Table of Contents

Data Integration with FME Desktop

This manual is an introductory-level training module for FME Desktop. It provides a framework for a basic understanding of FME and for using FME in data integration. While the scenario used is grounded in Geographic Information Systems for local government, urban planning, and administration, the skills are applicable to any use of FME. It has approximately four hours of content.

The module will introduce basic concepts and terminology, help students become efficient users of FME, and direct them to resources to help apply the product to their own needs.

Module Structure

The module is made up of three main sections. These sections are:

- Lecture
- Lab Demonstration
- Lab Exercises

The **Lecture** section is intended to be adapted by instructors to be taught in a classroom lecture. It covers conceptual issues of data integration and provides a high-level introduction to FME. Slides are available.

The **Lab** section provides a walkthrough of how to use FME. It is designed to be presented by an instructor in a lab, with students either following along or independently completing the provided exercises.

Finally, the **Scenario-based Exercise** section provides a activities for students to complete during lab time or as an assignment.

Lab questions are provided; instructors may fill out this form to request access to answers for grading.

Module Audience

This module is aimed at a university or college student audience. FME sees a wide range of applications in any discipline that works with data, but is most commonly used in:

- Architecture
- Construction
- Engineering
- Forestry
- Geography and Geographic Information Systems (GIS)
- Oil & Gas
- Public Administration
- Telecommunications
- Urban Planning
- Utilities

Students should be comfortable with data manipulation software in a scripted or GUI environment. Prior knowledge of FME is not required. If the instructor wishes to adapt the module to a higher level, the lecture and lab demonstration content can be shortened.

Module Length

The module's total length is approximately four hours. The lecture is approximately half an hour. The lab contains about two hours of walkthrough and exercises. Finally, the scenario-based exercise will take about an hour to complete. The instructor may choose to cover as many of these sections as they feel are required, or possible in the time permitted. They may also cover the module content in a different order and will skip or add new content to better customize the module to their needs. Therefore, the length and content of the module may vary.

Adapting this Module

All of this content is provided for instructors to use, adapt, and remix for their own modules. Please note that this use does come with some legal caveats, explained in About this Document.

If you wish to change the content of the manual, you can fork the GitHub repository. The manual is created using the GitBook toolchain and is written in easily editable Markdown. Detailed instructions on adapting this content are also available.

Some suggestions for adaptation:

- Localize the exercise data to your country or city. A list of open data portals is available here. You could also use OpenStreetMap data, which has good global coverage.
  - If you do change the data, feel free to modify your own Module Resources page.
- If you prefer to grade close-ended questions, you could edit the final question in the scenario exercise. You could restrict the datasets or provide a data integration problem to solve rather then leaving it open for students to choose their data.

Current Status

The current status of this manual is: **READY**. This manual applies to **FME2018.0**. The status of each chapter is:

- Lecture: Complete
- Lab Demonstration: Complete
- Lab Exercises: Complete
- Slides: Complete
- Answers: Complete

***NB**: Even for completed content, Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice. See the full licensing agreement in About this Document for further details.*

Thanks

Thanks to FME users Dave Laurier, Clair Ellul, and Patrick Lougheed for their feedback on an earlier version of this module.

About This Document

This manual is an introductory-level training module for FME Desktop.

Look out for residents of the City of Interopolis, who will appear from time-to-time to give you advice and dispense FME-related wisdom. In fact, here comes someone now:

Mr. E.Dict (Attorney of FME Law) says...

On behalf of the City of Interopolis, welcome to this training course. Here is the standard legal information about this training document and the datasets used during the course.

Be sure to read it, particularly if you're thinking about re-using or modifying this content.

Licensing and Warranty

Permission is hereby granted to use, modify and distribute the FME Tutorials and related data and documentation (collectively, the “Tutorials”), subject to the following restrictions:

1. The origin of the Tutorials and any associated FME® software must not be misrepresented.
2. Redistributions in original or modified form must include Safe Software’s copyright notice and any applicable Data Source(s) notices.

3. You may not suggest that any modified version of the Tutorials is endorsed or approved by Safe Software Inc.
4. Redistributions in original or modified form must include a disclaimer similar to that below which: (a) states that the Tutorials are provided “as-is”; (b) disclaims any warranties; and (c) waives any liability claims.

Safe Software Inc. makes no warranty either expressed or implied, including, but not limited to, any implied warranties of merchantability, non-infringement, or fitness for a particular purpose regarding these Tutorials, and makes such Tutorials available solely on an “as-is” basis. In no event shall Safe Software Inc. be liable to anyone for direct, indirect, special, collateral, incidental, or consequential damages in connection with or arising out of the use, modification or distribution of these Tutorials.

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

Data Sources

City of Vancouver

Unless otherwise stated, the data used here originates from open data made available by the City of Vancouver, British Columbia. It contains information licensed under the Open Government License - Vancouver.

Others

Forward Sortation Areas: Statistics Canada, 2011 Census Digital Boundary Files, 2013. Reproduced and distributed on an "as is" basis with the permission of Statistics Canada. © This data includes information copied with permission from Canada Post Corporation.

Digital Elevation Model: GeoBase®

Fire Hall Data: Some attribute data adapted from content © 2013 by Wikipedia, used under a Creative Commons Attribution-ShareAlike license

Stanley Park GPS Trail: Used with kind permission of VancouverTrails.com.

OpenStreetMap Datasets: © OpenStreetMap contributors.

Contains information licensed under the Open Government Licence – British Columbia

Copyright

Revisions

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

Safe Software Inc.
Phone: 604-501-9985
Email: train@safe.com
Web: safe.com
GitHub: github.com/safesoftware/FMETraining

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

Trademarks

FME® is a registered trademark of Safe Software Inc. All brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Document Information

Document Name: FME Desktop GIS and CAD Integration

Module Overview

This module provides an introduction to the concept of data integration and interoperability and contains hands-on examples of using Safe Software's FME to integrate data.

The module introduces basic concepts and terminology, help students become efficient users of FME, and direct them to resources to learn more about the product.

Module Structure

The full module is made up of three main sections. These sections are:

- Lecture
- Lab Instruction
- Lab Exercises

The **Lecture** section is intended to be adapted by instructors to be taught in a classroom lecture. It covers conceptual issues of data integration and provides a high-level introduction to FME. Slides are provided here.

The **Lab Demonstration** section provides a walkthrough of how to use FME. It can be delivered by an instructor and/or students can complete it on their own computers.

Finally, the **Lab Exercises** section provides hands-on activities for students to complete during lab time or as assignments. It contains questions; instructors may request access to answers for use in grading here.

Module Audience

This module is aimed at a second- to fourth-year university or college student audience. Computer and data analysis experience will help, but is not required. Prior knowledge of FME is not required. If the instructor wishes to adapt the module to a higher level, the lecture and lab demonstration content could be shortened.

Module Length

The module's total length is approximately four hours. The lecture is approximately half an hour. The lab contains about two hours of walkthrough and exercises. Finally, the scenario-based exercise will take about an hour to complete. The instructor may choose to cover as many of these sections as they feel are required, or possible in the time permitted. They may also cover the module content in a different order and will skip or add new content to better customize the module to their needs. Therefore, the length and content of the module may vary.

All of this content is provided for instructors to use, adapt, and remix for their own modules. Please note that this use does come with some legal caveats, explained in About this Document.

Module Resources

Downloading and Installing FME Desktop

You will need FME 2018 installed, plus a digital copy of this manual.

You can find the latest version of FME Desktop and FME Server for Windows, Mac, and Linux - together with the latest Beta versions - on the Safe Software web site.

A free 30-day trial is available. Safe Software also provides free licenses for higher education institutions, instructors, and students. Please visit the grant license page for more details.

Please refer to the Documentation if you need help installing FME Desktop.

Module Data

A number of sample datasets and workspaces will be used in this module.

The data used in this training module is based on open data from the City of Vancouver, Canada.

Most exercises ask you to assume the role of a city planner at the fictional city of Interopolis and to solve a particular problem using this data.

You can download the data for the module in a zip file. We recommend extracting it to C:\FMEData2018.

Location	Resource
C:\FMEData2018\Data	Datasets used by the City of Interopolis
C:\FMEData2018\Resources	Other resources used in the training
C:\FMEData2018\Workspaces	Workspaces used in the lab exercises
C:\FMEData2018\Output	The location in which to write exercise output
<documents>\My FME Workspaces	The default location to save FME workspaces
Slides	Slide deck for this module
Answer request form for instructors	Instructors can use this form to request answers.

Please alert your instructor if any item is missing from your setup.

Lecture

This section of the module discusses the concept of data integration, illustrates why it matters, and previews how FME can be used to integrate data.

It is designed to be delivered as a classroom lecture before students enter the lab. The concepts discussed here will be explored in more detail within the context of FME Desktop in the Lab Demonstration section.

Learning Objectives

Students will learn:

1. **Data integration** is a key technical and business challenge for contemporary organizations.
2. **FME** is a tool to translate and transform data.
3. **FME Desktop** is for data translations and transformations at the desktop level.
4. **Structural transformation** is perhaps better called ‘reorganization'. It includes the ability to merge data (as in the image above), divide data, re-order data, and define custom data structures.
5. **Content transformation** is perhaps better called ‘revision'. It refers to the ability to alter the substance of a dataset, for example changing a feature's attribute values or geometry.

Data Integration

> “Getting information off the Internet is like taking a drink from a firehose.”
>
> -- Mitchell Kapor, co-founder of Lotus and co-founder of the Electronic Frontier Foundation.

> “You can have data without information, but you cannot have information without data.”
>
> -- Daniel Keys Moran, American computer programmer and science fiction writer.

> "Information is the oil of the 21st century, and analytics is the combustion engine."
>
> -- Peter Sondergaard, Executive Vice President, Gartner, Inc.

As the quotations above illustrate, the creation, manipulation, and analysis of data represent a significant challenge for contemporary organizations. Never before has so much machine-readable data existed, but organizations still struggle to find ways to use this mass of information to aid in decision-making.

What is Data Integration?

Data integration means combining information from various sources into something useful. It’s about efficiently managing data and making it available to those who need it. Both a technical and a business process, IBM defines it as “discovery, cleansing, monitoring, transforming and delivery of data from a variety of sources.” Data integration allows the combination and analysis of data across isolated "silos" where it would normally be difficult to collaborate. It allows organizations with multiple departments, facilities, software, and workflows to bring all of their data together.

Why Do Organizations Need to Integrate Data?

Here are 9 reasons why integrating data is essential to technical and business practices.

1. Every data type has its strengths

Every data format was designed for a reason. Each one represents information in a way no other format can, with unique attributes, metadata, structure, and schema. Integrating data from different formats adds various levels of specialty to the dataset.

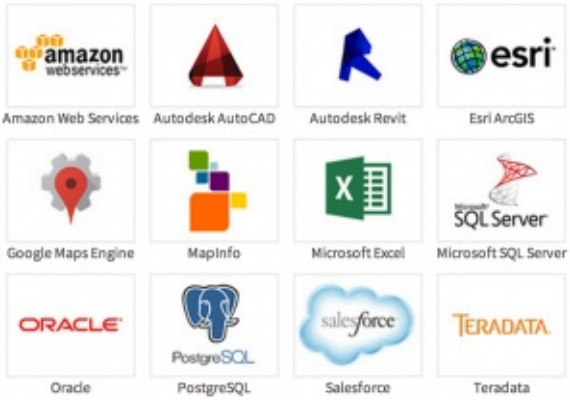
For instance, CAD and GIS data can be integrated to enhance a CAD drawing with specialized GIS information and attributes. The Culpeper Online GIS Mapping System is one example of a CAD-GIS data integration project.

This dataset is the result of integrated raster and point cloud information, and offers the best of both worlds.

2. Take advantage of specialized applications

Similarly, every application was designed for a reason. Every piece of software that works with data represents, analyzes, and transforms information in a specialized way. By integrating data into a format accepted by that application, you're giving yourself the power to open and use your data in that software.

For example, Cambian Business Services maintains data models from 60 different sources and at least 10 applications. They needed to integrate all of that data into a PostgreSQL database, then redeploy it to the original system. Data integration enables them to freely convert between formats and open their data in its original legacy application. (Read more about the Cambian Business Services data integration project.)



These are just a handful of popular applications that handle data in highly specialized ways.

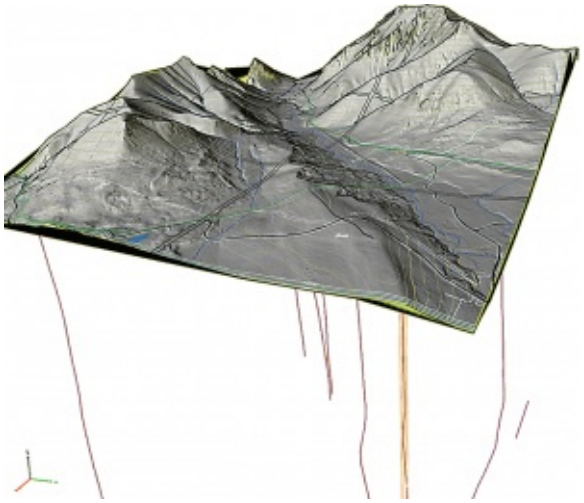
3. Reduce data complexity

April Reeve describes it well::

> The number of potential interfaces between applications in an organization is an exponential function of the number of applications. Thus, an organization with one thousand applications could have as many as half a million interfaces…

Data integration is about managing complexity, streamlining connections, and making it easy to deliver data to any system. This might involve creating a data hub that's easy to publish to and subscribe to.

For example, Shell Canada unifies complex 2D, 3D, raster, and vector information into an easily readable 3D PDF dataset. Their data integration plan gets information to users that might not have access to specialized GIS software. (Read more about Shell Canada's 3D PDF data integration.)



A data integration solution simplifies the interaction between diverse systems, like in this 3D PDF.

4. Increase the value of data through unified systems

Bringing disparate datasets together increases the value of the information. Examples include:

- merging with and leveraging external data (e.g. from vendors);
- combining data of varying natures (structured, unstructured, spatial, spreadsheet, web, raster, big data, etc.);
- applying spatial information to non-spatial data;
- combining databases from different repositories;

- creating a uniform schema for a group of datasets without metadata standards;
- bringing collections of data from different sources into the common GML structure for compliance with INSPIRE (see examples).

Talisman Energy, for example, integrates disparate datasets into a central GIS repository. The datasets are often not joined to GIS geometry, have a rigorous updating schedule, and may come from internal or external sources. For them, data integration is essential for efficient visualization and unified data access. (Read about Talisman Energy's integrated GIS database.)

This comprehensive representation of a project is the result of integrating various spatial, non-spatial, and web-based sources.

5. Make data more available

Centralizing your data makes it easy for anyone at your organization (or outside of it, depending on your goals) to retrieve, inspect, and analyze it.

Easily accessible data means easily transformed data. People will be more likely to integrate the data into their projects, share the results, and keep the data up to date. This cycle of available data is key for innovation and knowledge-sharing.

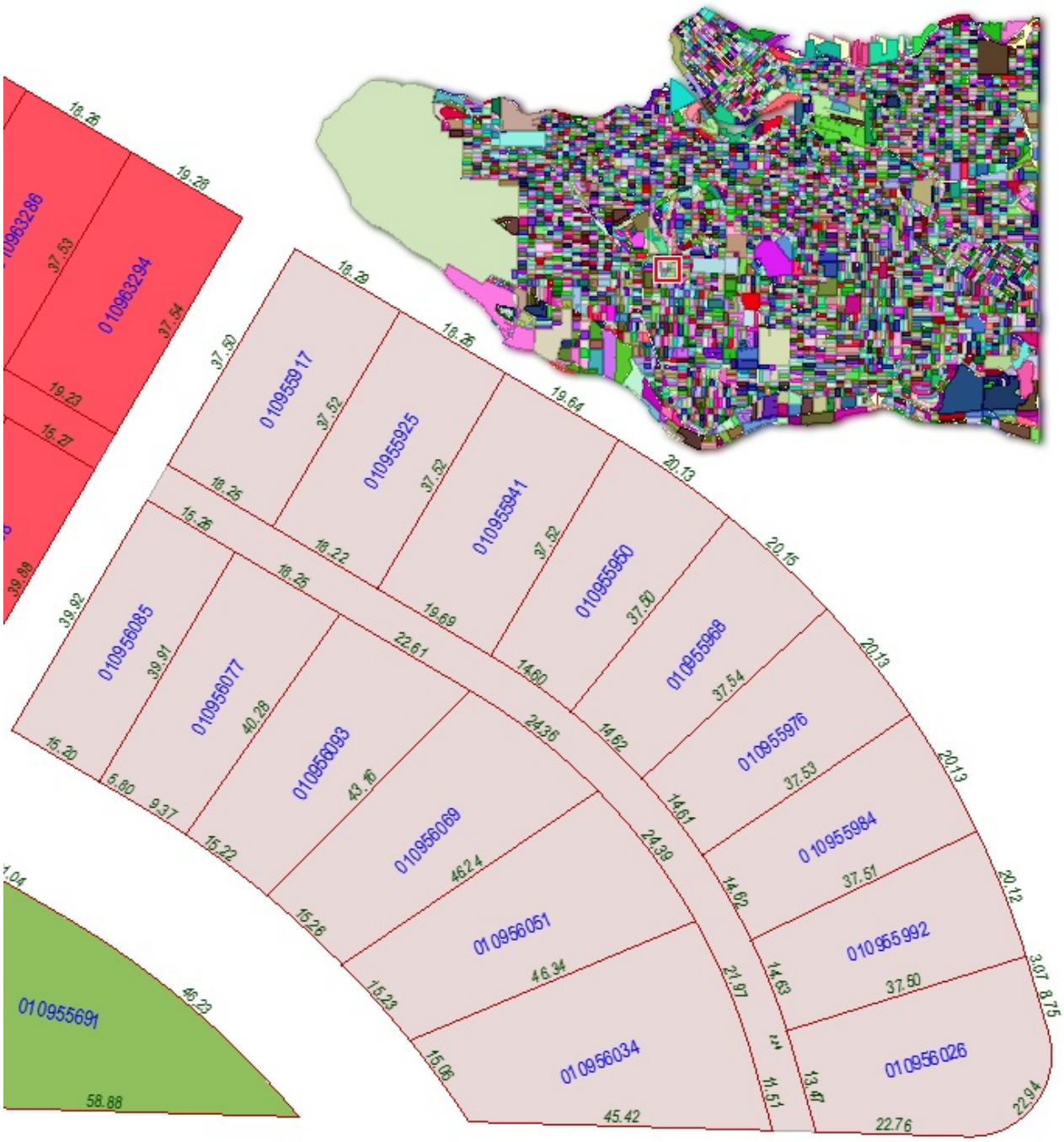
For example, Alpine Shire Council integrates a range of complex source data from varying formats, including digital elevation models, Esri Shapefiles, and more. Calculations are applied to the integrated data to yield spatial and non-spatial results, which are made accessible in real time via an iPad app. (Read about the BAL Plan app.)

BAL Plan is an iPad app that informs users in real time about the risk of bushfires in a given area. It makes critical data publicly available and is the result of a data integration project.

6. Easy data collaboration

With accessibility comes easier collaboration. Anyone who works with your data will find it easier to use brain power now that they can actually use the data in the format they require. Whether collaboration involves sharing among internal teams and applications, or across organizations, integrated data is more complete because it has more contributors.

For example, the State of Indiana needed to combine specialized data from 92 counties in the form of points, parcels, streets with address ranges, and boundaries into an existing online GIS portal. Their data integration plan resulted in a non-invasive, easy way for all counties to collaborate on the data portal, despite each having a separate data management system. (Read about how Indiana harmonizes data in a central database.)



Data integration makes it easier to collaborate on potentially overwhelming information.

7. Understanding data means smarter business decisions

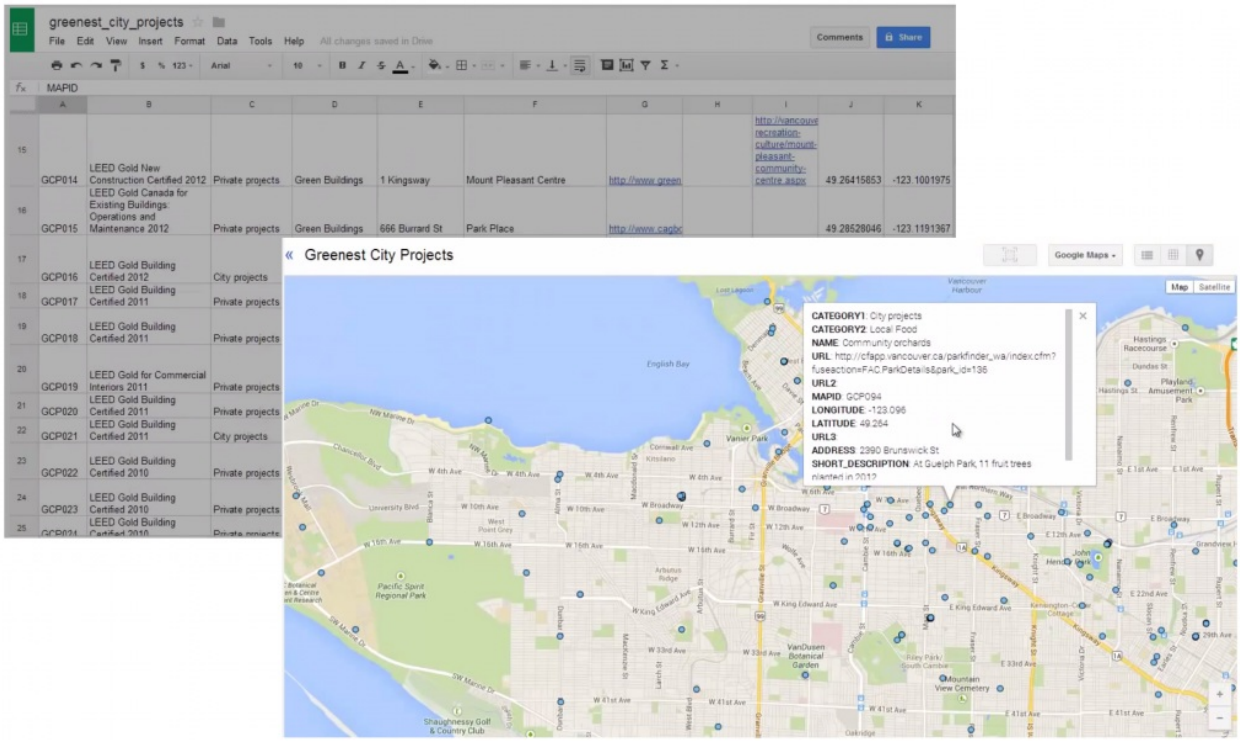
Applying location intelligence to a non-spatial dataset, like this CSV file, opens up new opportunities for decision-making.

Integrated data means transparent processes within your organization. By giving people the flexibility to use your data in whatever system, you're giving them the opportunity to better understand the information. It's much easier – and more informative – to navigate through organized repositories that contain a variety of integrated datasets.

Applying location intelligence to your dataset, for instance, makes it spatially comprehensive and offers new levels of insight around that dataset, which leads to better decision-making.

For example, the Skogskyrkogården data integration project combines databases, Esri Shapefiles, and JSON source data into one AutoCAD output that enables them to plan 100 years into the future.

Integrating with other data types can also mean conforming to interoperability standards to save on business costs. Further, open source data formats offer many advantages for businesses.

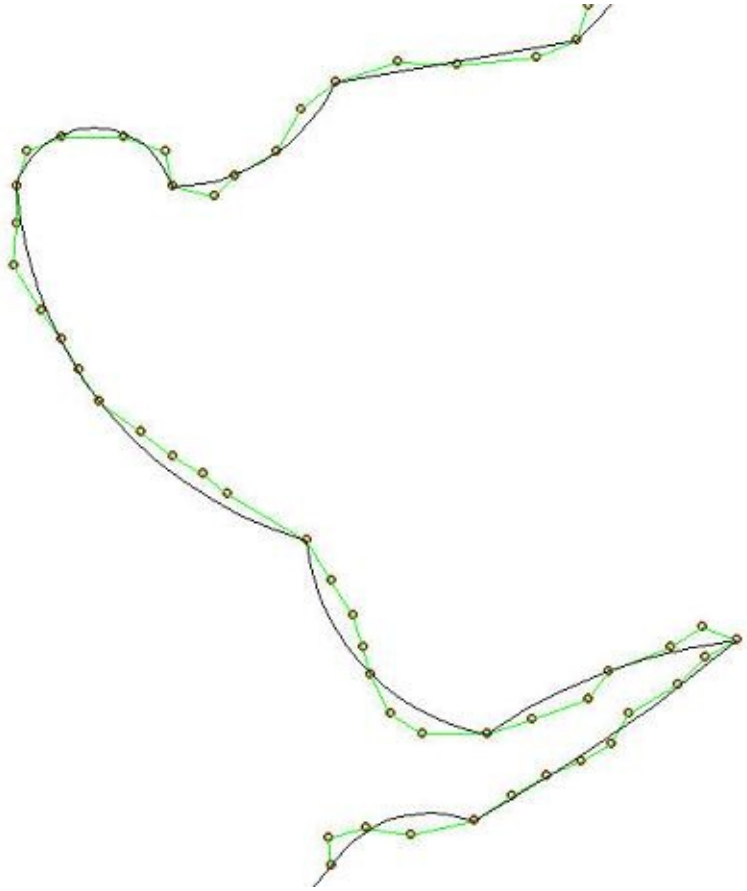


Applying location intelligence to a non-spatial dataset, like this CSV file, opens up new opportunities for decision-making.

8. Data integrity

Data integration technology should cleanse and validate the information passing through. Obviously, we all want our data to be robust and high quality. An integration strategy ensures data is free of errors, inconsistencies, and duplication.

For example, the BC Transit system is made up of disconnected information on bus stops, vehicles, schedules, routes and ridership, and streets. These singular systems have made central transit information vulnerable to inaccuracy. A data integration strategy ensures BC Transit's critical data is accurate and of high quality. (Read about BC Transit's data integration and validation process.)

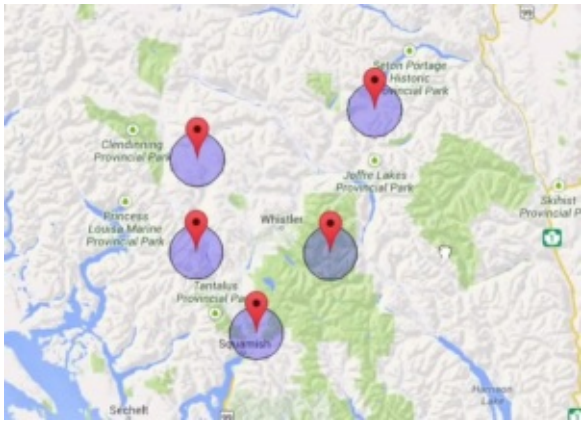


These line segments have been smoothed into arcs as part of a CAD-GIS data integration strategy.

9. Make your data live

An integrated data solution makes it easy to keep information up to date. One input can propagate across all integrated systems, keeping your data current. In fact, your data can even be real-time if a server or cloud solution is part of the integration strategy.

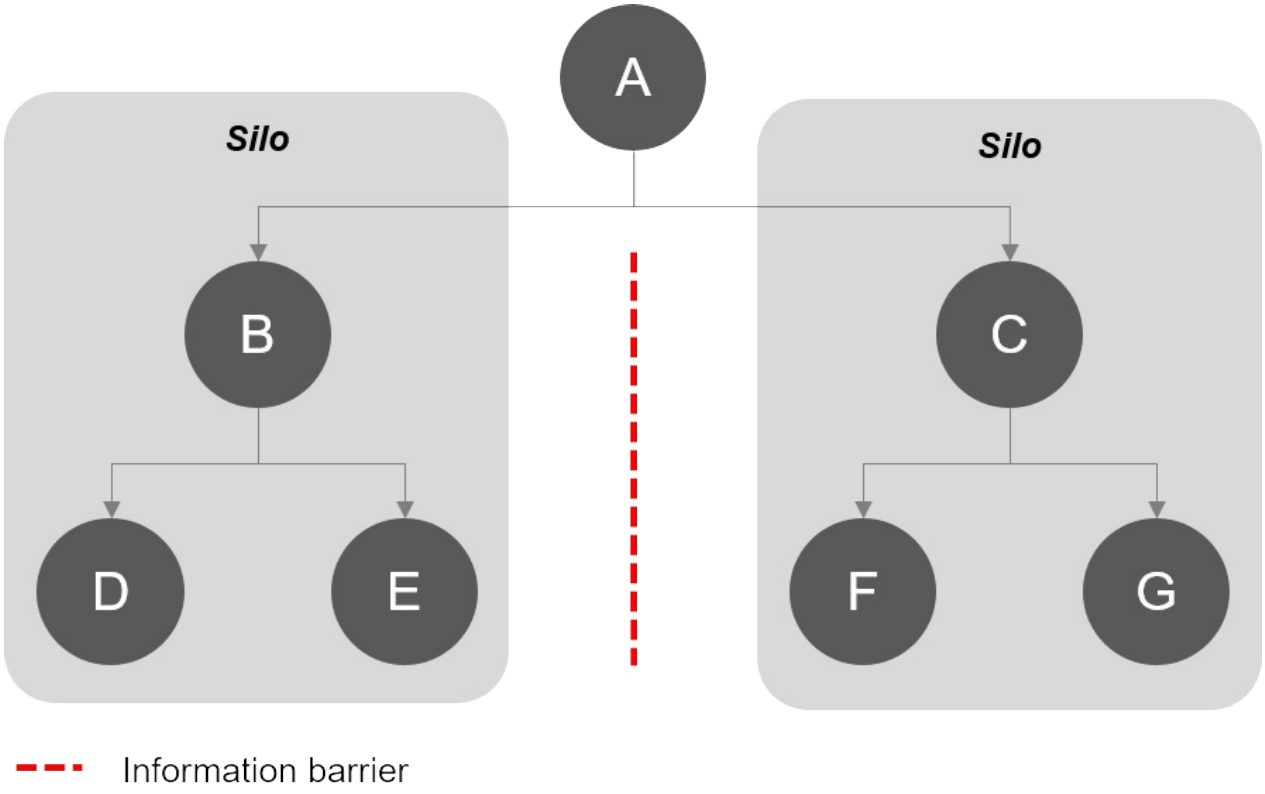
For example, The Weather Network integrates non-spatial sensor data and metadata and outputs it to spatial applications like Google Earth or ArcGIS Earth. From there, real-time KML updates and email alerts can be leveraged. (Read about the Pelmorex Lightning Detection Network's real-time data integration.)



Integrating with cloud systems is one way of enjoying the benefits of live data.

Conclusion: avoiding data silos

Integrating data not only provides the benefits discussed above; it also can prevent the problem of data silos – fixed repositories of information. Like farm silos, their contents are isolated from the outside world. Did you know the feed inside a farm silo is often fermented? Did you know the air and liquid emitted during fermentation is toxic? Data trapped in silos 'ferments' and becomes useless.



Data silos diagram by Theodoxxi (CC BY-SA 4.0, 2016).

Even if data doesn't need to be accessed regularly, siloed data is still inefficient to work with, impossible to collaborate on, and runs a high risk of conflicting with external data. By actively integrating your data, you avoid the dreaded silo and gain the potential outlined above. Any data type can live in a wide variety of sources. Developing an integration strategy—both the technical and business aspects—is critical to ensuring your organization's data reaches its maximum potential.

Further reading

For further reading, why not browse articles tagged with data integration on Safe Software's blog, the makers of FME.

What is FME?

FME (the Feature Manipulation Engine) is a data translation and transformation tool for solving problems of data interoperability, without the need for coding.

Safe Software created FME in 1993 to aid forestry companies exchange maps with the British Columbia provincial government. It was technically possible to share maps then, but only after hours of fighting with the data and often much information was lost. FME was built for better data integration: allowing the exchange of maps in different formats without losing any information.

FME is still used widely in forestry and other industries using Geographic Information Systems (GIS) for mapping and spatial analysis. The scenarios in this module are based on GIS use-cases, but the skills covered are widely applicable to any industry that requires data manipulation. If you would like to learn more about GIS, these articles from the University of Wisconsin and GIS Lounge provide useful information.

FME Products

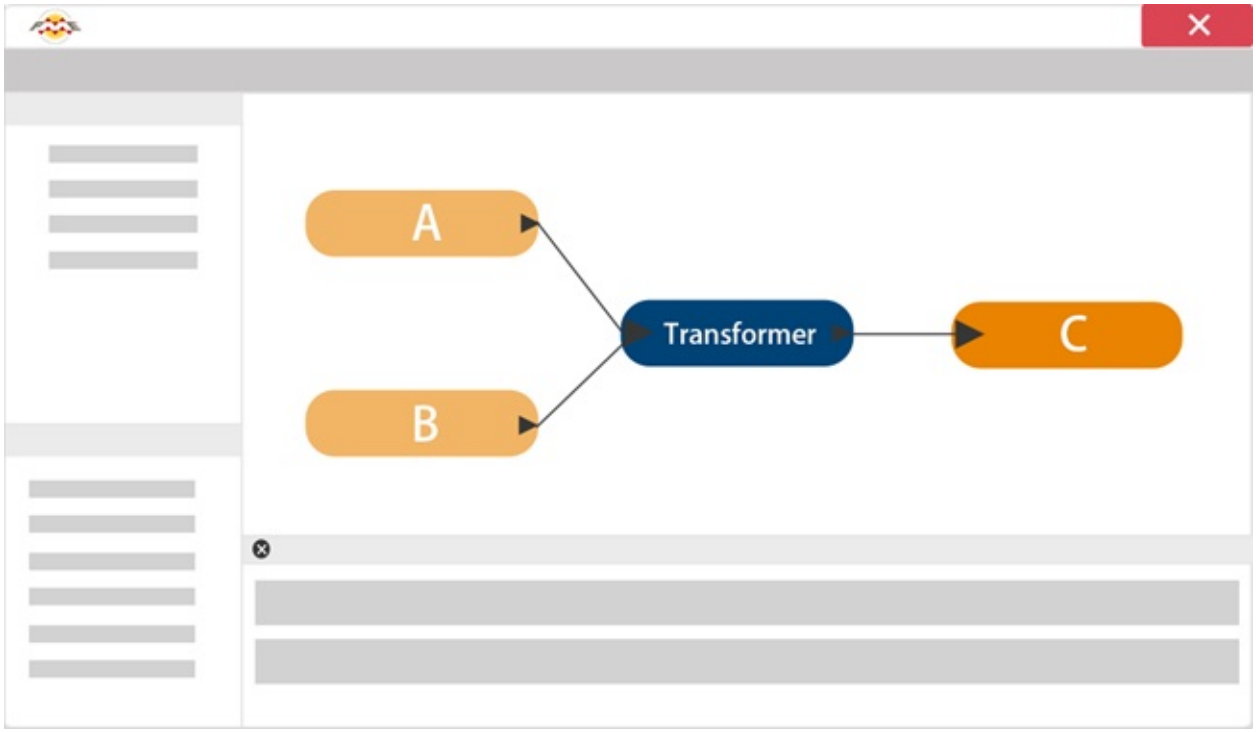
This module covers using FME Desktop for data translations and transformations at the desktop level. FME Desktop is one piece of software in a larger ecosystem:

- FME Desktop lets you connect and transform data.
    - For example, taking an Excel spreadsheet of business information and addresses and adding it to a database of neighborhood demographic information stored in a Geographic Information System.
- FME Server provides enterprise-level automation.
    - For example, allowing business licensing officers working for a city government to add new business licenses to the database created in Desktop in real time by sending an email or filling out a web form.
- FME Cloud is the hosted version of FME Server.
    - For example, the server that carries out the operation in the example above could be hosted in the cloud, rather than directly by the City.

FME Desktop consists of a number of different tools and applications. The two key applications are FME Workbench and the FME Data Inspector. We will cover FME Workbench and FME Data Inspector in the Lab Demonstration.

Extract, Transform, and Load

FME is sometimes classed as an **ETL** application. ETL stands for Extract, Transform and Load. It is a data warehousing tool that extracts data from multiple sources (here A and B), transforms it to fit the users' needs and loads it into a destination (C):

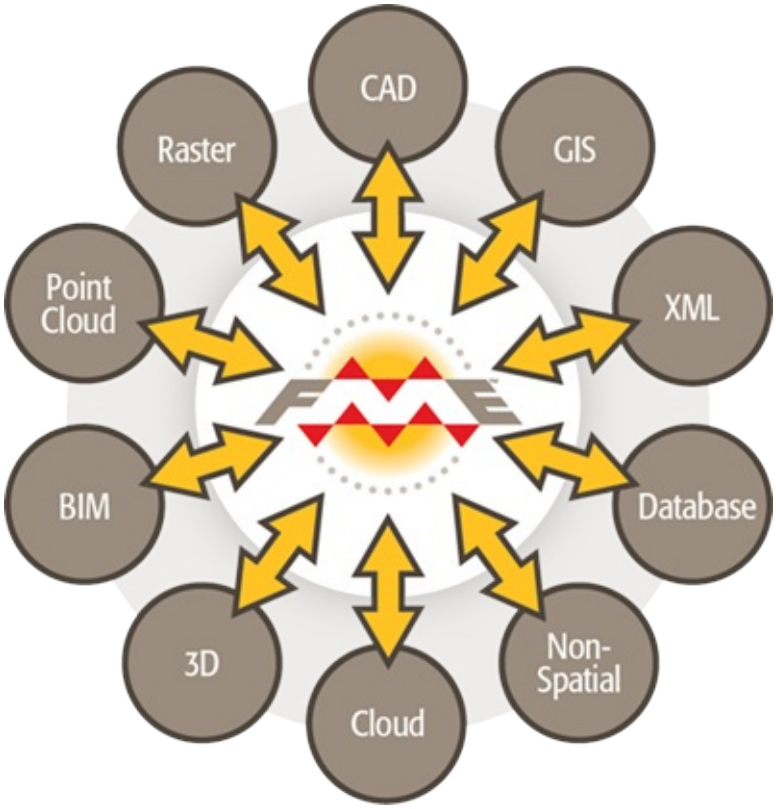


While most ETL tools process only spreadsheet (i.e. tabular) data, FME also has the processing capabilities required to handle spatial data, hence the term **Spatial ETL**.

How FME Works

At the heart of FME is an engine that supports an array of spatial and tabular data types and formats; GIS, CAD, BIM, Point Cloud, XML, Raster, databases, and many more.

The capability to support so many data types is made possible by a rich data model that handles all possible geometry and attribute types.



Most importantly, the data translation process is seamless to the user; FME automatically converts between data types as required, and automatically substitutes one attribute or geometry type for another where the destination format does not support it.

Data Translation

FME can both translate and transform data.

At its heart, FME is a data translation tool, and this is usually the first aspect users wish to learn about.

Data translation consists of taking data in one format and translating it into another format. For example, taking a Word document (.docx file format) and turning it into a Portable Document Format (.pdf). Ideally, the process of data translation will preserve all or as much information from the source data as possible.

Incompatible data formats are a major barrier to moving data between systems. FME's data translation or conversion capabilities ensure data can be used effectively in many applications.

Data Transformation

FME also can change or transform your data. For example, maybe you want to remove references to a list of confidential names as you translate your data from .docx to .pdf. Or, to draw on the earlier example, maybe you want to get an email or SMS alert every time a new business license is added to the city database in a certain neighborhood. You can do this using data transformation.

Miss Vector says...

Attention students! I'm Miss Vector, FME schoolteacher. I'm here to test you on FME. I hope you don't get these questions wrong!

Q) ETL is an acronym for...?

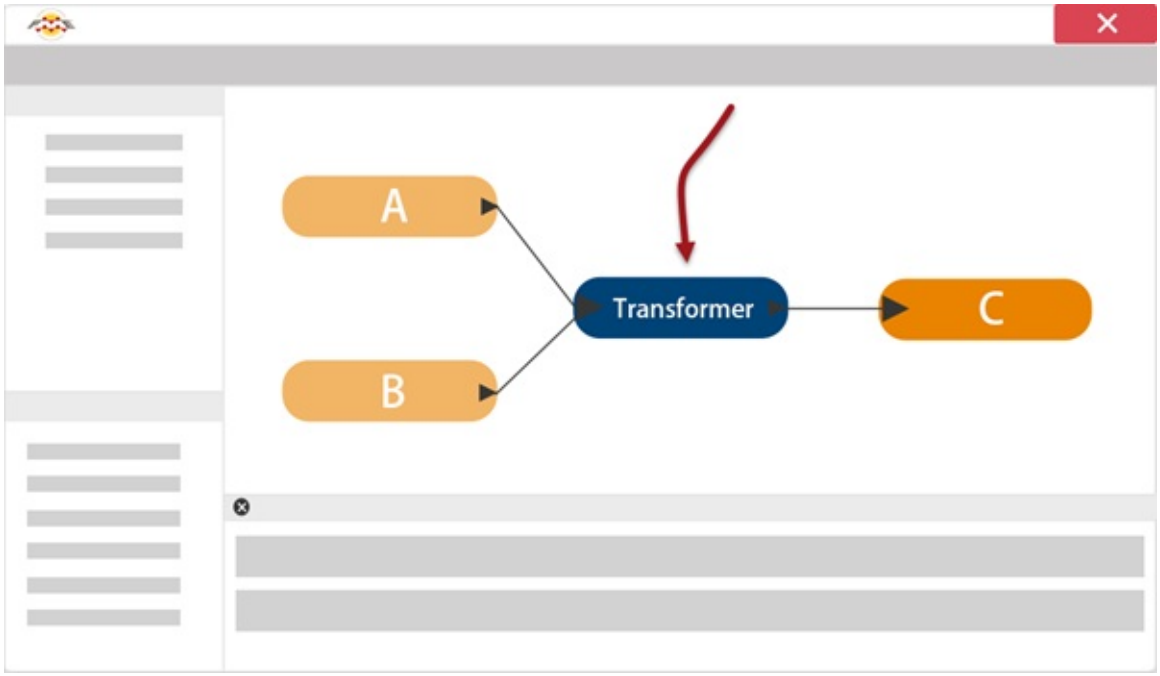
1. Extra-Terrestrial Lifeform
2. Extract, Transform, Load
3. Express Toll Lane
4. Eat, Transform, Love

Q) FME can seamlessly translate between so many formats because it has...

1. A sentient data dictionary
2. A retro-encabulator
3. A rich data model
4. A core of unicorn hairs

What is Data Transformation?

Data Transformation is FME's ability to manipulate data during a data translation, even to the extent of having an output greater than the sum of the inputs! The transformation step occurs during the process of format translation. Data transformation is important to ensure that key elements of a dataset are preserved.



Data Transformation Types

Data transformation can be subdivided into two distinct types: *Structural Transformation* and *Content Transformation*.

Structural Transformation

Structural transformation is perhaps better called 'reorganization'. It refers to FME's ability to channel data from source to destination in an almost infinite number of arrangements.

This includes the ability to merge data (as in the image above), divide data, re-order data, and define custom data structures. This includes the ability to merge data (as in the image above), divide data, re-order data, edit the name of attributes, and define custom data structures.

Transforming the structure of a dataset is carried out by manipulating its **schema**, or structure.

Content Transformation

Content transformation is perhaps better called ‘revision.' It refers to the ability to alter the content of a dataset.

Manipulating a feature's geometry or calculating new attribute values is the best example of how FME can transform content.

Content transformation can take place independently or alongside structural transformation.

Mr Flibble says...

Mr. Flibble - certified FME jester - here to entertain you. Here's a riddle, but can you solve it?

The most common format translation defined with FME is from Esri Shapefile to which format?

1. Esri Geodatabase
2. AutoCAD DXF/DWG
3. Google KML
4. Esri Shapefile

If you're in a class, have a group vote!

Lab

In this section of the module, the instructor can walk students through the basic functionality of FME Workbench and FME Data Inspector. If your lab space permits it, students can follow along.

This content could instead be delivered as a lecture or as part of a lab, if the instructor prefers.

This section contains exercises that students will complete using FME Workbench and FME Data Inspector. It is designed for students to complete in a lab with instructor supervision or on their own time as an assignment.

Refer to the Lab Questions as you complete the exercises.

Learning Objectives

Conceptual

Students will learn:

1. **FME Workbench** is an application to graphically define a translation and the flow of data within it. The definition of a translation is known as a workspace and can be saved to a file for later use.
2. **Quick translation** is the technique of carrying out a translation generated by FME, without further editing.
3. The **Generate Workspace** dialog is the main method by which a quick translation can be set up in FME Workbench.
4. Data inspection is a technique for checking and verifying data before, during, and after a translation. The **FME Data Inspector** is a tool for inspecting data.
5. A **schema** describes a dataset's structure, including its feature types, attributes, and geometries.
6. **Schema editing** is the act of editing the destination schema to better define what is required out of the translation.
7. The act of joining the source schema to the destination is called **schema mapping**. Differences between the two schemas lead to **structural transformation**.
8. **Content transformation** is the modifying of data content during a translation. In FME Workbench data transformation is carried out using objects called transformers.

9. **Transformers** are the building blocks used in FME Workbench. Each transformer has a specific function. They can be used alone in a simple workspace, or combined to create complicated processes.
10. **Group-by processing** is when FME transformers carry out transformations on a whole set of features at once, in contrast to **feature-based processing**, when transformations are carried out one feature at a time.
11. Working with spatial data requires setting and reprojecting data so they share a common **coordinate system**.
12. **Partial runs** allow you to only run part of a workspace, making debugging much easier.
13. FME Workbench users should follow **best practices** by using **annotations** and **bookmarks** to organize their workspaces so other users can understand them.

Skills

Students will gain the ability to...

1. **Open a workspace** in FME Workbench and run it
2. **Start FME Workbench** and set up a **quick translation**
3. **Start the FME Data Inspector**, and to **open and add datasets**
4. **Navigate** a dataset and to **query** features within it
5. Control Data Inspector **symbology** and **display** characteristics
6. Set **background maps** in the Data Inspector
7. Edit a **writer schema**
8. **Restructure** data by mapping a reader to a writer schema, both manually and through transformers
9. **Locate transformers** in Workbench and **use them** to transform data content
10. Set **transformer parameters** in either the Parameter Editor or transformer dialogs
11. Define feature groups using the **Group-By** option
12. **Reproject** data using Workbench

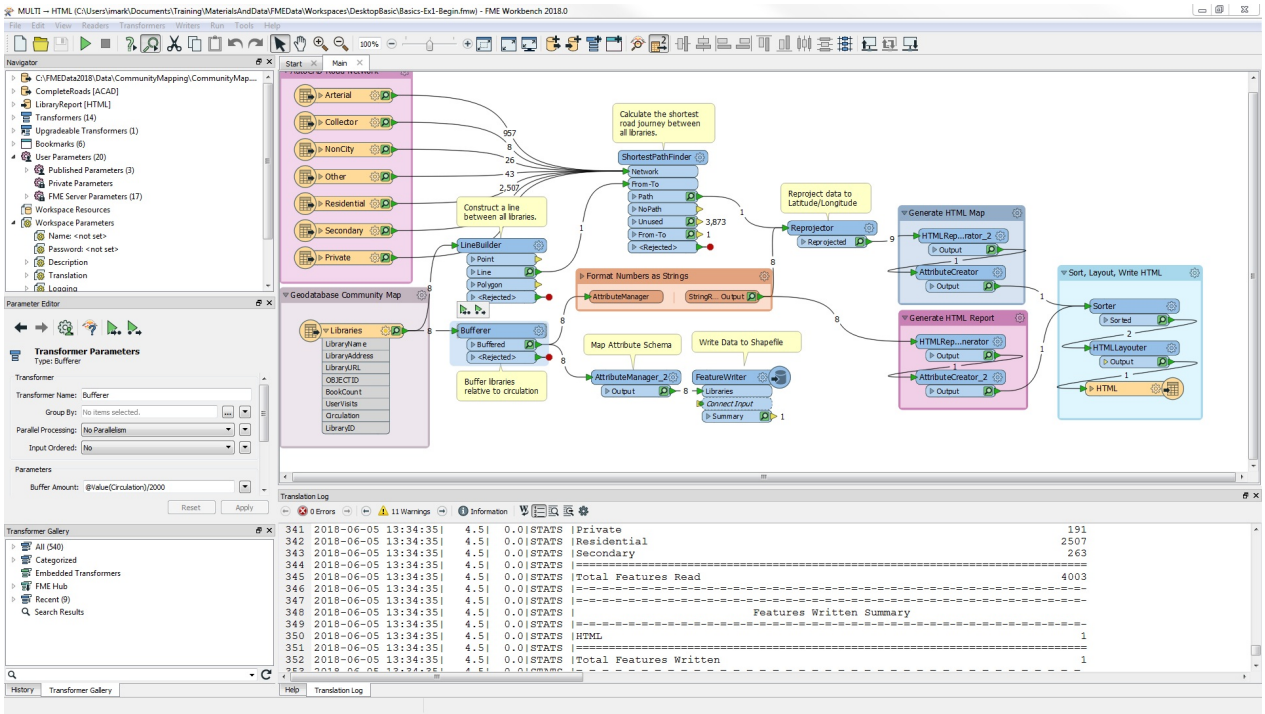
FME Desktop

This module is about FME Desktop. FME Desktop is for data translations and transformations at the desktop level (as opposed to FME Server, which is an enterprise-level, web-based product).

FME Desktop consists of a number of different tools and applications. The two key applications are **FME Workbench** and the **FME Data Inspector**.

FME Workbench

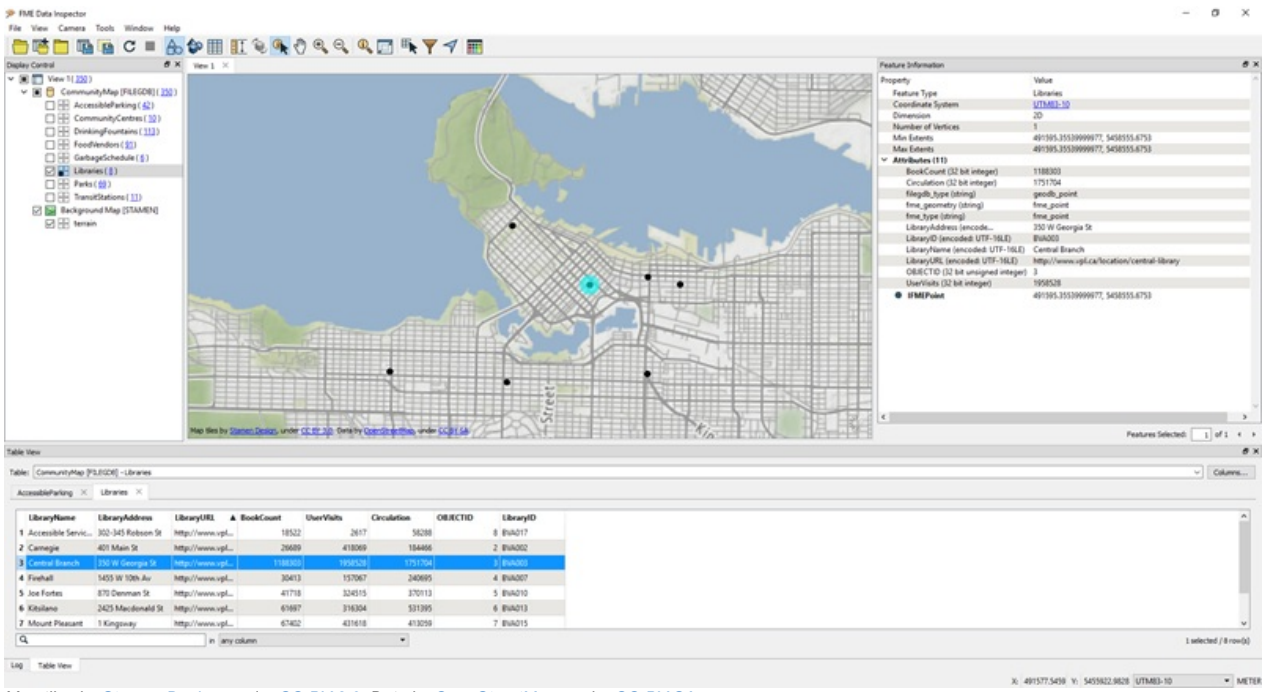
FME Workbench is the primary tool for defining data translations and data transformations. It has an intuitive point-and-click graphic interface to enable translations to be graphically described as a flow of data.



Workbench is not a standalone tool. It is fully integrated to interact with other FME Desktop applications such as the FME Data Inspector.

FME Data Inspector

The FME Data Inspector is a tool for viewing data in any of the FME supported formats. It is used primarily for previewing data before translation or reviewing it after translation.



Map tiles by Stamen Design, under CC-BY-3.0. Data by OpenStreetMap, under CC-BY-SA.

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:

Readers and Writers

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

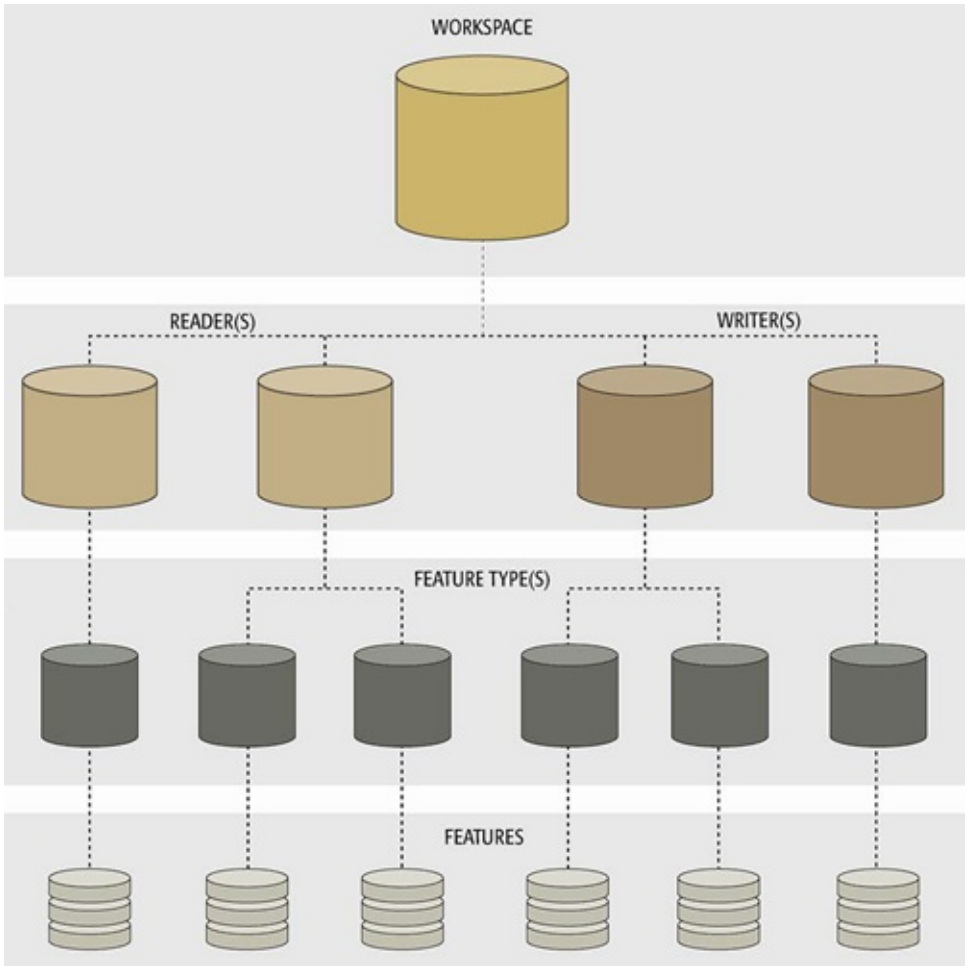
Features

Features are the smallest single components of an FME translation.

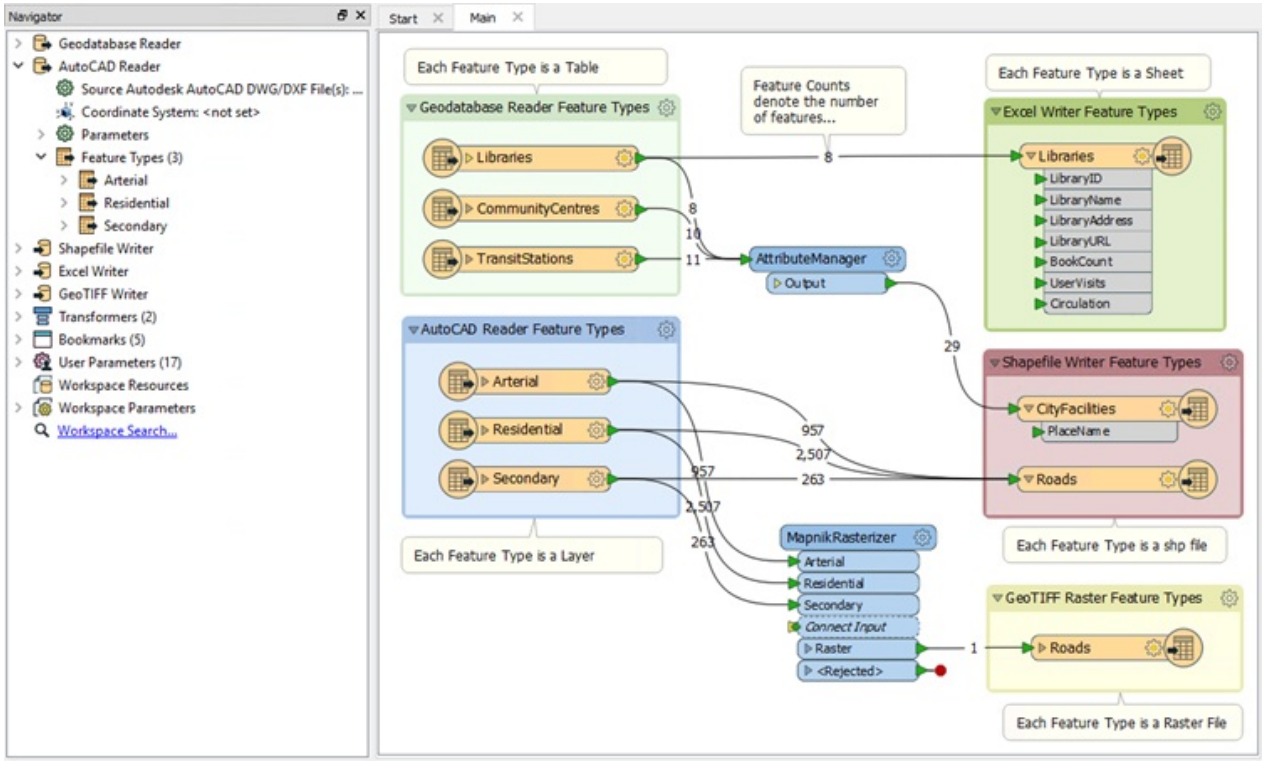
They aren’t individually represented within a workspace, except by the feature counts on a completed translation.

