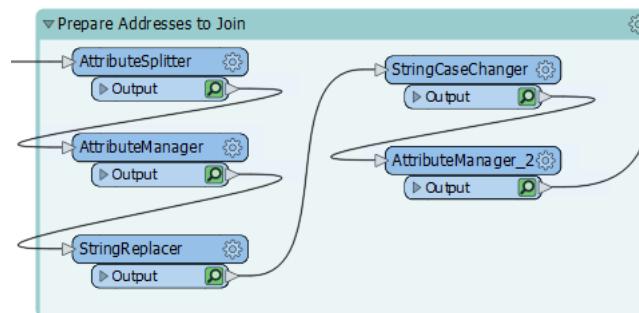


Changing port order can help tidy your workspace, but beware that the different order might confuse experienced FME users who expect ports to be in a particular location.

Move the transformers into a logical order and add a bookmark around any logical groupings:

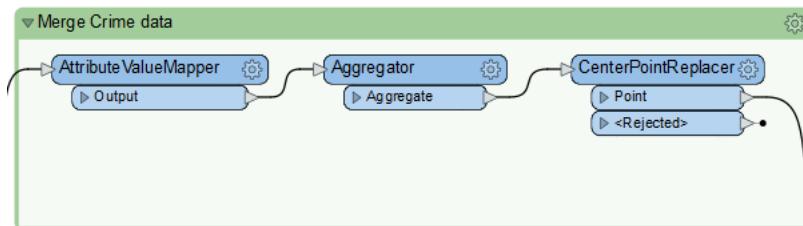


Don't forget to expand the Prepare Addresses to Join bookmark by clicking the right-pointing arrow in the bookmark's top-left corner and organize those transformers:



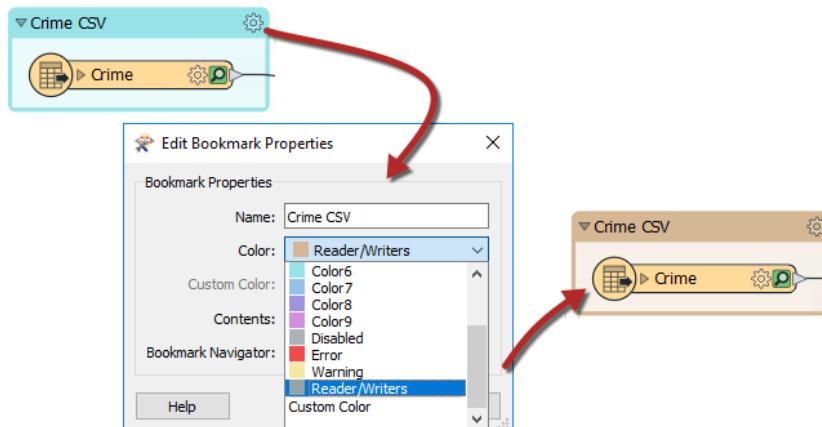
4) Add Style

Having rearranged the transformers and added bookmarks we can now add annotations and color to highlight what is going on. This step will require some inspection of the transformers to find out what they are doing as well as inspecting the readers to know which format they are in:

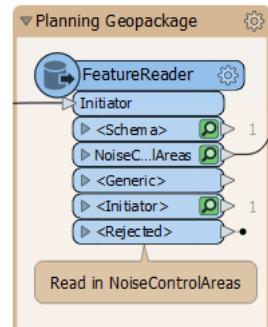


Adding good annotation where necessary will help determine what is going on in the workspace.

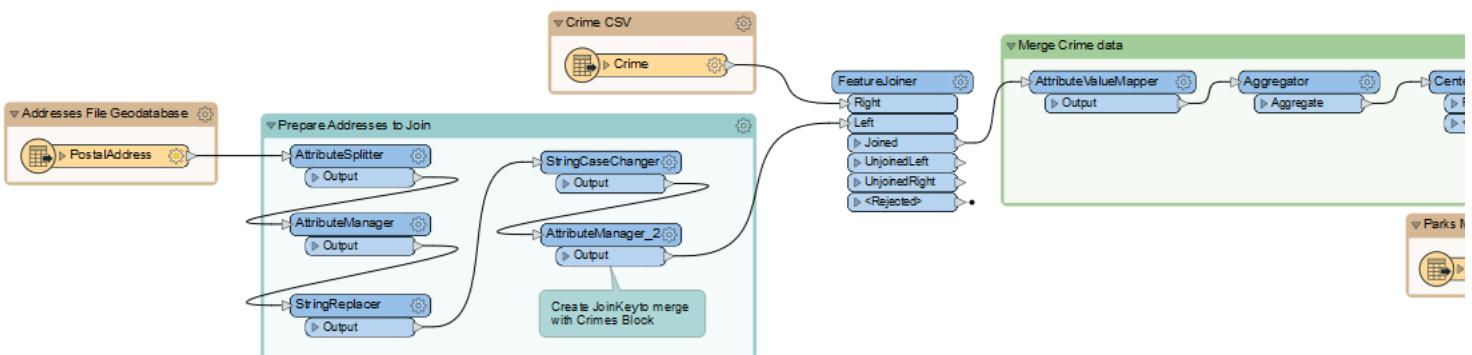
By adding a bookmark around a reader or writer and then setting the color to the preset Readers/Writers color, it is quick to see at a glance where your readers or writers are:



You can also do this with FeatureReaders and FeatureWriters:



Your final workspace should look something like this:



5) Run Workspace

Collapse all the bookmarks if you wish and run the workspace once more to ensure all the caches are fresh. It might be a good idea to re-run the entire workspace.

6) Save the Workspace

You can choose to save this workspace as a regular workspace or as a template workspace.

1Changing the port order on your FeatureJoiner cleans up your connection lines. However, what is a potentially negative result of taking this action?

- A.Experienced FME users will expect ports to be in a particular location.
- B.The FeatureJoiner will run slower.
- C.The Crime feature type has to be below the PostalAddress feature type on the canvas.
- D.It will cause an error.

2Which of these options would be the best annotation for the AttributeValueMapper transformer?

- A.Map the crime attribute.
- B.AttributeValueMapper version 3, FME 2020.1
- C.Assign each crime a number based on severity.
- D.Here I mapped some of the crime attributes to accomplish the walkability score goal of the workspace in the context of this section.

3Why might you choose the preset brown Readers/Writers color for your bookmarks containing readers, writers, and FeatureReaders/FeatureWriters?

- A.It matches the color of transformers.
- B.It is a pleasant color.
- C.Brown is a color naturally associated with reading and writing.
- D.It is quick to see at a glance the location of your readers or writers.

4If you add and collapse a bookmark after the FeatureJoiner containing the next three transformers, how many features come out of the CenterPointReplacer : Point port?

- A.262
- B.2939
- C.6840
- D.9,899
- E.13,597

[Check the Quiz to Earn 200 Points](#)

Use Prototyping and Incremental Development

Learning Objectives

After completing this unit, you'll be able to:

- Use workspace prototyping and incremental development to build workspaces.
- Create simplified input data while prototyping.
- Employ version control techniques in workspace development.
- Use rejected feature handling.

Prototyping

In the classical sense, prototyping means creating an incomplete application as a way to evaluate the feasibility of a project.

Here we'll stretch the definition to mean how to build a complex FME project incrementally; starting with an empty workspace and building it piece by piece to deliver a result that matches the final specification.

We will cover the techniques used for building a workspace incrementally, and how to handle data that is rejected by a transformer.

Incremental Development

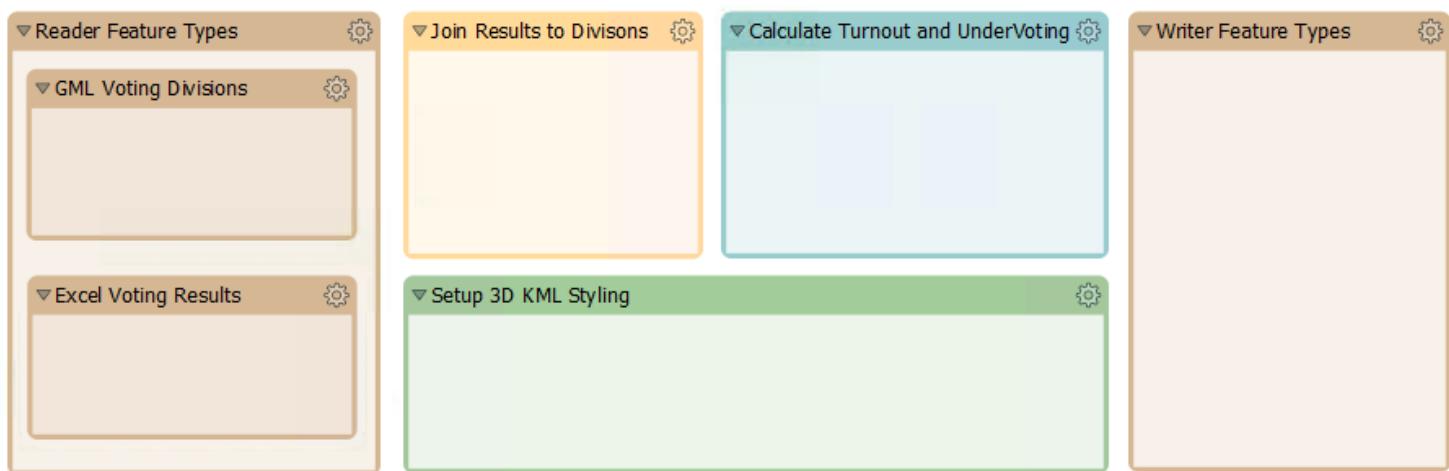
The key development technique for FME workspaces is [incremental updates](#).

The steps to this technique are:

- Plan your project as a series of small sections, each of which would fit inside a bookmark in FME.
- Design and implement a section in FME Workbench. It should ideally be between 3-10 transformers.
- Test each section immediately after it is completed. That way you can identify problems at an early stage, and identify them more easily because only a few changes are being made in any increment.
- Repeat the process, saving the workspace and testing it whenever you've added a new section.

Although a range of 3-10 transformers is an arbitrary number, the more transformers you add, the more difficult it would be to identify the source of any problems. Beyond ten transformers is the point at which you should consider chopping that process into smaller sections.

Here an author has planned their workspace by laying it out as a set of bookmarks on the canvas:



Now the author can complete and test each section at a time, keeping the overall goal in mind at the same time.



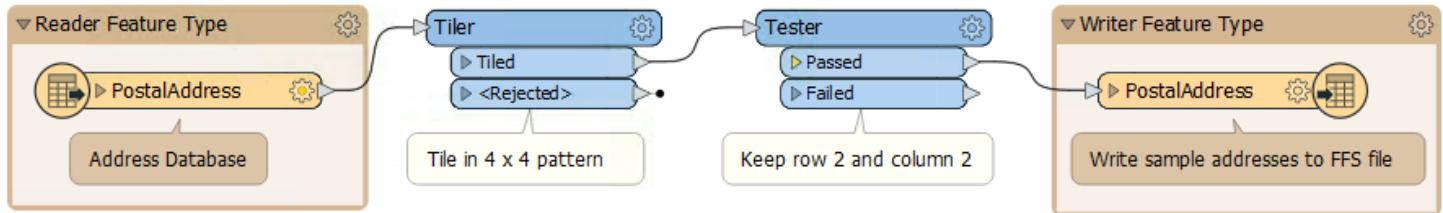
It can be all too easy to start developing a workspace and forget to save it at all! FME keeps a recovery file as soon as the workspace is saved for the first time, but until then you are running the risk of an irretrievable loss.

Source Data

When the FME project is large and complex, it's likely that the source data will be large and complex too. So when creating a workspace in small increments, testing each part in turn, it's better to avoid using the entire dataset.

It's better to use a sample of source data for testing. In fact, it's better to create a small sample of data - extracting it from your source and writing to a neutral format like FFS - rather than randomly sampling the data for each test run.

Sampling is particularly useful for databases because it also avoids the problems of waiting for network traffic and database responses.



Here the workspace author is extracting a section of source data by reading from a database, splitting it into tiles, and writing just one tile to the FFS format. This one tile can be used for prototyping a solution in a way that is representative of the entire source database table.

Another transformer to use would be the Sampler, although the features selected by it would not be spatially adjacent.

Version Control

When making a set of incremental changes to a workspace, it's easy to work on a single workspace file only. However, there are various problems with this:

- Faults cannot be easily tracked because there is no record of what has changed and when
- It is not easy to create different versions for different platforms
- If the workspace file is lost or corrupted, then the entire project is lost

Therefore, it is better to keep versioned workspaces, where a different copy is kept for each set of revisions. This precaution can be taken manually within the file system, or by using a version control system like [Git](#).

In fact, it is a good idea to keep and version all materials related to an FME project, including:

- Workspace files
- Python files
- Log files
- Source datasets

It's better not to store any information that is personal or that includes passwords. Also, there's no need to store temporary files.

You can use a few methods to keep track of version and editing history:

- Including the date of the most recent edit in the filename, e.g., digital-plan-submission-2022-11-25.fmw
- Including a version number or name in bookmark or annotation in the workspace itself
- Using the Overview and History [Workspace Parameters](#) to store workspace metadata



★ New for 2022.0: FME Workbench has a new tool (in Tech Preview) to help you collaborate and track changes in your workspaces: Workspace Compare and Merge. Learn more with [Visually Compare and Merge Workflows](#).

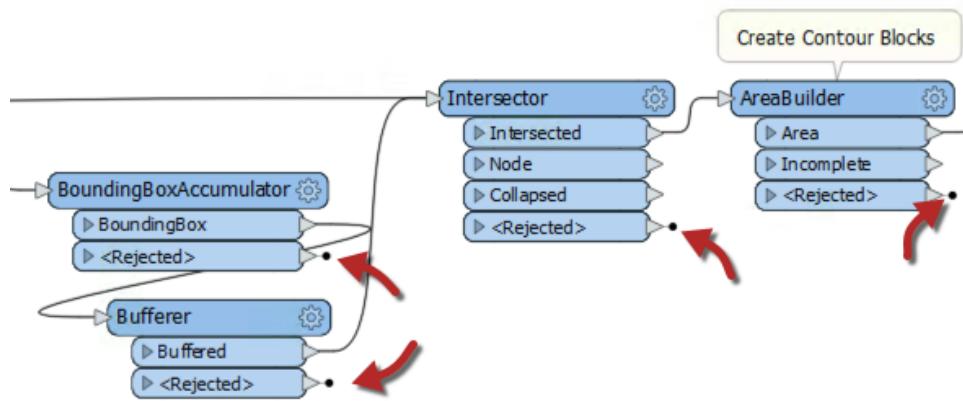


FME Server has a [version control system](#) built in. Even if you don't have an FME Server license, you can still install it for use as a repository system for workspaces and related files.

Rejected Features

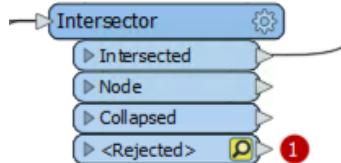
An important part of any workflow is handling data that fails to process. For example, where a feature with no geometry is sent into a geometry-based transformer like the AreaBuilder.

FME handles such failures by outputting the data through <Rejected> ports, which are found on many transformers:



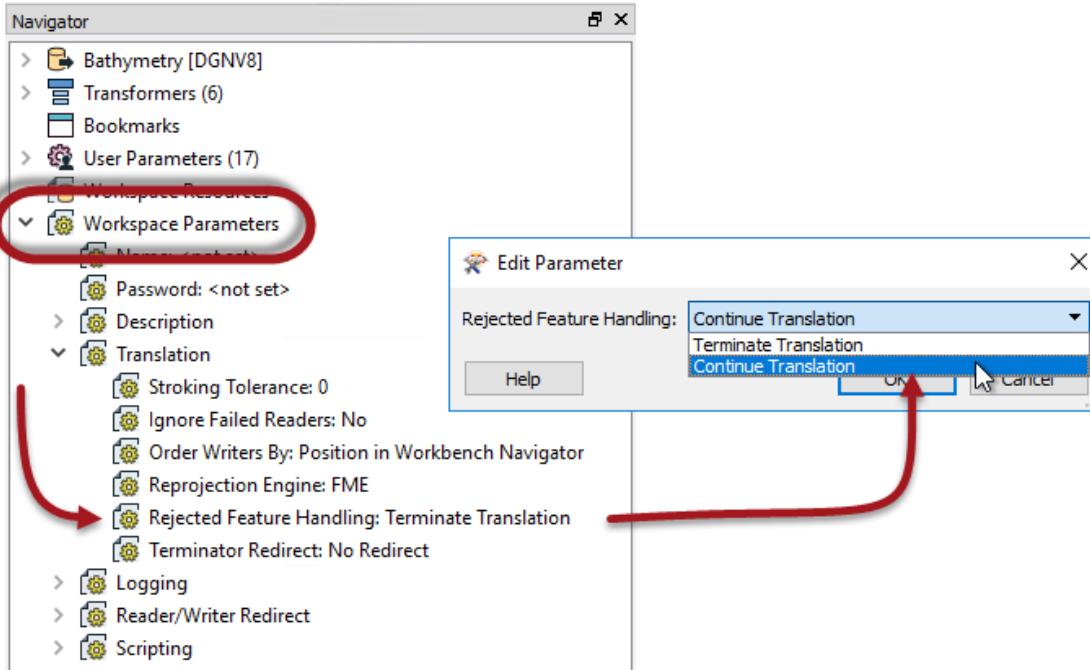
A parameter exists to control the action of <Rejected> ports and gives the workspace author a choice over what action to take.

When a feature is rejected, the translation will stop and a red circle with a number will appear on the <Rejected> port. You can click on the cache to inspect the feature and determine why it was rejected:



Rejected Feature Handling

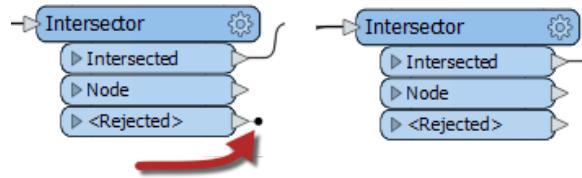
The parameter to control the handling of rejected features can be found in the Navigator window, under Workspace Parameters:



The two options are *Terminate Translation* and *Continue Translation*.

When the parameter is set to *terminate*, then a feature that exits via a <Rejected> port causes the translation to stop. To visually denote this, the <Rejected> ports have a small black marker on them.

When the parameter is set to *continue*, then the translation will continue, regardless of how many features exit <Rejected> ports. In that case, the small black marker is removed:



In *terminate* mode, a rejected feature gets written to the log window with the error message:

The below feature caused the translation to be terminated

There will also be an error message relating to the transformer:

`Intersector_<Rejected>(TeeFactory): Intersector_<Rejected>: Termination Message: 'Intersector output a <Rejected> feature.'`

Copy

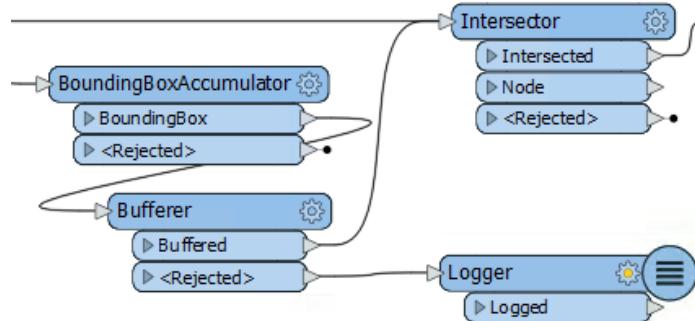
This error is useful because it tells the author which transformer experienced the failure.



To quickly find these error messages you can filter the Translation Log by clicking on the Errors button.

Mixed Mode

In *terminate* mode, a rejected feature will not cause the translation to stop, provided that the <Rejected> port is connected to a further object:



In short, an author can create a mixed mode, where some transformers stop the translation on rejecting a feature (the Intersector above), but others will handle the feature another way (the Bufferer). That way the author can try to handle rejected features that are expected, but stop the translation if there are truly unexpected failures.

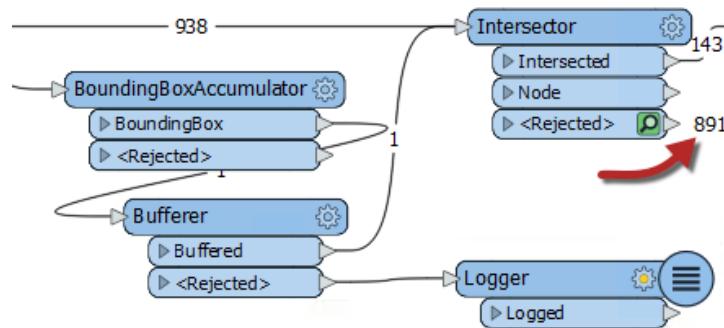


Generally:

- Terminate mode is useful if you do not want any features to be rejected. You want to verify all your data is correct.
- Continue mode is useful if you don't care if a feature is rejected. You assume rejections are happening because of bad data and don't want to use those records.
- Mixed mode is useful if you want to pick and choose which transformers' <Rejected> ports you care about.

Feature Counts and Inspection

In continue mode, features that exit a <Rejected> port are counted and saved for inspection:



Features will be saved for inspection even if there is no Logger or other transformer attached. The number tells us how many features were rejected and the green icon can be clicked to inspect the data.

1 Which of the following is not a step in using incremental development to create a prototype workspace? —

- A. Save the workspace after testing a section, then test the next section.
- B. Test each section immediately after it is completed.
- C. Create a custom transformer to improve the testing process.
- D. Plan your project as a series of small sections, each of which would fit inside a bookmark in FME.

2 Which transformer can you use to create a subset of data to use when prototyping? —

- A. Generalizer
- B. RasterResampler
- C. SampleDataCreator
- D. Sampler
- E. Simplifier

3 Which of the following is a problem that can be addressed by using a version control system? —

- A. Poor performance when conducting complex calculations on big data
- B. Getting rejected features when trying to buffer data without geometry
- C. Output data not matching desired schema
- D. Tracking down when problems emerged in a workspace is difficult because there is no record of what has changed and when

4 Why might you use Continue Translation mode for the Rejected Feature Handling parameter? —

- A. You assume rejections are happening because of bad data and don't want to use those records.
- B. You want to pick and choose which transformers' <Rejected> ports you care about.
- C. You want to verify all your data is correct.

Check the Quiz to Earn 100 Points

Exercise: Workspace Prototyping

Learning Objectives

After completing this unit, you'll be able to:

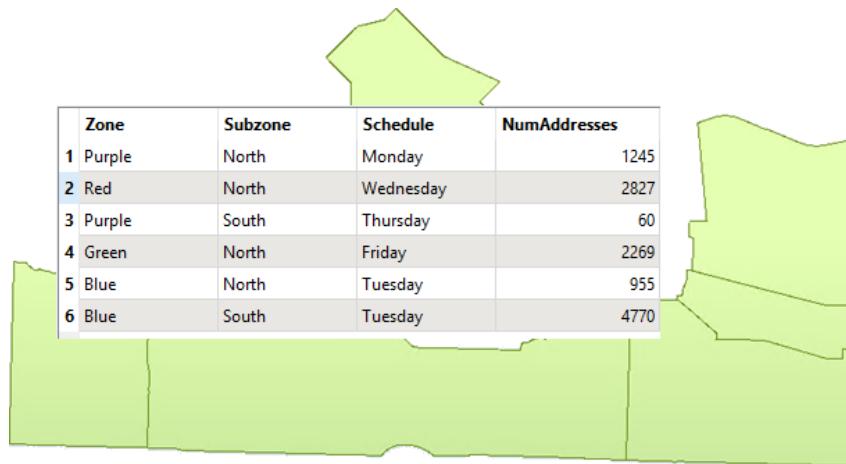
- Plan a workspace.
- Create a workspace, section by section.
- Restrict source data to a small sample.
- Save a workspace with a date or version number.

Resources

- [Starting workspace](#)

Exercise

The city maintenance department has a dataset of garbage collection schedules, to assign residents to a collection on a particular day:



However, because of shifting demographics and zoning changes, they have decided that new boundaries should be drawn.

