# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FME Server Authoring</td>
<td>1.1</td>
</tr>
<tr>
<td>Introduction</td>
<td>1.1.1</td>
</tr>
<tr>
<td>Course Overview</td>
<td>1.1.2</td>
</tr>
<tr>
<td>Course Resources</td>
<td>1.1.3</td>
</tr>
<tr>
<td><strong>Chapter 1: Introduction to FME Server</strong></td>
<td>1.2</td>
</tr>
<tr>
<td>What is FME Server?</td>
<td>1.2.1</td>
</tr>
<tr>
<td>FME Server Roles</td>
<td>1.2.2</td>
</tr>
<tr>
<td>FME Server Components</td>
<td>1.2.3</td>
</tr>
<tr>
<td>Workbench and FME Server</td>
<td>1.2.4</td>
</tr>
<tr>
<td>Publishing Workspaces</td>
<td>1.2.5</td>
</tr>
<tr>
<td>Exercise 1.1: Daily Database Updates: Publishing a Workspace</td>
<td>1.2.6</td>
</tr>
<tr>
<td>Web Interface Basics</td>
<td>1.2.7</td>
</tr>
<tr>
<td>Engines and Licensing</td>
<td>1.2.8</td>
</tr>
<tr>
<td>Running a Workspace</td>
<td>1.2.9</td>
</tr>
<tr>
<td>Jobs</td>
<td>1.2.10</td>
</tr>
<tr>
<td>Exercise 1.2: Daily Database Updates: Running a Workspace</td>
<td>1.2.11</td>
</tr>
<tr>
<td>Version Control</td>
<td>1.2.12</td>
</tr>
<tr>
<td>Exercise 1.3: Daily Database Updates: Adding to Version History</td>
<td>1.2.13</td>
</tr>
<tr>
<td>FME Server Scheduling</td>
<td>1.2.14</td>
</tr>
<tr>
<td>Exercise 1.4: Daily Database Updates: Scheduling</td>
<td>1.2.15</td>
</tr>
<tr>
<td>Sharing Repositories in FME Server</td>
<td>1.2.16</td>
</tr>
<tr>
<td>Exercise 1.5: Daily Database Updates: Sharing a Repository</td>
<td>1.2.17</td>
</tr>
<tr>
<td>Sharing Workspaces</td>
<td>1.2.18</td>
</tr>
<tr>
<td>Exercise 1.6: Daily Database Updates: Sharing a Workspace</td>
<td>1.2.19</td>
</tr>
<tr>
<td>FME Server Projects</td>
<td>1.2.20</td>
</tr>
<tr>
<td>Working With Projects</td>
<td>1.2.21</td>
</tr>
<tr>
<td>Sharing Projects</td>
<td>1.2.22</td>
</tr>
<tr>
<td>Exercise 1.7: Best Practice Workspace Analysis Project</td>
<td>1.2.23</td>
</tr>
<tr>
<td>Module Review</td>
<td>1.2.24</td>
</tr>
<tr>
<td>Q&amp;A Answers</td>
<td>1.2.25</td>
</tr>
<tr>
<td><strong>Chapter 2: FME Server and Data Handling</strong></td>
<td>1.3</td>
</tr>
<tr>
<td>Source Data Management</td>
<td>1.3.1</td>
</tr>
<tr>
<td>Database Connections</td>
<td>1.3.2</td>
</tr>
<tr>
<td>Web Connections</td>
<td>1.3.3</td>
</tr>
<tr>
<td>Publishing Data</td>
<td>1.3.4</td>
</tr>
<tr>
<td>Temporary Uploads</td>
<td>1.3.5</td>
</tr>
<tr>
<td>Exercise 2.1: Daily Database Updates: Publishing Data</td>
<td>1.3.6</td>
</tr>
</tbody>
</table>
Resource Filesystem 1.3.7
Authoring for Resources 1.3.8
Exercise 2.2: Daily Database Updates: Using Resources 1.3.9
Sharing Custom Resources 1.3.10
Module Review 1.3.11
Q&A Answers 1.3.12

Chapter 3: Self-Serve Foundations and Fundamentals 1.4
What is Self-Serve? 1.4.1
FME Server Services 1.4.2
Implementing Self Serve 1.4.3
Exercise 3.1: Data Download System: Workspace Creation 1.4.4
Other Self-Serve Services 1.4.5
Exercise 3.2: Data Streaming System 1.4.6
Self-Serve and Parameters 1.4.7
Published Parameters 1.4.8
Key Parameter Types 1.4.9
Geographic Area of Interest 1.4.10
Exercise 3.3: Data Download System: Published Parameters 1.4.11
Module Review 1.4.12
Q&A Answers 1.4.13

Chapter 4: Real-Time with FME Server 1.5
Automations 1.5.1
Elements of Automations 1.5.2
Automation Clients 1.5.3
Triggers and Actions 1.5.4
Automation Protocols 1.5.5
Protocol Details 1.5.5.1
Exercise 4.1: Building Updates Automation: Directory Watch Trigger 1.5.6
Automations and Workspaces 1.5.7
Workspaces as Actions 1.5.8
Incoming Message Handling 1.5.9
Exercise 4.2: Building Updates Automation: Run Workspace 1.5.10
Email Automations 1.5.11
Email Triggers: SMTP 1.5.11.1
Email Triggers: IMAP 1.5.11.2
Email Actions 1.5.11.3
Exercise 4.3: Building Updates Automation: Email Trigger 1.5.12
Workspace as Triggers 1.5.13
Automations Writer 1.5.13.1
Workspace Transformers as a Trigger 1.5.13.2
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace as Both Trigger and Action</td>
<td>1.5.14</td>
</tr>
<tr>
<td>Exercise 4.4: Building Updates Notification System: Outgoing Messages</td>
<td>1.5.15</td>
</tr>
<tr>
<td>Job Chains and Automations</td>
<td>1.5.16</td>
</tr>
<tr>
<td>Exercise 4.5: Authoring Workspace Chains in Automations</td>
<td>1.5.17</td>
</tr>
<tr>
<td>The Notification Service</td>
<td>1.5.18</td>
</tr>
<tr>
<td>Troubleshooting for Administrators</td>
<td>1.5.19</td>
</tr>
<tr>
<td>Module Review</td>
<td>1.5.20</td>
</tr>
<tr>
<td>Q+A Answers</td>
<td>1.5.21</td>
</tr>
<tr>
<td>Chapter 5: Message Streams</td>
<td>1.6</td>
</tr>
<tr>
<td>Elements of a Message Stream</td>
<td>1.6.1</td>
</tr>
<tr>
<td>Message Streaming Architecture</td>
<td>1.6.2</td>
</tr>
<tr>
<td>Exercise 5.1: Handling Emergency Phone Call Streams</td>
<td>1.6.3</td>
</tr>
<tr>
<td>Module Review</td>
<td>1.6.4</td>
</tr>
<tr>
<td>Q+A Answers</td>
<td>1.6.5</td>
</tr>
<tr>
<td>Course Wrap-Up</td>
<td>1.7</td>
</tr>
<tr>
<td>Product Information and Resources</td>
<td>1.7.1</td>
</tr>
<tr>
<td>Community Information and Resources</td>
<td>1.7.2</td>
</tr>
<tr>
<td>Course Feedback</td>
<td>1.7.3</td>
</tr>
<tr>
<td>Thank You</td>
<td>1.7.4</td>
</tr>
</tbody>
</table>
Workspace Authoring for FME Server Training Manual

This is the manual for the training course Workspace Authoring for FME Server.

This training provides a framework for authoring workspaces for FME Server. We hope that you will learn all the tools upon which to base your work, and go home with many new FME ideas!

Course Structure

The full course is made up of five main sections. These sections are:

- Introduction to FME Server
- Data Handling and FME Server
- Self-Serve Basics
- Real-Time with FME Server
- Real-Time Message Handling

Current Status

The current status of this manual is: COMPLETE. This manual can be used for training.

This manual applies to FME2020.1

The status of each chapter is:

- Chapter 0: Complete content. No exercises
- Chapter 1: Complete content and exercises
- Chapter 2: Complete content and exercises
- Chapter 3: Complete content and exercises
- Chapter 4: Complete content and exercises
- Chapter 5: Complete content and exercises
- Chapter 6: Complete content and exercises
- Slides: Complete
- FMEData: Complete
- Course Outline: 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 for further details.
About This Document

This manual is the introductory-level training course for authoring translations for FME Server.

FME Lizard says...

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. See www.openstreetmap.org/copyright.

Copyright

© 2005–2020 Safe Software Inc. All rights are reserved.

Revisions

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

Safe Software Inc.
Phone: 604-501-9985
Fax: 604-501-9965
Email: train@safe.com
Web: www.safe.com

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

Trademarks

FME® 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 Server Authoring Training Manual 2020.1

All screenshots relate to FME Desktop and FME Server 2020.1; This manual has been tested with FME Desktop and FME Server 2020.1 Build 20596.

What's New?
A list of changes to this manual and its accompanying datasets can be found on GitHub. The file includes a list of general revisions compared to the previous year's materials. It is designed to help trainers become up-to-speed with new content, and for students to identify which FME functionality is new for the current release.
Course Overview

This training material covers how to create FME translation models for use on FME Server.

The training will introduce basic concepts and terminology, help you become an efficient FME Server user, and direct you to resources to help apply the product to your own needs.

Course Structure

The full course is made up of five main sections. These sections are:

- Introduction to FME Server
- Data Handling and FME Server
- Self-Serve Basics
- Real-Time with FME Server
- Real-Time Message Handling

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 course content in a different order and will skip or add new content to better customize the course to your needs.

Therefore the length and content of the course may vary, particularly when delivered online.

Prerequisites

This training material is intended for persons with some prior experience of using FME. It assumes a basic familiarity with the concepts and practices covered of FME Desktop; at the very least to the extent covered by the FME Desktop Tutorial.

In particular it would be helpful to be familiar with:

- Parameters and published parameters
- Managing Readers, Writers and feature types in a workspace
- Using transformers similar in complexity to the Clipper and Joiner

FME Lizard says...

A further prerequisite for carrying out some exercises in the Real Time section is an email account that can be accessed through IMAP, for example a Google Gmail account. If you don't have one of these, please set one up before getting to that section of the manual.

About the Manual

The FME Server authoring training manual not only forms the basis for FME Server training – in-person or online – but is also useful reference material for future work you may undertake with FME.

This training material is designed specifically for use with FME 2020.1. You may not have some of the functionality described if you use an older version of FME.
Course Resources

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

On Your Training Computer

The following applications may already be installed, licensed, and located on your training computer (real or virtual):

- Apache Tomcat
- FME Desktop Version 2020.1
- FME Server Version 2020.1
- Microsoft SQL Server

The data used in this training course 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.

Whether it's a local computer or a virtual computer hosted in the cloud, you'll find resources for the examples and exercises in the manual at the following locations:

<table>
<thead>
<tr>
<th>Location</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:\FMEData2020\Data</td>
<td>Datasets used by the City of Interopolis</td>
</tr>
<tr>
<td>C:\FMEData2020\Resources</td>
<td>Other resources used in the training</td>
</tr>
<tr>
<td>C:\FMEData2020\Workspaces</td>
<td>Workspaces used in the student exercises</td>
</tr>
<tr>
<td>C:\FMEData2020\Projects</td>
<td>FME Server projects used in the student exercises</td>
</tr>
<tr>
<td>C:\FMEData2020\Output</td>
<td>The location in which to write exercise output</td>
</tr>
<tr>
<td>&lt;Documents&gt;\FME\Workspaces</td>
<td>The default location to save FME workspaces</td>
</tr>
</tbody>
</table>

You should also find a digital copy of this manual.

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

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.

Course Etiquette

For online courses, please consider other students and test your virtual machine connection before the course starts. The instructor cannot help debug connection problems during the course!

For live courses, please respect other students’ needs by keeping noise to a minimum when using a mobile phone or checking e-mail.
Introduction to FME Server

FME Server is a powerful product that lets you automate the flow of data between your applications; not to mention any mindless data tasks you're forced to do each day!

This chapter introduces you to FME Server, its interface, and explains how to run a workspace using it.
What is FME Server?

FME Server is a powerful product for handling large volumes of data at the enterprise level. It has three core capabilities: self-serve, real-time, and automation.

Self-Serve

Self-serve is the ability for the end-user to select and download the data they require, in the format and structure they require, or to upload data for processing. It eliminates the need for a data manager/analyst to carry out manual data distribution tasks.

Real-Time

Real-time is the ability to react to real-time events and sensors, to carry out immediate updates, and to deliver instant notifications. It allows subscribers to have the most up-to-date information for their business decision making.

Automation

Automation is the ability to carry out data processing on a specific schedule and to spontaneously move data through different systems and web services – even onto mobile platforms and devices. It allows data to move from anywhere, to anywhere, without manual intervention.

FME Server vs. FME Desktop

FME Server expands on the local processing tools of FME Desktop by providing the same core data translations and transformations, but at an enterprise level, and with the above enterprise-level capabilities.

With these capabilities, FME Server can take advantage of a huge variety of different communication technologies. This means it can provide many different ways to:

- Run a workspace
- Structure a translation
- Deliver the output
FME Server has limited editing capabilities (changing services and modifying published parameters), but cannot create workspaces, so FME Desktop is used to create and edit workspaces.

The focus of this course is how to create these workspaces in ways that will take full advantage of FME Server’s enterprise capabilities.
FME Server Roles

When using a product with the range and scope of FME Server, it’s no surprise that there are a number of different roles available to different users. Each different role usually has different interests in FME Server.

Author

An author is someone who creates translations (workspaces) using FME Desktop and publishes them to FME Server for use by end-users. This particular role is the focus of these training materials.

Typically an author would work at the analyst level and would be an experienced FME Desktop user with a good understanding of readers, writers, transformers, and other FME Desktop functionality.

User

A user is defined as a person who accesses data using an FME Server Service.

It is not expected that a user in this sense has, or needs to have, any experience of FME and does not even need to be aware of FME Desktop or FME Server.

The user might be someone like a CAD operator checking out spatial data, a business manager looking at corporate data in an FME Server client, or members of the public who want to download data for personal use.

Developer

FME Server can be used as the back-end to power other software applications.

A developer (in FME Server terms) is someone who creates applications that submit jobs to FME Server and then handle the results. This role is minimally covered by these training materials.

Web developers may choose to include FME Server Web Services within their own web applications or create their own services using the FME Server API.

Administrator

An FME Server administrator is the person responsible for installing and maintaining FME Server and its related services.

The administrator’s tasks include:

- Planning system architecture
- Installing prerequisite applications
- Installing and licensing FME Server
- Managing security
- Setting up FME Server services
- Monitoring FME Server services
- Troubleshooting
- Scaling FME Server
TIP

Safe Software offers various live-online and recorded courses for the authoring, developer and administration roles. If you are interested in learning more about these roles please see our online training catalog.
FME Server Components

FME Server is made up of several different components that work together to make it function. As an FME Server Author, you don't need to have a deep understanding of its full architecture. (If you are interested, the FME Server Administration Training course takes a deep dive into architecting FME Server).

There are a few components of FME Server that are important for you as an author to be aware of, so we're going to cover those now:

- **FME Engines**: Carry out data transformation processing
- **Server Core**: Queue jobs, handle scheduling, and manage load balancing
- **Web Services**: Handle requests made to FME Server

### FME Engines

FME Engines process job requests by running FME Workspaces. This is the same core engine, carrying out the same processing that is used by FME Desktop. An FME Server installation can possess multiple engines.

Each FME Engine processes a single request (job) at a time.

FME Server processing can be scaled by connecting additional FME Engines to the Server Core. These FME Engines can run on the same computer as the Core or on separate computers within a distributed FME Server environment.

### Server Core

The FME Server Core manages:

- FME workspaces
- Job management
- Job scheduling
- Notifications
- Automations
- Repository contents (workspaces, custom formats, custom transformers, data)
- Software Load Balancer
- WebSocket server
- Licensing
- Engine manager
- FME Server queue

The FME Server Core contains a Software Load Balancer (SLB) that distributes jobs to FME Engines.
Web Services

Much of the FME Server networking capabilities are handled using what we call "Web Services." These Web Services are software whose interface provides communication between FME Server and clients.

FME Server has a number of services:

- Data Download
- Data Streaming
- Job Submitter
- KML Network Link
- Notification
- Data Upload
- Token Security
- REST

Some services (for example, Data Download) are “transformation” services that carry out data transformation, whereas others (for example, Data Upload) are non-transforming "utility" services.
FME Workspaces and FME Server

FME Server has a model-driven architecture because its processes are expressed as models. In FME, these models are better known as workspaces.

Workspaces are created – we call it “authored” – using FME Desktop. In particular, the FME Workbench application is used. FME Workbench is a client of FME Server, and so they form a client-server pair. However, both share the same core engine and process data in the same way.

FME Lizard says...

*Let’s make sure you get the terminology right. FME Desktop is the desktop program that includes the FME Workbench, FME Data Inspector, and FME Quick Translator applications. Workspace authoring occurs in the application called FME “Workbench,” but the process defined in the canvas window of FME Workbench is called a “Workspace.” The terms are so similar that they are easily confused.*

Because FME Workbench is a client of FME Server, it may be used to transfer authored workspaces to and from FME Server. We call this transfer publishing.

FME Workbench has the ability to:

- Author a translation workspace
- Publish a workspace (transfer it to FME Server)
- Republish a workspace (upload a previously published workspace)
- Download a workspace (retrieve it from FME Server)

The ability to transfer a workspace back to FME Workbench means workspaces can be downloaded for editing and maintenance, then published back to FME Server.

Repositories

Workspaces are stored on FME Server in devices called repositories. Each FME Server may have multiple repositories, but any workspace can only belong to one of them, without creating a duplicated workspace.

A repository consists of two parts:

1) The .fmw files from any workspaces that have been published to FME Server are stored in the file-based part of the repository.
2) Metadata related to the workspace is held separately in the FME Server Database. This metadata includes information about the contents of the workspace; for example, source and destination datasets, workspace feature types, and published parameters.
Repositories are managed by the FME Core. They can be accessed (by authors and administrators) through the FME Server Web Interface.

FME Lizard says...

Security in FME Server is very important, and never more so than for repositories.

You can think of each repository as being like a folder on a file-system, with the same ability to grant access rights to individuals and groups. So, for each repository you create, be sure to check the security permissions. If you don’t, then end-users may not get access to the repository!
Transferring Workspaces

The functionality for publishing or downloading workspaces is accessed in FME Workbench either through the menubar:

![Menu Bar Snippet]

...or the toolbar:

![Toolbar Snippet]

Workspaces can also be uploaded and downloaded directly through FME Server on the Workspaces page:
Connecting to FME Server

The Publish to FME Server tool in FME Workbench opens a simple wizard interface, the first dialog of which defines a connection to FME Server.
Adding a web connection opens a dialog with fields in which to define connection credentials. These connection details are saved so that they can be reused in the future simply by picking from the drop-down list:

Repository Selection

The next dialog defines the repository in which to store the workspace:

Either an existing repository can be used, or a new one created. The workspace name can also be edited, even making it different to what it is saved as locally.