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.

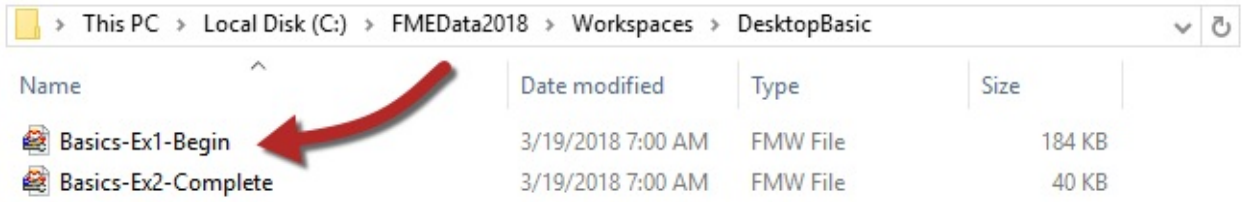
Exercise 1	Opening and Running a Workspace
Data	Libraries (Esri Geodatabase) Roads (AutoCAD DWG)
Overall Goal	To open and run an FME workspace to explore what it can do with data
Demonstrates	Opening and running a workspace
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Basics-Ex1-Begin.fmw
End Workspace	N/A

Rather than trying to explain what FME is and does, let's try it for ourselves!

1) Locate Workspace File

When translations and transformations are defined in FME, they can be saved in a .fmw file.

Using a file explorer, browse to the file listed above

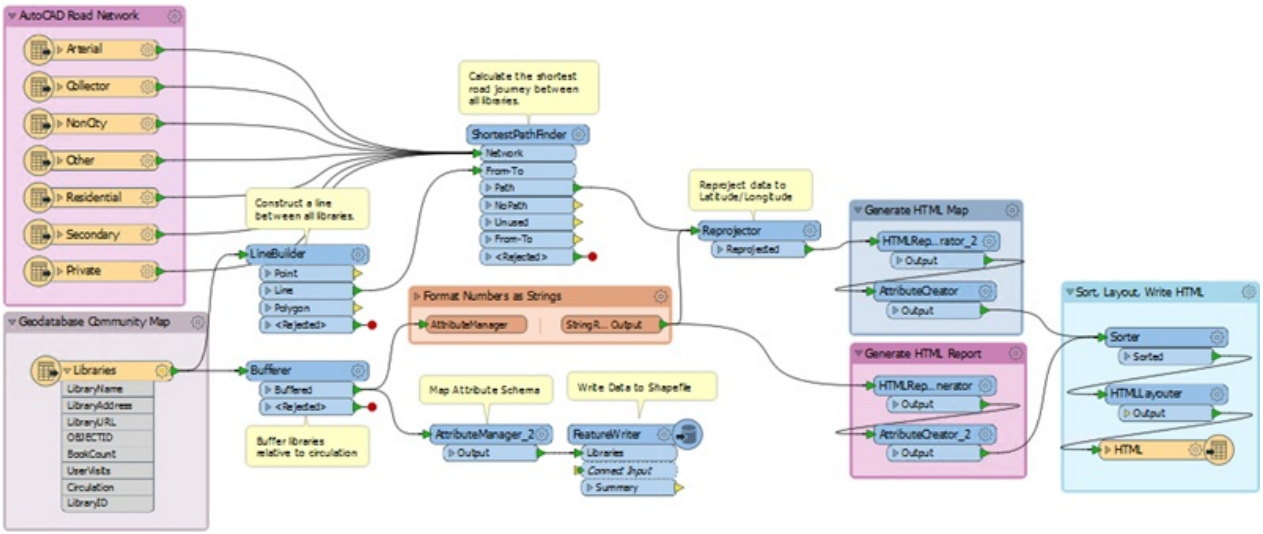


Double-click on the file. It will open an application called FME Workbench.

2) Explore FME Workspace

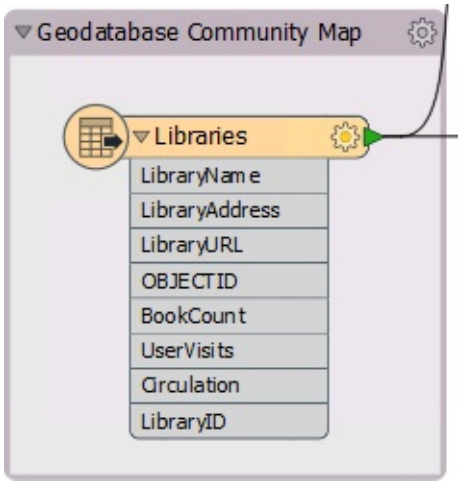
In FME Workbench, dismiss any Getting Started dialog that may open by clicking "Don't Show Me Again."

The main part of the application will look like this:



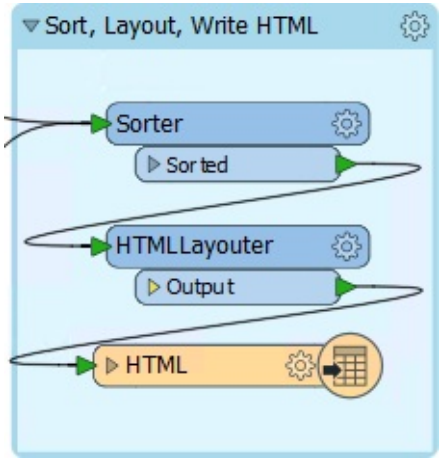
This part we call the canvas. It is where the translation and transformation of data is defined graphically. Although it might look complicated, it does not take much practice with FME to create workflows of this type.

Examine the left-hand side of the canvas:



This is where we read data, in this case a table of libraries from an Esri Geodatabase. This is the "E" (Extract) part of ETL.

Now look at the right-hand side:



This is where we write data, in this case a report of the libraries in HTML format. This is the "L" (Load) part of ETL.

In between the reader and writer are objects that transform data. They represent the "T" (Transform) part of ETL.

Labels and other annotations show us what the workspace does. It:

- Reads both roads (AutoCAD DWG) and libraries (Esri Geodatabase)
- Calculates the shortest road route taking in all libraries
- Creates circles whose diameter is relative to a library's book circulation
- Creates a HTML report and a HTML map of the libraries

- Writes the data to HTML and also to Esri Shapefile

Police-Chief Webb-Mapp says...

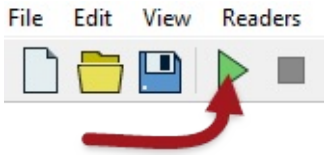
I'm the police chief, responsible for tracking down crimes against FME.

So, let's make sure you get the terminology right. The application itself is called FME "Workbench", but the process defined in the canvas window is called a "Workspace". The terms are so similar that they are easily confused, but please don't, otherwise I will have to send my grammar squad to arrest you!

Although mistreating FME terminology is a minor offence, the ignominy of being caught is long lasting!

3) Run FME Workspace

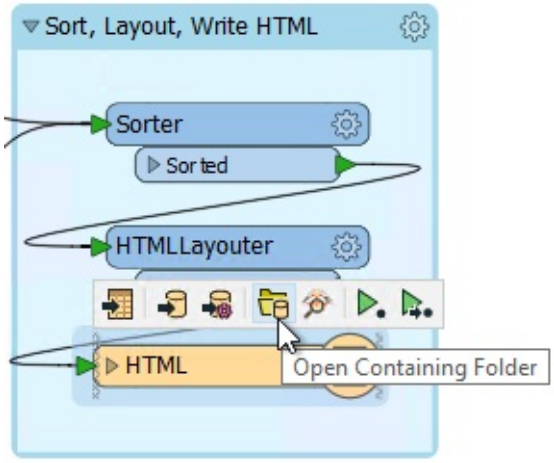
Let's run this workspace. To do this click on the Green run button on the Workbench toolbar:



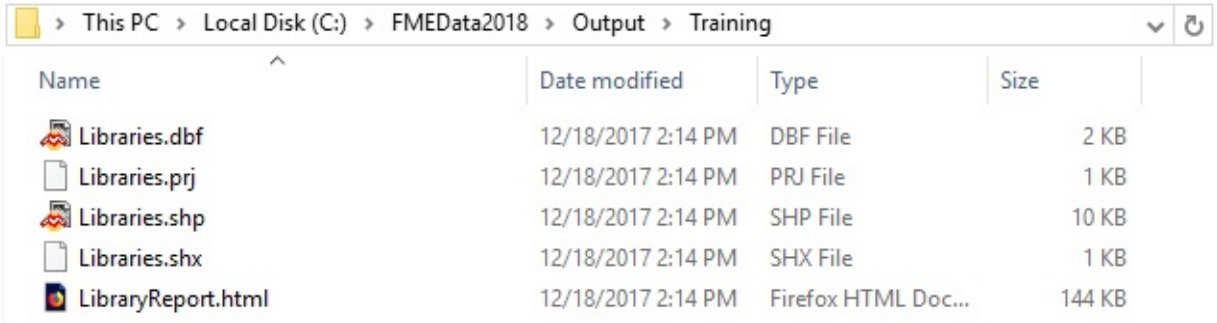
The workspace will now run. As it does so you will see messages pass by in a log window. You may also see numbers appear on the canvas connections and green annotated icons on each object. We'll get to what these are for later!

4) Locate and Examine Output

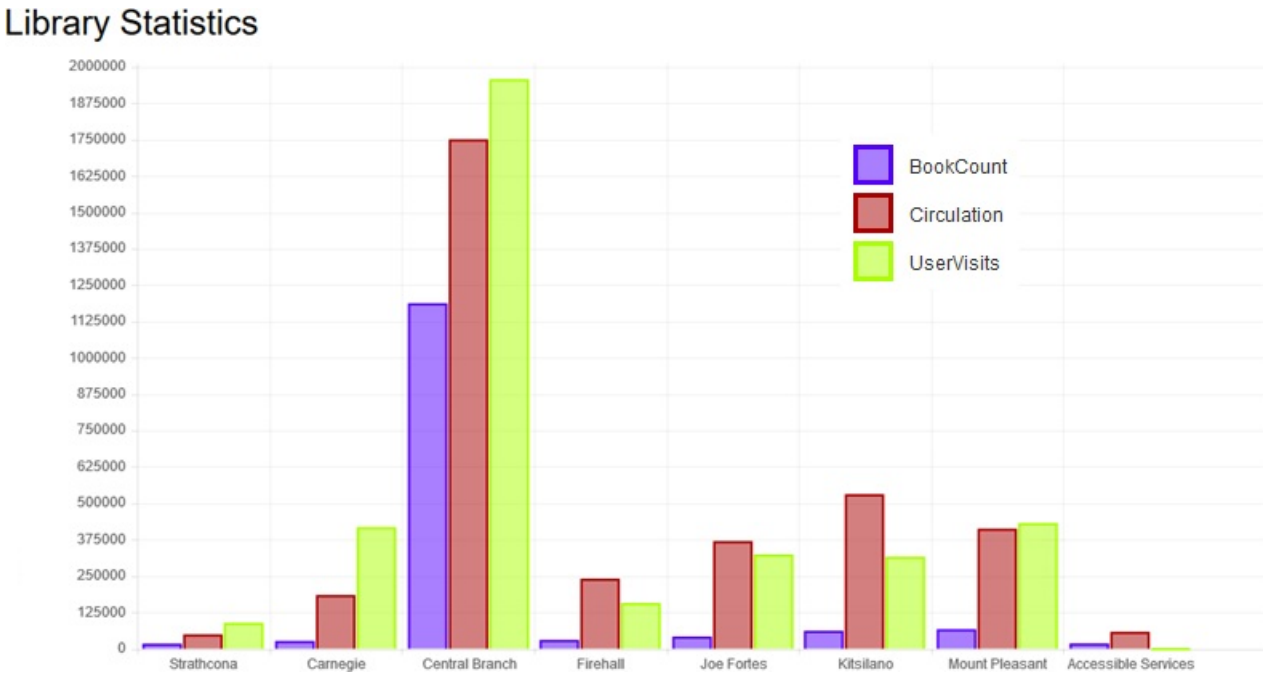
Once the translation is complete, click on the HTML writer object on the canvas. Choose the option to Open Containing Folder:



In the Explorer dialog that opens you will find both the HTML output and the Shapefile dataset:



Open a web browser such as Firefox or Chrome. Open the output file created by FME (usually Ctrl+O or File > Open is the easiest way). You will see a table of libraries, a graph of library statistics, and an interactive map showing where the libraries are located. All this has been generated by FME from the incoming Geodatabase points and attributes:



Miss Vector says...

This small demonstration illustrates the power of FME. This workspace read data from multiple spatial datasets and wrote it out to datasets in both spatial and "tabular" formats. In between it carried out a series of transformations and spatial analyses, buffering and reprojecting the data, and creating added value and information.

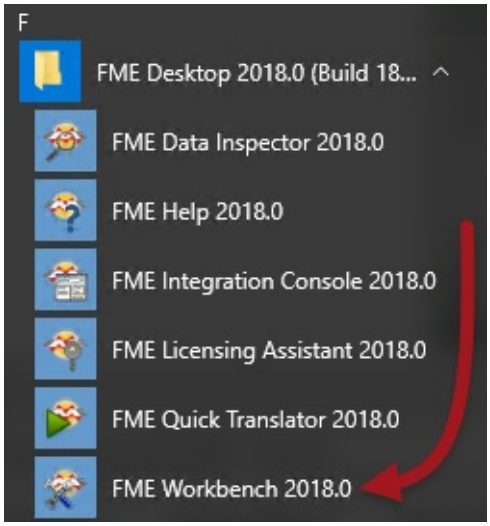
CONGRATULATIONS

By completing this exercise you have learned how to:

- *Open an FME workspace*
- *Run an FME workspace*
- *Locate the output from an FME workspace*

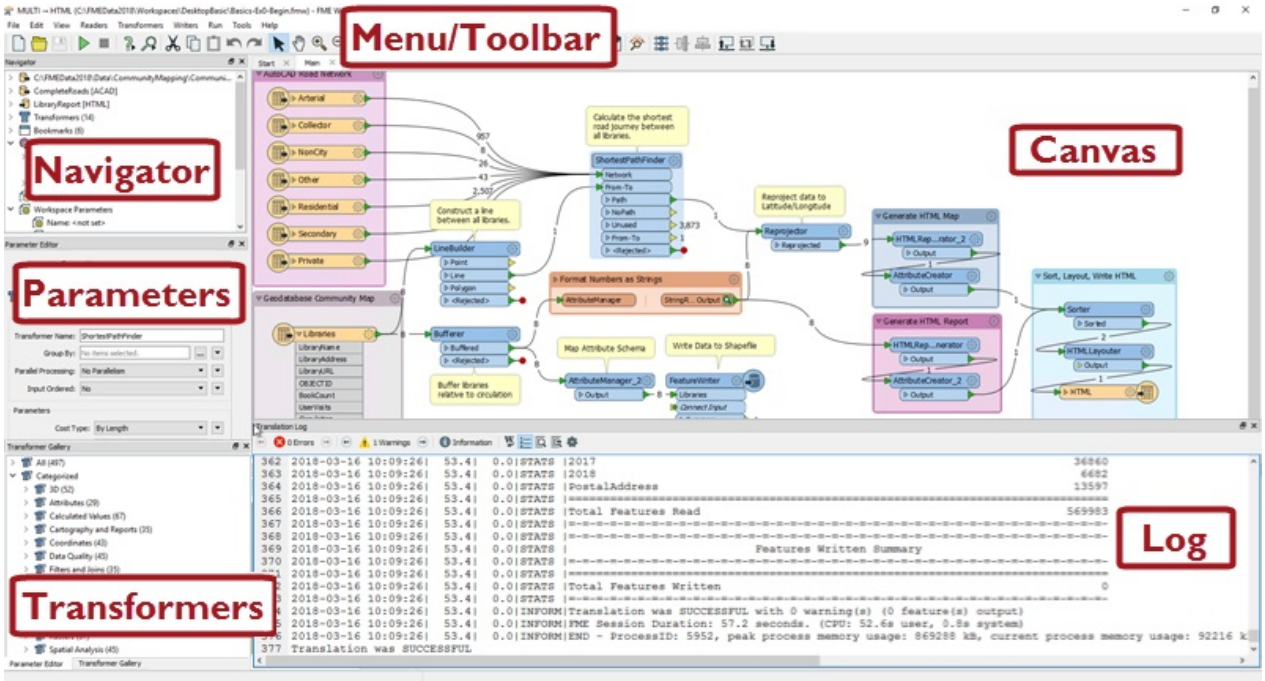
Introduction to FME Workbench

Let's take a closer look at FME Workbench, firstly how to start it. To start Workbench - besides double-clicking an fmw file - locate Workbench in the Windows start menu:



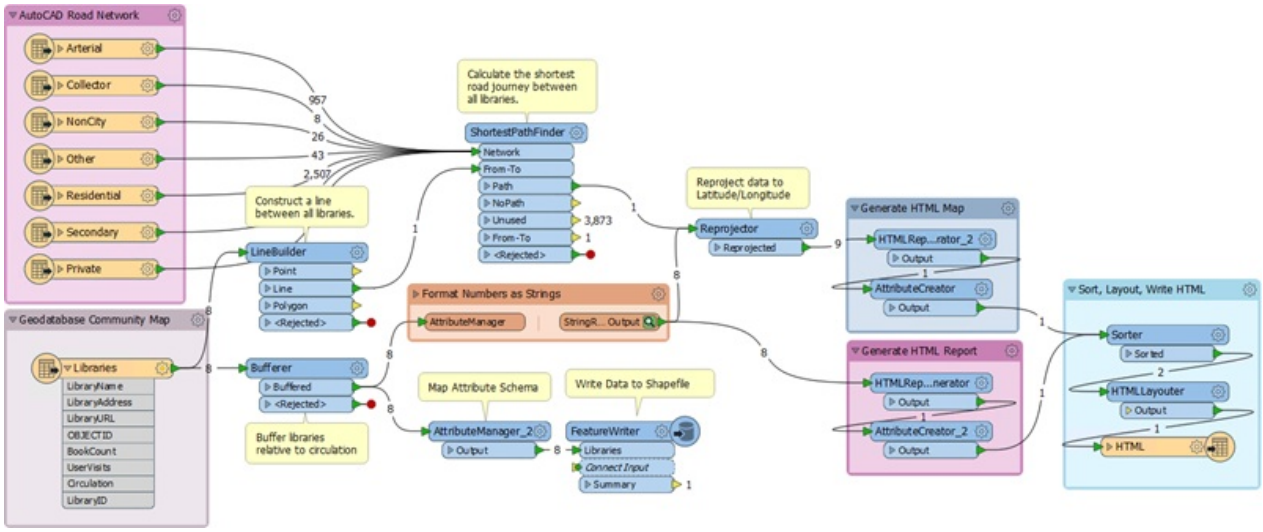
Major Components of FME Workbench

The FME Workbench user interface has a number of major components:



Canvas

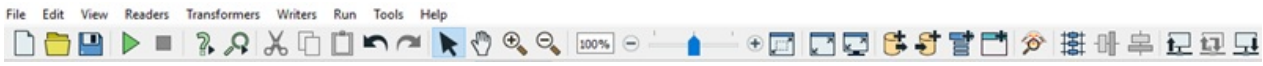
The FME Workbench canvas is where to define a translation. It is the primary window within Workbench:



By default the workspace reads from left to right; data source on the left, transformation tools in the center, and data destination on the right. Connections between each item represent the flow of data and may branch in different directions, merge together, or both.

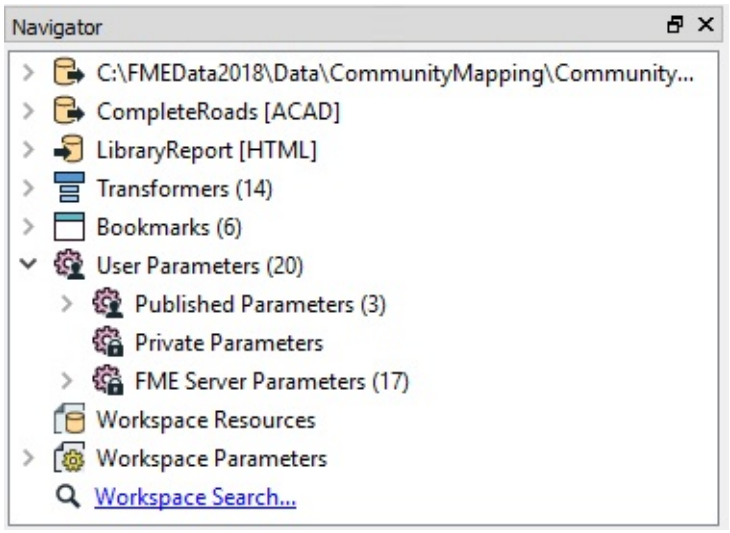
Menu/Toolbar

The menubar and toolbar contain a number of tools: for example, tools for navigating around the Workbench canvas, controlling administrative tasks, and adding or removing readers/writers:



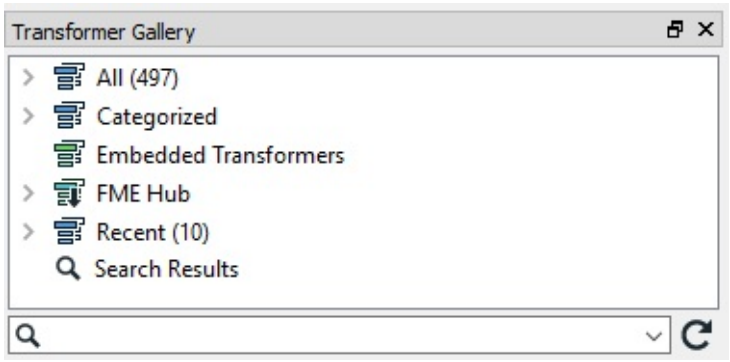
Navigator

The Navigator window is a structured list of parameters that represent and control all of the components of a translation:



Transformer Gallery

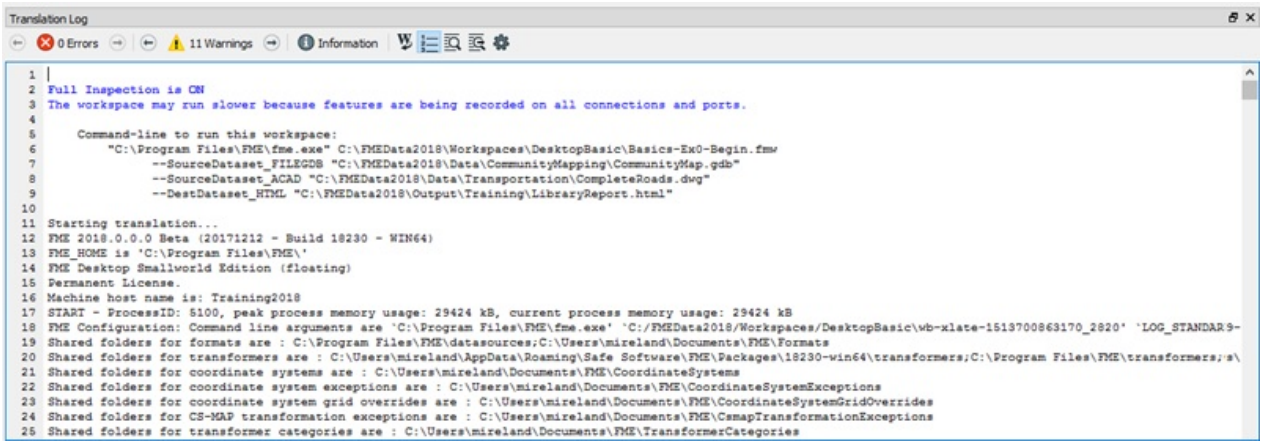
The transformer gallery is a tool for the location and selection of FME transformation tools.



The number of transformers (above, 484) will vary depending on the version of FME and any optional custom transformers installed:

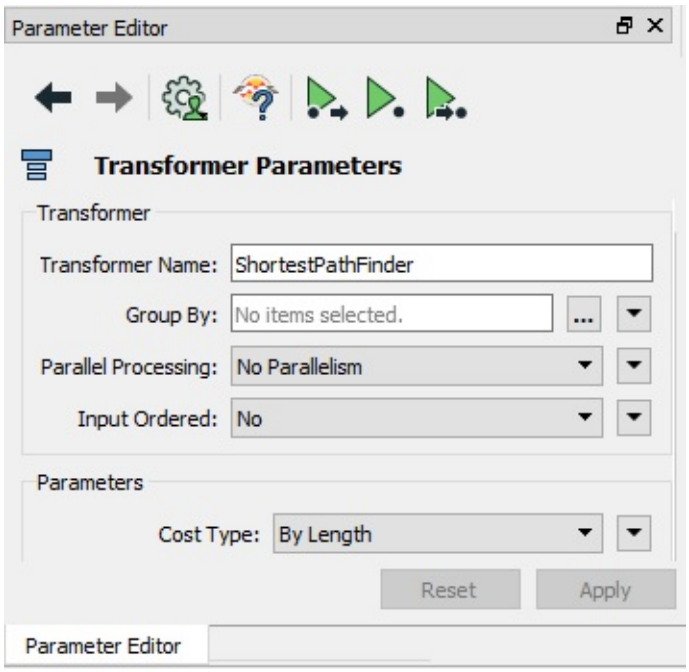
Translation Log

The translation log reports on translations and other actions. Information includes any warning or error messages, translation status, length of translation, and the number of features processed:



Parameter Editor Window

The Parameter Editor window is for editing parameters for objects on the canvas window:



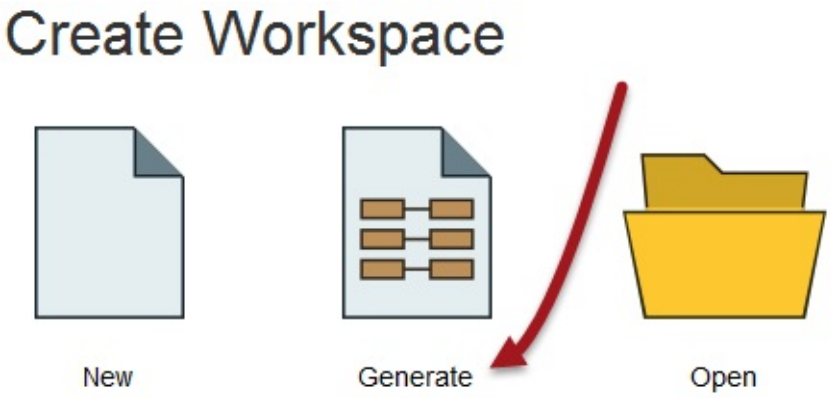
As the next section will show, you can open or close these windows at will, or rearrange them into any configuration that you like.

Creating a Translation

Workbench’s intuitive interface makes it easy to set up and run a simple format-to-format ('quick') translation.

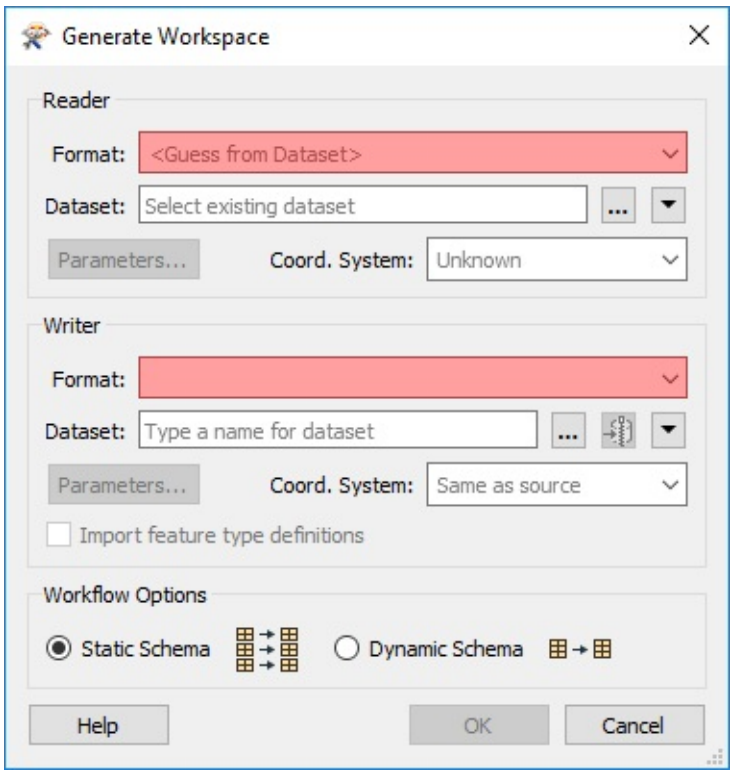
The Start Tab

The Start Tab in FME Workbench includes different ways to create or open a workspace. The simplest method is Generate Workspace:



Generate Workspace Dialog

The Generate Workspace dialog condenses all the choices to be made into a single dialog box. It has fields for defining the format and location of both the data to be read, and the data to be written.



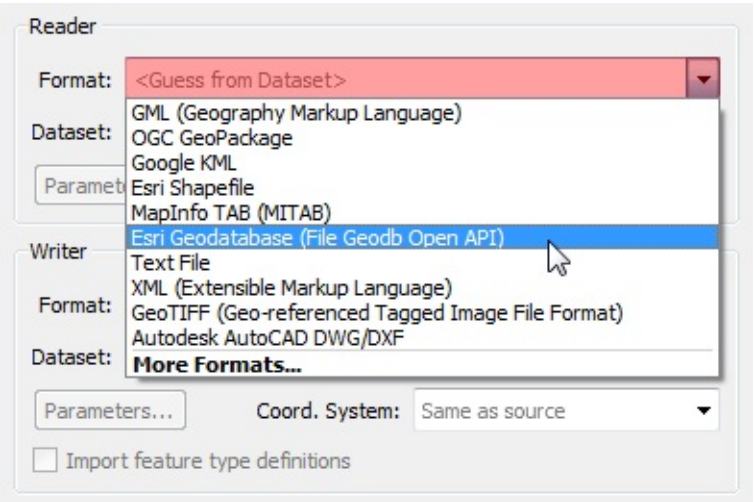
Red coloring in an FME dialog indicates mandatory fields. Users must enter data in these fields to continue. In most dialogs, the OK button is deactivated until the mandatory fields are complete.

Format and Dataset Selection

A key requirement is the format of the source data. All format selection fields in FME are both a pull-down menu and a text entry field.

The text entry field allows you to type a format name directly. It has an 'intelli-complete' function that selects close matches as you type.

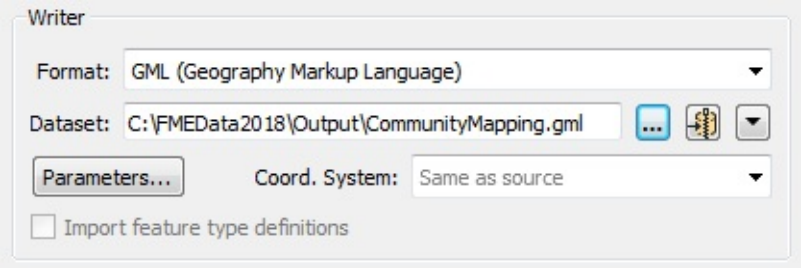
The drop-down list shows some of the most commonly used formats, so many favourite formats are instantly available:



Click 'More Formats' and a table opens showing ALL of the formats supported by FME.

The source dataset is another key requirement. Dataset selection fields are a text entry field, but with a browse button to open an explorer-like dialog for file selection.

Similarly, the Writer format and dataset are defined in this dialog:



Miss Vector says...

Here's a question you can't answer with 'a', 'b', 'c', or 'd'! In the Generate Workspace dialog, why might it be useful to set the data format before browsing for the source data?

Try browsing for a dataset before setting the format type and see if you can detect the difference.

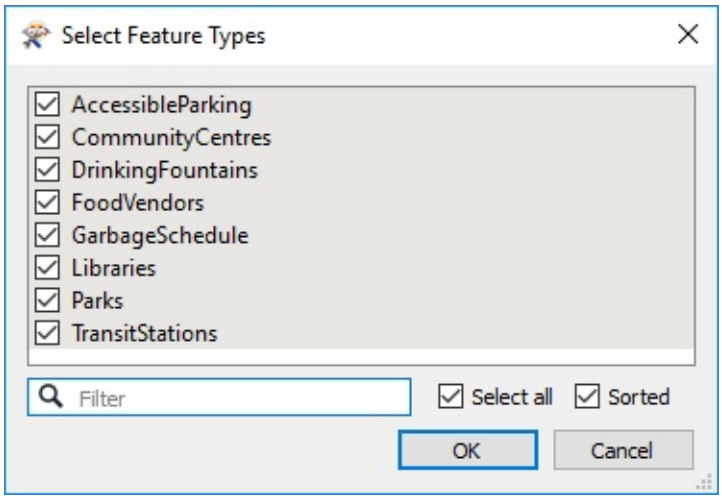
Feature Types Dialog

Clicking OK on the Generate Workspace dialog causes FME to generate the defined workspace. However, whenever a source dataset contains multiple layers, the user is first prompted to select which are to be translated.

This is achieved through the Select Feature Types dialog. In FME, ***feature types*** define the schema of the data being read. They define the dataset to be read or written and the attributes these datasets possess. A ***feature*** is an individual item within the translation.

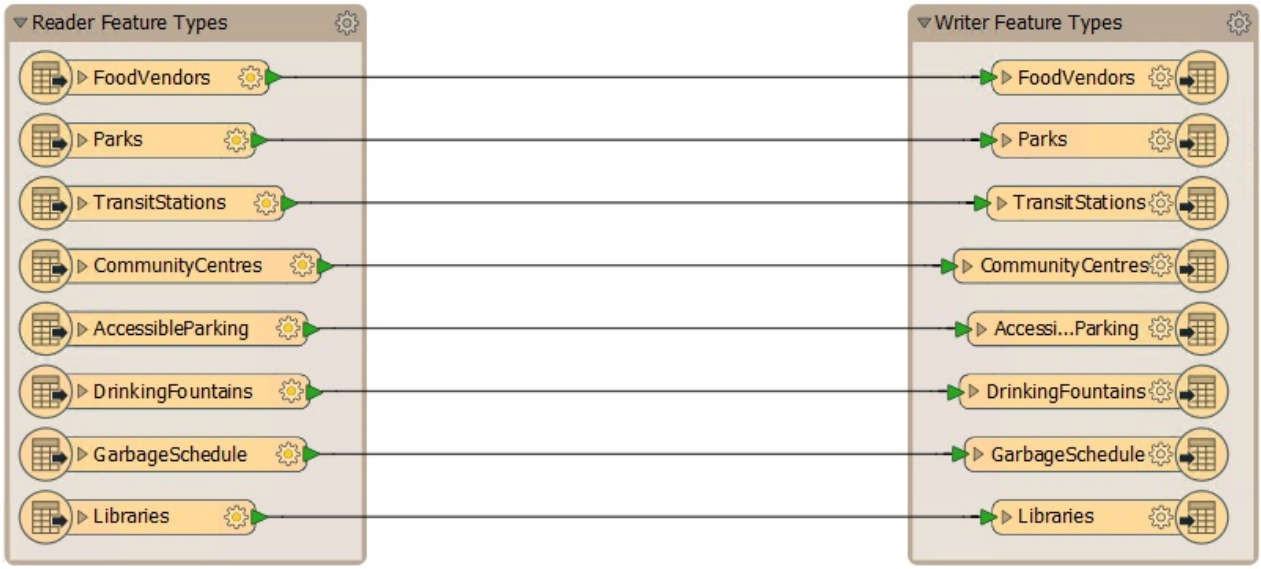
When working with spatial data, a feature type corresponds to a layer (Esri shapefile *.shp, AutoCAD drawing* .dwg), level (Microstation *.dgn), etc., depending on the format. Features are geometric features such as points, lines, or polygons. When working with spreadsheet data (e.g.* .csv, *.xlsx, a table in a database), a feature type corresponds to a table and features are rows. In all cases, the unit making up a feature type is determined by the data format.

The Select Feature Types dialog determines which layers show in the workspace. Here, for example, is a Select Feature Types dialog where the user has chosen to include all available layers in the workspace:



The New Workspace

A new workspace reads from left to right, from the source (Reader) layers, through to the destination (Writer) layers. Arrows denote the direction of data flow:



In the above screenshot, eight layers are being read from one format and written to another.

TIP

In most cases FME uses the terms 'Reader' and 'Writer' instead of 'Source' and 'Destination.' So a Reader reads datasets and a Writer writes datasets, in terms analogous to source/destination and input/output.

Saving the Workspace

Workspaces can be saved to a file so that they can be reused at a later date. The save button on the toolbar is one way to do this:

There are also menu options to do the same thing, in this case, File > Save (shortcut = Ctrl+S) or File > Save As. The default file extension is .fmw.

Sister Intuitive says...

Greetings. I am Sister Maria Intuitive of the Order of the Perpetual Translation. I'm here to demonstrate some of the intuitive, but less obvious, features of the FME Workbench interface.

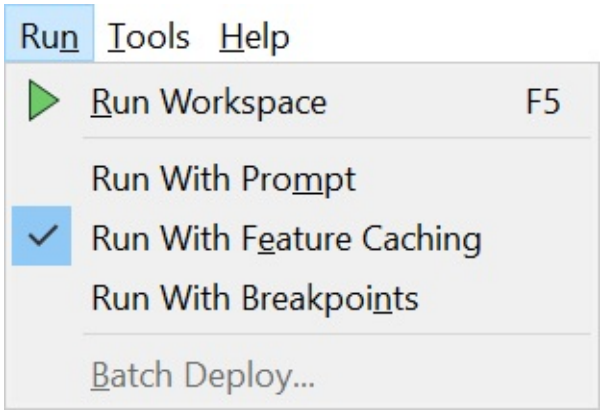
For example, select File > Open Recent on the menubar to reveal a list of previously used workspaces. This list can show up to a huge 15 entries.

Running a Workspace

The green arrow (or 'play' button) on the Workbench toolbar starts a translation:

Alternatively, look under Run on the menubar:

The same toolbar options appear on both the menubar and toolbar. Notice the shortcut option F5 can be used instead:



TIP

The action of the Run button can be modified by a series of options including the ability to prompt the user for input (Run with Prompt), the ability to cache intermediate data (Run with Feature Caching) and the ability to run in debug mode (Run with Breakpoints). These are toggle options that can be turned on and off at will.

Workspace Results

After running a workspace, related information and statistics are found in the translation log, which is displayed in the Workbench log window.

The translation log reveals whether the translation succeeded or failed, how many features were read from the source and written to the destination, and how long it took to perform the translation.