Your task is to use FME to create new boundaries. You must create five polygons, adjacent to each other, and with approximately the same number of residents in each. The analysis will be based on the city's address database. An estimate of the number of residents per address will be created depending on the zone type it falls within:

- Single-family residences: 2 adults
- Two-family residences: 4 adults
- Multi-family residences: 12 adults
- Comprehensive development zone: 8 adults
- Commercial properties: 1 adult

The output format will be OGC GeoPackage.

To develop this workspace, it's necessary to consider what different steps might be required. We can then create sections with a bookmark and fill them in as we go along.

1) Plan Workspace

Let's plan this workspace together.

We need to read the address data (or a sample of it) and write the output to OGC GeoPackage. We need to know what zone type each address falls inside, which needs zoning data and a transformer to carry out a spatial join.

We also need to create a resident count based on the zone type and then divide the residents into five different areas. Finally, we need to group the addresses with a boundary shape around them.

In short, we need this set of actions:

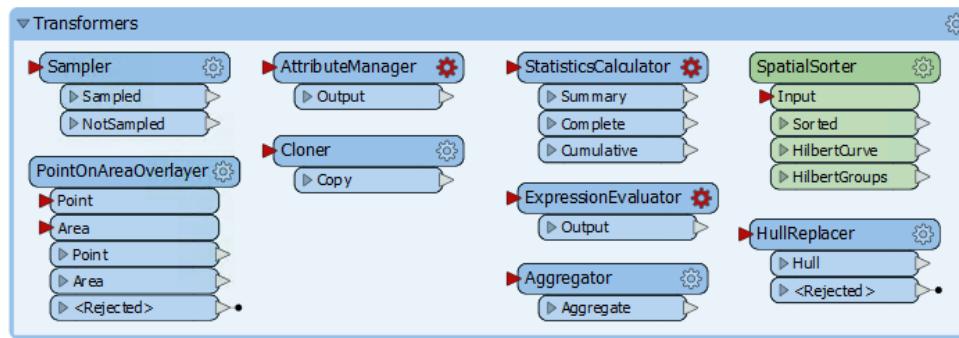
- Read/Sample Address database
- Read Zoning data
- Create a spatial join
- Calculate resident count
- Divide residents into five groups
- Aggregate the addresses into their group
- Create a boundary shape

- Write OGC GeoPackage

So, open the [starting workspace](#) in FME Workbench (2022.0 or later). It already has a set of bookmarks to represent these steps to be carried out but, as yet, we can't be sure which sections will be larger, so all bookmarks are the same size:



You'll also find a bookmark containing all of the transformers required for the exercise:

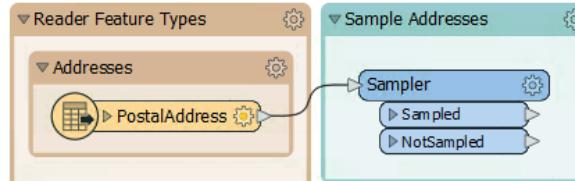


Or at least, these are the transformers we think will be required for the exercise!

2) Sample Source Data

There are more features in the address database than we need for workspace construction and testing, so let's reduce that to a smaller sample.

Rather than create a test dataset, here we'll use a Sampler transformer. There is a Sampler transformer in the "Transformers" bookmark, so simply move that transformer into the "Sample Addresses" bookmark and connect the PostalAddress feature type to it:



Inspect the Sampler's parameters. It will sample every 25th feature

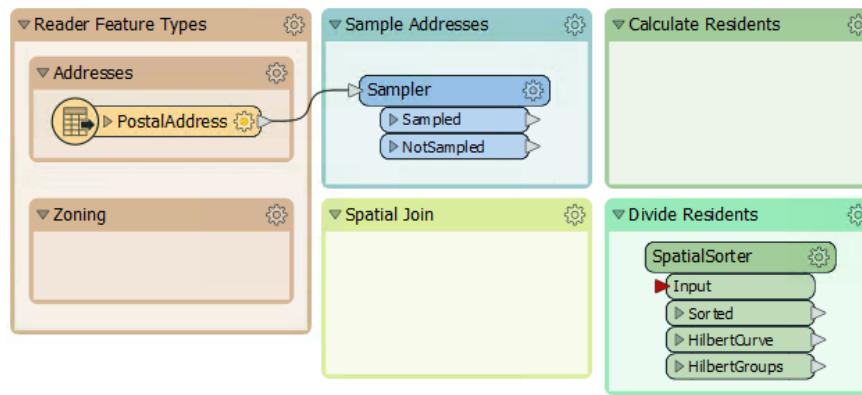


Run the workspace to be sure it is sampling the data correctly. Click on the magnifying glass on the Sampler:Sampled output port to view the data in the Visual Preview window. Take note of how many features come out of the Sampler:Sampled port; you will need this number to answer the quiz.

3) Divide Data into Groups

Before trying to add the Zoning dataset into the workspace, let's try and create groups from the basic dataset. We can do this with a custom transformer from the FME Hub, called the SpatialSorter.

So move the SpatialSorter from the "Transformers" bookmark to the "Divide Residents" bookmark:



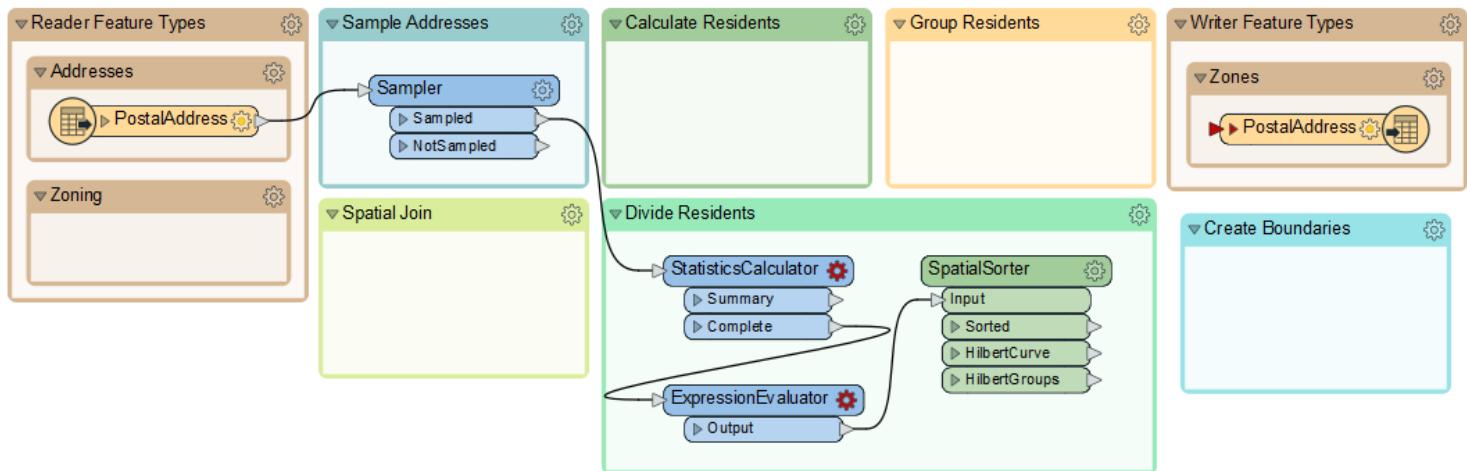
The SpatialSorter sorts data spatially (so features closer geographically become closer in the sorted output) and creates groups.

Check the parameters for this transformer. Notice that the group parameter asks for group size, not the number of groups. Therefore we'll need to calculate how many addresses there are when split into five groups.

4) Calculate Group Sizes

To calculate the number of addresses per group, we need the number of addresses and then divide that by five. We can do this with a combination of StatisticsCalculator and ExpressionEvaluator.

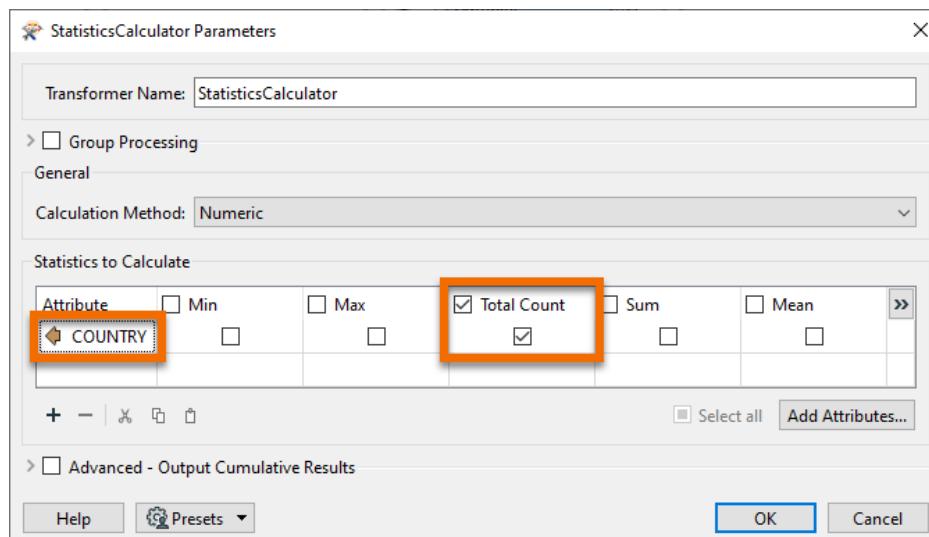
So, enlarge the Divide Residents bookmark as required and move the StatisticsCalculator and ExpressionEvaluator transformers from the "Transformers" bookmark. Connect them up to the Sampler:Sampled port like so:



5) Calculate Group Sizes

Inspect the parameters for the StatisticsCalculator. This transformer will tell us how many features there are (the Total Count). Pick an attribute for the first row of the Attribute column. Because we only want to count features not create true statistics, it can be any attribute you like. Here we are using COUNTRY.

Check Total Count.

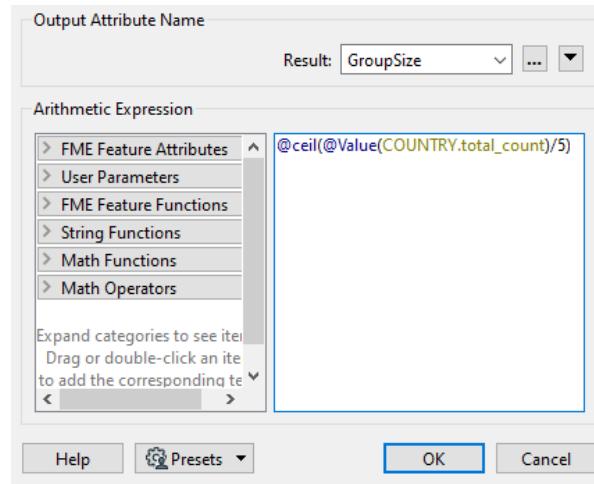


Click OK.

In the ExpressionEvaluator, enter GroupSize in the New Attribute parameter. In the Arithmetic Expression field enter the expression:

```
@ceil(@Value(COUNTRY.total_count)/5)
```

[Copy](#)



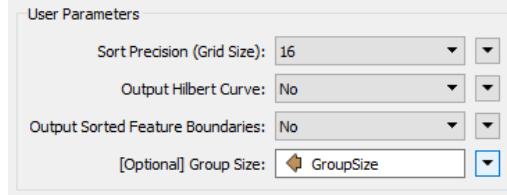
The simplest way is to double-click the ceil function to add it, then double-click the TotalResidents attribute, and manually add the /5 part.

This expression will divide the number of residents into five groups, rounding up. The rounding up part is essential, and it's what the ceil function does.

Run the translation and view the ExpressionEvaluator output in the Visual Preview window to provide that this part works. The TotalResidents should be 543 and the GroupSize should be 109 for each feature.

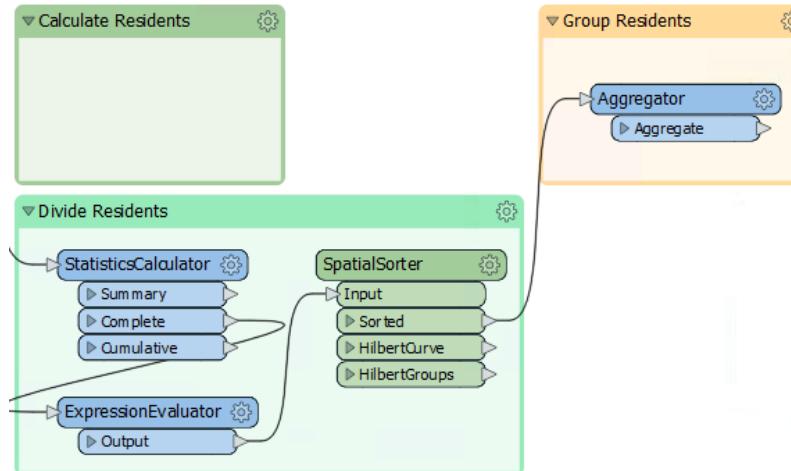
6) Group Residents

Now inspect the parameters for the SpatialSorter once more. We will leave Grid Size at 16 for now; this will give us a more coarse result but will run faster while we develop the workspace. Under the Group Size parameter, click the drop-down arrow and select Attribute Value > GroupSize:

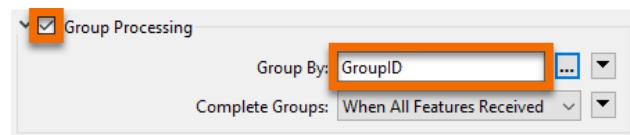


This sets the group size to the attribute just calculated.

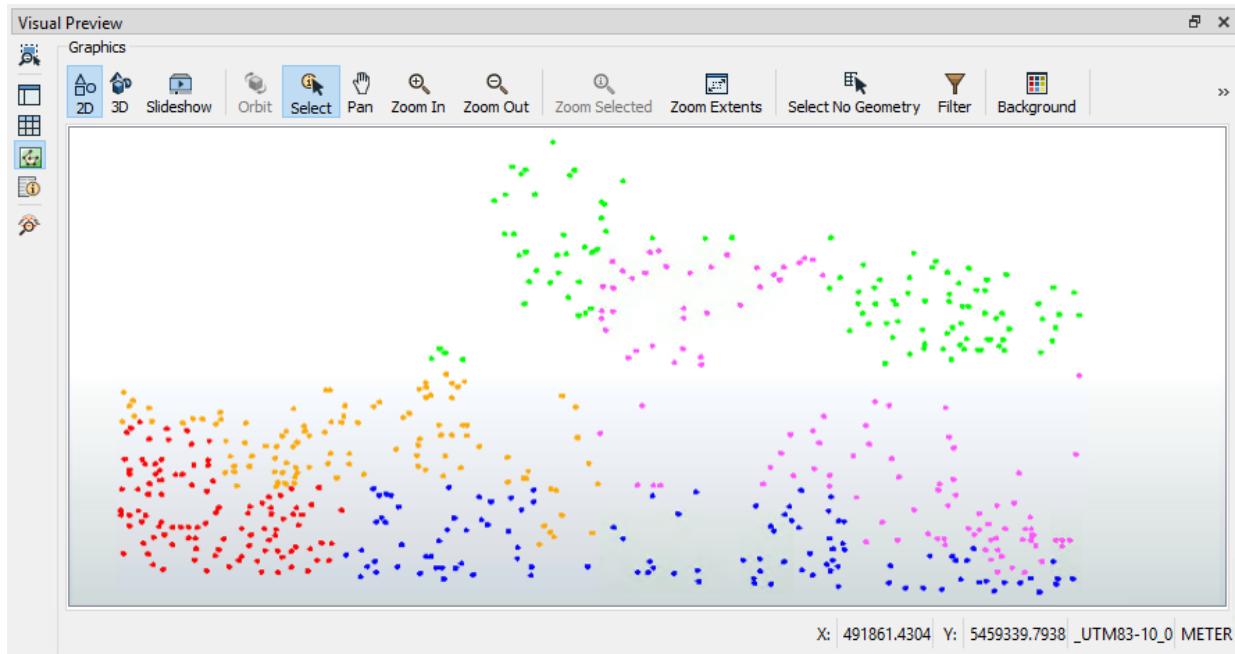
To create groups of addresses, move the Aggregator transformer to the "Group Residents" bookmark, and connect it to the SpatialSorter:Sorted output port:



Inspect the parameters for the Aggregator. Set the Group By parameter to the GroupID attribute (in other words, aggregate features together in the groups created by the SpatialSorter):



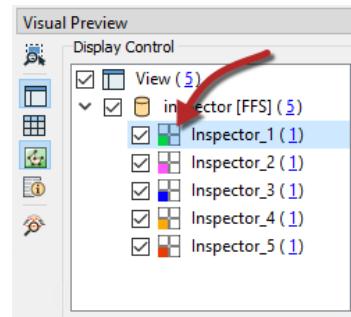
Using an Inspector transformer we can also set the Group By parameter, which will result in the data being represented with different colors for each group in the Visual Preview window. Connect an Inspector to the Aggregator:Aggregate output port and then in the parameters set the Group By to GroupID. Run the translation, and you should find there are five sets of point aggregates in the output, each of which has approximately the same number of point features:



Tip

The Inspector transformer can be used to inspect data, but with Visual Preview and feature caching there isn't much reason to use it often. This case, where you want to add a group-by to help visualize your data in groups, is one of the cases where this transformer is useful.

You can change the color of the groups by clicking on the Toggle Display Control button on the left-hand side of visual preview. Then double-click on the quadrant/four-square icon to open the Geometry Drawing Styles dialog, where you can set the color. When the color is manually set, the quadrant icon will display the color.



7) Save Workspace

Save the workspace, including a date or version number (like GarbageCollection-2022-05-18.fmw).

In this exercise, you learned to plan your workspace in sections and work on each section one at a time. The next step in the workspace will be to add in the Zoning data, create a spatial join, and calculate how many residents live in each property based on each address's zoning type.

1) It is best practice to use bookmarks to plan your workspace before adding transformers.

- A.True
- B.False

2) Once you have planned your workspace using bookmarks, it's best practice to work on one section at a time, testing your work as you go using partial runs and feature caching.

- A.True
- B.False

3How many features come out of the Sampler:Sampled port?

- A.195
- B.489
- C.543
- D.5,490

4Including a version number in your workspace name automatically checks your workspace into a version control system.

- A.True
- B.False

Check the Quiz to Earn 200 Points

Understand Workspace Structure

Learning Objectives

After completing this unit, you'll be able to:

- Define a workspace.
- Define a reader.
- Define a writer.
- Define a feature type.
- Define a feature.
- Understand the hierarchical structure of workspaces.

Workspace Components

A **workspace** is the primary element in an FME translation and is responsible for storing a translation definition. Think of a workspace as the container for all the functionality of a translation, which is stored in the following components.



Workspaces do not store the data they work with. Data is simply represented in the workspace as file paths, URLs, or Connections. The workspace just provides FME the instructions on what data integration workflow to carry out.

If you want to include your data in your workspace, e.g., for sharing, you can save it as a [template](#).

Readers and Writers

A **reader** is the FME term for a component in a translation that reads a source dataset. Likewise, a **writer** is a component that writes to a destination dataset.

Readers and writers are represented by entries in the Navigator window.

Feature Types

Feature type is the FME term that describes a subset of records. Common alternatives for this term are *layer*, *table*, *sheet*, *feature class*, and *object class*. For example, each layer in a DWG file, or each table in an Oracle database, is defined by a feature type in FME.

Feature types are represented by objects that appear on the Workbench canvas.

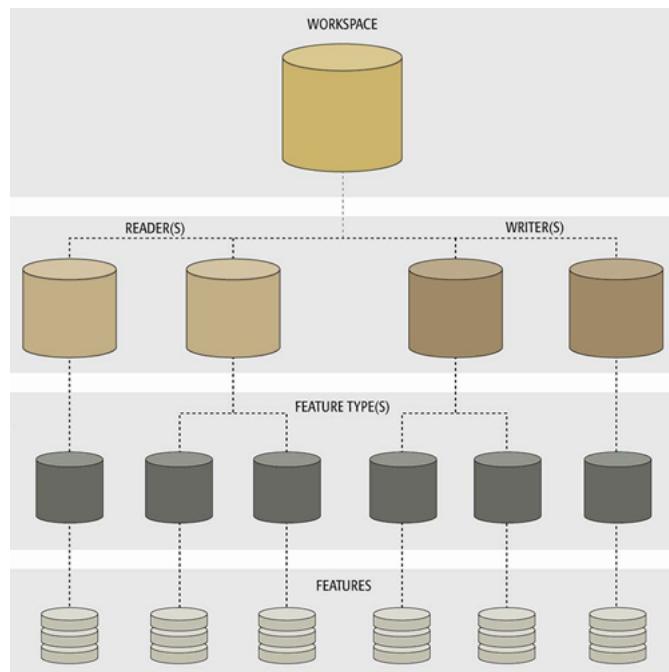
Features

Features are the smallest single components of an FME translation.

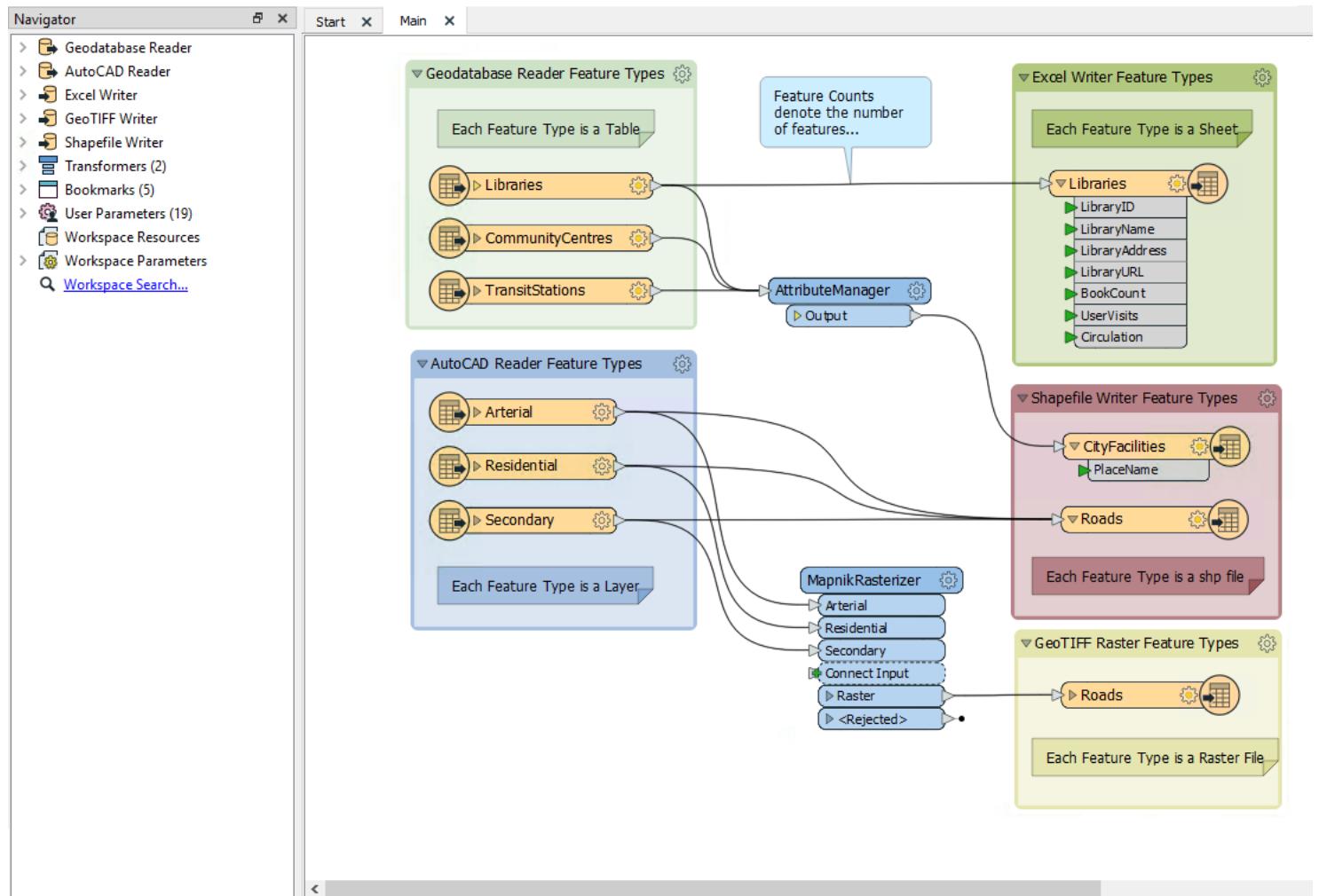
They aren't individually represented within a workspace, except by the feature counts on a completed translation.