Connections Upload

This dialog only appears when there are databases and/or web connections that need to be uploaded with the workspace.
This workspace contains both a database connection and a web connection that need to be uploaded to function on FME Server. Note that the Dropbox OAuth web connection needs the service to be added to FME Server, and will also require additional authentication before FME Server will be able to use it.

The database connection requires no further authentication, but care must be taken not to accidentally overwrite an existing database connection with the same name that might already be defined on FME Server.

Workspace Registration

The final dialog defines which service(s) the workspace is to be registered against. A workspace may be registered for use with any number of these services.

The Job Submitter service allows FME Server to run a workspace as-is. This is the closest to running a workspace in FME Workbench. All inputs and outputs are defined in the workspace, so data is simply written out and not streamed or delivered in any other manner.

Job submission is ideal for testing workspaces, writing to databases, and for running large-scale and batch translations that make use of the server process queue.

Republishing a Workspace

Once a workspace has been published, the republish tool becomes active. Further updates to the workspace (within the same FME Workbench session) can then be uploaded with a single click.
The same parameters are used as before. If changes need to be made to these parameters, then the full publishing wizard should be used.

**Downloading a Workspace**

Workbench can also download a workspace held in an FME Server repository. This is usually done in order to make edits to the workspace. Note that downloaded workspaces are copies of the original, which remains in the FME Server repository.

The downloading wizard begins with the same connection dialog as the publishing wizard. From there, the second – and final – dialog page is a repository and workspace selection tool:

The user is then prompted for a location to save the workspace. The default (on Windows) is `<User>\Documents\FME\Workspaces`. The workspace – and any resources – are then downloaded and saved to that location.

Once downloaded, the workspace is automatically opened within FME Workbench for editing.

---

**FME Lizard says…**

_Besides workspaces, it's also possible to publish/download FME custom transformers and custom formats to and from an FME Server repository._
For the exercises in this chapter, you are a technical analyst in the GIS department of your local city. You have plenty of experience using FME Desktop, and your department is now investigating FME Server to evaluate its capabilities.

There are many departments within the city, and one of your tasks is to take the data from each department and merge it together into a single corporate database.

Because each department produces its datasets in a different format and style, you use FME for this task and carry it out on a weekly basis.

One of the reasons for purchasing FME Server is to automate this procedure, so let's start implementing that.

### FME Lizard says...

*If you have lots of experience with FME Workbench - and if your instructor agrees - simply open the workspace listed in the header above and skip to step 8.*

1) **Start FME Workbench**

   Since this course is focused on FME Server, we will create a simple workspace in FME Desktop using a couple of formats.

   Start FME Workbench by selecting it from the Windows start menu. On a blank canvas, select Readers > Add Reader to start adding a reader to the workspace. When prompted, enter the following details for the GML FireHalls dataset:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>GML (Geography Markup Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\Emergency\FireHalls.gml</td>
</tr>
</tbody>
</table>

2) **Add KML Data**

   Now repeat the process one more time to add a reader for the KML Neighborhoods dataset:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>Google KML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\Boundaries\VancouverNeighborhoods.kml</td>
</tr>
</tbody>
</table>

   While adding the KML dataset, you'll be prompted to choose which Feature Types (layers) to add to the workspace. The only one we need is called Neighborhoods:
You should now have two readers on the canvas:

3) Inspect the Data
Before we proceed, we should inspect the data. Click on the FireHalls reader feature type to open the popup menu. On the popup menu, click on View Source Data, this will open the data in Visual Preview:

While viewing the data in Visual Preview, take note of the coordinate system in the bottom right corner of the Graphics window. Since we are working with more than one dataset, we will want to make sure that all of the datasets are in the same coordinate system:
Repeat the same steps to view the Neighborhoods reader feature type, and take note of the coordinate system.

4) Add Reprojector Transformer
The FireHalls dataset is in EPSG:26910, which is also known as UTM83-10, and the Neighborhoods is in LL84. Let's reproject the Neighborhoods dataset to UTM83-10, as it is a coordinate system more suited for this region.

Add a Reprojector transformer to the workspace. You can do this by simply clicking on the canvas and starting to type Reprojector. Connect it to the Neighborhoods feature type:

![Reprojector transformer connected to Neighborhoods feature](image)

In the Reprojector's parameters set the Destination Coordinate System to UTM83-10:

![Reprojector parameters](image)

This will ensure the Neighborhoods data is in the same coordinate system as the rest of the data.
5) Add Writer
Now we should add a writer to the workspace. For now, we'll just set up a dummy writer until we are more familiar with FME Server. To do this, select Writers > Add Writer on the menu bar to add a writer and set it up with the following parameters:

<table>
<thead>
<tr>
<th>Writer Format</th>
<th>NULL (Nothing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Class or Table Definition</td>
<td>Copy from Reader...</td>
</tr>
</tbody>
</table>

Click OK. When prompted, select both FireHalls and Neighborhoods as the feature types to add and OK again:

The workspace will now look like this:

6) Add Clipper Transformer
Add a Clipper transformer to the workspace. This will be used to divide the FireHall data by Neighborhood.

Connect the FireHalls feature type to the Clipper:Clippee port and the Reprojector:Reprojected output to the Clipper:Clipper port. You may wish to rearrange the feature types (or the port order) to avoid overlapping connections; this is done by right-clicking on the Clipper:Clipper port and selecting Move Down:
In the Clipper parameters, enable Merge Attributes, so that the neighborhood name is copied from the Neighborhood features to the FireHall features:

Connect the Clipper:Inside port to the FireHalls writer feature type. Also make a connection from the Reprojector:Reprojected port to the Neighborhoods writer feature type:
7) Set Firehall Feature Type Name
Finally, let's set the Feature Type Name for the FireHalls writer feature type.

Inspect its parameters and under Feature Type Name either enter:

FireHalls-@(Value(NeighborhoodName))

...or click the drop-down and use the text editor dialog to enter that value. This will cause firehalls in each different neighborhood to be written to a different table/layer.

8) Run Workspace
Here comes the Server part of the process.

First, save the workspace. It is always a good idea to save the workspace before publishing to FME Server. Next, ensure that Prompt for User Parameters is disabled, then run the workspace. If the workspace won't run on FME Desktop, then it is not likely to run on FME Server. If you get the Unexpected Input dialog, it is safe to ignore it.

Once the workspace has been run, inspect the translation log. Your translation log should look like the one below:

<table>
<thead>
<tr>
<th>Features Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>FireHalls-Downtown</td>
</tr>
<tr>
<td>FireHalls-Fairview</td>
</tr>
<tr>
<td>FireHalls-Kitsilano</td>
</tr>
<tr>
<td>FireHalls-Mount Pleasant</td>
</tr>
<tr>
<td>FireHalls-Streethcona</td>
</tr>
<tr>
<td>FireHalls-West End</td>
</tr>
<tr>
<td>Neighborhoods</td>
</tr>
<tr>
<td>Total Features Written</td>
</tr>
</tbody>
</table>
9) Publish to Server: Create Connection

Now we have a workspace and know that it works correctly, let's publish it to FME Server.

In FME Workbench, choose File > Publish to FME Server from the menubar. As this is the first time we've connected to our FME Server, we'll need to create a new connection, so in the Publish to FME Server wizard select Add Web Connection from the drop-down menu:

![Publish to FME Server dialog]

**WARNING**

As of FME 2019, the admin password is required to be set up after install. For this course, our administrator has chosen the password FMElearnings. If you are taking this course outside of Safe Software, please consult your FME Server Administrator for the admin password on their training machines.

In the dialog that opens, enter the parameters provided by your training instructor. In most cases, the parameters will be as follows:

- **Connection Name**: Training FME Server
- **FME Server URL**: http://localhost
- **Username**: admin
- **Password**: FMElearnings
You may or may not (probably not) need to enter a port number with the hostname, depending on how the system is set up.

Click Authenticate to confirm the connection and return to the previous dialog. Make sure the newly defined connection is selected and click Next to continue.

10) Publish to Server: Repository Selection

The next dialog prompts you to choose a repository in which to store the workspace.

For this exercise, we’ll create a new repository by clicking the New button. When prompted, enter the name Training.

Click OK to close the Create New Repository dialog. Enter a name for the workspace if it doesn't already have one. Place a checkmark against the Upload Data Files option:
Then click Next to continue the wizard.

11) Publish to Server: Select Service
In the final screen of the wizard, we can register the workspace for use with various services.

Select the Job Submitter service as this is the only service we are using for now:

... and click Publish to complete publishing the workspace.

After a workspace is transferred to Server, the log window displays a message reporting which workspace has been published to which repository and for which services. It will look something like this:
CONGRATULATIONS

By completing this exercise you have learned how to:

- Create a workspace converting data to a Null (dummy) format
- Use a Clipper to transfer attribute values from one feature to another
- Rename output layers according to the value of an attribute
- Create a repository on FME Server using the Publishing Wizard
- Publish a workspace to FME Server using the Publishing Wizard
- Register a workspace with the Job Submitter service using the Publishing Wizard

---

Publish Summary

<table>
<thead>
<tr>
<th>FNX Server URL</th>
<th><a href="http://localhost">http://localhost</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>admin</td>
</tr>
<tr>
<td>Repository</td>
<td>Training</td>
</tr>
<tr>
<td>Name</td>
<td>Ch1-Ewi-Complete.fmx</td>
</tr>
<tr>
<td>Direct Link</td>
<td><a href="http://localhost/fmeserver/t/workspaces/run/Training/Ch1-Ewi-Complete.fmx">http://localhost/fmeserver/t/workspaces/run/Training/Ch1-Ewi-Complete.fmx</a></td>
</tr>
<tr>
<td>Uploaded Resources</td>
<td>C:\FMEData2020\Data\boundaries\vancouverneighborhoods.kml</td>
</tr>
<tr>
<td></td>
<td>C:\FMEData2020\Data\Emergency\firehalls.xsd</td>
</tr>
<tr>
<td></td>
<td>C:\FMEData2020\Data\Emergency\firehalls.gml</td>
</tr>
<tr>
<td>Registered Services</td>
<td>Job Submitter</td>
</tr>
<tr>
<td>Time</td>
<td>Tue Jul 21 10:04:30 2020</td>
</tr>
</tbody>
</table>
Introduction to the FME Server Web Interface

Although translations are authored in FME Workbench, the core tools of FME Server are accessed through a web-based interface.

The web interface is accessed through the URL http://<servername>:<port>/fmeserver (the port may be optional) or through the start menu:

The web interface for FME Server looks like this:

The main part of the interface displays page information. The landing page, for example, has shortcuts to lists of recent jobs, projects, automations, and favorite workspaces. What is displayed on an individual user's home page is customizable.

The left-hand side of the interface is the side menu. Selecting a menu item changes the content of the page to match the menu item chosen. Additionally, the menu will change depending on the privileges the user has.
Web Interface Menu

In general, FME Server functionality is accessed through the web interface menu. There are two main sections in this menu:

The first section relates to the use of FME Server. It has - among others - options for running a workspace, accessing repositories, setting up schedules, and reviewing job history.

The next section of the menu relates to the administration of FME Server. It has - among others - options for managing engines, setting up security, and creating system backups.

NEW

In 2020.0, we have condensed the side menu. Resources can now be found under Files & Connections, and Repositories can now be found under Workspaces > Manage Workspaces.

There are a couple of additional menus located in the top-right part of the interface:
The first provides access to help tools for authors, users, administrators, and developers. And the second provides options for managing your user account options.

NEW

We want you to get the most out of FME Server, so we've added more ways for you to learn FME Server. Under the Help menu, check out FME Server in action with a demo, or take a quick tour, or even ask a specific question in our community.
Engines and Licensing

In a typical FME Server set up, the Engines & Licensing pages can only be accessed by the FME Server Administrator, but it is good for you as an Author to be aware of how the FME Server Engines and Queues function.

Licensing

To access the License page, expand Engines & Licensing then click on Licensing. This opens the Licensing page, where you can ensure FME Server is running correctly, is licensed, and has active engines:
In the first box, you can request a new license or refresh a license. Additionally, there is information about your machine key and serial number.

The second and third boxes display how many engines are available and when the license is set to expire. They also show which engine licensing style you have enabled, either Standard or Dynamic.

NEW

Dynamic engines are new for 2020.0, it is a different pricing model that runs on credits, which allows you to purchase additional FME Server processing power without having to install more engines locally. That way you only pay for what you need when you need it. For more information, see the Getting Started with
**WARNING**

The FME Server Administrator should have already set up the licensing before you log in for the first time. If you see either of the following two images when you log into FME Server, contact your FME Server Administrator. More details about how to licensing your FME Server can be found in the Server Administrator course manual:

---

**Engines**

To access the Engines page, click on the Engines tab while on the Licensing & Engines page or from the side menu bar. The Engines page shows the engines that are currently started, the queues that are assigned to them, their FME build number, operating system, and what job they are processing (if any). The platform (operating system) is important because a distributed FME Server setup can have engines running on a variety of operating systems at the same time.
**Hosts**

Below the Engines section is the Host section which shows what engines are running on each host, and allows you to easily change the number of engines running, up to the maximum number of engines provided by the current license. If you have dynamic engines enabled, you will also be able to control the number of dynamic engines here:

<table>
<thead>
<tr>
<th>Host</th>
<th>Standard Engines</th>
<th>Dynamic Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD-LS-WINDOWS2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

If your FME Server is licensed and has engines running that are assigned to the correct host, then you are ready to run a published workspace.

**Job Queues**

The Job Queue section of the Engines page provides a way to reserve FME Engines to processing jobs from specific repositories. For example – you could have an FME Engine dedicated for processing quick tasks so that slower jobs will not cause a backlog. Similarly, you might reserve an FME Engine that sits on a more powerful machine for processing LiDAR data translations. Job Queues are also where you can set the Job Priority.
<table>
<thead>
<tr>
<th>Queue</th>
<th>Description</th>
<th>Repositories</th>
<th>Engines</th>
<th>Priority</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default - System</td>
<td>Default</td>
<td>Samples, Utilities, Dashboards, Tools</td>
<td>FMETRAINING_Engine2, FMETRAINING_Engine1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Default Queue</td>
<td>Default</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Jobs</td>
<td>For training</td>
<td>Training</td>
<td>FMETRAINING_Engine1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Running a Workspace

Let's start our tour of the FME Server web interface at the Run Workspace page. You can access it from the corresponding menu option:

This opens the Run Workspace page, where you can choose a repository, workspace, and service to run a translation:
When you select a workspace that contains Published Parameters, those parameters will be available to be set before running the translation.

**Advanced Parameters**

The Run Workspace page in FME Server also has an additional section under Advanced. Clicking the arrow to the right of Advanced will expand the advanced parameters menu. These are FME Server specific parameters that fine-tune how the workspace will be run.
Job Queue

Job Queues provide a way to send FME Server jobs to specific Engines or to control the priority of job requests. This parameter allows you to select a predefined Queue to which this job should be sent when it is run. Job Queues can be created by an FME Server Administrator from the Engines & Licensing page.

Queued Job Expiry Time

This sets the maximum amount of time a job can wait in the Queued state before being run. If a job waits longer than this amount of time in the Queue, it will not be run. This option is useful for time-sensitive jobs that you do not want to run after the specified time is exceeded.

Running Job Expiry Time

This sets the maximum amount of time a job can stay in the running state. When this time is exceeded, the job will be canceled automatically.

Run Until Canceled

If checked, the job will run continuously even after a server shutdown or crash, until it is explicitly canceled.

Other Ways to Run this Workspace

Additionally, in the Advanced section, you can create an FME Server App or create a webhook to run your workspace, which will be covered in more detail at the end of this chapter.

Running a Job

When you are satisfied with the parameters, click the green Run button at the bottom of the page to run the translation. A progress dialog will appear indicating whether your workspace is Running or Queued. From this dialog you can cancel the job or view the details:
Once the workspace (job) has finished running, a completion message will appear, letting you know whether the translation was successful or if it failed. Depending on the Service you ran the workspace with, you will also see other details such as a download link:

Note that there are other ways to find and run a workspace. Recently published workspaces and workspaces "starred" as a favorite can be easily found on the interface landing page.

Additionally, it's possible to browse for a workspace to run by going through the Manage Workspaces page, which is more like a file browser than a simple selection tool.
Jobs

Information about all jobs that are running or have run on FME Server will be stored so you can easily view activity on the Server. To access this, choose the menu option for Jobs and select Completed, Queued, or Running jobs to view:

![Jobs Menu]

This opens the Jobs page, where you can see the status of jobs, whether Completed, Queued, or Running:

![Jobs Page]

This allows you to check that the translation you just ran finished successfully.

The Filters menu at the top allows you to filter by User, Status, Engine, Repository, Workspace, Source Type, and Source Name. This is especially useful when the job history table runs to thousands of jobs. Filtering helps to find information about a specific job or, for example, to quickly find all jobs that resulted in a failure message.

**Completed Jobs**
Clicking on a completed job opens up a page showing information about that job; such as Job ID, Job Priority, Time Started, Features Written and which user ran the job. You can also view, filter, and search the job log.

A series of buttons allow you to download the FME log from the translation, resubmit the job with a single click, view the workspace, and download the workspace:

This allows you to confirm that the workspace functioned correctly, with the same level of detail as you could find within FME Desktop.

**queued and running jobs**

Queued and Running jobs can be listed so that you can see what jobs the Server is currently handling. You can filter the jobs by a particular username.

One particularly useful feature is that these pages can be used to cancel jobs if they are no longer required.

**tip**

If your jobs are being queued regularly, talk to your FME Server Administrator about different options like Job Queues, dedicated engines or scheduling larger tasks to run outside of regular office hours.
Jobs
For the exercises in this chapter, you are a technical analyst in the GIS department of your local city. You have plenty of experience using FME Desktop, and your department is now investigating FME Server to evaluate its capabilities.

There are many departments within the city, and one of your tasks is to take the data from each department and merge it together into a single corporate database.

Because each department produces its datasets in a different format and style, you use FME for this task and carry it out on a weekly basis.

After creating a workspace to carry out this translation, and publishing it to FME Server, you now wish to log in to Server to run that workspace.

**NOTE**

You will need to complete Exercise 1.1 before starting this exercise. We will be running the published Ch1-Ex1-Complete.fmw workspace (or whatever your named it).

---

1) Connect to Server

To log in to the FME Server web interface, either select the Web Interface option from the Start menu or - in your web browser - enter the address of your FME Server.

If you are following this training on one of our training machines, the URL for your FME Server will be: [http://localhost/](http://localhost/)

**TIP**

When FME Server is installed on either physical or virtual hardware, the address is

http://<servername>/fmeserver

If you are using FME Cloud, then the address is: https://<servername>.fmecloud.com/fmeserver

This will open the web interface login screen for the FME Server being used.

2) Log In to Server

In the User Login dialog, enter a username and password for your FME Server account. The username/password combination for
a training installation is admin/FMElearnings:

Click the Login button.