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                            Features Read Summary
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
AccessibleParking                                                           42
CommunityCentres                                                            10
DrinkingFountains                                                          113
FoodVendors                                                                 91
GarbageSchedule                                                              6
Libraries                                                                    8
Parks                                                                       69
TransitStations                                                             11
==============================================================================
Total Features Read                                                        350
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                           Features Written Summary
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
AccessibleParking                                                           42
CommunityCentres                                                            10
DrinkingFountains                                                          113
FoodVendors                                                                 91
GarbageSchedule                                                              6
Libraries                                                                    8
Parks                                                                       69
TransitStations                                                             11
==============================================================================
Total Features Written                                                     350
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Translation was SUCCESSFUL with 0 warning(s) (8 feature(s) output)
FME Session Duration: 2.1 seconds. (CPU: 1.1s user, 0.5s system)
END - ProcessID: 4084, peak process memory usage: 131024 kB
Translation was SUCCESSFUL
```

In this example, the log file reveals that 350 features were read (from an Esri Geodatabase) and written out (to a GML dataset).

The overall process was a success, with zero warnings. The elapsed time for the translation was 2.1 seconds.

Miss Vector says...

I bet you've got all of the questions correct so far! Well done. Now see if you can get these:

Which of these is NOT a way to set the format of a translation?

1. Typing the format name
2. Selecting the format from a drop-down list
3. Browsing for the format in the formats gallery
4. By selecting a dataset with a known file extension
5. None of the above (they are all valid ways to set the format)

Which key is a shortcut to run a workspace?

1. F4
2. F5
3. F5.6
4. F#

Exercise 2	Basic Workspace Creation
Data	Zoning Data (MapInfo TAB)
Overall Goal	Create a workspace to translate zoning data in MapInfo TAB format to GeoJSON (Geographic JavaScript Object Notation)
Demonstrates	Basic workspace creation with FME Workbench
Start Workspace	None
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Basics-Ex2-Complete.fmw

Congratulations! You have just landed a job as technical analyst in the GIS department of your local city. Your old schoolteacher, Miss Vector, gave you a reference, so don't let her down!

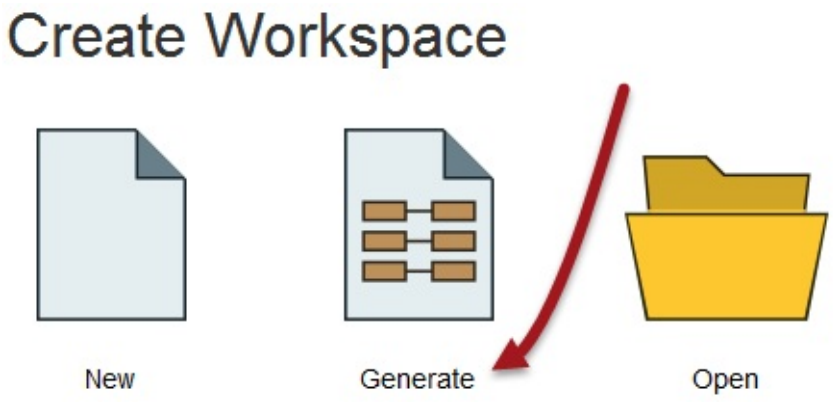
On your first day you've been asked to do a simple file format translation.

We've outlined all of the actions you need to take; though FME's interface is so intuitive you should be able to carry out the exercise without the need for these step-by-step instructions.

1) Start FME Workbench
If it isn't open already, start FME Workbench by selecting it from the Windows start menu. You'll find it under Start > FME Desktop 2018.0 > FME Workbench 2018.0.

If Workbench is already open, click on the Start tab above the main canvas.

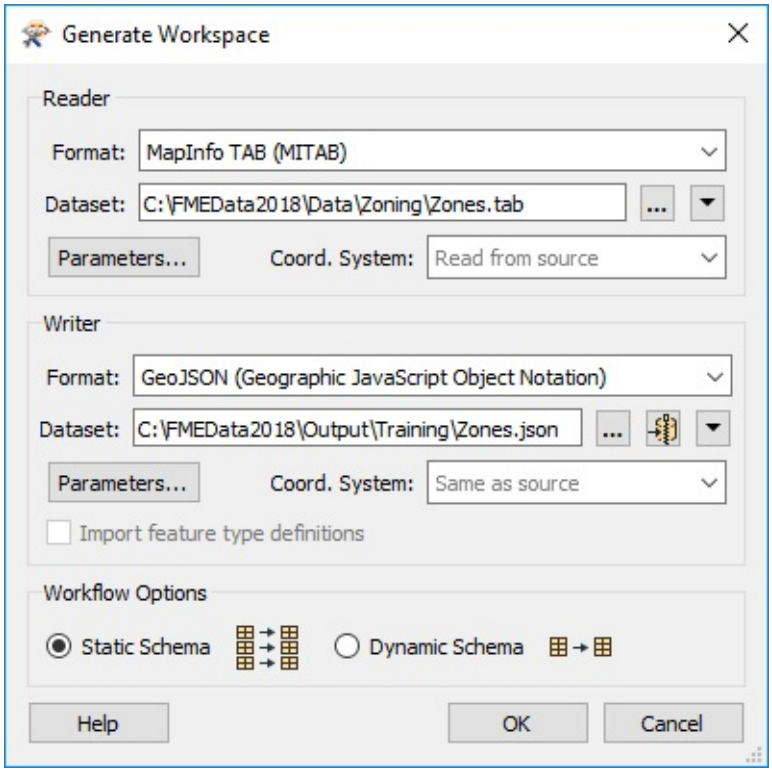
2) Select Generate Workspace
In the Create Workspace part of the Start page select the option to Generate (Workspace). Alternatively you can use the shortcut Ctrl+G.



3) Define Translation
The Generate Workspace tool opens up a dialog in which to define the translation to be carried out. Fill in the fields in this dialog as follows:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2018\Data\Zoning\Zones.tab
Writer Format	GeoJSON (Geographic JavaScript Object Notation)
Writer Dataset	C:\FMEData2018\Output\Training\Zones.json

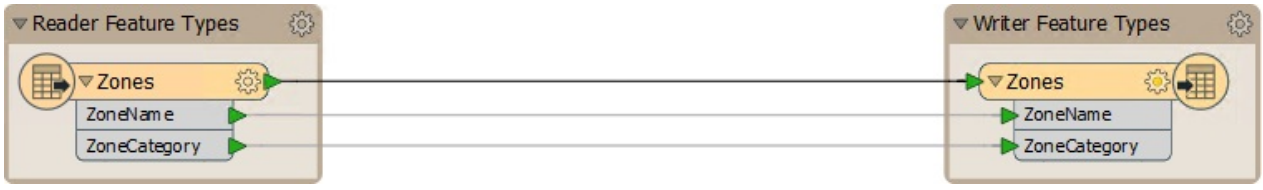
The dialog will look like this:



Remember, you can set a format by typing its name, by selecting it from the drop- down list, or by choosing “More Formats” and selecting the format from the full table of formats. For now, ignore the Workflow Options and leave the default of 'Static Schema.'

4) Generate and Examine Workspace

Click OK to close the Generate Workspace dialog. A new workspace will be generated into the FME Workbench canvas:

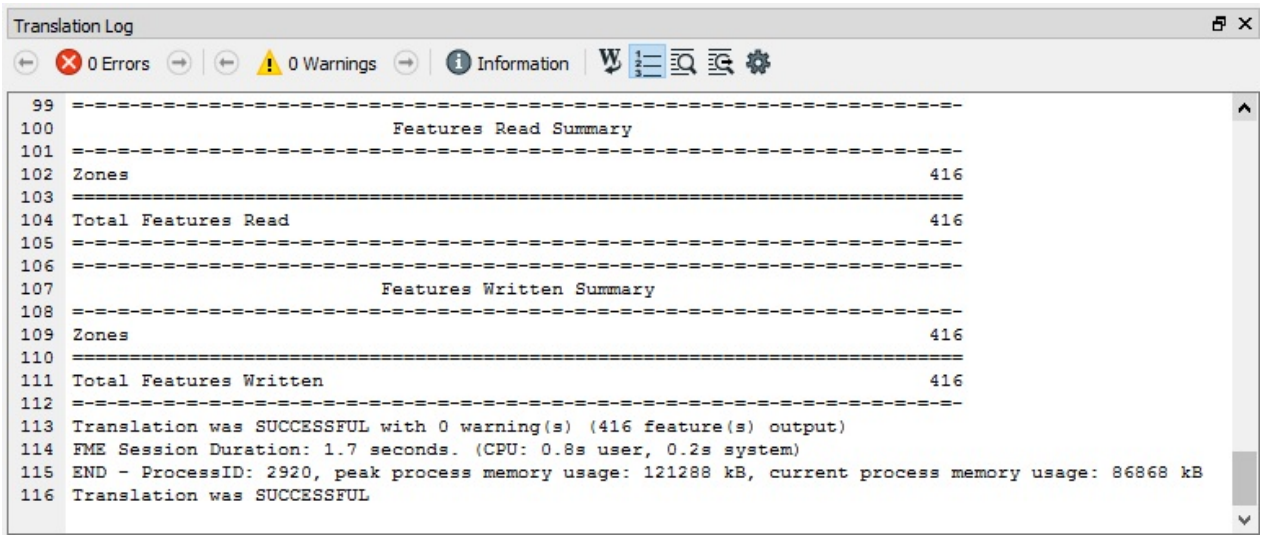


The list of attributes is exposed by clicking the arrow icon on each object.

5) Run Workspace

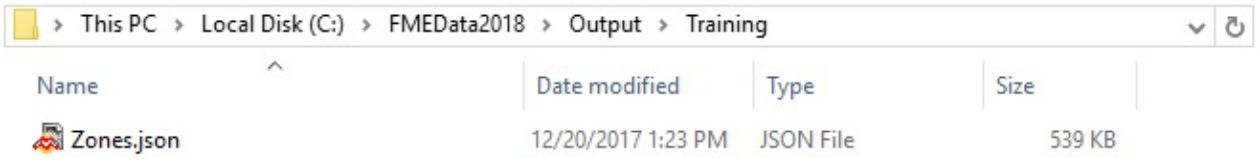
Run the workspace by clicking the run button on the toolbar, or by using Run > Run

Translation on the menubar. The workspace runs and the log file reports a successful translation:



6) Locate Output

Locate the destination data in Windows Explorer to prove that it's been written as expected (don't forget the Open Containing Folder button from Exercise 1). In the next section we'll cover how to inspect the dataset visually to ensure that it is correct.



7) Save Workspace

Save the workspace. We'll be using it in a later exercise. Remember there is a toolbar save button and on the menu there is File > Save As.

TIP

When a translation is run immediately without adjustment it's known as a "Quick Translation". Because FME is a 'semantic' translator, with an enhanced data model, the output from a quick translation is as close to the source data in structure and meaning as possible, given the capabilities of the destination format.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Create an FME workspace*
- *Run an FME workspace*

Introduction to Data Inspection

To ensure that you're dealing with the right information you need a clear view of your data at every stage of the transformation process.

Data Inspection meets this need: it is the act of viewing data for verification and debugging purposes, before, during, or after a translation.

What Can Be Inspected?

A number of different aspects of data may be inspected, including the following:

- **Format**: Is the data in the expected format?
- **Schema**: Is the data subdivided into the correct layers, categories or classes?
- **Geometry**: Is the geometry in the correct spatial location? Are the geometry types correct?
- **Symbology**: Is the color, size, and style of each feature correct?
- **Attributes**: Are all the required attributes present? Are all integrity rules being followed?
- **Quantity**: Does the data contain the correct number of features?
- **Output**: Has the translation process restructured the data as expected?

Chef Bimm says...

Hi. I'm Chef Bimm and I'm here to help you cook up some tasty data translations.

I have a great recipe for loading CAD files into a Building Information Model. Inspecting the ingredients... I mean data... before I use them lets me detect problems before they affect the translation.

Features in the wrong source layer could need the whole process to be repeated. Data Inspection saves me that hassle. Now I know that if the "sauce" is fine, the final result will be too!

Introduction to the FME Data Inspector

The best place to start inspecting data in FME is in a complementary application called the FME Data Inspector.

What is the FME Data Inspector?

The FME Data Inspector is a utility that allows viewing of data in any of the FME supported formats. It is used primarily to preview data before translation or to verify it after translation.

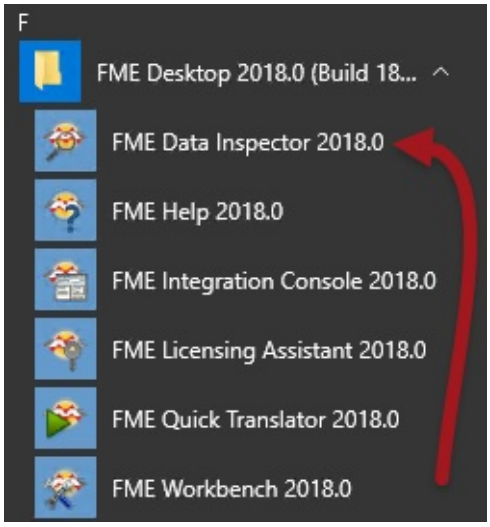
The FME Data Inspector is closely tied to FME Workbench so that Workbench can send data directly to the Inspector. It's also embedded inside FME Workbench too, to help set up and debug workspaces by inspecting data *during* the translation.

What the FME Data Inspector Is Not!

The FME Data Inspector isn’t designed to be a form of GIS or mapping application. It has no analysis functionality, and the tools for symbology modification or printing are intended for data validation rather than producing map output.

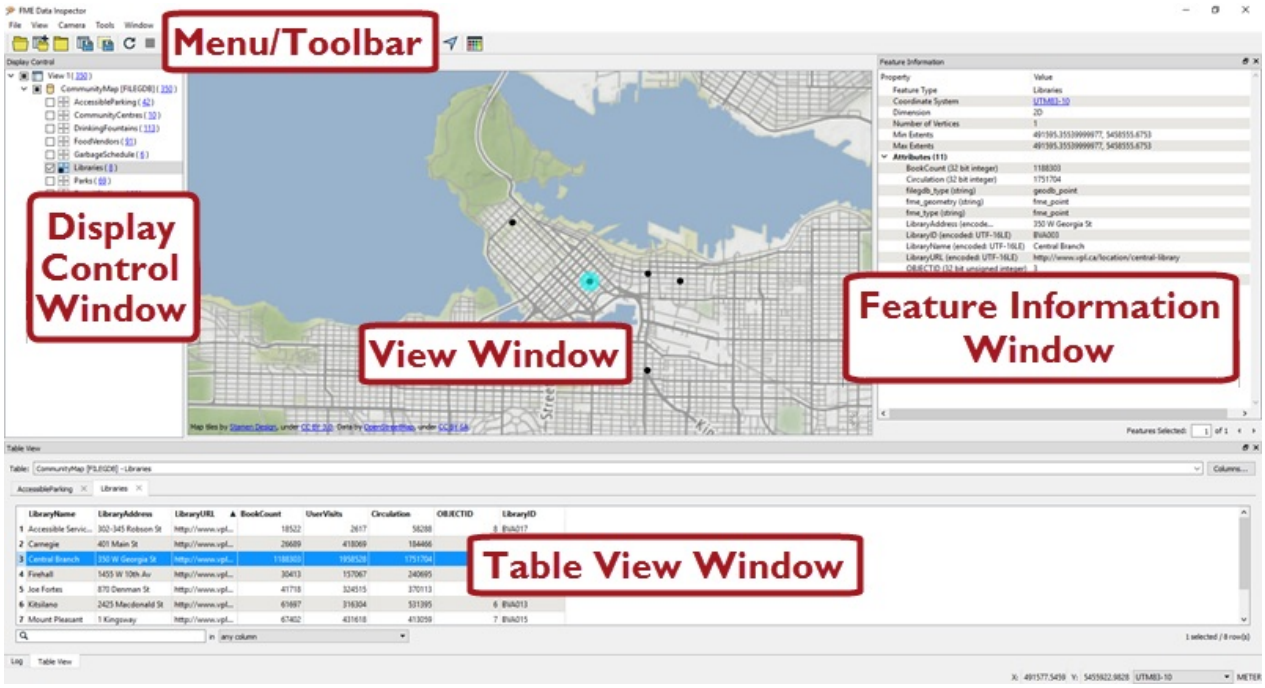
Starting the FME Data Inspector

To start the Data Inspector locate it in the Windows start menu:



Major Components of the FME Data Inspector

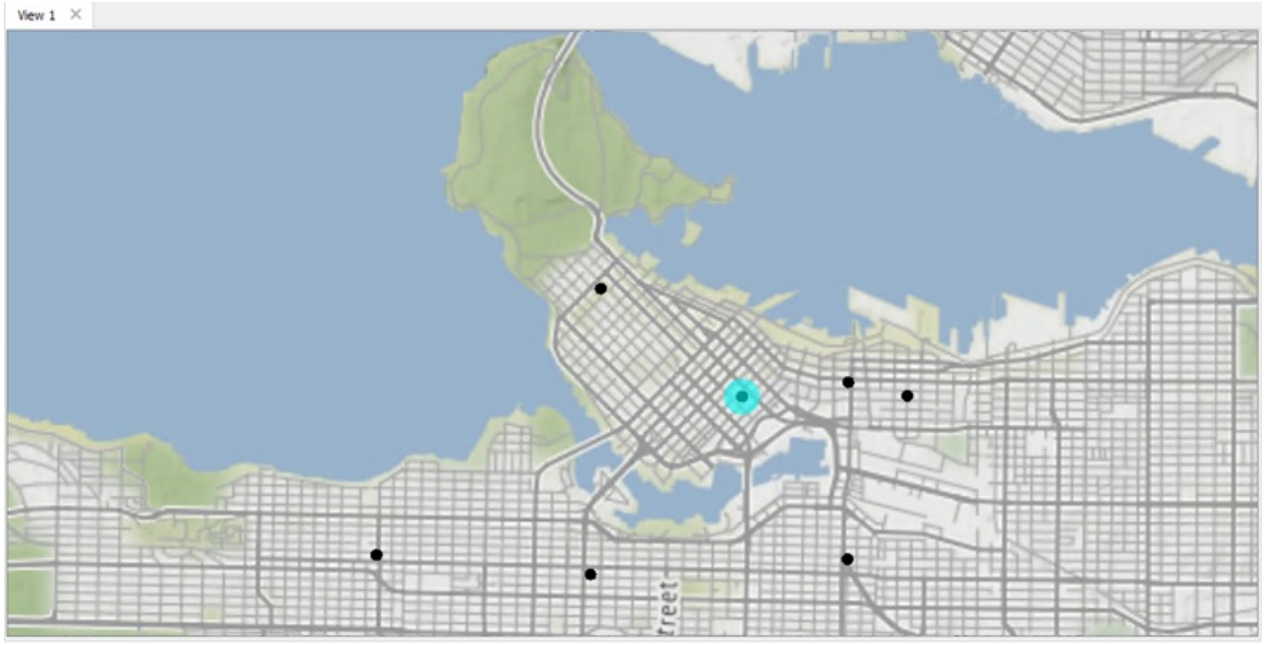
When the FME Data Inspector is started, and a dataset is opened, it looks something like this:



Map tiles by Stamen Design, under CC-BY-3.0. Data by OpenStreetMap, under CC-BY-SA.

View Window

The View window is the spatial display area of the FME Data Inspector. Multiple views of different datasets may be opened at any one time.



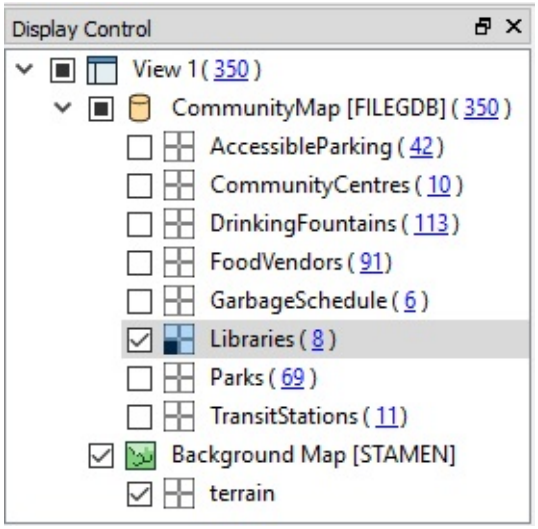
Map tiles by Stamen Design, under CC-BY-3.0. Data by OpenStreetMap, under CC-BY-SA.

Menu bar and Toolbar

The menu bar and toolbar contain a number of tools. Some are for navigating around the View window, some control administrative tasks such as opening or saving a dataset, and others are for special functionality such as selective filtering of data or the creation of dynamic attributes.

Display Control Window

The Display Control window shows a list of the open datasets and their feature types. Tools here let users turn these on or off in the display, alter their symbology, and adjust the display order.



Feature Information Window

When users query a feature in the View window, information about that feature is shown in the Information window. This information includes the feature's feature type, attributes (both user and format attributes), coordinate system and details about its geometry.

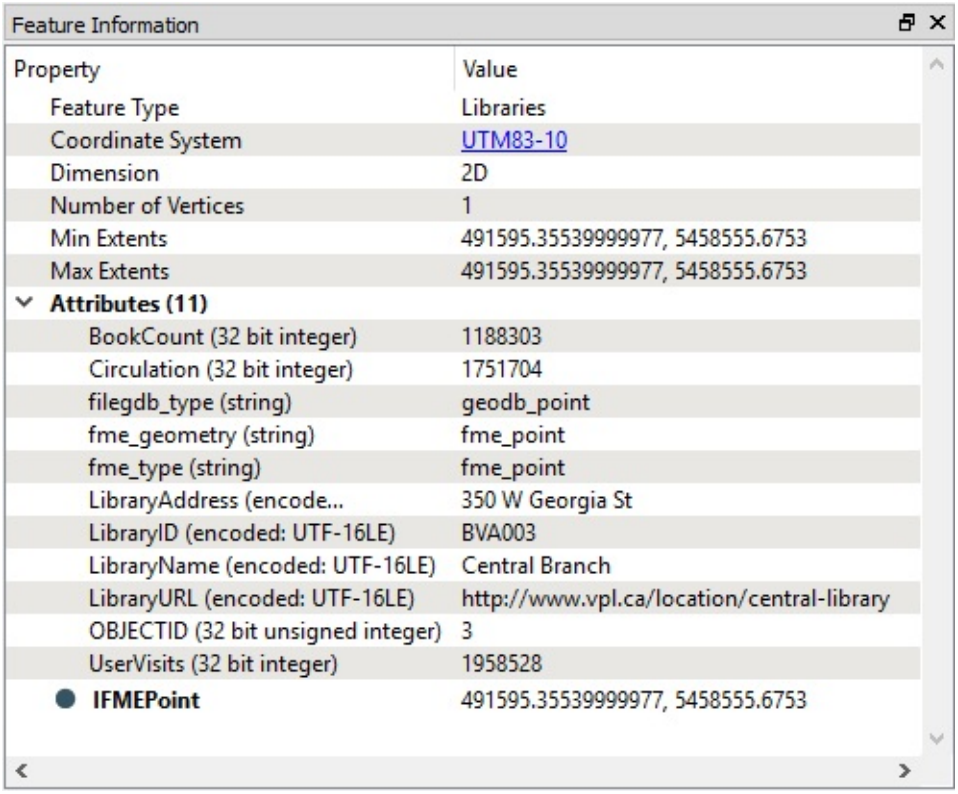
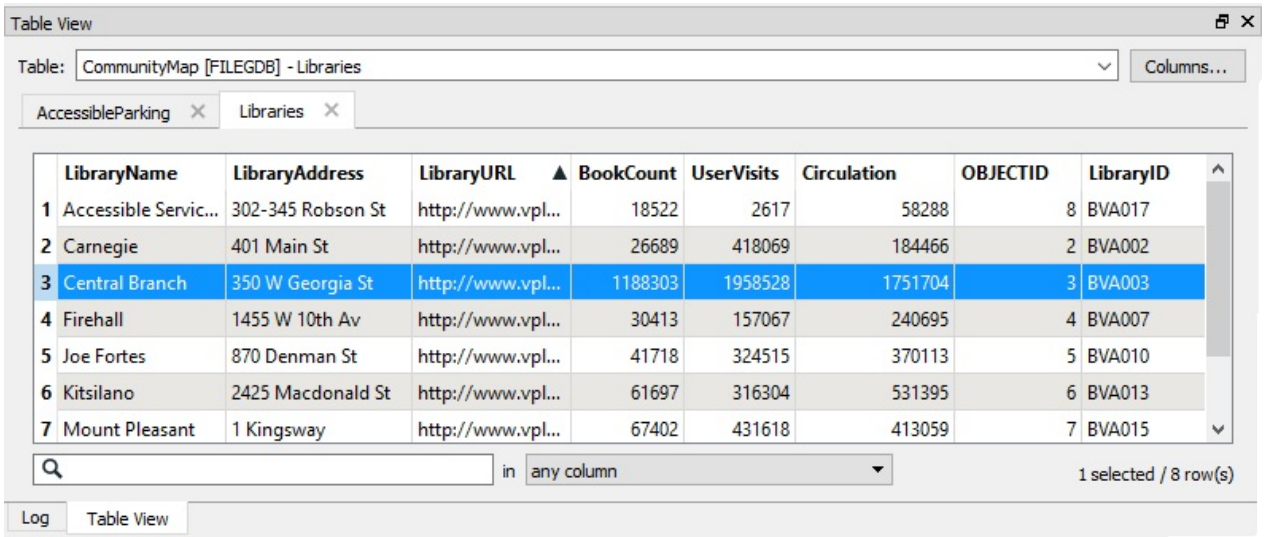


Table View Window

The Table View window is a spreadsheet-like view of a dataset and includes all of the features and all of the attributes, with a separate tab for each feature type (layer).



Miss Vector says...

FME is so easy to use, it's hard to think up some difficult questions! But I'll try.

*When you are inspecting **schema**, what are you trying to verify?*

1. The color and linestyle of features
2. The number of features
3. The feature types (layers, classes, tables) and their attributes
4. Where the nearest coffee shop is

Using the FME Data Inspector

With the FME Data Inspector, it's easy to open and view any number of datasets and to query features within them.

Viewing Data

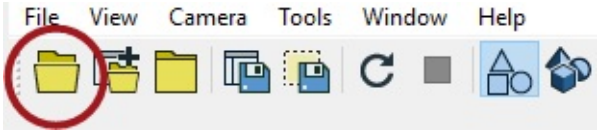
The FME Data Inspector provides two methods for viewing data: opening or adding.

Opening a dataset opens a new view window for it to be displayed in. ***Adding*** a dataset displays the data in the existing view window; this way multiple datasets can be viewed simultaneously.

Opening a Dataset

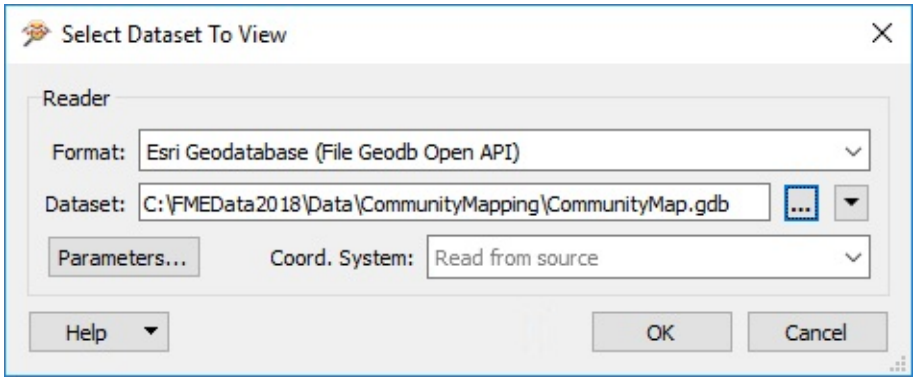
Datasets can be opened in the FME Data Inspector in several ways.

- Select File > Open Dataset from the menu bar
- Select the toolbar button Open Dataset.
- Drag and drop a file onto any window (except the View window)
- Open from within Workbench



Opening data from within FME Workbench is achieved by simply right-clicking on a canvas feature type (either source or destination) and choosing the option 'Inspect.'

All of these methods cause a dialog to open in the FME Data Inspector in which to define the dataset to view.



Adding a Dataset

Opening a dataset causes a new View tab to be created and the data displayed. To open a dataset within an existing view tab requires the use of tools to add a dataset.

- Selecting File > Add Dataset from the menu bar
- Selecting the toolbar button Add Dataset
- Dragging and Dropping a file onto the view window

Windowing Tools

Once data has been opened in the FME Data Inspector, there are many tools available for altering the view.

- Pan
- Zoom In
- Zoom Out
- Zoom to a selected feature
- Zoom to the full extent of the data

TIP

Press the Shift key on the keyboard and it will activate the zoom-in tool in the Inspector. Press the Ctrl key, which will activate the zoom-out tool. Release the key to revert to the previous tool.

This functionality allows users to quickly move between query and navigation modes at the press of a key, so there's no need to click between query and navigation tools on the menubar or toolbar.

Inspecting Data

The FME Data Inspector includes several querying tools, but two are particularly important:

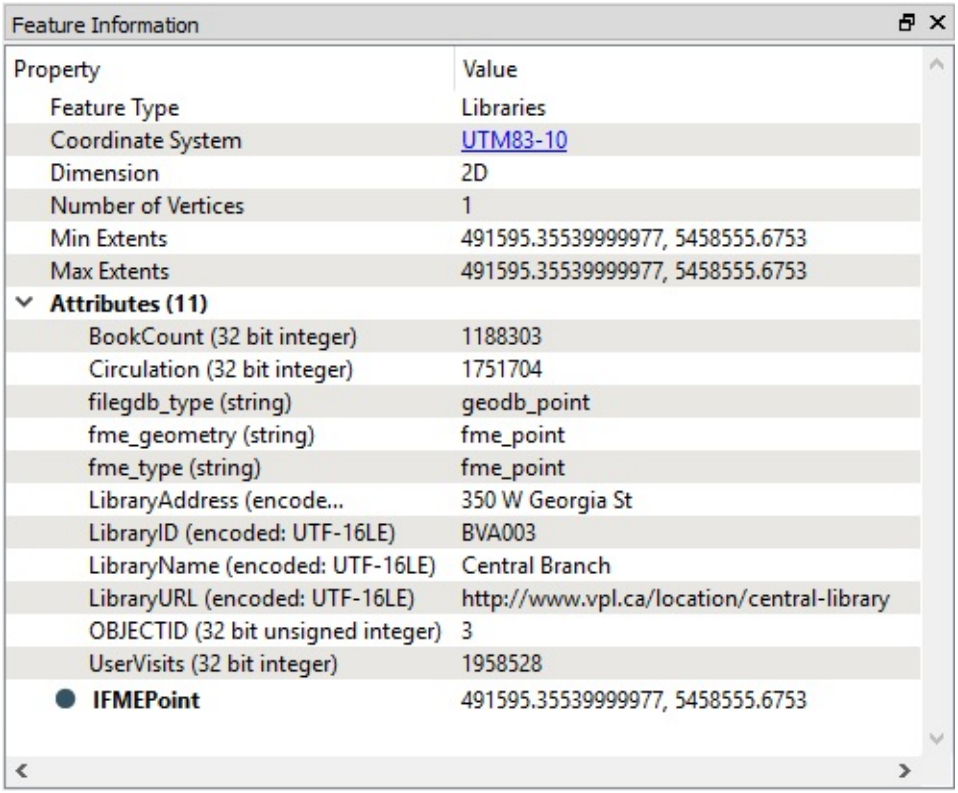
- Query individual feature(s)
- Measure a distance within a View window

The query tool button is like a toggle. By default it is active when you start the FME Data Inspector; if you click it again - or select a windowing tool - you turn the query tool off.

The results of a feature query are shown in the Feature Information window.

Feature Information Window

The upper part of this window reports on general information about the feature; which feature type (layer/table) it belongs to, which coordinate system it is in, whether it is two- or three-dimensional, and how many vertices it possesses.

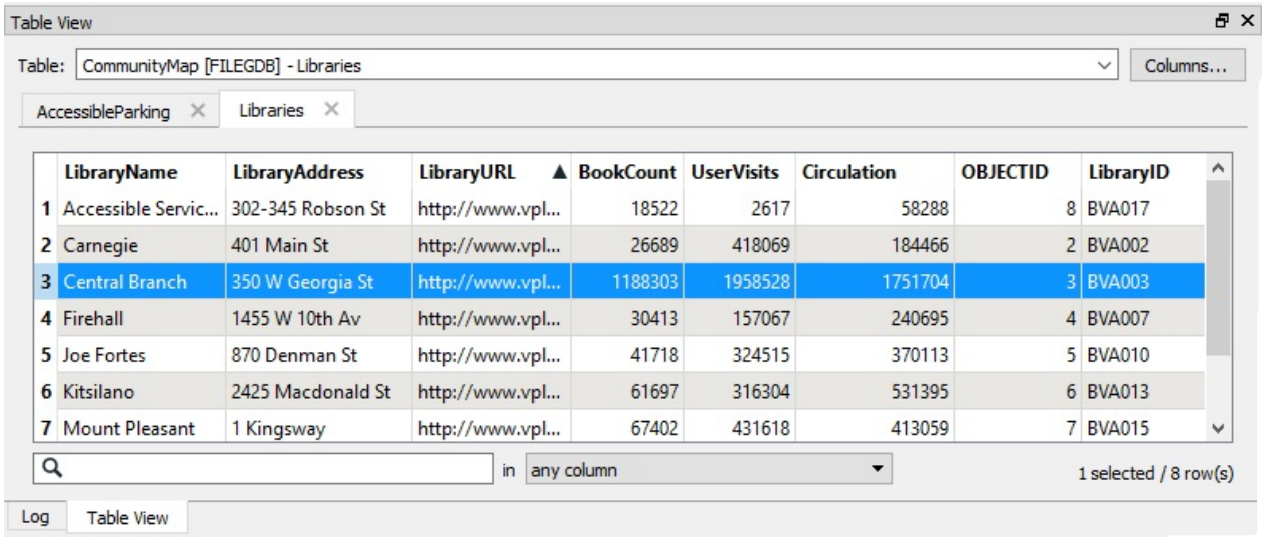


The middle part reports the attributes associated with the feature, including user attributes and format attributes (for example *fme_type*).

The lower part reports the geometry of the feature. It includes the geometry type and a list of the coordinates that go to make up the feature.

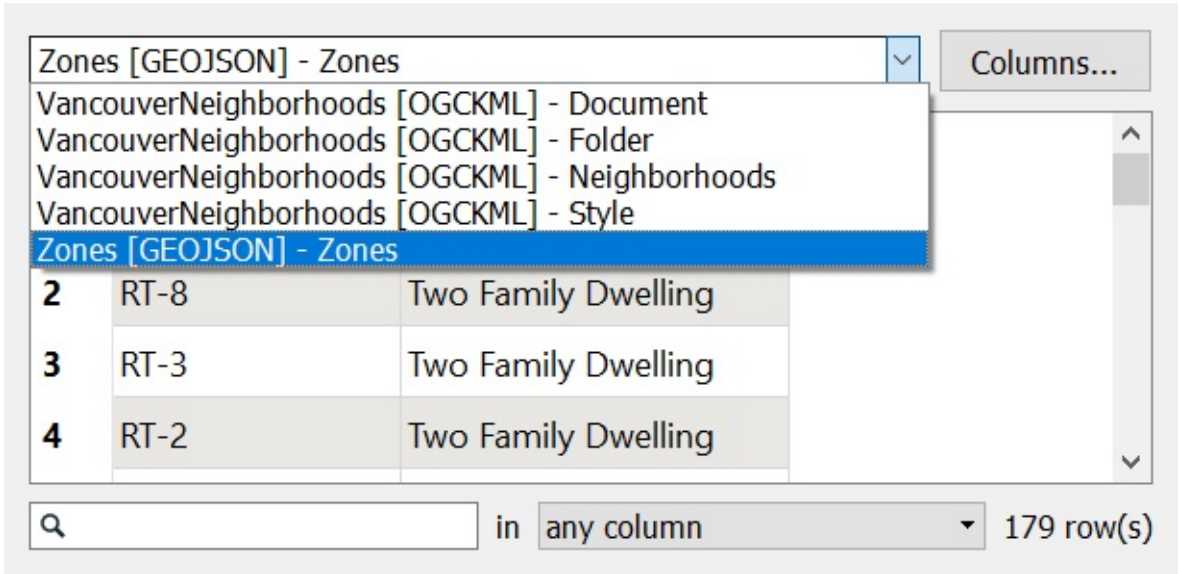
Table View Window

Also available is a window called the Table View.



The table view is a way to inspect data in a tabular, spreadsheet-like, layout. Although it does not have the same depth of information as shown by the Information Window, the Table View is particularly useful for inspecting the attribute values of multiple features simultaneously.

You can switch back and forth between feature types in the Table View by clicking the dropdown menu at the top of Table View.



TIP

To improve performance, tables are not all displayed automatically, only when selected from the drop-down list, or when queried in the current view window.

Miss Vector says...

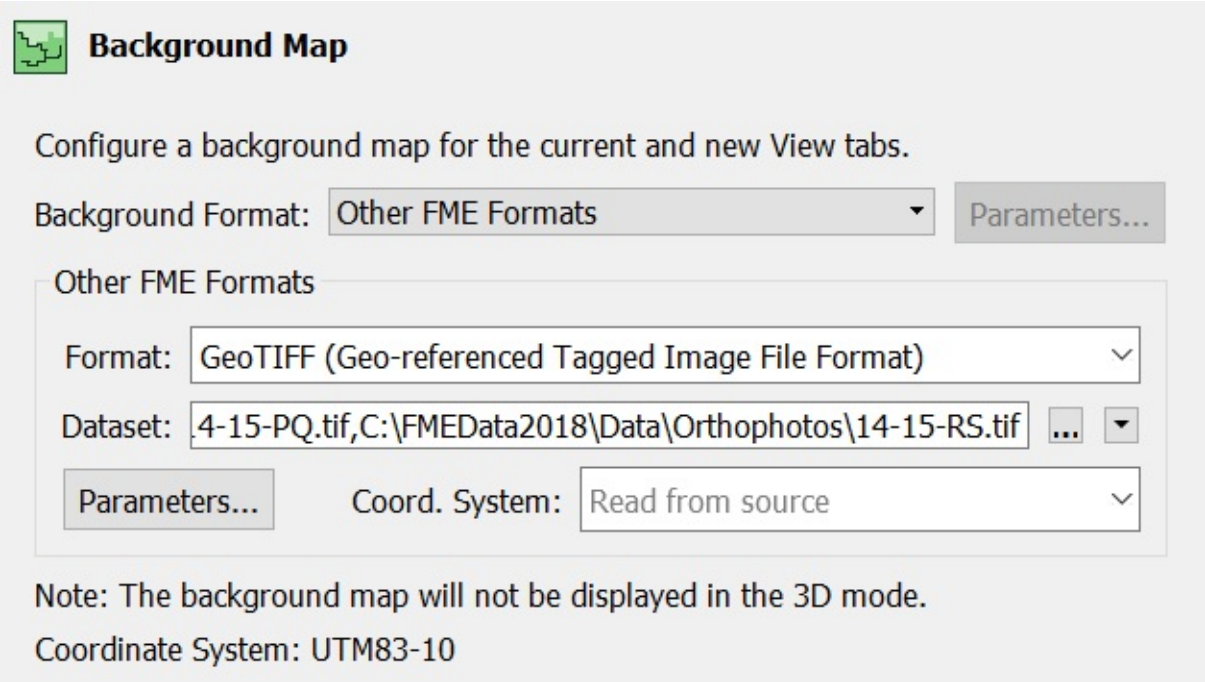
Start the FME Data Inspector and open a dataset. In the Table View window right-click on records and column-headers to view the context menus. Which of the following is NOT an available menu option(s):

1. Sort (Alphabetical or Numeric)
2. Inspect Value
3. Cut/Copy/Paste
4. Save Selected Data As

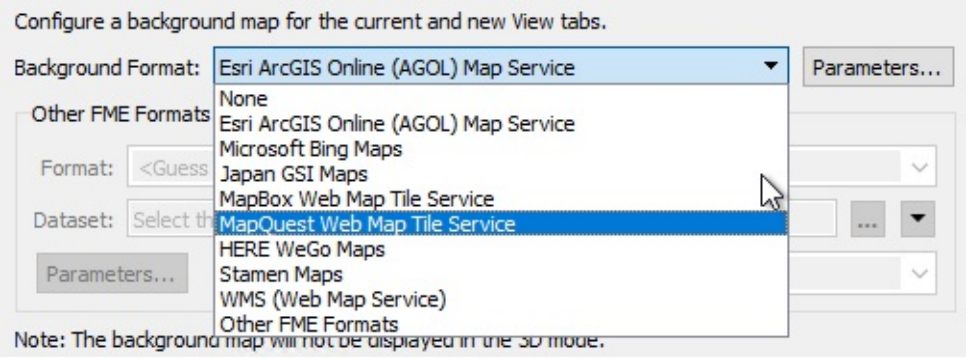
Background Maps

The ability to view maps (or other imagery) as a backdrop to your spatial data is activated by a tool under Tools > FME Options on the menubar.

The background map dialog lets the user select an existing dataset (of any FME-supported format) to use as a backdrop, like so:

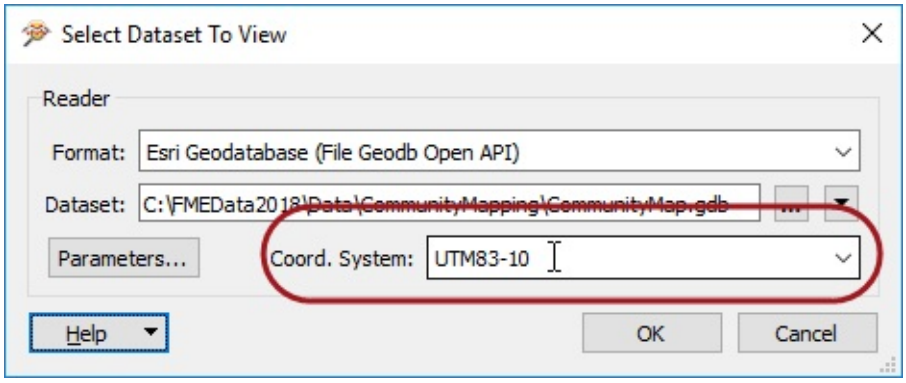


It's also possible to use web services that supply mapping on demand. Some of these - such as ArcGIS Online - require an existing account:



Coordinate Systems

Source data must be referenced with a valid coordinate system to view it with a background map. If the coordinate system is not recorded in the dataset itself, you may enter it into a field when opening the dataset:



FME can display the source data against a background map, even when there are several source datasets of differing coordinate systems. FME does this by reprojecting the data to the coordinate system used by the background map. Therefore it's recommended that you turn off the background maps when you want to inspect the data in its original form.

Police Chief Webb-Mapp says...

You can adjust the symbology and display order of the background map in the Display Control window, just as you can for any normal dataset.

Exercise 3	Basic Data Inspection
Data	Zoning Data (GeoJSON) Neighborhoods (Google KML)
Overall Goal	Inspect the output from a previous translation
Demonstrates	Basic data inspection with the FME Data Inspector
Start Workspace	None
End Workspace	None

In the previous exercise, you were asked to convert some data between formats. Before you send the converted data out, you should inspect it to make sure it is correct. Let's see how the FME Data Inspector interface works by inspecting the output from that quick translation.

1) Start FME Data Inspector

Start the FME Data Inspector by selecting it from the Windows start menu. You'll find it under Start > FME Desktop 2018.0 > FME Data Inspector 2018.0.

2) Open Dataset

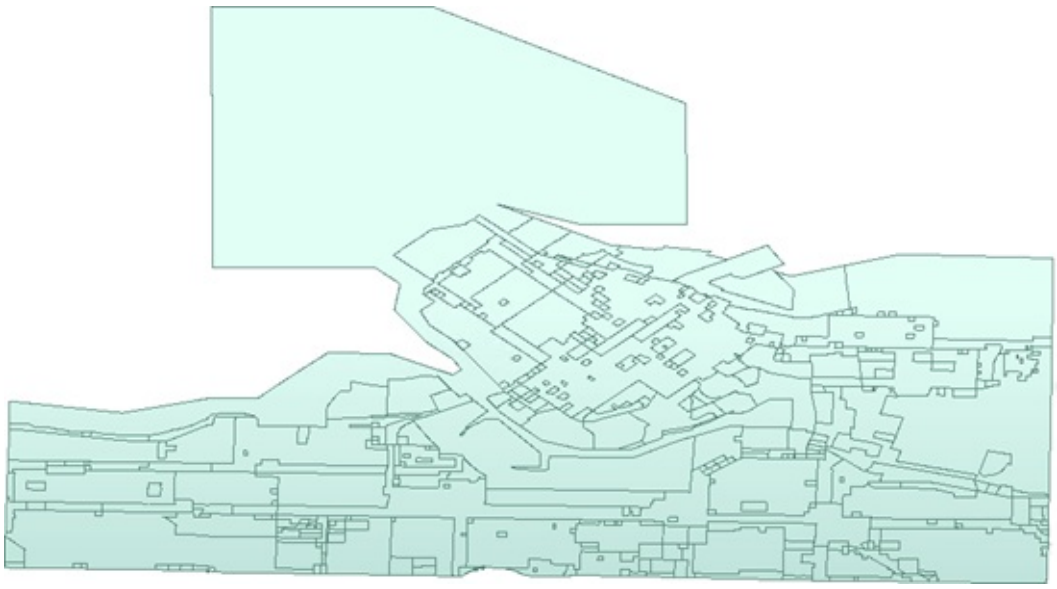
The FME Data Inspector will start up and begin with an empty view display.

To open a dataset, select File > Open Dataset from the menubar. When prompted, fill in the fields in the Select Dataset dialog as follows:

Reader Format	GeoJSON (Geographic JavaScript Object Notation)
Reader Dataset	C:\FMEData2018\Output\Training\Zones.json

NB: *If you can't find the dataset - maybe you didn't complete the first exercise, or wrote the data to a different location - then you can open the original zoning dataset as described in Exercise 2.*

The GeoJSON dataset looks like this:



3) Browse Data

Use the windowing tools on the toolbar to browse through the dataset, inspecting it closely. Use the Query tool to query individual features and inspect the information in the Feature Information Window.

Try right-clicking in the different Data Inspector windows, to discover functionality that exists on context menus.

4) Add Dataset

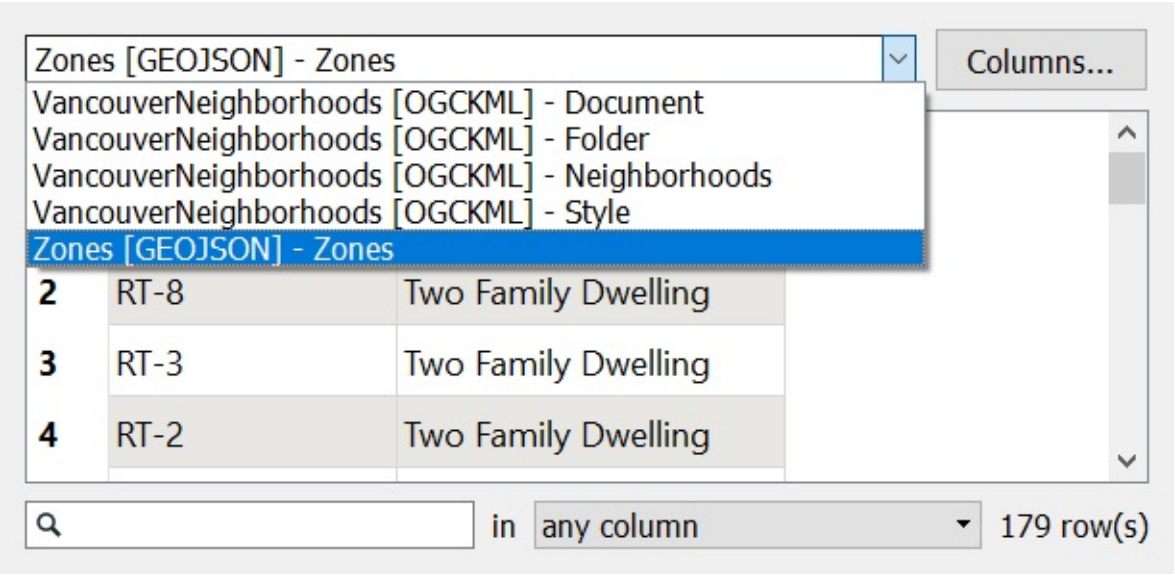
Let's add a second dataset to the display to compare to our zoning data. This dataset will be a KML file of neighborhood boundaries. Then we'll be able to see which neighborhood each zone overlaps.

To add a dataset, select File > Add Dataset from the menubar. When prompted, fill in the fields in the Select Dataset dialog as follows:

Reader Format	Google KML
Reader Dataset	C:\FMEData2018\Data\Boundaries\VancouverNeighborhoods.kml

The display now looks like this:

Use the Table View to practice inspecting the tabular data for each feature type. Click on the dropdown arrow at the top of Table View and switch back and forth between the Zones.json and Neighborhoods.kml tables:

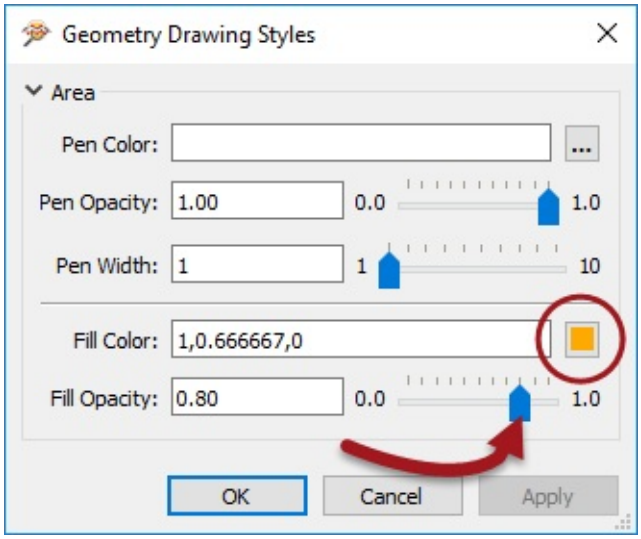


5) Set Neighborhoods Symbology

The Display Control window shows a number of different layers in the VancouverNeighborhoods dataset. In reality, most of these layers are extra information; the layer we are interested in is called Neighborhoods.

Click the symbology icon for the Neighborhoods data in the Display Control window:

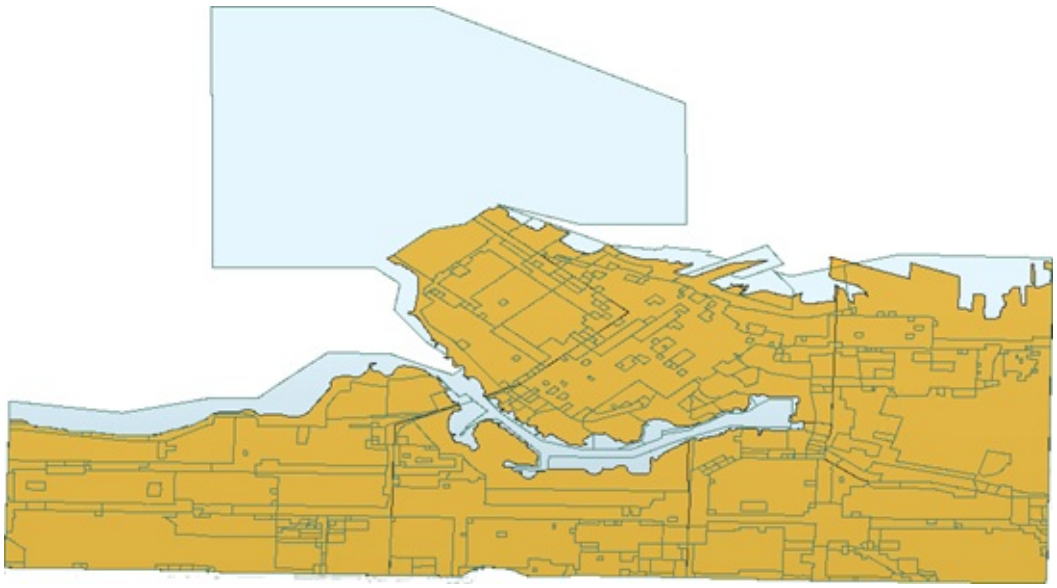
Set the color to be a neutral shade (like orange) and increase the opacity value to 0.8:



6) Set Draw Order

The previous change makes it clear that the zone features are below the neighborhoods in the drawing order. To solve this problem drag the Zones dataset above the VancouverNeighborhoods dataset in the Display Control Window.

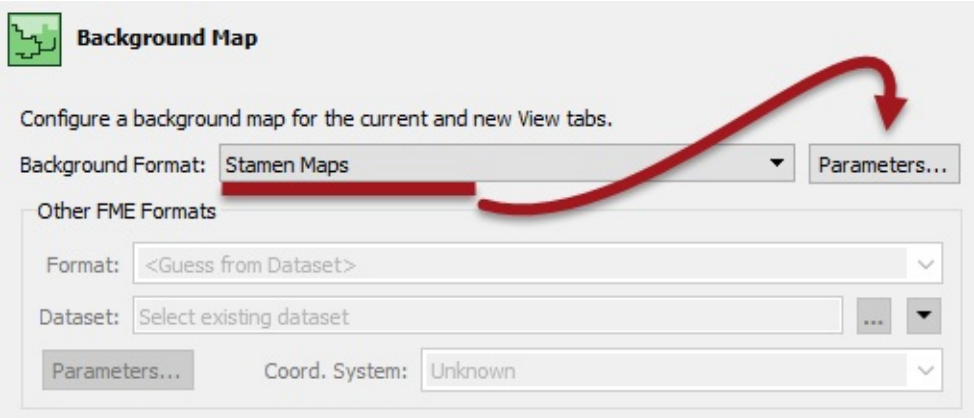
At the same time set a color for the zones data fill color and reduce the opacity value to 0.1. The main view will now look like this:



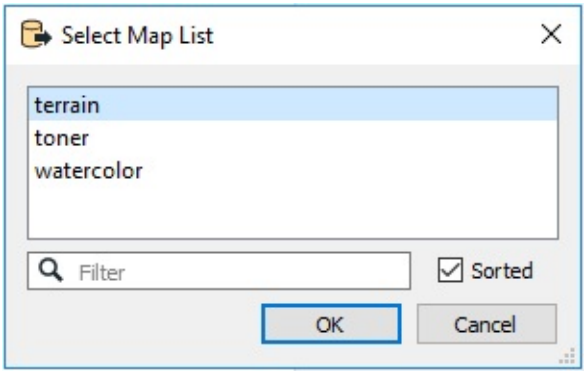
7) Set Background

When inspecting data it will help to have a background map to provide a sense of location. The FME Data Inspector is capable of displaying a backdrop from several different mapping services.

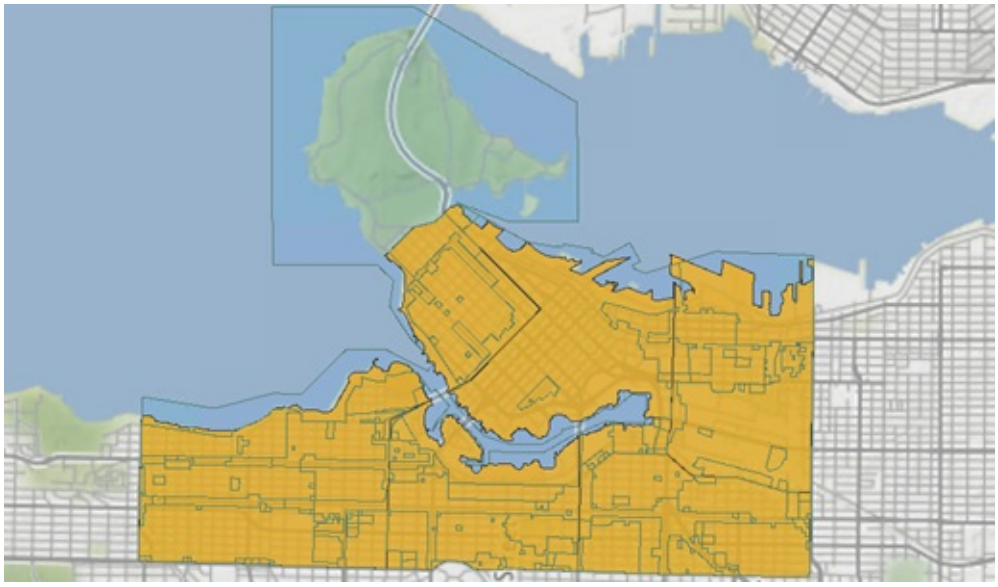
Select Tools > FME Options from the menubar. In the Background Map section, select a background map format. If you have an account with a specific provider, please feel free to use that. Otherwise, select Stamen Maps:



Click the Parameters button. A map constraint (type) dialog will open. Click the browse button and select "terrain":



Click OK and OK again to close these dialogs. A background map is added to the display. Notice that the data is reprojected to match the coordinate system of the chosen background. For example, the Stamen Maps background causes the data to reproject to Spherical Mercator, with a clear change of shape:



CONGRATULATIONS

By completing this exercise you have learned how to:

- *Open datasets in a new view in the FME Data Inspector*
- *Use the windowing and inspection tools in the FME Data Inspector*
- *Use FME Workbench functionality to open a dataset in the FME Data Inspector*
- *Set symbology for features in the FME Data Inspector*
- *Set a background map for the FME Data Inspector*

Structural Transformation

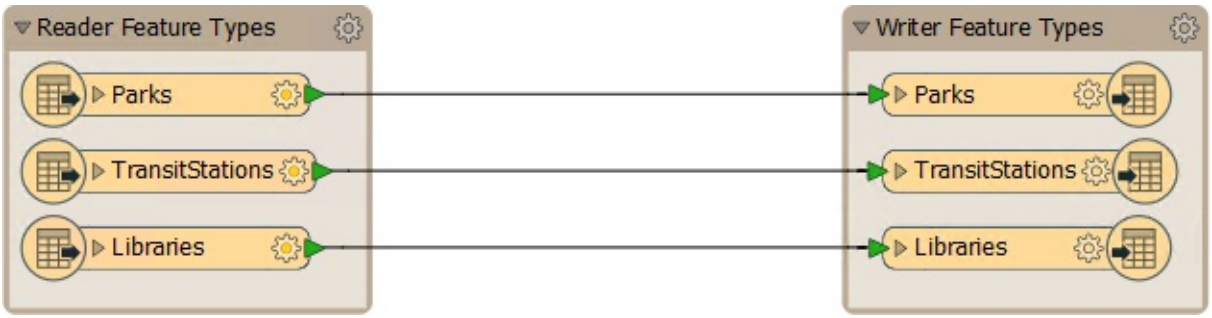
Transforming a dataset's structure requires using FME to manipulate *schemas*. FME uses the term "schema," but you may know this as *data model*.

Schema Concepts

A ***schema*** defines the structure of a dataset. Each dataset has its unique schema; it includes layers, attributes, and other rules that define or restrict its content.

Schema Representation