Relationships

Each workspace can contain multiple readers and writers, each of which can have multiple feature types, with multiple features. They exist in a hierarchy that looks like this:



A workspace with multiple readers and writers might look like this:



This workspace has two readers (each with three feature types), and three writers (with one, two, and one feature types). Each reader and writer is a different format, and each has a different name for its feature types.

1 Workspaces save all the data they work with into the workspace file itself.

- A.True
- B.False

2 Most data is divided or categorized into tables, layers, sheets, or classes. What is the FME umbrella term for these subdivisions?

<https://sale.mytrailhead.com/content/sale/modules/design-workspaces-for-advanced-reading-and-writing/understand-workspace-structure>

- A.Data Types
- B.Feature Types
- C.Object Types
- D.Record Types

— 3 In the hierarchy of FME components, which object is at the top level, containing all the others?

- A.Workspace
- B.Readers and Writers
- C.Features
- D.Transformers
- E.Feature Types

— 4 A workspace can have multiple readers and writers.

- A.True
- B.False

— 5 A reader or writer can have multiple feature types.

- A.True
- B.False

— 6 A feature type can produce multiple features.

- A.True
- B.False

[Check the Quiz to Earn 50 Points](#)

Create Workspaces With Multiple Readers and Writers

Learning Objectives

After completing this unit, you'll be able to:

- Add multiple readers and writers to your workspace.
- Update readers and writers.
- Set reader and writer parameters.

Reading and Writing Workflows

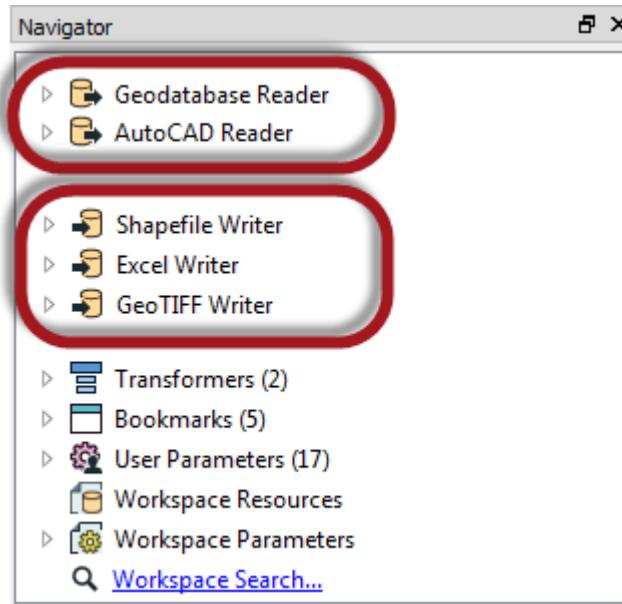
By default, the Generate Workspace dialog creates a workspace with a single reader and a single writer. However, this does not mean the workspace is forever limited to this. Additionally, FME can read/write data using:

- Multiple readers, multiple writers, or both
- FeatureReader and/or FeatureWriter transformers
- Integration transformers such as the DropboxConnector, Email, HTTPCaller, etc

Multiple Readers and Writers

An FME workspace is not limited to any particular number of readers or writers; readers and writers can be added to a workspace at any time, any number of formats can be used, and there does not need to be an equal number of readers and writers.

For example, the Navigator window shows this workspace contains two readers and three writers, of different data types and formats!

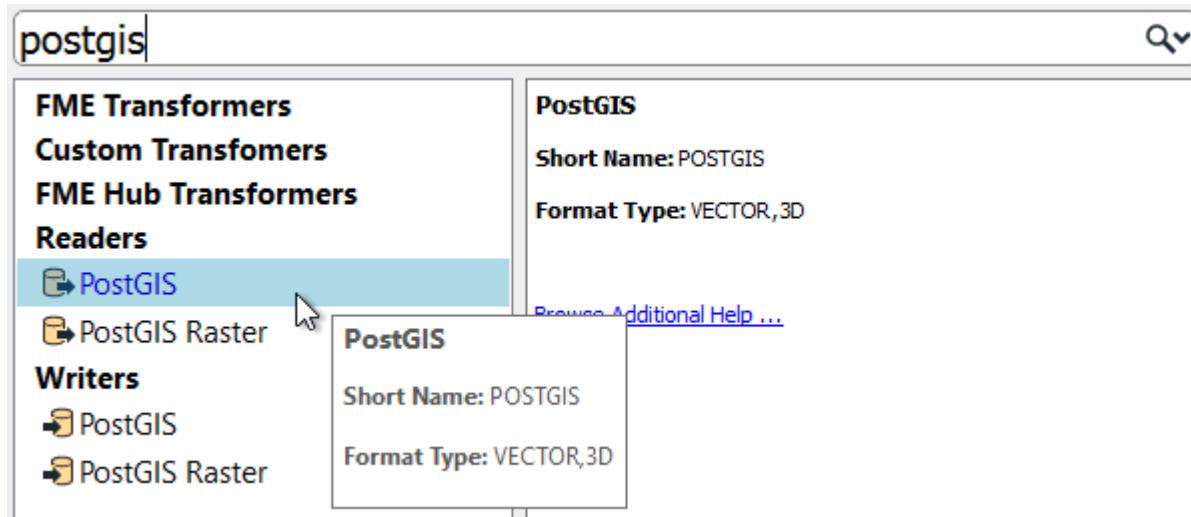




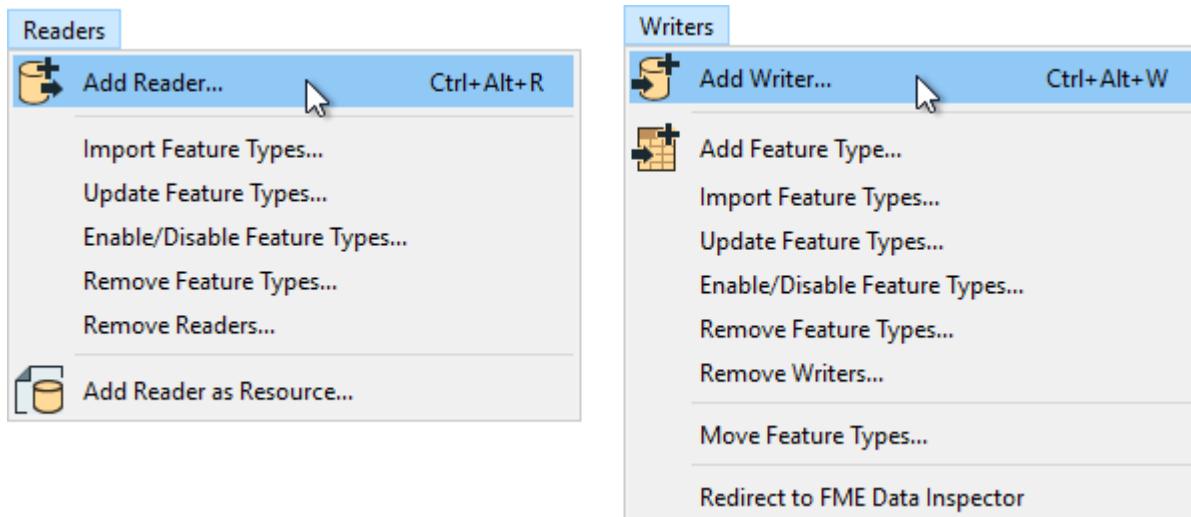
It's important to note that readers and writers don't appear as objects on the Workbench canvas. Their feature types do, but readers and writers don't. Instead, they are represented by entries in the Navigator window, as in the above screenshot.

Adding Readers and Writers

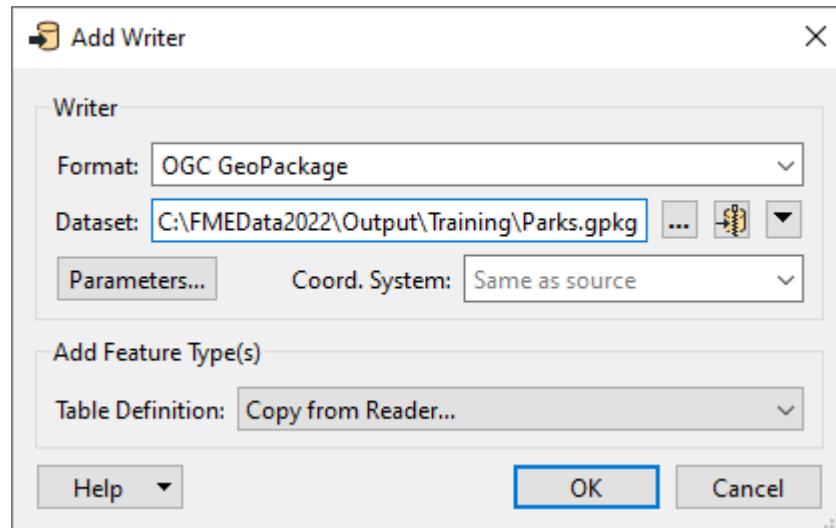
Additional readers or writers are added to a translation using the Quick Add menu:



...Or by selecting Readers > Add Reader (Writers > Add Writer) from the menu bar:



This action opens a dialog, similar to the Generate Workspace dialog, in which the parameters for the new reader or writer can be defined:



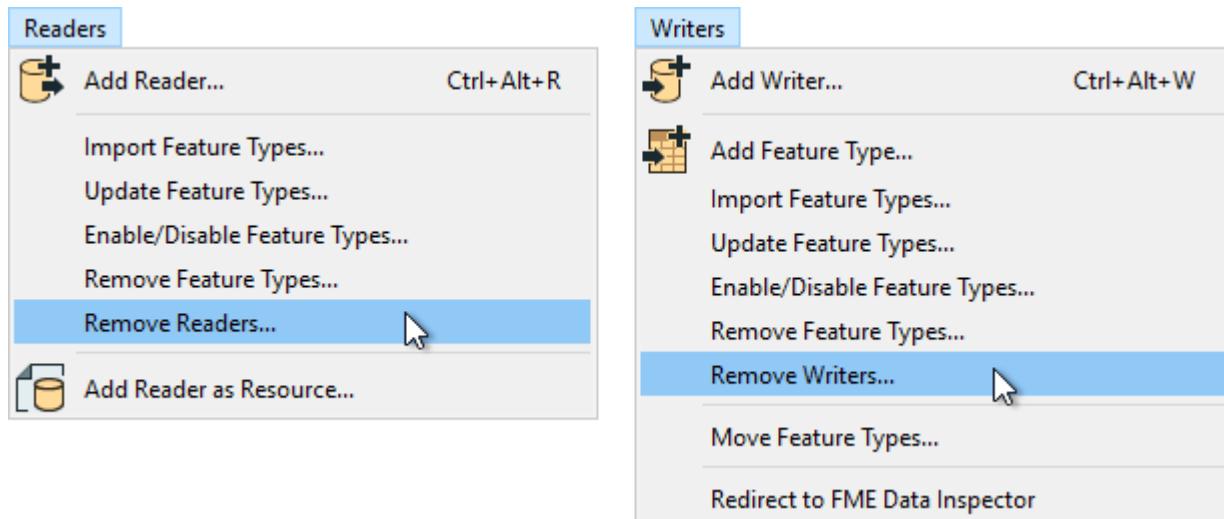
You can add as many readers and writers as you require in this way.



A reader can also be added by dragging a dataset from a filesystem explorer and dropping it onto the Workbench canvas.

Removing a Reader or Writer

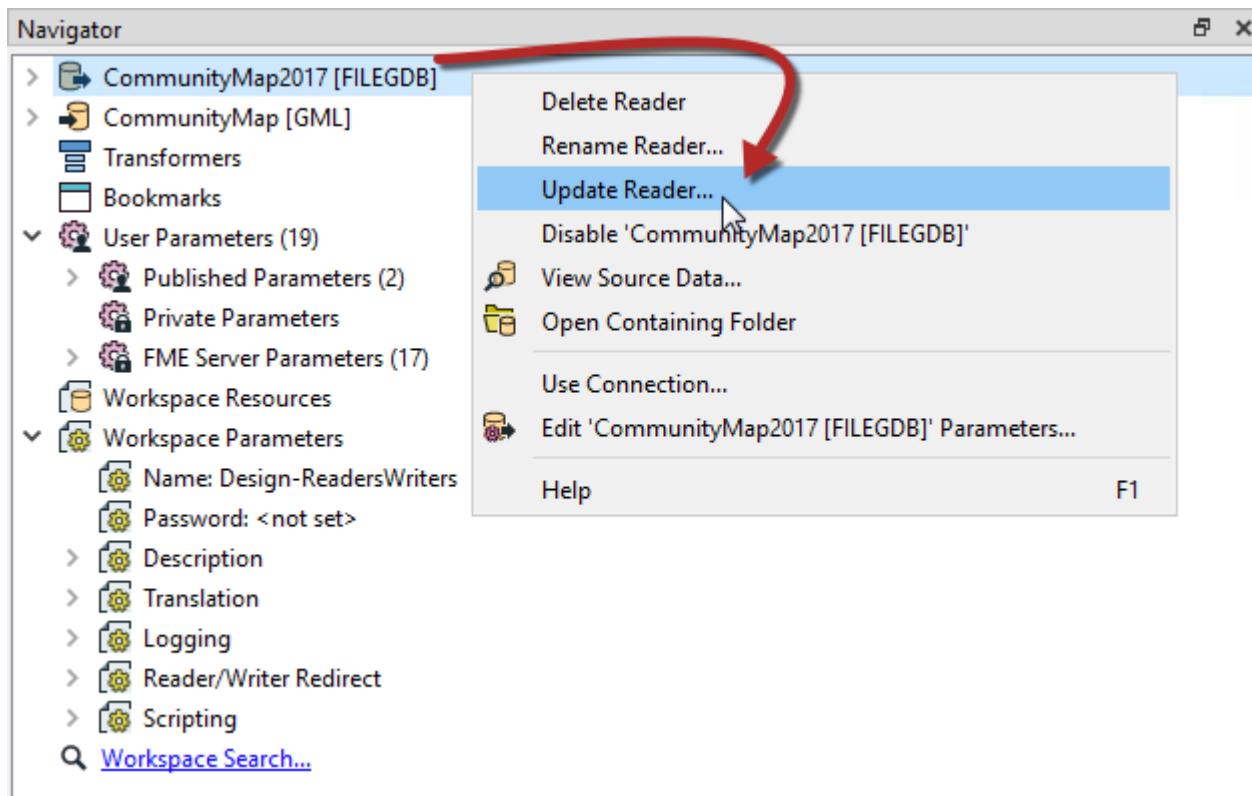
If a reader or writer is no longer required, then it can be removed very simply using options on the menu bar:



Alternatively, it's possible to right-click a reader/writer in the Navigator window and choose the Delete option.

Updating a Reader or Writer

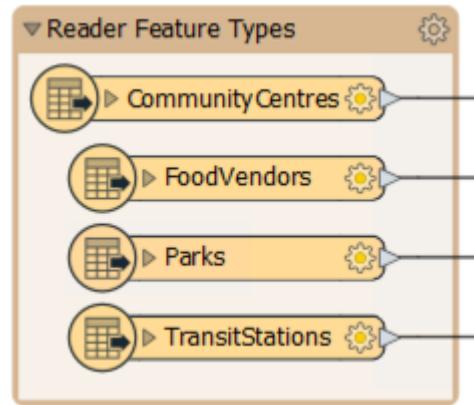
Readers and writers can be updated so that older workspaces have the speed and functionality available in a newer version of FME. You can update a reader/writer by right-clicking the reader/writer in the Navigator window and choosing the Update option:



For readers, this tool provides the option to either update the reader or to also update the list of feature types being read. This way the workspace can be updated if the source data changes. Another way to update feature types is Reader > Update Feature Types on the menu bar.

Reader Parameters

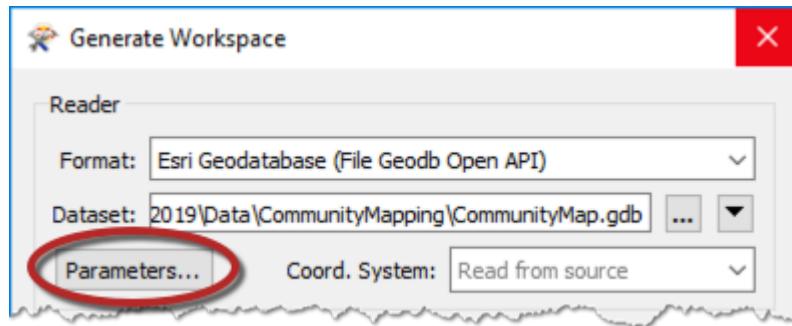
As we know, a workspace contains a reader to read a dataset, and each feature type in that dataset is shown in the workspace canvas:



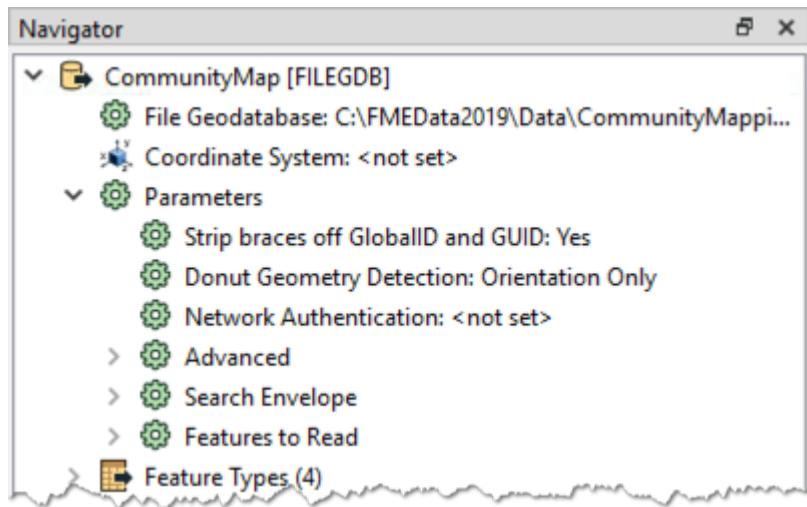
To control how that reader operates requires the use of **reader parameters**.

Finding Reader Parameters

Reader parameters can be located - and set - by clicking Parameters in the Generate Workspace or Add Reader dialogs:



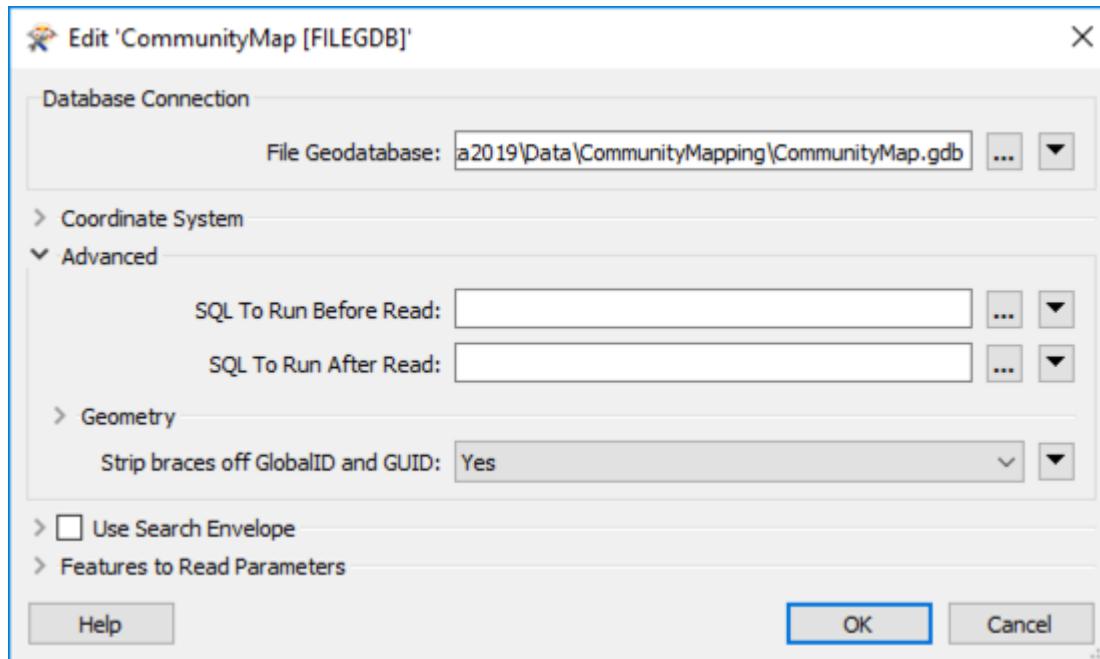
They can also be found in the Navigator window in Workbench:



Because parameters refer to specific components and characteristics of the related format, readers of different formats have a different set of control parameters.

Setting Reader Parameters

To edit a parameter in the Navigator window, double-click on any of the parameters. Doing so opens up a dialog where the parameter's value may be set:



Reader parameters control all feature types in the dataset. Think of it like brewing a pot of coffee. The strength control on the coffee machine affects all the cups that are poured.

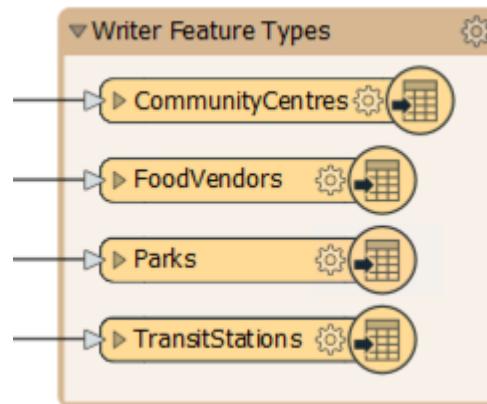


Because some reader parameters affect how feature types are generated (e.g. how do you want to slice a JSON or XML file?), they can only be set when you add the reader. If you set them incorrectly or want to change them, you have to delete the reader and add it again.

This behavior does not exist in writer parameters.

Writer Parameters

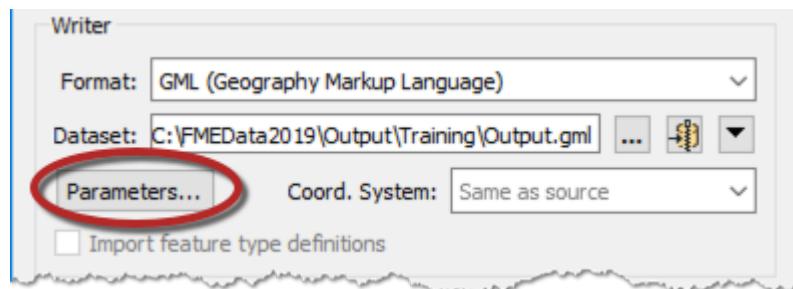
Like readers, we know a workspace contains a writer to write a dataset, and each feature type to be written is shown in the workspace canvas:



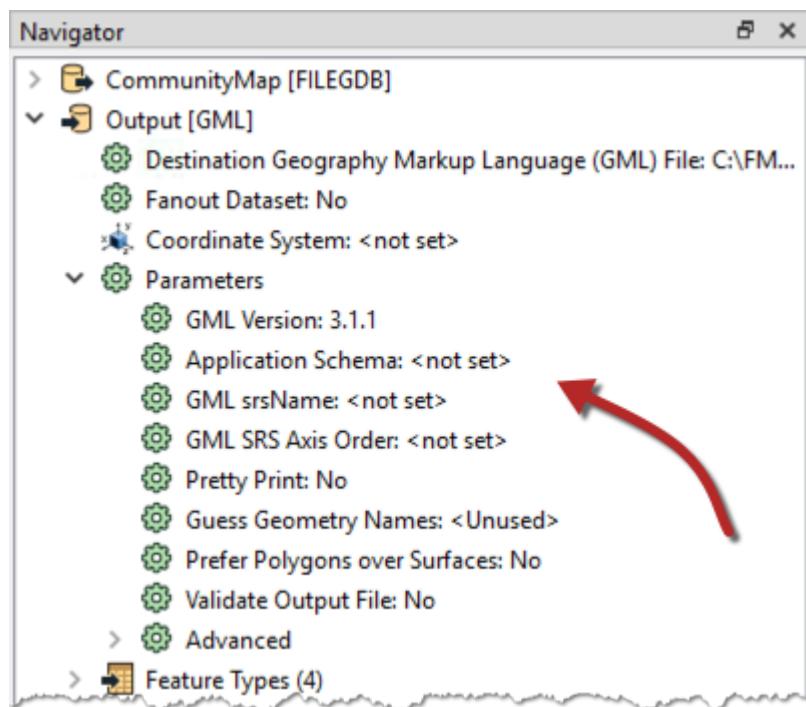
To control how that writer operates requires the use of **writer parameters**.