3) Examine the User Interface
Welcome to the FME Server Web Interface. Take a moment to familiarize yourself with this interface. In the top-right corner, you can access the Help menu and your user settings:

The side menu is where all of the FME Server functions can be accessed. If you need more space, this menu can be collapsed. Note that the side menu will look different depending on which account you are signed into to. In the below image the user is logged in as admin:
Finally, while still on the FME Server Home page, you can access Recent Jobs, Recent Projects, Last Published Workspaces, Favorite Workspaces, and Help pages. There are also options in the top right to customize the layout and content of this page. If you clicked away from this page, just click the FME logo in the top left corner to get back to the Home page.

Under Last Published Workspaces, you should be able to find the workspace you published in Exercise 1. Click the star icon next to this workspace to set it as a favorite:

After clicking the star, the workspace appears in the Favorite Workspaces panel. This allows for quick access to this workspace from anywhere in the FME Server Web Interface:
We'll run the workspace shortly, but perhaps first we should make sure FME Server is running correctly (the fact that we could log in is a good sign) and that we are licensed and have engines running.

4) Confirm Licensing
Expand Engines & Licensing in the ADMIN section of the side menu, then click on Licensing. This will open up the licensing page. You should see a message informing you that FME Server is licensed and a list of the engines available:

![Licensing & Engines](image)

**TIP**
If your machine is unlicensed, or is missing engines, then check with your instructor for troubleshooting tips.

5) Run Workspace
Click the FME Server Logo in the very top-left of the interface, to return to the Web Interface Homepage.
Click on the published workspace in the Favorite Workspaces panel to open the web page for this workspace. If you don't see the workspace under the Favorite Workspaces panel, you didn't click the star in step 3.

The workspace page shows a few options, the first of which are for the repository, workspace, and service. These should already be filled in with values since we opened the workspace directly. If we opened the workspace through the Run Workspace page, we would have to fill these values in:

![Workspace Options]

Because this workspace has a few published parameters, they are also listed; but we can ignore these for now (we'll deal with source datasets later in the course).

Click the Run button to run the workspace. The workspace will run to completion, and a message to that effect will appear:

![Completed Message]

6) Examine Jobs Page
Click Jobs on the side menu to expand it, then click Completed to view a list of the completed jobs. A list of previously run jobs will open, including the one we just ran:
Notice some interesting parts of the interface:

1. There are tabs to show Completed Jobs (the default), Queued Jobs, and Running Jobs.
2. There is an option to turn on Filters for the view to allow you to filter the list of jobs to make it easier to find specific ones on a busy server.
3. An icon is used to indicate if jobs succeeded or failed. The green check indicates a successful job and a red x marks a failed job.
4. The jobs are displayed in the chronological order in which they finished (whether successful or not).

Click on your job to inspect the results in more detail. You'll see a summary at the top showing the number of features written as well as the time it started and finished. There are more timing details under STATUS that include the time the job was submitted, queued, etc. Information about the specific request made to FME Server can be found under REQUEST DATA. And full results of the translation are under RESULT DATA.

You may also inspect the FME translation log file on this page.

---

**FME Lizard says...**

*Remember, this workspace did not write any data, only sent it to a Null writer. So, for now, search for the summary in the log file to view any results.*

---

**Advanced Exercise**

*If you want to see a job in a different state, then we'll have to slow this workspace down some.*

*Open the workspace in FME Workbench and add a Decelerator transformer (say, before the Reprojector). Set it to delay the workspace by five (5) seconds per feature. Publish the workspace back to FME Server and re-run it.*

*Now the workspace will take 30+ seconds to run, and you should be able to find it under the Running state. Also, if you run it three or four times in quick succession, then you will have more jobs than engines and be able to find some jobs in the Queued state.*

---

**CONGRATULATIONS**
By completing this exercise you have learned how to:

- Log in to FME Server and check that it is running and licensed
- Locate a workspace using the Last Published list
- Run a workspace and inspect the job history to confirm it ran correctly
Workspace Versioning

Version Control makes it much easier to keep track of updates made to your workspaces as you make changes and publish them to FME Server.

You can commit new versions of workspaces while publishing them to FME Server or after you’ve published and tested your workspace through the web interface. Then access the full version history for any Repository or Workspace directly through the Repository page.

Configuring Version Control

Version Control will be turned off by default on a new FME Server installation, but can easily be switched on and configured by your FME Server Administrator. The options for this can be found under the System Configuration > Version Control menu.

TIP

Setting up Version Control from an administrator’s perspective is covered in detail in the FME Server Administrator training course. So be sure to check that out if you’re interested in learning more about how to configure it.

Authoring and Version Control

Once an FME Server Administrator has switched on Version Control, versioning tools will be available to any workspace authors.

Not every change made to an FME Server workspace will automatically be added to the version history. The workspace author can choose to ‘commit’ a version that they decide is significant or worth checkpointing. There are two ways to commit a new version of a workspace to the Version History.
Commit while Publishing

With Version Control enabled, a Commit button will appear on the Publish Workspace dialog of the Publish to FME Server Wizard inside FME Desktop. It opens a Version Options window that has options to add the current workspace to Version History as well as add a commit message with a summary of any changes made.

Commit after Publishing

The Commit button is also available within the Repositories page in the FME Server web interface. This is useful if you want to publish and test your workspace on FME Server before committing the new version. Just open the Repository, select the workspace, and click Commit.

This opens a page to optionally include a commit comment containing a summary of what has changed in the new version.
View Version History

To see the full version history of committed changes for any repository, go to the Repositories page and click the History button at the top.

This will open a dialog that lists all the commits in chronological order with the most recent at the top. It will also show the workspace that it relates to, the commit message, which user committed the change, and when it was added. The history will only show any changes after version control was enabled.
TIP

*By default, you will see all the versions for all workspaces within the repository. If you select a workspace first, then the history will only show the commits for the selected workspace.*

---

**Restore a Previous Version**

To restore a preview version of a workspace, open the Version History for that workspace and click the download button next to the version you want.
This will download a copy of that workspace to your computer. You can then open it using Workbench and publish that version back to FME Server.
Exercise 1.3: Daily Database Updates: Adding to Version History

| Data                  | Firehalls (GML)  
<table>
<thead>
<tr>
<th></th>
<th>Neighborhoods (KML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Goal</td>
<td>Commit a copy of the workspace built in the previous exercises to Version History on FME Server.</td>
</tr>
<tr>
<td>Demonstrates</td>
<td>Interacting with Version History tools</td>
</tr>
<tr>
<td>Start Workspace</td>
<td>None</td>
</tr>
<tr>
<td>End Workspace</td>
<td>None</td>
</tr>
</tbody>
</table>

You have already (in Exercises 1 and 2) created a workspace to carry out this translation, published it to FME Server, and run it to confirm it works.

You would now like to 'checkpoint' this working version of the workspace to ensure that you can always come back to this state if any edits are made to it in the future.

**WARNING**

*In the workplace you will have to have your FME Server Administrator enable Version Control before you can start using it.*

1) **Connect to Server**
Browse to the login page of the FME Server interface, and log in using the administrator account (admin/FMElearnings).

2) **Turn on Version Control**
Version Control will be turned off by default, so you'll need to switch it on before you can use it. From the Admin part of the interface menu, expand System Configuration on the side menu then click Version Control. On the Version Control page, toggle the switch to turn on Version Control, the switch will turn green when the feature is enabled.
Don't worry about the other settings on that page, just leave them with the default values.

3) Add Workspace to Version History
As the Ch1-Ex1-Complete.fmw workspace has already been published to FME Server, you can add it to the version history from the web interface. Go to the Manage Workspaces page and open the Training repository. Select the checkbox beside the workspace and click Commit.

Add a meaningful commit comment for this version of the workspace and click Commit to add it to the history.

4) Check Version History
Confirm that this new version has been added to your Version History. Go to the Manage Workspaces page and open the Training Repository. Select the checkbox beside the workspace and click the History button to see the information about the workspace version you just committed.
CONGRATULATIONS

By completing this exercise you have learned how to:
- Commit a new version of a workspace to Version History
- View the version history for a Repository
Scheduling

Scheduled translations are the best way to start a workspace at a particular time or date.

What is Scheduling?

Scheduling is the ability to configure FME Server to run a workspace in a repository at a specific time in the future. The schedule can cause the workspace to run once or on a repeating basis.

Managing Scheduled Tasks

Scheduled tasks are set up in the web interface. They can be accessed through a button on the main menu:

The interface supports all the capabilities you would expect, including the ability to create, remove, copy, enable and disable tasks:

TIP

In FME Server, there are two ways to schedule jobs. Scheduling from this Schedules page is one way to do this, but you can also create a scheduled process within Automations.

Creating a schedule through this page lets you set up a single workspace to run at a specified interval. This is ideal for simple scenarios.

Creating a schedule inside an Automation lets you trigger multiple workspaces to run at once and
provides many more tools for controlling exactly what will happen when that schedule kicks off. We’ll look at this in more detail later in this course.

NEW

The Show Automations button is new for 2020.1, it allows the user to see which automations are running on a schedule.

Creating a Scheduled Task

There are a number of parameters that can be set when creating a scheduled task.

The first group of parameters includes options for naming and describing the schedule as well as setting up the actual schedule. Here the workspace will be set to run once a day starting on the 22th of July at 2:00 pm (14:00).
NEW

If you are creating a quick schedule that runs daily, weekly, monthly, or yearly, there is now the Basic Schedule Type that allows you to quickly set this up.

Notice that each schedule can be assigned to a particular category.

The parameter labeled Skip if Job in Progress provides the ability to prevent the next scheduled run of a job from being submitted if the previous run is still incomplete.

The next few parameters concentrate on the workspace to be run.

Once a workspace is selected, there will be a short pause while FME retrieves information about the workspace. It will then expose any published parameters that exist in the workspace:
Finally, there are advanced options to control job queue, and job expiry (for jobs that are time-sensitive and would be no longer useful if held back past a particular time by higher priority tasks).

Once the parameters are set for a scheduled task, it is added to the main Schedules interface, where it can be easily enabled or disabled.
Exercise 1.4: Daily Database Updates: Workspace Scheduling

You have already (in Exercises 1, 2, and 3) created a workspace to carry out this translation, published it to FME Server, run it to confirm it works, and committed it to version history.

As a daily task, you plan to run the translation every day after work. However... what happens if you are not there or leave early? How can you make sure it runs every day at the same time then?

This is where Scheduling comes in handy. You can set up the workspace to run on an automatic schedule so you don't have to remember to run it daily.

1) Connect to Server
Browse to the login page of the FME Server interface, and log in using the administrator account (admin/FMElearnings).

2) Create Schedule
Now let's create a new schedule. Expand Schedules on the side menu and select Build Schedule.

Set a name of Test Schedule, then create a new Category named Training, by clicking on the plus sign '+':

Change the Schedule Type to Repeat on Interval. Set the Repeat Every to 1 Days at 9am, then enable Start Immediately. Then leave the end time blank, and ensure that Does Not Expire is enabled.
Be aware that the times are given in 24-hour format, so 1:30 means AM and 13:30 means PM. It is also important to note that this time is the local time of the machine on which you are running the web browser that is connected to FME Server. Keep this in mind if your FME Server is on a machine in a different time zone from the machine where you are accessing FME Server.

Under Workspace Settings, select the Training repository and within that the workspace previously uploaded (Ch-Ex1-Complete.fmw):

There are no user parameters we need to change for this workspace, so any can be ignored.

Now click OK to add the new schedule.

3) Trigger Schedule  
Our schedule is set up to run every day at 9am, depending on what time you are taking this course, you might have to wait until tomorrow to see the schedule in action. Thankfully, we can trigger the schedule to run whenever we want. This is great for testing purposes or whenever you might need the workspace to run.

On the Manage Schedules page, select the Test Schedule schedule we just made and then click the Actions drop-down and select Trigger.
4) Examine Jobs Page

Open the Jobs page. A list of previously run jobs will open. You will see the workspace we triggered with the schedule. You can tell it was the workspace associated with the schedule as schedule will appear under Source Type.

Notice that the username is set to admin; since that is the user who created the schedule, that is the username under which the job will be run.

CONGRATULATIONS

By completing this exercise you have learned how to:
- Schedule a translation in FME Server
- Trigger a schedule
- Check the job history to ensure the scheduled translation took place
Exercise 1.4: Daily Database Updates: Scheduling
Sharing within FME Server

FME Server security is based on whether you own a component or have been given access to it. A component might be a set of functionality or an object like a repository.

When you create something, you have full permission for that component. Even if you don't have permission to manage security on your FME Server, you do have the ability to share a component that you own with another user.

Sharing a Repository

Choose the menu option for Repositories in the Server web interface and you are presented with a list of repositories on the system.

If you are the owner of a repository, then you have the ability to click the button to *Share with Others*:

This opens a pop-up dialog in which to select a user and choose the level of permission that you wish to give to them:
FME Security is also based on users and roles. Roles are analogous to a group of users. When sharing a component, the "user" field can be an individual user, or it can be applied to a particular role; for example, you can give the ability to run workspaces in a repository to anyone in the fmeuser role.

FME Lizard says...

This is a very important capability. As an author you might publish a workspace intended for use by multiple users within FME Server. However, the workspace is of little use if those users don’t have access to it.

The Share with Others tool allows you to open up access to your workspace, without you needing the advanced permissions required for full security control.

Besides repositories, other components of FME Server can also be shared with other users. Keep watch within the user interface, and throughout this manual, for other sharing opportunities.
In the last exercise, you created a schedule to automatically run the workspace you created in Exercise 1 once a day. But what happens if something goes wrong with this workspace or it needs to be updated and you're not available to fix it? It would be a good idea to allow other FME Server Authors within your department to be able to edit and run this workspace.

Let's ensure that other users that are part of the FME Server Author role have access to this repository to run and modify the workspace.

1) Enable Author Account

In FME 2020, the default accounts of Author, User, and Guest are now disabled on installation for enhanced security. We will need to enable the Author account before we continue.

Browse to the login page of the FME Server interface, and log in using the administrator account (admin/FMElearnings).

Under the Admin section on the side menu bar, expand User Management, then click on Users.

Select the Author user, then in the Actions drop-down select Enable. A green checkmark should appear for the Author user under status.
2) Log In to Author Account

Now open an Incognito or Private window in your browser and open another instance of FME Server. Log in using the credentials author/author

The first thing you'll notice is that the menu and functionality is more restricted for this account (notice the Admin section is now gone):
Also, if you try to run a workspace, you'll find that this account does not have access to the Training repository where the existing workspace resides:

3) Share Repository
Minimize the incognito/private browser window where you are logged in as Author, and return to the browser where you are logged in as Admin.

You have the full set of menu entries, expand Workspaces and click on Manage Workspaces on the side menu. Under the list of repositories, locate the Training repository. Click the Share icon to the right:

In the Sharing Options dialog, select fmeauthor as the role to share with, and allow them full access to the repository:

By selecting the fmeauthor role (rather than the single Author account), we allow anyone who is tagged as an Author to access the workspace; and by allowing them full access to the repository, we allow them to run, download, and make edits to our workspace.

4) Check Sharing
Switch back to the incognito/private window with the Author account.
This time, you should have access to the Training repository. Click Run Workspace (or refresh the page), select your workspace in the Training repository and run it. Check the Jobs page, and you'll see one entry for the workspace when it was run as the Author. There is only one entry because the Author does not have the privileges required to view any other users’ jobs:

Switch back to the Admin browser. Now, in the Jobs > Completed page, you should be able to see both the administrator’s jobs and the Author's jobs:

That's because the administrator account has the permission to view all jobs.

CONGRATULATIONS

By completing this exercise you have learned how to:
- Share a repository in FME Server and tested to ensure it is available to the right users
Sharing Workspaces

FME Server Apps

Sharing a repository is an excellent option for sharing workspaces with other users who also have access to FME Server, but what if you want to allow anyone to be able to submit a job without needing to have an account on FME Server?

This can be accomplished using FME Server Apps. You can create an FME Server App by clicking on Server Apps from the FME Server Menu:

To manage FME Server Apps, click on Manage Workspace Apps. From here, you can manage all your existing apps and create new ones.

When creating a new Server App, you will be able to select which workspace you would like your app to run and also set an expiration date for the app. This is useful if you would like to temporarily share the app, then disable it after a set period of time.
Next, you can select which Published Parameters you would like to be displayed for your end users to set when they use your Server App:

And finally, you can customize the appearance of your Server App by changing the background color, adding logos, icons and banners.
An FME Server App URL will then be generated. Anyone with that URL will be able to run the app that you created without having to log into FME Server first.

The FME Server App link will open up a simplified Run Workspace page:
FME Lizard says...

You can also create an FME Server App from the Run Workspace page for any workspace by clicking on Workspace Actions > Create Server App or from under the Advanced parameters section. This can be a handy shortcut as it will open the Create FME Server App page directly and automatically select the Workspace for you.

NEW

New for FME Server 2020.1 are Gallery Apps. Gallery Apps act as landing pages for several Workspace Apps and URLs. They can be fully customized to suite your organizations needs all without any coding! For more information see Getting Started with Gallery Apps.

Webhook URL

While FME Server Apps allow other people to run a workspace on FME Server, a Webhook allows for an application to programmatically run a workspace. A Webhook URL will include all the workspace parameters directly in the URL itself. So, when the URL is triggered, it will immediately run the workspace without the need to prompt for those parameter values.

Webhooks are useful for building your own web applications that access FME Server services because you can copy the HTTP request and embed it on your own website or a 3rd party application. You could also embed the URL into an email, or paste the URL directly into a web browser.

You can create a Webhook URL from the Run Workspace page for any workspace. Once you have selected your workspace to run, click on Workspace Actions > Create Webhook or go to the Advanced parameters and select Create a Webhook from there. You will then be able to configure an your Webhook. You can set an expiry time to control how long the URL will be active for and also decide what values should be filled in for any Published Parameters associated with the workspace.
When finished, your Webhook URL will be generated and you will be able to download a text file containing the Webhook information and see some examples of how to use the Webhook within 3rd party or custom applications.
**Share Training/Ch1-Ex1-Complete**

**Webhook URL**

This webhook will run with the parameters configured on the Run Workspace page. Change Parameters

To use the webhook, a token can be specified in a header or query string parameter. For security reasons, providing the token in a header is the preferred method, because a request URL may be logged by proxies or web servers.

This is the only time you will be able to access the token and instructions to use this webhook.
You can save the information now.