When a new workspace is created, FME scans the source datasets. It creates a ***reader*** whose layers are illustrated on the left side of the workspace canvas and a ***writer*** whose layers are illustrated on the right side of the workspace canvas:

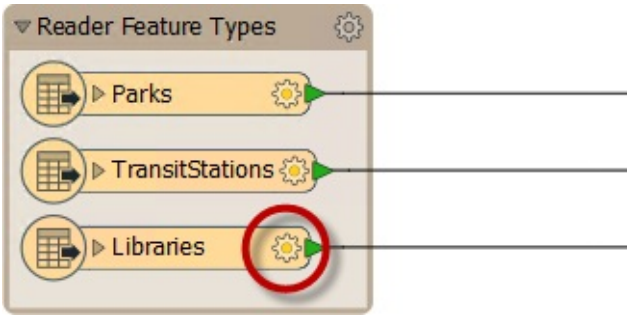


TIP

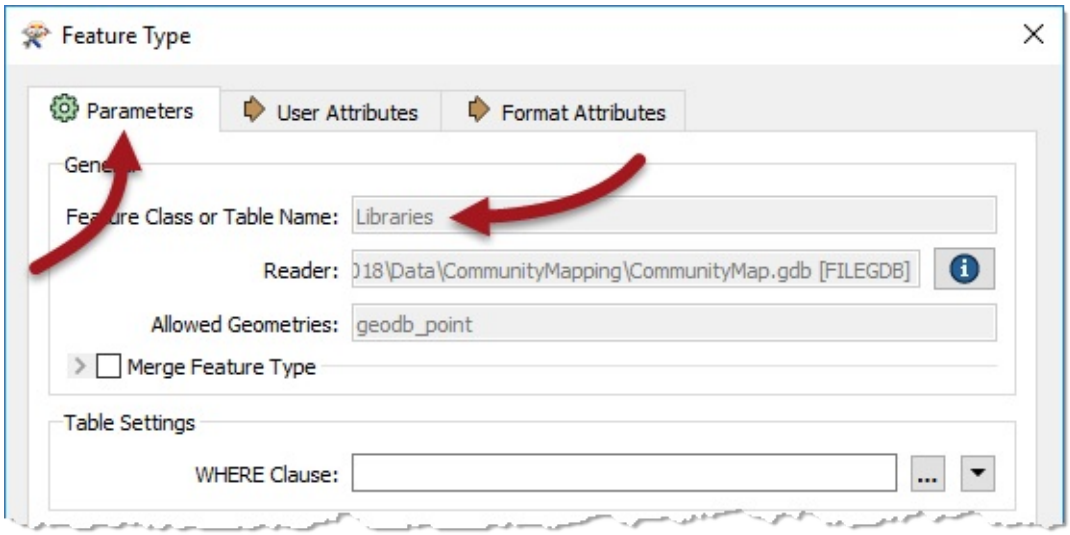
Each object in this illustration represents a subdivision in the source dataset. In FME terminology these objects are called ***feature types****. Multiple feature types means there are multiple layers in the source dataset.*

Reader Schema

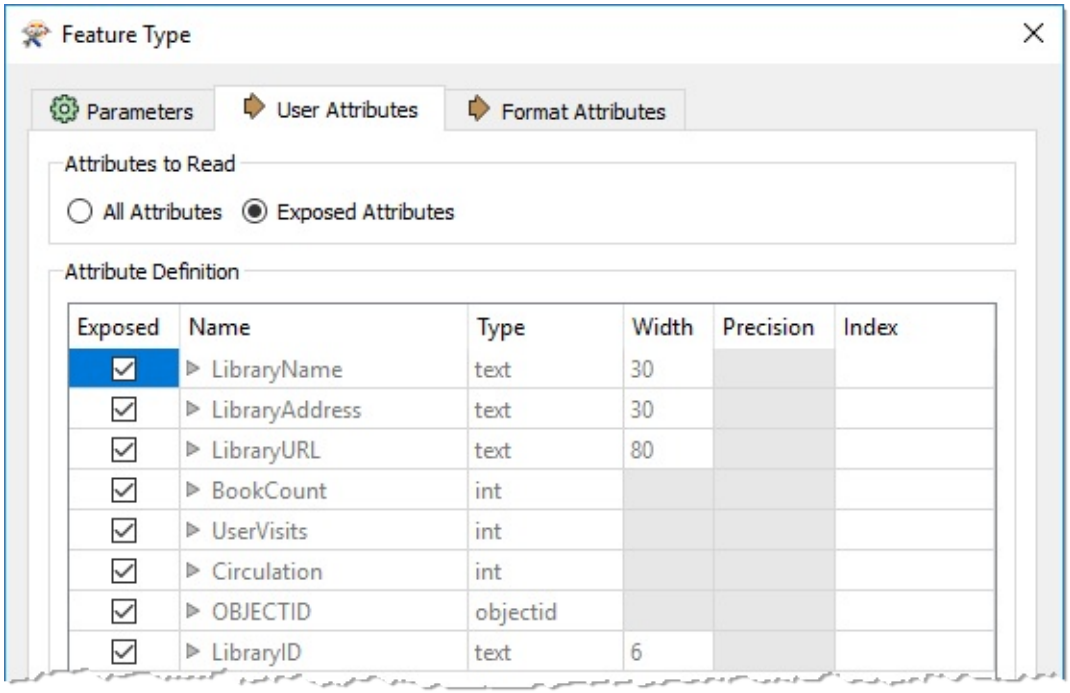
For the reader, more information about the schema is revealed by clicking the cog-wheel icon on each feature type object:



This Feature Type dialog has a number of tabs. Under the Parameters tab is a set of general parameters, such as the name of the feature type (in this case Libraries) the allowed geometry types, and the name of the parent dataset:



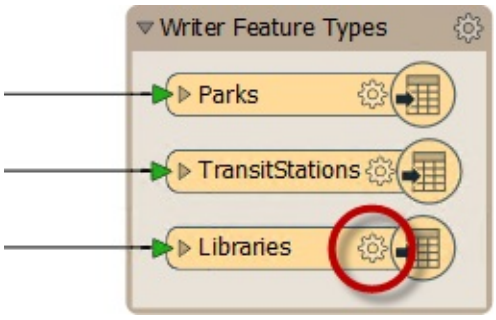
The User Attributes tab shows a list of attributes. Each attribute is defined by its name, data type, width, and number of decimal places:



Each layer has a different name and can also have a completely different set of attributes. All of this information goes to make up the reader schema. It is literally ***"what we have"***.

Writer Schema

As with the reader, each writer has a set of detailed schema information accessed by opening the dialog for a feature type:



By default, the writer schema (***"what we want"***) is a mirror image of the source, so the output from the translation will be a duplicate of the input. This feature allows users to translate from format to format without further edits (*Quick Translation*).

If *"what we want"* is different to the default schema definition, we have to change it using a technique called ***Schema Editing***.

TIP

FME supports 400+ formats and there are almost as many terms for the way data is subdivided. The most common terms are layer, table, class, category, level, or object.

*Although the general FME term for these subdivisions is **feature type**, all dialogs in FME Workbench use format-specific terminology where the correct term is applicable.*

Schema Editing

Schema Editing is the process of altering the writer schema to customize the structure of the output data. One good example is renaming an attribute field.

After editing, the source schema still represents *"what we have"*, but the destination schema now truly does represent *"what we want"*.

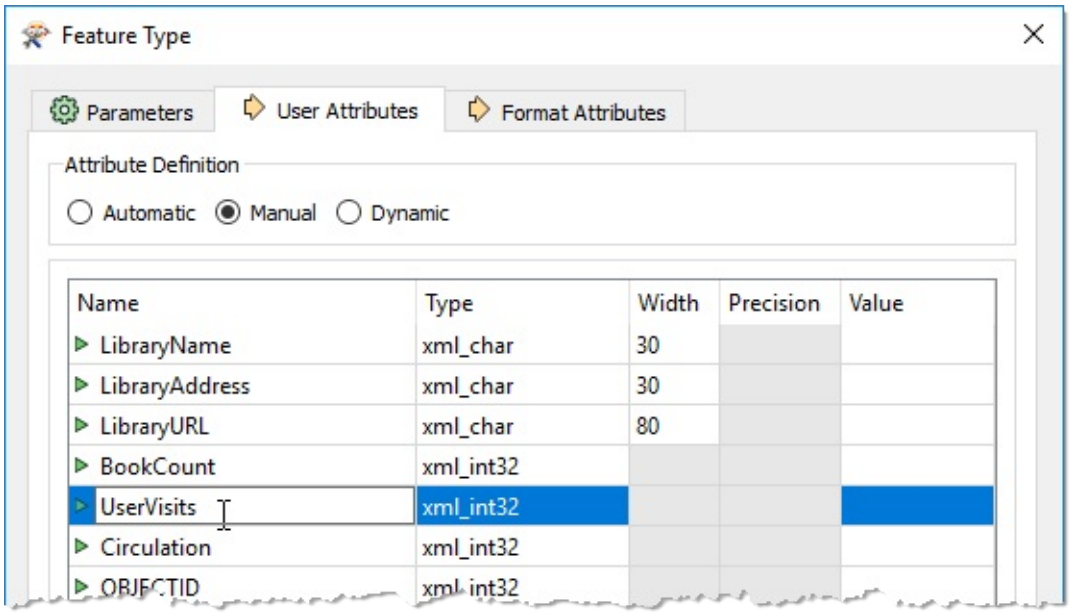
Editable Components

There are a number of edits that can be performed, including, but not limited to the following:

Attribute Renaming

Attributes on the destination schema can be renamed, such as renaming POSTALCODE to PSTLCODE.

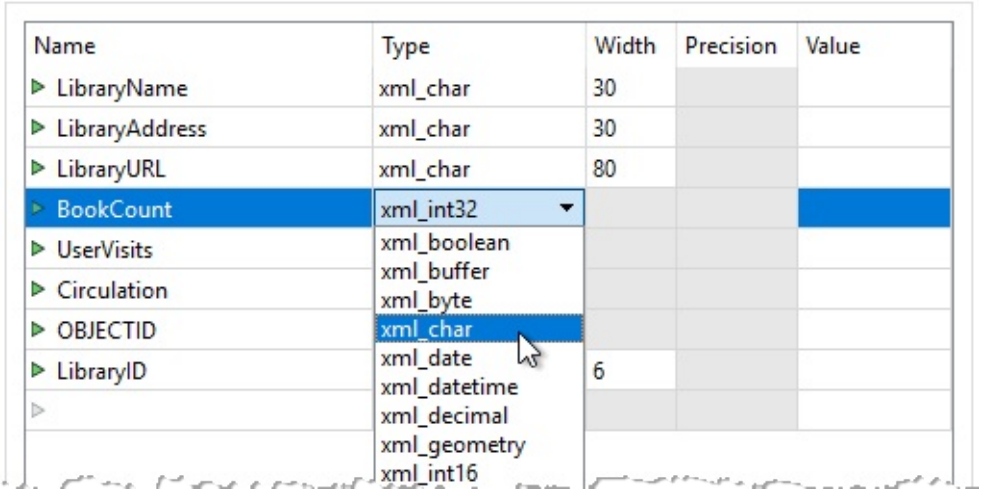
To rename an attribute open the Feature Type dialog and click the User Attributes tab. Click the attribute to be renamed and enter the new name.



Attribute Type Changes

Any attribute on the writer schema can have a change of type; for example, changing from an integer field to a character field.

To change an attribute type open the User Attributes tab and set the Type field to the new type:

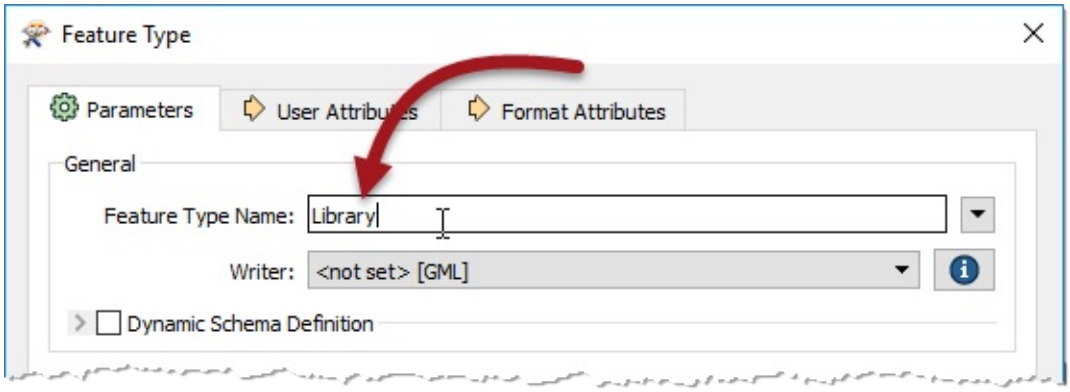


TIP

The Type column for an attribute shows only values that match the permitted types for that data format. For example, an Oracle schema permits attribute types of varchar or clob. MapInfo does not support these data types so they would never appear in a MapInfo schema. The above screenshot shows data types for the GML format.

Feature Type Renaming

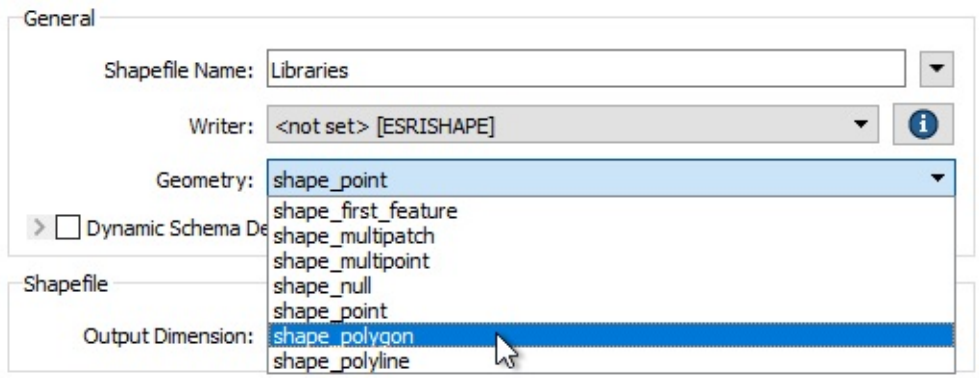
To rename a writer feature type open the Parameters tab, click in the Feature Type Name field (the label may vary according to the format type) and edit the name as required:



You can also rename a feature type by clicking on it in the Workbench canvas and pressing the F2 key.

Geometry Type Changes

To change the permitted geometry for a feature type, (for example, change the permitted geometries from lines to points) open the Parameters tab and change the permitted geometry type:



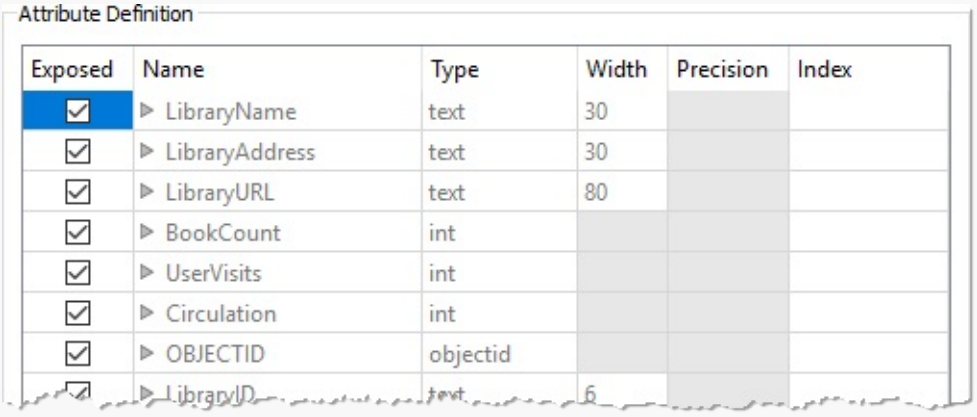
This field is only available where the format requires a strict decision on geometry type. Where formats allow any mix of geometry in a single feature type, this setting is not shown.

Once the user has made all the required changes to the writer schema, the workspace reflects *"what we want"* - but it doesn't yet specify how the reader and writer schemas should be connected together. This is the next task and it is called ***Schema Mapping***.

TIP

*You might be wondering "what would happen if I edited the **reader** schema, instead of the writer"?*

Well, by default you can't! The schema fields for a reader are greyed out to prevent this, since changes would put the schema definition out of sync with the source dataset:



There are a few, rare, use cases - but they're outside the scope of this training module!

Exercise 4	Grounds Maintenance Project - Schema Editing
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Structural Transformation, Schema Editing
Start Workspace	None
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex1-Complete.fmw

You have just landed a job as technical analyst in the GIS department of your local city.

The team responsible for maintaining parks and other grassed areas needs to know the area and facilities of each park in order to plan their budget for the upcoming year. You have been assigned to this project and will use FME to provide a dataset of this information.

The first step in this example is to rename existing attributes and create new ones in preparation for the later area calculations.

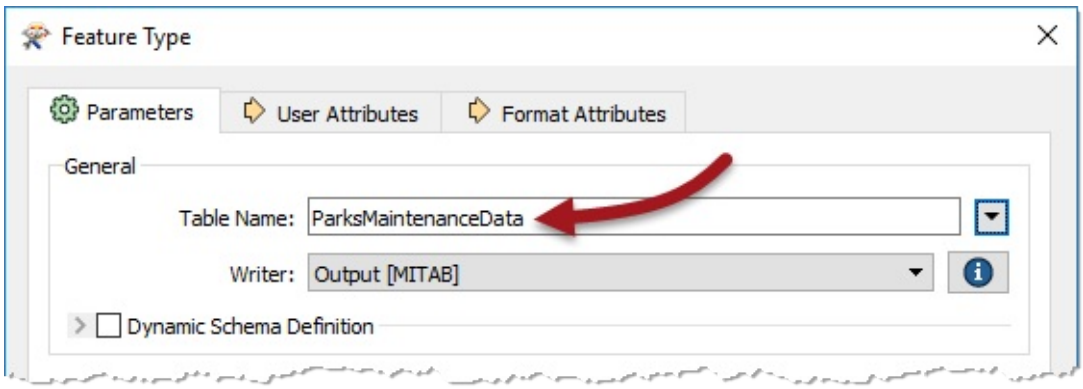
1) Start Workbench
Use the Generate Workspace dialog to create a workspace using these parameters:

Reader Format	MapInfo TAB (MITAB)
Reader Dataset	C:\FMEData2018\Data\Parks\Parks.tab
Writer Format	MapInfo TAB (MITAB)
Writer Dataset	C:\FMEData2018\Output\Training

Yes! Here we write back to the same format of data we are reading from! You may click the Parameters buttons in the Generate Workspace dialog to check the reader/writer parameters, but none of them need changing in this exercise. Note that the writer needs only a folder and not a specific file name.

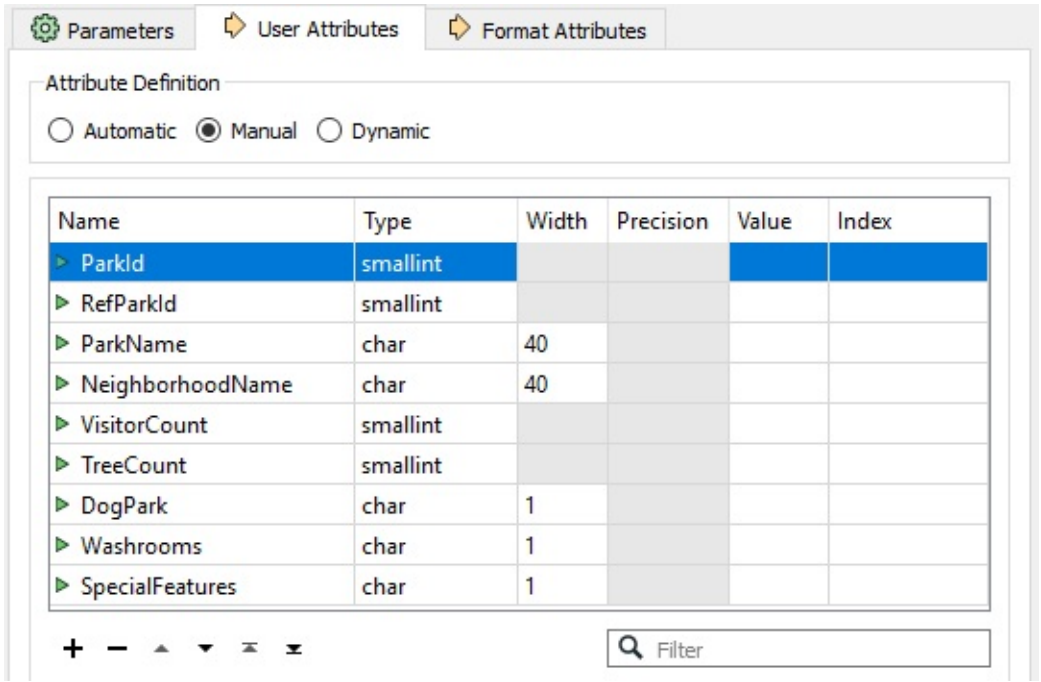
2) Rename Feature Type
FME creates a workspace where the destination schema matches the source. However, the end user of the data has requested parts of the schema are cleaned up.

Inspect the writer feature type parameters. Click in the field labelled Table Name (remember this label is format-specific and in MapInfo we deal with "tables") and change the name from Parks to ParksMaintenanceData:



3) Update Attributes

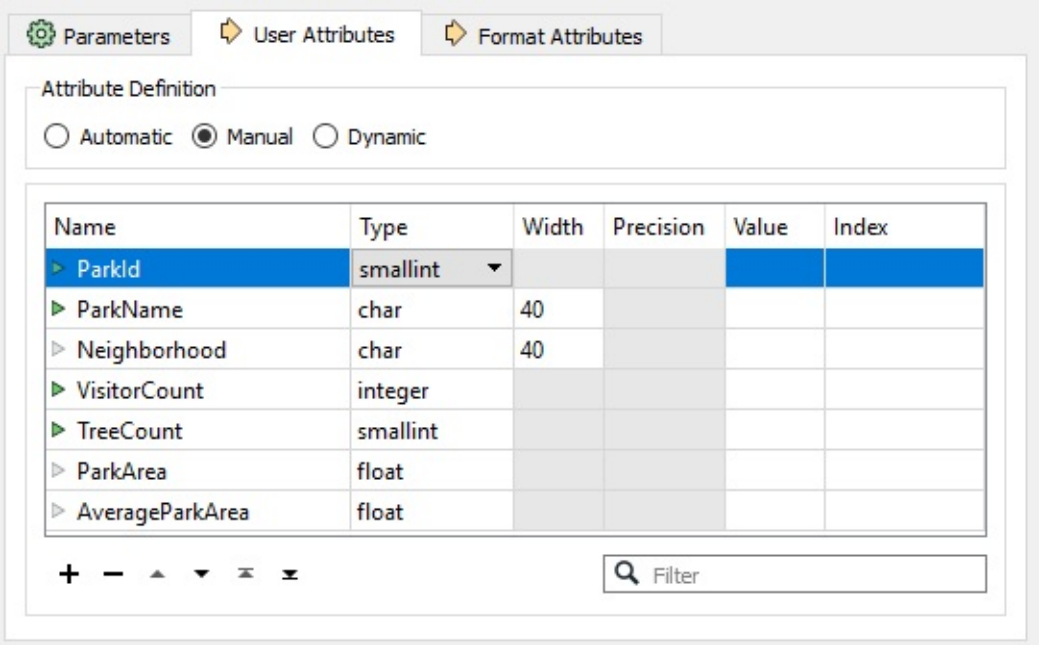
Now inspect the user attributes. They will look like this:



These must be cleaned up so that unnecessary information is removed. Other attributes need to be updated and some extra ones added to store the calculation results. So carry out the following actions:

Delete Attribute	RefParkID	
Delete Attribute	DogPark	
Delete Attribute	Washrooms	
Delete Attribute	SpecialFeatures	
Rename Attribute	from: NeighborhoodName	to: Neighborhood
Change Type (VisitorCount)	from: smallint	to: integer
Add Attribute	ParkArea	type: Float
Add Attribute	AverageParkArea	type: Float

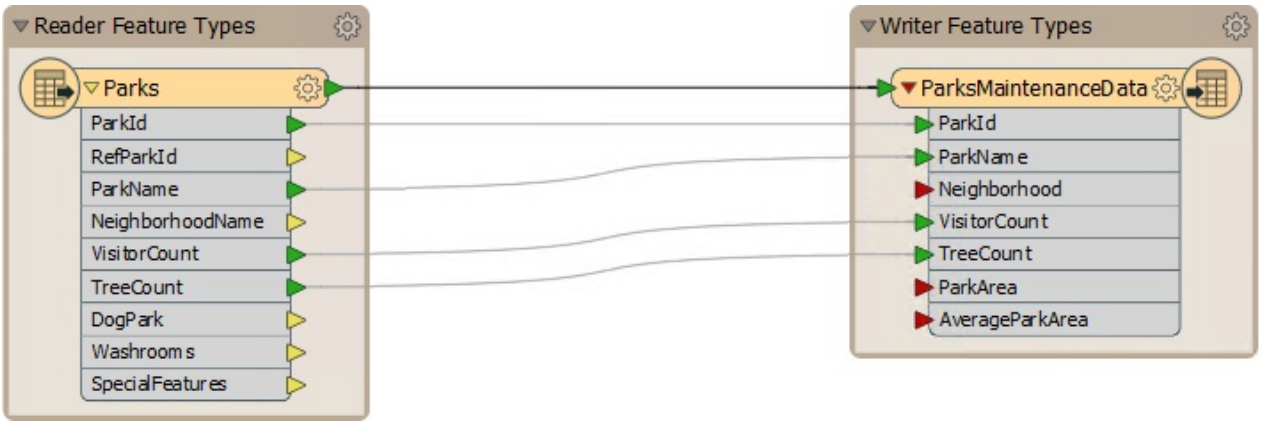
...and click the Parameter Editor "Apply" button. The attribute list should now look like this:



Now when the workspace is run the output will be named ParksMaintenanceData.tab and will contain an updated attribute schema.

4) Un-Expose Source Attributes

The workspace will now look like this:

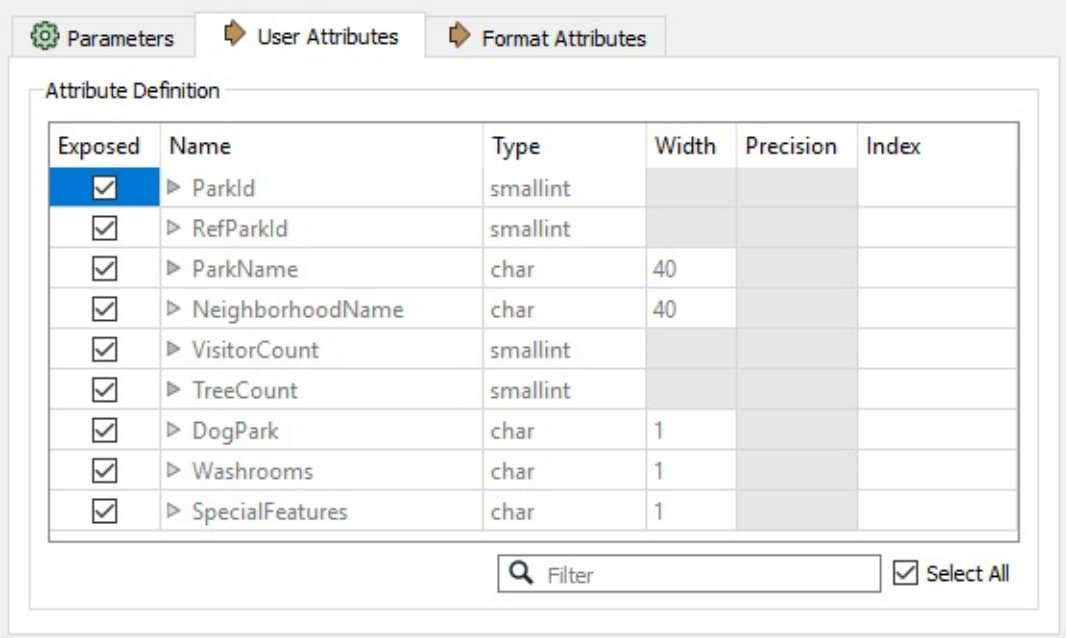


TIP

Objects on the canvas can be resized (as in the above screenshot) if the feature type name or attribute names are too large to be displayed properly at the default size. The brown markers around the feature types are called ***bookmarks****. They too can be resized to better fit their contents.*

Notice there are several source attributes that are not going to be used in the workspace or sent to the output. We can tidy the workspace by hiding these.

Inspect the User Attributes tab on the reader feature type parameters. It will look like this:



Uncheck the check box for the following attributes, which we do not need:

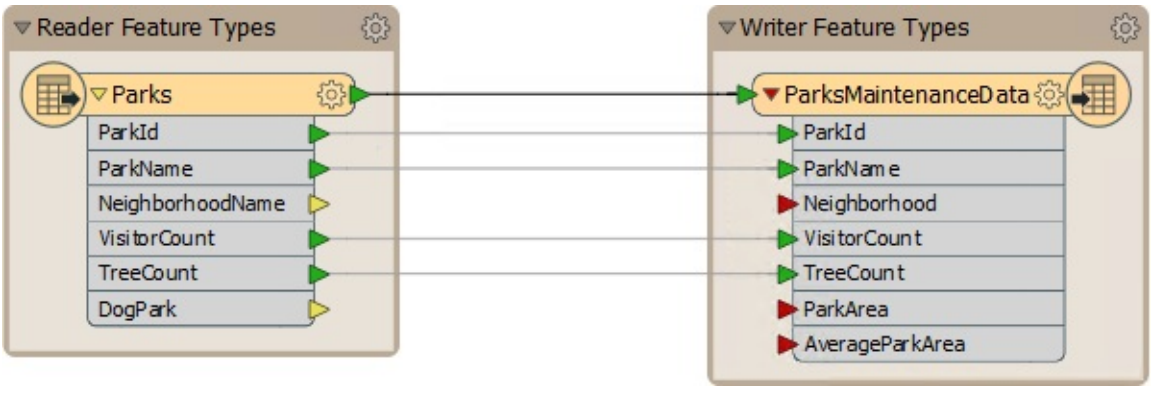
- RefParkID
- Washrooms
- SpecialFeatures

This is basically the list of attributes we deleted, except for DogParks, which we will make use of in the translation.

Click Apply/OK to confirm the changes.

5) Save the Workspace

Save the workspace – it will be completed in further examples. It should now look like this:



Police Chief Webb-Mapp says...

Some Writer attributes (ParkArea and AverageParkArea) have red connection arrows because there is nothing yet to map to them, while another (Neighborhood) is just unconnected.

That's OK. I'll let you off with a caution if you promise to connect them later. You can still run this workspace just to see what the output looks like anyway.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Edit the attributes on a writer schema*
- *Edit the output layer name on a writer schema*
- *Hide attributes on a reader schema*

Schema Mapping

Schema Mapping is the second key part towards data restructuring in FME.

In FME Workbench, one side of the workspace shows the reader schema (*"what we have"*) and the other side shows the writer schema (*"what we want"*). Initially, the two schemas are automatically joined when the workspace is created, but when edits occur these connections are usually lost.

Schema Mapping is the process of connecting the reader schema to the writer schema in a way that ensures the correct reader features are sent to the correct writer feature types and the correct reader attributes are sent to the correct writer attributes.

FME permits mapping from source to destination in any arrangement desired. There are no restrictions on what feature types or attributes may be mapped.

Ms. Analyst says...

Hi. I'm Ms. Analyst, one of your colleagues at the city. I think of schema editing and schema mapping as a means of reorganizing data.

A good analogy is a wardrobe full of clothes. When the wardrobe is reorganized you throw out what you no longer need, reserve space for new clothes that you're planning to get, and move existing items into a more usable arrangement.

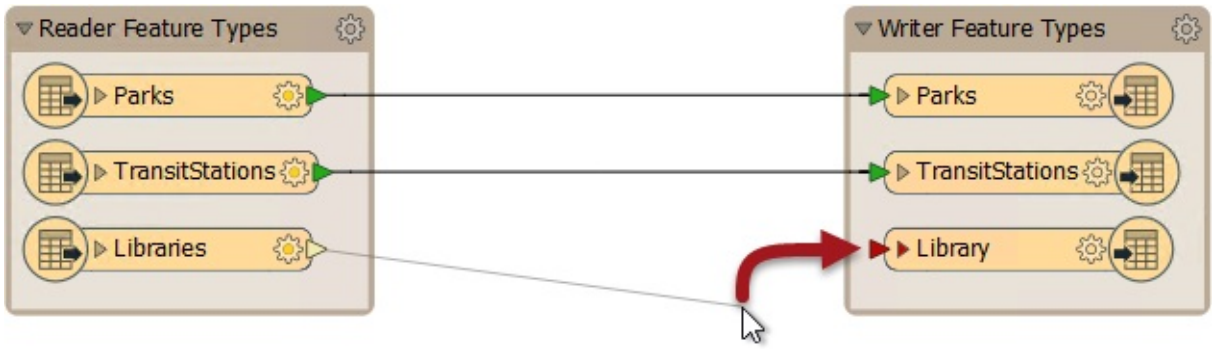
The same holds true for spatial data restructuring: it's the act of reorganizing data to make it more usable.

Feature Mapping

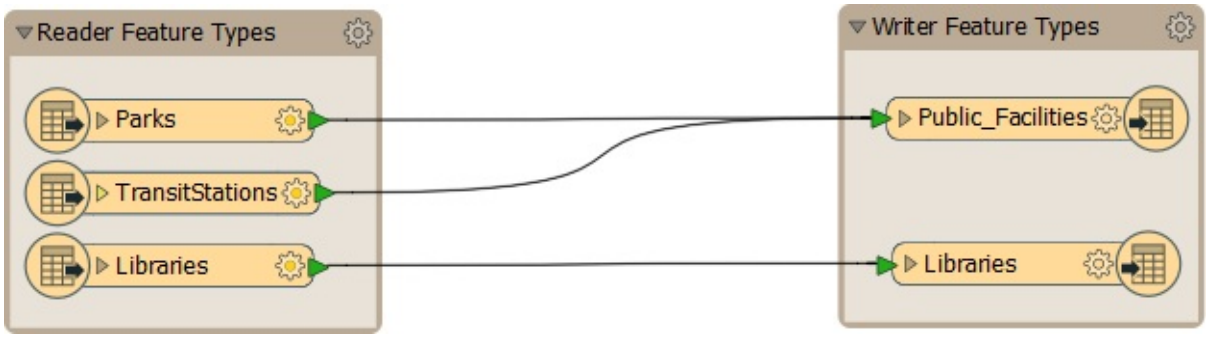
Feature Mapping is the process of connecting source feature types to destination feature types.

Feature mapping is carried out by clicking the output port of a reader feature type, dragging the arrowhead to the input port of a writer feature type, and releasing the mouse button. A thick, black line represents connections.

Here, a connecting line from the source to the destination feature type is created by dragging the arrowhead from the source to the destination:



Merging and splitting of data is permitted:



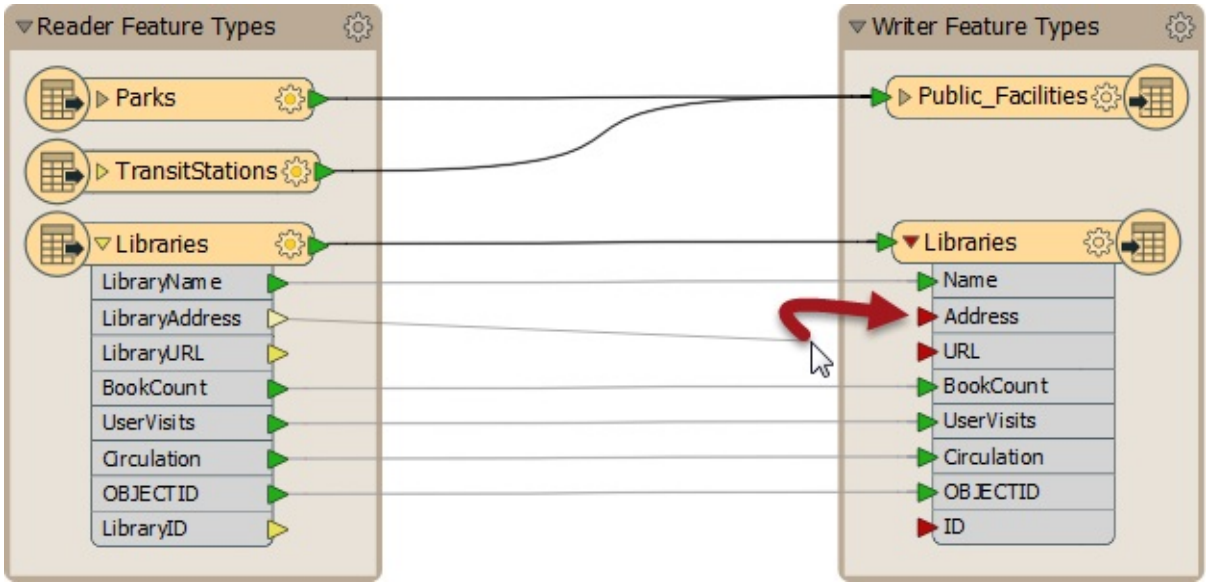
Here a user requires both Parks, TransitStations, and Libraries to be on a single layer in the output and so merges those three reader feature types into one writer feature type (PublicFacilities).

Attribute Mapping

Attribute Mapping is the process of connecting reader attributes to writer attributes.

Attribute mapping is performed by clicking the output port of a reader attribute, dragging the arrowhead to the input port of a writer attribute, and releasing the mouse button. Attribute connections are shown with a thinner, gray line.

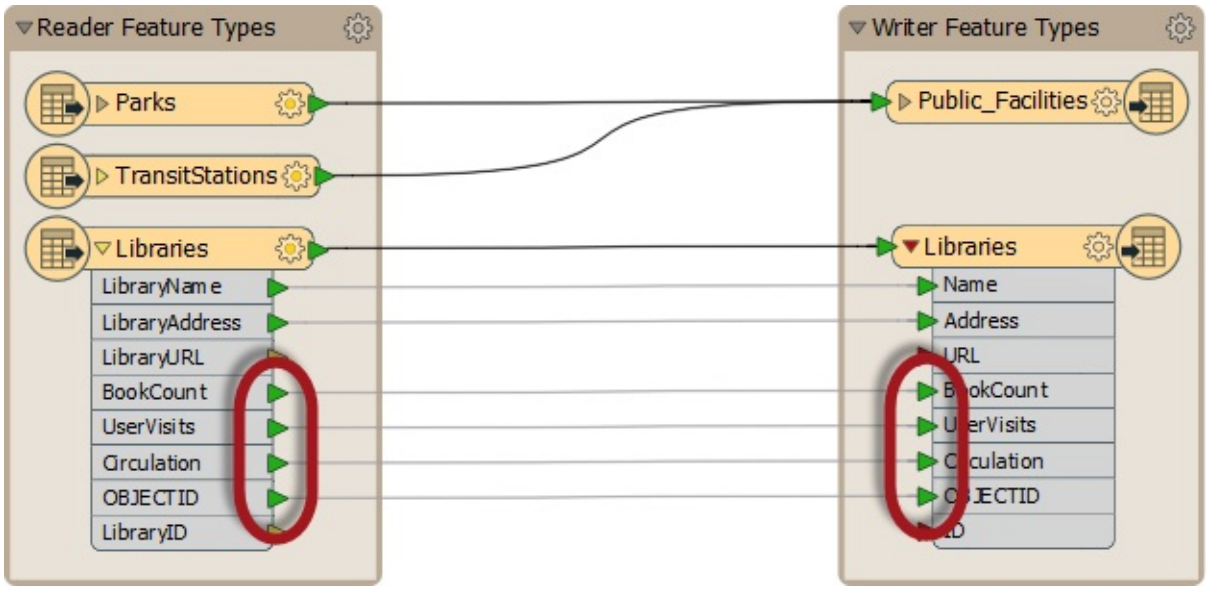
Here feature mapping has been carried out already and attribute connections are being made:



Notice the green, yellow, and red color-coding that indicates which attributes are connected.

Green ports indicate a connected attribute. Yellow ports indicate a reader attribute that's unconnected to a writer. Red ports indicate a writer attribute that's unconnected to a reader.

Attributes with the same name in reader and writer feature types are connected automatically, even though a connecting line might not be visible; the port color is the key:



WARNING

Names are case-sensitive, therefore ROADS is not the same as roads, Roads, or rOADS.
That's important to know if you are relying on automatic attribute connections!

Exercise 5	Grounds Maintenance Project - Structural Transformation
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Structural Transformation with Transformers
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex2-Begin.fmw
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex2-Complete.fmw C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex2-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In case you forgot, the team responsible for maintaining parks and other grassed areas needs to know the area and facilities of each park in order to plan their budget for the upcoming year.

In this part of the project we'll filter out dog parks from the source data (as these have a different scale of maintenance costs) and write them to the log window. We'll also handle the renamed attribute NeighborhoodName.

1) Start FME Workbench

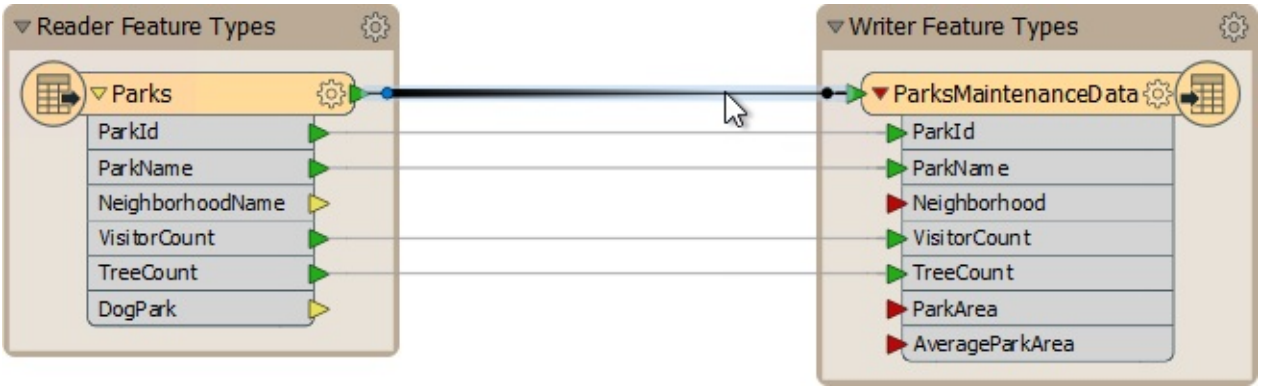
Start FME Workbench and open the workspace from the previous exercise. Alternatively you can open C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex2-Begin.fmw.

2) Add Transformer

Let's first handle the source attribute NeighborhoodName, which was renamed Neighborhood on the writer but not yet connected.

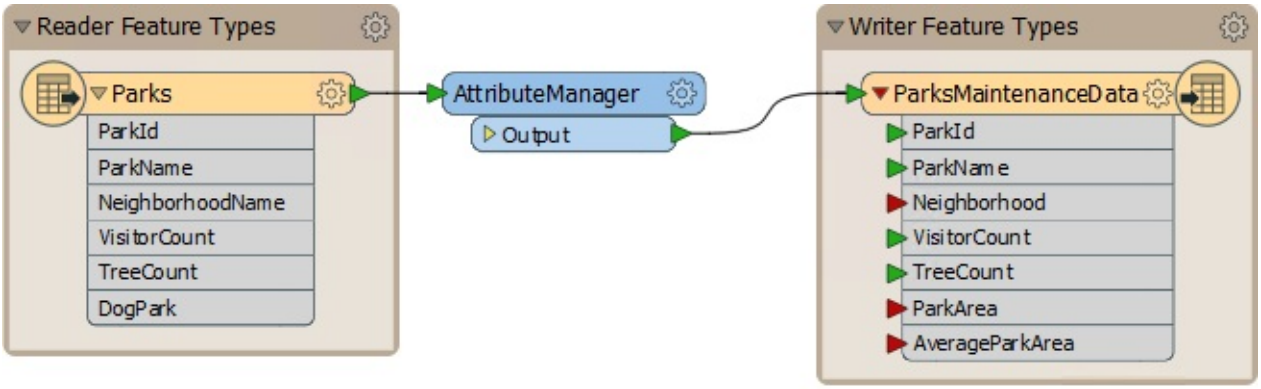
We could do this by simply drawing a connection, but it's generally better to use a transformer. There are two transformers we could use. One is called the AttributeRenamer and the other - which we shall use here - is the AttributeManager.

Therefore click on the feature connection from reader to writer feature type:



Start to type the phrase "AttributeManager". This is how we can place a transformer in the workspace. As you type, FME searches for a matching transformer. When the list is short enough for you to see the AttributeManager, select it from the dialog (double-click on it):

This will place an AttributeManager transformer like so:

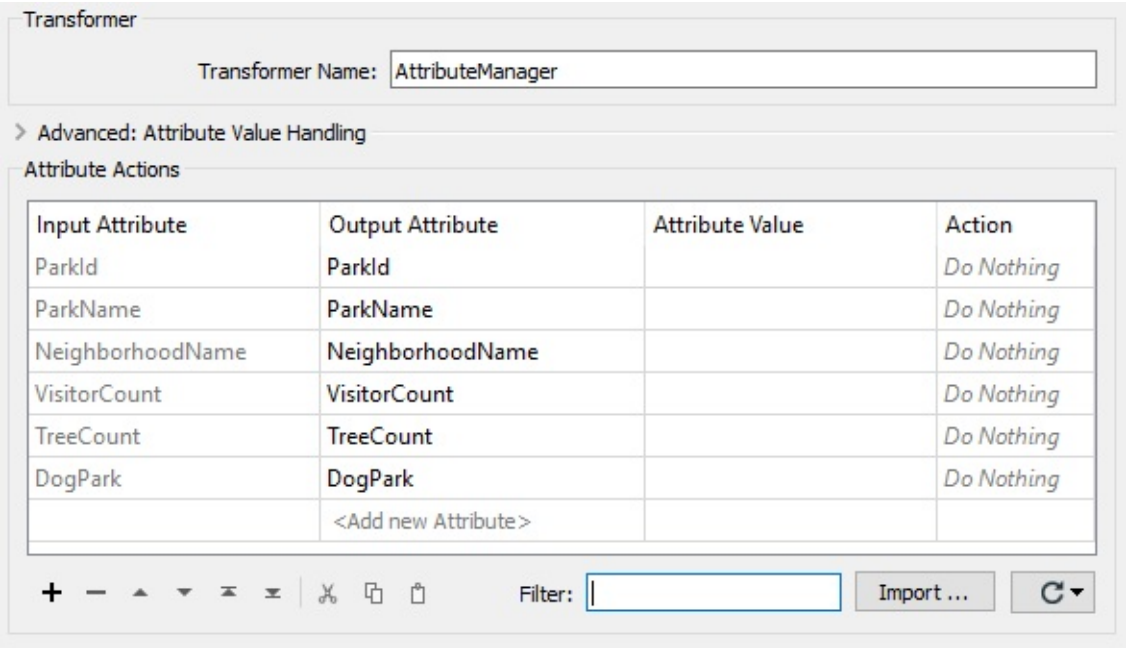


TIP

For a great tip on adding transformers see #5 in our list of ***The Top Ten FME Tips of All Time!***

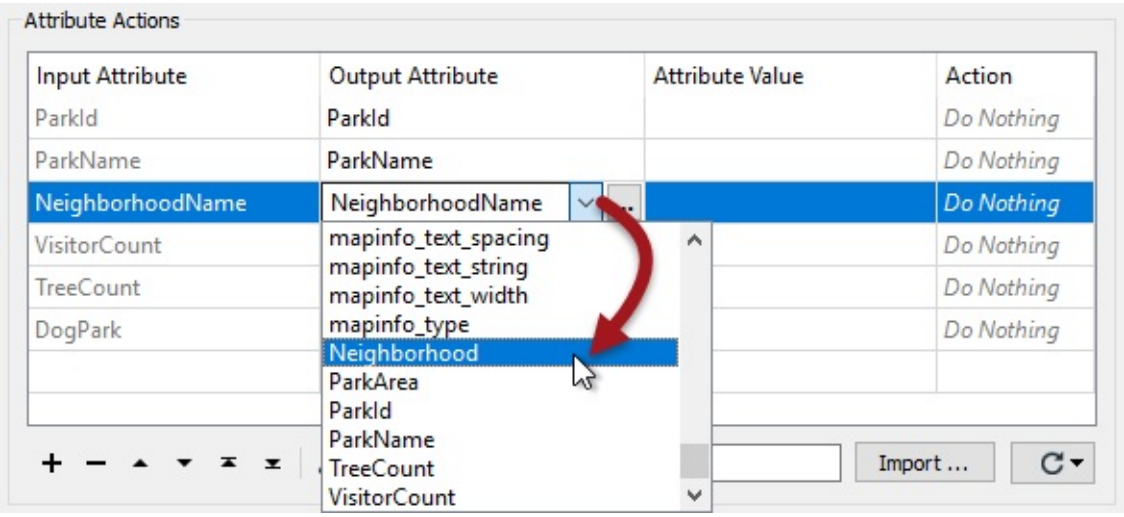
3) Set Parameters

View the AttributeManager parameters (either its dialog or in the Parameter Editor window). It will look like this:



Notice that all of the attributes on the stream in which it is connected will automatically appear in the dialog.

Where the Input Attribute field is set to NeighborhoodName, click in the Output Attribute field. Click on the button for the drop-down list and in there choose Neighborhood as the new attribute name to use:

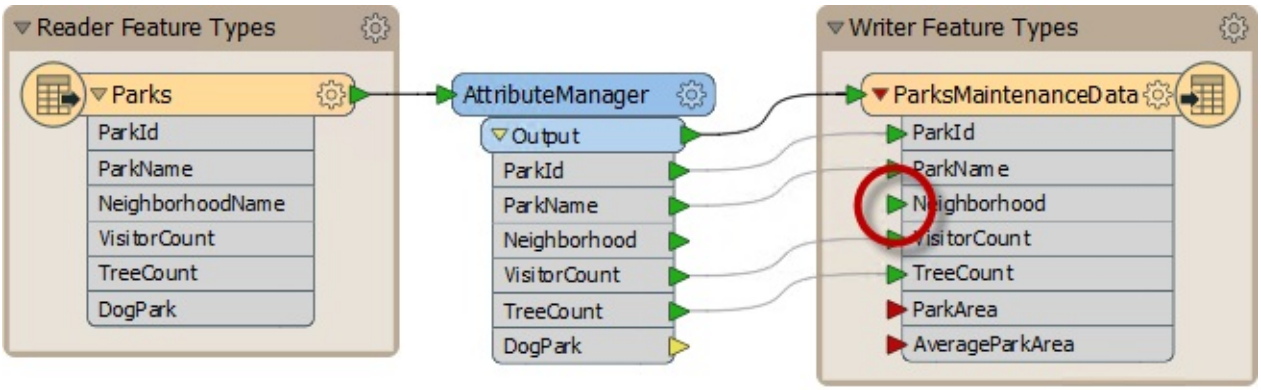


In response the Action field will change to read *Rename*.

TIP

Neighborhood only appears in the list because it already exists on the writer schema. If we had done this step before editing the writer schema, we would have had to manually enter the new attribute name in this dialog.

Click OK to close the dialog. Now in the Workbench canvas window you will see the Neighborhood attribute is flagged with a green arrow, to confirm that a value is being provided to that attribute.

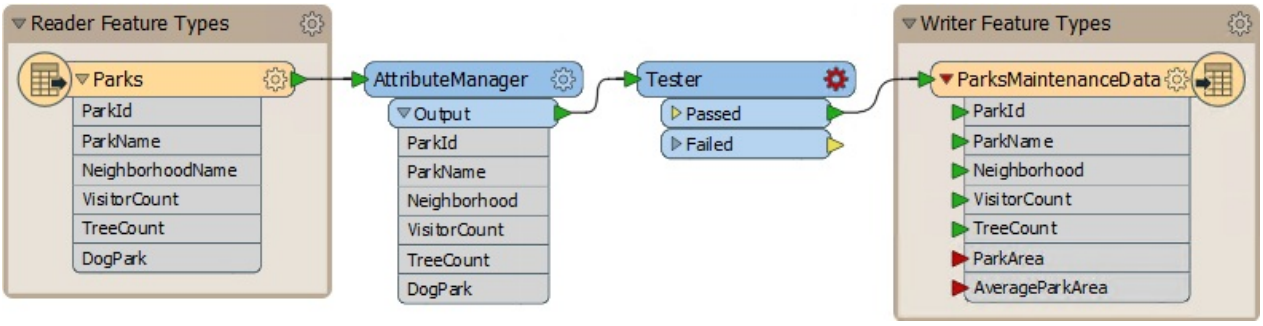


4) Add Transformer

Now we should remove dog parks from the data, because these have their own set of costs.

This can be done with a Tester transformer. Click on the connection from the AttributeManager output port to the ParksMaintenanceData feature type on the Writer.

Start typing the word Tester. When you spot the Tester transformer double-click on it to add one to the workspace. After tidying up the layout of the canvas, the workspace will now look like this:

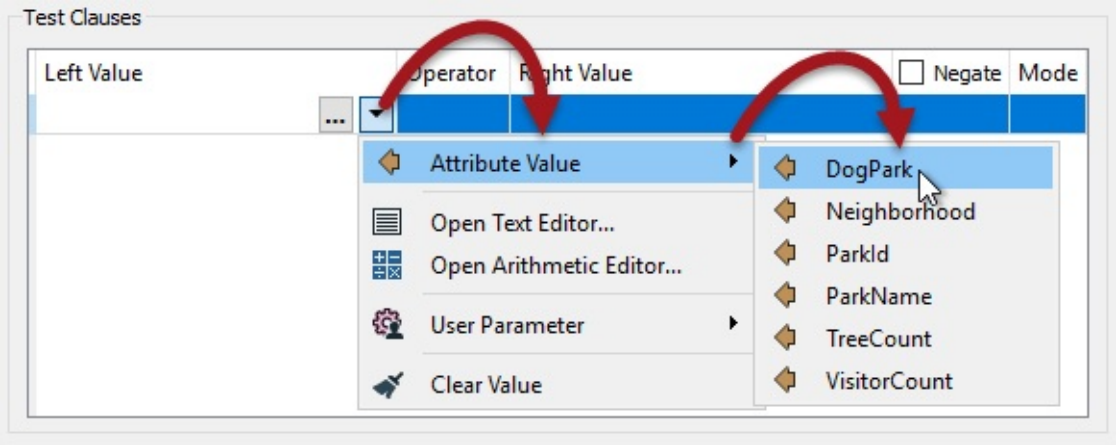


Notice that the Passed output port is the one connected by default.

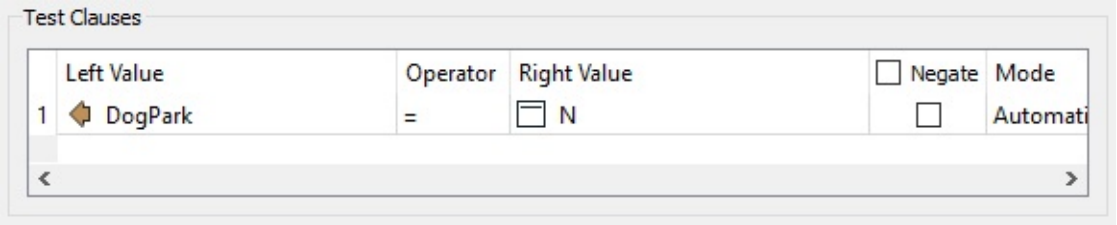
5) Set Parameters

Inspect the parameters for the Tester transformer. Click in the Left Value field and from there

click the down arrow and choose Attribute Value > DogPark:



For the Right Value click into the field and type the value N. The operator field should be filled in automatically as the equals sign (=), which is what we want in this case.



Click OK to accept the values and close the dialog.

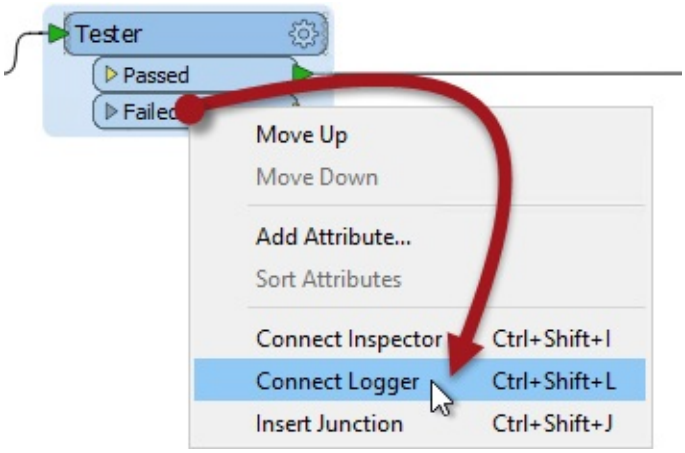
TIP

The test is for DogParks=N because we want to keep those features, and it is the Passed port that is connected. If the test was DogParks=Y then the Failed port would be the one to connect.

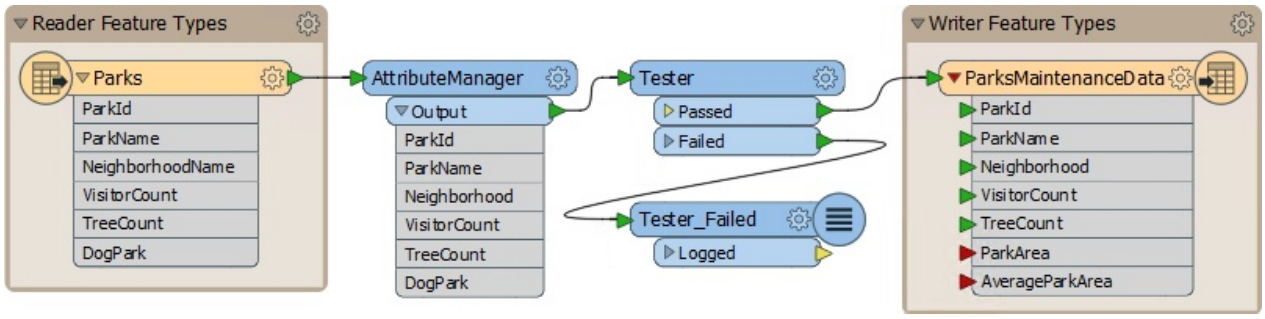
6) Add Transformer

We are now filtering out dog parks from our data, using a test on an attribute value. The question is, what should we do with the features we have filtered out? There are many things we could do, but for now we'll simply log the information to the FME log window.

To do this, right-click on the Tester Failed port and choose the option Connect Logger:



A Logger transformer will be added to the workspace. This will record all incoming features to the log window:



Loggers inserted by this method are named after – and reported in the log with – the output port they are connected to, here Tester_Failed.

7) Run Workspace

Save and run the workspace. It is not yet complete but running it will prove that everything is working correctly up to this point.

Advanced Exercise

Readers and writers on the canvas had a bookmark object added around them automatically. You can do the same to the transformers by selecting them all and either clicking Bookmark on the toolbar or pressing the Ctrl+B keys. Give the bookmark a name of your choice that describes the actions being carried out.

A bookmark is an example of what we call ***Best Practice*** *in FME.*

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Add transformers to a workspace*
- *Carry out schema mapping with the AttributeManager transformer*
- *Filter data using the Tester transformer*
- *Record data using the Logger transformer*
- *Add bookmarks to a workspace*

Content Transformation

Content transformations are those that operate on the components of a feature.

What is a Feature?

A ***feature*** in FME is an individual item within the translation. For spatial data, a feature is generally a geometric object (with or without a set of related attributes).

For spreadsheet (i.e. tabular) data, a feature is generally a record in a database, a row in a spreadsheet, or a line in a text file. Each column or cell is known as an ***attribute***.

Features in FME have a flexible, generic representation that is unrelated to the format from which they originated. That means any transformer can operate on any FME feature, regardless of its source format. Sometimes content transformation operates on single features, sometimes on multiple features at once.

Ms. Analyst says...

You can think of Content Transformation as altering or editing data.

The wardrobe analogy still works here. You might take your clothes from the wardrobe to clean them, or alter them, or repair them, or dye them a new color, or all sorts of other tasks, before returning them to their place.

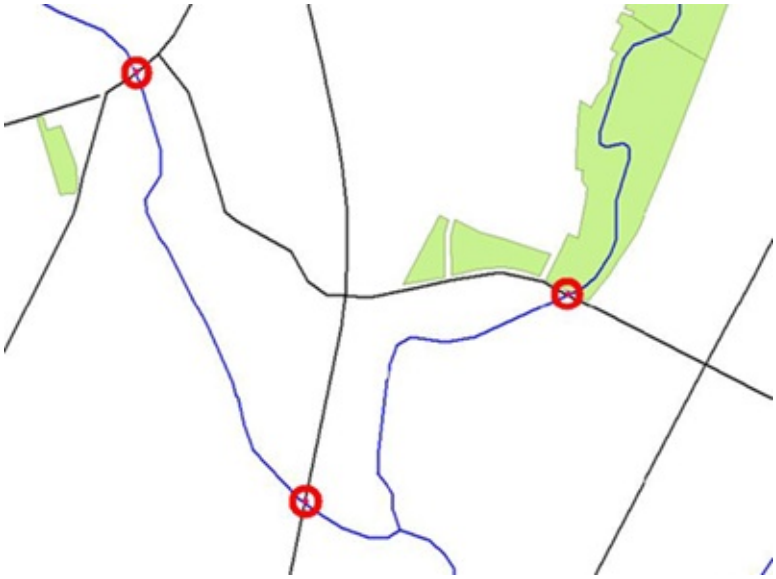
The same holds true for spatial data transformation: it's the act of fixing up your data to be cleaner and in the style you really want.

Geometric Transformation

Geometric Transformation is the act of restructuring the spatial component of an FME feature. In other words, the geometry of the feature undergoes some form of change to produce a different output.

Some examples of geometric transformation include the following:

- **Generalization** – a cartographic process that restructures data so it's easily visualized at a given map scale
- **Warping** – adjustment of the size and shape of a set of features to more closely match a set of reference data
- **Topology Computation** – conversion of a set of linear features into a node/line structure
- **Line Intersection** - calculation of the intersection points between line features



Here roads have been intersected with rivers to produce points that mark the location of bridges.

Attribute Transformation

Attribute Transformation is the act of restructuring the tabular components of an FME feature. In other words, the attributes undergo some form of change to produce a different output.

Some examples of attribute transformation are:

- **Concatenation** – joining together of two or more attributes
- **Splitting** – splitting one attribute into many, which is the opposite of Concatenation
- **Measurement** – measuring a feature's length or area to create a new attribute
- **ID Creation** – creating a unique ID number for a particular feature

Attribute concatenation as an example of attribute transformation.

Each line of the address is concatenated to return a single line address.