Finding Writer Parameters

Writer parameters can be located - and set - by clicking Parameters when a new workspace is being generated:



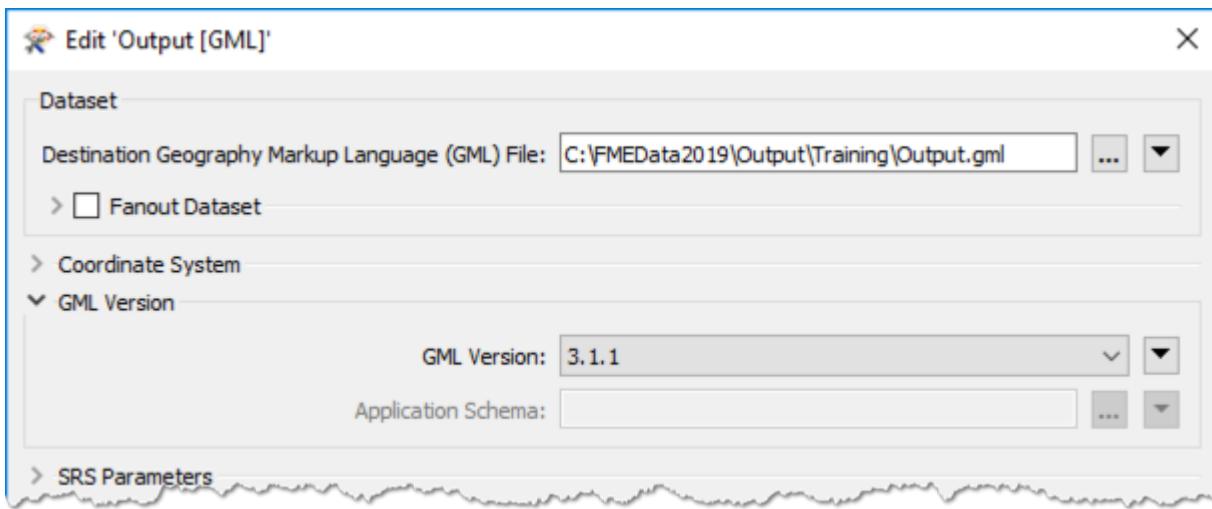
They too can also be found in the Navigator window in Workbench:



Because parameters refer to specific components and characteristics of the related format, writers of different formats have a different set of control parameters.

Setting Writer Parameters

To edit a parameter in the Navigator window, double-click on any of the parameters. Doing so opens up a dialog where the parameter's value may be set:



Like readers, writer parameters control all feature types in the dataset. For example, in the above screenshot, all feature types will be version 3.1.1.

But each reader and writer feature type does have its own settings (in the same way that each cup of coffee can be adjusted with cream and sugar). [Learn more in the documentation](#).

Cross-Operating System Compatibility

One important consideration in overall workspace design is cross-OS compatibility. Generally, FME is designed so your workspaces will run on any of our supported operating systems. However, there are a few best practices to keep in mind that could save you from problems in the future. These include:

- File paths
 - Because operating systems use different rules for file path construction, we recommend using the [Directory and File Pathnames reader](#) and FME [user parameters](#) to handle file paths wherever possible. Hardcoding paths is a sure-fire way to cause cross-OS compatibility issues
- Special characters
 - It's best practice to at a minimum, avoid using non-printable characters (including [ASCII control characters](#)) and `\/:*?"<>|`, null, and `/` in your file and folder names. For more advice, see this [thread](#).
- Case insensitivity
 - Databases you are using, including those used in conjunction with FME Server, might have different case expectations. For example, some databases allow `myObject` and `myobject` while others will reject these fields as a duplicate violation. Assuming case insensitivity is the safest method.

1 Which of the following is not a valid way to add a reader or writer to your workspace?

- A. Double-clicking the Navigator window
- B. Dragging and dropping a file onto the canvas
- C. Readers > Add Reader or Writers > Add Writer
- D. Quick Add

2 Updating readers can be used to update the reader version if you have upgraded FME and/or to reload the dataset to add new feature types.

- A.True
- B.False

3Some reader parameters affect how feature types are created. Therefore, they can only be set when adding the reader.

- A.True
- B.False

Check the Quiz to Earn 100 Points

Exercise: Use Multiple Readers and Writers

Learning Objectives

After completing this unit, you'll be able to:

- Add a reader to a workspace.
- Manage rejected features.
- Set a writer parameter.

Resources

- [Starting workspace](#)
- [Complete workspace](#)

Exercise

Here we continue with a project to redefine garbage collection schedules.

In the first exercise, we used various transformers to divide addresses into five separate groups. Now the task is to refine that work by estimating the number of residents per address based on the zone type it falls within:

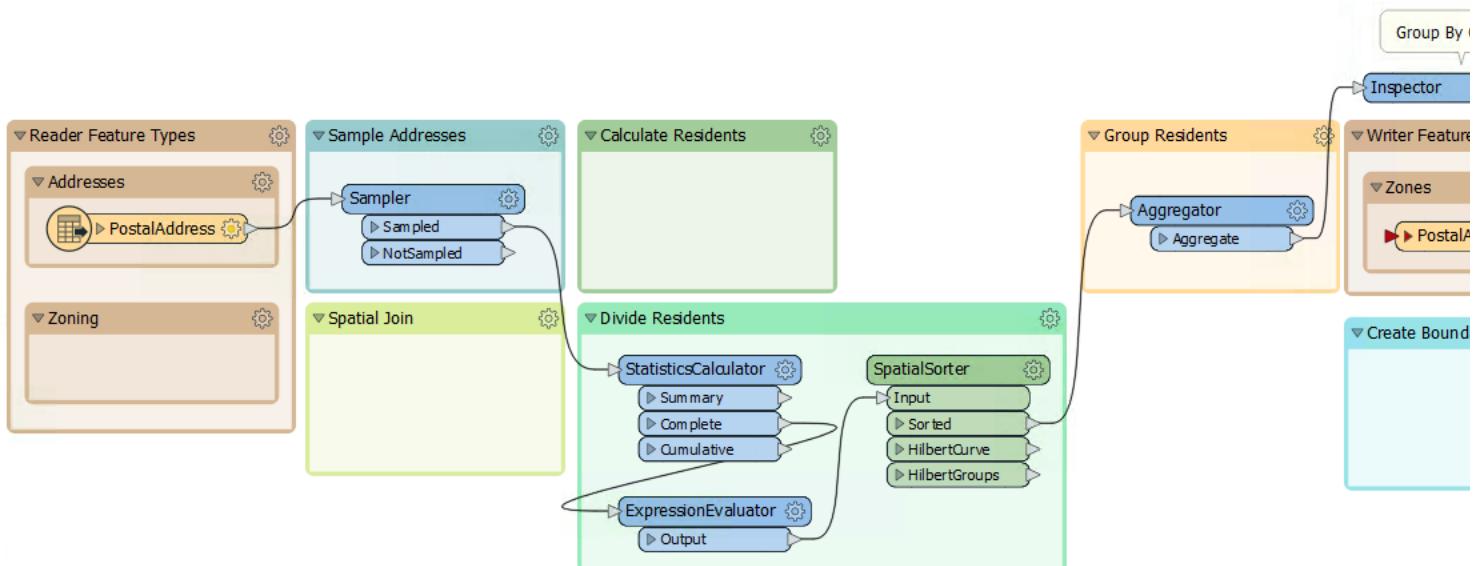
- Single-family residences: 2 adults
- Two-family residences: 4 adults
- Multi-family residences: 12 adults
- Comprehensive development zone: 8 adults
- Commercial properties: 1 adult

1) Open Workspace

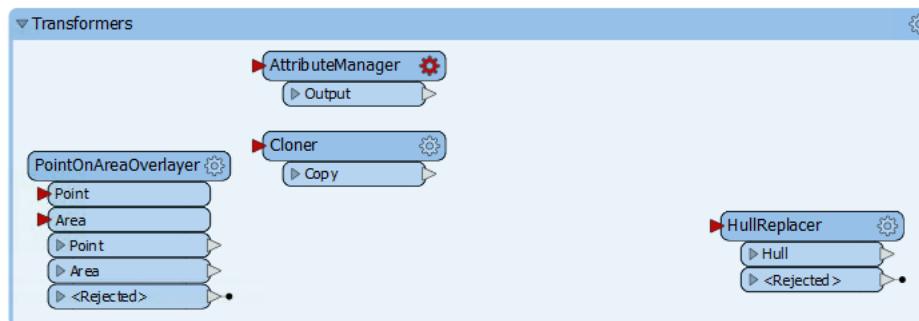
Open your workspace from the previous exercise.

Save a copy of the workspace with a new date or version number. For example, if you saved the previous workspace as GarbageCollection-05-18-2022.fmw, then make a copy named GarbageCollection-05-19-2022.fmw and open that for editing.

Alternatively, you can open the [starting workspace](#).



The remaining transformers in the "Transformers" bookmark are these:

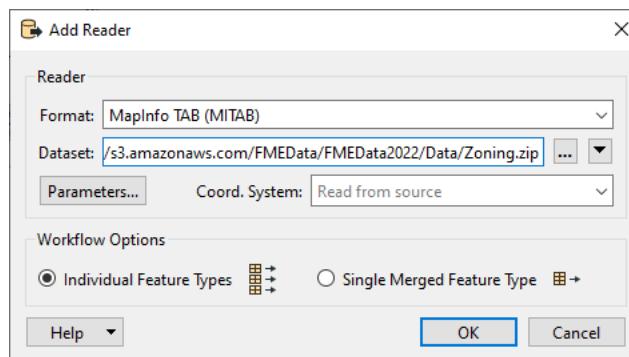


2) Add Reader

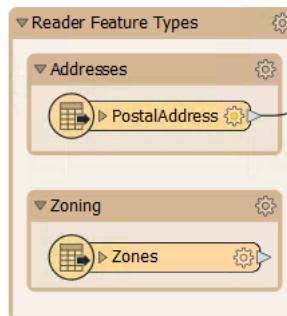
The first task here is to identify which planning zone each address falls inside. We need to read the zoning data and carry out a spatial join. To read a new dataset of data in a different format requires a new reader.

So, select Readers > Add Reader from the menu bar. When prompted enter the following parameters:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	https://s3.amazonaws.com/FMEData/FMEData2022/Data/Zoning.zip



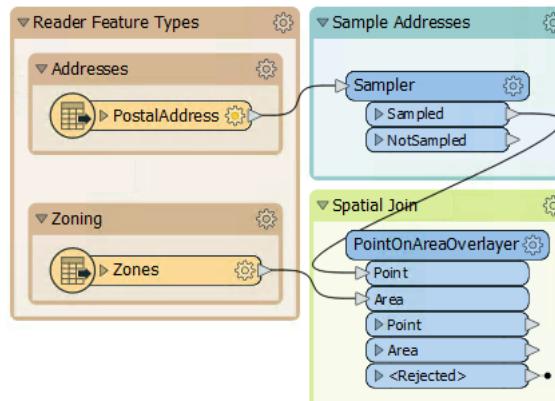
A reader is added to the Navigator window and a feature type to the canvas. Move the feature type into the Zoning bookmark:



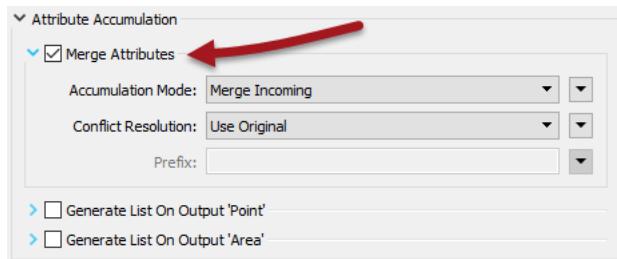
3) Create Spatial Join

To carry out a spatial join we'll use a PointOnAreaOverlayer transformer; this is a type of join called Point-in-Polygon.

So, move the PointOnAreaOverlayer transformer from the "Transformers" bookmark to the "Spatial Join" bookmark. Connect the newly added Zoning data to the Area port and the output from the Sampler to the Point port:



Inspect the PointOnAreaOverlayer parameters. Expand the Attribute Accumulation section and check Merge Attributes:



This transformer is the first we've used that has a live <Rejected> port. For now, we'll leave it to stop the translation, since during testing we want to know about anything that causes a failure of the transformer.

Run the translation, ignore the Invalid Transformer Parameters dialog that pops up and click Run. This dialog pops up because we have previously run the translation to the Aggregator, but now we have broken that connection. We will fix it shortly.

Click on the PointOnAreaOverlayer:Point output port to view the data in the Visual Preview window. View both the Graphics and Table view. The overlay and attribute merging should cause each address to be given a zone name and category, click on any of the zones to confirm this.

4) Calculate Residents

The next step is to set how many residents live at a certain address according to its zoning type.

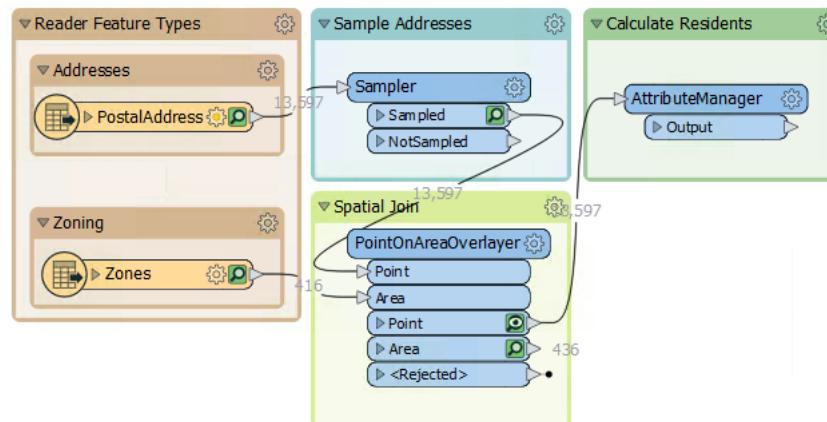
We know that:

Zone Begins With	Zone Type	Residents
RS	Single Family	2
RT	Two Family	4
RM	Multiple Family	12
CD	Comprehensive	8
C	Commercial	1

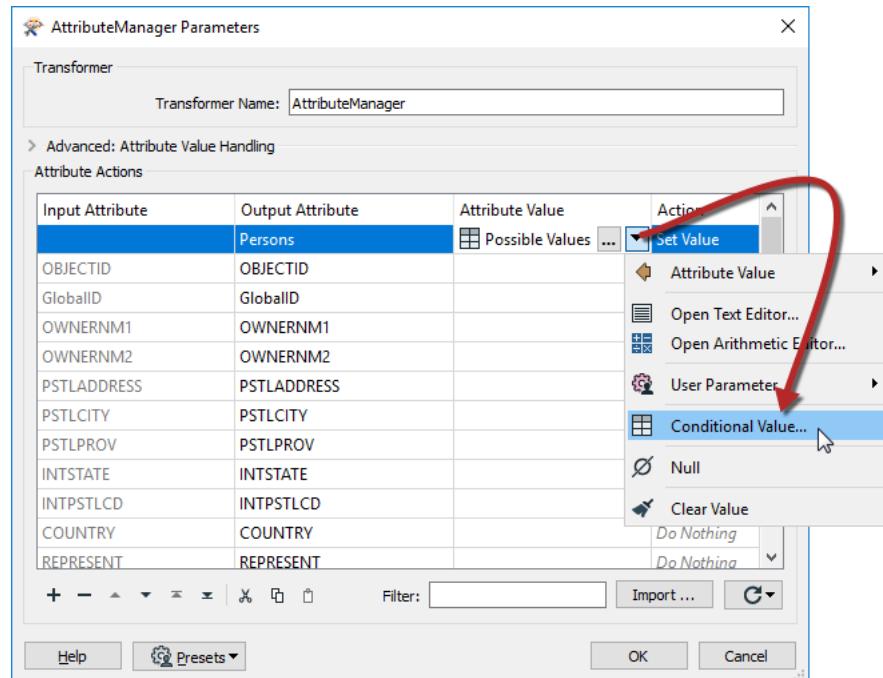
For example, zones RS-1, RS-2, RS-3 are all single-family zones, and we assume a total of two adults per address. This assumption makes it slightly more complicated because we need to match a zone type using a "begins with" string comparison.

This match can be done using an AttributeManager with **Conditional Values**.

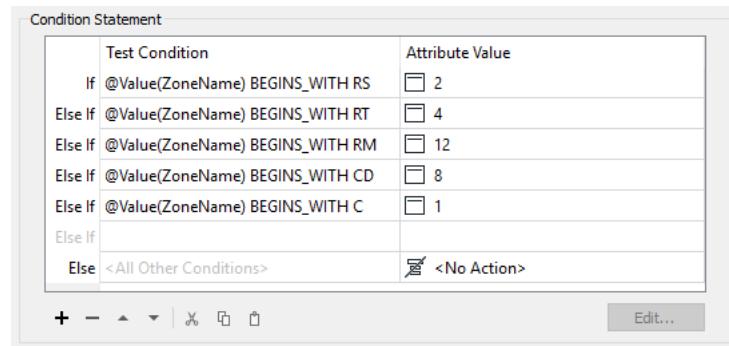
This step is slightly complex, but luckily the AttributeManager inside the "Transformers" bookmark is already set up for this purpose. So move the AttributeManager into the "Calculate Residents" bookmark and connect it to the PointOnAreaOverlayer:Point output port:



If you are interested in what Conditional Values look like, open the parameters dialog for the AttributeManager and click the drop-down arrow in the Attribute Value field for the Persons attribute. Choose Conditional Value:



Doing so opens a Tester-like dialog with multiple conditions that test for each zone type, and an attribute value to set them to:



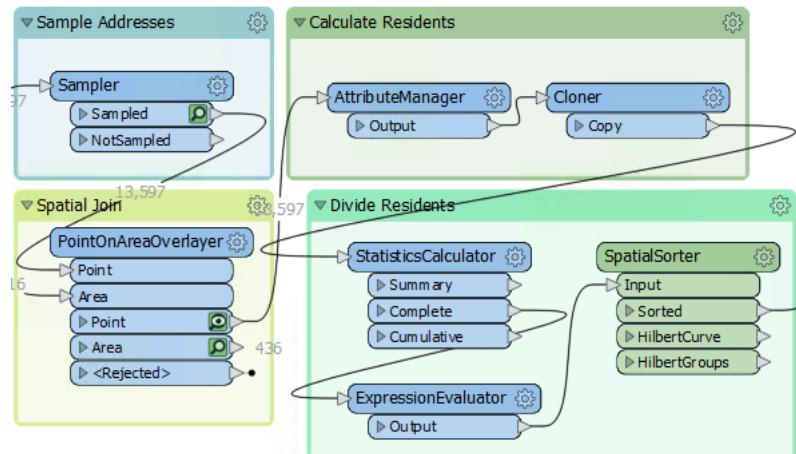
If you want to challenge yourself, try adding a blank AttributeManager instead of moving the existing one. Then create the conditional statements to match the image above.

5) Create Residents

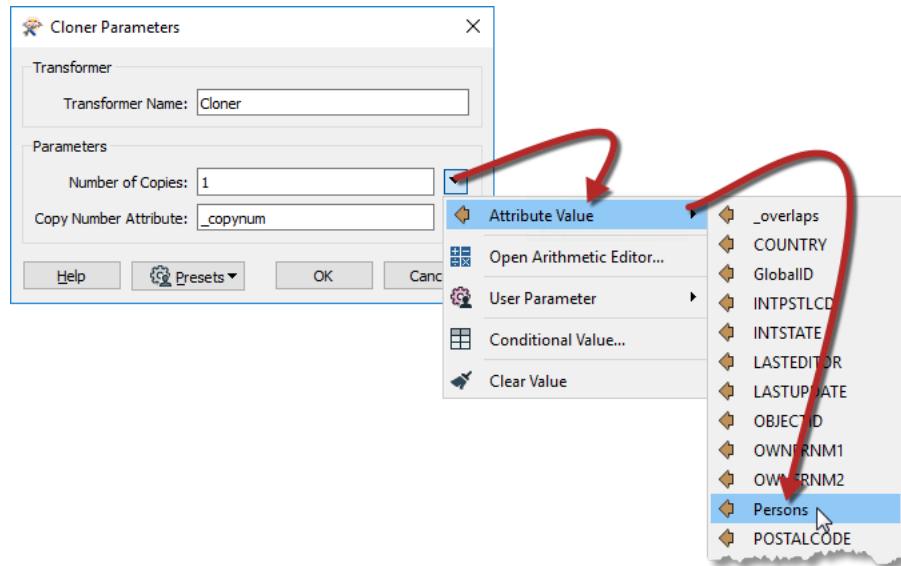
We now know (or have approximated) the number of residents per address. However, we must use that number in a way that will affect the output. The simplest way to do this is to create multiple copies of each address, one for each resident.

For example, for an address with eight residents, we'll create eight address points.

We can do this very simply with a Cloner transformer. So, move the Cloner transformer from the "Transformers" bookmark to the "Calculate Residents" bookmark. Connect the AttributeManager to the Cloner's input and its output to the StatisticsCalculator:



Inspect the Cloner parameters. For the Number of Copies parameter, click the drop-down arrow and choose Attribute Value > Persons:



Doing so will create `<Persons>` copies of the original addresses (note that the transformer doesn't output the original as well, so the output is `<Persons>` features, not `<Persons>+1`).

6) Run Translation

Make sure an Inspector is still attached to the Aggregator transformer and run the translation. The translation will fail with the error message:

`Cloner_<Rejected>: Termination Message: 'Cloner output a <Rejected> feature.'`

[Copy](#)

The translation failed because addresses without a resident (for example, Industrial) have no Persons attribute and are being rejected by the Cloner transformer. The `<Rejected>` port is still set up to stop the translation, and so we get this error.

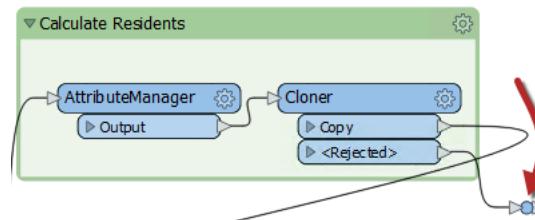
There are various choices to handle this. We could:

1. Change the Workspace parameter **Rejected Feature Handling** to *Continue Translation*
2. Add a transformer to handle the Cloner's rejected features
3. Set the Conditional Values to give a value of zero, instead of not including a value at all

Setting the Conditional Values would be the best solution to deal with the problem directly. But there might be other causes for rejected data, and we want to deal with that without having the translation stopped.