[Download Webhook]

**Authorization with Header**

Use the following URL and specify the Authorization header in your request header. Note that this method will not work with a GET request.

http://FMETRAINING/fmejobsubmittor/Training/Ch1-Ex1-Complete.fmw?SourceDataset_GML=%24(FME_MF_DIR)FireHalls.gml&SourceDataset_OGCXML=%24(FME_MF_DIR)VancouverNeighborhoods.kml&opt_showresult=false&opt_servicemode=async

<table>
<thead>
<tr>
<th>Header Key</th>
<th>Header Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>fnetoken token=018e95f74dcbf8547924db9031ac7725f2f42</td>
</tr>
</tbody>
</table>

**Authorization with Query String**

http://FMETRAINING/fmejobsubmittor/Training/Ch1-Ex1-Complete.fmw?SourceDataset_GML=%24(FME_MF_DIR)FireHalls.gml&SourceDataset_OGCXML=%24(FME_MF_DIR)VancouverNeighborhoods.kml&opt_showresult=false&opt_servicemode=async&token=018e95f74dcbf8547924db9031ac7725f2f42
### Exercise 1.6: Daily Database Updates: Workspace Sharing

| Data          | Firehalls (GML)  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neighborhoods (KML)</td>
</tr>
<tr>
<td>Overall Goal</td>
<td>Create a workspace to read and process departmental data and publish it to FME Server</td>
</tr>
<tr>
<td>Demonstrates</td>
<td>Share a workspace so that anyone can run it</td>
</tr>
<tr>
<td>Start Workspace</td>
<td>None</td>
</tr>
<tr>
<td>End Workspace</td>
<td>None</td>
</tr>
</tbody>
</table>

You have already (in Exercises 1, 2, and 3) created a workspace to carry out this translation, published it to FME Server, run it to confirm it works, and committed it to version history.

In the last exercise, you shared the Training repository you created in Exercise 1 with other authors on FME Server. Now you have gotten requests from other users who do not have FME Server accounts to be able to run the translation on demand.

Let's create an FME Server App so that anyone with the URL can run this workspace to update the database whenever they need to.

1) **Connect to Server**

Browse to the login page of the FME Server interface, and log in using the administrator account (admin/FMElearnings).

2) **Open Server Apps**

Expand Server Apps from the side menu, then click Build Workspace App, to create a new Workspace App.

3) **Configure Server App**

Set a Name, Title, and Description for your new Server App. The Name is what is displayed in the list of Workspace Apps and the Title and Description is what will be displayed on the actual App. Select the Training Repository and Ch1-Ex1-Complete.fmw Workspace.

You can leave the expiration time at its default value, which will allow the API Token to expire after 10 years. You could set this to a shorter time if you only want to grant access for a smaller time window.

Keep the User Can Upload option turned on, this will allow your users to upload their own datasets to send as input to the workspace.
4) Customize Server App

Below the workspace selection area, you will see sections for Additional Permissions and Parameters.

- Additional Permissions will let you decide if you want the users who will access your Server App to be able to access items within the FME Server Resources.
- Parameters will allow you to configure which published parameters should be displayed for your users to set when using the Server App.

Leave those with their default values and move on to Customize Appearance. Expand Customize Appearance, then change Header Background Color to green. Then for Footer Logo browse to C:\FMEData2020\Resources\ServerAuthoring\ServerApps and upload CanoeZipster.png. You can also edit any of the other customization parameters if you wish.
Click OK to create the Server App.

4) Test the FME Server App
Now that your App has been created, you'll see that a URL was generated for it.
Click on the URL to open it. You will see that it opens a webpage very similar to the Run Workspace page in FME Server, but it has no options other than to run this one workspace and it does not require a user to enter a username and password to access it. The styling for the page will also match what you selected within the customization options.
Exercise 1.6: Daily Database Updates: Sharing a Workspace

Advanced Exercise

We've been requested to create a webhook to share our workspace with customers in an email. From the Run Workspace page, ensure that the Workspace is set to Ch1-Ex1-Completed.fmw, then from the Workspace Actions drop-down select Create Webhook.
On the Create Webhook page, we can just accept the defaults for the parameters. Optionally, you can add additional permissions for the user, such as access to the data folder. Click OK to create the Webhook. Before closing the Webhook URL page, ensure that you download your Webhook token and parameters as this will be the only time you can do so. If you forget to download it, or lose the file, you will have to recreate the Webhook.

CONGRATULATIONS

By completing this exercise you have learned how to:
- Create an FME Server App to share a workspace with users who do not have FME Server accounts
FME Server Projects

A Project in FME is a way to group FME Server items in a way that mimics a real-world project.

Grouping items together allows them to be handled together in a single action. For example, a workspace author can back up their work with a single action by grouping together the relevant parts as a project.

Project Components

A project can incorporate most FME Server items, including common ones such as:

- Workspaces
- Custom Transformers
- Repositories
- Schedules
- Automations
- Resources
- Web, Database, and Resource Connections
- Users and Roles

Project Uses and Advantages

Projects have a number of primary uses:

- Backups
- Migrations
- Sharing

Backups

The Backup tools on FME Server allow the entire FME Server to be backed up by an administrator.
However, a Project allows a workspace author to create backups of content they created (without being an administrator) and allows that author to pick which components are included in the backup (it does not need to be the entire server).

**Migrations**

In a similar way to backups, an administrator can transfer an entire FME Server onto a new server setup, but sometimes a workspace author needs to be able to transfer just their work to a different FME Server instance.

Projects allow an author to migrate their work between machines without requiring the assistance of an administrator.

**Sharing**

The system administrator has access to security functions on FME Server to control who has permission to use various components and resources. Authors also have the ability to share repositories that they have created with other users.

However, a project usually consists of more than just workspaces and an FME Project allows an author to share multiple components instead of just a single repository.
Working with Projects

All Project related tasks can be completed from within the Projects page of the FME Server Web Interface.

Creating a Project

Creating a new project is done by expanding the Projects section on the side menu and clicking Build Project:

On the New Projects page, enter in a Name, Description, and Readme information:

Below the Project parameters is a dialog that shows all content currently added to the project (if any) with a button to add new items on the right-hand side:
Clicking on the ‘+’ button opens the Add Project Items dialog which allows components to be added to the project:

Once you have added some items to the project, they will be listed under the Contents dialog:
When you click OK to create the project, if any of the components you selected have dependencies, you are able to fine-tune what exactly gets saved in the project:

Add Dependencies

Some included items contain additional dependencies. Any checked items listed below will be included with this project.
Please review and uncheck any dependent items that you do not wish to include with this project.

- Test Schedule (Schedules)
  - Training/Ch1-Ex1-Complete.fmw (Workspace)

- Exercise6App (Server Workspace Apps)
  - Workspace App - Exercise6App (Token)
  - author (User)
  - Training/Ch1-Ex1-Complete.fmw (Workspace)

Once complete the project is added to the list on the Manage Projects page:
Importing a Project

You can import an existing project into your FME Server by simply clicking the Import button.

The Import page allows a .fsproject file to be uploaded or read from a folder in Resources. It also includes other parameters to control the import:
Exporting a Project

Exporting a project is carried out on the Projects page (accessed via the main menu) by selecting the Project to be exported and clicking the Export button:
This opens a dialog in which to configure and carry out the export:

The export writes to a file with a .fsproject extension. There is an option to either provide the export as a download or to write it to a Resources folder.

**Removing a Project**

Once a Project has been exported, you may wish to remove it from the FME Server instance on which it currently resides. That can be done by selecting the Project and clicking the Remove button.

The confirmation dialog that opens allows you to choose whether to also delete all of the components that make up the Project:
Warning

Are you sure you want to delete this item?

Also remove contents  

[Box with options for Cancel and OK]
Sharing a Project

A project is often carried out by a group of FME users and authors. Therefore it makes sense that a Project can be shared among multiple FME users.

Share Project

Sharing a project is carried out on the Projects page (accessed via the main menu) by simply clicking the Share with Others button for the Project to be shared:

You don’t even need to select the Project first.

The Share with Others tool works the same way as the tool for sharing a repository: it opens a pop-up dialog in which to select a user and choose the level of permission that you wish to give to them:

Remember, FME Security is based on users and roles. Roles are analogous to a group of users. When sharing a Project, the "user" field can be an individual user, or it can be applied to a particular role; for example, you can give the ability to view your project to anyone in the fmeuser role.

Sharing a Project gives the ability to access not just repositories and workspaces, but all of the components related to that Project.
Share on FME Hub

FME Server Projects can be shared on the FME Hub. Check out this list of projects that are already available for download to import into your own FME Server.

When creating a project, there is a set of FME Hub specific parameters that you can fill in when creating a Project to prepare it for publishing to the FME Hub:
Job orchestration is an FME Server Automations concept. It allows for two or more parallel workflows with different processing times to run simultaneously, then the next action will wait until everything prior is completed, then proceed as one unified process. Automations will be covered in detail in Chapter 4; for now, we are just going to look at how to import a project a colleague has shared with us.

1) Browse To Projects
Open the FME Server web interface and log in with an account that has administrator privileges.

Expand Projects on the side menu and then go to the Manage Projects page:

2) Import Project
Click on the Import button to open the Import page:
We can accept the default Import parameters. Click Upload File and then browse to the following location:

C:\FMEData2020\Projects\ServerAuthoring\Ch1-Ex7-JobOrchestrationProject.fsproject

Click Import and the project will quickly be imported:

3) Check Log
Click the View Details button to examine the Project Import log. A successful import will look something like this:
TIP

The history page under projects will show a full history of all projects that have been imported to the system.

4) Check Components

Now let’s check for some of the components that should have been imported.

Click Manage Projects on the menu again, and click on Automations:JobOrchestration-Complete. You should now see a list of the imported contents:
Use the menu options (Automations > Manage Automations and Workspaces > Manage Workspaces) to ensure that the imported components do exist:
5) **Clean Up Project**

One part of the project that is not needed is a user account.

So, return to the project contents, select the Temp User, and remove it.
This will remove the account from the project, but since the project has already been imported, the account will also exist on the machine. So also go to the User Management > Users page and remove the Temp user:

You will be prompted to which user account to transfer the ownership of items that were created with the Temp user account. Select admin and then click OK:
6) Export Project

Now the project has been updated, export it so that it can be imported in its proper form elsewhere.

To do so, browse to the Manage Projects page, select the project (using the checkbox on the left), and click the Export button.

In the dialog that opens, you can choose whether to save the project file to a download or a resources folder. Once complete the following message will appear:

CONGRATULATIONS

By completing this exercise you have learned how to:
- Import a Project
- Check the Project History and confirm a Project was successfully imported
- Edit a Project’s contents
- Remove a user
- Export a Project
Module Review

This module introduced you to FME Server, and the Web Interface used to access most of its functionality.

What You Should Have Learned from this Module

The following are key points to be learned from this module.

Theory

- FME Server is a powerful product for handling large volumes of data at the enterprise level. It has three core capabilities: self-serve, real-time, and automation.
- FME Server provides the same processing core as FME Desktop, but on the enterprise level. It does not include an authoring tool, for which FME Workbench is required.
- Someone who creates translations for use on FME Server is called an author.
- The three main components of FME Server are FME Engines, the Server Core, and Web Services.
- The FME Server interface includes tools for running a workspace, examining job history, managing resources and data files, and scheduling jobs.
- Databases can have a container of connection information created and stored on FME Server.
- Workspaces can be shared with users or applications that do not have access to FME Server as FME Server Apps or Webhook URLs.
- Projects are a collection of different FME Server components
- Projects are an efficient way to manage multiple components
- Projects allow administrative-type tasks to be carried out by authors
- Projects can be used to import/export/share FME Server components

FME Skills

- The ability to open the FME Server web interface
- The ability to browse repositories and run a workspace
- The ability to check job status
- The ability to schedule a job to run at a particular time
- The ability to share a repository with another user
- The ability to share a workspace with anyone
- The ability to create an FME Server project and add components to it
- The ability to share an FME Server project with other users
- The ability to import, export, and remove an FME Server project

Further Reading

For further reading why not check out...

- This blog article on FME Server Projects

Troubleshooting Tips

Common issues when working with Projects:
- FME Server Troubleshooting: Projects
Questions

Which of these is **not** one of the three core capabilities of FME Server?

**Automation**

**Real-time**

**NoSQL Database**

FME Server is many things - but it is not a NoSQL database!

I have a Server with two engines. Four users submit jobs at the same time. What happens?

Two jobs are processed. Two jobs are returned to their authors. Two extra engines will fire up automatically to process all four jobs. The four jobs will be processed simultaneously, sharing the two engines. Two jobs are processed. The other two sit in a queue until an engine becomes free. Yes, the server core keeps a queue of jobs and submits them as engines become available. Extra engines will not fire up, even if you do have spare licenses, and jobs will never be ignored just because no engine is currently available.

If I wanted to find out about workspaces stored in a repository - for example I'm building a tool to catalogue my workspaces - what is the best way to do it?

Use the FME Server REST API. Scrape the contents of the Server repository page. Get a file listing from the repository folder. Connect to the FME Server database to query it directly. The REST API is a quick, simple, and official method to query the workspace repositories. Querying the database directly is permissible, however, under no circumstances should you write into or update directly the contents of the database.

Which of these are good reasons for running engines on multiple operating systems at the same time? Pick all that apply.

- A required format is only available on 32-bit or 64-bit, not both. A required format is only available on Windows, or Linux, not both. FME Desktop users author workspaces using a mix of Windows, Linux, and Mac platforms. You want to process heavy-scale jobs on a more powerful platform. Basically some formats are only available on certain platforms and so you may need to mix of operating systems to cover all your requirements. You may also want to redirect large-scale jobs to a more powerful platform. However, it doesn't matter what platform the workspace author used; their jobs will run on whatever system FME Server is based on.

A Project can be shared only in the following circumstances:

- You must own the Project. Only the Project owner can share it. You must be a user with permission to manage security. Only such a user can share a Project. You can own the Project OR be a user with permission to manage security (i.e. you can be one or the other). You cannot share a project once it has been imported to FME Server. If you own the Project then you can share it. But if you have permission to manage security, then you can share any Project.

If you choose to export a Project to a Resources folder (rather than download it) then what additional capability do you gain?

- The ability to trigger a notification topic on completion of the export. The ability to export the FME license for the server. The ability to remove all components of the project as they are exported. The ability to change ownership of the components to your own user account. When you choose to export to Resources, you'll see a Notification panel at the bottom of the export settings page. You can use that to select a topic to notify if the export succeeds or fails.

Checking the box to remove the contents of a Project removes all of the project components:

- True False A Project can contain users and roles. The remove tool won't remove the user carrying out the removal, nor will it remove any role for which that user is part of. Also it won't remove components you don't have permission to remove.
FME Server and Data Handling

This chapter introduces you to FME Server and handling data.
Source Data Management

Nearly every FME workspace starts by reading features from a source dataset.

In some cases that source dataset may be held in a database, or stored on a web service such as Google Drive; it may even be a database running on a web service! On other occasions, the data may not be web-based at all, but stored on a file system and shared for others to access.

In general, it's easy to author a workspace for use on FME Server because what works on FME Desktop can be published and run on Server with minimal alteration of the source data practices.

However, there are additional methods that can be used to take source data management on FME Server to the next level. These methods include:

- Using Database Connections for data stored in databases
- Using Web Connections for datasets stored on a web service
- Managing file datasets
  - Publishing file datasets to a repository
  - Uploading file datasets at run time
  - Storing file datasets on the 'Resources' file system
Using Database Connections

When the source data for a dataset is a database, FME is capable of storing connection parameters in a secure container. That container can be either published to FME Server or recreated on it.

What is a Database Connection?

Database connections are containers for a set of database connection parameters. These parameters include the database server, port number, username, password, and other parameters that vary according to the database type.

The two main advantages of database connections are:

- Connection parameters are no longer embedded in a workspace, meaning less of a security risk
  - For example, your parameters would not be exposed to anyone who downloaded the workspace
- Connection parameters can be reused among multiple workspaces
  - For example, two workspaces that use the same database can use the same connection

Database connections can be published with a workspace from FME Desktop, or they can be added directly within FME Server.

Creating Database Connections

Creating a database connection usually starts in FME Desktop. The connections can be created using Tools > FME Options > Database Connections in the FME Workbench menubar. The defined connection can then be used in a reader, writer, or transformer. Alternatively, when the author adds a Transformer, Reader or Writer to FME Workbench that requires a Database Connection but there is no existing connection for this format, they are prompted with the option to add a Database Connection from within this wizard.

As an example, this workspace has a connection for a PostGIS database, as seen in Navigator under the reader parameters and in the list of database connections:

When the workspace is published to FME Server a new dialog asks the author whether they would also like to publish the database connection:
The connection is then added to the connections container on FME Server.

FME Lizard says...

Note that you don't have to upload the connection with the workspace. If a connection for that database already exists on FME Server you can use that. FME Workbench should recognize whether a connection with the same name is already available, if it is, by default the box will not be checked.

If you don't upload the connection, and you don't already have one to use on FME Server, then you will need to use the Database Connections page to create one.

Managing Database Connections

FME Server has a page for managing database connections accessed through the main menu:

This page allows workspace authors - but usually administrators - to create new connections and copy, delete or edit existing connections:
Using Database Connections

When a workspace is run, if it has a database reader (for example) the end-user is prompted with a published parameter and can select the database connection to use:
The workspace then runs to completion as normal.

FME Lizard says...

Connections, like other objects on FME Server, have security permissions. Only the owner, someone with whom the connection is shared, or (by default) an administrator, can make use of it. It's not the case that any random user will be provided access to all database connections via the published parameter.

This also means that a workspace can be tested in FME Desktop using the author’s connection parameters, but then require the end-user’s connection once published to Server; all in a way that is both easy and secure.
Using Web Connections

Just as for databases, when the source data for a dataset is a web service, FME is capable of storing connection parameters in a secure container. That container can be either published to FME Server or recreated on it.

What is a Web Connection?

Web connections are containers for a set of web service connection parameters. These parameters include the service, username, password (or authenticated connection), and others that vary according to the service type.

The two main advantages of web connections are:

- Connection parameters are no longer embedded in a workspace, meaning less of a security risk
  - For example, your parameters would not be exposed to anyone who downloaded the workspace
- Connection parameters can be reused among multiple workspaces
  - For example, two workspaces that use the same web service can use the same connection

Web connections can be published with a workspace from FME Desktop, or they can be added directly within FME Server.

Creating Web Connections

Creating a web connection often starts in FME Desktop. They can be created using Tools > FME Options > Web Connections in the FME Workbench menubar. The defined connection can then be used in a reader, writer, or transformer.

For example, this workspace reads a CSV dataset using a connection to a Google Drive web service, as seen in the Navigator under both the reader parameters and the list of web connections:

When the workspace is published to FME Server a new dialog asks the author whether they would also like to publish the web connection:
The connection is then added to the connections container on FME Server.

**FME Lizard says...**

**WARNING:** While uploading a web service from Desktop is enough to run the connection, this service definition is provided by Safe Software for demonstration purposes and ease of use. The client id and client secret may become deactivated at any time. For production use, we recommend users obtain their own client id and client secret from the web service provider. Setting up the service definition for OAuth authentication is a task best left to system administrators. See the FME Server Administrator’s Guide for more information on how to carry this out.

**Using Web Connections**

When a workspace is run, if it has a transformer or reader that references a web service then it will run correctly, just as on an FME Desktop installation.

In the published parameters on FME Server, the web connection can be defined in the source dataset URL:

![Published Parameters](image)

Or, it can appear in a dropdown menu item if used as a published parameter in a workspace:

**FME Lizard says...**

As with database connections, this functionality allows a workspace to be tested in FME Desktop using the author’s connection parameters, but then switched to a general account once published to Server; all in a way that is both easy and secure.
Publishing File Datasets to a Repository

When the source data for a translation is stored as files (rather than a feed or database), it is possible to publish data to an FME Server repository along with the workspace. This data upload method is fast and simple, but limits future access to the data. A better practice is to publish your data to the Resources folder separate from the workspace, but we'll walk through those steps in a later exercise.