```
 Address1     Suite 2017,+
Address2     7445-132nd Street,+
City            Surrey,+
Province     British Columbia,+
PostalCode     V3W 1J8

= Address     Suite 2017, 7445-132nd Street, Surrey, British Columbia, V3W 1J8
```

Miss Vector says...

Did you miss me? You did? Well I'll cure that with some new questions for you!

Which three colours represent checked, need checking, and unset parameters on transformer objects?

1. blue, yellow, red
2. green, yellow, red
3. red, green, blue
4. green, blue, yellow

If I use a transformer to remove irregularities (like self-intersecting loops) in the boundary of a polygon, what type of transformation is it?

1. Structural Transformation of attributes
2. Structural Transformation of geometry
3. Content Transformation of attributes
4. Content Transformation of geometry

Exercise 6	Grounds Maintenance Project - Calculating Statistics
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Content Transformation. Schema Mapping
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex3-Begin.fmw
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex3-Complete.fmw C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex3-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In case you forgot, the team responsible for maintaining parks and other grassed areas needs to know the area and facilities of each park in order to plan their budget for the upcoming year.

In this part of the project we'll calculate the size and average size of each park, and ensure that information is correctly mapped to the destination schema.

1) Start Workbench

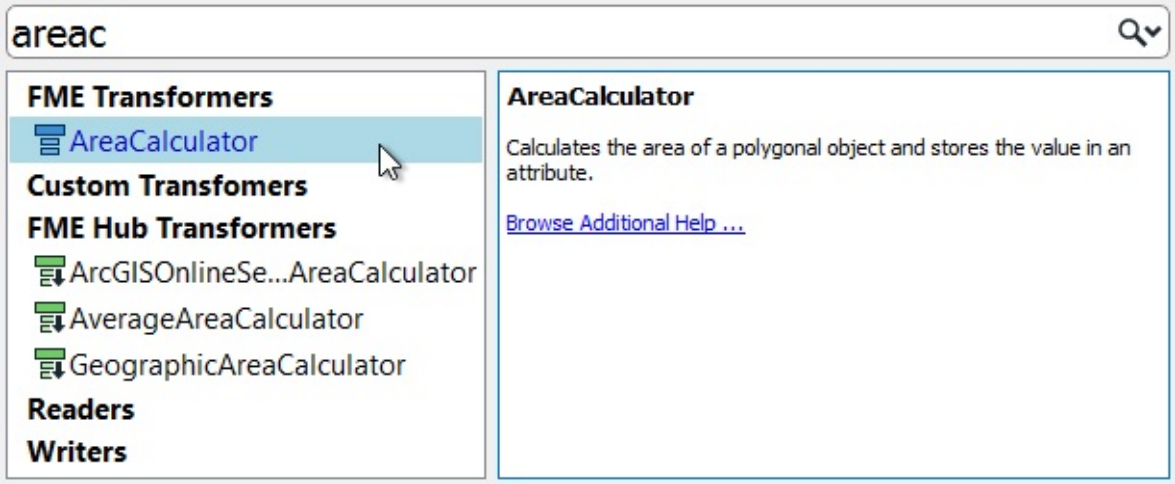
Start Workbench (if necessary) and open the workspace from Exercise 2. Alternatively you can open C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex3-Begin.fmw

2) Add an AreaCalculator Transformer

To measure the area of each park feature, an AreaCalculator transformer must be used. "Calculator" is the term for when FME computes new attribute values.

Click onto the connection between Tester:Passed port and the writer feature type *ParksMaintenanceData*. Start typing the letters "areac". You will see the Quick Add list of matching transformers appear beneath.

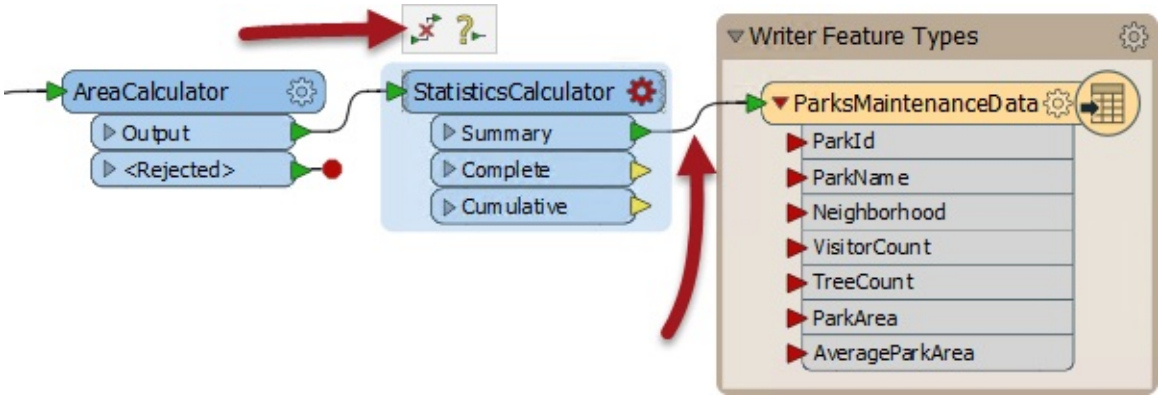
Select the transformer named AreaCalculator by double-clicking it:



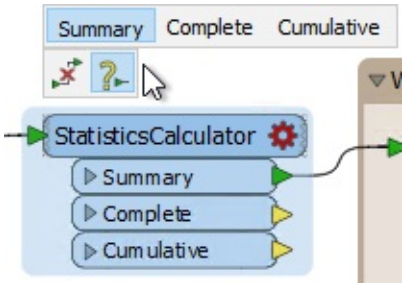
3) Add a StatisticsCalculator Transformer

Using the same method, place a StatisticsCalculator transformer between the AreaCalculator:Output port and the *ParksMaintenanceData* feature type.

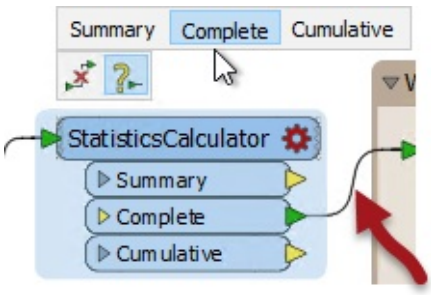
BUT! Do not click anything else yet! The transformer will now look like this:



By default the Summary port has been connected, and we need the Complete port connected instead. But notice the little pop-up icons over the top. Click the right-hand icon (the one with the ? character). This pops up a further list of ports:



Click on the Summary port entry to disconnect that, and then on the Complete port entry to connect that:

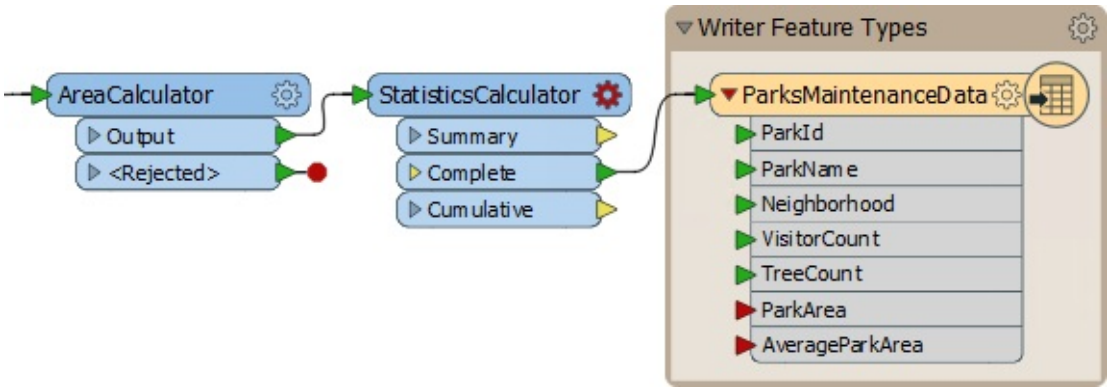


Notice how the connection has been changed from the incorrect Summary port to the correct Complete port.

TIP

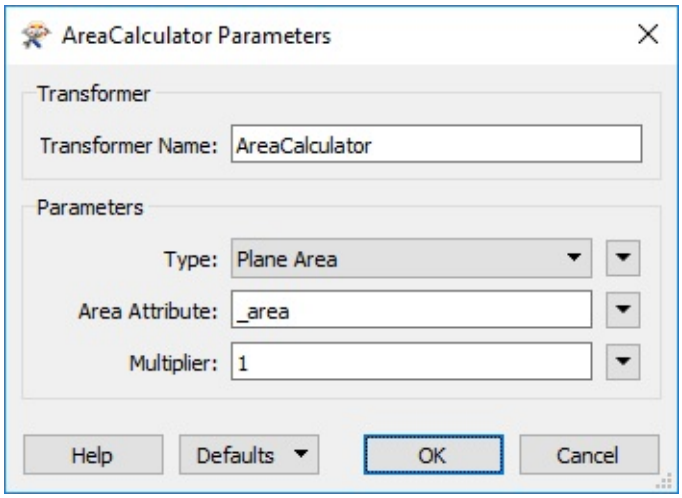
These pop-up menus are a great help in schema mapping and other feature connections.

The latter part of the workspace now looks like this:

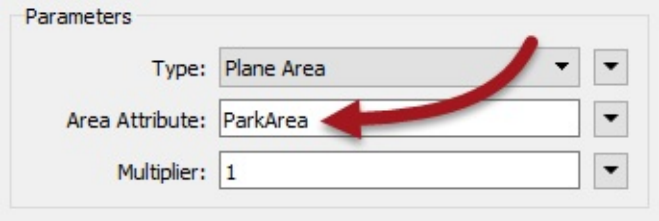


4) Check AreaCalculator Settings

Inspect the AreaCalculator parameters:



The default settings cause the calculated value to be placed into an attribute called _area. However, the *ParksMaintenanceData* schema requires an attribute called ParkArea, so change this parameter to create the correct attribute:

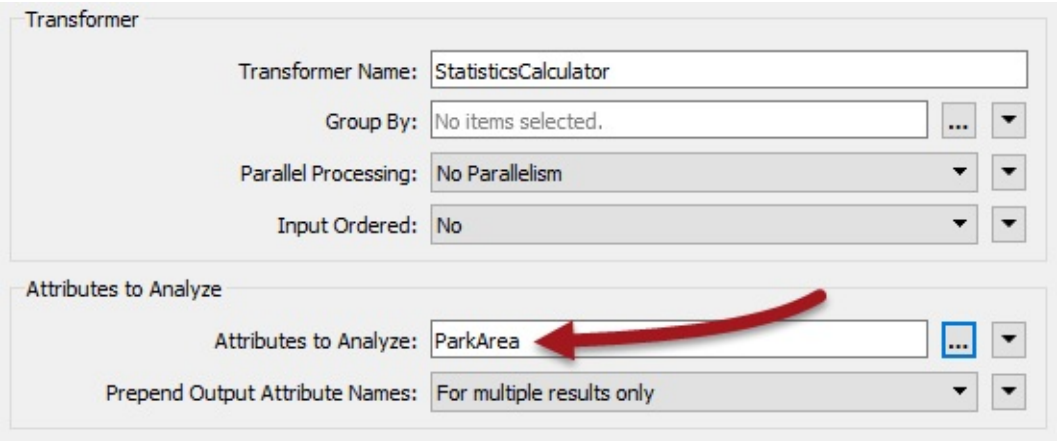


Once you click OK or Apply notice that the attribute on the writer feature type is now flagged as connected.

5) Check StatisticsCalculator Settings

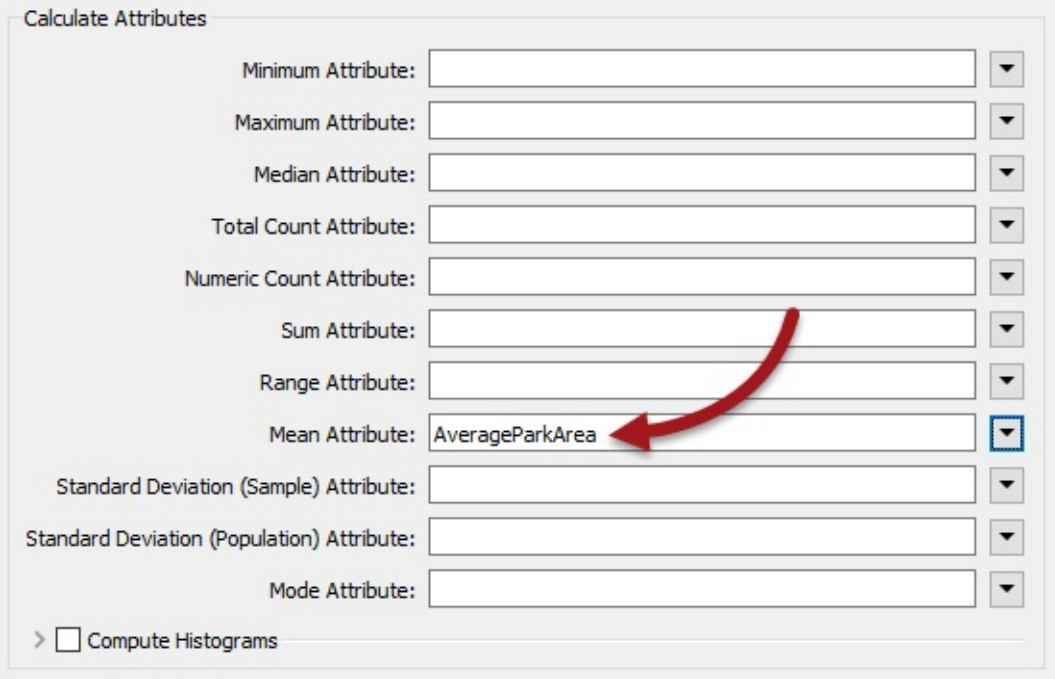
A red icon indicates the StatisticsCalculator has parameters that need to be defined.

Inspect the StatisticsCalculator transformer's parameters. The attribute to analyze is the one containing the calculated area; so select ParkArea:



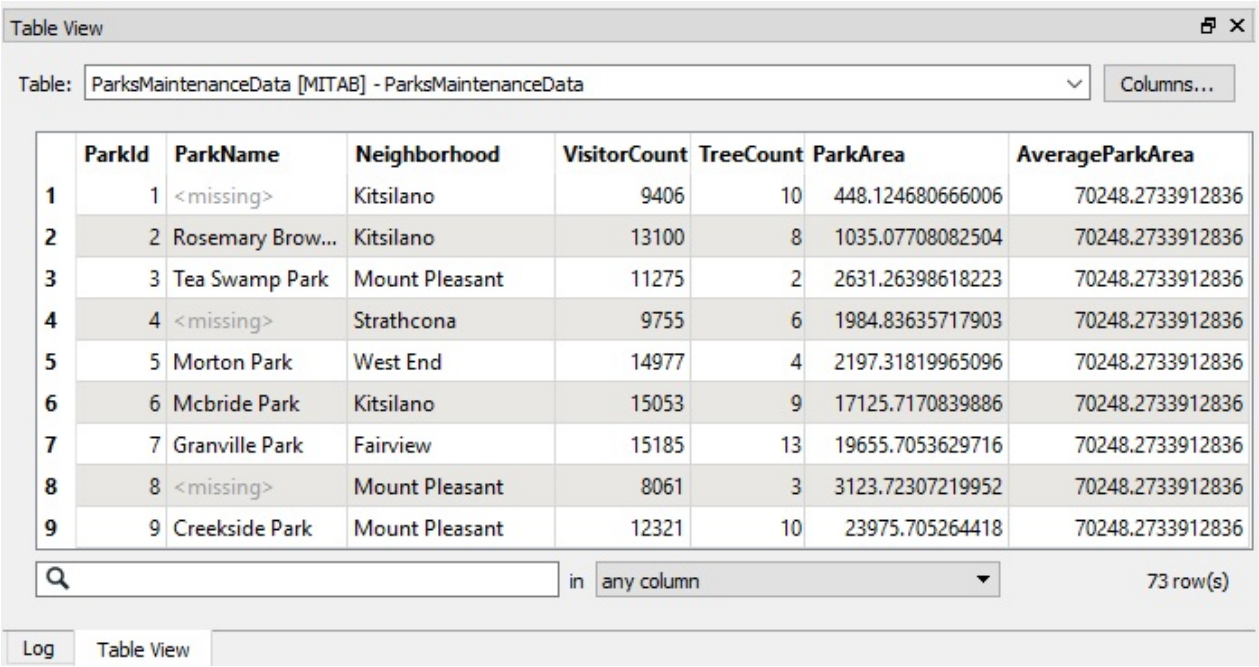
Examine what the default setting is for an attribute name for average (mean) park size. Currently it doesn't match the *ParksMaintenanceData* schema, which requires an attribute named AverageParkArea.

Change the attribute from _mean to AverageParkArea. For Best Practice reasons, delete/unset any StatisticsCalculator output attributes that aren't required (for example _range and _stdev).



6) Run the Workspace

Add another bookmark if you wish, run the workspace, and inspect the result of the translation using the FME Data Inspector:



	ParkId	ParkName	Neighborhood	VisitorCount	TreeCount	ParkArea	AverageParkArea
1	1	<missing>	Kitsilano	9406	10	448.124680666006	70248.2733912836
2	2	Rosemary Brow...	Kitsilano	13100	8	1035.07708082504	70248.2733912836
3	3	Tea Swamp Park	Mount Pleasant	11275	2	2631.26398618223	70248.2733912836
4	4	<missing>	Strathcona	9755	6	1984.83635717903	70248.2733912836
5	5	Morton Park	West End	14977	4	2197.31819965096	70248.2733912836
6	6	Mcbride Park	Kitsilano	15053	9	17125.7170839886	70248.2733912836
7	7	Granville Park	Fairview	15185	13	19655.7053629716	70248.2733912836
8	8	<missing>	Mount Pleasant	8061	3	3123.72307219952	70248.2733912836
9	9	Creekside Park	Mount Pleasant	12321	10	23975.705264418	70248.2733912836

The Table View window shows the area of each park and the average area of all parks.

7) Save the Workspace

Save the workspace – it will be completed in further examples.

Advanced Exercise

Notice that the numbers in the Table View show the results have been calculated to 12 decimal places. This is in excess of the precision that you require. As an advanced task - if you have time - use the AttributeRounder transformer to reduce the values to just 2 decimal places.

If you wish, you can also calculate the smallest, largest, and total park areas; but don't forget to add them to the writer schema if you want them to appear in the output.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Carry out content transformation with transformers (AreaCalculator, StatisticsCalculator)*
- *Manage transformer connections using pop-up buttons*
- *Use transformer parameters to create attributes that match the writer schema*

Transformation with Transformers

Besides Schema Editing and Schema Mapping, transformation can be carried out using objects called ***transformers***.

What is a Transformer?

As the name suggests, a transformer is an FME Workbench object that carries out the transformation of features. There are lots of FME transformers, each of which carries out many different operations.

Transformers are connected somewhere between the reader and writer feature types, so that data flows from the reader, through a transformation process, and on to the writer.

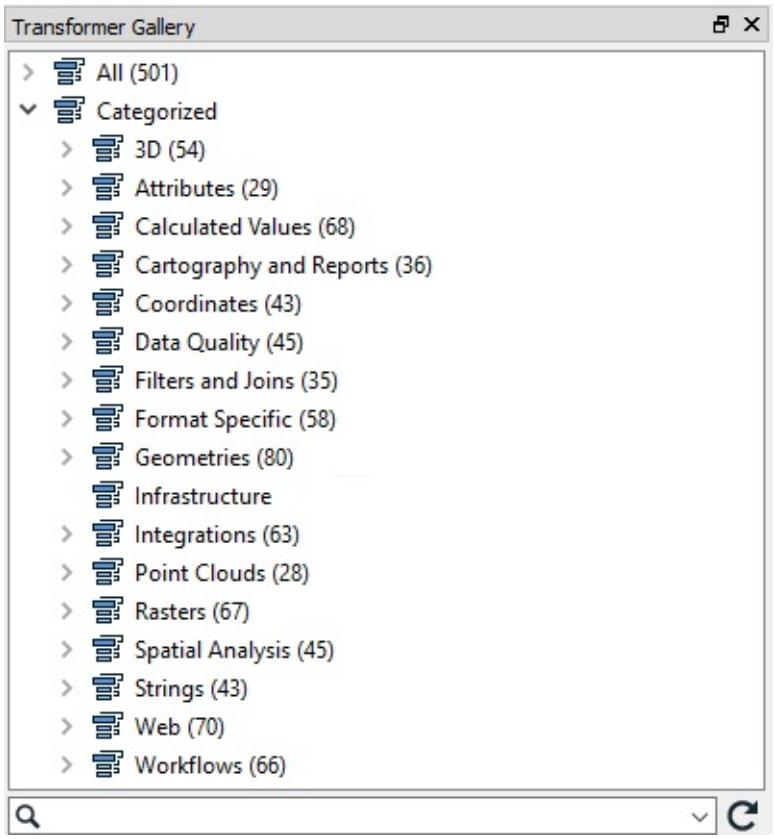
Transformers usually appear in the canvas window as rectangular, light-blue objects.

Adding Transformers to the Canvas

You can add a transformer to the canvas in two ways.

Transformer Gallery

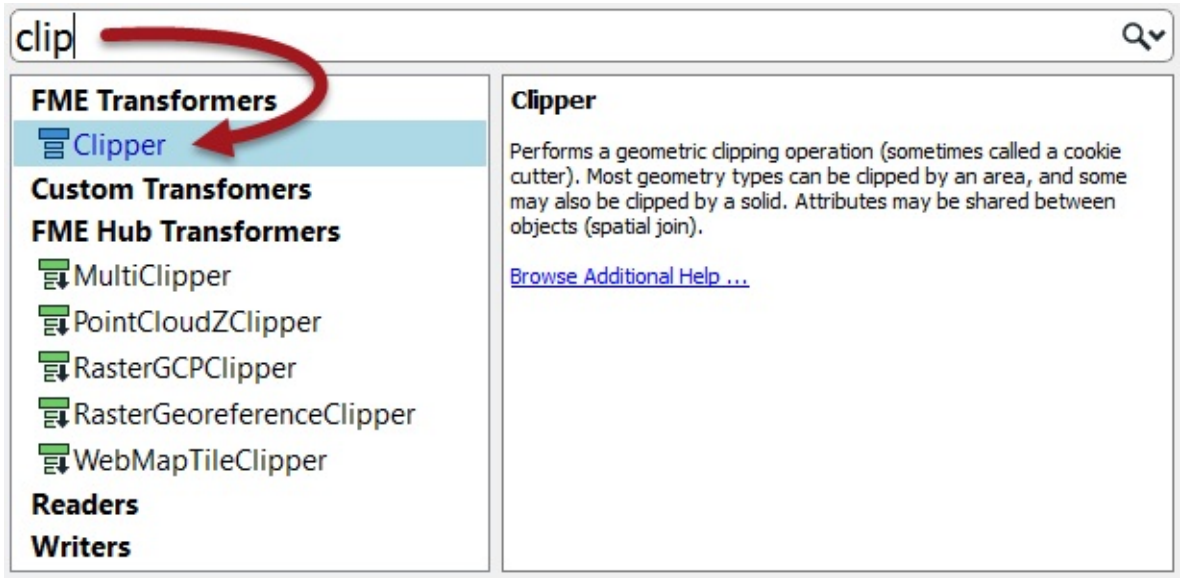
As discussed earlier, the Transformer Gallery window allows you to browse all your transformers. They are organized by category. This method is useful if you are not quite sure which transformer you need, as you can expand the categories and look for the one you need.



The Transformer Gallery is also available online, where you can also view related transformers and sort by the most used transformers.

Add Transformer or Quick Add

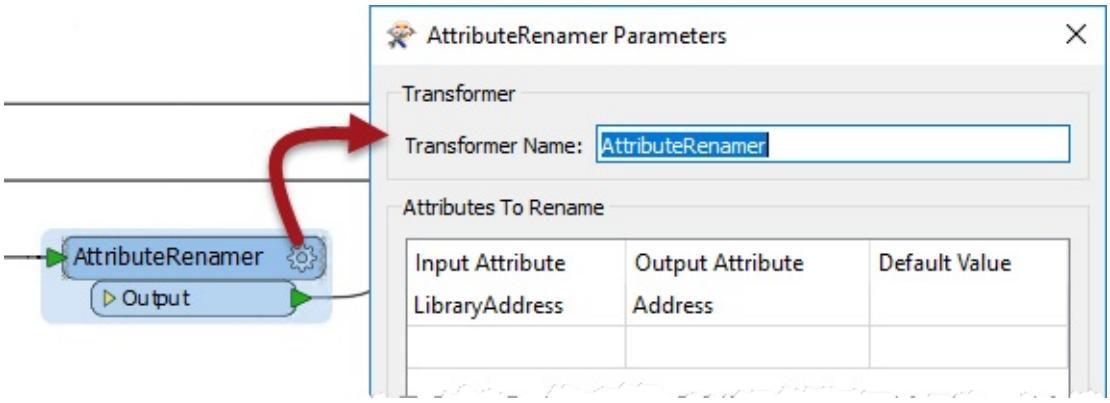
You can add a transformer to your canvas by clicking Transformers > Add Transformers (keyboard shortcut /). This opens a search box on your canvas where you can start to type the name of your transformer and view your options.



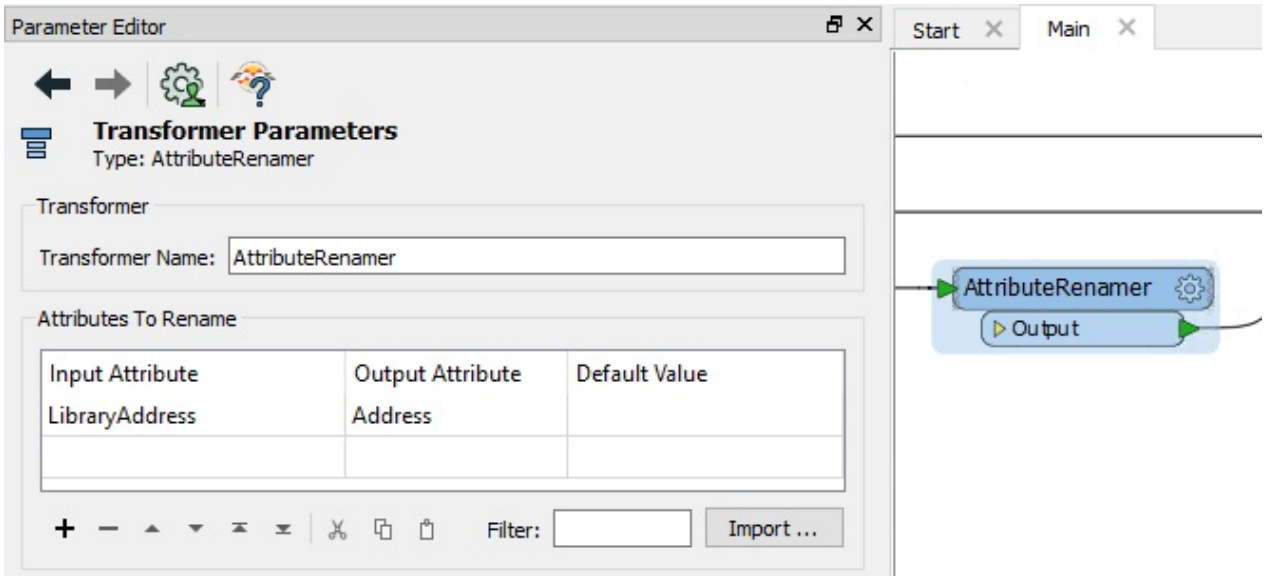
You can also open this search box by clicking on a blank spot of your canvas and beginning to type. This functionality is called Quick Add.

Transformer Parameters

Each transformer may have a number of parameters (settings). Parameters can be accessed (like feature types) by clicking the cogwheel icon:



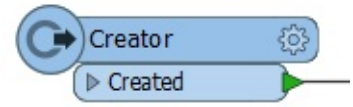
Alternatively, if the Parameter Editor window is open, parameters can be found there simply by clicking on the transformer (or any other canvas object):



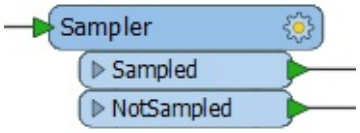
Color-Coded Parameter Buttons

The parameter button on a transformer is color-coded to reflect the status of the settings.

A blue parameter button indicates that the transformer parameters have been checked and amended as required and that the transformer is ready to use.



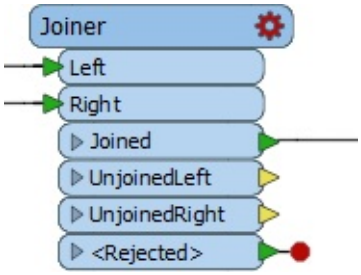
A yellow parameter button indicates that the default parameters have not yet been checked. The transformer can be used in this state, but the results may be unpredictable.



TIP

If the Parameter Editor window is open then you will rarely see a yellow icon, because a transformer's parameters are automatically opened and assumed to be reviewed. If that window is open, you should be sure to check it to ensure you don't miss setting a parameter that you need.

A red parameter button indicates that there is at least one parameter for which FME cannot supply a default value. The parameter must be provided with a value before using the transformer.



First-Officer Transformer says...

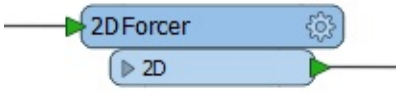
Good morning everyone, I'm First Officer Transformer and I'd like to welcome you aboard today's training.

Please be sure to check your parameters before your try to take off. Your workspaces just won't fly if there are any red-flagged transformers in them!

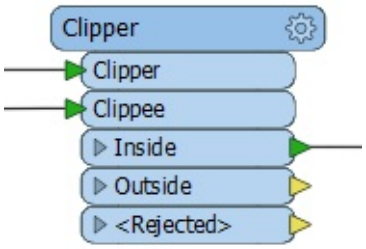
Transformer Ports

Far from having just a single input and output, a transformer can have multiple input ports, multiple output ports, or both.

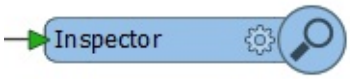
This 2DForcer transformer has a single input and output port.



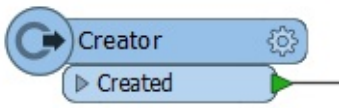
This Clipper has multiple input and output ports. Notice that not all of them are – or need to be – connected.



This Inspector has just a single input port...

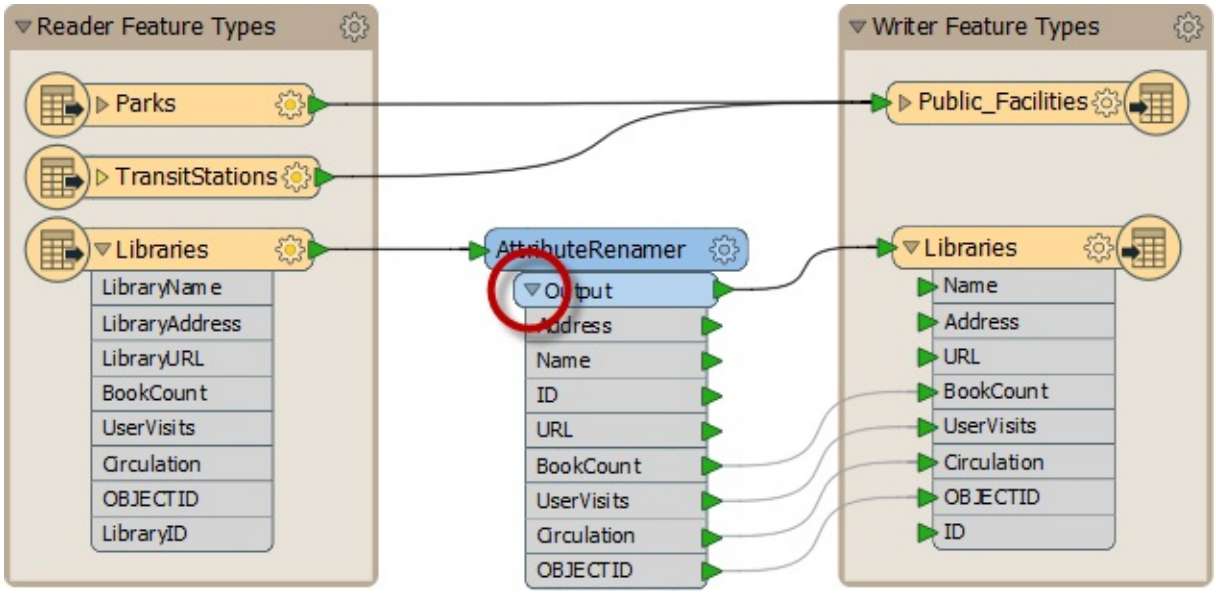


…whereas this Creator has only a single output port!



Transformer Attributes

Click on the drop-down arrow of a transformer output port to see all of the attributes that exit the transformer. This list includes all changes applied within the transformer.



This feature lets one visualize which attributes have been created, lost, or otherwise transformed within the transformer.

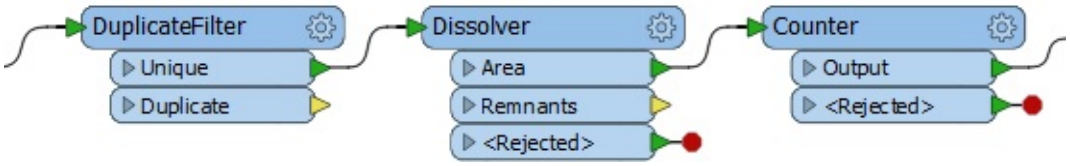
Transformers in Series

Much like a set of components in an electrical circuit, a series of Workbench transformers can be connected to have a cumulative effect on a set of features.

Chaining Transformers

Even with the large number of transformers available in FME, users frequently need a combination - or chain of transformers - instead of a single one.

A string of transformers that graphically represent an overall workflow is a key concept of FME:



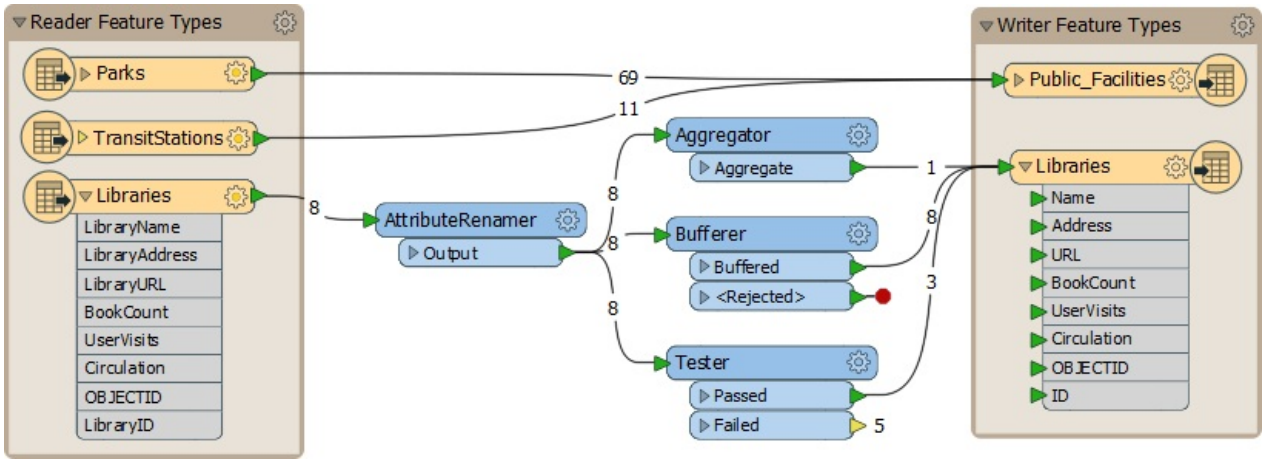
In this example, a DuplicateFilter transformer removes duplicate polygon features. A Dissolver transformer merges each remaining (unique) polygon with its neighbor where there is a common boundary. Finally, each merged area gains an ID number from the Counter transformer.

Transformers in Parallel

A ***stream*** is a flow of data represented by connections in the workspace. A key concept in FME is the ability to have multiple parallel streams within a workspace.

Multiple Streams

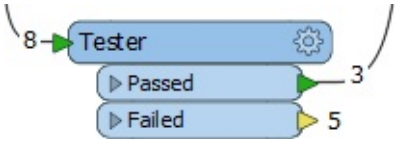
Multiple streams are useful when a user needs to process the same data but in many different ways. A workspace author can turn one stream into several, or combine several streams of data into one, as required:



Here an author is creating three data streams, each of which is processed separately then combined back into a single stream.

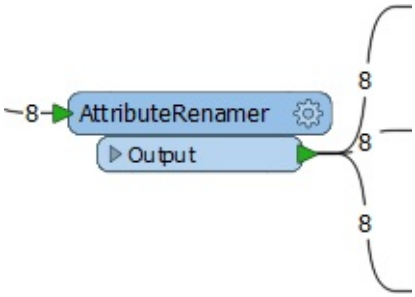
Creating Multiple Streams

Creating multiple data streams can occur in a number of ways. Sometimes a transformer with multiple output ports (a Tester transformer is a good illustration of this) will divide (or filter) data with several possible output streams:



Here data is divided into two streams, one of which is not connected to anything.

Additionally, a full stream of data can be duplicated by simply making multiple connections out of a single output port. In effect it creates a set of data for each connection:

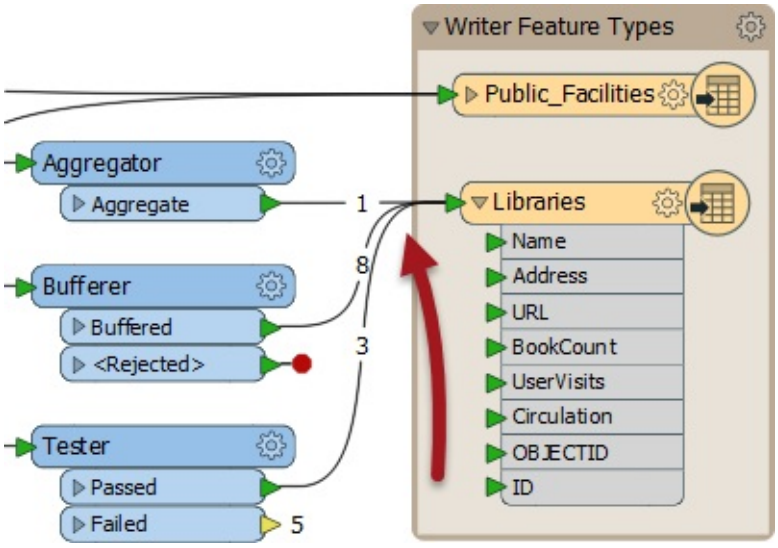


Here FME reads 8 features but, because there are multiple connections, creates multiple copies of the data:

Bringing Together Multiple Streams

When multiple streams are connected to the same input port no merging takes place. The data is simply accumulated into a single stream. This is often called a *union*.

Here, three streams of data converge into a writer feature type:

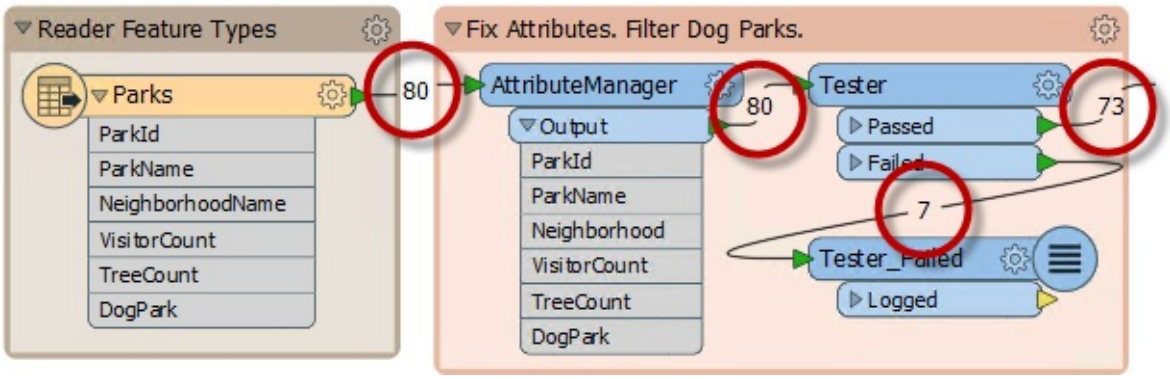


No merging takes place; the data simply accumulates into 12 distinct features in the output dataset. Think of it as three pipes of water emptying into a single contained.

To carry out actual merging of data requires a specific transformer such as the FeatureMerger or FeatureJoiner.

Feature Count Display

When you run a translation, each connection on your canvas is updated with the number of features that pass along it.



The final feature counts show that 80 features were read, 73 passed the Tester while seven failed (for being dog parks) and went on to the Logger.

The Log window confirms the number of features written and lists the features that failed the Tester.

These numbers help analyze the results of a workspace and provide a reference for debugging if the output differs from what was expected.

Exercise 7	Grounds Maintenance Project - Labelling Features
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Content transformation with parallel transformers
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex4-Begin.fmw
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex4-Complete.fmw C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex4-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

In this part of the project we'll create a label for each park and write it to a new output layer. This is best done using a parallel stream of data.

1) Start Workbench

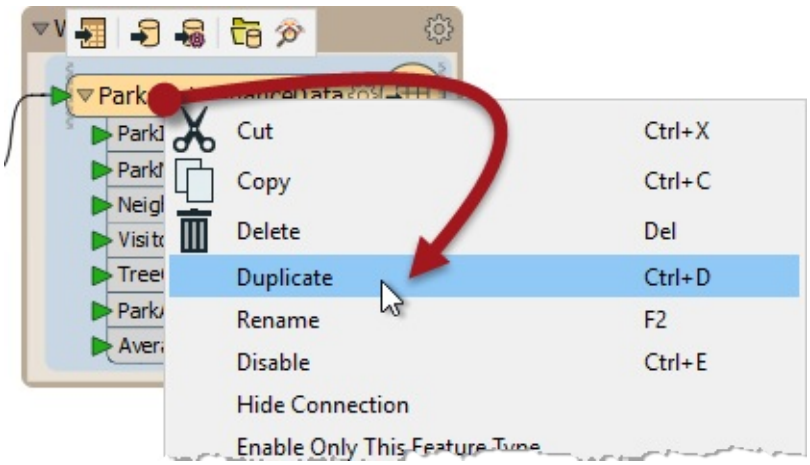
Start Workbench (if necessary) and open the workspace from Exercise 3. Alternatively you can open C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex4-Begin.fmw

The previous exercise measured park areas with the AreaCalculator. Now we are asked to add this information as labels to the output dataset.

This can be achieved using the LabelPointReplacer transformer.

2) Create New Writer Feature Type

Because we want to write label features to a separate layer (table) in the output, we need to create another feature type object on the canvas. There is more about this in a later chapter, but for now right-click the writer feature type and choose the option Duplicate. This creates a new feature type (layer) in the output dataset.

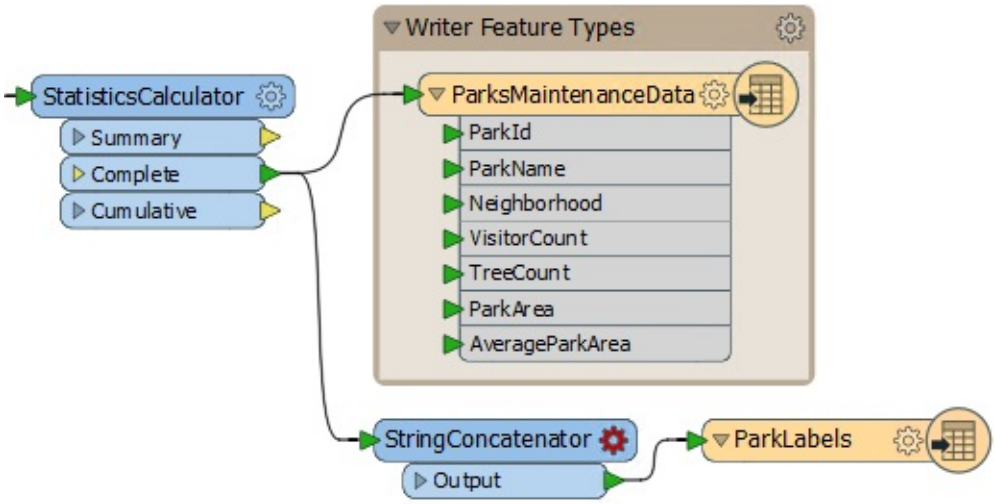


Now clean up this feature type's schema. View the feature type's dialog and rename the new type to ParkLabels. In the User Attributes tab delete all of the existing user attributes.

3) Place a StringConcatenator Transformer

Click onto a blank area of canvas. Type "StringConcatenator" to add a transformer of this type.

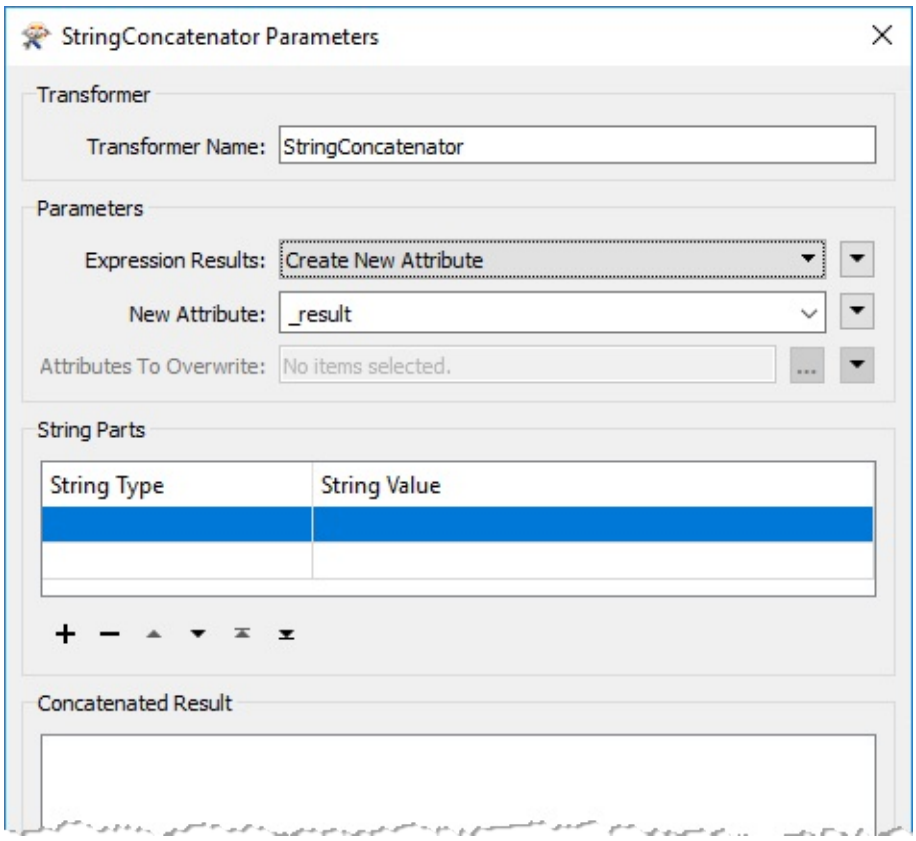
Connect it to the Complete port of the StatisticsCalculator by dragging a second connection from there to the new transformer.



Make a new connection from the StringConcatenator to the new feature type.

4) Check Transformer Parameters

View the parameters for the StringConcatenator transformer. There are both basic and advanced dialogs, and the basic one looks like this:

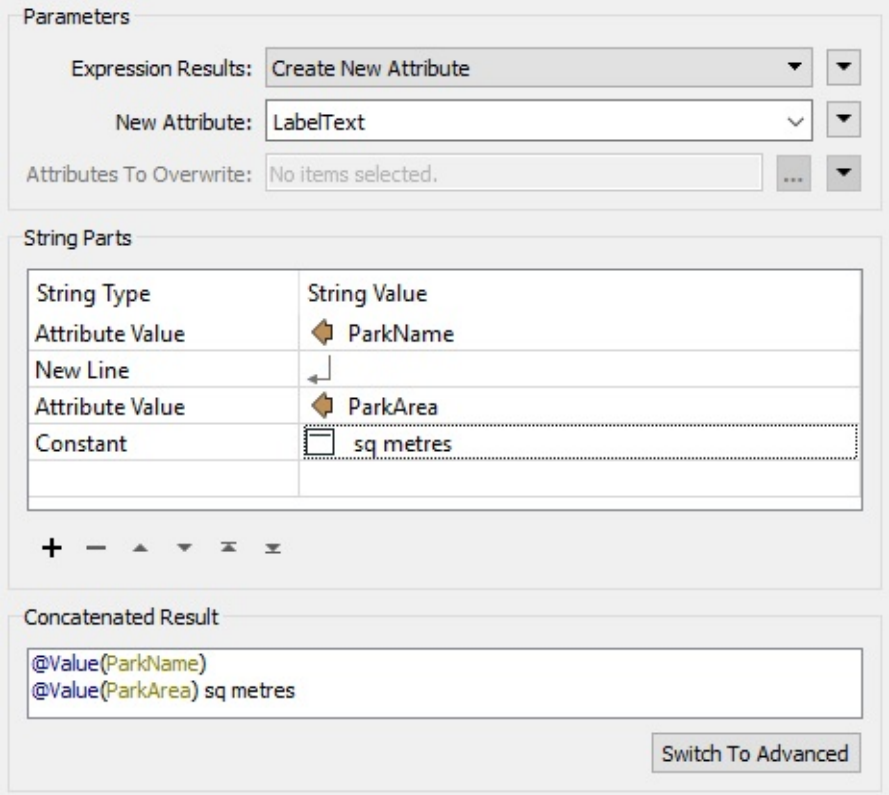


Enter *LabelText* as the name for the new attribute to create.

In the String Parts section, set the following four parts:

String Type	String Value
Attribute Value	ParkName
New Line	
Attribute Value	ParkArea
Constant	sq metres

Be sure to include a space character in the constant before "sq metres".



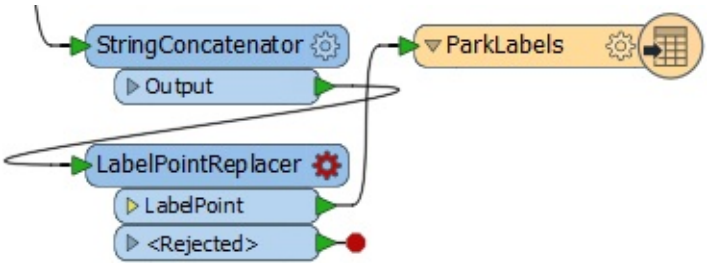
TIP

You may find it quicker to switch to the Advanced editor dialog and enter the content directly:
@Value(ParkName)
@Value(ParkArea) sq metres

5) Place a LabelPointReplacer Transformer

Click onto the connection between StringConcatenator:Output and the ParkLabels feature type. Type "LabelPointReplacer" to add a transformer of this type.

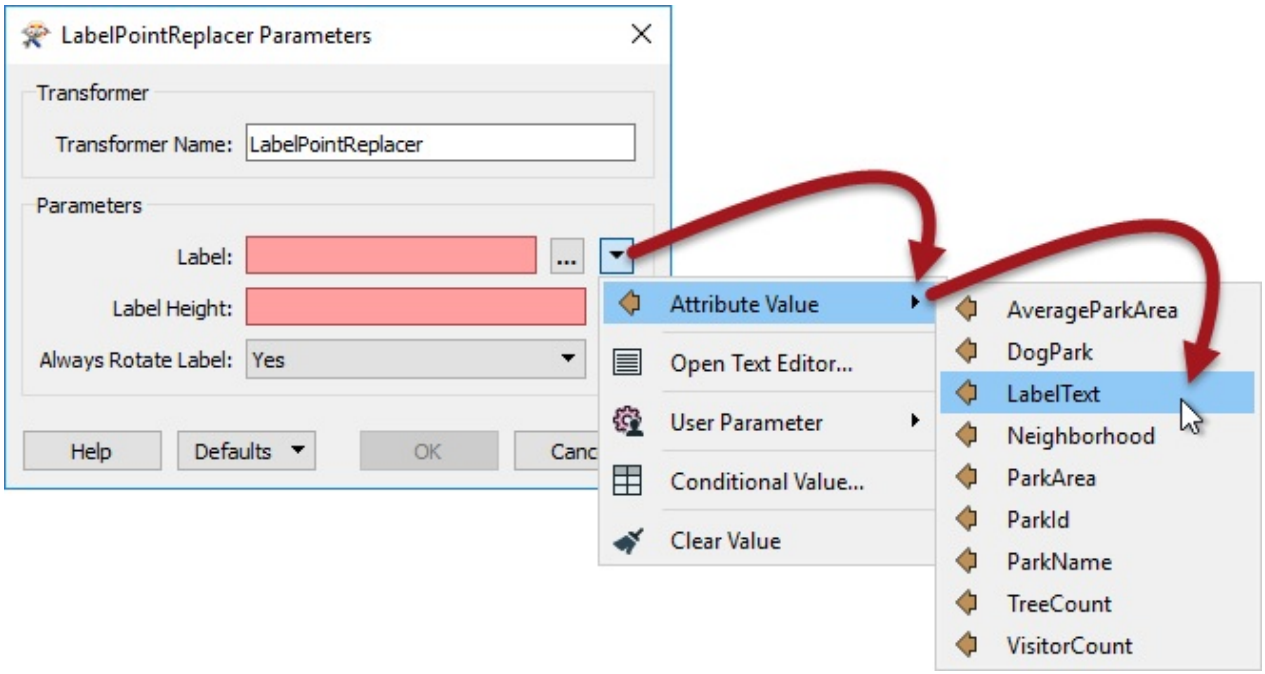
The new transformer will be added and automatically connected between those two objects.



6) Check LabelPointReplacer Parameters

Inspect the LabelPointReplacer parameters.

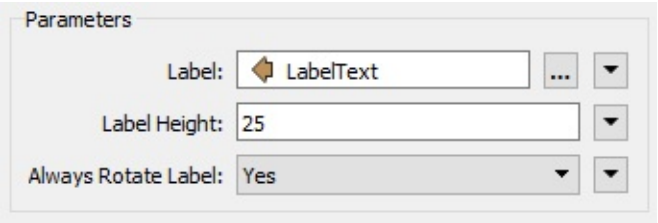
Firstly click the dropdown arrow to the right of the Label parameter:



Select Attribute Value > LabelText to select the label previously defined in the StringConcatenator.

Now click in the Label Height field and type 25 (that's 25 working units, which in this case is metres).

The "Always Rotate Label" parameter can be left to its default value.



TIP

Many parameter fields (like Label Height) can be set either as a constant value (by typing it in) or set to an attribute by clicking the drop down arrow and selecting Attribute Value.

And - as you'll see shortly - it's also possible to construct a parameter value directly inside the transformer settings

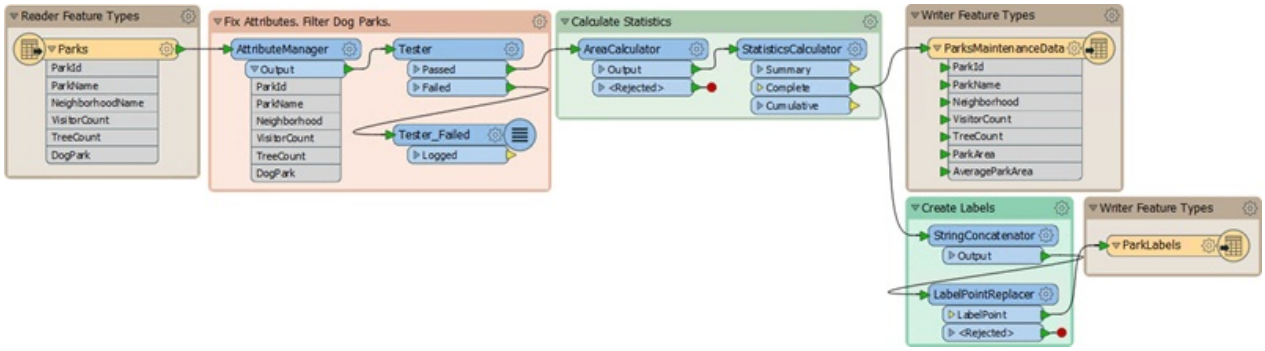
7) Run the Translation

Add another bookmark if you wish, run the translation, and inspect the output.

Notice that the output is in two layers in two files. Use the FME Data Inspector to open both output files in the same view.

Map tiles by Stamen Design, under CC-BY-3.0. Data by OpenStreetMap, under CC-BY-SA.

Save the workspace – it will be completed in further examples.



Advanced Exercise

Now you know how to create a new feature type (layer) in the output, how to test data, and how to use parallel streams, why not try this task: Identify which parks are smaller than average and which parks are larger than average, and write them out to different feature types.

CONGRATULATIONS

By completing this exercise you have learned how to:

- *Create a new writer feature type*
- *Use multiple streams of transformers in a single workspace*
- *Use the StringConcatenator to construct a string for use elsewhere*
- *Use an attribute as the value of a transformer's parameter*

Group-By Processing

Group-By parameters allow features to be processed in groups by a single FME transformer.

What is a Group?

FME transformers carry out transformations on either one feature at a time, or on a whole set of features at once.

For example, the *AreaCalculator* transformer operates on one feature at a time (to measure the area of a single polygon feature). We call it a ***feature-based transformer***.

The *StatisticsCalculator* operates on multiple features at a time (to calculate an average value for them all). In FME we call this set of features a ***group*** and the transformer is a ***group-based transformer***.

Creating Groups

So a group is simply a defined set of features being processed by a transformer. By default, a group-based transformer treats ALL the features that it is fed as a single group.

However, such transformers also have a ***Group-By*** parameter. This parameter lets the user define several groups based upon the value of an attribute.

Mr. Statistics-Calculator, CFO says...

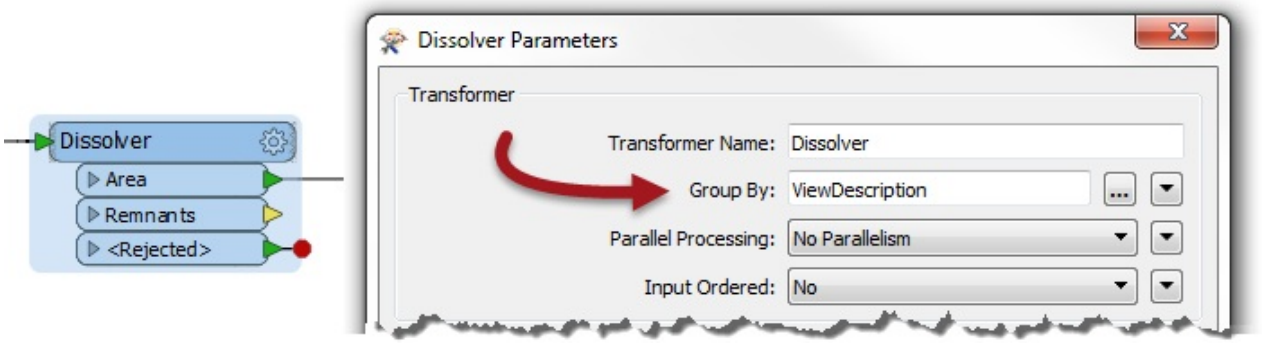
Hi. I don't think we've met yet. I'm Mr. Statistics-Calculator. I bet you can't guess my favourite transformer!

To illustrate groups let's consider calculating the mean age of FME users. Don't worry - I'll be discrete (ha ha)! The default group for the calculation includes ***all*** *FME users.*

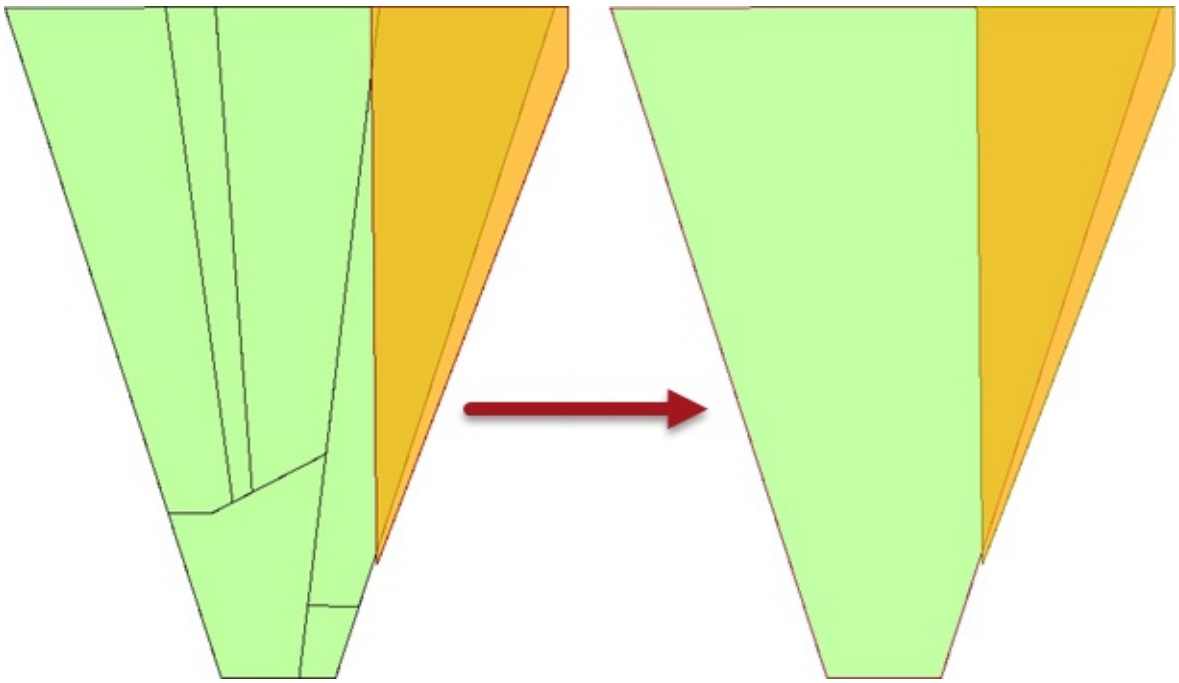
But you could instead divide everyone up on into men and women, creating two groups, and calculate average age per gender. Or you could divide everyone into their nationality, and calculate average age per country.

This is the same as having a gender (or nationality) attribute in a dataset and selecting that in an FME group-by parameter.

Here, a Dissolver transformer is being used to dissolve (merge) a number of polygon features. The selected Group-By attribute is *ViewDescription*:



FME creates a series of groups for overlaying, where the features in each group share the same value for the *ViewDescription* attribute. The practical outcome is that polygon dissolving takes place only where line features share the same description:



Miss Vector says...

Let's see if you've picked up the idea of what a group-based transformation is.

Which of the following transformers do you think is "group-based"? Feel free to use Workbench to help you answer this question.

1. StringFormatter
2. Clipper
3. Rotator
4. AttributeRounder

Exercise 8	Grounds Maintenance Project - Neighborhood Averages
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Group-By Processing
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex5-Begin.fmw
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex5-Complete.fmw

Let's continue your work on the grounds maintenance project.

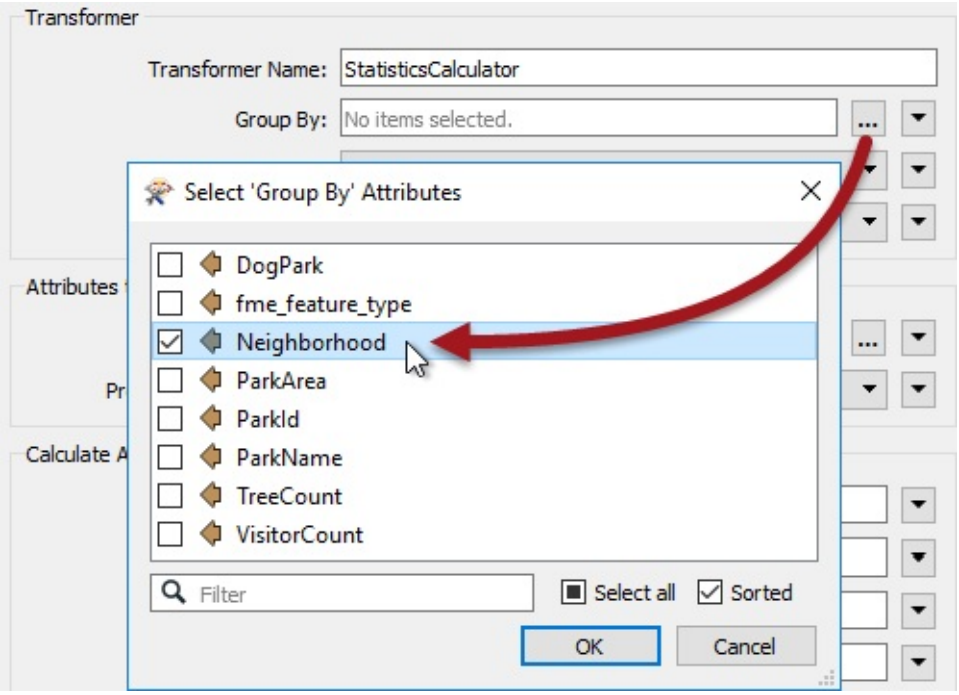
The parks team has decided that they do not want the average area of park for the city as a whole. Instead they want the average size of park in each neighborhood; so let's do that for them.

1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 4. Alternatively you can open C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex5-Begin.fmw

2) Set Group-By in StatisticsCalculator

This is a really simple task to do. View the parameters for the StatisticsCalculator transformer and click the 'browse' button next to the Group By parameter. Select the attribute called Neighborhood:



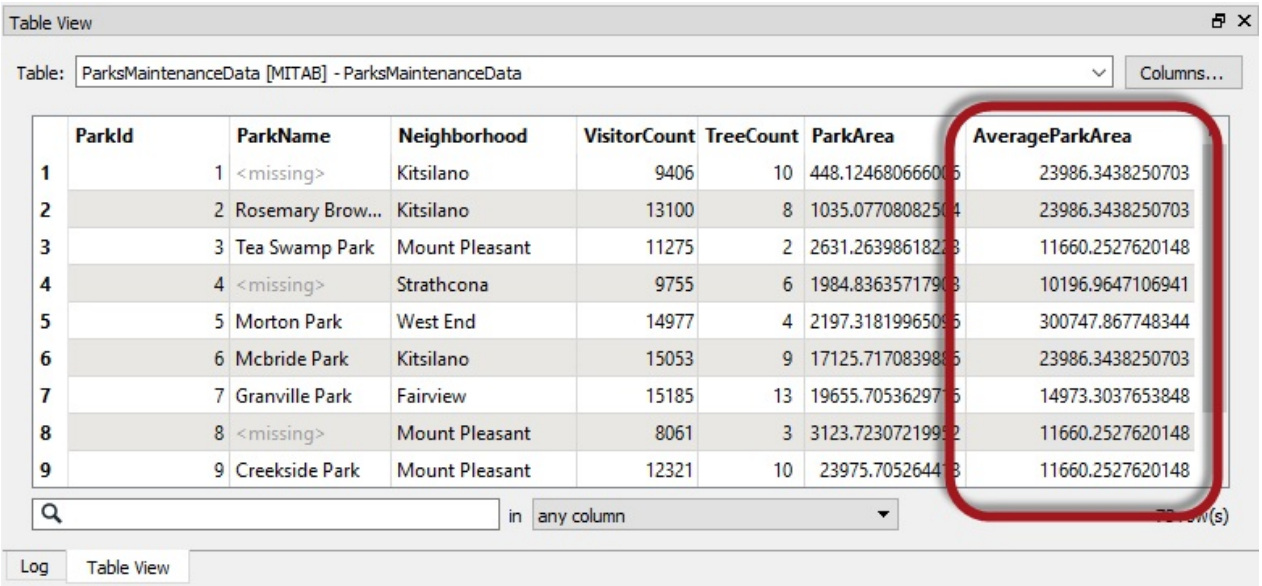
Click OK/Apply to apply the changes to the transformer.

3) Run the Workspace

Save and then run the workspace.

Inspect the output data in the Table View window of the FME Data Inspector.

You should see that each neighborhood now has its own value for AverageParkArea:



CONGRATULATIONS

By completing this exercise you have learned how to:

- *Use the group-by parameter in FME transformers*

Coordinate System Transformation

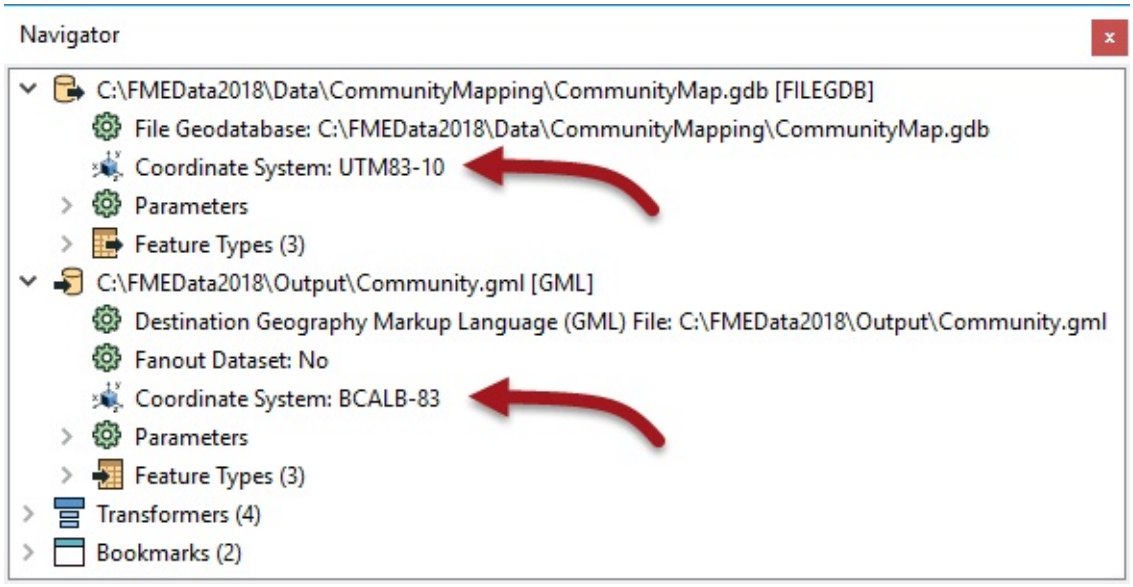
To be located in a particular space on the Earth's surface, the majority of spatial data is related to a particular spatial reference.

Some users call this location of data a "projection," but projection is just one component of what we call a **coordinate system**. A coordinate system includes *projection*, *datum*, *ellipsoid*, *units*, and sometimes a *quadrant*.

Coordinate System Settings

Each reader and writer within FME can be assigned a coordinate system. That coordinate system is set in the Navigator window of Workbench, or in the Generate Workspace dialog.

Like the source schema, the reader coordinate system is ***"what we have"*** and the writer coordinate system is ***"what we want"***. Here the source coordinate system has been defined as UTM83-10 and the destination as BCALB-83:



Each feature processed by the reader is tagged with the coordinate system defined in its parameter.

When a feature arrives at a writer, if it is tagged with a different coordinate system to what is defined for that writer, then FME automatically reprojects the data, so that the output is in the correct location.

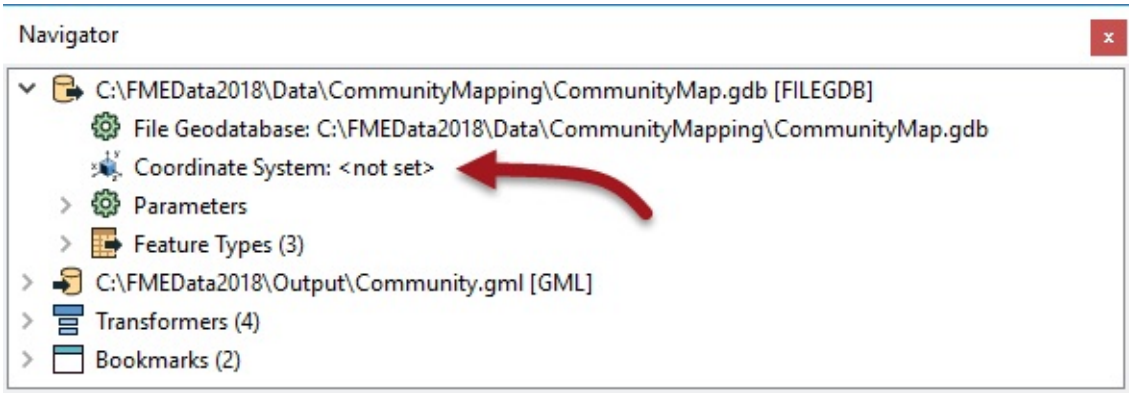
TIP

Once tagged with a coordinate system, each feature retains this throughout the translation; FME knows what coordinate system it belongs to at all times.

This is important when carrying out geometric transformations (like calculating area) or when reading multiple datasets that belong to different coordinate systems (yes, FME will handle that).

Automatic Detection of Coordinate Systems

It's not always necessary to set the coordinate system parameters manually. Some data formats (for example Esri Shapefile) are capable of storing information about the coordinate system in which they are held, and FME will retrieve this information where it can.



Here, because the reader coordinate system is marked <not set>, FME will try to determine the coordinate system from the source dataset. If it can't, then the feature will be tagged with a coordinate system of <unknown>.

There are a number of reprojection scenarios that may occur depending on the combination of coordinate system (CS) information available:

Dataset CS	Reader CS	Writer CS	Reprojection
N	Y	Y	Reprojects from Reader CS to Writer CS
Y	N	Y	Reprojects from Dataset CS to Writer CS
N	N	Y	Error: Cannot reproject without Dataset or Reader CS
Y	Y	Y	Reprojects from Reader CS to Writer CS

If the coordinate system is not set on the writer, then no reprojection will take place unless the output format requires it. For example, the KML format requires data to be in Latitude/Longitude. If neither the source dataset or the reader coordinate system is defined, then the translation will fail.

Exercise 9	Grounds Maintenance Project - Data Reprojection
Data	City Parks (MapInfo TAB)
Overall Goal	Calculate the size and average size of each park in the city, to use in Grounds Maintenance estimates for grass cutting, hedge trimming, etc.
Demonstrates	Data reprojection
Start Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex6-Begin.fmw
End Workspace	C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex6-Complete.fmw C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex6-Complete-Advanced.fmw

Let's continue your work on the grounds maintenance project.

The parks team has decided that the output data should be in an Albers Equal Area projection (coordinate system = BCALB-83). They think it will take ages to set this up! We'll show them differently...

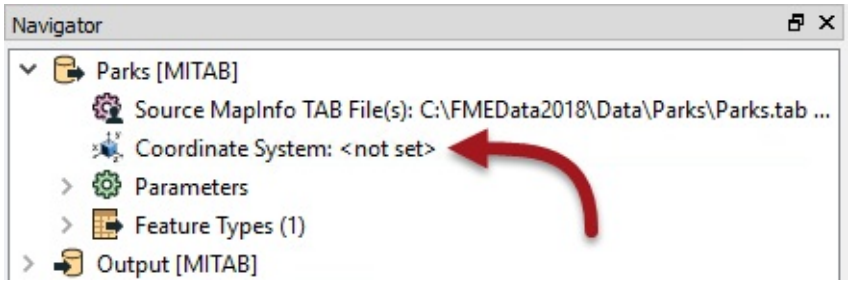
1) Start Workbench

Start Workbench (if necessary) and open the workspace from Exercise 5. Alternatively you can open C:\FMEData2018\Workspaces\DesktopBasic\Transformation-Ex6-Begin.fmw

2) Edit Reader Coordinate System

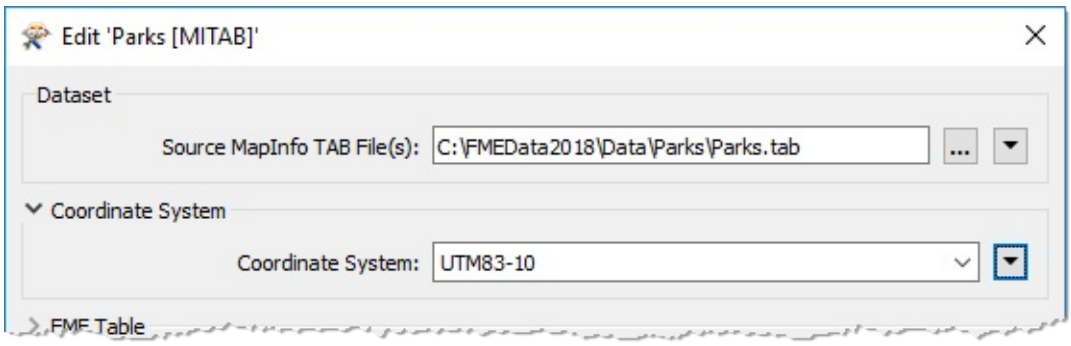
In the Navigator window locate the Parks [MITAB] reader, and expand its list of settings.

Locate the setting labelled ‘Coordinate System’. The original value should be <not set>:



Double-click the reader Coordinate System parameter to open an edit dialog.

In the Coordinate System field enter the name UTM83-10 or select it from the Coordinate System Gallery by selecting "More Coordinate Systems..." from the bottom of the drop-down list:



TIP

Remember, when a reader's Coordinate System parameter is defined as <not set> FME will automatically try to determine the correct coordinate system from the dataset itself.

When the source dataset is in a format that stores coordinate system information (as it does in this example) you can safely leave the parameter unset. So this step isn't really necessary.

However, you MUST set this parameter when you wish to reproject source data that does not store coordinate system information; otherwise an error will occur in the translation.

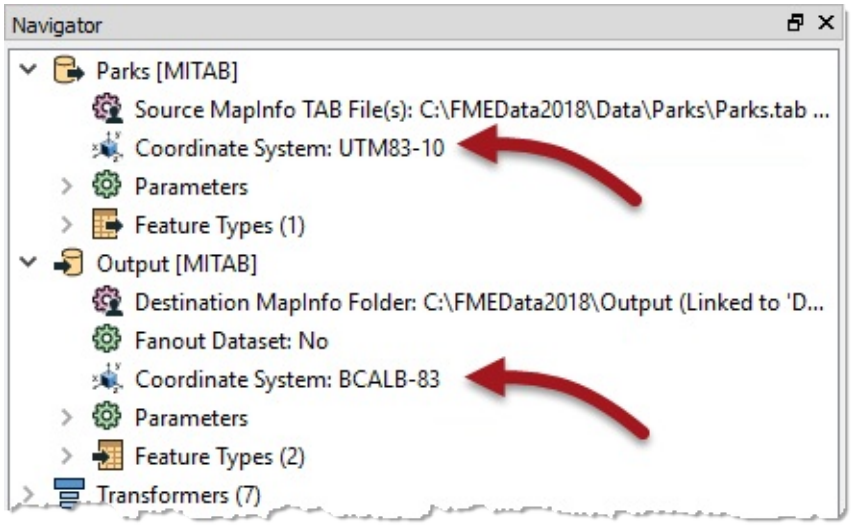
3) Edit Destination Coordinate System

Now locate the coordinate system setting for the destination (writer) dataset.

Again the current value should be the default of <not set>.

Double-click the parameter and enter the coordinate system name BCALB-83 or select it from the Coordinate System Gallery by selecting "More Coordinate Systems..." from the bottom of the drop-down list.

The Navigator window will now look like this:



4) Run the Workspace

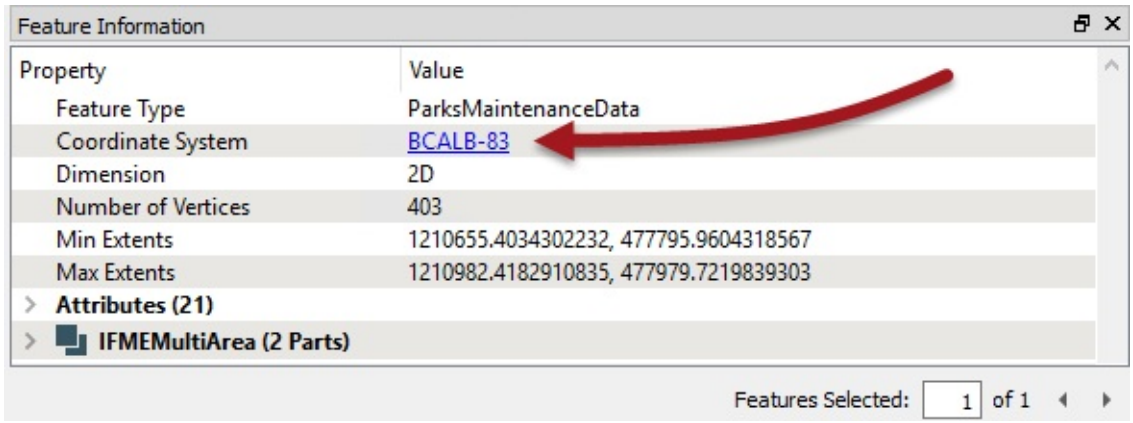
Save and then run the workspace.

In the log file you should be able to find:

```
FME Configuration: Source coordinate system for reader
MITAB_1[MITAB] set to `UTM83-10'
FME Configuration: Destination coordinate system set to `BCALB-
83'
```

5) Inspect the Output

Open the newly reprojected dataset and query a feature. The Feature Information window should report that the data is now in BCALB-83. Optionally, click on the coordinate system name in that window; a new dialog will open to display all of the coordinate system parameters.



TIP

If the background map is activated when a dataset is opened then the contents of that dataset are automatically reprojected to Spherical Mercator to match the background map. If you wish to see the data as it appears in its own coordinate system, then use Tools > FME Options to turn off background maps ***before*** *opening the source dataset.*

Advanced Exercise

Instead of using the reader/writer parameters in the Navigator window, why not try this exercise using the Reprojector (or CSMapReprojector) transformer? Where should the transformer be placed in the workspace and why is this important?

CONGRATULATIONS

By completing this exercise you have learned how to:

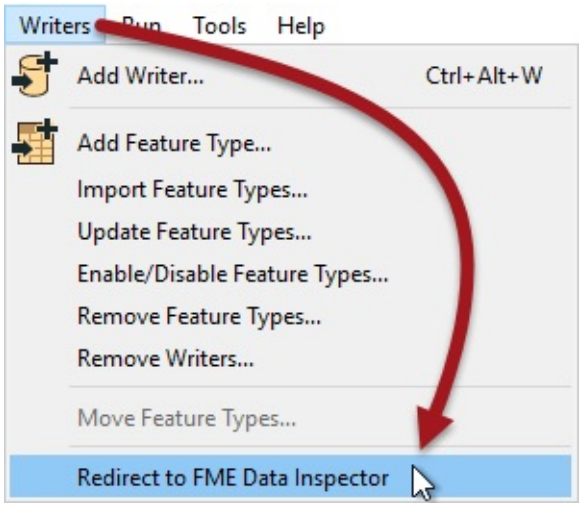
- *Use Coordinate System parameters to reproject spatial data*
- *Query features in the Data Inspector to inspect coordinate system information*

Integrated Inspection

When developing and testing a workspace, it's necessary to inspect the data being produced on a regular basis. There are various methods to send data directly to a data inspection window.

Redirect to Inspector

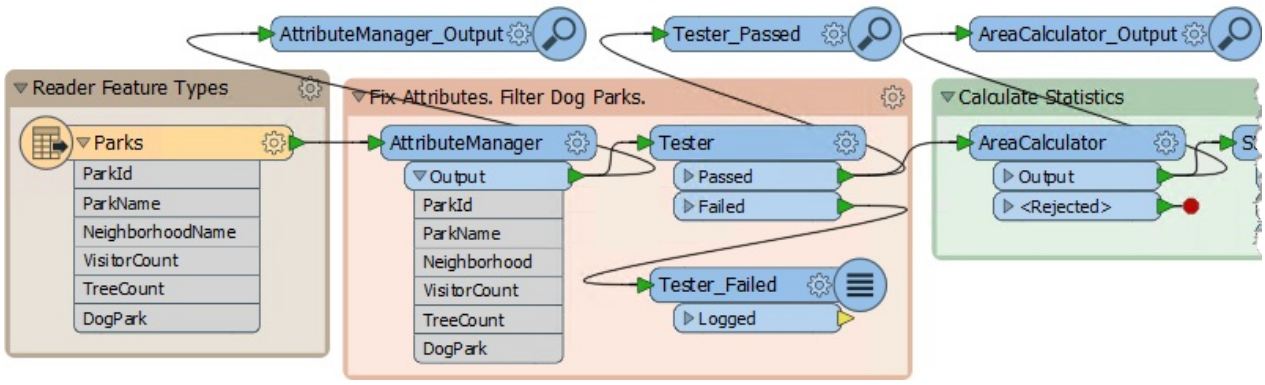
The simplest method to see output before it is written is to use the Redirect to FME Data Inspector option:



In this scenario all data is sent to the Data Inspector instead of being written; no output is produced, and the data can be checked for flaws before writing.

Inspector Transformers

In most cases it's necessary to inspect data at certain points in a workspace - not at the very end - and this can be done with Inspector transformers:

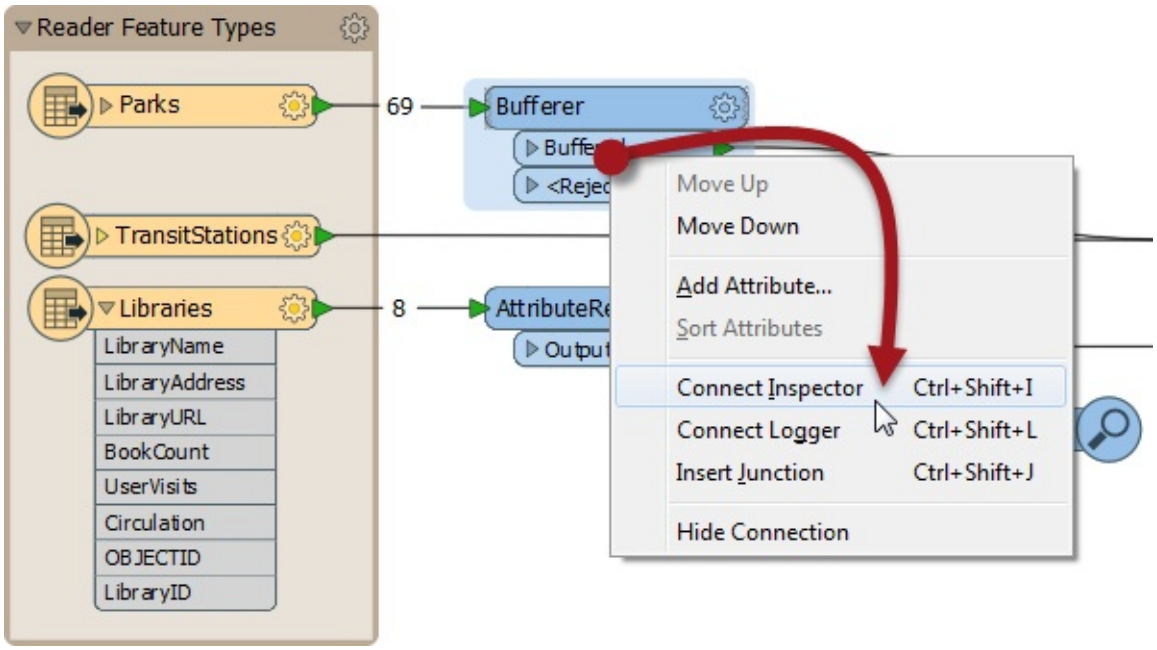


For example here the author has attached Inspectors to three different transformers.

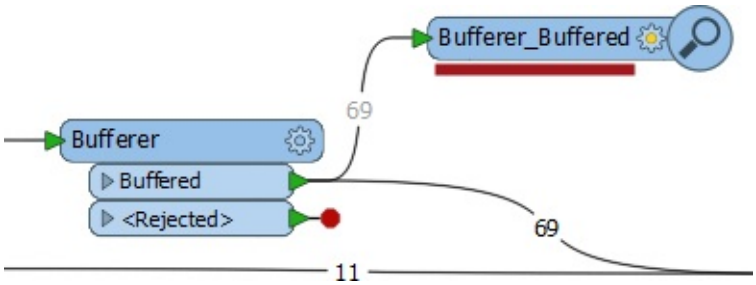
Placing an Inspector Transformer

The best, and simplest way to apply an Inspector is to right-click the output port of an object in a workspace and select the Connect Inspector option.

Here the user right-clicks the Buffered port of a Bufferer transformer and chooses the option Connect Inspector:



Notice that an Inspector is named automatically using the transformer and output port names. Here it will be "Bufferer_Buffered":



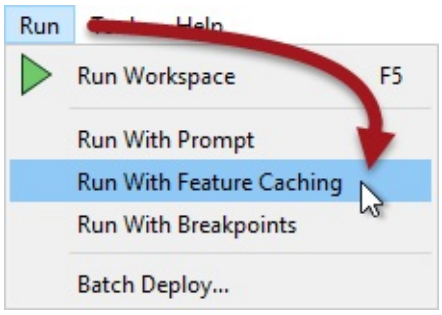
This helps to identify the data from this Inspector (as opposed to any others) in the FME Data Inspector.

Note that the Inspector transformer only opens the FME Data Inspector when there are features to view. If there are zero features, then the Data Inspector will not open!

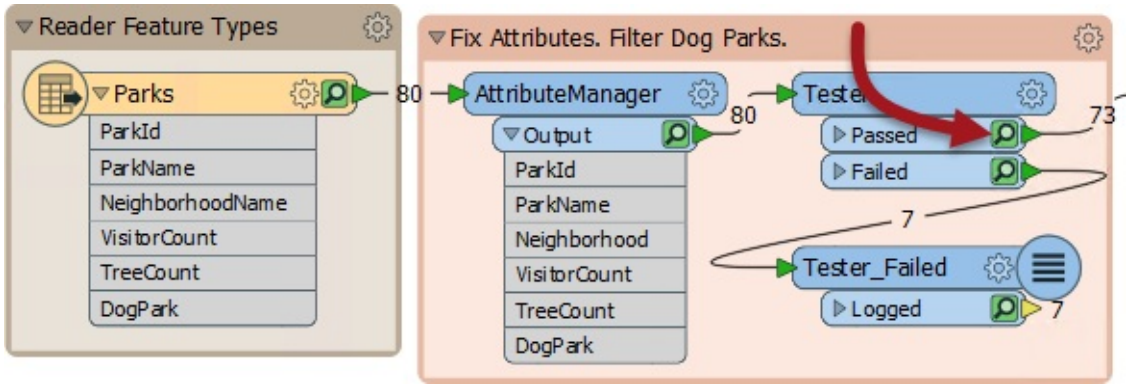
Feature Caching

Sometimes it's important to be able to inspect data at any step of the translation. Adding an Inspector transformer at every step would be tiresome, so instead FME has an option to cache data automatically.

This behavior is activated using Run > Run with Feature Caching on the menubar:



With this option active, FME generates caches at every step of the translation when the workspace is run:



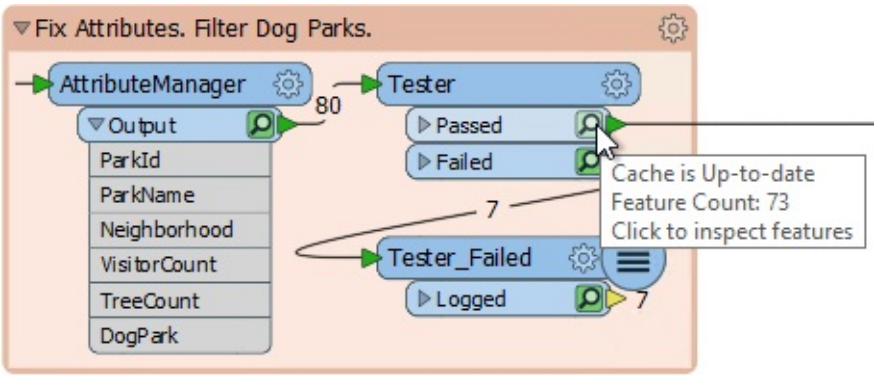
The caches are indicated by the small icons on each object. In the above screenshot, the caches are green, but they can change to yellow or red depending on how fresh the data is.

NEW

Run with Data Caching is essentially the same as Run with Full Inspection in prior versions of FME. It has been renamed in FME2018 to better match new functionality that takes advantage of these caches.

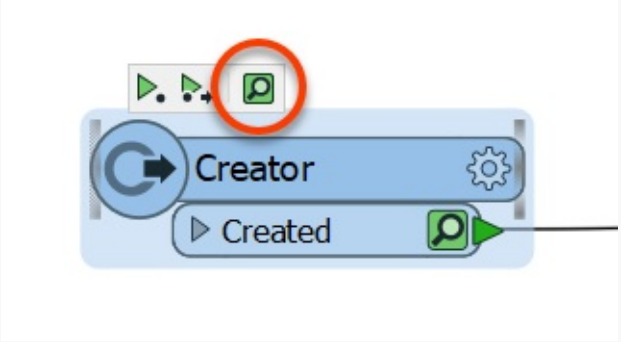
Inspecting Cached Data

Cached data can be inspected by clicking on the icon on a particular object.



2018.1 UPDATE

In 2018.1 you can immediately inspect cached data with a new "Inspect cached features" button displayed in the pop-up icons above objects on the canvas:



TIP

It's certainly quicker to set up "Run with Data Caching" than to manually add Inspector transformers. However, be aware that caching data obviously causes the translation to be slower, and to use system resources such as disk space.

Data caching is very useful while developing a workspace, but should be turned off before putting a workspace into production.

Partial Runs

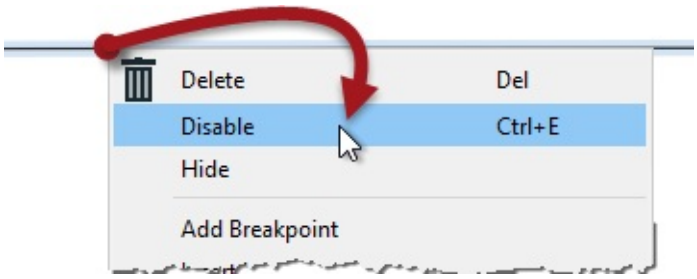
A partial run is when only one section of a workspace is executed. One way to do this is to disable objects in the canvas to only run certain enabled sections. Another method is to use a tool called Partial Runs, which is represented by pop-up options when a workspace is run with caching turned on.

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!

Disabled Objects

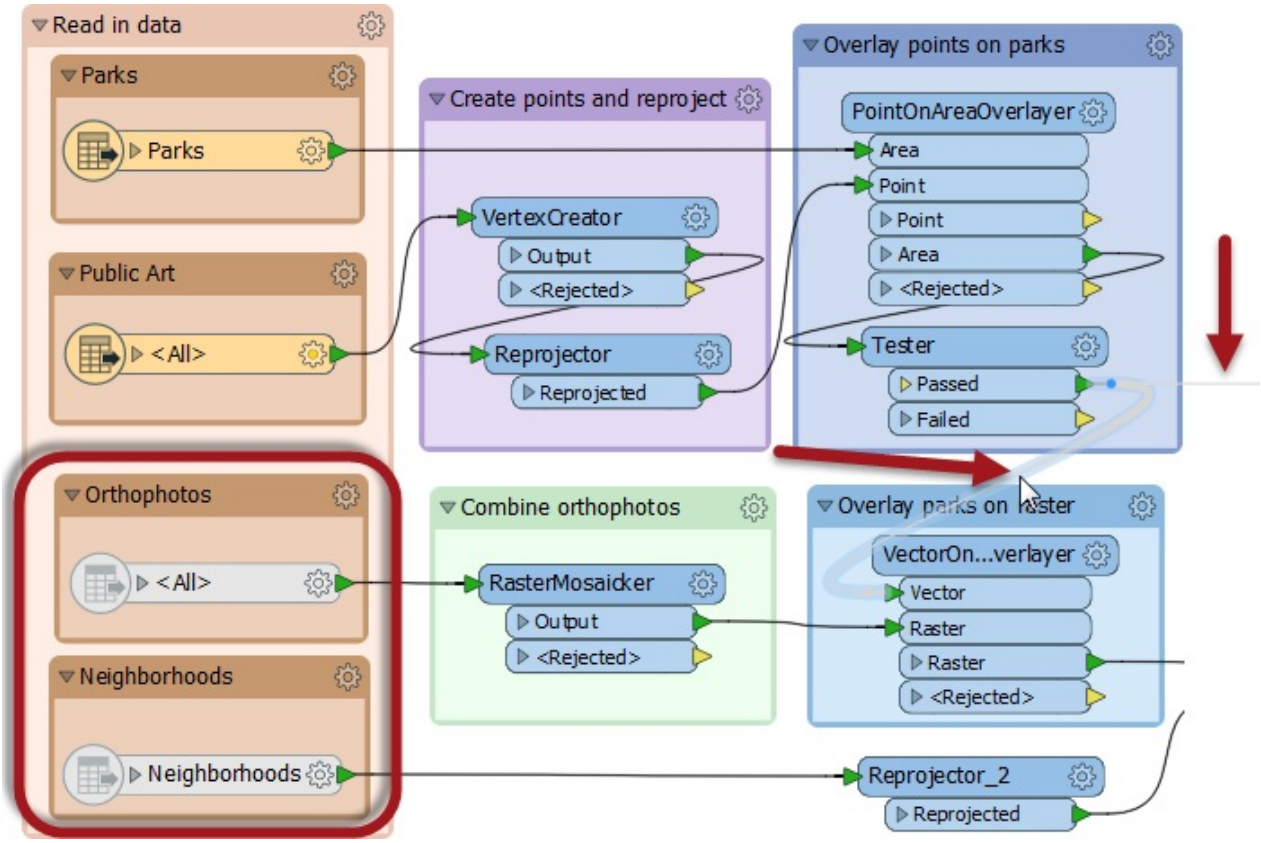
If designed correctly, a large workspace should be made up of small sections. Isolating a section (or part of a section) for testing is possible by disabling connections to all other components.

An object (connection 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:



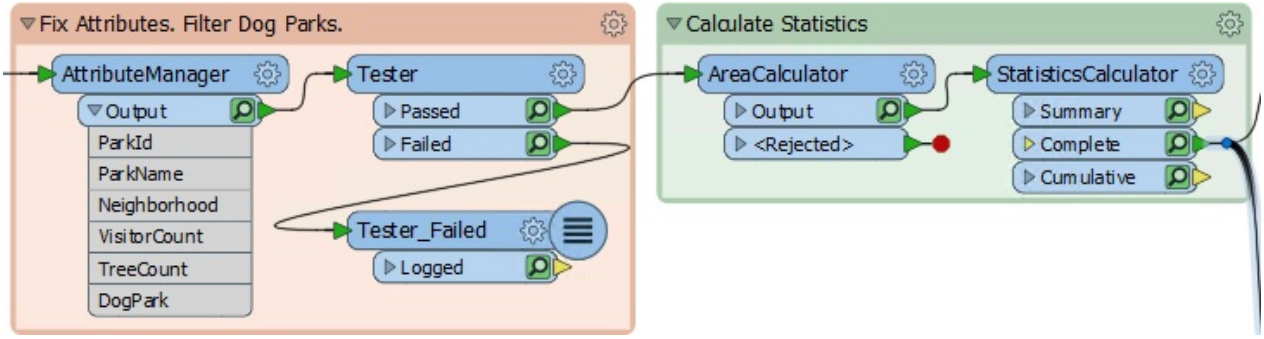
With that setup the top part of the workspace will operate up until (and including) the Tester. The bottom portion will not run at all. No data will emerge from the disabled feature types, and no data is passed to it from the Tester.

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.

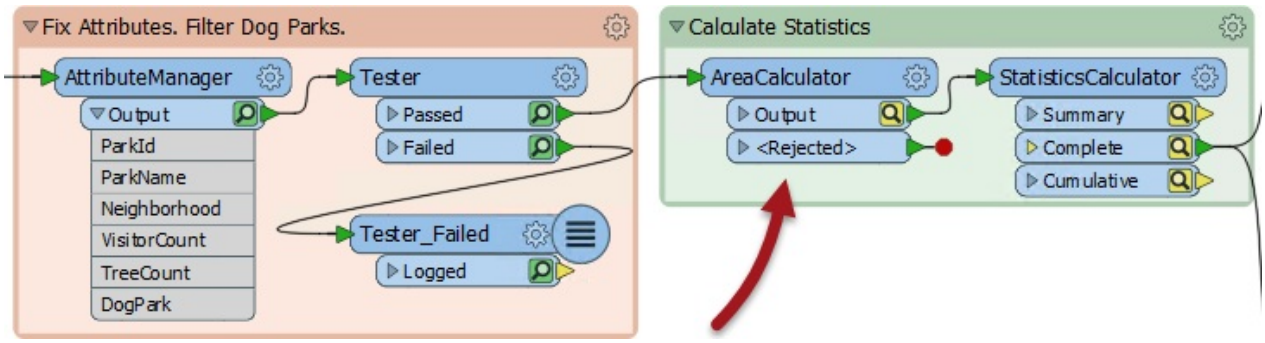
Partial Runs

When caching is turned on, running a translation causes data to be cached at every part of the workspace. In subsequent runs, those caches can be used instead of having to re-run entire sections of the workspace.

Here, for example, a workspace has been run with caching turned on:

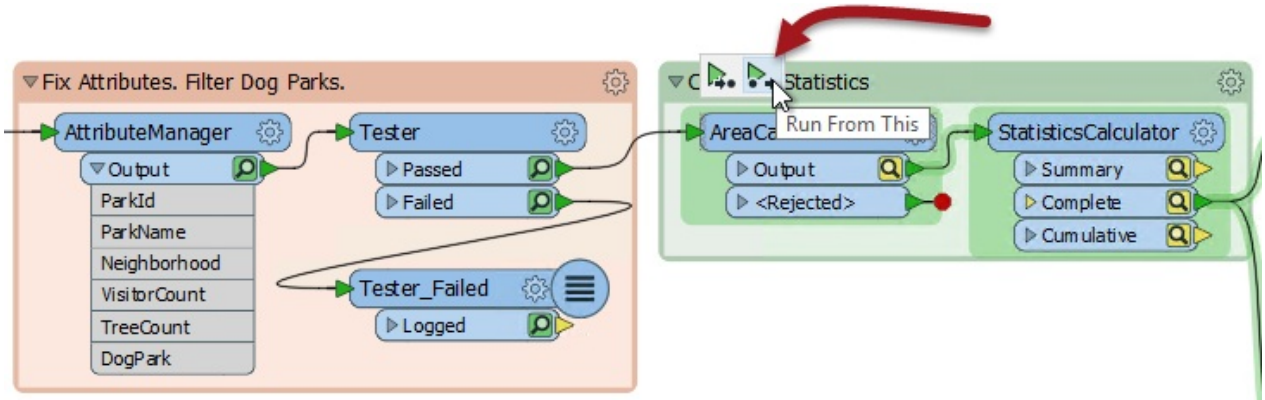


Now the author makes a change to the AreaCalculator parameters:



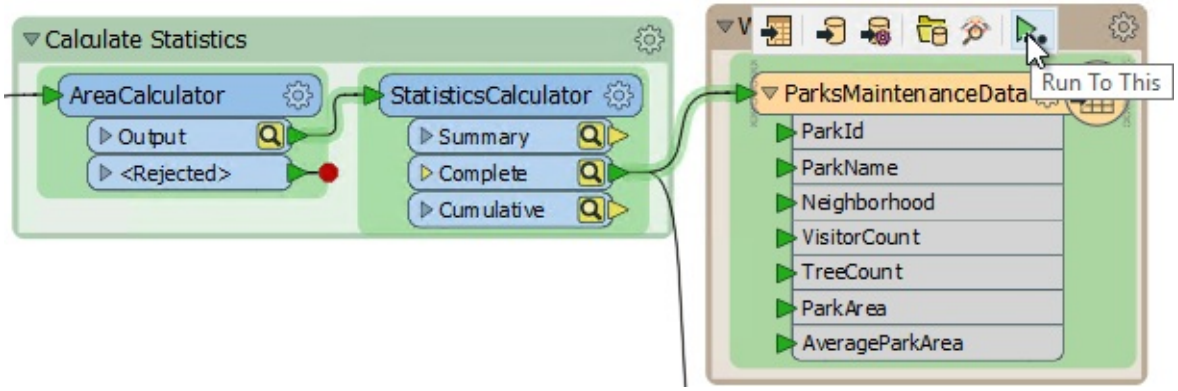
Notice that the caches change color (to yellow) on the AreaCalculator and subsequent transformers. This color denotes that caches are stale; their data contents no longer match what the workspace would produce.

To get the new results, the author must re-run the workspace. However, they do not have to re-run the entire workspace; they can start the workspace at the point of change - the AreaCalculator:



Run From This causes the workspace to run from that point only, using data cached up until that point. Notice how hovering over the option causes all "downstream" transformers to be highlighted. They are the only ones that will be run. That makes the translation quicker.

The other option is *Run To This*. The author could use that option on the writer feature type and get much the same effect:



...but notice how the second branch from the StatisticsCalculator does not get highlighted. It will not be run. That shows how you can avoid running a particular section of workspace, in much the same way as if that connection had been disabled.

TIP

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.

Also, caches can be saved with the workspace, when it is saved as a template. That means the workspace can be re-run using the caches from a previous session or even from another author!

WARNING

Partial runs are not compatible with variables (VariableSetter/Retriever transformers).

Style

> *“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 just looks; a properly designed workspace provides many benefits as it is further developed and edited in the future.

An FME Workspace Style Guide

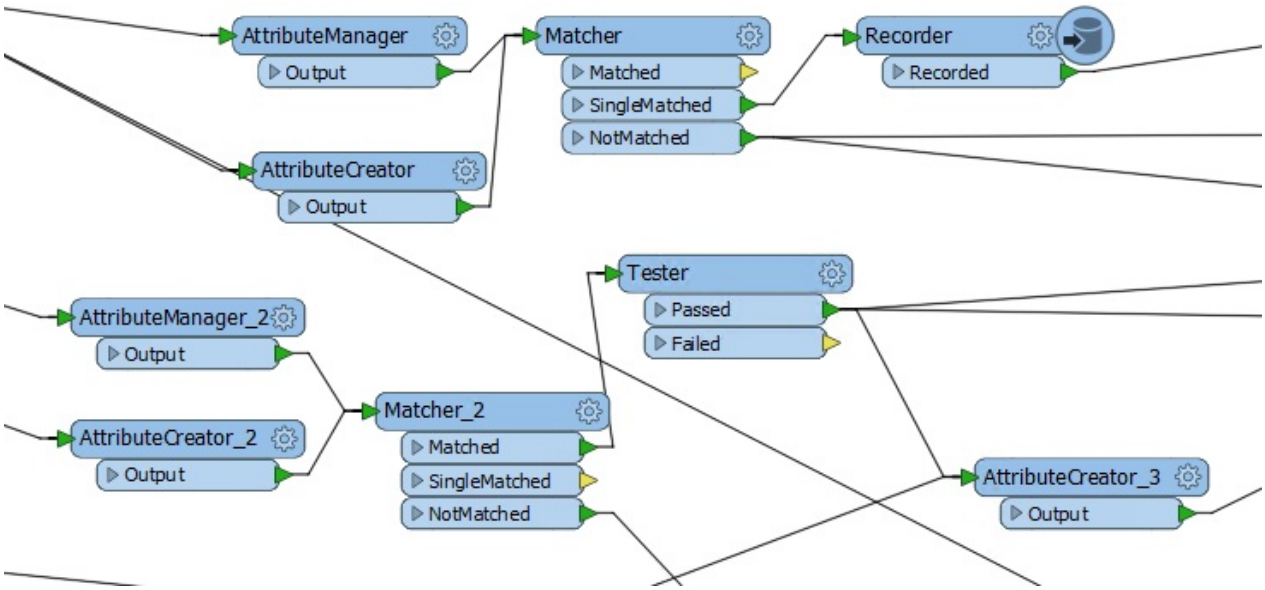
A good style of design makes it easier to navigate and understand an existing workspace. This is important when workspaces might need to be edited by other users, or when you intend to make edits yourself at a later date.

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

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?



Aunt Interop says...

I'm Aunt Interop. I write a blog helping out users with their various FME problems.

As I always say, size doesn't matter! You should always use Best Practice, whether it's a small workspace or 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 just become harder and harder to make edits as you work on it.

Annotating Workspaces

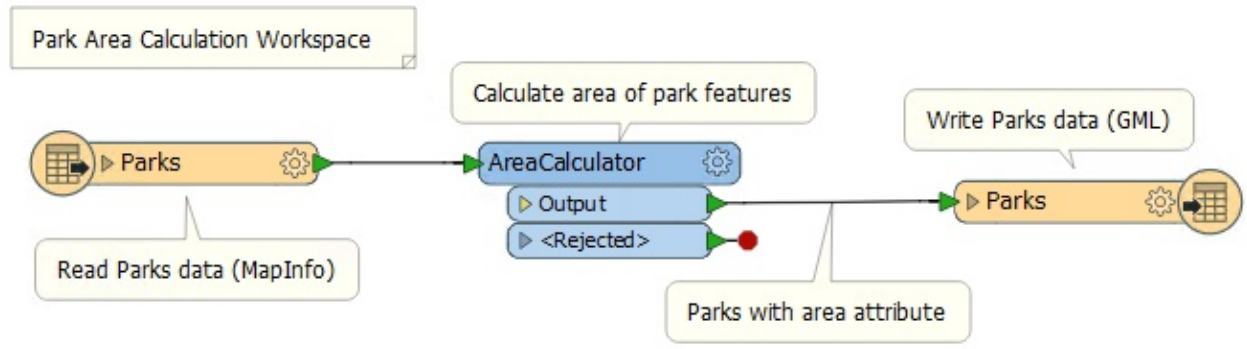
Annotation is a key method for a clear and comprehensible design.

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

Two different types of annotation can be applied to a workspace.

User Annotation

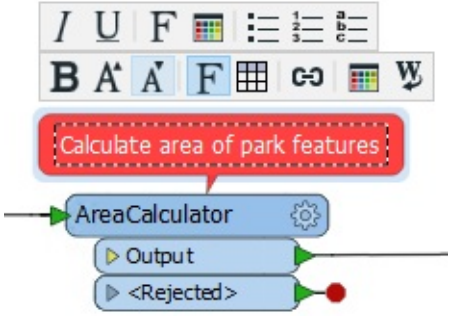
User annotation is a comment created by the user. It can be connected to a workspace object (transformer or feature type), can be connected to a workspace connection, or can float freely within the workspace.



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 and 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.



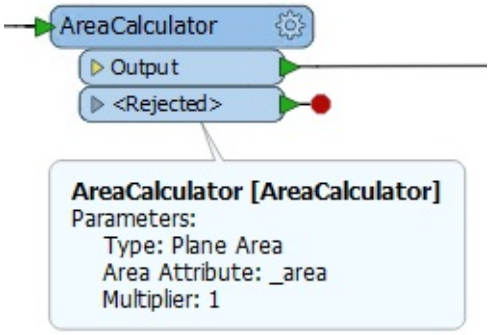
NEW

In FME2018, user annotation can now be attached directly to a bookmark.

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.

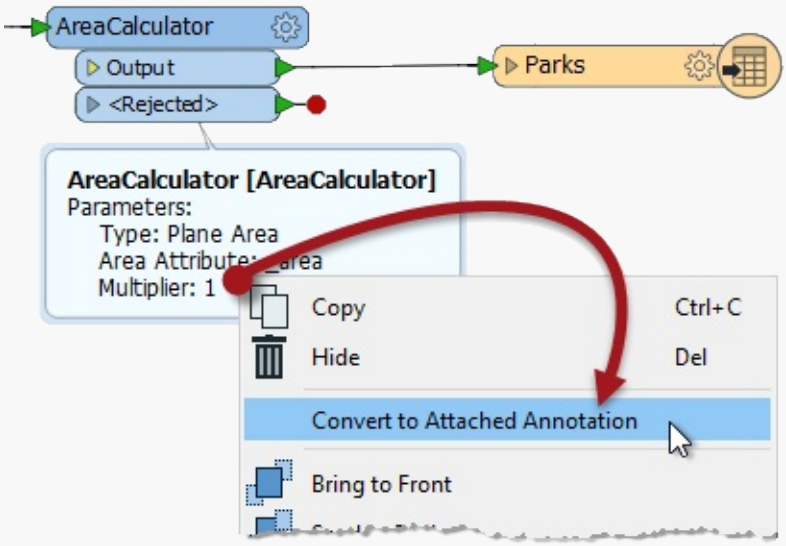


The nice thing about Summary Annotation is that it automatically updates in response to changes. That makes it very useful for checking transformer parameters (or reader/writer schemas) at a quick glance. It's particularly useful in situations where the parameters are set through a wizard and are more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

TIP

*A good idea is to use summary annotation to show **what** actions are taking place; but use user annotation to clarify **why** an action is being carried out.*

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



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

Bookmarks

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

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

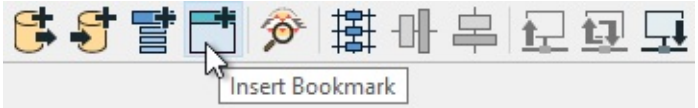
Why use Bookmarks?

Bookmarks play an important role in a well-styled workspace for a number of 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 section 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

Adding a Bookmark

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



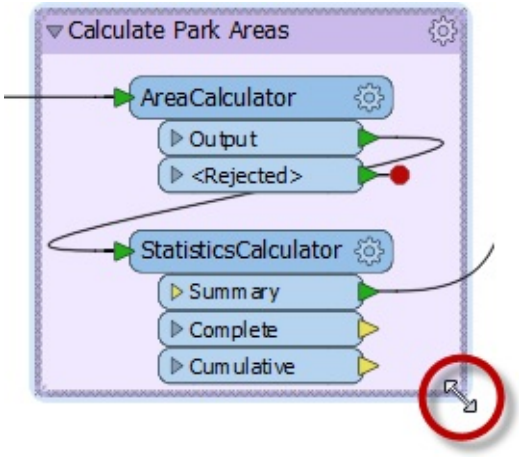
Whereas a traditional bookmark marks just a single page in a book, the FME bookmark can cover a wide area of the canvas. A single workspace can be divided into different sections by applying multiple bookmarks.

TIP

If any objects on the workspace canvas are selected when a bookmark is created, the bookmark is automatically expanded to include those items.

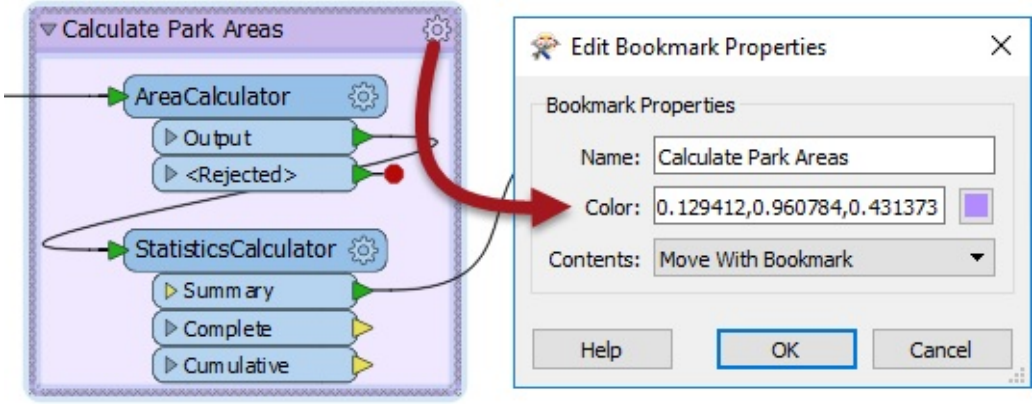
Resizing and Editing a Bookmark

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



Bookmark Properties

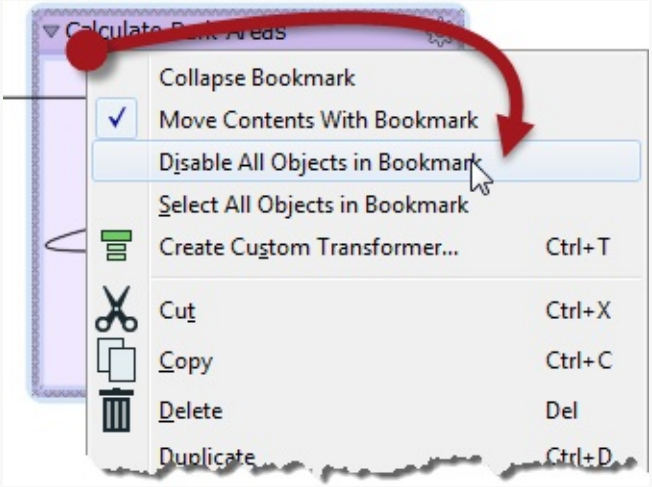
Click the cogwheel icon on a bookmark header to open the bookmark properties dialog:



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

TIP

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:



Data Integration Scenario	
Data	City of Vancouver Open Data (see table below)
Overall Goal	Use your FME skills to integrate data from several sources to answer a question.
Demonstrates	Content Transformation, Structural Transformation, Schema Editing
Start Workspace	C:\FMEData2018\Workspaces\DataIntegration\ScenarioBegin.fmw
End Workspace	Instructors can request access to the final workspaces here.

Introduction

In the previous exercises you have learned how to translate and transform data using FME. Now let's put that knowledge to use! As we covered in the Lecture section, data integration is a key requirement for modern organizations. In this exercise you will develop a workspace that solves a data integration problem in a city government scenario.

You will:

- inspect your data
- combine your datasets
- query your data to answer a question or solve a problem
- answer lab questions about the process of integrating data

This exercise will provide an example of integrating three datasets into a central database. After that you can follow similar steps to choose one additional dataset to answer another question. If you have trouble carrying out the steps you can refer to the Additional Procedures section, FME Workbench Help (also available online in our Documentation) or the community Knowledge Center.

Please continue to refer to the Lab Questions throughout, as some of the questions refer to the scenario.

Exercise Goals

- Integrate data into a single database using multiple feature types on a single writer
- Use transformers to create the desired schema in the destination database.
- Merge data using an Overlayer transformer
- Use ChartGenerator to make analyze integrated data
- Select another dataset and add it to your workspace to answer a question.

Inspect Data

To start off, inspect the data you have been given. There is a wide variety of topics and formats available. Following the instructions from Exercise 3, open some of these files in the FME Data Inspector to see what attributes they have.

For the walkthrough we will be using the following datasets:

- Bike Paths
- Neighborhoods
- Public Art

More data is available from the City of Vancouver Open Data catalogue.

Topic	Format	Location (FMEData2018\Data...)
Addresses	Esri Geodatabase	Addresses\Addresses.gdb
Building Footprints	SpatialLite Database or AutoCAD DWG	Engineering\BuildingFootprints\building_footprints.sl3 C:\FMEData2018\Data\Parcels\BuildingFootprints.dwg
Bike Paths	Esri Shapefile	Transportation\Cycling\BikePaths*.shp
Business Licenses	Excel Spreadsheet	Planning\BusinessLicenses.xlsx
City Grid	Geography Markup Language	Boundaries\CityGrid.gml
City-owned Properties	Esri Shapefile	Parcels\CityProperties\CityProperties.shp
Coastal Zones	Geography Markup Language	Boundaries\CoastalZones\CoastalZones.gml
Community Resources	Esri Geodatabase	CommunityMapping\CommunityMap.gdb
Crime	Comma Separated Values	Emergency\Crime.csv
Drinking Fountains	Comma-separated Values	Engineering\DrinkingFountains.csv
Election Results	Excel Spreadsheet	Elections\ElectionResults.xls

Election Voting Districts	Geography Markup Language	Elections\ElectionVoting.gml
Elevation	Digital Elevation Model	ElevationModel\DEM-Full.dem
Firehalls with Zones	Geography Markup Language	Emergency\FireHallsWithZones.gml
Forward Sortation Areas (for postcode)	Esri Shapefile	Addresses\ForwardSortationAreas.shp
Land Boundary	Esri Shapefile	Boundaries\LandBoundary\VancouverLandBoundary.s
Neighborhoods	Google Keyhole Markup Language	Boundaries\VancouverNeighborhoods.kml
Orthophotos (scale corrected aerial photographs)	TIFF	Orthophotos*.tif
Parcels	AutoCAD DWG	Parcels\Parcels.dwg
Parking Meters	MapInfo TAB	Transportation\Parking\Meters.tab
Parking Tickets	Comma-separated Values	Transportation\Parking\ParkingTickets.csv
Parks	MapInfo TAB	Parks\Parks.tab
Planning Restrictions (Business Improvement Areas, Historic Areas, Noise Control Areas, and View Cones)	OCG Geopackage	Planning\PlanningRestrictions.gpkg
Point Cloud (LiDAR)	LAS	PointClouds\CoV*.laz
Public Art	Excel Spreadsheet	Culture\PublicArt.xlsx

Public Trees	Comma-separated Values	Parks\Parks.tab
Roads	AutoCAD DWG or Microstation DGN	Transportation\CompleteRoads.dwg or RoadsDGN.dg
Street Lighting	AutoCAD DWG and Microstation DGN	Engineering\StreetLighting.dwg and StreetLightingDGN.dgn
Traffic Signals	AutoCAD DWG and Microstation DGN	Engineering\TrafficSignals.dwg and TrafficSignalsDGN.dgn
Zoning	MapInfo TAB	Zoning\Zones.tab
Zoning Descriptions	Adobe PDF	Zoning\ZoneDescriptions.pdf

Walkthrough: Integrating Multiple Feature Types into a Geodatabase

In this section we will integrate data of several different formats into a single database: an Esri geodatabase used commonly in GIS applications.

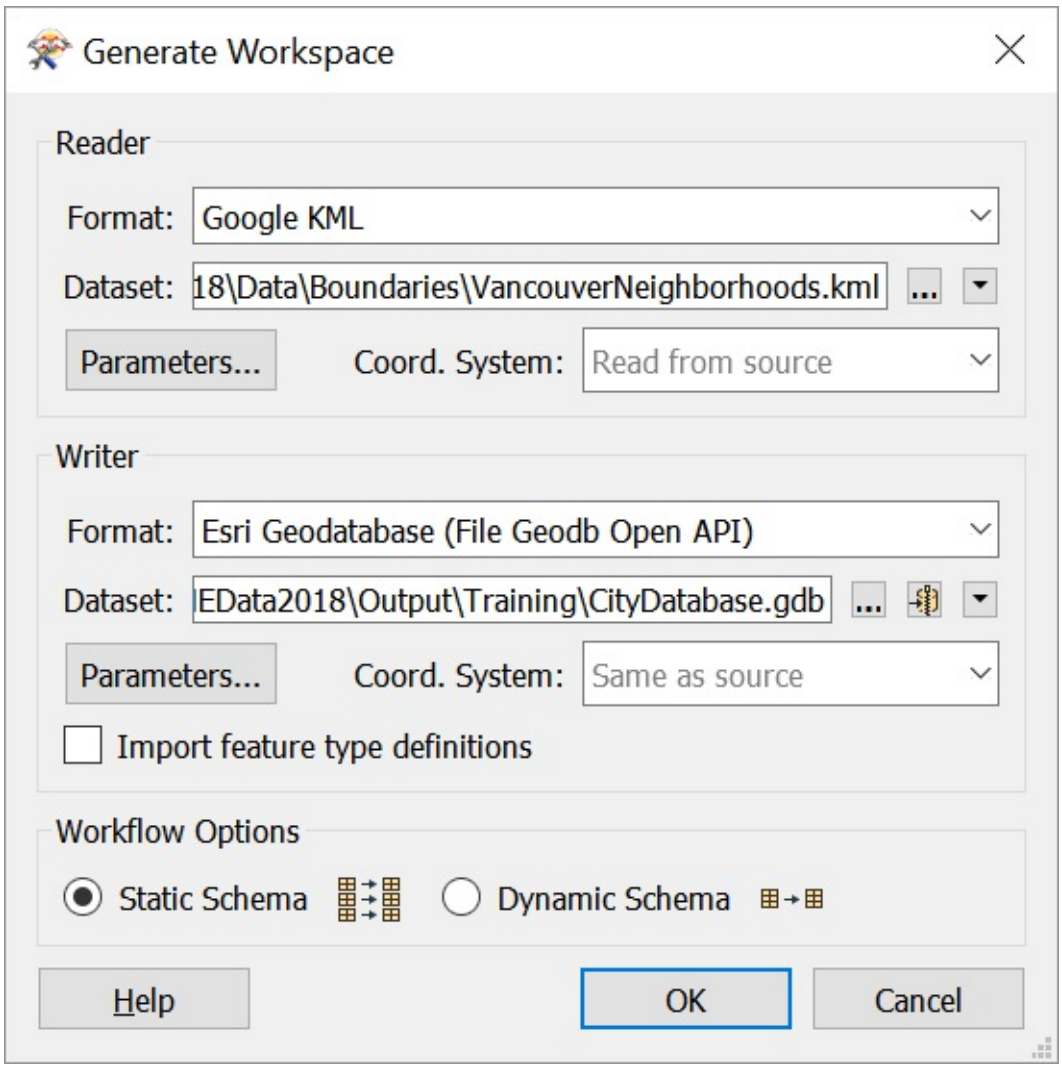
This walkthrough will reiterate procedures from the previous exercises, in addition to showing how to write multiple feature types using a single writer. After integrating the data into a single database, we will carry out some basic analysis of the data.

1) Start Workbench
Use the Generate Workspace dialog to create a workspace using these parameters:

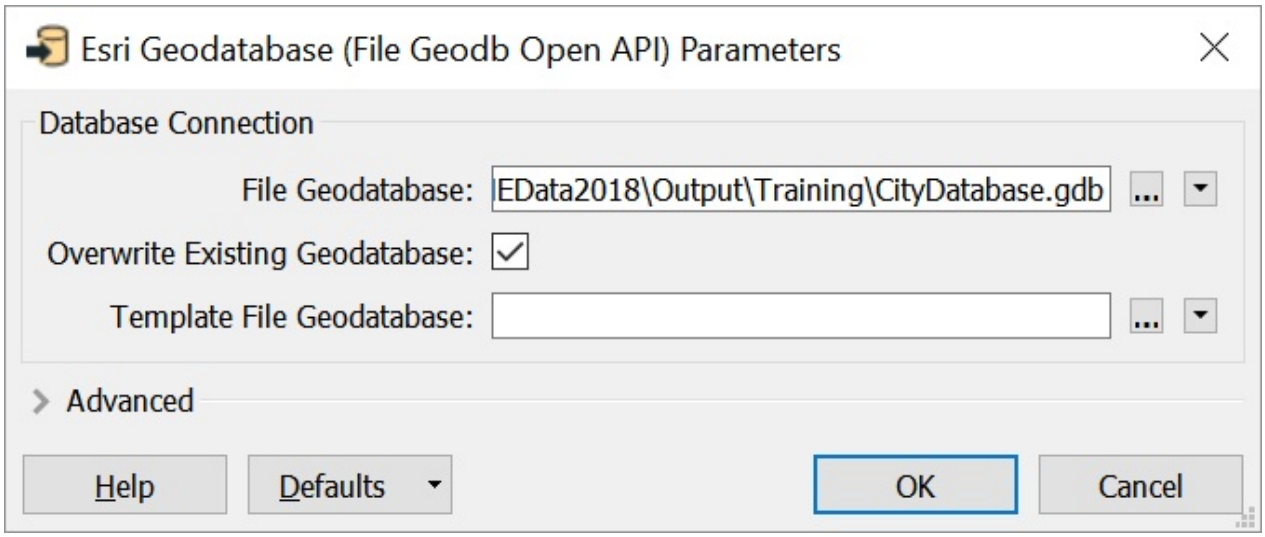
Reader Format	Google KML
Reader Dataset	C:\FMEData2018\Data\Boundaries\VancouverNeighborhoods.kml
Writer Format	Esri Geodatabase (File Geodb Open API)
Writer Dataset	C:\FMEData2018\Output\Training\CityDatabase.gdb

Note: Make sure you select File Geodb Open API for your geodatabase format or you may run into problems later on. The Open API implementation of GDB will allow you to create and view this geodatabase in FME Data Inspector without an Esri license. However, you will require a licensed copy of ArcMap, ArcGIS Pro, or ArcGIS Online to view the GDB in its native software. Alternatively, your instructor might ask you to use a different format, e.g., a PostGIS database; FME makes it easy to write whatever format you want.

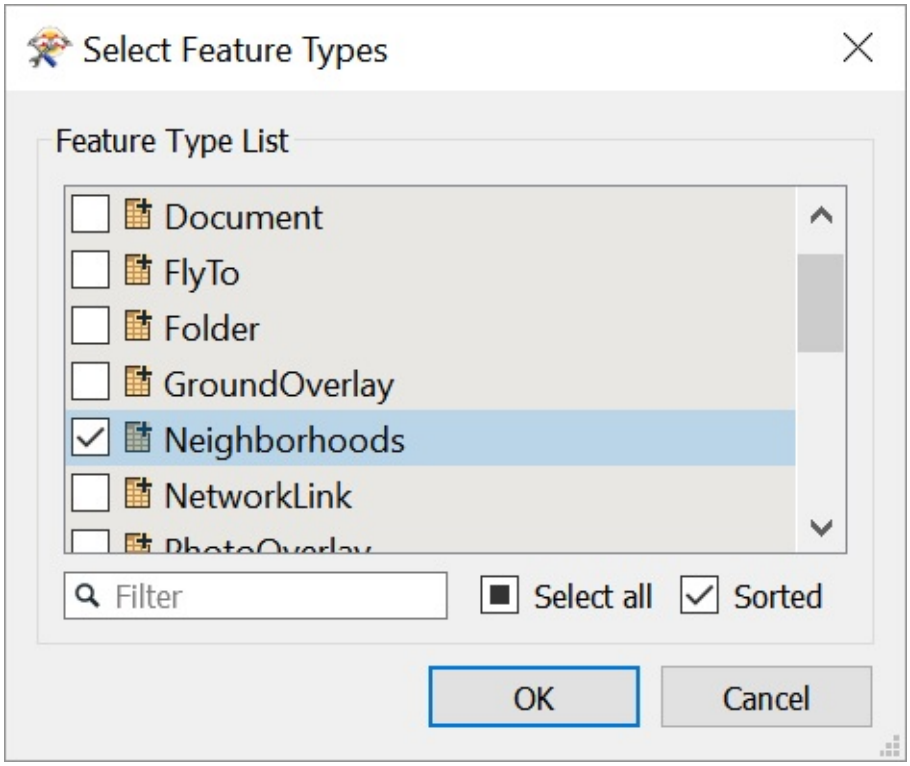
Your dialog should look like this:



Click the Parameters buttons in the Generate Workspace dialog to check the reader/writer parameters. We will change one for this exercise. Under your Writer Parameters, in the Database Connections section, check Overwrite Existing Geodatabase. With that checked we will recreate the database entirely every time we write out the data. Your dialog should look like this:



Click OK twice. You will be presented with a Select Feature Types dialog because our reader data set contains multiple layers. We only need the feature type named Neighborhoods, which contains the polygons of neighborhood boundaries. Make sure Neighborhoods is the only feature type selected:

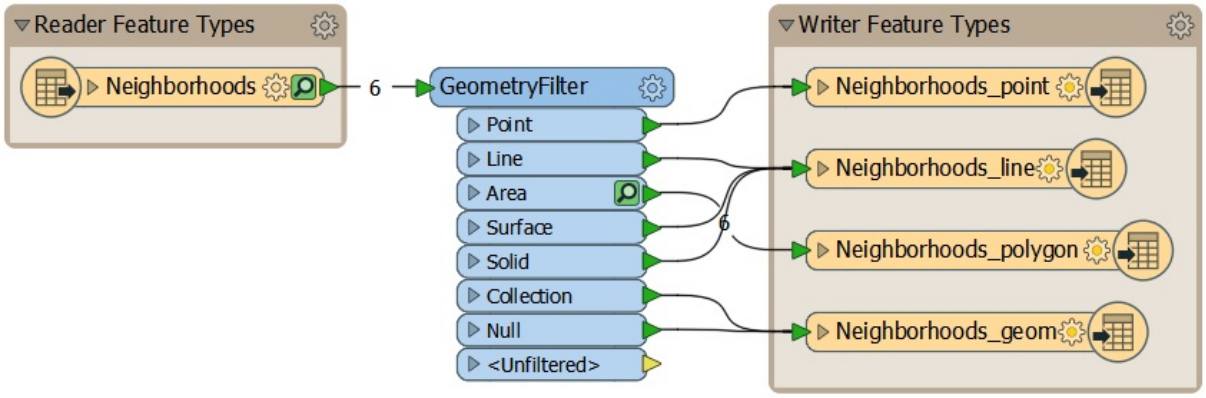


Click OK.

Note: if you pressed OK before setting Writer Parameters, you can change this after generating your workspace. In the Navigator window under your CityDatabase FILEGDB writer > Parameters > Overwrite Existing Database (set to Yes).

2) Clean Up Generated Workspace

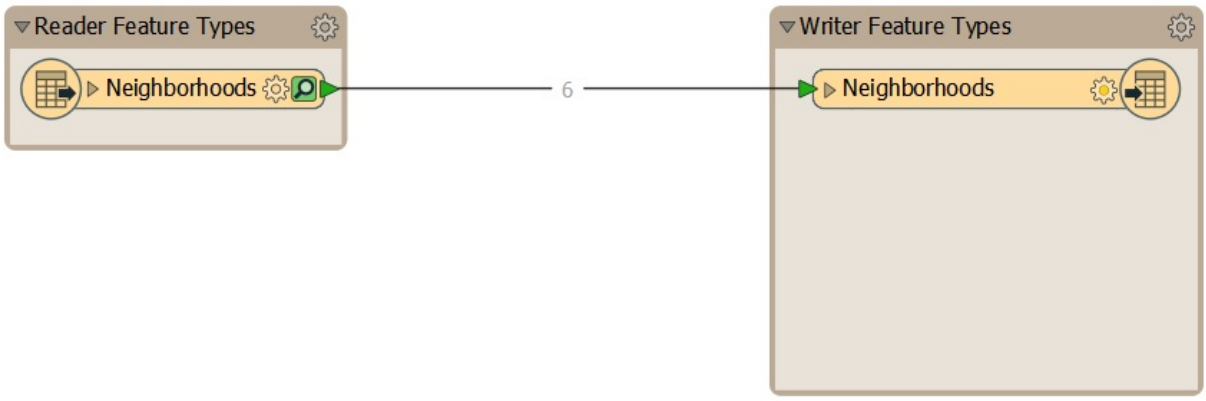
The generated workspace added a geometry filter automatically. This happens because Esri geodatabases store layers with only one geometry type.



However, our reader feature type only contains polygons, so we can clean up the starting workspace by:

- deleting the GeometryFilter
- deleting the Neighborhoods_point, Neighborhoods_line, and Neighborhoods_geom writer feature types
- renaming the Neighborhoods_polygon writer feature type to just "Neighborhoods"

Your workspace should now look like this:



3) Add Excel Reader

Let's add two more readers. We'll look at two datasets from the list above that need changes to the defaults to work as we wish. Other situations that might require similar changes are covered in the Additional Procedures section.

You can add readers by clicking Readers > Add Reader, or by clicking an empty space on the canvas and typing the name or file extension of the format you wish to add and picking it from the Quick Add Menu.

First, let's add an Excel file of public art:

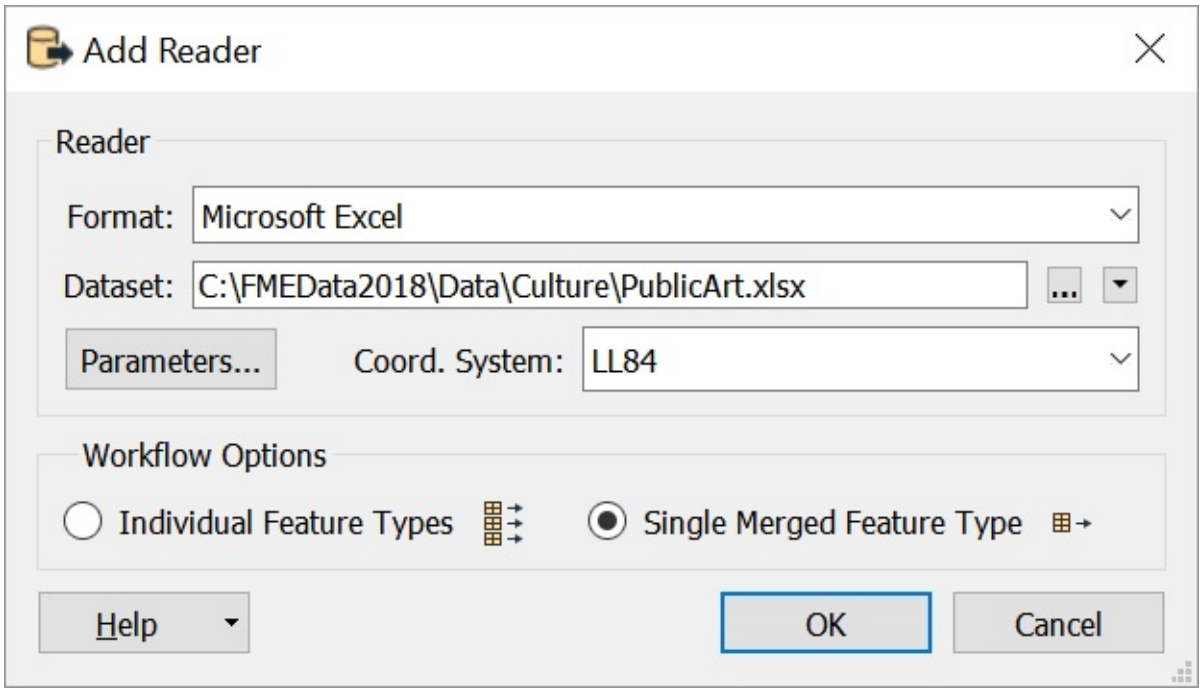
Reader Format	Microsoft Excel
Reader Dataset	C:\FMEData2018\Data\Culture\PublicArt.xlsx

We will make two changes to the reader defaults.

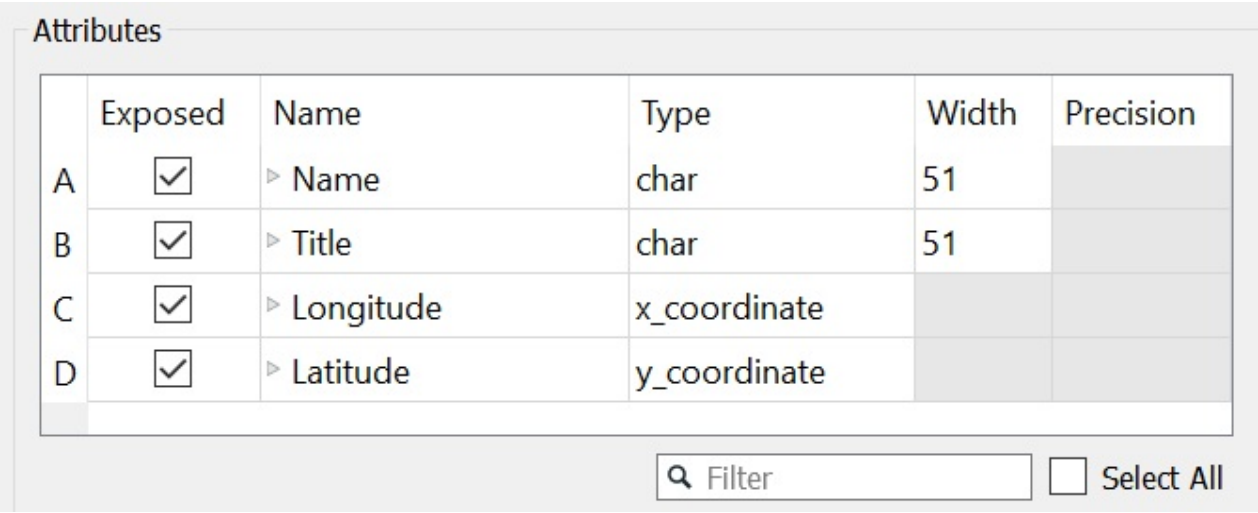
First, change Workflow Options from Individual Feature Types (the default) to Single Merged Feature Type. The Excel reader treats rows as features and worksheets as feature types. This means by default, FME would try to read every sheet in the file as a separate feature type. If you inspect the data, you'll find it has one sheet per neighborhood. This works in Excel, but we would prefer that our database treat all the public art as one point layer. Choosing Single Merged Feature Type does this for us.

Second, we need to add a coordinate system. The reader has latitude and longitude coordinates in the Excel file, but because Excel does not store coordinate system information, we have to tell FME which one to use. We know that it is LL84, so type that into the Coordinate System box in your dialog.

Your dialog should look like this:

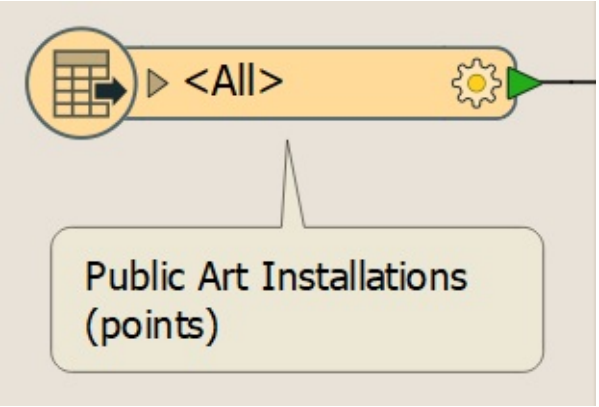


Before clicking OK, click on Parameters. We have to ensure the Reader recognizes the right columns in the Excel sheet as coordinates. Under the Attributes table column C named Longitude should by type x_coordinate and column D named Latitude should be type y_coordinate. Your dialog should look like this:



Click OK twice. This adds a new Excel reader to our Navigator window and gives it a feature type on our canvas named <All>, the default name for merged feature types. This isn't very descriptive, however. Let's add an annotation to indicate that this is the merged public art

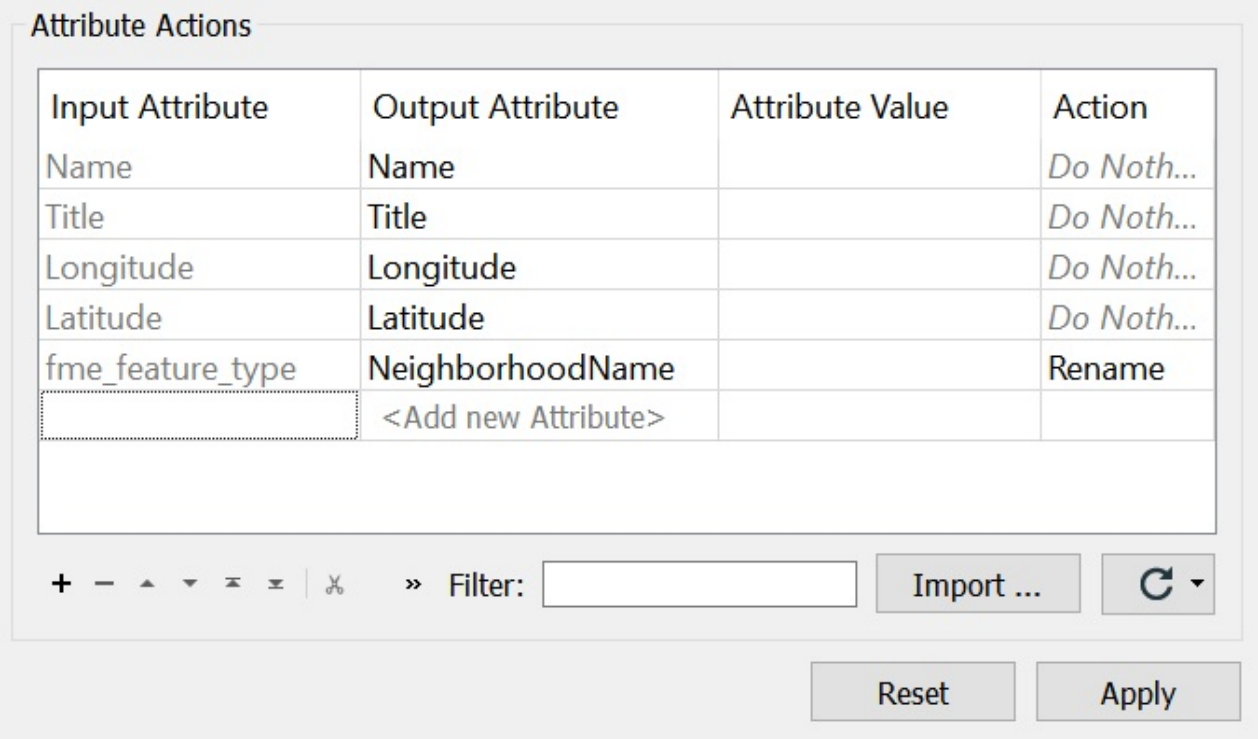
data. Right-click your feature type and select Attach Annotation. You can write something like, "Public Art Installations (points)":



Even though we added the public art installations from all neighborhoods, we can still distinguish between neighborhoods through the fme_feature_type attribute. This attribute simply gives the name of the feature type the feature belongs to. It exists on all FME features, but is not always exposed. It is exposed by default whenever you add a merged feature type. You can see it if you click the drop-down arrow next to your new <All> reader feature type.

If you right-click this feature type and select Inspect you can see that the data is still organized by neighborhood. If you select a feature the first attribute shown in the Feature Information window is FME Feature Type, which gives the neighborhood name.

Let's turn that fme_feature_type attribute into something more meaningful for our geodatabase. Add an AttributeManager after the <All> reader feature type. Open the Parameter Editor and click in the Attribute Actions table in the Output Attributes column. This will let you rename that attribute. Let's change it to NeighborhoodName instead of fme_feature_type. The Action column will automatically change to Rename. Your Attribute Actions table should now look like this:



Click OK/Apply.

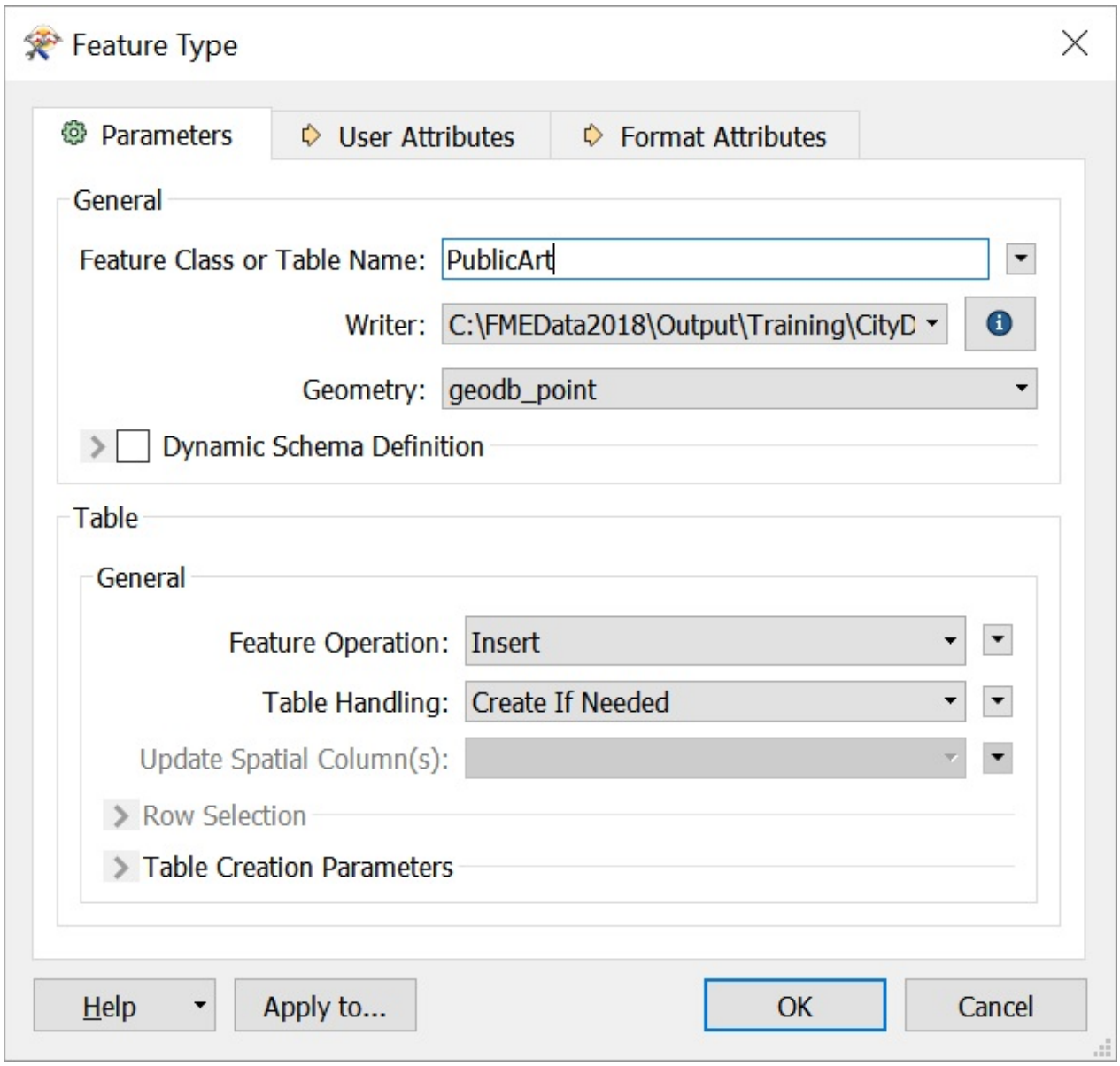
4) Add Writer Feature Type for Public Art

Now we need a writer feature type for our public art points.

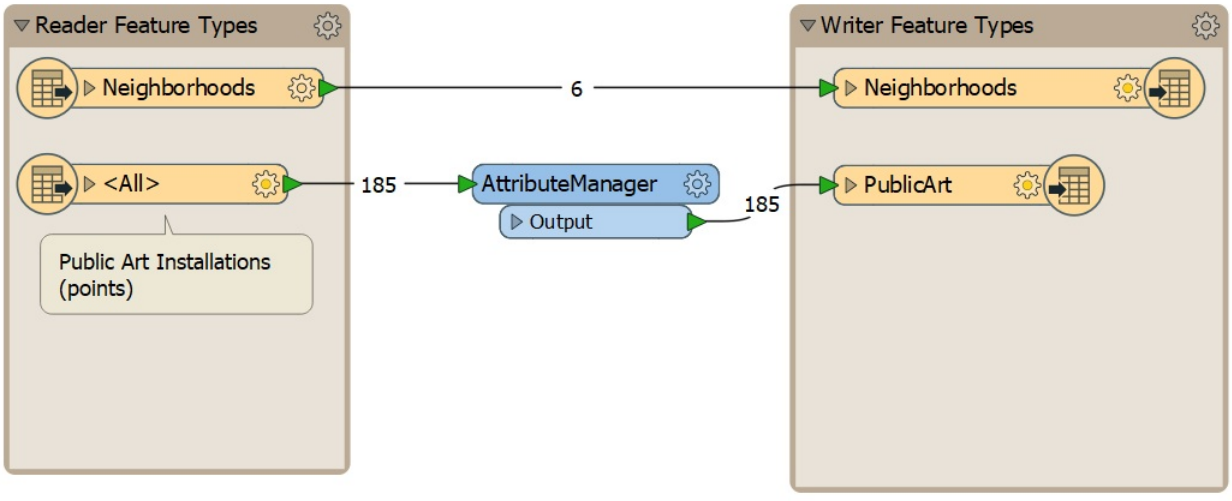
TIP