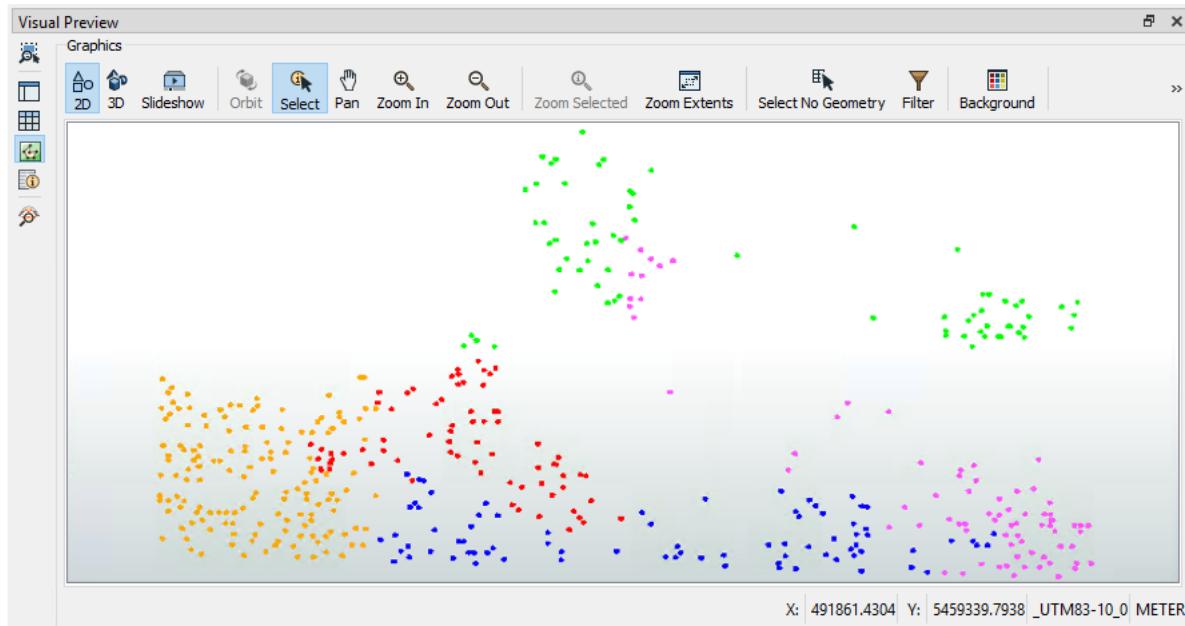
Setting the Rejected Feature Handling parameter means all `<Rejected>` ports would ignore rejected output. This setting might be useful in a production workspace, but in testing, we would probably want to stop the translation so that we can be aware of issues immediately.

So for us, the better solution is to add a transformer to the Cloner `<Rejected>` port. We don't need to inspect or log these features because we know that they will exist. So, connect the `<Rejected>` port to a small transformer called a Junction:



This Junction will handle the rejected output, but quietly drop it without further fuss.

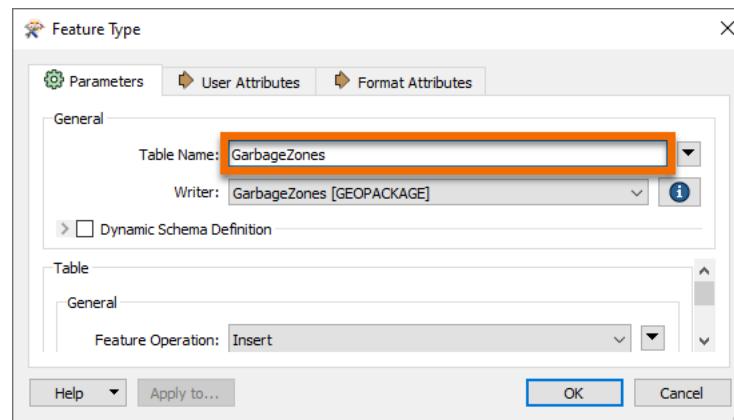
Re-run the translation. The output should be five groups of point features again, but in a different pattern to the end of the previous exercise:



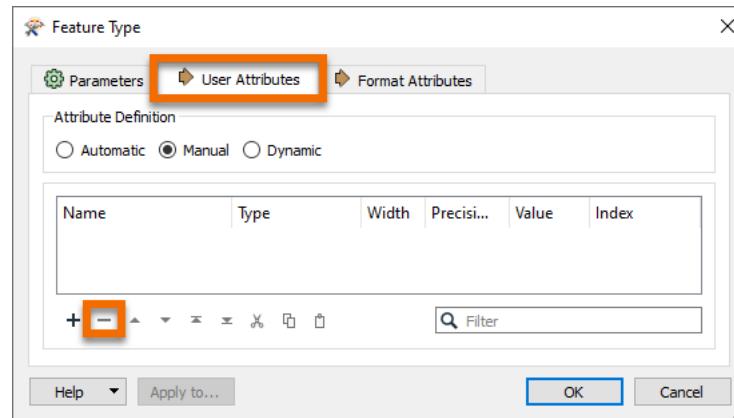
7) Write the Output

Now to write some output. The simplest method is to connect the Aggregator output to the PostalAddress output feature type and re-run the workspace.

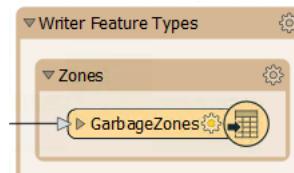
However, it would also be useful to rename the output feature type and remove all of its attributes, since they are from the reader dataset and don't apply here. So open the writer feature type parameters dialog. In the Parameters tab rename the feature type to GarbageZones.



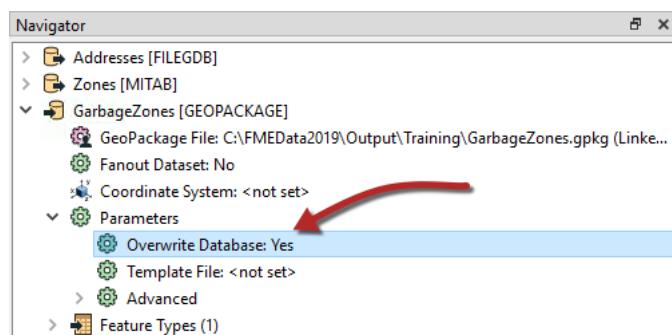
In the User Attributes tab, remove all of the attributes that are being written.



Your writer feature type should look something like this:



Also, we should change the GeoPackage writer parameter Overwrite Database to overwrite the database each time we run the workspace. To do this, find the GarbageZones [GEOPACKAGE] writer in the Navigator, expand the Parameters, double-click Overwrite Database, and then check the box and click Ok:



Enabling this parameter means we won't accumulate more and more results in the same dataset.

Congratulations. Now you have a dataset of addresses grouped by their new garbage collection zone. In the next exercise, we will turn these points into non-overlapping polygons designating the zones.

If you completed this workspace and then realized you had an additional (separate) MapInfo TAB file of zones you needed to include, which of the following steps would not allow you to add this data to your workspace?

- A.Add the new MapInfo TAB file to your existing MapInfo TAB reader source path parameter.
- B.Add a new MapInfo TAB reader with the new file as the source path parameter.
- C.Combine the MapInfo TAB files into a single file (in FME or MapInfo) and read that instead.
- D.Update the existing MapInfo TAB reader to read the new feature type.

—2Why was changing Rejected Feature Handling to Continue Translation not the best choice while testing this workspace?

- A.If the Cloner was rejecting address features for a reason other than not being within a residential zone, for example, not being within a zone at all.
- B.It would slow the workspace down too much due to increased performance requirements.
- C.Changing the parameter takes too long while testing a workspace.
- D.It would have introduced compression errors into the data.

—3What would have happened if we did not enable the Overwrite Database writer parameter?

- A.The output would be created correctly.
- B.An error would occur.
- C.Writing would stall because of a file read lock on the writer dataset.
- D.The new garbage zone polygons would be appended/inserted to the Geopackage, creating duplicates.

—4Approximately how many residents live in each garbage collection zone? You must remove the Sampler, rerun the finished workspace, and inspect the Aggregator:Aggregate port to find out.

- A.11,616
- B.29,234
- C.58,079
- D.58,185
- E.94,055

[Check the Quiz to Earn 200 Points](#)

Test Your Workspace With Partial Runs

Learning Objectives

After completing this unit, you'll be able to:

- Use partial runs to test your workspace.
- Disable objects to test your workspace.

Workspace Testing Techniques

When developing a workspace incrementally, or making edits to a structured workspace, a lot of testing will take place. Each time a change is made the author will wish to test the change, and this process of change/test repeats itself again and again.

To effectively and efficiently carry out this process, there are a number of tools available in FME Workbench.

The first set of tools is for inspecting data as it changes. The second set of tools is for defining which parts of the workspace need to be tested.

Partial Runs

A partial run is when only one section of a workspace is carried out. One way to do this is to disable objects in the canvas to only run certain enabled sections. Another method is to use a tool called Partial Runs, which is represented by pop-up options when a workspace is run with caching turned on.

The technique you use will depend on how large the workspace is, and how much of it you need to run. You may use one technique or the other - or you may use both!

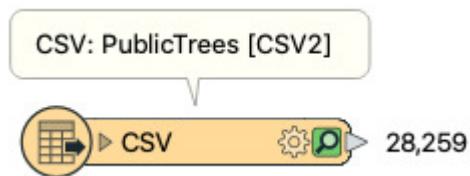
The technical aspect of partial runs is covered in other modules (see [Viewing Part of Your Workflow](#)). However, using partial runs is an important part of workspace testing. You should use partial runs to develop your workspace incrementally, testing each new section to ensure it works.

A partial run is particularly useful in avoiding re-reading data from its source; especially when the data comes from a slow, remote location such as a web service. Creating a cache can help increase the development of your workspace.

Finally, caches can be saved with the workspace when it is saved as a template (File > Save As Template, enable Include Feature Caches). That means the workspace can be re-run using the caches from a previous session or even from another author!

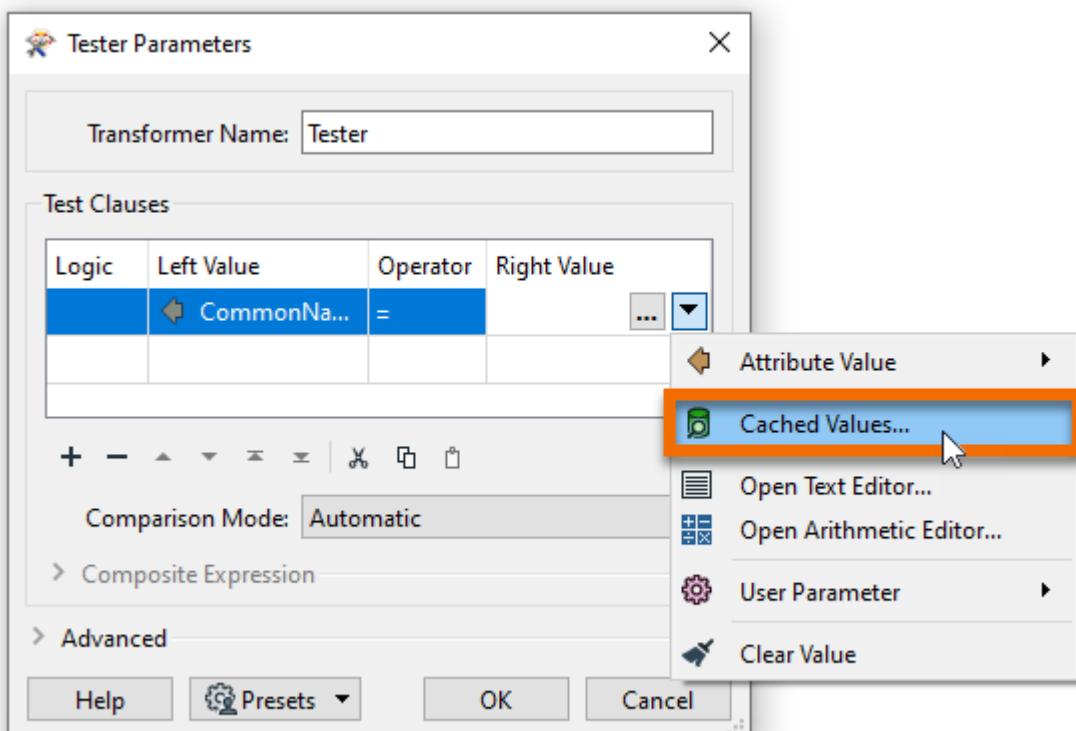
Data-Aware Transformer Dialogs

Starting in FME 2021.0, when you have feature caches in your workspace, you can browse unique values to make filling in transformers easier. The first step is to ensure you have feature caches:

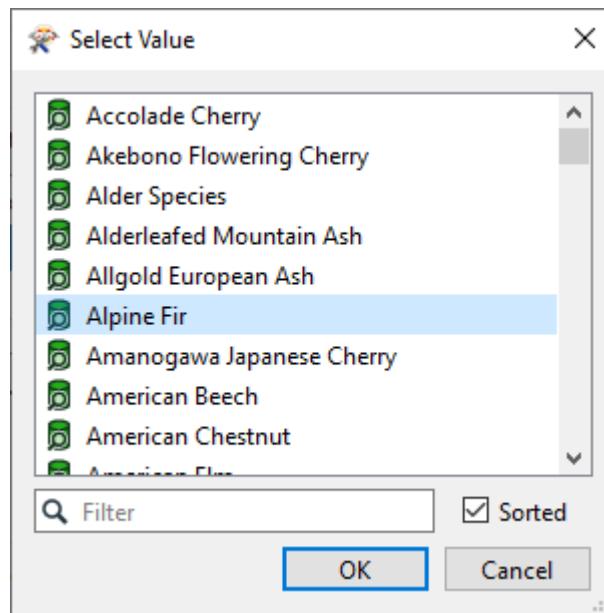


Then add a Tester, TestFilter, AttributeExposer, or any other transformer that uses test clauses. The AttributeManager is also supported. Once connected, open the parameters. In the parameters, select the Left Value attribute. For this example, I'm using Common Name. Then select your Operator.

To retrieve the cached attributes, click on the drop-down arrow for Right Value, then select Cached Values. If there are only a few values for the attribute, a drop-down list will be available. If there are lots of values, a popup list will appear.



Then select the value you want to test for.



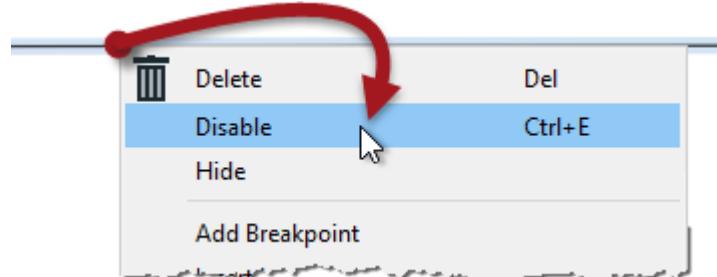
Once the value is selected, click OK to create another test or click OK again to accept the parameters.

You can continue to build your workspace as normal. To use this feature again later in the workspace, remember to run with feature caching up to the point where the transformer is about to be added.

Disabled Objects

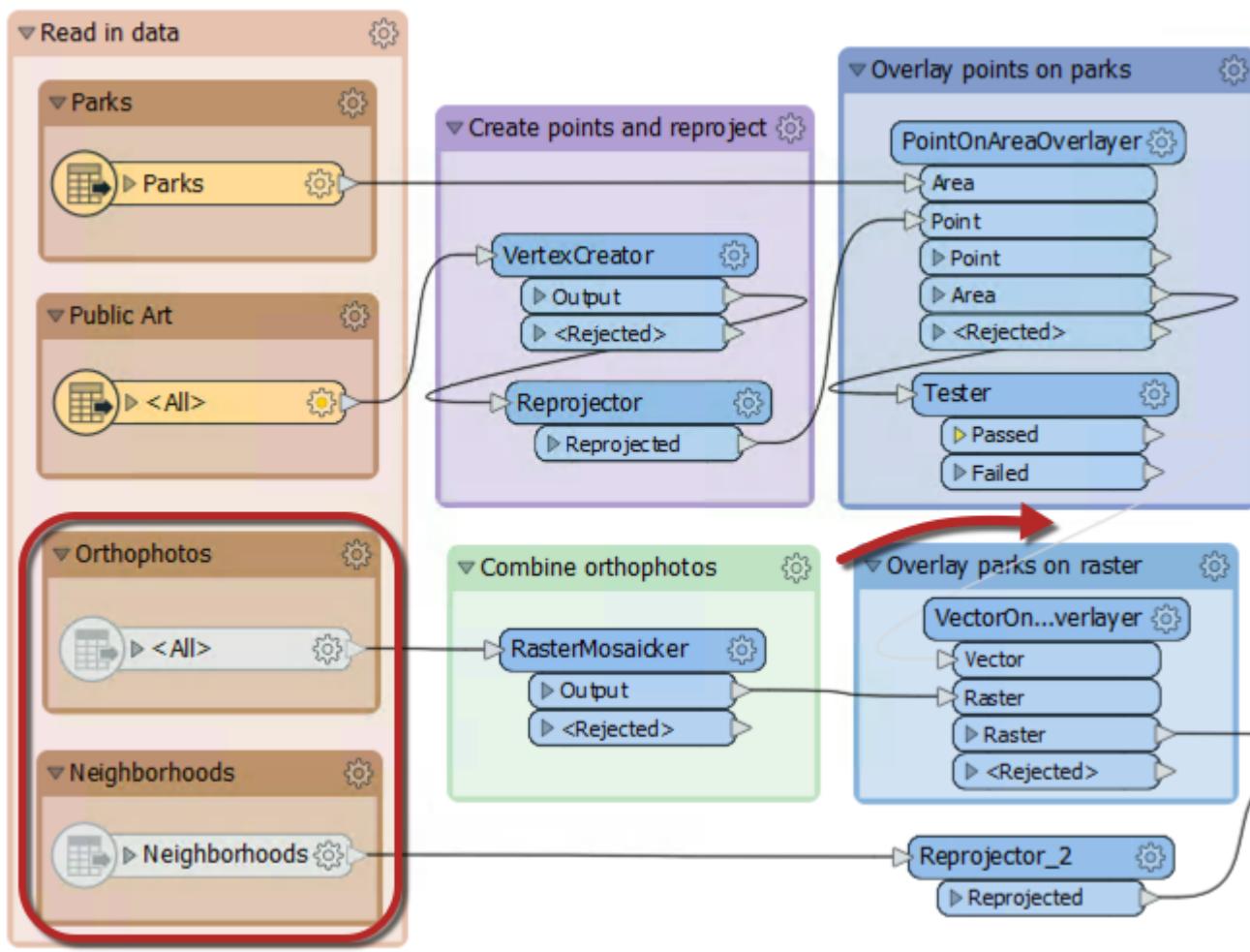
If designed correctly, a large workspace should be made up of small sections. Isolating a section (or part of a section) for testing is possible by disabling connections to all other components.

An object (connection, transformer, or feature type) is disabled by right-clicking it and choosing the option to Disable (or selecting it and using the shortcut Ctrl+E):



A disabled connection is rendered inoperative in much the same way as if it had been deleted, and no features will pass through. The same disabling can be done to other canvas objects such as transformers and feature types. Even a reader/writer can be disabled through the Navigator window.

Here an author has disabled two connections (both from the Tester:Passed port) and two feature types:



With that setup, the top part of the workspace will operate up until (and including) the Tester. The bottom portion will not run at all. No data will emerge from the disabled feature types, and no data is passed to it from the Tester.

With caching turned on, the author can inspect part of the workspace without having to run the entire translation. This feature is a significant advantage when (like here) the disabled section takes up most of the overall processing time.

1 Which of the following is not a valid reason to use partial runs when developing a workspace?

- A. Using incremental development to test each section of your workspace as it is built.
- B. Avoiding slow workspace development waiting to read in source data on every run.
- C. Feature caching mode automatically detects changes in schema for you and issues a warning.
- D. You can save caches with a template to share with colleagues while troubleshooting or validating workflows.

2 Which of the following scenarios would benefit from using a data-aware transformer dialog?

- A. You need to configure an AttributeManager to create some new attributes with no values.
- B. You need to configure an AttributeRangeFilter to separate features into several streams based on arbitrary band values.
- C. You need to configure a Tester to pull out a few features matching some specific ID values.
- D. You need to configure a TestFilter to filter features into different kinds of empty or missing values.

3 You can increase workspace development speed by disabling feature types to prevent reading and writing data while you are testing other sections of the workspace.

- A. True

B.False

Check the Quiz to Earn 50 Points

Exercise: Use Partial Runs To Test Your Workspace

Learning Objectives

After completing this unit, you'll be able to:

- Use partial runs for testing.
- Disable feature caching when running the final version of a workspace.

Resources

- [Starting workspace](#)
- [Complete workspace](#)

Exercise

Here we conclude our project to redefine garbage collection schedules.

In the first two exercises, we used various transformers to divide addresses into five separate groups, according to zoning type. Then we wrote the data to Geopackage.

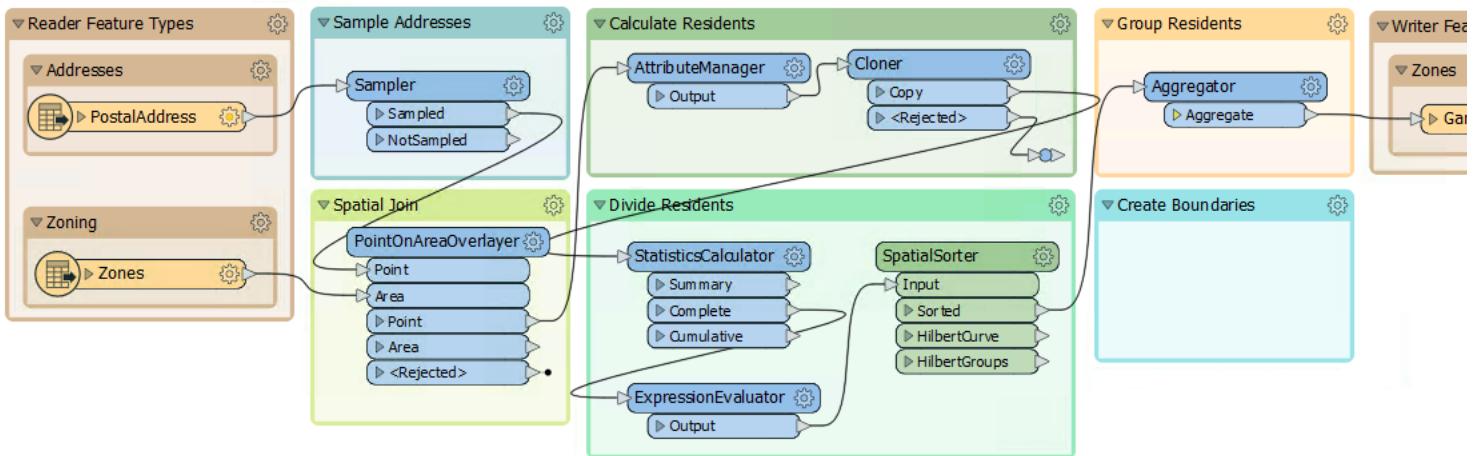
Now the task is to replace the groups of point features with a polygon boundary.

1) Open Workspace

Open your workspace from the previous exercise.

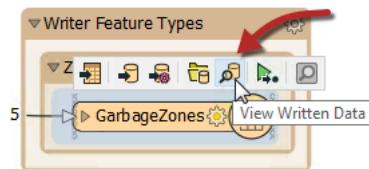
If you gave that workspace a date or version number in its name, then you should make a copy of the workspace with a new one. For example, if you saved it to GarbageCollection-05-19-2022.fmw then make a copy named GarbageCollection-05-20-2022.fmw and open that for editing.

Alternatively, you can open the [starting workspace](#).