Remember, we already have our geodatabase writer set up. We don't want to add another writer, because we want all of our data to be written to the same output database. Instead, we need to use feature types, because we want our data to be organized by layer.

We can add a feature type to a writer in the menu bar under Writers > Add Feature Type. Because we only have one writer it will automatically be selected. Name your new feature type PublicArt and give it Geometry type geodb_point. Your Feature Type dialog should look like this:



Click OK. Connect your new feature type to the Output port of your Attribute Manager. Your canvas should look like this:



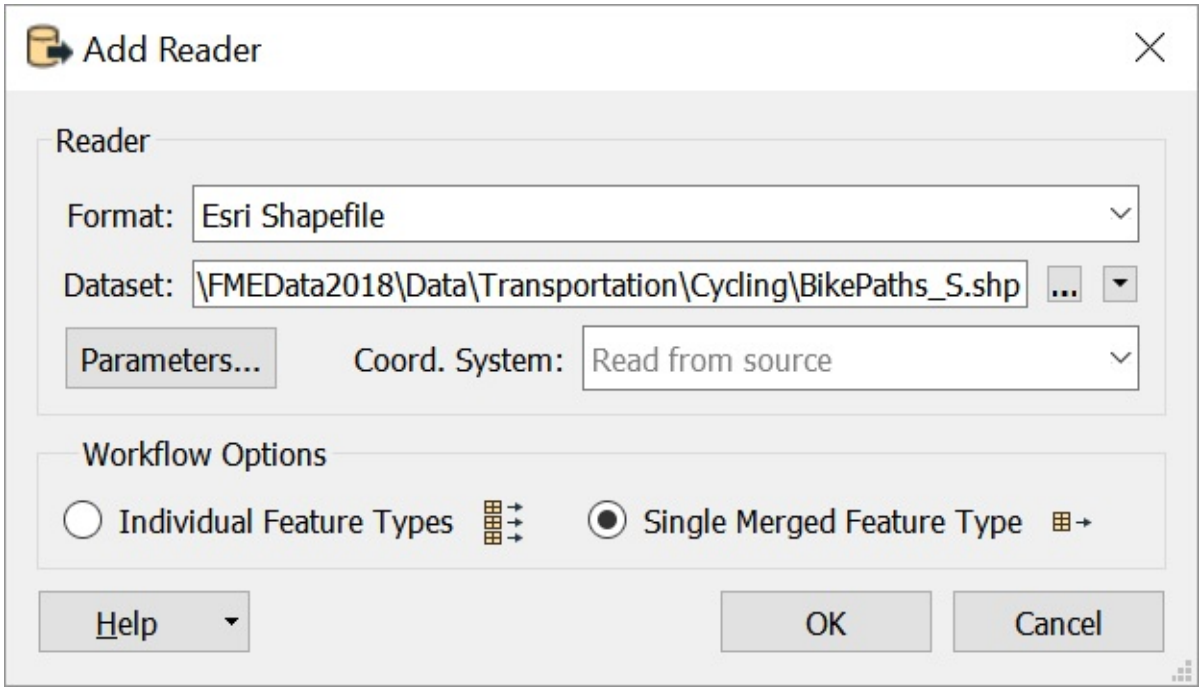
5) Add Shapefile Reader

Add another reader by clicking Readers > Add Reader, or by clicking an empty space on the canvas and typing the name or file extension of the format you wish to add and picking it from the Quick Add Menu. In this case that would be shp for Esri Shapefile.

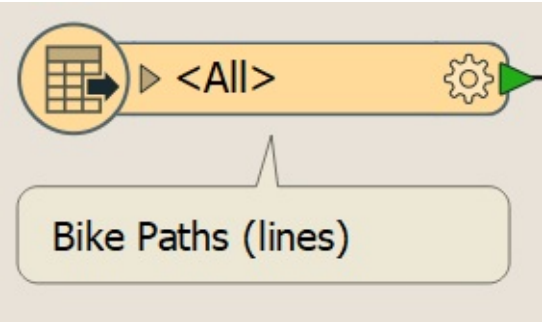
Reader Format	Esri Shapefile
Reader Datasets	C:\FMEData2018\Data\Transportation\Cycling\BikePaths_L.shp C:\FMEData2018\Data\Transportation\Cycling\BikePaths_M.shp C:\FMEData2018\Data\Transportation\Cycling\BikePaths_S.shp

When you select your dataset, use Ctrl or Shift click to select all three bike path shapefiles.

The bike path data is split up into three shapefiles by length of the bike path (L for long, M for medium, and S for short). Just like the public art points, we don't need these features separated in our database. Therefore, let's change Workflow Options from Individual Feature Types to Single Merged Feature Type. Because shapefiles contain coordinate system information, we don't need to change that here. Your dialog should look like this:

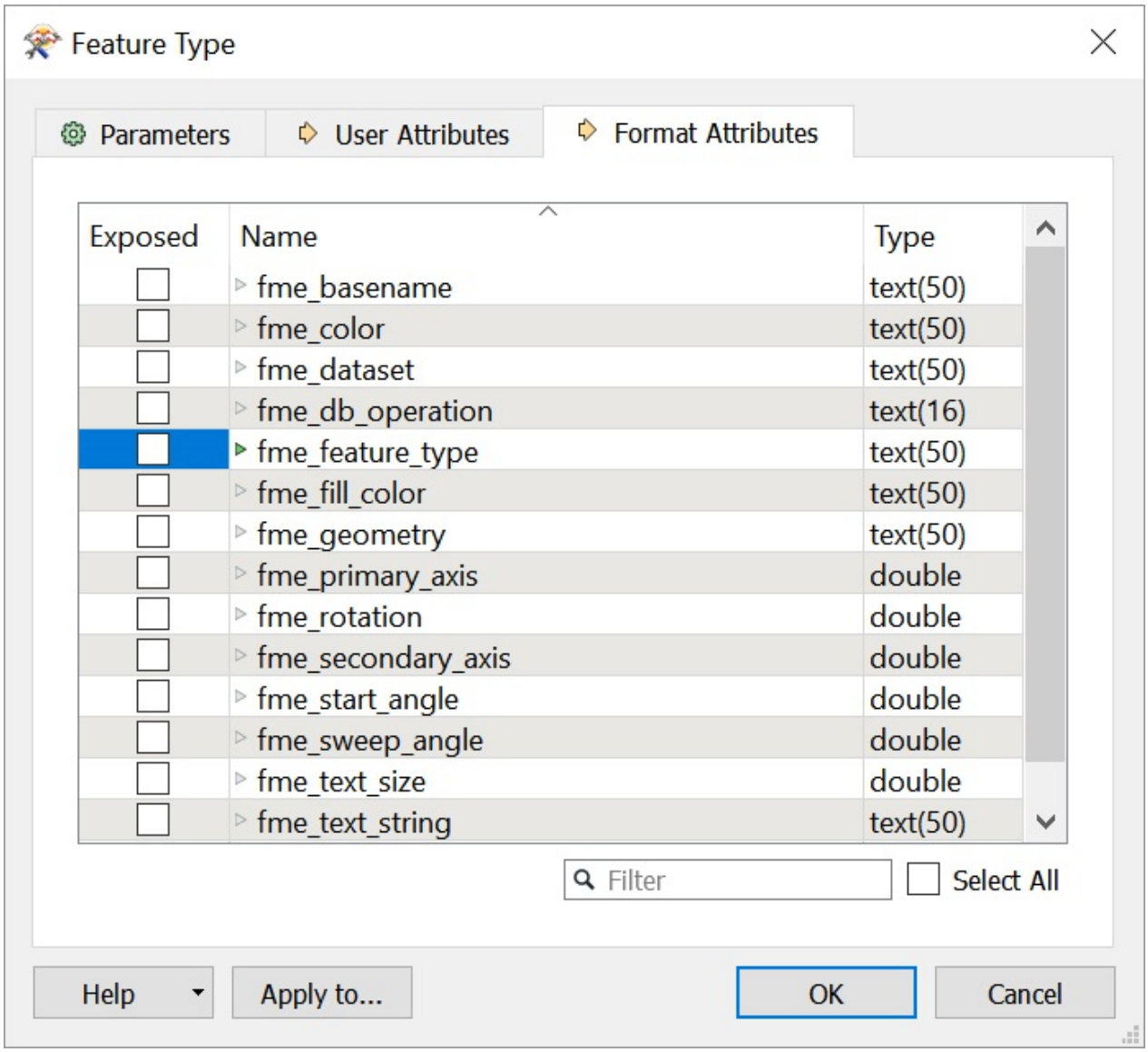


Click OK. This adds another <All> feature type to our canvas. Again, we should add an annotation so it is clear which reader is which:



If you inspect the reader feature type you'll find the data already has a PathType attribute with values S, M, or L. Therefore, we don't need to use AttributeManager to rename fme_feature_type like we did for the public art data. However, we also don't want fme_feature_type to be written to our final data.

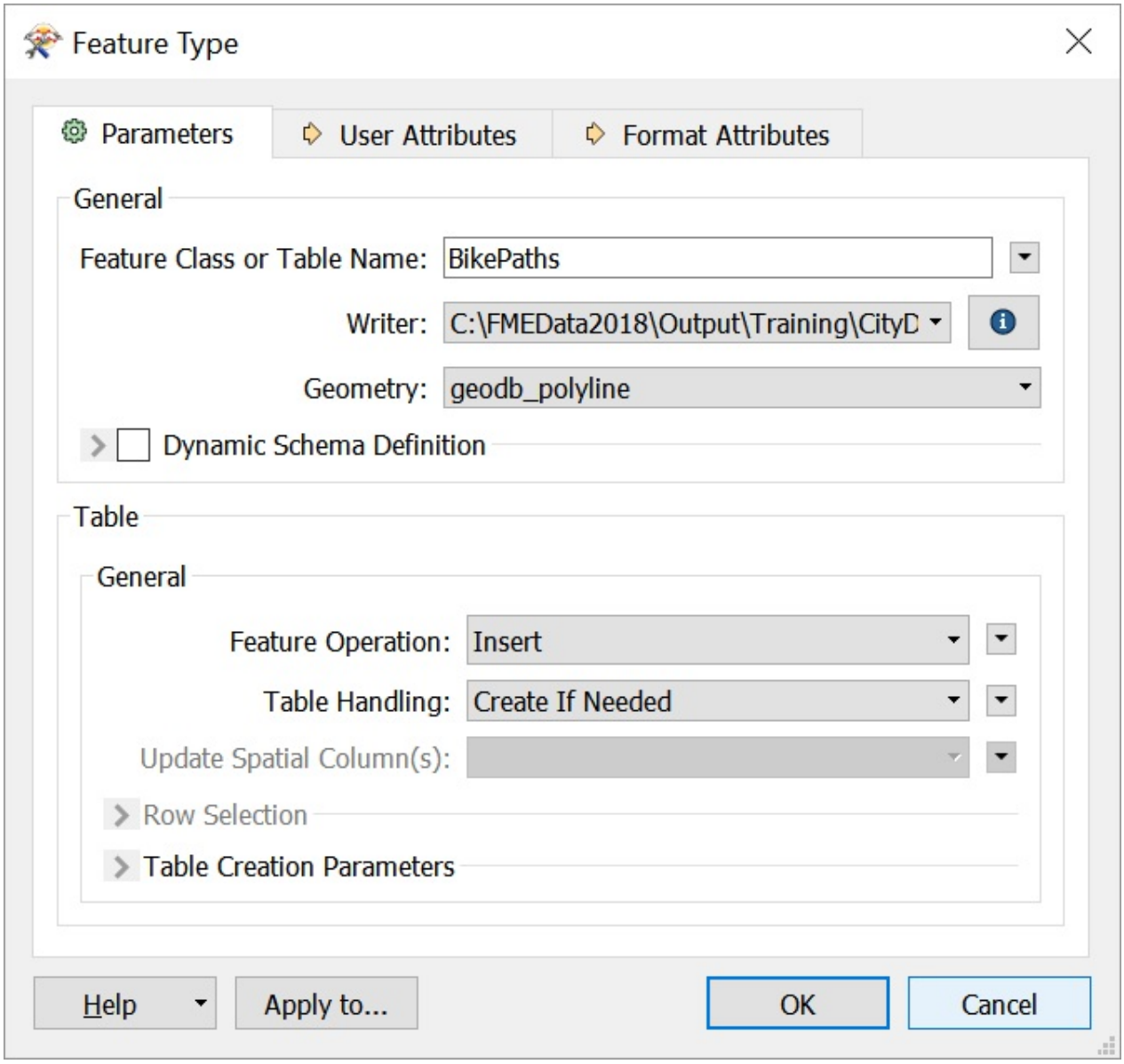
We can unexpose it on our reader feature type. Select your bike paths reader feature type and go to the Format Attributes tab in the Parameter Editor. You'll see that fme_feature_type is checked. Simply uncheck it and click OK/Apply:



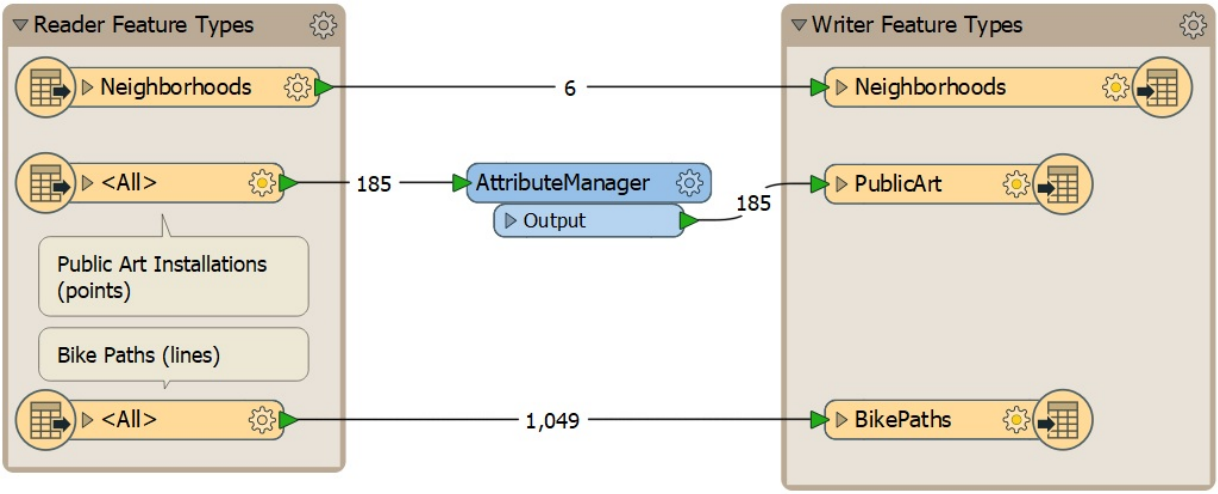
Now if you click the drop-down arrow on your bike paths reader feature type you'll see that fme_feature_type has been removed from the schema as we wanted.

6) Add Writer Feature Type for Bike Paths

Now we need a writer feature type for our bike path lines. Click Writers > Add Feature Type, name your new feature type BikePaths, and give it Geometry type geodb_polyline. Your Feature Type dialog should look like this:



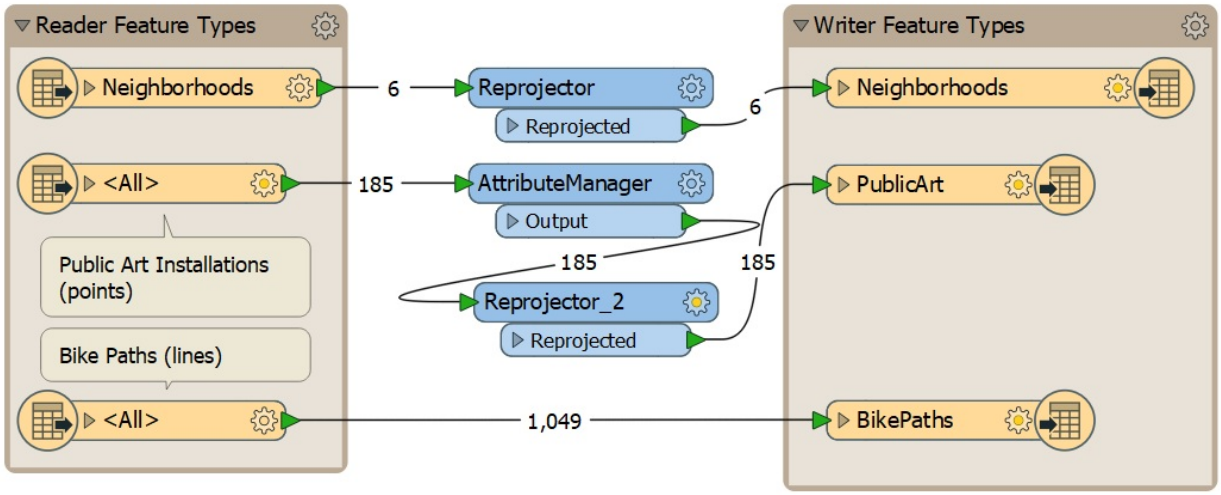
Click OK. Connect your ALL reader feature type containing the bike path features to your writer feature type BikePaths. Your canvas should look like this:



7) Reproject Data

Add a Reprojector transformer to the canvas and then connect it to the Neighborhoods feature type. Choose UTM83-10 as the Destination Coordinate System. Right-click the Reprojector transformer and select Duplicate to add another. Connect this between your AttributeManager output port and your Public Art writer feature type.

This will ensure our neighborhoods and public art data is in the same projection as our bike paths. Your canvas should look like this:

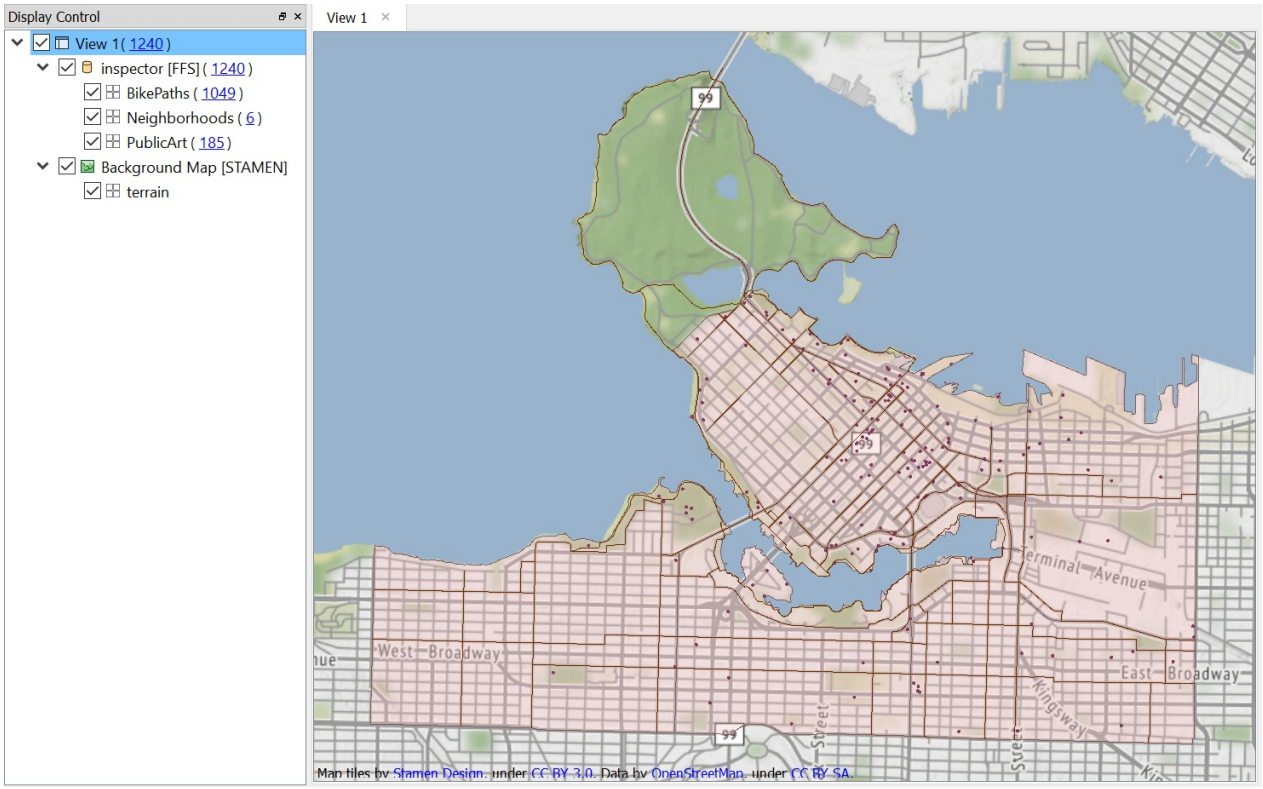


8) Inspect Output

Let's take a look at our data integrated into one geodatabase. You can run your workspace and then open the resulting geodatabase in Data Inspector, or you can click Writers > Redirect to Data Inspector.

When this function is on no data is actually written; instead the results of your translation are sent directly to the Data Inspector. It is useful for checking the results of your translation while your workspace is still in development.

You should see all three layers displayed in Data Inspector, now all stored in the same format in a centralized database:



Map tiles by Stamen Design, under CC-BY-3.0. Data by OpenStreetMap, under CC-BY-SA.

9) Example Data Integration Analysis

Let's look at an example of how integrating data facilitates analysis.

What if the City Planning Department wanted to know the total length of bike paths and number of public art installations by neighborhood? How would we do this using this workspace?

Take a minute and write or draw out how you would tackle this problem using what you have learned so far. Don't worry if you can't remember the exact name of transformers. Instead focus on outlining the process you would undertake to perform this analysis.

Let's find out if you were right! Note that there are usually multiple ways to solve a problem in FME, so your solution might still be valid.

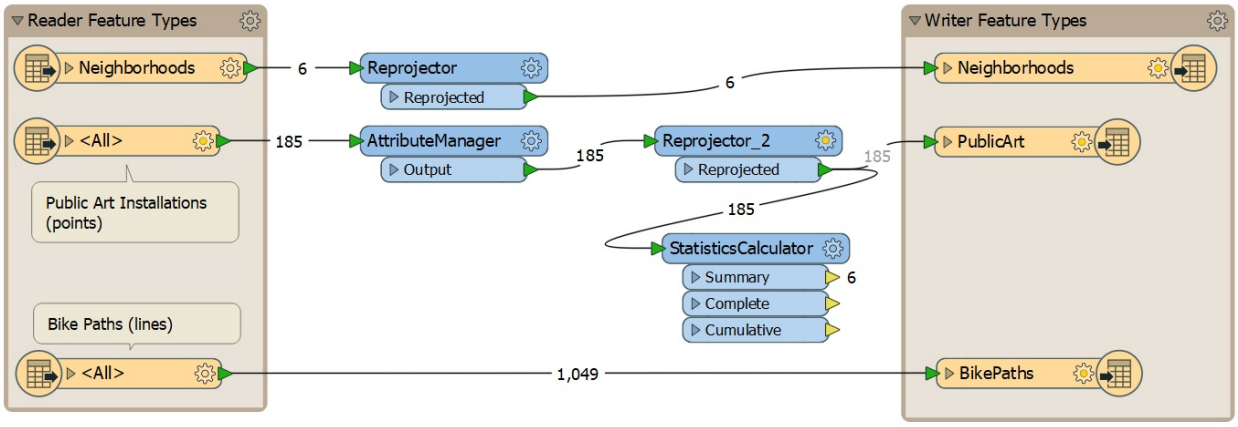
Here are the steps we will take to conduct this analysis:

1. Sum the total of bike path lengths by neighborhood.
2. Sum the count of public art installations by neighborhood.
3. Output a table or chart to show the results.

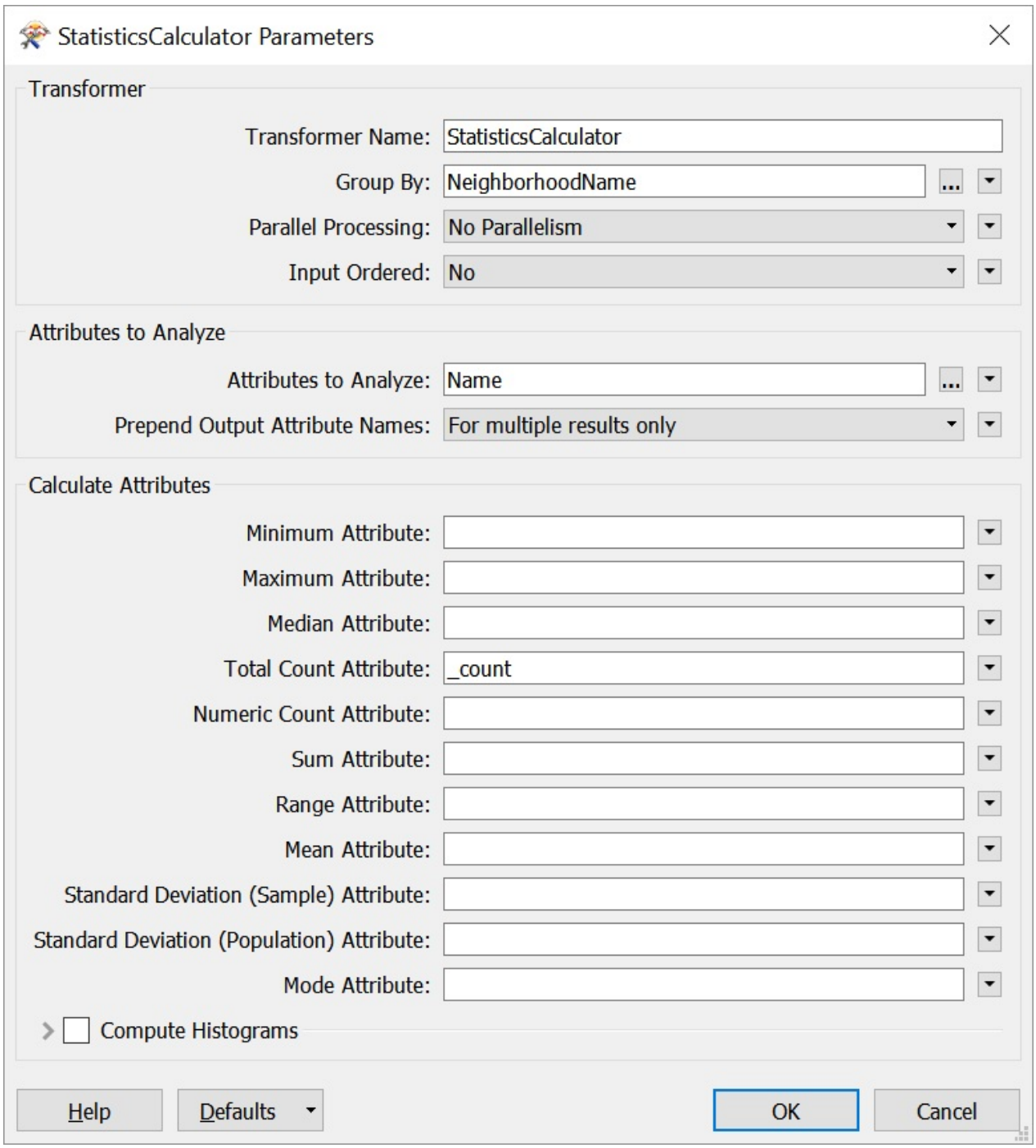
Let's walk through how to do that in Workbench.

10) Calculate Statistics for Public Art

Add a StatisticsCalculator after your public art Reprojector. We will use this to count the number of public art installations by neighborhood. Your canvas should look like this:

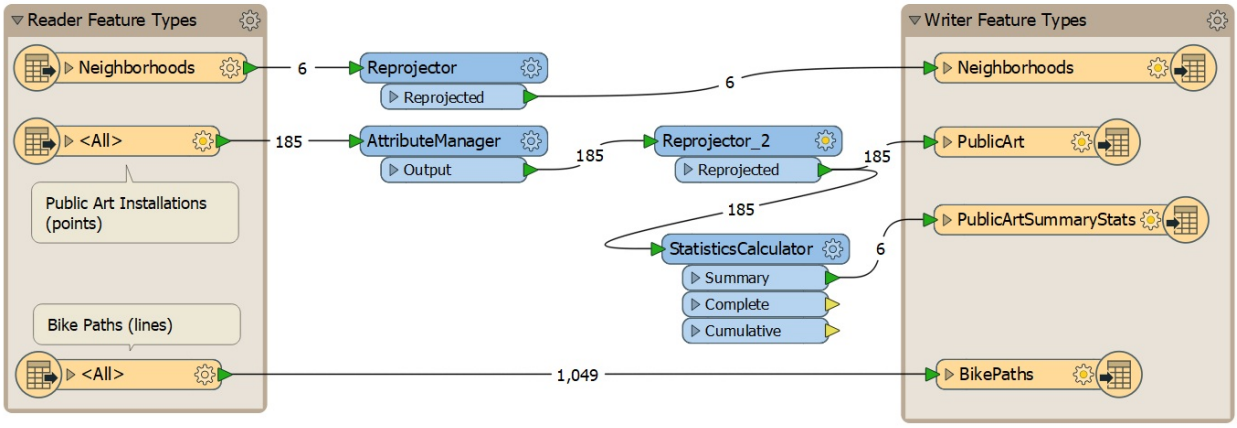


Open the parameters for the StatisticsCalculator. Select any attribute for Attributes to Analyze (it is just counting if there is a value, so any attribute will do). With the exception of Total Count Attribute, clear all the boxes in Calculate Attributes so they are not generated. Finally, set Group By to NeighborhoodName. Your dialog should look like this:



Click OK/Apply. The Summary output port will now output a table with the count of public art installations by neighborhood.

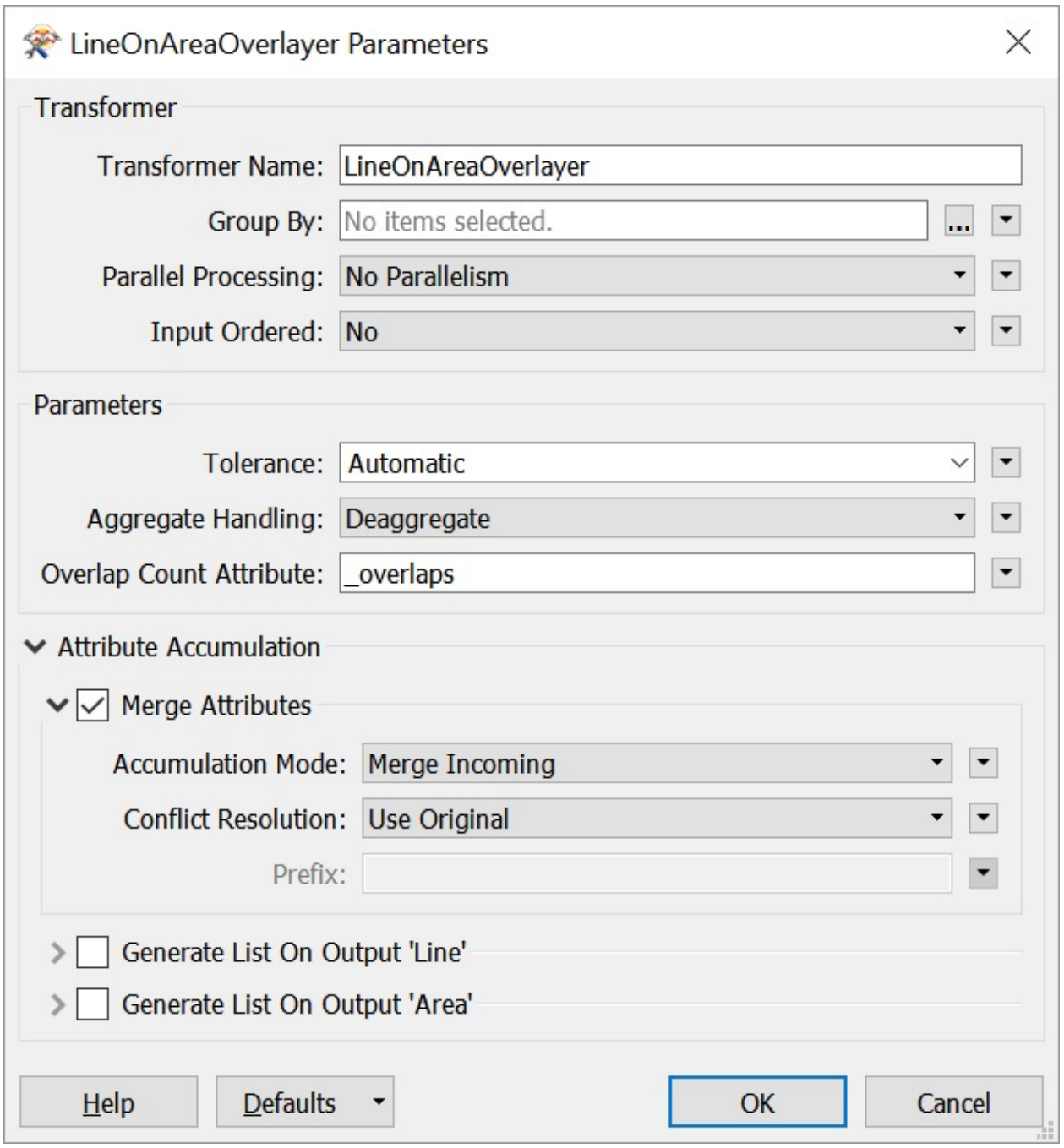
Let's write that out as a table in our geodatabase. Click Writers > Add Feature Type. Call it PublicArtSummaryStats and give it Geometry type geodb_no_geom. This will store it without geometry as a table. Click Ok. Once your new feature type is added, connect it to the Summary port of your StatisticsCalculator. Your canvas should look like this:



You can run the translation and inspect the table if you want.

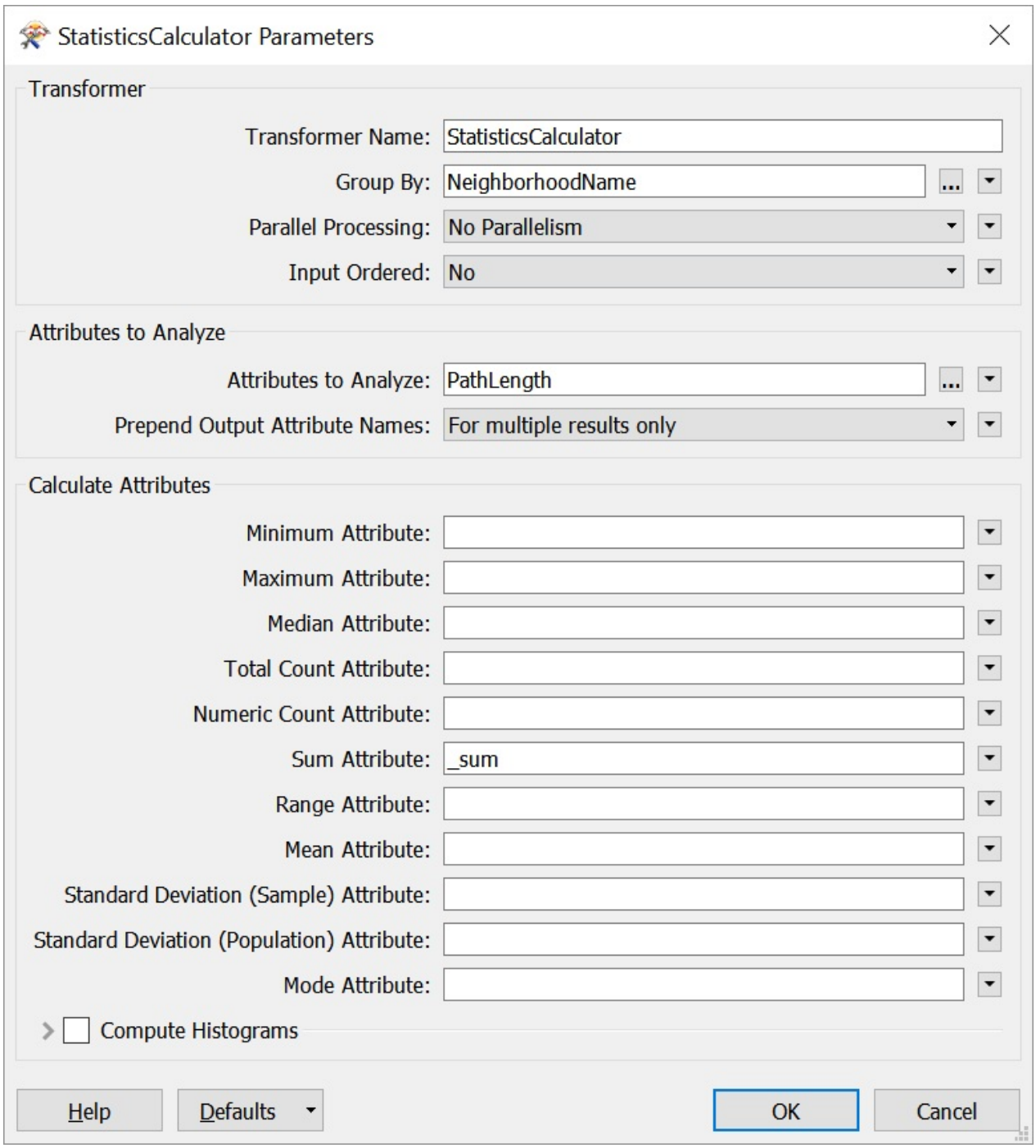
11) Calculate Statistics for Bike Paths

Add a LineOnAreaOverlayer to your canvas. This transformer will let us add attributes from the neighborhoods to the bike paths that overlap them. This will let us know which neighborhood each bike path segment is in. The Reprojector should connect to the Area port (because it is polygons of neighborhoods) and the bike path reader feature type should connect to the Line port. Open the LineOnAreaOverlayer parameters and check the box Attribute Accumulation > Merge Attributes. Your dialog should look like this:



Click OK/Apply.

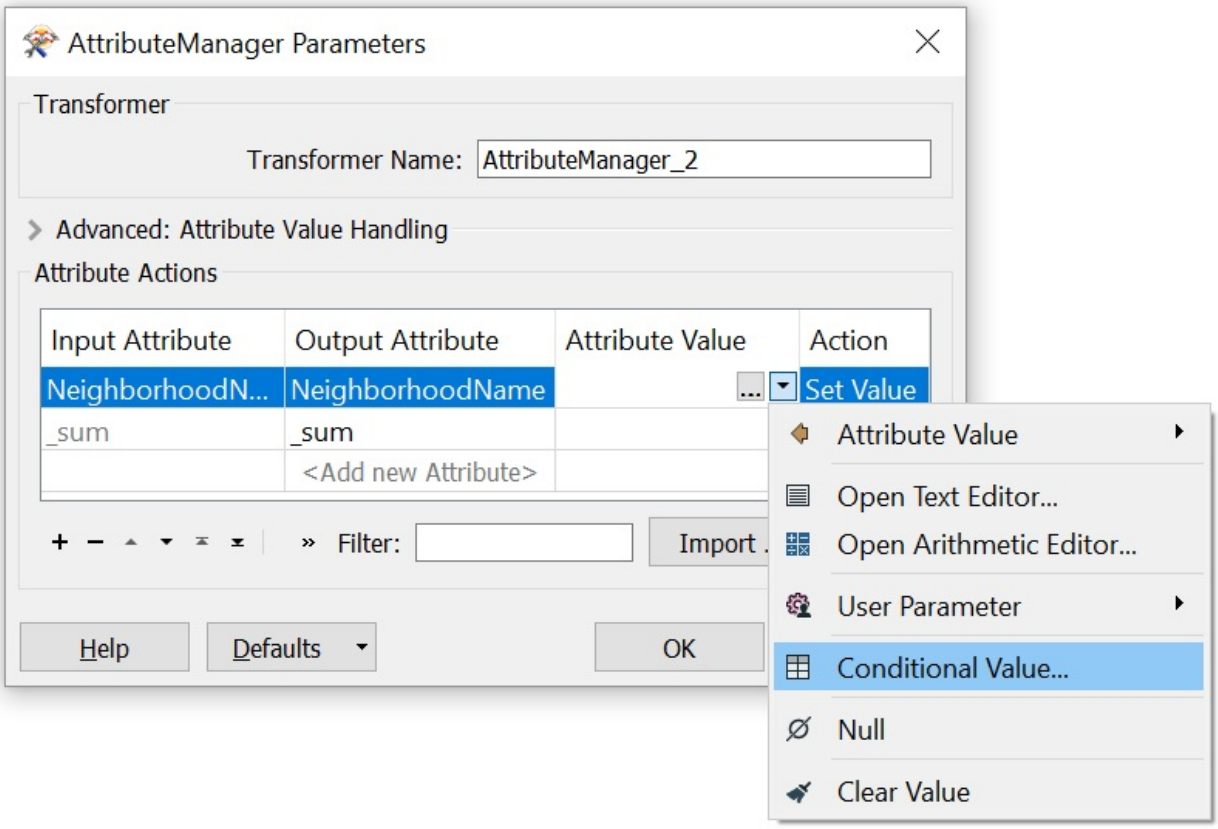
Now let's add a StatisticsCalculator to the Line output port of the LineOnAreaOverlayer. We will use this to sum up the PathLength for each feature and report the total length of bike paths, by neighborhood. To do this, open the paramters for the StatisticsCalculator. Set Group By to NeighborhoodName. Set the Attributes to Analyze to PathLength. With the exception of Sum Attribute, clear all the boxes in Calculate Attributes so they are not generated. Your dialog should look like this:



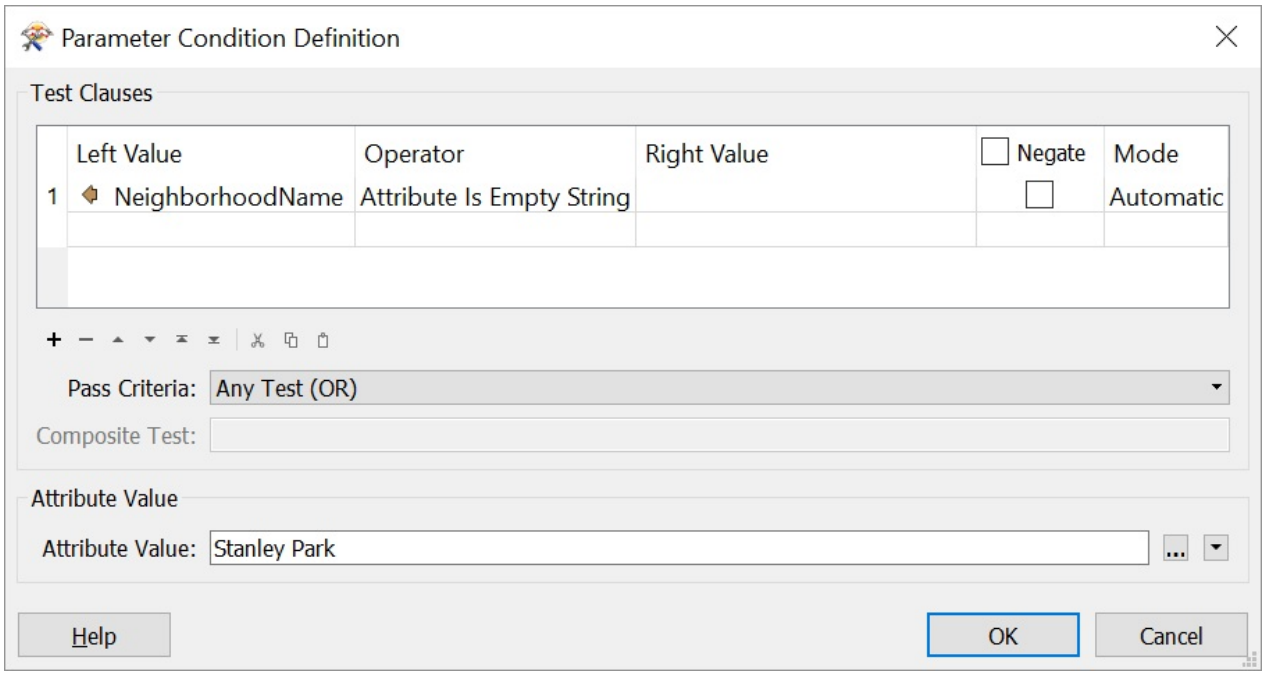
Click OK/Apply.

If you inspect the results of this transformer (using Feature Caching), you'll find that one of the neighborhood names is blank. This is because there is no neighborhood polygon for the bike paths in Stanley Park, the large park northwest of Vancouver's downtown. Let's fix this in the output table.

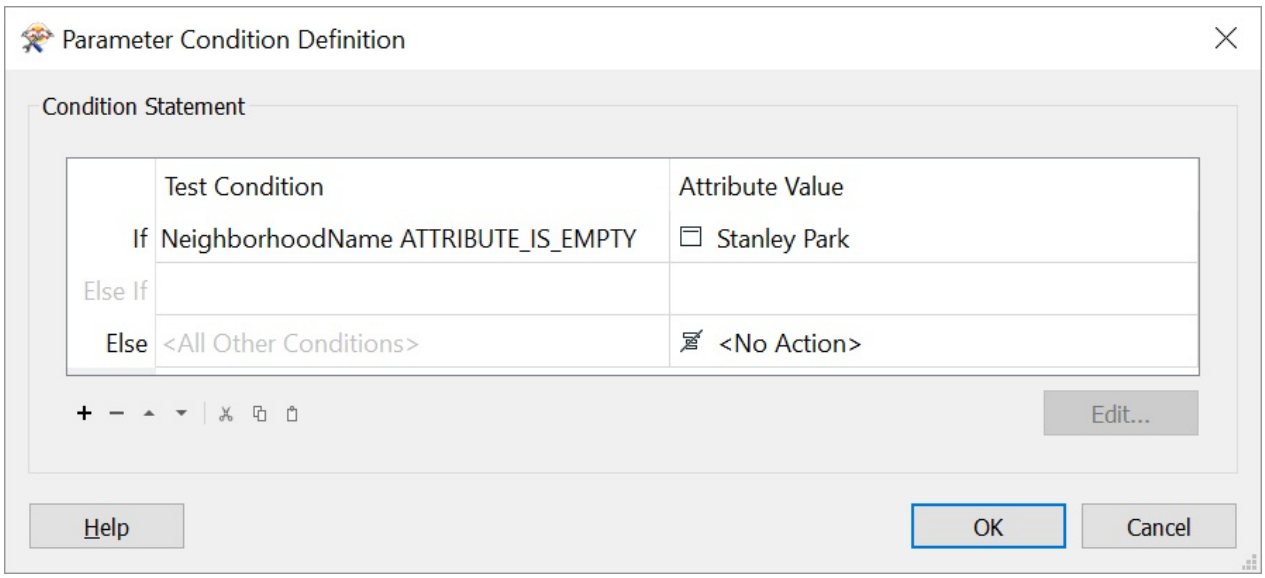
Add an AttributeManager and connect it to the StatisticsCalculator Summary output port. Open its parameters and click the drop-down arrow in the Attribute Value column for the Input Attribute NeighborhoodName. Then select Conditional Value:



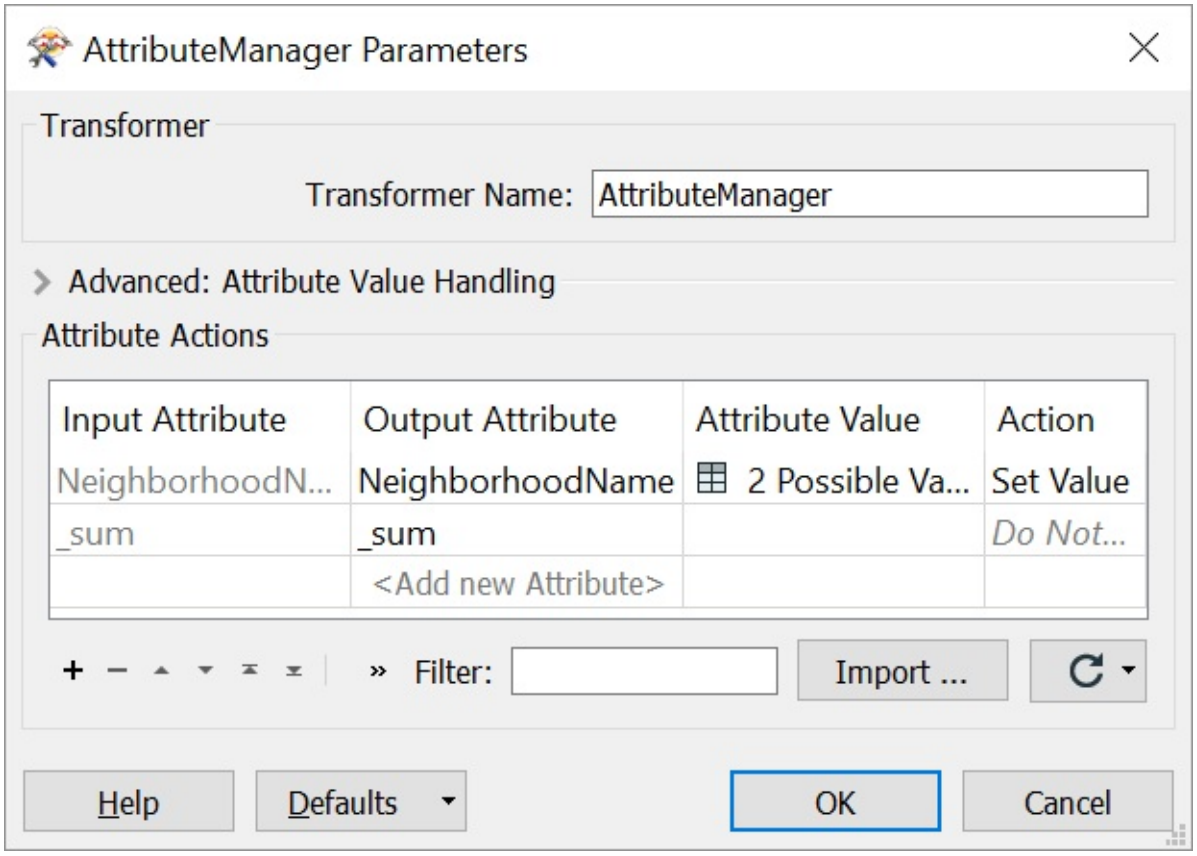
We will set the AttributeManager to set the value of NeighborhoodName to Stanley Park if it doesn't already have a value. Double click on the cell in the row If and the column Test Condition. In the new window, for Left Value select the attribute NeighborhoodName. For the Operator Select Attribute is Empty String. Finally, under Attribute Value > Attribute Value, enter Stanley Park. Your dialog will look like this:



Click OK. Now your Parameter Condition Definition should look like this:

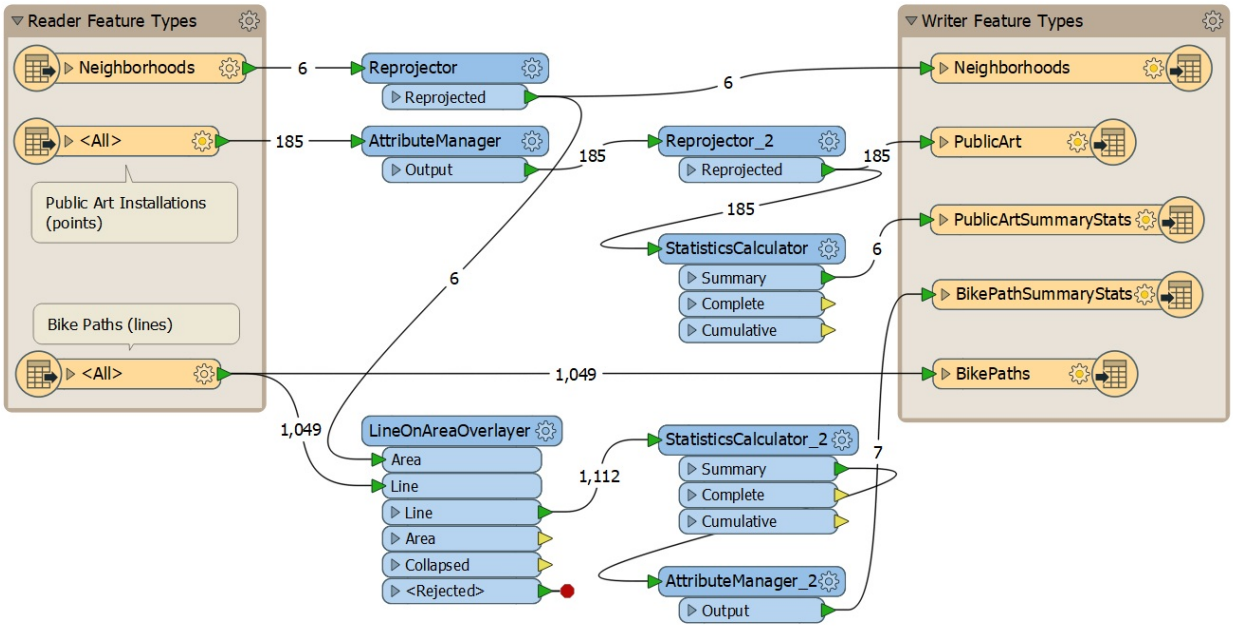


Click OK again. Now your Attribute Manager will look like this:



Great! Now our AttributeManager will take care of that empty NeighborhoodName value and replace it with Stanley Park.

Let's write these results as a table in our geodatabase as well. Click Writers > Add Feature Type. Call it BikePathSummaryStats and give it Geometry type geodb_no_geom. Click Ok. Once your new feature type is added, connect it to the Summary port of your bike paths StatisticsCalculator. Your canvas should look like this:

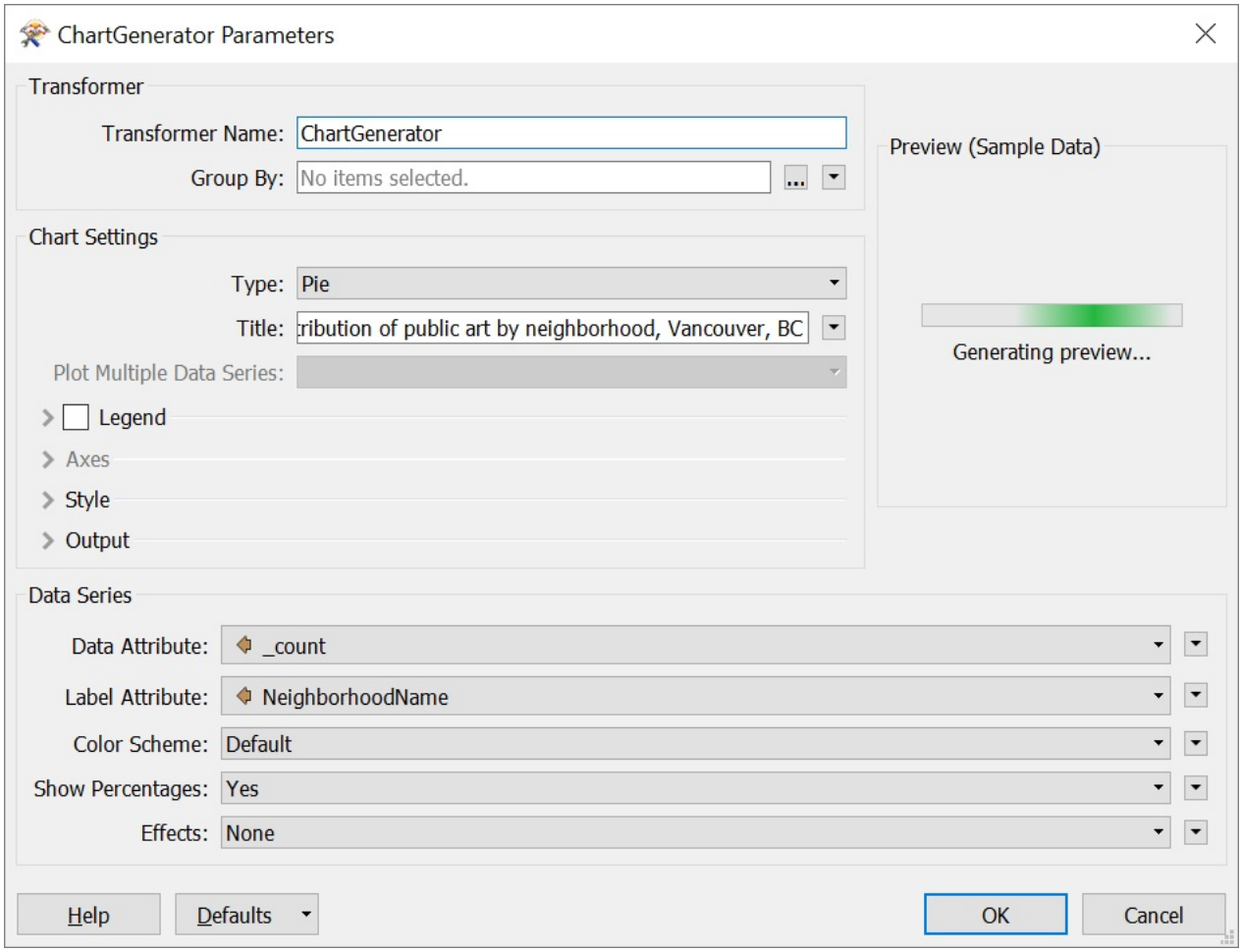


You can run the translation and inspect the table if you want. Note the presence of the new Stanley Park value.

12) Create Charts

Finally, lets create two charts to summarize our findings.

Add a ChartGenerator transformer to the canvas. Attach it to the Summary output port of your public art StatisticsCalculator. Open its parameters. Under Chart Settings, for Type select Pie. For Title enter: Distribution of public art by neighborhood, Vancouver, BC. Under Data Series, set the Data Attribute to _count and the Label Attribute to NeighborhoodName. Change Show Percentages to Yes. Your dialog should look like this:



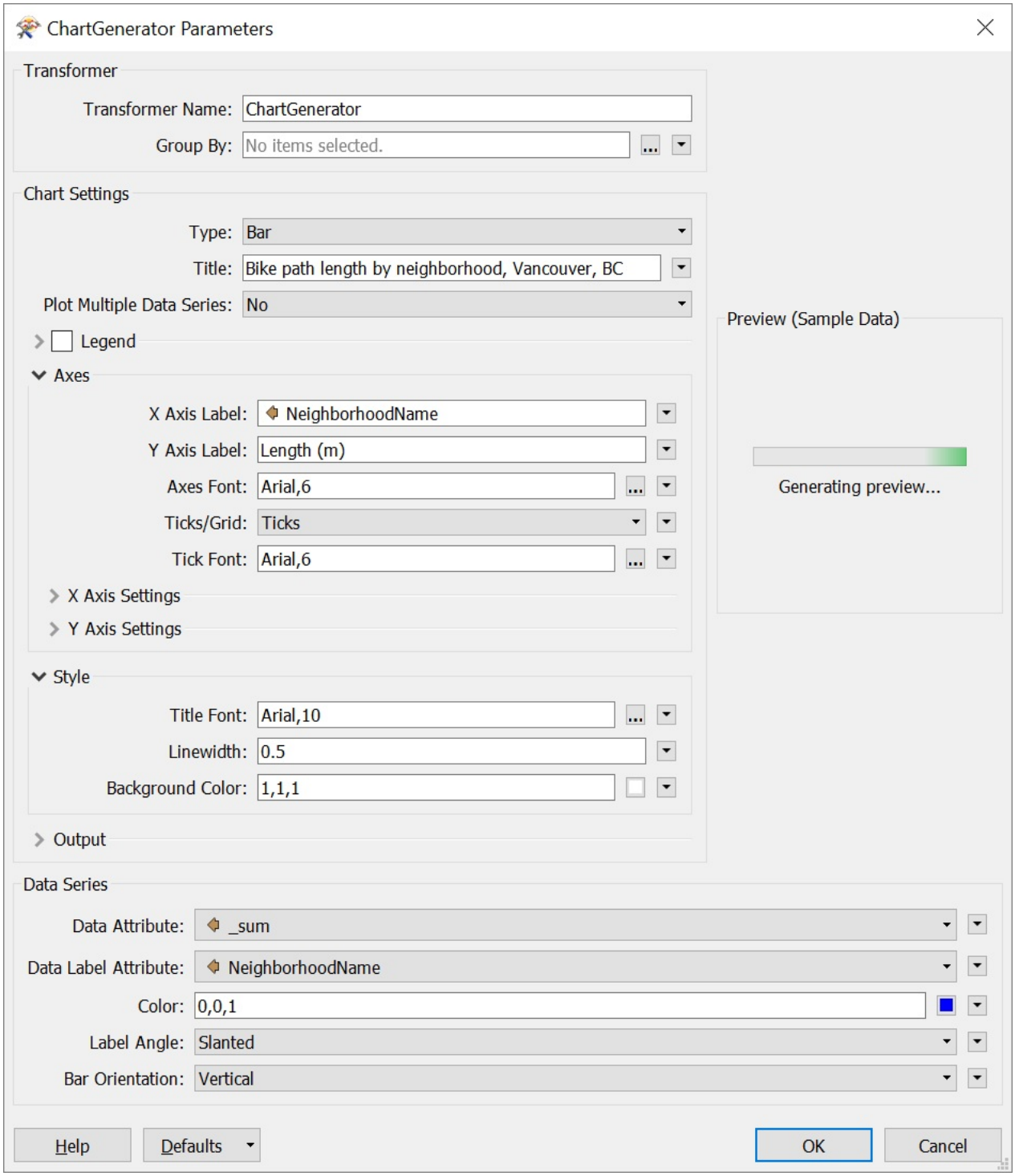
TIP

If you want to change the order the neighborhoods are displayed in for this or the bike paths chart, add a Sorter transformer before the ChartGenerator.

Add another ChartGenerator to the canvas, this time connected to the Summary output port of your bike paths StatisticsCalculator. Let's make this chart type Bar and title it: Bike path length by neighborhood, Vancouver, BC. Change the following parameters:

- Axes | X Axis Label: NeighborhoodName
- Axes | Y Axis Label: Length (m)
- Axes | Axes Font: Arial, 6
- Axes | Tick Font: Arial, 6
- Style | Title Font: Arial, 10
- Data Series | Data Attribute: _sum
- Data Series | Data Label Attribute: NeighborhoodName

This will generate a chart that shows the total length of bike paths in each neighborhood. Your dialog should look like this:



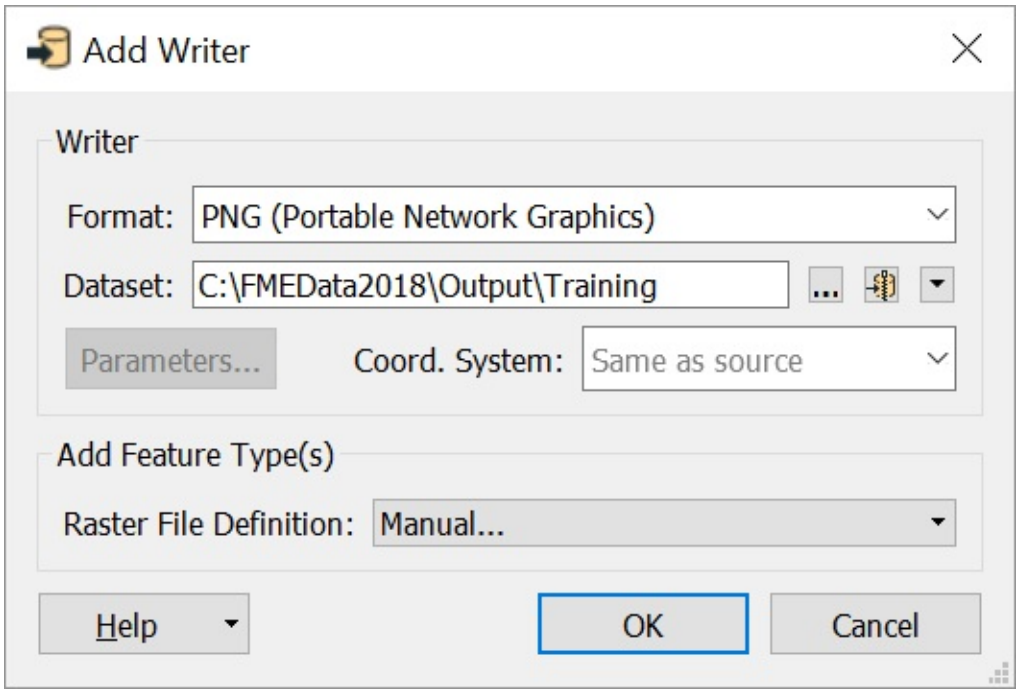
Click OK/Apply.

13) Write Charts to PNGs

Now let's hook these ChartGenerators up to writers so we can write the charts as images. Click Writers > Add Writer and use the following parameters:

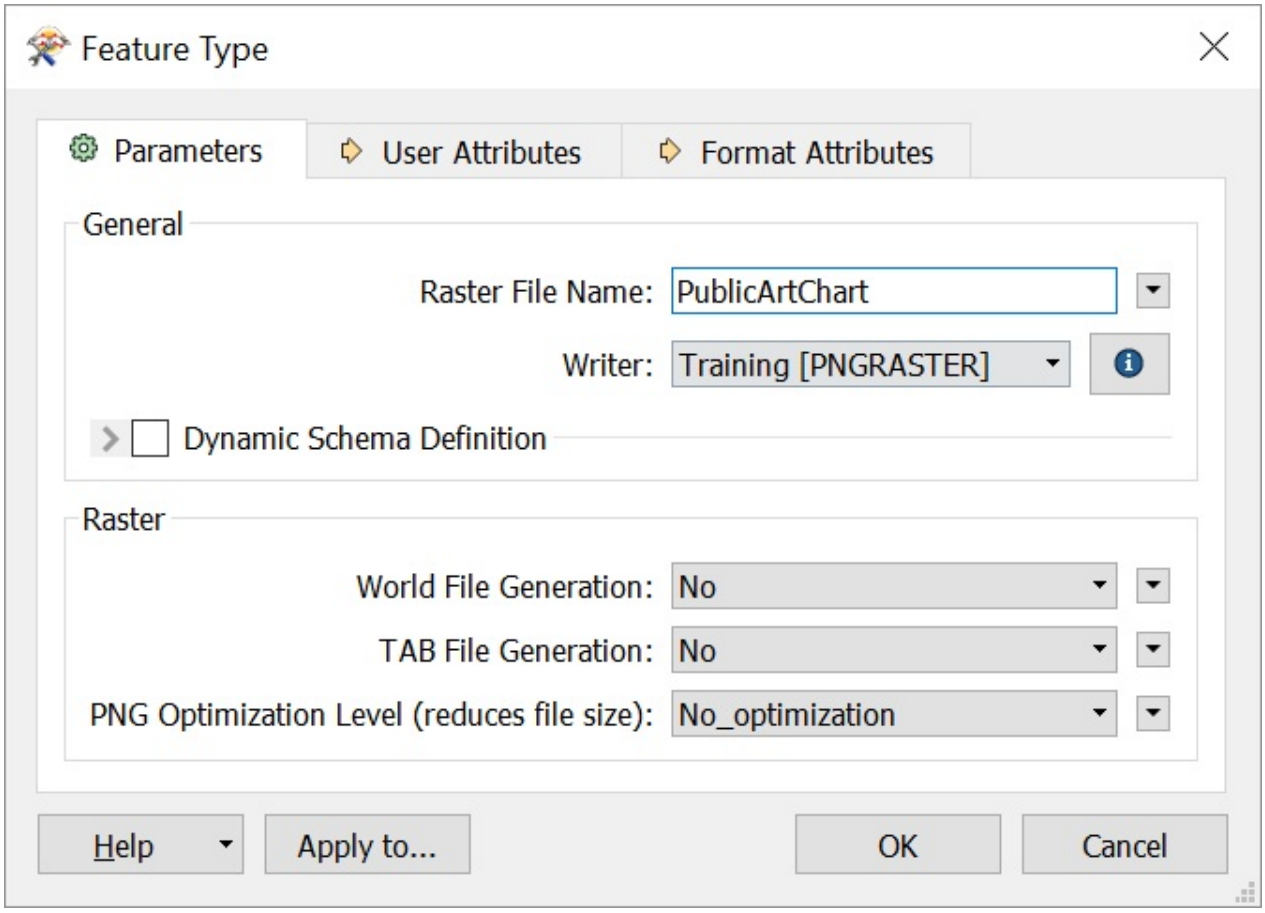
Writer Format	PNG (Portable Network Graphics)
Writer Dataset	C:\FMEData2018\Output\Training

For Add Feature Type(s) > Raster File Definition, choose Manual. We are choosing this because we don't want these chart images to map any schemas that already exist in our workspace. The dialog should look like this:



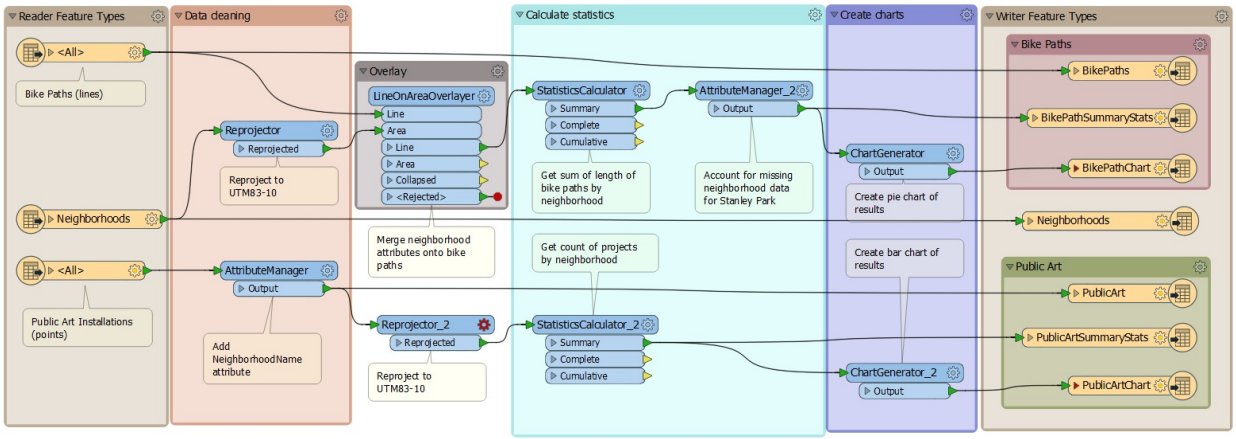
Click OK.

Another dialog will open to specify a feature type for our PNG writer. For Raster File Name enter PublicArtChart. Change Raster > World File Generation to No. Your dialog should look like this:



Click OK. Now connect your new writer feature type to the Output port of your public art ChartGenerator.

Repeat this process for a new feature type named BikePathChart: click Writers > Add Writer(s) and ensure Writer is set to PNGRASTER. Connect that to the bike path ChartGenerator. Since this is as far as we'll be going in this example, add bookmarks and/or annotations to explain your workspace (see the Style section if you need a reminder how to do this). After that your canvas should look something like this:



CONGRATULATIONS

By completing this exercise you have learned how to:

- *Integrate data into a single database using multiple feature types on a single writer*
- *Merge data using an Overlayer transformer*
- *Use ChartGenerator to make analyze integrated data*

Now that you have some practice integrating data, it's your turn! Use the skills you gained in the previous exercises to add one more dataset to the workspace to answer a question or solve a problem.

Here are some example questions:

1. How many addresses (C:\FMEData2018\Data\Addresses\Addresses.gdb) are within 100 meters of a bike path?
    - **Hint**: use the Bufferer and PointOnAreaOverlayer transformers. Don't forget to make sure all your data shares a coordinate system before analyzing it.
2. Where could the city locate a new public art installation? First find out which neighborhood has the fewest public art installations. Then find a city-owned property (C:\FMEData2018\Data\Parcels\CityProperties\CityProperties.shp) in that neighborhood that is the furthest away from existing public art installations. This is the site for a new installation.
    - **Hint**: use the Sorter, Tester, PointOnAreaOverlayer, NeighborFinder (check out the _distance attribute), and Sampler (check out Sampling Type: First N Features) transformers.
3. Do any city parks (C:\FMEData2018\Data\Parks\Parks.tab) not have access to bike lanes? If so, which ones? If not, which have the best and worse access?
    - **Hint**: use the Bufferer, LineOnAreaOverlayer, and/or NeighborFinder.

As a reminder, please refer to your lab requirements.

The next section contains optional advice on some of the procedures you may have to carry out depending on the data you choose.

Finally, don't forget to answer your lab questions.

Additional Procedures

While many datasets will work well using FME defaults, some formats require additional steps to control how they are read or analyzed. In this section we will walk through a few procedures that can help when working with some of the open Vancouver data. You can skip this section and go straight into your analysis if you want, referring back to this section if you run into any roadblocks.

Reading (X, Y) Coordinates as Points

If you add any tabular readers (*.csv,* .xlsx, etc.), you might notice they are imported without geometry. For example, adding the DrinkingFountains.csv file with default settings just adds features with string attributes. In order to be read as point geometry, FME must be told which columns contain the x and y coordinates of the data. There are two ways to do this:

1. When creating the reader, click Parameters and look at the Attributes table at the bottom of the window. The x_coord and y_coord columns are just being read as strings, not coordinates. To change this, change the Attribute Definition button from Automatic to Manual. Then use the dropdown in the Type column of the Attributes table to change x_coord to x_coordinate and y_coord to y_coordinate.
2. You could instead add the reader with default parameters and then attach a VertexCreator transformer to the reader. This creates point geometry from x and y coordinate attributes. For DrinkingFountains.csv you would open the VertexCreator parameters and for X Value select x_coord and for Y Value select y_coord.

Note: even if you have point geometry, some file formats like CSV do not store coordinate system information. You can refer to the City of Vancouver metadata to look up the coordinate system, or chances are it is stored as either latitude and longitude (LL84), UTM Zone 10N (UTM83-10), or BC Albers (BCALB-83 or NAD83.BC/Albers). Once you have identified the correct system you can set it under Navigator > Reader > Coordinate System, or under Coordinate System in the dialog box when you are adding a Reader, or by using the CoordinateSystemSetter transformer. Refer to Exercise 9 for more information.

Individual Versus Merged Feature Types

Some datasets contain more than one feature type. For example, if you add CommunityMap.gdb with the default settings you will be presented with a Feature Types To Read dialog. Here you can select which layers from the geodatabase you wish to read in as feature types.

When you have data like this you can choose to read the data as individual or merged feature types. This option appears only in the Add Reader dialog when you initially add a reader. It cannot be changed after the fact without removing feature types.

If you choose individual feature types each layer (or level, table, etc. depending on the format) will be added as a separate feature type. If you choose merged feature types the reader will combine all the layers into one feature type named <All>. This can be useful if you don't care about the distinction between layers and just want all the features grouped together.

Reading Attributes in Special Formats

Some formats do not store attributes in a straight-forward way. One example is CAD data. Because this format is designed for creating digital technical drawings, the features are not by default stored with attributes as in a GIS. However, they can store attributes as "extended entity data."

This has implications for reading CAD data. For example, if you add an Autodesk AutoCAD DWG/DXF reader and choose C:\FMEData2018\Data\Transportation\CompleteRoads.dwg, by default you will only read geometry. However, if you select Parameters > Group Entities By > Attribute Schema in the reader dialog, FME will read the geometry and associate it with each feature's extended entity data. This allows you to use these attributes in your analysis. You can see this in action if you inspect the reader with the default (Parameters > Group Entities By > Layer Name) and compare it to a reader with Parameters > Group Entities By > Attribute Schema. The former will have an empty column <no schema> in Table View and the latter will have a series of attributes.

There are other complicating factors in reading AutoCAD data. You can check out our Getting Started with AutoCAD series if you want to learn more.

Joining Features

By default, bringing two data streams together in FME **does not** merge data. Instead, data is combined or accumulated. You may know this as a union of data.

In order to merge data you have to carry out a join. There are two major kinds of joins in FME: key-based and spatially-based joins. Key-based joins connect two data sets based on a shared unique key, e.g. an ID number, a census tract ID, or a street address. Spatially-based joins connect two data sets based on spatial location, e.g. giving the attributes of census tract polygons to point locations of businesses that are contained by the tracts.

There are many transformers you can use in FME to carry out these operations. Here we will just show two of the most common join operations. For more information, you can refer to this joiner transformer article and flowchart.

Key-based joins with the FeatureJoiner

The FeatureJoiner transformer uses SQL join terminology to combine two data sets based on a shared unique key. The default Join Mode is an Inner join. For an inner join the data will be combined based on a common identifier, or attribute, from the left table and the right table.

Spatially-based joins with Overlayer transformers

FME comes with many Overlayer transformers:

- AreaOnAreaOverlayer
- LineOnAreaOverlayer
- LineOnLineOverlayer
- PointOnAreaOverlayer
- PointOnLineOverlayer
- PointOnPointOverlayer
- SurfaceOnSurfaceOverlayer
- VectorOnRasterOverlayer

These transformers overlay two data streams based on geometry. By default they simply count the number of overlaps in each stream and add a new attribute to both layers called _overlaps. However, if you change the settings under Attribute Accumulation > Merge Attributes you can combine attributes to join the data. For example, if you use VancouverNeighborhoods.kml as the polygon and DrinkingFountains.csv as the points (making sure they share the same coordinate system), and you check Merge Attributes, the polygons will get the attributes of the first point they overlap and the points will get the attributes of the neighborhood polygon they overlap. With an operation like this you can integrate data between layers.

Creating Charts and Reports

One easy way to show the results of your data integration is to create a visualization. FME has several built-in transformers that can help here. ChartGenerator can create charts. Simply add a ChartGenerator, pick your chart type, and select an attribute to analyze.

You can also easily create HTML reports using the HTMLReportGenerator. You can add elements to your page by clicking in the Page Contents table and adding a new piece of content from the dropdown menu. Each content type has its own options for you to fill in and can be dynamically created by using attributes. For example, if you have joined some data sets together using an Overlayer transformer and then calculated some average values using StatisticsCalculator, you can create a table of those values here. You can use Group By to split up your analysis by groups. Finally, don't forget to write the content of the HTMLReportGenerator out to an HTML writer. Once you do that you can open the resulting HTML file in a web browser.

Creating Raster Maps using MapnikRasterizer

Check out this blog post for FME cartography tips or this article on the Mapnik Rasterizer.

Creating Labels using LabelPointReplacer

We covered creating labels in Exercise 7. Refer to that exercise if you run into problems creating labels for your map. There are a few key points to remember:

- You can create labels combining multiple attributes using the StringConcatenator transformer.
- The label height is in the units of your data, so it depends on the coordinate system. You can check the linear units of your data by opening it in FME Data Inspector, selecting a feature, and looking in the Feature Information window. One of the attributes displayed is Coordinate System. Click on the link there (e.g. UTM83-10) and a Coordinate Systems Properties window with more information about the system opens. Here you can find the UNIT information, e.g. for UTM83-10, UNIT: METER.

Rejected Features?

You might encounter this error message in your log:

```
ERROR |{TRANSFORMER NAME}_<Rejected>: Termination Message: '{TRANSFORMER NAME} output
a <Rejected> feature.  To continue translation when features are rejected, change 'Wor
kspace Parameters' > Translation > 'Rejected Feature Handling' to 'Continue Translatio
n''
```

This means that one of your transformers rejected a feature (the message above was edited to use a placeholder `{TRANSFORMER NAME}`). Rejected features can sometimes mean something is wrong with your workspace, especially if a transformer is outputting only rejected features. However, in some cases rejected features are to be expected. For this reason FME gives you the option to decide how to handle rejected features. See this section of our training manual for more information.

Lab Questions

Please record your answers to these questions according to the direction of your instructor. Questions are provided in multiple choice or open-ended format. Answers for instructors may be requested here.

Multiple Choice Questions

Lecture

	Question
1	Data integration can be defined as... 1. discovery, cleansing, monitoring, transforming and delivery of data from a variety of sources. 2. using algebra to find the integral of a numeric attribute. 3. comparing predicted data to observed data. 4. using pivot tables/cross tabulation tables to find subtotals of grouped data.
2	What does ETL stand for? 1. Enumerate, Translate, List 2. Expand, Transform, Loop 3. Extract, Transform, Load 4. Exact Technical Loading
3	What distinguishes FME's ETL capabilities? 1. Low computer memory requirements 2. Multi-language support 3. Required coding knowledge 4. Ability to work with spatial data
4	In the terminology of FME, a *translation*... 1. translates data from one format to another. 2. translates scripts from one programming language to another. 3. translates string data from one language to another. 4. translates the geographic coordinates of data to a different coordinate system.
5	In the terminology of FME, a *transformation*... 1. transforms data from one format to another. 2. transforms data by moving it from one digital storage location to another. 3. transforms data by showing it on a map. 4. transforms data by changing the structure or content of the data.
6	The FME Data Inspector is a fully-featured GIS and cartography application. True or false? 1. True 2. False
7	What is the difference between *structural* and *content* data transformation? 1. Structural transformation performs statistical analysis, while content transformation uploads data to a new location on the web. 2. Structural transformation merges, divides, re-orders, or otherwise changes data structures, while content transformation changes the underlying values in a dataset. 3. Structural transformation makes long datasets wide and vice versa, while content transformation adds metadata to the dataset. 4. Structural transformation transforms the data into a different format, while content transformation performs spatial analysis on the data.

Lab

	Question
8	In Exercise 1, which library has the highest circulation? 1. Mount Pleasant 2. Kitsilano 3. Central Branch 4. Firehall
9	In Exercise 3, what is the value of ZoneName for the westernmost Industrial zone? *Hint*: you can try using the Filter function (Tools > Filter Features) of Data Inspector to narrow down your choices. 1. IC-1 2. IC-2 3. IC-3 4. I-1 5. I-2 6. I-3 7. M-1 8. M-2 9. M-3 10. M-4
10	In Exercise 4, what is the type of the attribute VisitorCount? 1. char 2. integer 3. smallint 4. float
11	In Exercise 4, what is the type of the attribute ParkArea? 1. char 2. integer 3. smallint 4. float
12	Complete the Advanced Exercise for Exercise 5. Who might benefit if you include bookmarks and annotation in your workspace? Select all that apply. 1. A client or customer provided with the workspace 2. A coworker who has to edit the workspace 3. Yourself in the future if you return to the workspace 4. The end-user of the data produced by the workspace
13	Complete the Advanced Exercise for Exercise 6. What are the names of the smallest and largest parks? 1. Smallest: Carolina Park; Largest: Stanley Park. 2. Smallest: Pioneer Place (Pigeon Park); Largest: Vanier Park. 3. Smallest: Pioneer Place (Pigeon Park); Largest: Stanley Park. 4. Smallest: Jean Beaty Park; Largest: Vanier Park.

14	In Exercise 7, how many parks have greater-than-average areas (i.e. have a ParkArea that is >= AverageParkArea)? How many are below average? *Hint*: use a Tester transformer after the StatisticsCalculator transformer. 1. 5 and 68 2. 14 and 96 3. 12 and 61 4. 4 and 69
15	In Exercise 8, what is the average size of parks in the Kitsilano neighborhood? 1. 24,469 m^2 2. 28,638 m^2 3. 23,986 m^2 4. 27,628 m^2
16	Complete the Advanced Exercise for Exercise 9. Assuming you want to calculate the park areas using the B.C. Albers coordinate system and units, you should place the Reprojector transformer after the AreaCalculator. True or false? 1. True 2. False

Data Integration Scenario

	Question
17	What percentage of Vancouver public art installations are located in the downtown neighborhood? 1. 84% 2. 47% 3. 66% 4. 62%
18	Which neighborhood has the longest total bike path length? 1. Downtown 2. Fairview 3. Kitsilano 4. Mount Pleasant 5. Stanley Park 6. Strathcona 7. West End
19	Why do the Bike Paths and Public Art Installation reader feature types have their names displayed as <All>? 1. The readers are in Merge Feature Type mode, reading all features in the dataset as a single feature type 2. The readers are reading all the files in the specified folder 3. The readers are set up to read all SHP and XLS files in the C:\FMEData2018\Data folder 4. We changed the name of the feature types on purpose to read <All>
20	How many public art installations are in the Mount Pleasant neighborhood? 1. 12 2. 13 3. 14 4. 16

Open-ended Questions

Lecture

	Question
1	Think of an existing business, organization, or technology that relies heavily on data (e.g. VRBO, the United Nation Statistics Division, or autonomous cars). What kind of data sources might it need to integrate? Describe three data sources, their formats, and what information they provide. How would data integration in this example relate to one of the nine reasons to integrate your data covered in the lecture?
2	What is spatial ETL?
3	In the terminology of FME, what is the difference between a *translation* and a *transformation*?
4	Is the FME Data Inspector a fully-featured GIS and cartography application? Why or why not?
5	What is the difference between structural and content data transformation?

Lab

	Question
6	What is another use case for this workspace in Exercise 1 if the data were different? What organization or business could benefit from a similar data integration workspace?
7	In Exercise 3, what is the value of ZoneName for the westernmost Industrial zone?
8	In Exercise 4, why is VisitorCount stored with Type “integer” and variable while ParkArea and AverageParkArea are stored with Type “float”?
9	Complete the Advanced Exercise for Exercise 5. Why might bookmarking your workspaces be considered a best practice for FME users? Best practice is defined by Merriam-Webster as “a procedure that has been shown by research and experience to produce optimal results and that is established or proposed as a standard suitable for widespread adoption.”
10	Complete the Advanced Exercise for Exercise 6. What are the smallest, largest, and total park areas? What are the names of the smallest and largest parks? Don’t forget to include units. Where can you confirm the data’s units?
11	In Exercise 7, how many parks have greater-than-average areas (i.e. have a ParkArea that is >= AverageParkArea)? How many are below average? *Hint*: use a Tester transformer after the StatisticsCalculator transformer.
12	In Exercise 8, what is the average size of parks in the Kitsilano neighborhood?
13	Complete the Advanced Exercise for Exercise 9. Where should the Reprojector transformer be placed in the workspace and why is this important?

Data Integration Scenario

	Question
14	What percentage of Vancouver public art installations are located in the downtown neighborhood?
15	Which neighborhood has the longest total bike path length?

16. Write a 200 word report on the data integration solution you created.

- Include the brainstorming diagram or outline you created before carrying out the walkthrough analysis. How close were you? What did you not anticipate?
- Which additional dataset did you use?
- What problem or question does your solution address?
- What analysis steps did you carry out?
- Provide a deliverable of some kind to show the value of your project. This could be a map (screenshot of Data Inspector is ok), a table, or a description of a new organizational process made possible by the solution.

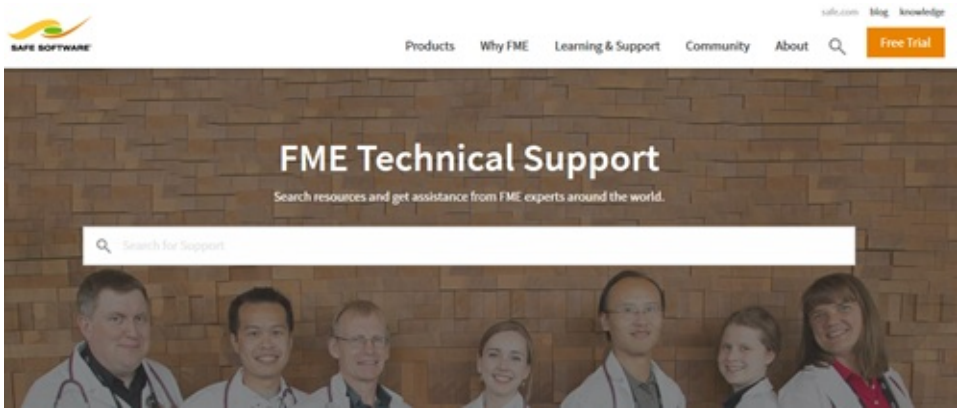
Product Information and Resources

Safe Software Web Site

The **Safe Software web site** is the official information source for all things FME. It includes information on FME products, Safe Software services, FME solutions, FME support and Safe Software itself.

Safe Support Team

Behind FME are passionate, fun, and knowledgeable experts, ready to help you succeed, with **a support team** philosophy built on the principle of knowledge transfer.



You can request product support through a Support Case (web/email) or using a Live Chat.

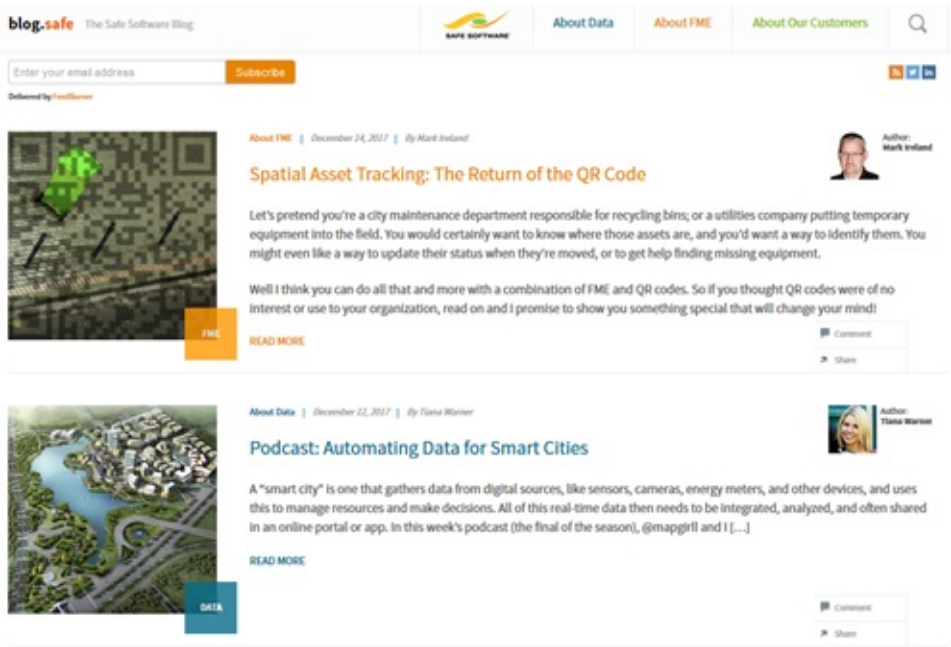
Your Local Partner

Safe Software has partners and resellers around the world to provide expertise and services in your region and your language.

You can find a list of official partners on the **Safe Software Partners Page**.

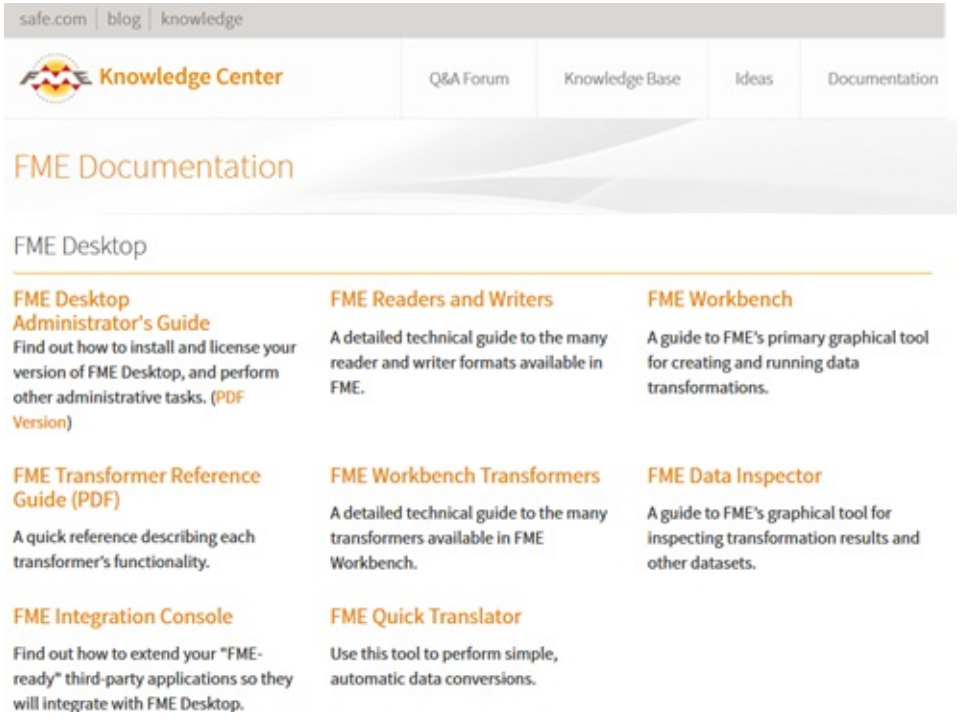
Safe Software Blog

The **Safe Software blog** provides technical information and general thoughts about FME, customers' use cases, and spatial data interoperability. It includes articles, videos, and podcasts.



FME Manuals and Documentation

Use the Help function in FME Workbench to access help and other documentation for FME Desktop. Alternatively, look on our website under the **Knowledge Center section**.



Community Information and Resources

Safe Software actively promotes users of FME to become part of the FME Community.

The FME Knowledge Center

The **FME Knowledge Center** is our community website - a one-stop shop for all community resources, plus tools for browsing documentation and downloads.

Knowledge Base

The FME Knowledge Base contains a wealth of information; including tips, tricks, examples, and FAQs. There are sections on both FME Desktop and FME Server, with articles on topics from installation and licensing to the most advanced translation and transformation tasks.

Q&A Forum

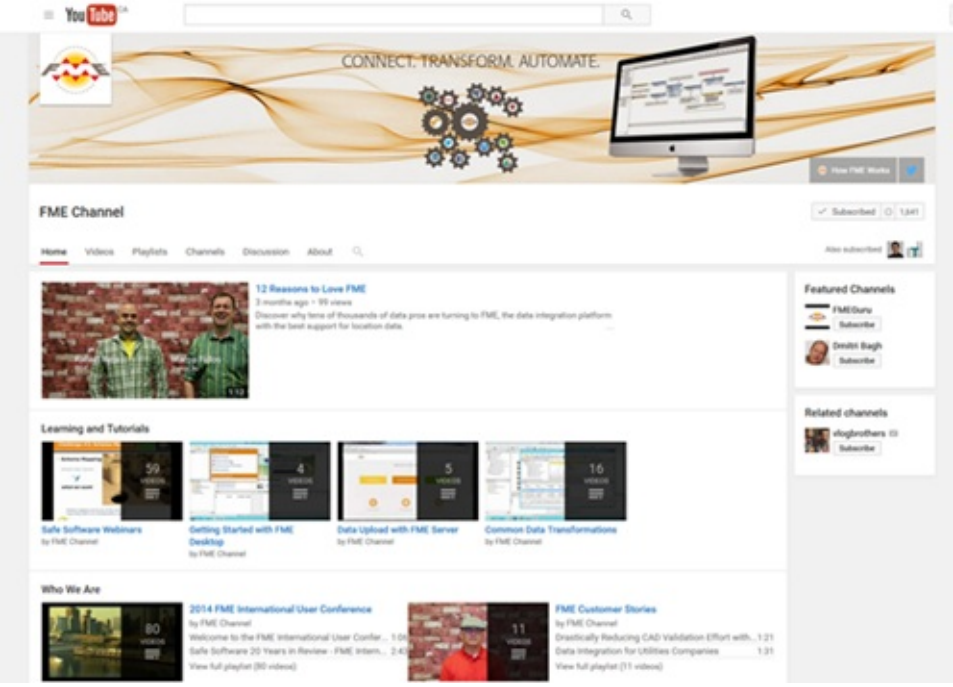
FME community members post FME-related messages, ask questions, and share in answering other users' questions. Members earn "reputation" and "badges," and there is a leaderboard of the top-participating users. Come and see how they can help with your FME projects!

Ideas Exchange

FME development is very much user-driven. The Ideas Exchange gives users the chance to post their ideas for new FME functionality, or improvements to existing functionality, and allows everyone to vote on the proposed ideas. The more votes an idea gets, the more likely it is to be implemented!

The FME Channel

This **FME YouTube channel** is for those demos that can only be properly appreciated through a screencast or movie. Besides this, there are a host of explanatory and helpful videos, including recordings of most training and tutorials.



Module Feedback

Your feedback would be greatly appreciated. Please fill out this quick survey to tell us about your experience with this module.

Feedback QR code