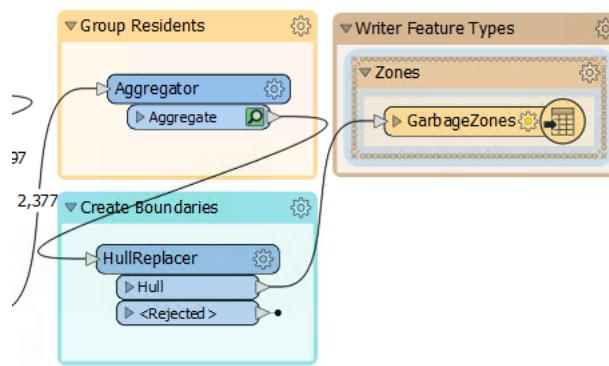
2) Run the Workspace

Run the workspace to finish writing out the data. You can inspect the output dataset if you desire by clicking on the GarbageZones writer feature type and then clicking on the View Written Data button in the popup menu:



3) Add a HullReplacer Transformer

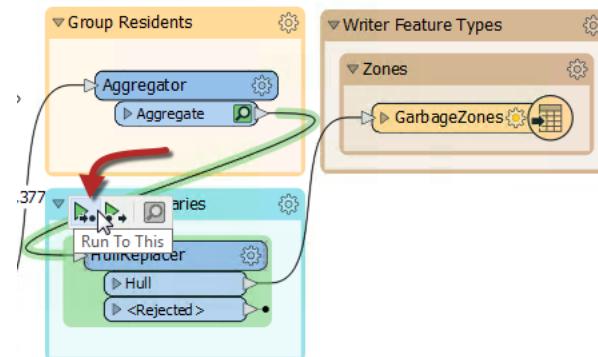
Move the HullReplacer transformer from the "Transformers" bookmark into the "Create Boundaries" bookmark. Connect it between the Aggregator and writer feature type:



Notice how the HullReplacer has no cache because it is newly placed.

4) Re-Run the Workspace

Now let's re-run the workspace. But rather than re-write the output data, we can run just the new transformer we just added. Click on the HullReplacer transformer and on the icons that pop up, click Run To This:

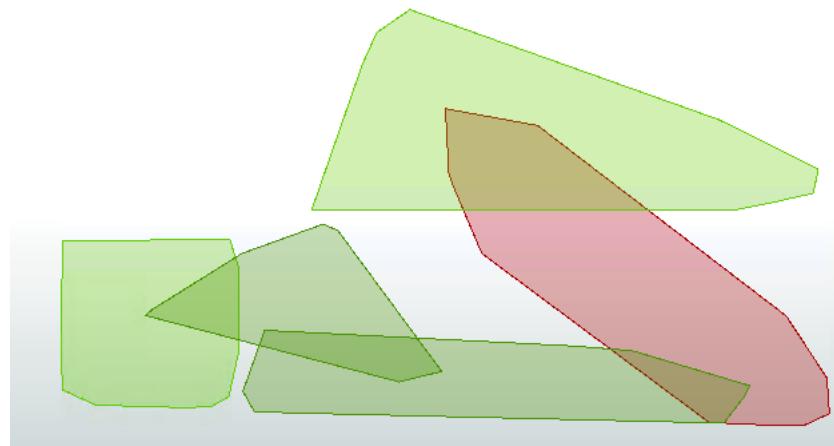


Notice how hovering shows what parts of the workspace will be run. Since we already have features cached up to the Aggregator (assuming you haven't closed the workspace since it was last run), only the section between the Aggregator and the HullReplacer will be run.



Using the Run To This option is a good method to check your translation before writing the data out, especially if you are writing to a database or an online source.

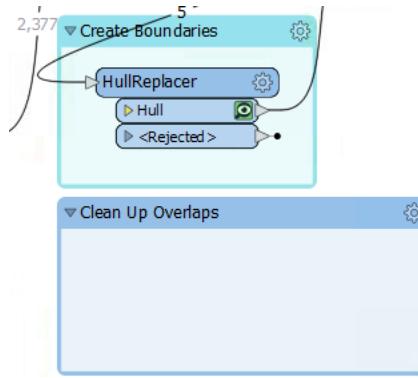
Click on the cached features on the HullReplacer:Hull output port to confirm the data. The output now includes polygons, to prove that the translation has functioned correctly:



5) Clean Up Overlaps

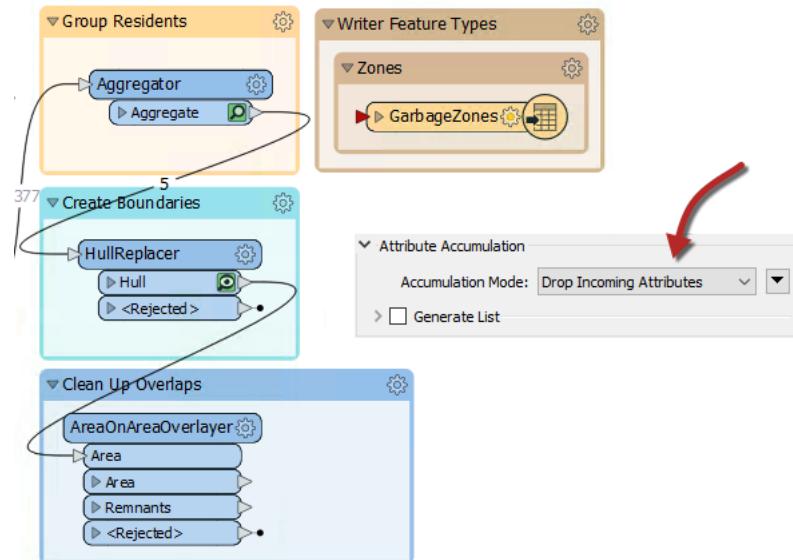
The problem with the output is that all of the polygons overlap to some extent. That needs to be fixed so that there are no overlaps. What's more, we should check which zone an overlap belongs to by seeing which group contains most of its addresses.

Because this is unexpected, we don't have an area of the workspace set aside yet. Add a new bookmark (or move the now-empty Transformers bookmark) and name it Clean Up Overlaps:



6) Add an AreaOnAreaOverlayer Transformer

Overlaps can be dissected using the AreaOnAreaOverlayer transformer, so add one of these to the new bookmark, connected to the HullReplacer transformer. Check the parameters and set the **Attribute Accumulation Mode** to *Drop Incoming Attributes*.



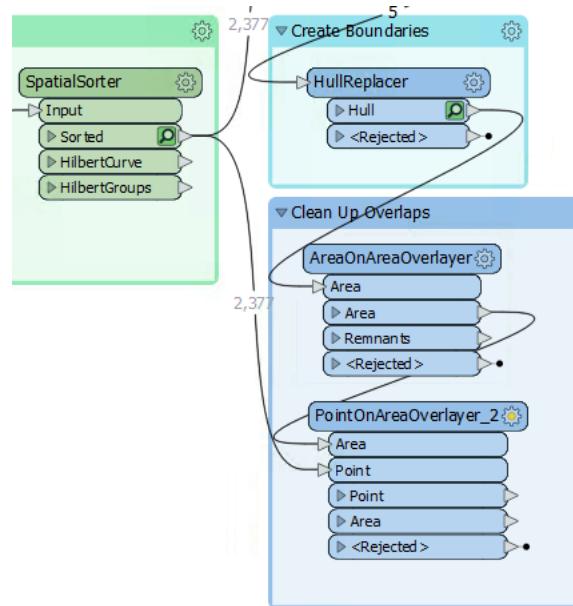
7) Add a PointOnAreaOverlayer Transformer

The overlaps are now separate features, but we do not yet know to which area they should be assigned. It should be the one with most addresses; for example, if an overlap contains 31 addresses from group one, and 52 addresses from group two, then it should be assigned to the group two polygon.

We can start on this by using a PointOnAreaOverlayer. This transformer will let us create a list of which addresses an overlap contains.

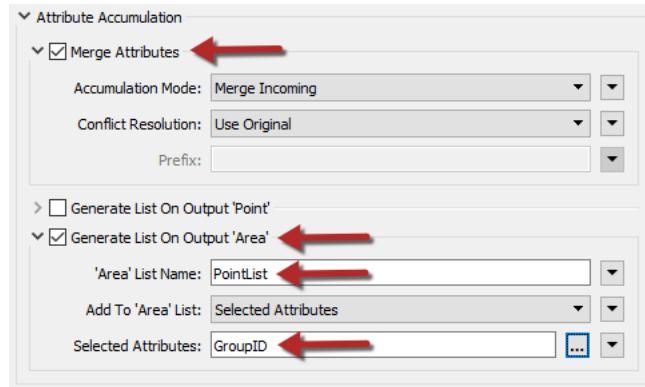
So add a PointOnAreaOverlayer transformer. The area features will be the output from the AreaOnAreaOverlayer.

The point features should be a copy of the addresses. The simplest way to get these is to make a second connection from the SpatialSorter:



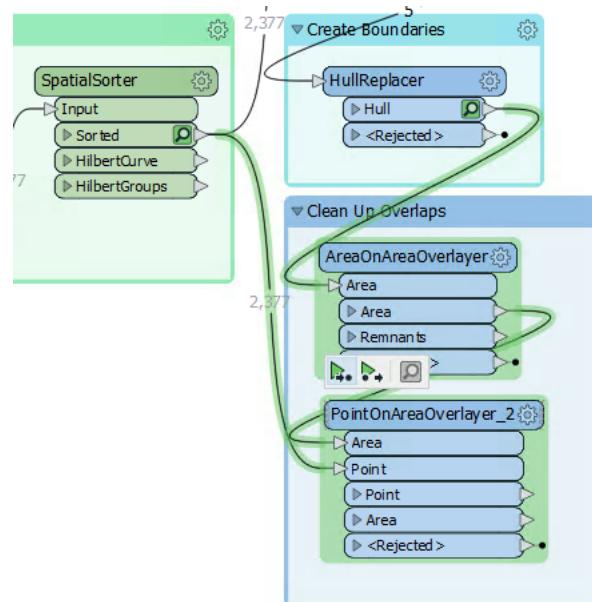
Inspect the parameters. Under Attribute Accumulation, set the following parameters:

Merge Attributes	<input checked="" type="checkbox"/>	Yes
Generate List on Output 'Area'	<input checked="" type="checkbox"/>	Yes
'Area' List Name	PointList	
Selected Attributes	GroupID	



Doing so will create an FME list attribute. A list attribute is an attribute with multiple values. Here the list will contain a record of the point features (and their GroupID values) that fall inside a polygon.

Confirm this works correctly by running the workspace at the new PointOnAreaOverlayer. Notice how the translation pulls data from two caches; the AreaOnAreaOverlayer and SpatialSorter transformers:



Inspect the PointOnAreaOverlayer:Area output port features. Selecting a feature will show (in the Feature Information window) the list attribute and all of its values:

Property	Value
mapinfo_centroid_x (64 bit real)	486537.52950604144
mapinfo_centroid_y (64 bit real)	5456428.919675885
mapinfo_pen_color (32 bit unsigned integer)	8322304
mapinfo_pen_pattern (32 bit integer)	2
mapinfo_pen_width (32 bit integer)	1
multi_reader_full_id (32 bit integer)	0
multi_reader_id (32 bit integer)	0
multi_reader_keyword (string)	FILEGDB_1
multi_reader_type (string)	FILEGDB
OBJECTID (32 bit unsigned integer)	12575
OWNERNM1 (encoded: UTF-16LE)	Lidia Stiffler
OWNERNM2 (encoded: UTF-16LE)	Kevin Shira
Persons (64 bit integer)	2
PointList(0).GroupID (32 bit integer)	4
PointList(1).GroupID (32 bit integer)	4
PointList(2).GroupID (32 bit integer)	4
PointList(3).GroupID (32 bit integer)	4
PointList(4).GroupID (32 bit integer)	4
PointList(5).GroupID (32 bit integer)	4
PointList(6).GroupID (32 bit integer)	4
PointList(7).GroupID (32 bit integer)	4
PointList(8).GroupID (32 bit integer)	4
PointList(9).GroupID (32 bit integer)	4
PointList(10).GroupID (32 bit integer)	4
PointList(11).GroupID (32 bit integer)	4
PointList(12).GroupID (32 bit integer)	4

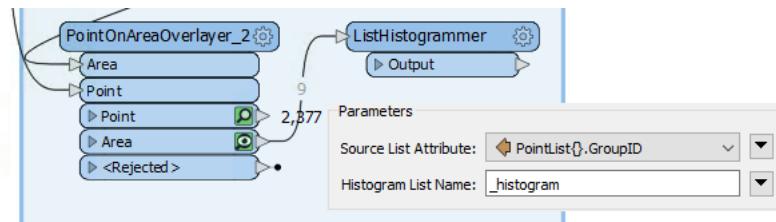
8) Add a ListHistogrammer Transformer

To count the most frequent GroupID for each list on an overlap's we'll use the ListHistogrammer transformer.

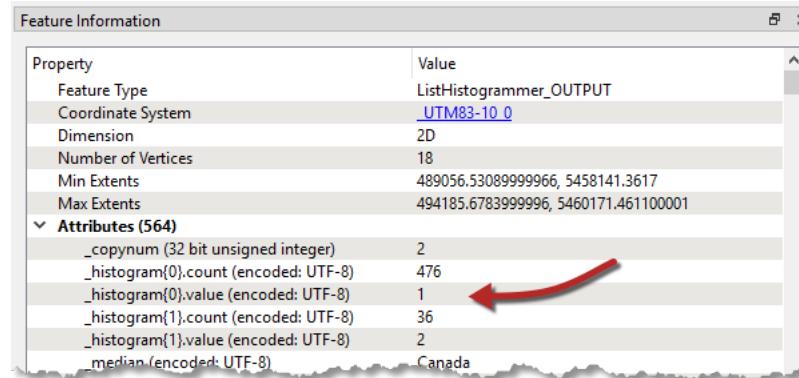


This is not a commonly used transformer, so don't worry if you weren't aware of it, or if you are concerned about the large number of transformers available in FME. You will learn more about these transformers with practice. For now the ability to use partial runs is much more important.

Place a ListHistogrammer transformer connected to the PointOnAreaOverlayer:Area output port. Inspect the parameters and select PointList{ }.GroupID as the source attribute:

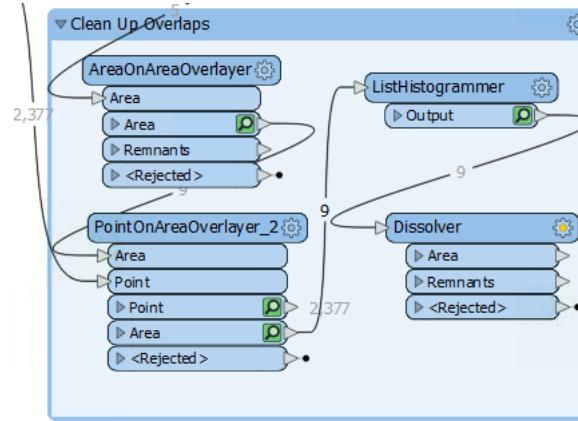


Re-run the workspace (from the ListHistogrammer) and inspect the results. Notice that a new list attribute is created; a list of the number of different GroupID values with the most frequent GroupID at the top of the list. So we merely need to use that GroupID to merge areas.

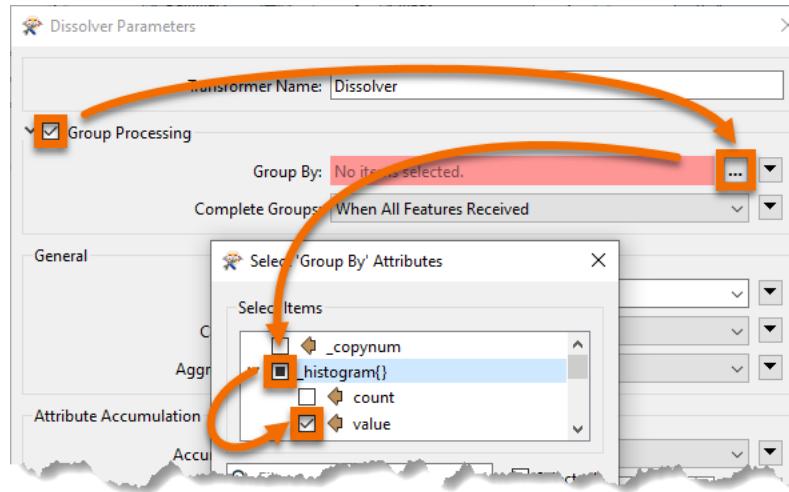


9) Add a Dissolver Transformer

Finally add a Dissolver transformer to merge the features together. Connect the Dissolver to the ListHistogrammer output port:



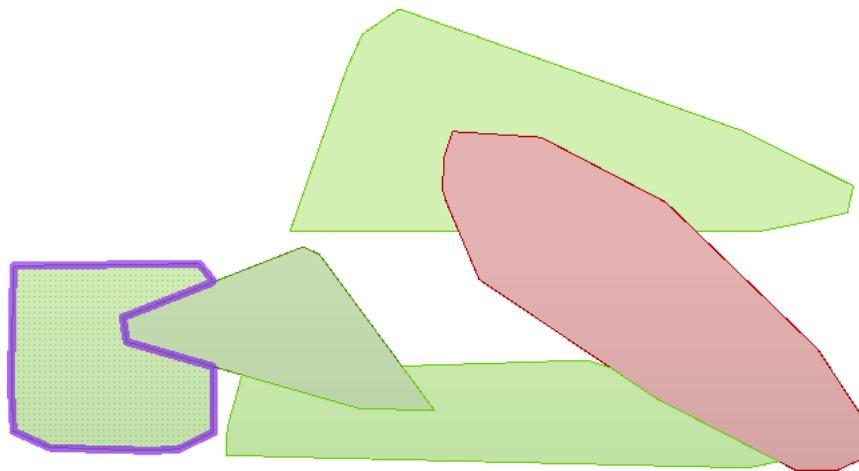
Inspect the parameters. Under Group Processing > Group By select the attribute `_histogram.value`:



You'll be prompted for a value; this is which item in the list do we want. We want the first element because it has the most values, so this field should be set to zero (which it will be by default):

Select Elements From: `_histogram{0}.value`

Run the workspace to the Dissolver and inspect the Dissolver:Area output port:



We now have five polygon features to represent garbage collection areas, each with approximately the same number of residents. Connect the Dissolver:Area port to the writer feature type and this workspace is nearly complete.

10) Remove the Sampler Transformer

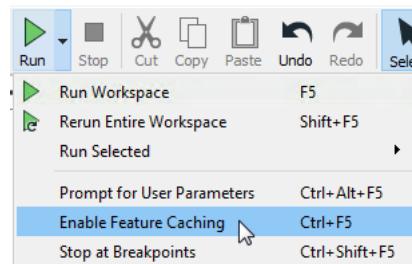
To complete the project let's run the workspace on the full dataset, but first let's get the workspace ready for production.

Delete the Sampler transformer, ensuring that PostalAddress and the PointOnAreaOverlay:Point input port are connected.

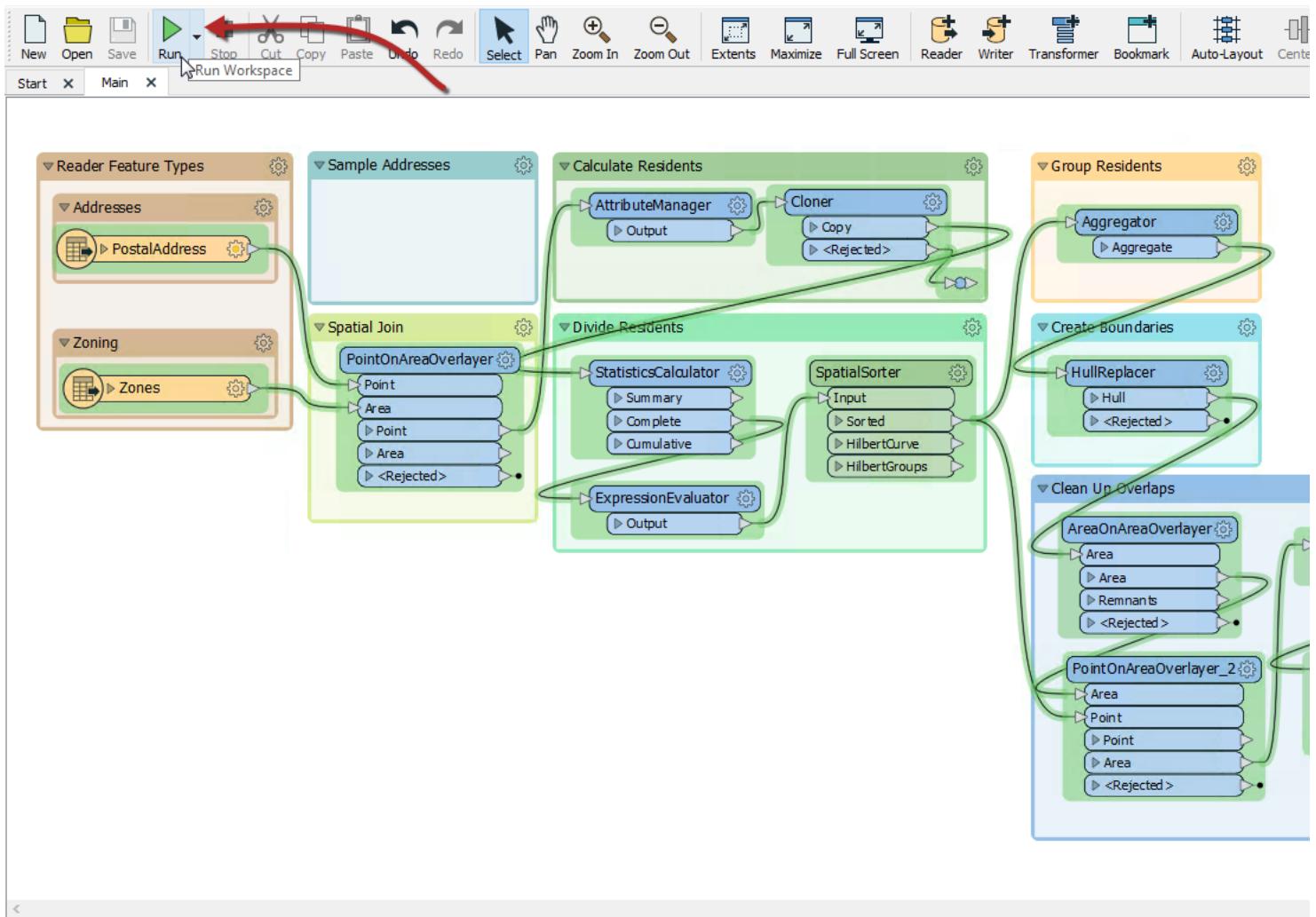


Instead of deleting the Sampler, you can just disable it. Right-click on the Sampler and choose Disable, then connect the PostalAddress and PointOnAreaOverlay just like the step above. This way if you need to come back and tweak something, the dataset can be sampled again easily.

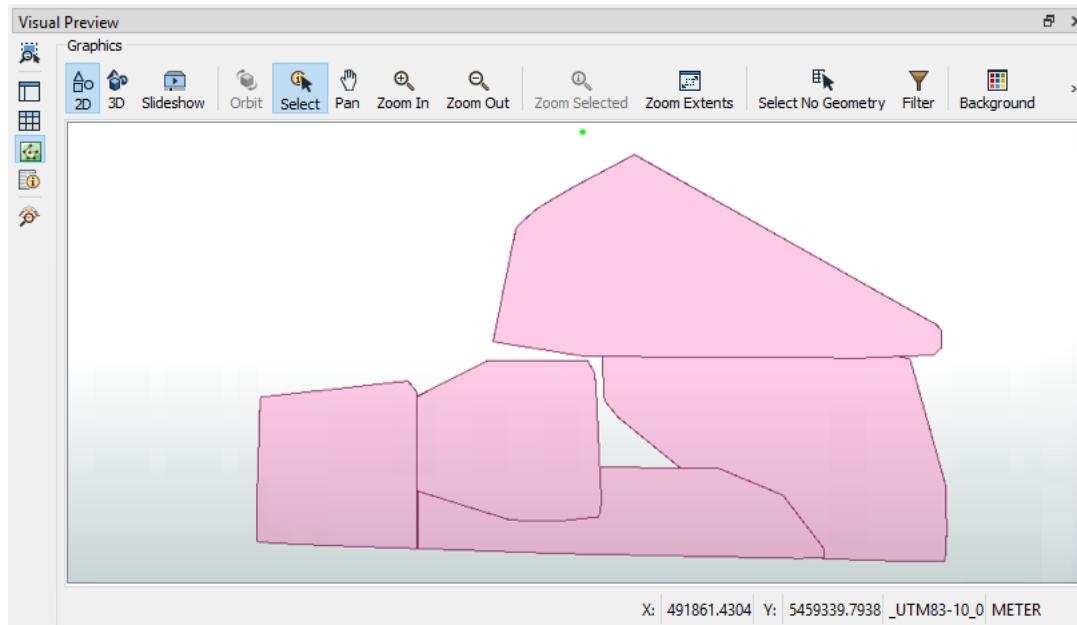
Before we run the translations, let's disable feature caching as the PostalAddress dataset is large. Click on the dropdown next to the Run button and click Enable Feature Cache to disable it:



Now with feature caching disabled, the Run button will run the entire workspace:



As expected, the result will look different, now that we're using the full dataset:



1 It would be best practice to use Run To This when testing a workspace output before writing to an online database.

- A.True
- B.False

2 Under most circumstances, which scenario will run faster?

- A.Running a workspace with feature caching enabled
- B.Running a workspace with feature caching disabled

3The Aggregator transformer's GroupSize attribute suggests there are a certain number of residents per garbage collection zone (represented by the GroupSize attribute). This attribute is still present in the final data before writing. Is it still accurate?

- A.Yes
- B.No

[Check the Quiz to Earn 150 Points](#)

Collaborate on Data Integration Workflows

Learning Objectives

After completing this unit, you'll be able to:

- Explain why collaboration is key to data integration projects.
- Explain what Workspace Compare and Merge does.
- Understand when to use Workspace Compare and Merge in your work.

Collaboration is Key



"Talent wins games, but teamwork and intelligence win championships."

- Michael Jordan

A similar adage could be said for building data integration workflows with FME: “You can build a *good* workspace in isolation, but you can build a *great* one collaborating with your teammates and stakeholders.”

Collaboration is vital in data integration projects. Lynda Gratton and Tamara J. Erickson from the Harvard Business Review [report](#) that the average business team size has increased from 20 to 100 members. With this increase in scale, effective and efficient collaboration becomes simultaneously more challenging and more critical.

Additionally, Philip Russom, Senior Manager at TDWI Research, [argues](#) that collaboration is essential because data integration involves bringing together a variety of technical specialists. Data integration projects are often

as much about integrating the workflows of diverse stakeholders as they are about integrating the data itself. He points out several trends that make collaboration even more critical:

- More teams are working on data integration projects.
- Data integration requires connecting many data sources and applications.
- Data integration professionals are increasingly working remotely.
- Data integration often must collaborate with other data management disciplines.
- Staff on the business side of organizations are getting more involved in data integration projects.
- Data integration teams often have to work with data governance committees and other oversight bodies.

These changes to the industry require that you build robust systems for collaborating and keeping track of changes to your workflows. Even as a solo system integrator, tracking the changes to your workspace over time is key to accurate data delivery.

Workspace Compare and Merge

Workspace Compare and Merge in FME Workbench helps you manage collaboration in a secure, reliable, and intuitive way.

This new tool in FME 2022.0 allows you to review the differences between two workspaces across all major non-cosmetic workflow components, including added/modified/deleted:

- User parameters
- Transformers
- Custom transformers
- Connections
- Readers, writers, and feature types
- Etc.

After comparing the changes, you can choose which ones you would like to merge.

Building workflows to automate integration tasks is not enough; you have to maintain them. FME is flexible and offers a low barrier to building complex solutions; this means that users from various departments within an organization could build or contribute to a given FME workspace. Additionally, you might work with contractors or solutions providers and need to exchange workspaces to iterate upon them. Workspace Compare and Merge lets you do this quickly and effectively.

Use Cases

When might you want to use Workspace Compare and Merge? Consider the following list of use cases. Do any apply to your work with FME?

- You follow best practices by using a version control system of some kind with your workspaces. This process could be as simple as renaming workspace files with different update dates or versions, e.g., “workspace-2022-05-19.fmw,” “workspace-2022-05-20.fmw,” etc. While workspace files are plain-text (XML, in fact), it’s not easy to track and compare changes that way since the file is structured so FME Workbench can read it. This tool solves the problem! You can quickly load up different versions, compare them, and choose which changes to keep.
- You often share workspaces with your colleagues and use external documentation, screenshots, or videos to track and explain changes. This manual method of comparison can be time-consuming and inaccurate. Workspace Compare and Merge offers a more streamlined way to collaborate.

- You maintain separate versions of workspaces locally, on a shared network drive, or in FME Server repositories. Sometimes it can be difficult to tell which version is most up-to-date. Now you can download the workspace from FME Server and compare it to your local version to ensure your FME Server workflows stay up-to-date.



Workspace Compare and Merge is a powerful way to collaborate using FME. It does not connect directly to the existing [version control system in FME Server](#), but you can use these tools in tandem. For example, you could pull down version 1 of a workspace from FME Server, make some changes, view the changes by comparing to version 1 to confirm they are correct, and then publish the workspace to FME Server as version 2.

Many users have other ideas to further aid collaboration in future FME releases, such as [integrating with external version control systems](#) or [continuous integration/deployment](#) systems. If you would like to see features like this, please submit, comment, and vote on ideas on our [Ideas page](#). The last unit in this module also contains a feedback survey on this new feature where you can share your thoughts.

1 What trend do Lynda Gratton and Tamara J. Erickson use to argue that collaboration is becoming both harder and more important?

- A.Increasing average budgets for data integration projects
- B.Increasing average team sizes
- C.Increasing demand on data integration teams
- D.Shorter average project times

2 Visual Workspace Comparison and Merge is useful for solo workspace authors.

- A.True
- B.False

3 Which of the following is not a valid use case for Visual Workspace Comparison and Merge?

- A.Comparing local workspaces and workspaces stored on FME Server.
- B.Collaborating on shared workspaces with colleagues.
- C.Viewing changes to your own personal workspaces over time.
- D.Comparing the performance metrics of different workspaces.

Check the Quiz to Earn 100 Points

Compare and Merge Workspaces

Learning Objectives

After completing this unit, you'll be able to:

- Open the Workspace Comparison window.
- View and understand the list of changes in the Workspace Comparison window.
- Merge selected changes in the Workspace Comparison window.

What is Workspace Comparison and Merge?

Workspace Comparison and Merge (Workspace Comparison for short) allows users to compare two workspaces simultaneously, seeing the changes between the two workspaces. These changes are represented visually, allowing the user to quickly distinguish what has been modified, added, or deleted. They can then merge these changes into the workspace they are working on.

One of FME's strengths is its ability to build complex solutions that apply in multiple departments of an organization. Colleagues often share workspaces and make iterations on those workspaces. Inheriting a workspace from a colleague or working on a workspace together has often resulted in a lot of back and forth, sharing screenshots. Workspace Comparison aims to save time and improve on this collaborative experience.

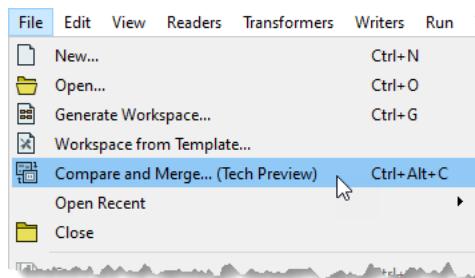


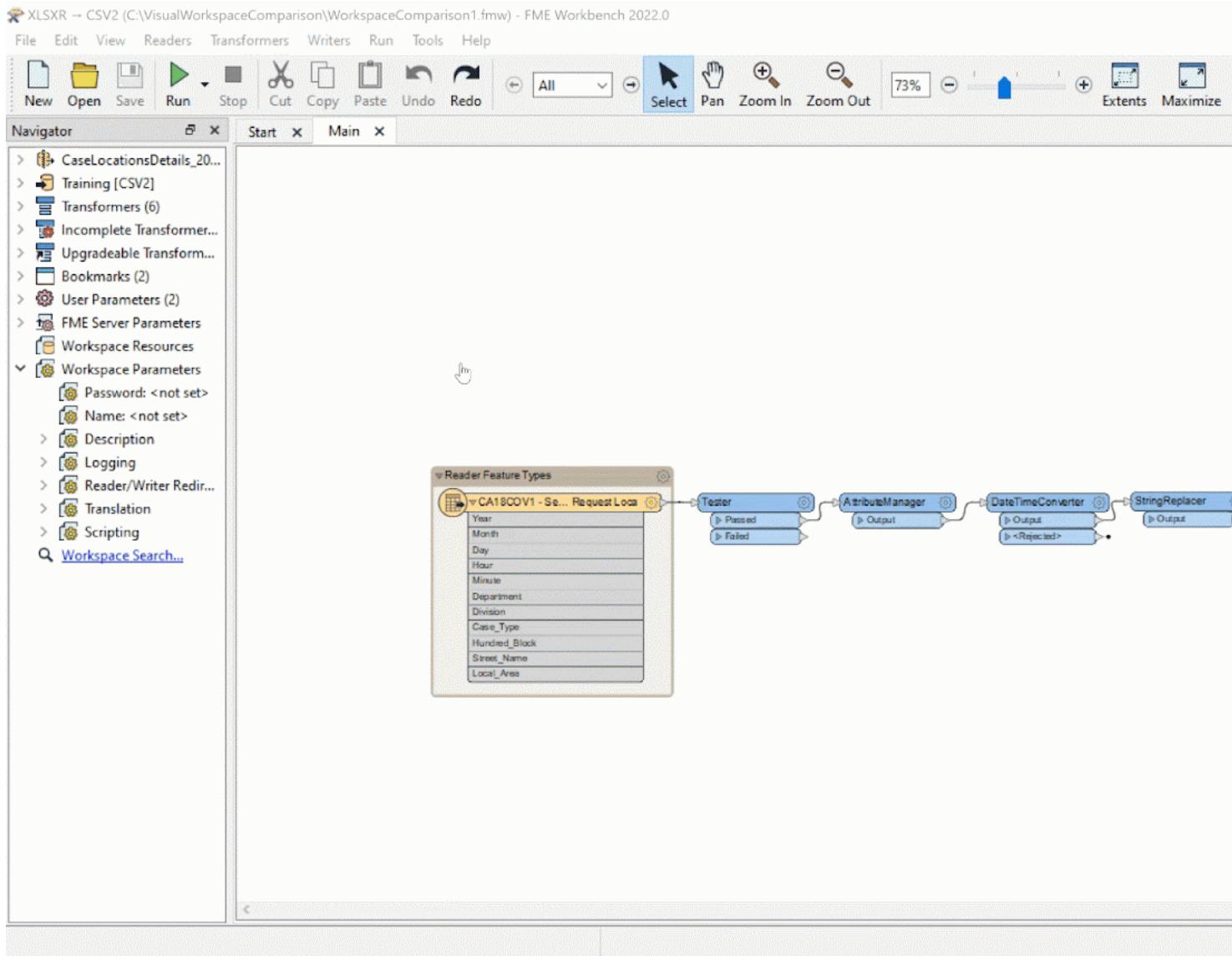
Workspace Comparison is one of the exciting new features for FME 2022.0. A user first posted this feature to the Community Ideas in August 2017. With 1,380 points to this idea, it is one of the most requested ideas from the members of the Community. And with a lot of careful consideration, user testing, and help from the FME users, it is now in FME Desktop 2022.0.

Workspace Comparison is currently in Tech Preview.

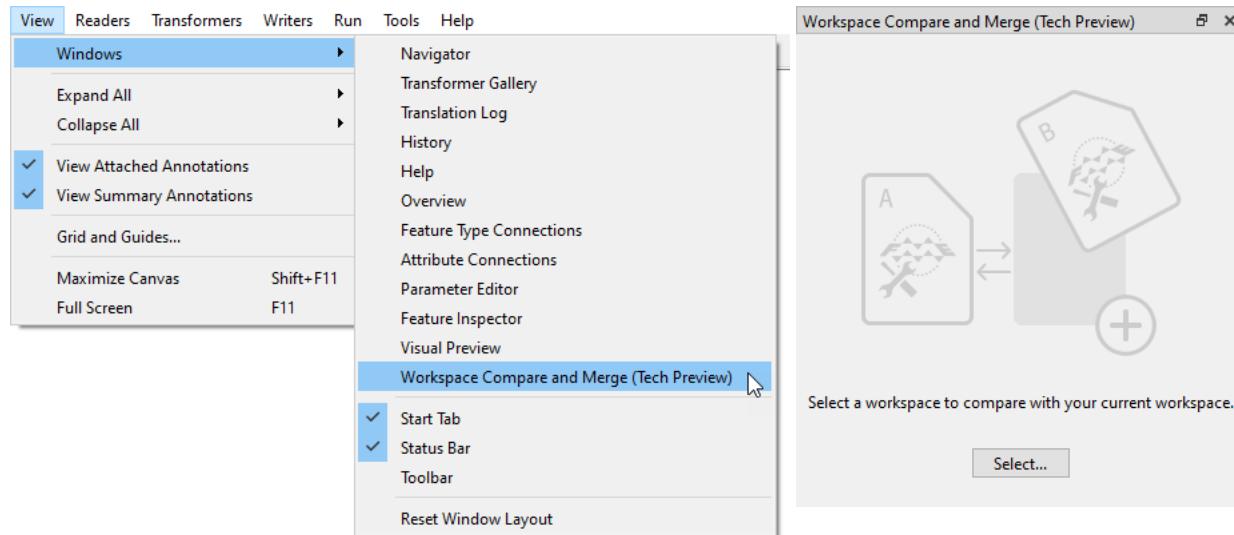
Where Can I Find Workspace Comparison?

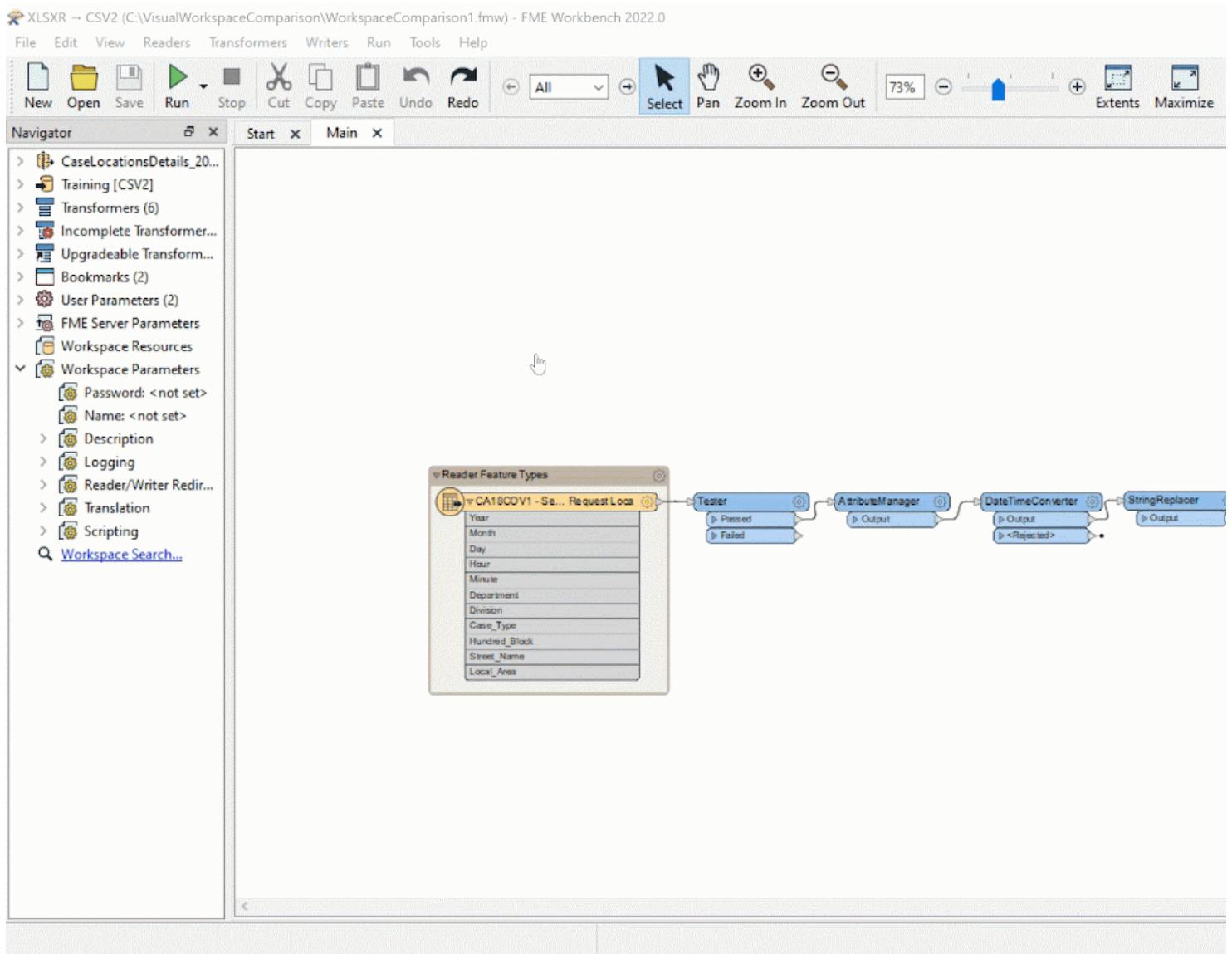
Workspace Comparison can be found in two ways. Within FME Desktop, in a blank workspace or already populated workspace, click the **File > Compare and Merge (Tech Preview)**. This will open the file explorer, where you can select the workspace you would like to compare.





You can also find Workspace Comparison by clicking **View > Windows** and selecting **Workspace Comparison (Tech Preview)**. This will open the *Workspace Comparison* window. Here you can select the workspace you would like to compare to your current workspace, just like the first method described above.





How Does Workspace Comparison Help?

For those of you that are new to FME, traditionally to compare two different workspaces in previous versions of FME Desktop, you would need to open two instances of FME Desktop Workbench and open the two workspaces you would like to compare side by side. This method had its pros and cons. But if you were only working on one screen it meant lots of going back and forth between the two instances of FME, and it was often time-consuming trying to find where changes occurred, e.g., if a parameter had been changed in a transformer. Or, if you are working on a workspace with a colleague, there would be a lot of back and forth - sharing of screenshots and even videos.

Likewise for some users of FME Desktop that have very large workspaces which accomplish a lot of tasks, often trying to find the changes that had been made between the two different versions of these workspaces could take quite a while. Workspace Comparison will help solve these challenges. This tool will help to reduce time spent looking for changes in workspaces, allow for better collaboration between teams, colleagues, and departments in organizations - in turn saving you time to solve even more data challenges.

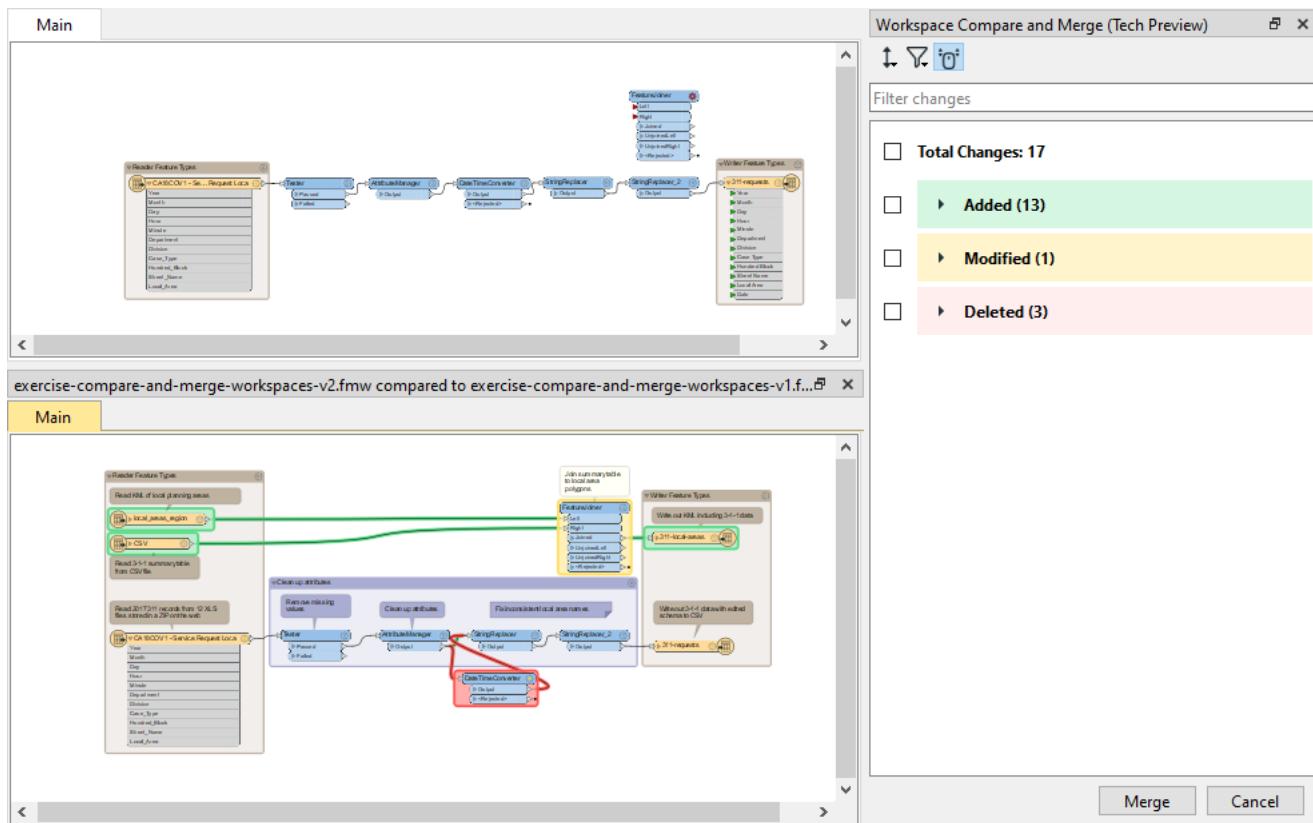
Which Workspace Should I Open First?

A common challenge using this tool is deciding which workspace to open first, and which to open via **File > Compare and Merge (Tech Preview)**. Here are some common scenarios and the correct order to open the workspaces:

1. If you have the main version open and want to compare someone else's changes, you keep your main (but potentially outdated) version open and compare to the new workspace. Then you can choose changes to merge into the main workspace
2. If you want to selectively roll back changes to an earlier version of the workspace, open the workspace that you want to roll back. Then, compare to the older workspace. From here you can merge changes to roll them back. In this case, the colors and terminology might throw you off a bit. Deleted is more like "didn't exist yet" or "missing."
3. If you are in a situation where there has been unintentional branching of two workspaces, e.g., if a colleague started editing an older version of the workspace. It is basically the same process as case #1 above. You just have to think a bit more carefully about merging, as you might have more incompatible changes.

How Does Workspace Comparison Work?

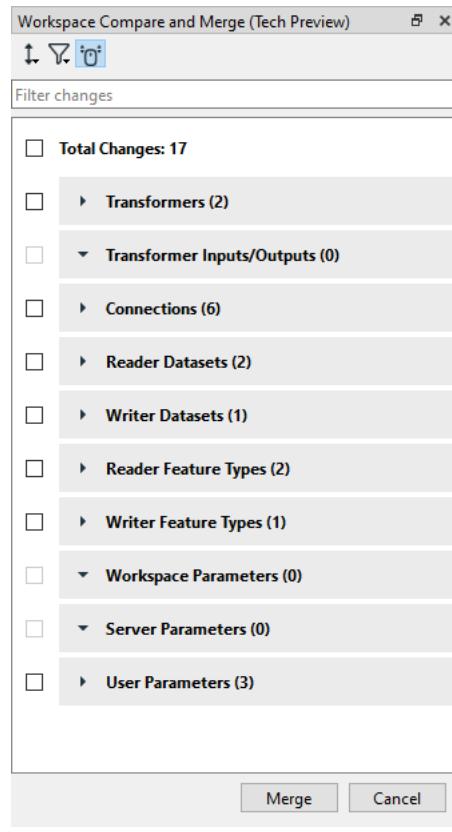
Workspace Comparison works by looking at each workspace side by side and providing you with a list of the changes that appear between the two workspaces.



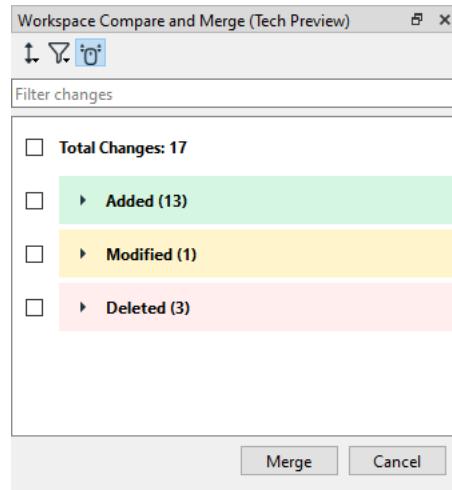
Changes are color-coded to make it easier to see the changes.

- Green means something has been added to the workspace.
- Yellow means something has been modified.
- Red means something has been deleted.

What changes does Workspace Comparison look at? FME will look for non-cosmetic changes, meaning it will look for changes made to Transformers (Including custom transformers), Transformer Input/Outputs, Connections (between transformers), Reader datasets, Writer datasets, Reader Feature Types, Writer Feature Types, Workspace Parameters, Server Parameters, and User Parameters.

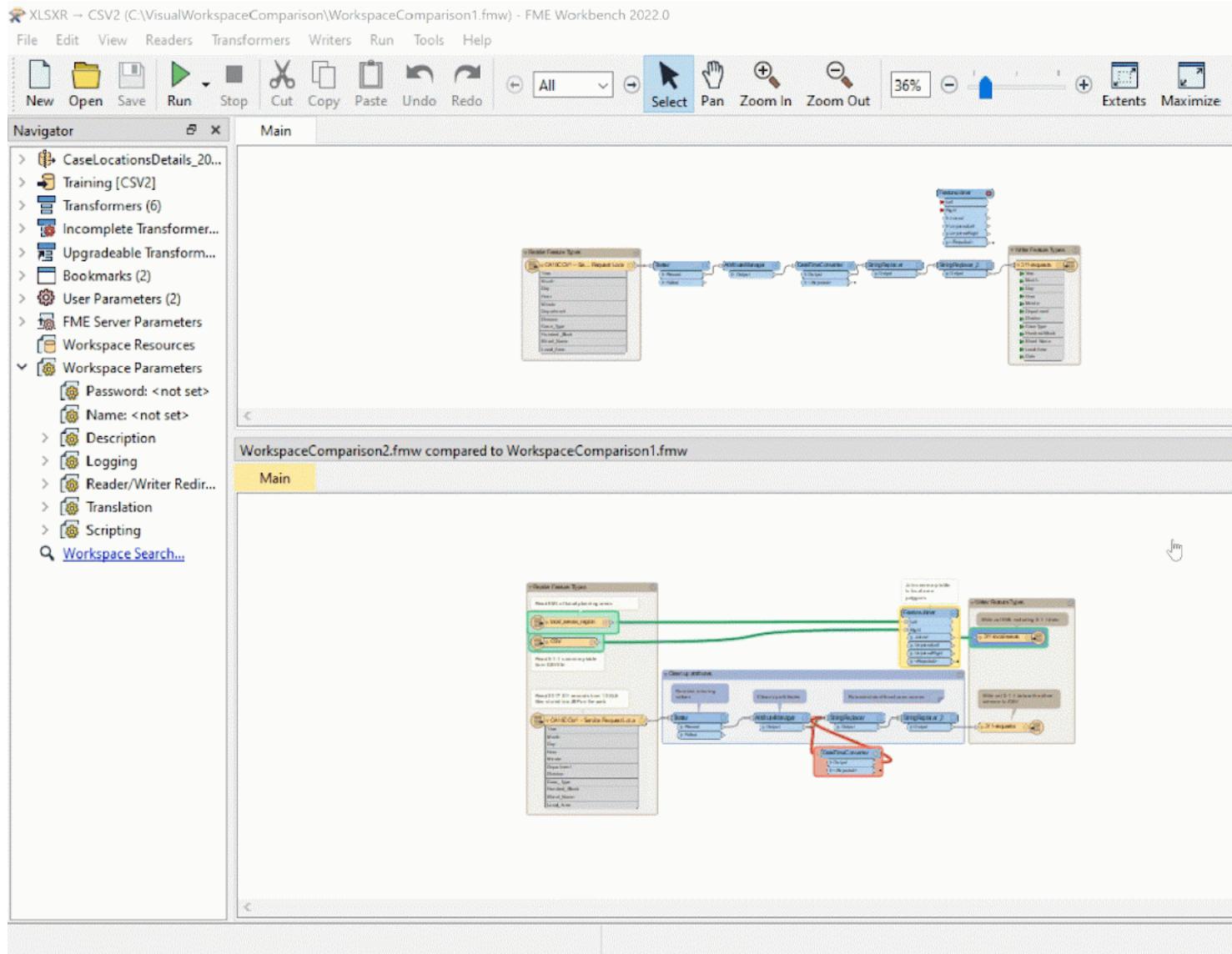


Or, you can specify to view the changes by Change Type, meaning it will show what has been Added, Modified, and Deleted from the workspace.



You can search for changes using the *Filter changes* bar.

You can review the changes by clicking on each change in the *Workspace Comparison* window. The canvas will zoom to that change, and FME will highlight changed objects with the corresponding color to match the change. Both canvases are synchronized, meaning that if you click on a modified transformer, for instance, the canvas navigates to that transformer. This also applies to zooming and panning on the canvas. This option can be disabled by clicking the **Toggle Synchronized Navigation** option on the *Workspace Comparison* window (by default this is turned on). Likewise, by clicking the highlighted change on the canvas, the corresponding change will be highlighted in the *Workspace Comparison* window.



Clicking the cogwheel next to changes in the *Modified* section of the *Workspace Comparison* window will show the parameters side by side. In this case, we can see the parameters of the FeatureJoiner. Within a transformer, the same methodology applies: added parameters are green, modified parameters are yellow. FME will also highlight if there has been a change in the version as well as new parameters in the transformer.

XLSXR → CSV2 (C:\VisualWorkspaceComparison\WorkspaceComparison1.fmw) - FME Workbench 2022.0

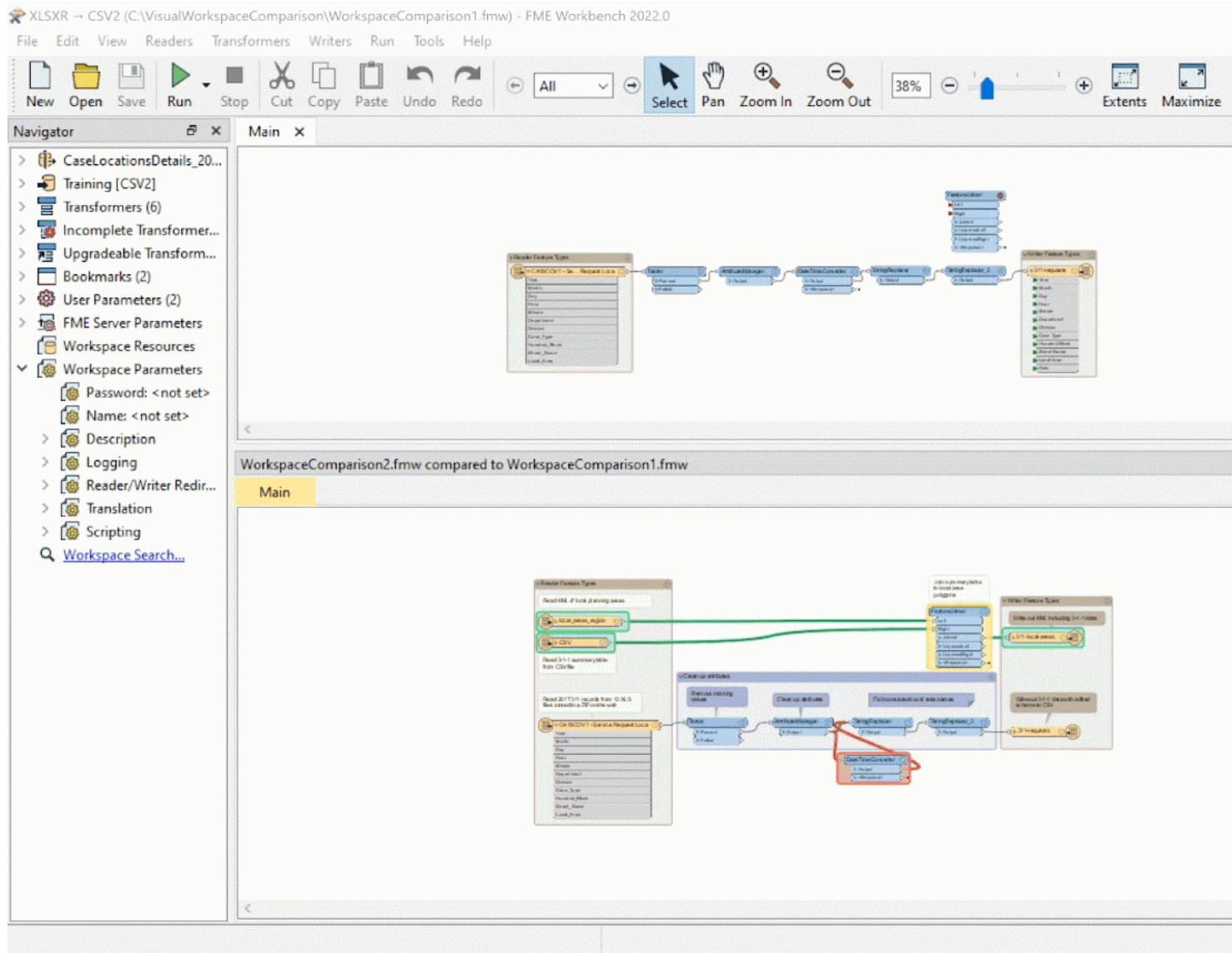
The screenshot shows the FME Workbench interface with two main windows. The top window displays a workspace named 'CaseLocationsDetails_20...' containing a transformer workflow. The bottom window shows a comparison between 'WorkspaceComparison2.fmw' and 'WorkspaceComparison1.fmw'. A specific transformer, 'AttributeRenamer', is highlighted in both workspaces, and a detailed comparison dialog is open for it.

Transformer Comparison Dialog: 'AttributeRenamer' Properties Comparison

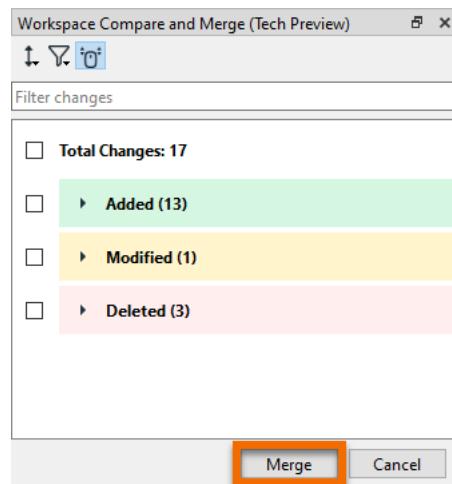
UpdatePostAddressV1.fmw		UpdatePostAddressV2.fmw														
Version 4	Version 5	Version 4	Version 5													
AttributeRenamer (AttributeRenamer)		AttributeRenamer (AttributeRenamer)														
Attributes To Rename		Attributes To Rename														
<table border="1"> <thead> <tr> <th>Input Attribute</th> <th>Output Attribute</th> <th>Default Value</th> </tr> </thead> <tbody> <tr> <td>_distance</td> <td>ParkDistance</td> <td></td> </tr> <tr> <td>PSTLADDRESS</td> <td>PostalAddress</td> <td></td> </tr> </tbody> </table>		Input Attribute	Output Attribute	Default Value	_distance	ParkDistance		PSTLADDRESS	PostalAddress		<table border="1"> <thead> <tr> <th>Input Attribute</th> <th>Output Attribute</th> </tr> </thead> <tbody> <tr> <td>_distance</td> <td>PoolDistance</td> </tr> </tbody> </table>		Input Attribute	Output Attribute	_distance	PoolDistance
Input Attribute	Output Attribute	Default Value														
_distance	ParkDistance															
PSTLADDRESS	PostalAddress															
Input Attribute	Output Attribute															
_distance	PoolDistance															
<input type="button" value="Filter"/>		<input type="button" value="Filter"/>														
		If Input Attribute Is Missing: <input checked="" type="radio"/> Preserve Existing Output Attribute														
		<input type="button" value="Close"/>														

This dialog will show transformer version number changes for when you are upgrading workspaces between FME versions.

Now that you have reviewed the changes, you can modify your existing workspace with some of the changes in the *Workspace Comparison* window. You can choose the changes that you would like to implement. Holding CTRL (or Cmd ⌘ on Mac) and clicking the checkboxes will let you select the changes you want to merge.



When you've selected the ones you want, click the **Merge** button, which can be found at the bottom-right corner of the *Workspace Comparison* window.



—1 What would you do if you already had a workspace open and wanted to compare it to another?

- A.Drag and drop the other workspace onto the Canvas.
- B.File > Compare Workspace...
- C.File > Open...
- D.Make changes to the workspace and view the differences in your version control system.

—2 Which of the following changes would not appear in the *Workspace Comparison* window?

None of the following changes would not appear in the workspace comparison interface.

- A.A MapInfo TAB writer feature type was deleted.
- B.A new bookmark was added.
- C.A new InlineQuerier was added.
- D.The workspace Overview parameter was changed.

When merging workspaces, you can choose to merge a selection of changes instead of all changes.

- A.True
- B.False

Check the Quiz to Earn 100 Points

Exercise: Compare and Merge Workspaces

Learning Objectives

After completing this unit, you'll be able to:

- Open a workspace to compare using Workspace Comparison.
- View the list of changes using Workspace Comparison and find them on the canvas.
- View parameter changes using Workspace Comparison.
- Merge selected changes using Workspace Comparison.

Resources

- [Starting workspace](#)
- [Updated workspace](#)
- [Complete workspace](#)

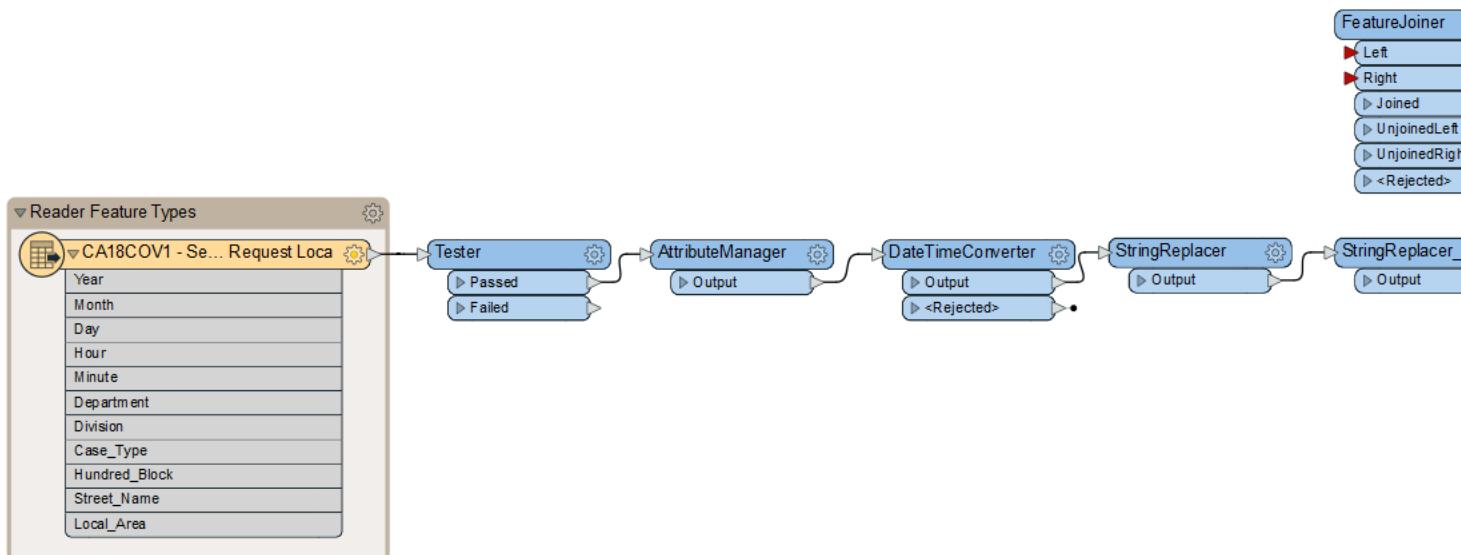
Introduction

You created a workspace to do some attribute cleanup on some 3-1-1 call data. [3-1-1](#) is a special phone number used by municipalities in Canada and the United States to manage non-emergency municipal service calls.

A few weeks later, your colleague modified the workspace to summarize the call records and join them to spatial data. They've added a few feature types and added/modified transformers. Now you have two versions and would like to rectify them into an accepted shared version of the workspace. You can take advantage of FME 2022.0's new Workspace Comparison and Merge (Workspace Comparison for short).

1) Open Workbench

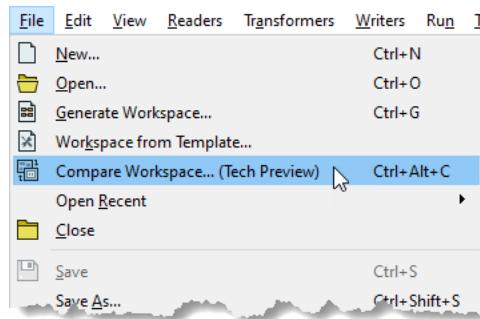
Open the [starting workspace](#) in FME Workbench (2022.0 or later).



2) Open Workspace Comparison

Download the [updated workspace](#).

Click File > Compare and Merge...



Select the workspace with changes, **exercise-compare-and-merge-workspaces-v2.fmw**, and click **Open**.

The workspace with changes opens in the **Workspace Comparison** window.

Main

exercise-compare-and-merge-workspaces-v2.fmw compared to exercise-compare-and-merge-workspaces-v1.fmw

Main

Visual Preview exercise-compare-and-merge-workspaces-v2.fmw compared to exercise-compare-and-merge-workspaces-v1.fmw

Workspace Comparison (Tech Preview)

Total Changes: 18

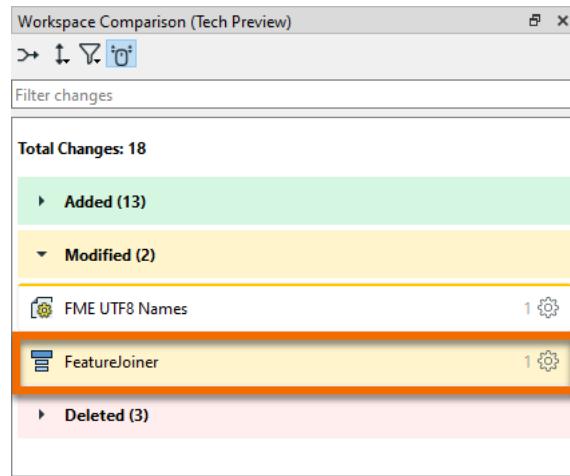
Added (13)

- Source Google KML File or URL
- Destination Google KML File
- Source CSV (Comma Separated Value) F
- local_areas_region [LocalAreas [OGCKN]
- FeatureJoiner [FeatureJoiner Version 10C]
- CSV [311-summary [CSV2]] -> FeatureJoiner
- AttributeManager [AttributeManager Ver
- 311-local-areas
- local_areas_region
- CSV
- 311-local-areas [OGCKML]
- LocalAreas [OGCKML]
- 311-summary [CSV2]

3) Find a Specific Change

Look in the *Workspace Comparison* window on the right. You should see a list of changes.

One of the changes reports that the **FeatureJoiner** was **Modified**. Find it in the list.



After finding it, click it once to highlight the transformer with changes on the canvas.

The screenshot shows the FME Workbench interface with the following components:

- Main Window:** Displays a workflow diagram with several transformers and a writer feature type.
- FeatureJoiner Transformer:** A central component in the workflow, highlighted with a yellow border.
- Writer Feature Types:** A dialog box associated with the FeatureJoiner transformer.
- Comparison Window:** The 'Workspace Comparison (Tech Preview)' window is open on the right side, showing the following changes:
 - Total Changes: 18**
 - Added (13)**
 - Modified (2)**
 - Deleted (3)**
 - Specific items listed under 'Modified' include 'FeatureJoiner' and 'DateTimeConverter [DateTimeConverte]'.

4) View Changed Parameters

Double-click the **FeatureJoiner** change in the *Workspace Comparison* window to open the *FeatureJoiner Properties Comparison* dialog.

Here you can confirm that in the original workspace, you did not configure the FeatureJoiner. However, the new version joins tabular CSV data of 3-1-1 call records to local planning area polygons read from a KML file. The shared key is the name of the local planning area stored in attributes `Name` and `Local Area`, respectively.

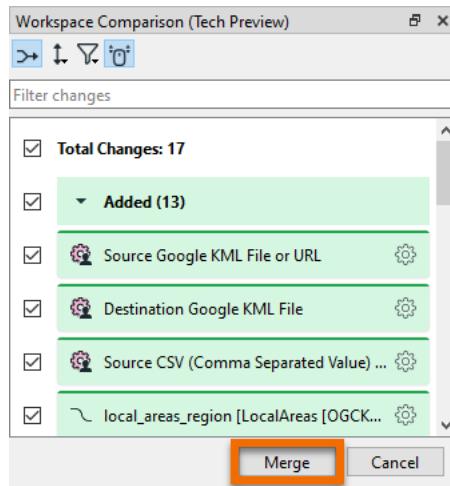
Click **Close**.

5) Select Changes to Merge

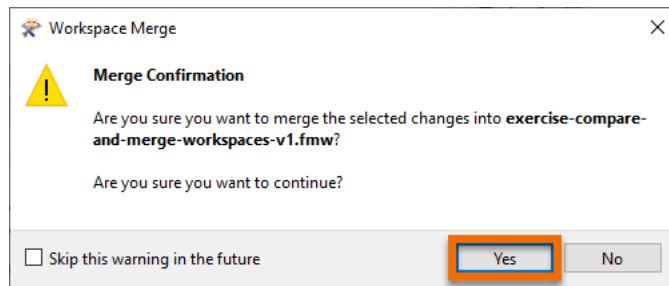
Browse the rest of the changes. It looks like all of them are necessary for the new workspace that summarizes calls by local area.

Select all changes by clicking the checkbox next to **Total Changes: 17** at the top of the window.

Click **Merge** at the bottom of the window.



FME will prompt you to ensure you want to merge the changes. Click **OK**.



You will see the new, modified version of the workspace. You can save it to store the merged changes.



In this scenario, you could also have decided to compare changes, confirm they were all correct, delete the v1 workspace, and continue using v2. Which method you choose depends on if you want to accept *all* changes or just *some* of them (essentially creating a “fork” or parallel development path) and if you maintain separate copies of similar workspaces in your organization.

Survey

We'd appreciate it if you could fill out this quick optional survey on this new feature. Your answers will not affect your score on the module; take the Quiz below to complete the module.

- 1When viewing the list of changes, which of the following methods won't let you check the change's component type (transformer, feature type, etc.)?
- A.Look at the icon, which will match the component icon in Workbench
 - B.Click the "Group cards by category" button and change the group mode from "Change Type" to "Component Type."
 - C.Right-click the change and select "View Change Location."
 - D.Use the "Filter by component type" button in the toolbar.
- 2How many feature types were added in the modified (v2) workspace?
- A.1
 - B.2
 - C.3
 - D.4
- 3When selecting changes to merge, you can click checkboxes to select all the Added, Modified, or Deleted changes at once.
- A.True
 - B.False

[Check the Quiz to Earn 100 Points](#)