Publishing Source Data

In this workspace the source dataset is MapInfo TAB:

A MapInfo TAB dataset is made up of a series of files (.tab, .dat, .id, .map). When this workspace is published the wizard allows us to publish the data files alongside it by simply checking the box labeled *Upload data files*. 
FME automatically selects the files to upload based on what it thinks is necessary to run the translation. If there are other files you wish to upload, or files FME selected that you don't wish to upload, the Select Files button allows you to make changes:

This dialog also allows you to change where the files are published to, but for now we'll ignore that setting and go with the default of publishing to the repository.

Once the publishing wizard is complete, those files are uploaded to FME Server and tagged for use with this workspace.

### Using Published Source Data

When a workspace published with its data is run on FME Server, the uploaded data automatically is used as the source:

**Published Parameters**

- **Source MapInfo TAB (MITAB) File(s)**: $(FME_MF_DIR)Parks.tab

There are no other settings that need to be changed, and the workspace will run to completion using the published data as its source.
Uploading Datasets at Run Time

Although it's easy for an author to publish data to an FME Server repository along with the workspace, it isn't a method that an end-user has access to.

Therefore, for files (rather than a feed or database), functionality exists to allow the end-user to upload data at run-time.

**Uploading Source Data**

This workspace was created with a dynamic Reader and Writer. That means it is possible to process any source dataset (of the right format) and have it translated:

Of course, in this scenario publishing the data with the workspace does not make much sense. It is better if the user uploads data at runtime.

Provided the source dataset is a published parameter, this can be done easily in the Run Workspace page of the FME Server interface by dragging and dropping a file into the browser, or clicking the browse button:

This opens a dialog to select one or more files for the data type. To only upload the file temporarily, switch to the Temporary Uploads tab (1), then to upload the data click on the Upload Button (2). When uploading data that contains multiple files (like MapInfo Tab) ensure that you select all of the files. Once the data is uploaded to the Temporary Uploads, it will need to be added to the workspace, this is done by clicking on the green plus sign next to each file (3). Finally, before selecting OK, ensure all of the necessary files are selected (4). Note that all the files need to be uploaded, but only the main file needs to be selected for the workspace to run correctly, in this case, Parks.tab:
Now when the workspace runs, the selected user-uploaded data gets translated.

FME Lizard says...

In the above example, the source dataset is in a format (MapInfo TAB) that consists of several files. Although all files need to be uploaded, only the TAB file itself needs to be selected.

The method to use is to deselect files under the section Selected Items; here the .dat, .id, and .map files.

The files are still available, but FME won't treat each of them as a separate dataset, which is what would happen if they all remained selected.

Cautions and Limitations

There are some cautions and limitations to be concerned about when data is uploaded for translation by the end-user:
● Giving the user upload ability is risky because their dataset's schema has to match the workspace's schema definition; otherwise, the translation will fail with unexpected input. Alternatively - as above - a dynamic (and maybe generic) translation could be used to avoid such issues.

● Data uploaded by the user is only temporarily available. The System Cleanup page shows us that such files are (by default) deleted when they become more than 24 hours old. User uploads are not a long-term solution.

● Data uploaded by the user is, in theory, also accessible through the Resources page (more on that to come). However, in practice, it's in an obscure location where an end user would not be expected to find it. For that reason, temporary data should be considered inaccessible by any other means and unavailable for use by any other workspace.
For the exercises in this chapter, you are a technical analyst in the GIS department of your local city.

You have already (Exercises 1.1, 1.2, and 1.3) created a workspace to carry out a translation, published it to FME Server, ran it to confirm it works, shared the repository, and set the workspace to run on a schedule.

Now you have a task to create a new workspace. One of the datasets it uses is the same as in the previous exercise, so we will try to have this second workspace use the data belonging to the first.

FME Lizard says...

*If you have lots of experience with FME Workbench - and if your instructor agrees - simply open the end workspace listed in the header above and skip to step 7*

1) Create Workspace

Open the starting workspace listed above.

You might notice that it's a copy of our previous workspace since the requirements for this workspace are so similar. If you do choose to carry on working in that workspace, be sure to save it under a different name - otherwise the data we will publish will not work for this exercise.

The workspace looks like this:

2) Remove Firehalls

For this workspace we need to process election data instead of FireHalls, so firstly delete the writer feature type for the FireHalls, and then the reader feature type.

When you delete the reader feature type, you will be asked if you wish to delete the entire reader. We could reuse it but, for the sake of simplicity, click yes.
The workspace now looks like this:

3) Add VotingPlaces
Now select Readers > Add Reader to start adding a reader to the workspace. When prompted, enter the following details for the VotingPlaces data:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>GML (Geography Markup Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\Elections\ElectionVoting.gml</td>
</tr>
</tbody>
</table>

Click OK to add the Reader to the workspace. When prompted only select the VotingPlaces feature type, not VotingDivisions:

After adding any source data to a workspace the first task should always be to inspect the data, so let's do that using Visual Preview. Select the Reader and in the ribbon floating above it click on the View Source Data Icon.

You can turn off the VotingDivisions layer under Display Control. All we are interested in for this exercise are the point features designated as VotingPlaces:
Exercise 2.1: Daily Database Updates: Publishing Data

4) Add VotingPlaces to Writer

To add VotingPlaces to the writer, right-click the newly placed reader feature type and choose Duplicate on 'NULL':

There will now be a reader and writer feature type for the VotingPlaces dataset:

Change the connections to pass the VotingPlaces data through the Clipper transformer just as the FireHalls used to be:
5) Set VotingPlaces Feature Type Name
Finally, as with the FireHalls, let's set the Feature Type Name for the VotingPlaces writer feature type.

Inspect its parameters and under Feature Type Name either enter:

```
VotingPlaces-@Value(NeighborhoodName)
```

...or click the dropdown and use the text editor dialog to enter that value. This will cause voting places in each different neighborhood to be written to a separate table/layer.

Save the workspace. As already mentioned, make sure it has a different name than the first project.

6) Publish to Server
Publish the workspace to FME Server.

If its saved, choose the previously created FME Server connection. Or, select Add Web Connection again from the dropdown menu to reconnect with the provided credentials:

- **FME Server URL**: http://localhost
- **Username**: admin
- **Password**: FMElearnings

Create a repository by clicking the New... button, enter the name "Training". Enter a name for the workspace if it doesn't already have one.

This time, instead of simply checking the box to upload all the data files, click the Select Files button:
This dialog lists the files we are about to publish to the repository with the workspace. Technically, the VancouverNeighborhoods dataset may already be published to the repository with the previous workspace, but it's not very good practice to try and re-use data this way (even though we could) so place a check mark against all files and click OK:

In the final dialog of the publishing wizard, once again choose the Job Submitter as the web service to register the workspace against.
7) Examine Files
If you have access to the FME Server computer itself, open a file browser and browse to the location that repository data is stored. Here it is C:\ProgramData\Safe Software\FME Server\repositories\Training:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics-Ex1-Complete</td>
<td>8/14/2018 9:51 AM</td>
<td>File folder</td>
</tr>
<tr>
<td>DataHandling-Ex1-Complete</td>
<td>3/14/2016 10:04 AM</td>
<td>File folder</td>
</tr>
</tbody>
</table>
```

You’ll see that each workspace is saved to a separate folder. If you inspect the contents of a folder, you’ll see the uploaded datasets within it.

This is how a workspace has access to files published with it. It can also, with some manual effort, access files stored with another workspace in the same repository.

8) Run Workspace
Log in to FME Server and then locate and run the workspace. In the Run dialog notice that the published parameters denoting the source data include an FME environment variable, FME_MF_DIR:

```
$(FME_MF_DIR)\VancouverNeighborhoods.kml

$(FME_MF_DIR)\ElectionVoting.gml
```

This variable tells FME to look in the same folder as the workspace for the source data files. As you can see, it isn't particularly user-friendly to handle data in this way, even though the workspace will run just fine.

9) Upload Temporary Data
Now let’s pretend that the layer of VotingPlaces data has changed in some way. You can simulate that by simply opening a file browser and making a copy of the GML file.

For example, rename C:\FMEData2020\Data\Elections\ElectionVoting.gml to NewElectionVoting.gml

**NB:** You don’t also have to copy ElectionVoting.xsd - it’s okay to use that schema file for the new GML dataset.*

We wish to run the workspace with the new data. To use the new dataset for the Source GML prompt, click the browse option:
In the dialog that opens, click the Temporary Uploads tab and then on the Upload File button:

Select both the files NewElectionVoting.gml and ElectionVoting.xsd and click Open to upload them. Now - back in the prior dialog - click the X button to deselect the XSD file:

The file needs to exist, but it doesn't need to be selected. Now click OK and then click the Run button.

The workspace will now run to completion using the uploaded dataset.
However - and this is the important part - this was only a temporary upload. The workspace can be re-run immediately, and the data will still appear in the temporary upload section, but it is not a permanent solution. The data is likely to be cleaned up automatically within 24 hours, this is the default unless an FME Server Administrator has modified the clean up task.

CONGRATULATIONS

By completing this exercise you have learned how to:
- Update a workspace with a new reader and a new writer feature type
- Publish a workspace to FME Server and include source data
- Locate source data on the FME Server filesystem
- Select a source dataset to upload temporarily at run-time
The Resources File System

The final and preferred method of managing source data in FME Server is to use the system of tools called Resources.

What are Resources?

"Resources" is an inbuilt file management system that allows data (and other files) to be published to an FME Server instance and used within all Server operations. Resources can be managed through the Resources page accessed from the main menu, under "Files and Connections".

The Resources page is where you can find all of the files that are uploaded to FME Server and also the files that are created by FME Server:
As you can see, the Resources filesystem is set up with many default folders in which files can be stored. Custom folders that you create can also be added to the filesystem, we’ll do this in the next exercise. For now, there are a number of different folders data can be stored in, for datasets the most logical folder to use is the Data folder:

Above is the data folder containing several files. Notice once a dataset has been checked, the Actions button is no longer greyed out and lists various actions that can be performed on the file or folder, such as duplicate, edit, upload, copy, delete, or move files (or folders).

**Other Upload Methods**

Besides the web interface, there are other ways of getting data into the Resources filesystem.

Firstly, the FME Server publishing wizard in FME Workbench allows this. If not otherwise specified at installation, the default method is to select the files and upload them to the same repository as the workspace. However, it is preferable to change the location to the resources filesystem:
Alternatively, FME Server resources actually exist on the operating system’s filesystem, meaning the data can be copied there directly. The default location (on a Windows operating system) is:

```
C:\ProgramData\Safe Software\FME Server\resources
```

**TIP**

*ProgramData* is a hidden folder, in order to see it on your Windows Machine, click the View tab on the ribbon in Windows Explorer, then in the Show/Hide category, enable Hidden Items:
Finally, clicking the New button in the main Resources page in FME Server allows a connection to be made directly to a Network Based Resource via a UNC path or an Amazon S3 filesystem to be used to store backfiles (either for upload or download).

Some important limitations include that while Network Based Resources can be used within workspaces, Amazon S3 filesystems cannot. Additionally, the resources must be shared with the FME Server Service Account to use them with FME Server Services.

**Using Uploaded Data**

Using Resources data in a translation is simply a case of selecting it from that folder where prompted. All prompts for data will allow selection of files from a Resources folder.

For example, a user has uploaded a MapInfo TAB Parks dataset to the Resources data folder. Provided the source dataset is a published parameter, when the workspace is run the user is able to select data from the Resources folders, like so:
In fact, it's even possible to set the output data folder to be a Resources folder too:

**Published Parameters**

<table>
<thead>
<tr>
<th>Source MapInfo TAB File(s)</th>
<th>$(FME_SHAREDRESOURCE_DATA)/Parks.tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Geography Markup Language (GML) File</td>
<td>$(FME_SHAREDRESOURCE_DATA)/Parks.gml</td>
</tr>
</tbody>
</table>

Don't be concerned that the value doesn't show the full file path but rather a value contained within $( ). When a folder is stored in Resources it is assigned an FME Server User Parameter value, which is essentially a shortcut to the FME Server System Share file path. We'll cover this more in the next section.

**Benefits for Data Management**

There are several benefits to using the Resources filesystem as a data storage tool:

- Data can be used by any workspace, without having to upload it every time
- Data can be stored locally (to the FME Server Engines), even when access to the operating system's filesystem is restricted by an FME Server Administrator
- A Resources folder can be mapped and shared among many users as a physical drive
- A Resources folder is a more permanent solution. Data is not removed by automated system cleaning tools
Authoring for the Resources System

Using the FME Server web interface, it's simple to select data from the resources folder at run-time. However, in some cases, the author will want to read data from a resources folder without the end-user having to select it.

To do this requires the use of an FME parameter to define the data as coming from the resources folders.

FME Parameters for Server

In FME Workbench, the Navigator window has a section called User Parameters. You might have noticed that one part of this is a list of FME Server-specific parameters. For a detailed description of all of the Server Parameters, please see the FME Server Parameters documentation:

The uses of these are many and varied; for example, FME_SECURITY_USER returns the name of the user running the workspace and could be used to either write to a custom log or perhaps filter data in different ways in the workspace based on the specific user. FME_TOPIC would return the name of the notification topic (if any) that invoked the workspace.

However, when authoring for resources data, the most useful parameter is FME_SHAREDRESOURCE_DATA
**WARNING**

A common factor to all these parameters is that they only have an effect when the workspace is run on FME Server. If the workspace is run on FME Desktop, it won't return a value. Therefore to test on Desktop a workspace containing such a parameter requires you to provide a dummy value.

### FME_SHAREDRESOURCE_DATA

The FME_SHAREDRESOURCE_DATA parameter returns the path of the shared resource data. When authoring a workspace to read data directly from the resource folder, you would normally create the workspace using a local copy of the data, and then update the source dataset field to include the FME Server parameter:

![Image of workspace configuration](image)

Updating the field can be done directly by typing into the dialog, but is easier to achieve by clicking the drop-down arrow and choosing to use the Text Editor.

The entry in the Navigator window now looks like this:

![Navigator view](image)

Although the parameter is colored red in Desktop, when the workspace is run on Server the parameter is replaced by the actual path and the data is read as expected.

**WARNING**

It's important to remember (for example notice in the screenshots above) that the Server parameter
FME_SHAREDRESOURCE_DATA includes the 'Data' folder in its path.

For example, I use FME_SHAREDRESOURCE_DATA\Zoning\Zones.tab not FME_SHAREDRESOURCE_DATA\Data\Zoning\Zones.tab

Shortcut to Resource Paths

Rather than setting the path in Workbench before uploading the data, you can also copy the path for an uploaded dataset and then paste that directly into Workbench.

The path for a shared resource is obtained by examining its properties in the FME Server Resources pages. To do this, locate the resource, click the checkbox beside the file, then click the Actions drop-down menu and select Properties:

File Properties

| Name:     | Parks.tab |
| Size:     | 0.36 KB   |
| Date:     | 2018-3-19 12:04:23 |
| Type:     | FILE     |
| System Path: | $(FME_SHAREDRESOURCE_DATA)/Parks.tab |

This path can then be copied and pasted into a workspace in order to reference that dataset directly without manually entering it. Of course, this does require that the data has already been uploaded to FME Server, and isn't going to be uploaded with the workspace when it is published!
Exercise 2.2: Daily Database Updates: Using Resources

<table>
<thead>
<tr>
<th>Exercise 2.2</th>
<th>Daily Database Updates: Using Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Neighborhoods (KML)</td>
</tr>
<tr>
<td></td>
<td>Election Voting (GML)</td>
</tr>
<tr>
<td><strong>Overall Goal</strong></td>
<td>Create a workspace to read and process departmental data and publish it to FME Server</td>
</tr>
<tr>
<td><strong>Demonstrates</strong></td>
<td>Uploading data to a resources folder and authoring a workspace to make use of it</td>
</tr>
<tr>
<td><strong>Start Workspace</strong></td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\DataHandling-Ex2-Begin.fmw</td>
</tr>
<tr>
<td><strong>Start Server Project</strong></td>
<td>N/A</td>
</tr>
<tr>
<td><strong>End Workspace</strong></td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\DataHandling-Ex2-Complete.fmw</td>
</tr>
<tr>
<td><strong>End Server Project</strong></td>
<td>C:\FMEData2020\Projects\ServerAuthoring\DataHandling-Ex2-Complete.fsproject</td>
</tr>
</tbody>
</table>

You have already (Exercise 2.1) created a workspace to carry out a translation, and published it to FME Server; both with data and using data uploaded temporarily.

However, such data management tools are not particularly suited to a long-term project, so the task here is to upgrade the workspaces to use datasets stored in a Resources folder. There we can store source data and write destination data.

1) Open FME Server Web Interface
Log in to the FME Server web interface using an administrator account (such as admin/FMElearnings). Click Resources on the menu bar to navigate to the resources management pages.

2) Create Folder
In most cases data should be stored under the Data folder, so click on Data in the Resources dialog to open that folder. To avoid mixing datasets, our data should go into its own subfolder. So click on the +New button and create a folder called Election:

Next click on the Election folder and within there create new subfolders called Input and Output:
3) Upload Source Datasets
Browse to the Input folder and click the Upload button and then select Files. Upload the source datasets for the current translation:

**Reader Datasets**
- C:\FMEData2020\Data\Elections\ElectionVoting.gml
- C:\FMEData2020\Data\Elections\ElectionVoting.xsd
- C:\FMEData2020\Data\Boundaries\VancouverNeighborhoods.kml

So we now have both source datasets and a folder to write the output data to.

4) Add Writer
Up until now all of our workspaces have had only a NULL (dummy) writer. Now we know about Resources we can add a proper writer and point the output dataset to the Resources Output folder.

So, open the starting workspace listed above in FME Workbench and then select Writers > Add Writer on the menu bar and set up a new writer with the following parameters:

<table>
<thead>
<tr>
<th>Writer Format</th>
<th>Esri Geodatabase (File Geodb Open API)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Writer Dataset</strong></td>
<td>C:\FMEData2020\Output\Training\DepartmentData.gdb</td>
</tr>
<tr>
<td><strong>Feature Class or Table Definition</strong></td>
<td>None (Advanced)</td>
</tr>
</tbody>
</table>

The reason we want to add no feature types is that we can move the existing ones from the NULL writer. So when you click OK the workspace will look no different, but there will be a new writer in the Navigator window:
5) Move Feature Types
Inspect the parameters dialog for each writer feature type in turn. For each type move it from the NULL writer to the FILE GDB writer, like so:

This will expose a number of extra parameters. The key one to set is Geometry. For the Neighborhoods they should be set to `geodb_polygon`:

For the VotingPlaces feature type, the Geometry parameter should be set to `geodb_point`.

Now the two feature types belong to the Geodatabase writer, and the NULL writer can be deleted from the Navigator window if you wish.

6) Set Geodatabase Parameter
One (very quick) last thing to change: locate the Geodatabase writer in the Navigator window and expand its list of parameters. Double-click the parameter labeled Overwrite Existing Geodatabase and set it to Yes. To do this check the checkbox Overwrite Existing Database.
This ensures we aren't continually adding data to the same dataset if we run the workspace more than once.

7) Run Workspace
Test run the workspace in FME Workbench. Inspect the output in Visual Preview. If it is not already selected click Toggle Display Control on the left hand side of the visual preview page. You should find the output is a Geodatabase containing seven tables (the Neighborhoods table and a separate table for each set of voting places).

8) Publish and Run Workspace
Connect and publish the workspace to FME Server. Be sure not to check the button to upload any data. Register the workspace against the Job Submitter service as usual.

Return to the FME Server web interface. Locate the workspace under the Run Workspace dialog. Notice how all the dataset parameters are blank. This is because the paths referenced the original file locations through a mapped drive. If the default file path value does not reference a file using a UNC path or Resource Connection FME Server will strip out that value since FME Server does not have access to those files:
However, because we already uploaded them to the Resources folders, we can use those files.

So, for each file, click the browse button, browse to the Input subfolder in Resources, and select/set the file location. Since the Geodatabase does not exist yet, for the Geodatabase output location you’ll need to type the file name manually:

$\{(FME\_SHAREDRESOURCE\_DATA)\}/Election/Output/DepartmentData.gdb

Specify destination Geodatabase (existing geodatabase folder, empty folder, or specify a folder that doesn't exist yet) for File Geodatabase

Now when the workspace is run, a completed Geodatabase file should appear in the folder Resources\Data\Election\Output:
9) Apply FME Server Parameter
Although the workspace ran correctly, and used the data in the resources folder, that's only because we selected that data at runtime. It is not a permanent feature of the workspace.

It would be much better if the workspace was programmed to look into the resources folders automatically.

So, return to the workspace in FME Workbench.

If we do set the workspace to read from the resources folders, we don't want to give users the chance to change that. So in the Navigator window locate the three parameters for source and destination datasets and delete them:

10) Set Source/Destination Parameters
Now, in turn, locate the source and destination dataset parameters for the two readers and one writer. Double-click each in turn and change them to:

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>KML Reader</td>
<td>$(FME_SHAREDRESOURCE_DATA)/Election/Input/VancouverNeighborhoods.kml</td>
</tr>
<tr>
<td>GML Reader</td>
<td>$(FME_SHAREDRESOURCE_DATA)/Election/Input/ElectionVoting.gml</td>
</tr>
<tr>
<td>Geodatabase Writer</td>
<td>$(FME_SHAREDRESOURCE_DATA)/Election/Output/DepartmentData.gdb</td>
</tr>
</tbody>
</table>
Save the workspace and publish it back to FME Server.

**TIP**

If you try running this workspace in FME Workbench now that you’ve updated the dataset parameters to use the FME Server Shared Resource parameter, you will get an error because Workbench won’t recognize that FME Server parameter.

If you still want to be able to test this workspace in Workbench, you can set a value for the FME_SHAREDRESOURCE_DATA parameter. Configure it to point to a local file path that matches the folder structure used on FME Server. The workspace will still run on FME Server as Server will automatically override any value for that parameter and replace it with the correct path.

11) Re-Run Workspace

Now run the workspace on FME Server. This time you will not be prompted with a parameter to select the source (or destination) datasets, but they will be used from the resources folders just the same.

**CONGRATULATIONS**

By completing this exercise you have learned how to:
- Create resources folders and upload data to them
- Add a writer to a workspace and move feature types from another writer
- Run a workspace and select data from resources folders
- Edit a workspace to permanently use the resources folders
- Delete parameters to prevent the end-user changing them
Sharing Custom Resources with FME Engine

As an FME Author you may create workspaces that require additional references to files not installed with FME as part of the standard library, in which case these files need to be uploaded to FME Server in a specific location for the FME Engine to have access to these. This process is separate from pushing up source datasets and may require involvement from your FME Server Administrator.

Sharing Custom Formats and Transformers

There are three ways to upload custom Formats and Transformers to FME Server: 1) Upload the .fmx file to Resources in the FME Server Web UI under the appropriate subdirectory in the Engine folder. This will make the format or transformer accessible to all workspaces on FME Server.

2) Publish the custom format/transformer .fmx file to FME Server by opening the .fmx in FME Workbench like you would a .fmw and using the File > Publish to FME Server option. Using this method it will be available to all Workspaces stored in the same repository that it is published to. 3) Publish when publishing a workspace by adding the .fmx as a file to Upload in the publishing wizard. Specifying either to upload to Repository (2) or to a shared resource folder (1) depending on who you would like to give access to this format.

Using Python with FME Server

When using Python with FME Server, there are two possible scenarios an FME author should be aware of that may prevent the workspace from running as expected:

1) If you run workspaces that reference python from FME Server, the FME Engines must know which Python interpreter to use. The Python interpreter that the FME Engines use is the one that most closely matches the Python Compatibility workspace parameter set in the Navigator.
2) When Using Additional Python Modules not shipped as a part of the standard library it is necessary to upload the Python modules to FME Server Resources under Engine > Plugins > Python.

Using R with FME Server

Similar to using R with FME Workbench, there are some additional steps needed to execute R scripts in a workspace running on FME Server:

1) Install R
2) Install the sqldf package for R
3) Upload the package libraries to FME Server under Resources > Engine > Plugins > R

Steps 1 and 2 must be performed on all machines that run FME Engines.
Module Review

This module introduced you to how FME Server handles data.

What You Should Have Learned from this Module

The following are key points to be learned from this module.

Theory

- Data (and other related files) can be uploaded and stored on FME Server in various ways; either published data, temporary uploads, or as “Resources.”
- Workspaces can be authored in FME Desktop specifically to use data stored in FME Server resource folders.

FME Skills

- The ability to publish data with a workspace, upload data for a workspace, or manage data in resource folders.
- The ability to create a workspace that references data in a resource folder.

Further Reading

For further reading why not check out...

- The Configuring the FME Engine Documentation that covers the solutions for working with Grid Shift Files, R and Python in more depth.
- The Sharing Custom Formats, Transformers, and Coordinate Systems on FME Server Documentation for more information on sharing custom resources.
**Questions**

If I create a database connection that has superuser permissions then it would bypass any permission checks the database would make for creating and dropping tables. So how do you think I might prevent a user misusing that capability?

Remove that user's permission to run that workspace on FME Server Remove that user's permissions to access the entire repository that workspace resides in Remove permission to access that particular database connection for that user's role Remove from their role permission to manage database connections If I want to deny a user access to a superuser database connection, then I (or a system admin) just removes permission to access that connection for that user's role. Well done.

Although simple, there is a major limitation to publishing data with a workspace. What do you think it is?

The data is only temporary and will be deleted once the workspace is run The data is hidden within FME Server's system files and limited in its use The data becomes available to anyone regardless of role and security settings The workspace cannot be run using any other data than that's published with it The limitation is that a dataset published in this way can only be referenced by its own workspace or workspaces run from the same repository. Even then there is no browse capability in the FME Server web interface; the source dataset parameter would need setting manually.

Incidentally, none of the other answers are true: the data won't be deleted, it isn't open to anyone (unless they have specific access to this repository), and the workspace can be run using other data if required.

I copy a workspace into a resources folder using the upload tool. What then?

I can run it by browsing the resources, selecting the workspace, and clicking run I can run it through the Manage > Workspaces menu tools I can run it by calling it with the FMEServerJobSubmitter transformer in FME Desktop I can't run it because it's not properly published to a repository Basically, if you don't publish the workspace properly, you aren't able to have Server run it.

Uploading an entire folder of files come with what restriction?

Folder upload only works on certain web browsers Folder upload requires the folders to be zipped into a single file Folder upload only works on Windows C: drive (not D:, E:, etc) Folder upload requires FME Desktop to be installed on the computer being uploaded from Folder upload works in Chrome, but not in Firefox, Internet Explorer, or any other web browser.

I can make my workspace read specific data from the resources folders - but how do I stop the end-user from being able to change that?

Remove their security permissions for the Job Submitter service Remove their security permissions for the Resources folders Make the source dataset parameter optional for that Reader Delete the published parameter for that source dataset from the workspace Yes, in the Navigator window look for a published parameter that relates to the source dataset, and remove it. The option to change the dataset will then not be presented to the user.
Self-Serve with FME Server

Creating self-serve data integration workflows takes the burden of user requests away from expert staff, enabling them to concentrate on more important work.

This chapter looks at how self-serve is implemented on FME Server, and how to allow the end user to set parameters, select format, choose a coordinate system, and define which layers to download.
Self-Serve with FME Server

Self-serve is the term for a system that is set up to allow the end-user to carry out their own data translations and transformations. In this way, routine data management tasks are offloaded from staff to the user, who is empowered to carry out processes at their own convenience.

Usually the system is set up in such a way that the end-user needs no prior FME experience or training to carry out their goals; for example, they can access the functionality through a web interface customized to their particular needs. In fact the user does not even need to know of FME, or that FME is the engine driving their applications!

Self-Serve Types

In general there are two types of self-serve systems.

Data Upload systems are where the user is able to upload their data to be processed on FME Server. A typical application would be a user uploading data to be validated. The data is run against a number of tests in an FME workspace and the results sent back to the user.

Data Download systems are where the user is able to serve themselves with data. A typical application would be an organization that frequently provides data to either staff or customers. With a Data Download system, the user can fetch their own data rather than having to be provided with it in a more manual way. Data can be downloaded as a set of files or streamed directly into an application.

Data Uploads

A Data Upload system is one where a user provides data to FME Server for it to process.

Data Upload can be used from any FME Server client, including:

- The FME Server web interface
- An FME workspace
- An FME Server App
- A custom web page or application

Data Upload is often used for submitting data to an organization; for example a property developer submits a planning application containing a DWG dataset to a municipal planning department.

It is also often used for publishing data to be processed on FME Server; for example FME Server can provide a data validation web service and an end-user would upload a set of data to be checked.

It’s worth noting that data upload also includes not just data, but other resources that may be required for a translation to run; for example custom transformers or text-file lookup tables may also be uploaded.

Data Downloads

A Data Download system is one where a user selects their own choice of data to download.

Data Download can also be used from any FME Server client, usually:

- The FME Server web interface
- An FME Server App
- A custom web page or application
Data Download operates in a different way to simply running a workspace.

When you run a workspace (with the Job Submitter) the data is written to the location specified by the workspace; for example a file, directory, or database.

Data Download instead writes the output to a zip file and presents the user with a link to that file. This makes it ideal for self-serve, because the data is delivered directly to the user.
Self-Serve and Services

Self-serve is implemented through a number of Services on FME Server. A Service is a particular method of communication between client and server. FME Server provides a wide range of services to carry out different forms of data self-serve.

What is a Service?

In the simplest of terms, a service is a piece of software that handles communications between a client and a server. In other words, it’s a tool that allows users to access complex functionality through a simplified interface.

In terms of FME Server, the client is often—but not always—a web browser that passes requests to FME Server using a service.

In short, a service allows the sending of specific types of requests to FME Server, and allows results to be provided to client applications in a specific way.

For example, instead of just running a workspace, you can have a web page ask for the results of the workspace as a package of data compressed in a zip file.

FME Lizard says...

Good morning class. I’m here to guide you through this chapter on Self Serve with FME.

Let’s start with the idea of services. Although the concept sounds complicated, a service is just a simpler way of communicating requests to FME Server than using the API. Also, FME Server includes a number of predefined services that cover a lot of the functionality you are likely to need.

Available Services

FME Server includes the following services:

<table>
<thead>
<tr>
<th>Data Download Service</th>
<th>Transformation Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Streaming Service</td>
<td></td>
</tr>
<tr>
<td>Job Submitter Service</td>
<td></td>
</tr>
</tbody>
</table>
Remember that services can communicate in both directions. Transformation services – for example Data Download – are primarily self-serve tools for Server to deliver data to the end user.

Utility services can be described as “helper” services. They interact with FME Server to assist in menial tasks such as uploading data or providing token security. In most cases these are facilities that an author or developer will be using in a way that’s hidden from the user.

The Notification Service is used for passing short messages into and out of FME Server. Incoming messages notify FME Server to take some action, whereas outgoing messages alert an end-user (or system) that some sort of event has occurred. Automations are powered by the Notification Service.

**Workspaces and Services**

When a workspace is published to FME Server the last panel of the publishing wizard is for registering it with a particular service:

```
<table>
<thead>
<tr>
<th>Service</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Download</td>
<td>Edit...</td>
</tr>
<tr>
<td>Data Streaming</td>
<td>Edit...</td>
</tr>
<tr>
<td>Job Submitter</td>
<td>Edit...</td>
</tr>
<tr>
<td>KML Network Link</td>
<td>Edit...</td>
</tr>
<tr>
<td>Notification Service</td>
<td>Edit...</td>
</tr>
</tbody>
</table>
```

The Job Submitter service is automatically selected in the FME Server publishing wizard, whenever a workspace is published, but many other services are available too.

Registering a workspace with a service makes the workspace available for use in that service although, as you’ll discover, not every workspace is capable of being used by every service.

Be aware of the Edit button to the right of each service.

Every service has a set of parameters available that determine how a workspace will be run with that service:
Notice how these parameters include ones for notification topics to trigger on completion of the workspace.

**TIP**

*It’s important to understand that a workspace may be registered against one service, many services, or no services at all!*
Implementing Self Serve

Implementing a self-serve system makes use of the appropriate services available on FME Server.

Creating Data Download Services

Creating and using a data download service is very easy. A workspace becomes available for data download use when the author registers it against the Data Download service when publishing it to FME Server:

Once registered this way, Data Download becomes an acceptable way to run this workspace. This might be run in a number of ways, one of which is selecting Data Download as the service in the FME Server Web interface:

The results of the workspace are not written to a specific output location; instead they are delivered to the user in the form of a hyperlink to a zipped dataset:

Creating Data Upload Services
The Data Upload service is different in that a workspace is not registered specifically to this utility service. Instead, publishing source dataset parameters in the workspace allows data upload to take place.

On FME Server there are two ways to upload data, depending on the requirements for the data. There is a specific Data Upload Service and there is also the Resources filesystem.

The Data Upload Service uploads data to a particular workspace in a particular repository. The data is held temporarily for the workspace to be run.

The Resources filesystem allows data to be uploaded to a folder for use by any workspace in any repository. This upload is persistent and the data held there for as long as it required.

FME Lizard says...

You will have already met the Data Upload service: It’s what is used when data is published to a repository alongside a workspace in the Workbench FME Server Publishing Wizard.

When creating a customized solution that involves data upload, the developer can choose whether to use the Data Upload service or the Resources filesystem instead. The main difference is whether the data needs to reside on the Server permanently (Resources) or just temporarily (Data Upload).
As a technical analyst in the GIS department of your local city you have plenty of experience using FME Desktop, and are just getting started with FME Server.

Other departments frequently ask the GIS team for orthophoto imagery of the city. Their format of choice is usually JPEG. Currently you use FME Desktop to translate the data, adding to your workspace any special requests they have such as a particular resolution, a specific area of interest, or even sets of vector data stamped onto the raster.

However good you are with FME Workbench, it does take time to set up each of these individual requests. It would be far better if the other departments could help themselves to the raster data, with options for all of their special requests built in.

Of course, you very soon realize that a Data Download system implemented on FME Server would be an ideal solution.

FME Lizard says...

*If you have lots of experience with FME Workbench - and if your instructor agrees - simply open the workspace listed in the header above and skip to step 6. But make sure you inspect the transformers and their parameters, so you know what you are working with.*

**1) Create Workspace**

Let's start off by creating a basic FME workspace to translate and transform the source raster data. Start FME Workbench and select Readers > Add Reader. When prompted enter these parameters:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>GeoTIFF (Geo-referenced Tagged Image File Format)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\Orthophotos\06-07-LM.tif</td>
</tr>
<tr>
<td>Reader Parameters</td>
<td>Feature Type Name(s): From File Name(s)</td>
</tr>
<tr>
<td>Workflow Options</td>
<td>Single Merged Feature Type</td>
</tr>
</tbody>
</table>

It's important to use the Single Merged Feature Type option because there are many source tiles of data, and we may want to read any of them without having to add them as individual feature types.

The Feature Type Name parameter is important because it will help us later allowing the user to select which layers to read.

**2) Add Writer**

Now we need a Writer. Select Writers > Add Writer from the menubar. When prompted enter these parameters:

| End Workspace          | C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex1-Complete.fmw |
| End Server Project     | C:\FMEData2020\Projects\ServerAuthoring\SelfServe-Ex1-Complete.fsproject |
Your workspace will now look like this:

3) Add Transformers
We'll start out with two transformers in our workspace; a RasterResampler and a RasterMosaicker. So place one of each of these and connect up everything in the workspace:

4) Set Transformer Parameters
Inspect the RasterResampler's parameters and set:

<table>
<thead>
<tr>
<th>Resolution Specification</th>
<th>Cell Spacing</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Cell Spacing</td>
<td>5</td>
</tr>
<tr>
<td>Y Cell Spacing</td>
<td>5</td>
</tr>
</tbody>
</table>

You may inspect the RasterMosaicker's parameters, but there aren't any that need changing at the moment.
5) Save and Run Workspace
Save the workspace and - just to ensure that all is well - run it in FME Workbench. The result should be a JPEG file (06_07_LM.jpg) along with a world file (06_07_LM.wld).

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>06_07_LM</td>
<td>5/21/2019 4:37 PM</td>
<td>JPG File</td>
<td>23 KB</td>
</tr>
<tr>
<td>06_07_LM.wld</td>
<td>5/21/2019 4:37 PM</td>
<td>WLD File</td>
<td>1 KB</td>
</tr>
</tbody>
</table>

6) Publish Workspace
Now publish the workspace to FME Server. Register it with the Data Download service. Remember, you'll need to either publish the data with the workspace or upload it to an FME Server Resources folder.

7) Run Workspace
Log in to the FME Server web interface, locate the workspace, and run it.

The workspace will run and you will be presented with a hyperlink to a zip file of the output dataset:

**CONGRATULATIONS**

By completing this exercise you have learned how to:
- Create a workspace to read and write raster data
- Publish and run a workspace using the Data Download service
**Other Self-Serve Services**

Although most people look at self-serve mostly in the context of Data Download, that is not the only self-serve service available.

---

**Data Streaming**

Data Streaming is another service that a workspace can be registered against:

<table>
<thead>
<tr>
<th>Service</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Download</td>
<td>Edit...</td>
</tr>
<tr>
<td><strong>Data Streaming</strong></td>
<td>Edit...</td>
</tr>
<tr>
<td>Job Submitter</td>
<td>Edit...</td>
</tr>
<tr>
<td>KML Network Link</td>
<td>Edit...</td>
</tr>
<tr>
<td>Notification Service</td>
<td>Edit...</td>
</tr>
</tbody>
</table>

Whereas the Job Submitter service writes data, and the Data Download service returns a link to the data, a Data Streaming service returns a file or the data itself, streamed back to the client.

For example, if the Data Streaming URL for a workspace is posted into a web browser, the data will be automatically downloaded and opened in whatever application the browser associates with that file type (some data might open directly in the web browser itself).

Alternatively, the URL can be used directly as the source for a client application, like a GIS tool. When the client actively downloads the contents on a regular basis – as a GeoRSS reader would – then you have a feed, which is significantly different to a regular data download service.

---

**FME Lizard says...**

*Data Streaming is a slight misnomer in that a data streaming service does not supply a continuous stream of data; it merely provides a snapshot of the data at a particular point in time.*

---

**What Formats can be Streamed?**

You can use any workspace with the data streaming service, provided it writes data in a format that is file-based or folder-based (i.e. not to a database or web URL).

If an output dataset is comprised of more than one file, the data streaming service automatically creates a compressed (zip) folder out of the data. For example, AutoCAD DWG format could be streamed, whereas Esri Shape would be returned in a zipped file.

The most popular formats to stream are those that have a suitable client to read the feed. Some of the main formats that are output using the data streaming services include:

- RSS
- GeoRSS
- GeoJSON
- KML
MIME Types in Workspaces

A MIME header is a component of a file or e-mail message that is capable of indicating the content type of the file; for example, `Content-Type: text/plain` indicates a simple text file.

The application chosen to open a streamed file will depend on the MIME type and file association on the client’s system.

Setting MIME type is most important for FME writers where the content is not specifically defined by the writer. For example, the HTML writer has no MIME type setting because it is obviously providing text/html. The TextFile writer has a MIME type setting because the nature of its contents are ambiguous; it might be writing plain text (text/plain) or XML (text/xml) or it might even write the contents of a blob attribute containing a raster png image (image/png):

Here the author is saying that the content of the text file is valid PNG and should be opened in the default PNG application (possibly a web browser, possibly a graphic editor).

KML Network Link

Google describes a KML Network Link as a type of bookmark; a link from Google Earth (or any other KML-reading application) to the true set of data.

In FME you can register any workspace that writes KML as a KML Network Link service:

- HTML
- JSON
- XML
- PDF
- PNG
- JPEG
Running the workspace using this service merely returns a small KMZ (compressed KML) file that contains this "bookmark". The bookmark is a link back to the workspace using a URL that points to the Data Streaming service.

When Google Earth opens a KML Network Link service from FME Server, it receives the link to the workspace:

![Google Earth Network Link](image)

When the link is followed, it triggers FME Server into running the workspace and returning the results as a stream of KML data.

Because Google Earth permits a refresh rate for network links, the translation can be re-run at a user-defined interval. This way the results are always as up-to-date as the chosen interval.

Of course, in this scenario the output is never written to a permanent dataset; the resulting data is simply streamed to Google Earth, which writes it to a cache.
Exercise 3.2: Data Streaming System

<table>
<thead>
<tr>
<th>Data</th>
<th>Orthophoto images (GeoTIFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Goal</td>
<td>Create an FME Server Data Streaming system for orthophotos</td>
</tr>
<tr>
<td>Demonstrates</td>
<td>Data streaming</td>
</tr>
<tr>
<td>Start Workspace</td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex2-Begin.fmw</td>
</tr>
<tr>
<td>Start Server Project</td>
<td>C:\FMEData2020\Projects\ServerAuthoring\SelfServe-Ex2-Begin.fsproject</td>
</tr>
<tr>
<td>End Workspace</td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex2-Complete.fmw</td>
</tr>
<tr>
<td>End Server Project</td>
<td>C:\FMEData2020\Projects\ServerAuthoring\SelfServe-Ex2-Complete.fsproject</td>
</tr>
</tbody>
</table>

As a technical analyst in the GIS department of a city you have just created a system to allow other departments to download orthophoto data, rather than having to ask you to create it for them.

Sometimes the end-users download data as JPEG just to open it in a browser or image viewer to inspect it. You realize that, in cases like this, they may be able to use a data streaming service, instead of a data download.

1) Open Workspace
Open the workspace from exercise 1, or the begin workspace listed above.

2) Publish to FME Server
Publish the workspace to FME Server.

In the final dialog of the publishing wizard, check the boxes to register the workspace with both Data Download and Data Streaming (but don't click Finish yet):

![Edit Data Streaming Properties](image)

Click the Edit button for the Data Streaming service. Ensure that service is using the output of the JPEG Writer (for now we're limiting the streaming of data to JPEG format):

![Edit Data Streaming Properties](image)
3) Run Workspace
In the FME Server web interface locate the newly published workspace and run it. In the parameters for the workspace, be sure to set the web service to Data Streaming instead of Data Download.

5) Check Results
You should find that the results of the translation are returned as a streamed JPEG file. Most likely it will open directly in your web browser:

CONGRATULATIONS
By completing this exercise you have learned how to:
- Set up a workspace for use in a Data Streaming service
- Publish a workspace to the Data Streaming service
Self-Serve and Parameters

Workspace parameters are the key to controlling self-serve and the Data Download service.

What are Parameters?

Parameters are what control FME translations and transformations. Remember that a translation has a hierarchy of different translation components:

- Workspace Parameters
- Reader Parameters
- Writer Parameters
- Feature Type Parameters

Each different level of the hierarchy has a set of parameters that belong to it. That means there are:

- Workspace Parameters
- Reader Parameters
- Writer Parameters
- Feature Type Parameters

Most of the available parameters are determined by the author of the workspace.

FME Lizard says...

Order food at a restaurant and the chef will decide how long it needs to be cooked, at what temperature, and using what equipment. They will also decide the amount of seasoning it needs and what plate to present it on. Like an FME author, these are the parameters that control the results, and as the creator of the meal I get to choose how best to set them.

Published Parameters

Although workspace authors set most of the parameters, in some cases, the end user needs to be able to set some of them.
To enable users to select the requirements for their translation, FME includes functionality called *User Parameters*. User parameters are methods for getting input into a workspace.

When a user parameter is made available to the end-user, it is called a *Published Parameter*. In a self-serve application, published parameters are important for letting the end user control how the data is served.

**Parameter Uses in FME Server**

For self-serve systems, published parameters are most commonly used to set:

- What coordinate systems to deliver data in
- What feature types (layers) to deliver
- What geographic area (Bounding Box or Area of Interest) to deliver
- Any other Reader, Writer, or Transformer parameters of use to the user

With FME Server, the key to successful workspace authoring is flexibility. Workspaces need to be flexible to allow end-users to make choices without seeing all of the complexity of the workspace or the data behind it. Parameters are one way to accomplish this.
**Miscellaneous Published Parameters**

Any parameter in FME can be published and presented to the user as a choice to be made when running a workspace. They don’t have to be specifically related to the Data Download service.

Parameters are published in FME Workbench for use on FME Server.

**Publishing a Parameter**

In FME Workbench, parameters are located in several places, but the FME Workbench Navigator window is the one place where you will find them gathered in a single location:
As you can see, there are parameters for the source data location (1) and the destination dataset (2) (which is the one that the Data Download service overrides). There are also parameters for controlling the coordinate system (3), general Reader and Writer parameters (4), parameters for each feature type (5) and parameters for transformers (6).
Parameters are published by right-clicking on them and choosing “Create User Parameter.” For example, here a workspace author is publishing a parameter on the LabelPointReplacer transformer:

The author gets to choose the type of parameter (in this case a Choice of different fixed values), the user prompt, and the default value. When the workspace is run the end-user will now be able to decide what value to set the parameter to:

**Published Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plane Area</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Sloped Area</strong></td>
<td></td>
</tr>
</tbody>
</table>
Key Parameter Types

There are multiple types of parameters, mainly related to the data type required; for example integer, float, string. However, a notable parameter type for a self-serve setup is “Choice with Alias.”

Choice with Alias

A Choice parameter is one in which a user is presented with a list of choices, and their selection is returned directly to FME. A Choice with Alias parameter is the same thing - the user is presented with a choice - but this time a different value is returned to FME based on a lookup table.

For example, here the author wishes the user to select the height of labels to be created in the output dataset:

![Image showing Choice with Alias parameter]

The user is presented with the list of (more intuitive) options on the left, but the value returned to FME will be the (more useful) number on the right.

On FME Server, this parameter type is very useful for Data Download tasks. That's because many of the values we want to pass to FME - for example, format or coordinate system names - have an abbreviated format that wouldn't make sense if presented to the user. A Choice with Alias parameter allows us to provide a user-friendly list of options that hides a more complex response to the server.

TIP

Scripted parameters are a way to generate a parameter value using a Python or Tcl script. These parameters can be useful for generating values in a more complex way, e.g. creating a list of feature types from other parameters, or creating a parameter that depends on the value of other user parameters submitted at runtime.
Geographic Area of Interest

Some self-serve workspaces benefit from letting the user choose to download data within a geographic area of interest. There are three ways of doing so with FME:

- A simple, rectangular bounding box
- An existing boundary (any area feature, such as a regional or municipal boundary)
- An ad-hoc boundary (an area drawn by the user as needed)

Bounding Box

Letting the user define a bounding box is simple. Every reader can be restricted to only read features within a Search Envelope. The Minimum and Maximum X and Y parameters and the Search Envelope Coordinate System parameters can be published to give the user the ability to manually define a bounding box:

Existing Boundary

You can give your readers the option to choose an existing boundary, like a county or a service area, and download the data within that area. There are several ways to accomplish this with FME, but the most effective is to use a WHERE Clause on the reader feature type combined with either a FeatureReader or a Clipper transformer.

WHERE Clause

Many reader feature types have the option to apply an SQL WHERE clause to the data before reading it into the workspace. For example, this reader feature type has a WHERE clause that means only garbage zones in the blue zone and the north subzone will be read:
You can easily publish this WHERE Clause parameter to give users control over what existing boundary is used as the area of interest.

Even better, you could create specific user parameters that are then used within a WHERE clause that is already written. That way the use doesn’t need to know how to write a WHERE clause.

For example, you might create one Choice or Choice (Multiple) parameter called Zone and one called Subzone and then include them in the WHERE Clause parameter through the WHERE Clause Builder:
FeatureReader

Once you have setup a WHERE clause to let the user choose which boundaries to use, you then need to ensure they only receive their desired data within those boundaries. You can do this using a FeatureReader.

The polygon area of interest feature (or features) is routed into the FeatureReader Initiator port.

Here a single garbage schedule area (blue zone, north subzone) is passed to the FeatureReader for use as a filter. The idea is to return only addresses that fall inside that garbage collection zone. The parameters are set up like this:

- Transformer Name: FeatureReader
- Format: Esri Geodatabase (File Geodb Open API)
- Dataset: C:\FMEData2020\Data\Addresses\Addresses.gdb
- Parameters...
- Coord. System: Read from source
- Feature Types to Read: <All Feature Types>
- WHERE Clause:
- Spatial Filter: Initiator Contains Result
- Max Features to Read:

Notice that the FeatureReader is set to read from an address geodatabase. The Spatial Filter parameter tells the transformer to only read features (addresses) inside the incoming polygon feature (garbage collection zone).

These features are output through a port dynamically added to the FeatureReader.
The feature counts here show that there are 955 addresses inside that garbage collection zone. The user can pick a zone and subzone and just receive the data from that area.

**Ad-hoc Boundary**

---

**NEW**

*FME 2020.0 introduces the Geometry parameter, a powerful new feature that lets FME Server users interactively define their geographic area of interest. The parameter accepts a polygon in GeoJSON format. To define it on Desktop, a user has to paste in GeoJSON. To define it on FME Server, they can interactively draw their area of interest on a web map.*

---

To let users interactively draw their own area of interest, you can use the Geometry user parameter. In this example, we want to create a workspace where the construction company can draw on a map where they will be working. Then a 100m buffer surrounding their construction area will be set and notify the residents of the neighboring houses.

1. We use a FeatureReader to read in the address points, and a Reprojector to put them into the UTM83-10 coordinate system.
2. We create a Geometry published parameter that lets the user provide their area of interest.
3. We then use that parameter to create geometry from GeoJSON using the GeometryReplacer.
4. We use another Reprojector to make sure the area of interest matches the rest of the workspace.
5. Then, we add a Bufferer to buffer 100m around the area of interest to capture addresses within the area affected by construction noise.
6. After that, we use a Clipper to clip the source address data to the area of interest.
7. At this point the features have quite a few attributes the construction company won't need, so we employ an AttributeKeeper to keep only the necessary attributes.
8. Finally, we add an Excel writer feature type to create a spreadsheet that can be used by the construction company to notify affected addresses.
When this workspace is run on FME Server, the user can select their area of interest using an interactive webmap:

TIP

Previous versions of FME can accomplish this, but they require using the FME Server REST API and HTML/CSS/Javascript. You can view an example on our Knowledge Base.
### Exercise 3.3  
**Data Download System: Published Parameters**

<table>
<thead>
<tr>
<th>Data</th>
<th>Community Mapping (Esri File Geodatabase)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Goal</strong></td>
<td>Create an FME Server Data Download system community mapping data, allowing the user to choose the format and area of interest</td>
</tr>
<tr>
<td><strong>Demonstrates</strong></td>
<td>Creating published parameters for user control in Data Download</td>
</tr>
<tr>
<td><strong>Start Workspace</strong></td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex3-Begin.fmw</td>
</tr>
<tr>
<td><strong>End Workspace</strong></td>
<td>C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex3-Complete.fmw</td>
</tr>
<tr>
<td><strong>End Server Project</strong></td>
<td>C:\FMEData2020\Projects\ServerAuthoring\SelfServe-Ex3-Complete.fsproject</td>
</tr>
</tbody>
</table>

As a technical analyst in the GIS department of a city, you have just commenced an initiative to allow other departments to download community mapping data, rather than having to ask you to create it for them. Not only will their requests be processed quicker, but you will also spend less time on that task.

So far you have created a workspace that allows users to choose the format for their data download.

Now you need to add a Geometry published parameter to let users interactively choose their area of interest.

1) **Open Workspace**  
Open C:\FMEData2020\Workspaces\ServerAuthoring\SelfServe-Ex3-Begin.fmw.

2) **Inspect Published Parameters**  
The starting workspace is in-progress. It already has a published parameter that lets users choose the output format. You can find them by looking at the Navigator > User Parameters > Published Parameters. Right-click FORMAT and choose Edit Definition to view its configuration:

   ![Published Parameter](image)

   This parameter lets the user choose the output format for the data they receive. The default is Microsoft Excel. Click the ellipsis button next to Configuration to view the options for this Choice with Alias parameter.
This parameter gives the user the option of four output formats (GeoJSON, OGC GeoPackage, Esri Shapefile, or Microsoft Excel). Using Choice with Alias like this lets you provide a set of formats or coordinate systems to the user, instead of letting them pick from the entire list. This option can be beneficial as it is less overwhelming to the user and can prevent incorrect outcomes. Click Cancel twice to close the parameter dialog.

**TIP**

You can learn how to create user parameters like this in this chapter from our FME Desktop Advanced training.

3) Create a Geometry Published Parameter

Now let's edit this workspace so the user can define the area where construction will be occurring. The first step is to add a Geometry published parameter. In the Navigator, right-click User Parameters and choose Create User Parameter:

In the Add/Edit User Parameter window enter the following:

<table>
<thead>
<tr>
<th>Type</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>GEOM_COORDS</td>
</tr>
<tr>
<td>Prompt</td>
<td>Select construction area:</td>
</tr>
<tr>
<td>Published</td>
<td>Checked</td>
</tr>
</tbody>
</table>
Optional | Unchecked
--- | ---
Attribute Assignment | Off

For Configuration, click on the ellipsis and enter the following:

<table>
<thead>
<tr>
<th>Geometry Types</th>
<th>Box, Polygon, Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify initial bounds for map display</td>
<td>Checked</td>
</tr>
<tr>
<td>Top (-90..90)</td>
<td>49.2548</td>
</tr>
<tr>
<td>Left (-180..180)</td>
<td>-123.244</td>
</tr>
<tr>
<td>Bottom (-90..90)</td>
<td>49.3034</td>
</tr>
<tr>
<td>Right (-180..180)</td>
<td>-123.071</td>
</tr>
</tbody>
</table>

The initial bounds will be the area shown in FME Server. Larger bounds will have the map zoomed out, and smaller bounds will have the map zoomed in.

Click OK twice to close the parameter dialogs.

4) Create the Area of Interest Polygon

Now that we have set up the geometry published parameter, we need to use it within the workflow. Add a GeometryReplacer in a new stream after the Creator transformer.
Open the parameters for the GeometryReplacer. Set the Geometry Encoding to GeoJSON and then set the Geometry Source to the GEOM_COORDS published parameter.

Click OK.

5) Reproject the Area of Interest
We want to ensure that FME knows our data is in LL84, as this is what the Geometry published parameter accepts as values. Add a CoordinateSystemSetter transformer after the GeometryReplacer. In the parameters, set the Coordinate System to LL84.
Our source address data is in UTM83-10. It is more appropriate to buffer and intersect data in a projected coordinate system, so we will reproject both streams of data to UTM83-10. Add the first Reprojector after the CoordinateSystemSetter. Set the Destination Coordinate System to UTM83-10.

Add a second Reprojector between the FeatureReader and the NotifyList writer feature type. We have to make a slight change to the address data coordinate system. Set the Destination Coordinate System to UTM83-10 and click OK.

Your workspace should look like this:

6) Buffer the Area of Interest
We need to add a 100-meter buffer around the area of interest to find which neighboring residents might be affected by construction noise and must be notified. Add a Bufferer transformer connected to the first Reprojector. In the parameters, set the Buffer Distance to 100 and set the Buffer Distance Units to Meters.
7) Clip the Addresses to the Area of Interest
Now we need to apply the buffered area of interest to our data. To do this we will use a Clipper transformer. Add a Clipper transformer to the canvas and connect the Bufferer's Buffered port to the Clipper input port. Then connect the Reprojector_2 to the Clippee input port. In the Clipper parameters, enable Merge Attributes.

Click OK.

8) Clean up Attributes
One final step before we can write out our data is to clean up the attributes. Add an AttributeKeeper to the canvas and connect it to the Inside output port on the Clipper.

In the parameters, for Attributes to Keep, select:
- OWNERNM1
- PSTDADDRESS
- PSTLCITY
- PSTLPROV
- POSTALCODE

Click OK.
9) Test Writing Results to Shapefile
Let's test our workspace by writing the results to a Shapefile. Connect the AttributeKeeper to the NotifyList writer feature type and run your workspace. Select Esri Shapefile as the output format.

For the Geometry parameter, we have to supply GeoJSON to test on FME Desktop. On FME Server you can use a web map. Paste the following GeoJSON code in to test:

```
{"type":"Polygon","coordinates":
[[-123.131762,49.282752],
[-123.132148,49.282465],
[-123.131579,49.282087],
[-123.131139,49.282332],
[-123.131762,49.282752]]}
```

When the translation finishes, click the NotifyList writer feature type once to select it, and then click View Written Data. Specify Esri Shapefile for the Format and add NotifyList.shp to the end of the Dataset parameter:

Click OK. The addresses to notify, those within 100m of the area of interest, should appear in the Visual Preview window.

TIP
We have provided GeoJSON code for testing the Geometry parameter. If you want to get your own GeoJSON to test, you can publish your unfinished workspace to FME Server, fill out the Geometry parameter, and copy the resulting GeoJSON code. Alternatively, you can use an online service to generate the GeoJSON for you, e.g. https://geojson.io/. Just remember the parameter expects a single feature.

10) Change Data Source to Shared Resources Folder
To ensure everyone can run this workspace on FME Server, we should change the address GDB dataset parameter to look for the data on FME Server's Shared Resources Data folder instead of the C:\ drive. We will do this before publishing it to FME Server, and then we can upload the data there when we publish the workspace.
Double-click the FeatureReader to open its parameters. Change the Dataset parameter to 
$(FME\_SHAREDRESOURCE\_DATA)\backslash Addresses.gdb$. This path will look for the GDB in the FME Server Data folder. To ensure the FeatureReader still works when reading from FME Server, click Output > Output Ports > Single Output Port.

Next, expand the Attribute and Geometry Handling dropdown. Then, expand the <Generic> Port dropdown as well. Since, FME Desktop read the Dataset file we need to import the schema for the file manually.

Click the ... button besides the attributes to expose bar. Then, select Import... in the Attributes to Expose dialog.
Under Format, select Esri Geodatabase (File Geodb Open API). Then, under dataset enter C:\FMEData2020\Data\Addresses\Addresses.gdb

Click Next to continue. Then, under Feature Types select PostalAddress. Then, click Next again.

Under Attributes to Expose select Select all. Then, click Import to complete the process.

Now click OK twice to exit the FeatureReader.

Note that this workspace won't work on FME Desktop now, but it will run properly on Server. Click OK. The Addresses output port disappears; connect the <Generic> port to the Reprojector_2.

11) Publish to FME Server
With the workspace complete, we can now publish to FME Server. Click on the Publish button on the toolbar. Select the Training FME Server connection and the Training repository. Select Data Download and Job Submitter as the Services.

After publishing, make sure you upload the GDB to FME Server's Data folder. In FME Server, go to Files & Connections: Resources. Then click the Data folder, then the Upload button, and choose Folder. Navigate to C:\FMEData2020\Data\Addresses\Addresses.gdb and upload it.

12) Test on FME Server
Click the Direct Link to the workspace in the Translation Log (http://localhost/fmeserver/#/workspaces/run/Training/SelfServe-Ex3-Complete.fmw/) and login if necessary.

Select Training as the Repository and set the Service to Data Download.

For Published Parameters, you should see the Select construction area parameter. Click on the map icon to select the area.
When you click on the map icon, the Geometry Picker dialog will appear, which will have a map of the area set up in the published parameter, as well as options along the side to determine which geometry type you wish to use for your selection.

Zoom into the area of interest. For this example, we will zoom into downtown Vancouver and use the polygon tool to pick a small area of buildings.

Note: For this workspace to run correctly, buildings with addresses need to be selected. If an area is selected without an address the workspace won’t write out any data. To close off an area in a polygon, double-click on the starting point.
Once the area is selected, click Confirm. GeoJSON coordinates will now be available in the Select construction area text box.

Once the map area has been selected, click Run to run the workspace.

When the workspace has finished running, click on the download link to get the data. Extract and open the data in Microsoft Excel or FME Data Inspector. This list can then be used to send letters to the residents affected by the construction.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OWNERNM1</td>
<td>PSTLADDRESS</td>
<td>PSTLCITY</td>
<td>POSTALCODE</td>
</tr>
<tr>
<td>2</td>
<td>Brandon Scanlan</td>
<td>1188 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>3</td>
<td>William Haggard</td>
<td>1234 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>4</td>
<td>Sheba Bogle</td>
<td>1222 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>5</td>
<td>Guy Eck</td>
<td>1137 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>6</td>
<td>Shu Najar</td>
<td>1095 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>7</td>
<td>Ayako Speno</td>
<td>1156 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>8</td>
<td>Tomi Atterbury</td>
<td>1162 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>9</td>
<td>Neida Poulsen</td>
<td>1140 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>10</td>
<td>Reena Sokoloff</td>
<td>1160 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>11</td>
<td>Setsuko Feldmann</td>
<td>1168 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>12</td>
<td>Tyisha Hullett</td>
<td>1150 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>13</td>
<td>Duncan Scaglione</td>
<td>1230 Comox St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>14</td>
<td>Tomi Atterbury</td>
<td>1220 Comox St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>15</td>
<td>Madeline Reno</td>
<td>1065 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>16</td>
<td>Inga Fitzhenry</td>
<td>1122 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>17</td>
<td>Antonia Landin</td>
<td>1116 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>18</td>
<td>Kimi Rozelle</td>
<td>1110 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>19</td>
<td>Reena Sokoloff</td>
<td>1104 Bute St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
<tr>
<td>20</td>
<td>Stanton Kush</td>
<td>1173 Pendrell St</td>
<td>Vancouver</td>
<td>British Columbia</td>
</tr>
</tbody>
</table>
By completing this exercise you have learned how to:

- View user parameters designed for FME Server workflows
- Create a Geometry published parameter
- Use a Geometry published parameter in a self-serve data download workflow
- Publish a workspace and use published parameters
Module Review

This module introduced you to FME Server’s self-serve services and related functionality in FME Workbench.

What You Should Have Learned from this Module

The following are key points to be learned from this module.

Theory

- Services handle communication between FME Server and its clients. Each workspace published to FME Server can be registered with any number of services.
- The Data Download service overrides the destination dataset parameter and presents data to the end-user as a link to a zip file.
- Other FME Services allow data to be streamed live into a suitable application.
- Parameters control the different actions of a Data Download system, including the output coordinate system.
- The Generic Writer is a writer whose format is only defined at runtime. This can be provided by the user through a published parameter.
- Selection of source data by layer is achieved using the Feature Types to Read parameter
- Selection of source data by area can be achieved using Bounding Box parameters, a transformer such as the Clipper or FeatureReader, or the new Geometry published parameter

FME Skills

- The ability to publish a workspace and register it to the FME Server Data Download service
- The ability to use Data Streaming services
- The ability to create and use published parameters
- The ability to give the end-user control over format, coordinate system, and layers
- The ability to give the end-user control over the area of interest

Further Reading

For further reading why not check out this guide to sharing open data using FME Server, or this fine blog article that explores how to implement an open-data portal using FME, Amazon S3, Leaflet, JQuery, and a whole lot more!
Questions

For each of these scenarios, tell me if it is a Data Download project, Data Upload, both, or neither.

The user logs on to a web page, draws an area on the map, and is sent a copy of the data within that area The user submits a dataset to a web site that scans the data for errors, and returns a corrected copy The user publishes a workspace that writes data to the user's account in an online PostGIS database The user starts a GIS application, clicks File > Add Data to Map, and pastes in a URL from FME Server 

#1 is a simple download of data. #2 is obviously both. #3 is neither. Publishing a workspace is not Data Upload and writing to a database is not Data Download. #4 would be a special type of Data Download. In FME we call it Data Streaming; the workspace runs and the output is sent directly to the application that requested it.

When a workspace is not registered against any service, how can you run it? Select all that apply.

With the FMEServerJobSubmitter transformer
With the run dialog in the web interface
With the URL specified under Advanced > Webhook URL in the run dialog
By setting it to run under a schedule
You cannot run the workspace in the web interface (the run button won't work unless services are available) or with a URL (none will be provided).

When a workspace is registered against the Data Download service (and no other), how can you run it? Select all that apply.

With the FMEServerJobSubmitter transformer
With the run dialog in the web interface
With the URL specified under Advanced > Webhook URL in the run dialog
By setting it to run under a schedule
It will actually run under all of these tools. Of course, only the web dialog and the URL return a zip file for download. The others just output the data to the specified workspace location.

Why is the FeatureReader the preferred option [for filtering data in a self-serve system]? Pick all the reasons that apply:

- It can be quicker and more resource efficient
- It allows multiple areas to be used as the existing areas It works with raster data
- It has more choices for spatial filtering
- The FeatureReader is more efficient where the source dataset is a format that has a spatial index, because it won't have to read all of the source data. It also has more filtering choices than a Clipper transformer (though not a SpatialFilter)
- The FeatureReader does allow multiple areas to be used, and it does support raster data - however so does the Clipper transformer, so there is no difference between the two in those scenarios. Additionally, the two are equivalent in performance where there is no spatial index (i.e. both will need to read the full dataset)
- How well do you know the types of FME published parameters? Decide which of the following are real parameters

Color Double Password Text (Multiline) There are over 25 types of published parameters.

If the Generic Writer parameter is published to determine what format to write data in a data download system, what would the Generic Reader parameter be used for?

To determine what format of data to read in a Data Download system To determine what format of data to read in a Data Upload system To determine the correct Styler transformer to use in the workspace To determine whether I'm connected to a Data Upload or a Data Download system Right, I'm uploading some data and I'm going to tell FME Server what format it is. Why would I care (#1) about what format of data is being read in a Data Download system (and how could I tell)? Similarly (#3) it's the output format that determines which Styler to use, not the input format. #4 is just plain nonsense!
Real-Time with FME Server

Real-time systems are those that act on events as they happen, and send information as it becomes available.

Real-world examples of data that would arrive as individual events include processing simple location data from a vehicle tracking system or sending an email to a system administrator in response to database table updates. An example of continuous data would be handling a stream of data being sent from a temperature sensor or lightning detector.

Real-time data in FME Server is handled in two ways: Automations and Message Streams.

Automations

Automations are how FME Server handles individual message streaming and alerts.

FME Server allows you to build Automation workflows that can be set up to listen for an incoming message from outside of FME and trigger a certain action in response or to send a single alert in response to an event that takes place on FME Server.

Triggers and actions include email, FME Server system events, directory watchers, WebSockets and more.

Message Streaming
Message Streaming is a real-time technique like Automations. However, where Automations receive/send one-off messages, Message Streaming involves a continuous flow of information.

Instead of a workflow being run once for each message, handling a message stream involves creating a workspace that will connect to the stream and run continuously receiving messages as soon as they are sent. Because of this reduced overhead, this technique can process data at a much faster rate than Automations. For our purposes, "continuous" means that messages arrive at the FME Server at a faster rate than Automations could handle; say more than one message per second.

Message Streaming will be explained in more depth in Chapter 5.
Automations

Automations provide a way for data to be pushed to FME Server in the form of short messages and react to these messages by triggering actions either internally or to external clients.

What is an Automation?

Automations are built on the concept of the FME Server Notification Service using a simple message (sometimes called an “alert”) that informs someone or something that a particular event has happened.

Automations in FME Server are made up of two different components: incoming triggers and outgoing actions.

Triggers alert FME Server to an event that has taken place either on FME Server or on an external application.

Actions are divided into two categories: Internal and External. Internal actions submit jobs to run on FME Server whilst External actions either send notifications to a client that an event has taken place on FME Server, or push result data out to a location.

In this way, FME Server can take action in response to an event notification or a user can take action in response to a notification from FME Server.

Automations is the part of the FME Server architecture that handles all incoming and outgoing notifications.

When to Use Automations

Automations allows you to incorporate a variety of triggers and actions into a single workflow, as such Automations should be used when you want to chain a number of different notification elements together in FME, or build multiple reactions to a single event. The event should not be a continuous series of messages; if there is more than one message per second you should consider using Message Streaming techniques instead.

As well as using Automations to trigger an FME Server response to something that happened outside of FME Server, they are also useful when you want to send a message about something that happened on FME Server to an external client. Again, if there is likely to be more than one message per second you should consider using Message Streaming.

In either case, an automation is meant for sending a short message, usually in order to trigger an action from the recipient.
Elements of Automations

Automations includes a number of different components:

- **Clients**: External users or systems that send or receive a message
- **Triggers**: Event handlers that listen for incoming messages
- **Internal Actions**: Event handlers that submit tasks to FME Server
- **External Actions**: Event handlers that dispatch outgoing messages to external services
- **Protocols**: Methods by which FME Server can receive or send notifications
Clients

A client is a user or system that sends or receives a message to/from FME Server. The client may be a physical person or may just be a component in a computer system. Either way, the majority of the time, a client is not a core part of FME Server, rather someone or something that interacts with it. There are two trigger protocol exceptions to this (Schedules and System Events) whereby FME Server is also the client sending the message.

For example, a database update might cause a trigger to send a notification to FME Server, in which case the database system is the client. However, a client could also be a person who, for example, triggers a notification by sending an email to FME Server.

Likewise, FME Server can send a notification for another client system to receive. Alternatively, this client can also be a real person, who might receive a notification in the form of an email.
**Trigger**

A trigger is an FME Server component that receives incoming notifications from a client.

To set up a trigger in FME Server, a workspace author (or administrator) must build a new Automation. An Automation is created in the FME Server Web Interface on the Automations > Build Automation page:

![Trigger Diagram](image)

**Action**

An action is an FME Server component that sends outgoing notifications to a client.
To send a notification in FME Server, a workspace author (or administrator) must first create an Automation that contains a trigger, the outport of this node is then connected to the action - just like how you connect transformers in FME Workbench.

Comparable to the FME Workbench canvas, Automations are constructed in a user friendly drag and drop interface. Triggers and actions are displayed on the canvas as nodes identifiable by their different color. Input and output ports are used to create connections that will pass messages from the trigger to the associated action. The user must configure a series of parameters for the Automation to succeed dependent on the protocol.
Protocols

A protocol is a system of data exchange between FME Server and a client.

We know that FME sends and receives messages. Protocols are the method by which these messages are sent and received. Each Trigger and Action is defined using a particular communication protocol.

To trigger an incoming message by email – for example – you would create an Automation using an Email Trigger. To send a notification to a Java Message Service, you would create an Automation that contains a JMS message Action.

There are many different protocols available in FME Server; some of them are only for use as a Trigger, others are only available as Actions, and some of them can be used with both types.

This table lists the different Trigger and Action protocols and the following pages go into greater detail on some of the most commonly used types.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
<th>Trigger</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon S3</td>
<td>Monitoring activity in an AWS S3 bucket/Communication to Amazon’s Simple Storage Service</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Amazon SNS</td>
<td>Communication with Amazon’s Simple Notification Service</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Amazon SQS</td>
<td>Communication with Amazon’s Simple Queue Service</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Azure Event Grid</td>
<td>Communication with Microsoft Azure Events</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Directory Watch</td>
<td>Monitoring activity in a Directory</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Dropbox</td>
<td>Monitoring activity in a Dropbox folder/Communication of a notification (file) to the Dropbox web service</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Email (IMAP)</td>
<td>Receive email messages from an IMAP Server</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Email (SMTP)</td>
<td>FME Server itself receives email messages/Communication via an email server</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Filter</td>
<td>Test incoming messages for the presence of a specified string</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>FME Server Topic</td>
<td>Receive/Send messages to a Topic to initiate workflow processes</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>FME Workspace</td>
<td>Run an FME Desktop Workspace published to FME Server</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>FME Dynamic Workspace</td>
<td>Run an FME Desktop Workspace published to FME Server, with the repository and workspace name coming from output keys in the Automation</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>FTP</td>
<td>Monitor activity/upload a file to an FTP Site</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>HTTP Request</td>
<td>Post HTTP requests to the specified URL</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>JMS</td>
<td>Communication with a Java Message Service</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Log</td>
<td>Write Event/Automation details to a log</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Merge</td>
<td>Combine messages before proceeding downstream in an Automation</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>SFTP</td>
<td>Communication with an SFTP site</td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>
Protocols are pre-defined components in the FME Server architecture and do not need to be defined in the web interface. However, a number of fields are made available to configure when an Automation is created.

For example, here are the parameters for a Directory modified Trigger:

These parameters must be set when the Automation is created as they are needed in order for FME to be able to watch and receive notifications about this Directory. Without configuring these parameters the Automation will have no behave as anticipated when enabled.
Automation Protocols

Email and FME Workspace protocols are among the most popular in FME Server, and they are covered in detail in the next few sections. Some other commonly used protocols are:

Watch Triggers

There are several watch triggers: Directory Watch and FTP Watch are the two most common but other watch services include Amazon S3 buckets and Dropbox directories.

These are trigger-only protocols that monitor folders and FTP sites for any changes, whether that be new files to be added, or existing files modified or deleted. When FME Server sees a change, the trigger message contains the affected file path value.

A common, and perhaps the primary use case for this, is the ability to monitor a folder where datasets are stored. When a new dataset is placed in this folder, FME Server Automations will pass this file path into a workspace to process the data with no need for any manual intervention.

This protocol allows files to be filtered by the action type so that it is not just the creation of a new file that can be monitored. It's also possible to monitor for existing file deletions and modifications.

Some of these watch trigger types have corresponding action protocols whereby FME Server can push files out to that file storage location.

Amazon Triggers/Actions

FME Server Automations support several Amazon protocols. SNS and SQS are two types of notification systems. FME Server is capable of pushing messages to both of them and receiving messages from both of them. The difference (if you are interested) is that SNS pushes messages to triggers immediately, whereas SQS stores the messages in a queue until the trigger fetches them.

Amazon S3 is an online file storage service. It can be either a trigger or action in FME Server; for example, FME can be notified to read from S3 “buckets” and can issue a notification in the form of writing to an S3 bucket. The S3 Trigger also falls under the "Watch" type category.

WebSocket Notifications

A WebSocket is a TCP-based line of communication. FME Server can accept and send messages from/to another WebSocket client. WebSocket protocols are supported by most web browsers, meaning any web application can communicate to FME Server and FME Server can send notifications (even small quantities of data) directly to any web browser.
As a technical analyst in the GIS department, you want to start experimenting with Automations in FME Server. The Directory Watch protocol seems like a good place to start, and you were already thinking about a shared folder where users place Shapefile datasets for adding to, or updating, the corporate database.

1) Create Resources Folder
Connect to FME Server (admin/FMElearnings). The first step is to create a Resources folder to upload the data. Open the FME Server web interface and navigate to the Files & Connections > Resources page.

Browse to the Data folder and create a new subfolder called BuildingUpdates:

FME Lizard says...

This exercise utilizes the FME Server Resource folders, but you could also watch a Directory outside of FME Server Resources by creating a Resource Connection or reference a shared UNC path. There is also native support in FME Server to watch for new resources in Amazon S3 Buckets, Dropbox, and FTP. Using the same concepts described here, you could use one of these protocols instead of Directory Watch.
2) Create Automation
Now to create the Automation that will watch the BuildingUpdates Directory for incoming files. Navigate to Automations: Build Automation on the side menu bar. In the Getting Started dialog that appears when you go to the Automations page for the first time, click on the Build tab and click Create New to start a new Automation.

By default, Automations starts in guided mode. This means that there is already a Trigger node on the canvas but it will still need to be configured.

Start by double-clicking the Trigger and a parameter box will appear on the right hand side of the canvas. Select Directory modified from the drop-down list as the Trigger for this Automation.

3) Define Trigger Parameters
After selecting a Trigger type a list of configurable parameters appears in the dialog. Click the ellipsis button to browse the FME Server Resources and set the Directory to Watch parameter. Select the newly created BuildingUpdates folder under the Data subfolder:

Leave the Watch Subdirectories and Watch Folders parameters set to No, since we are only interested in monitoring for files in the BuildingUpdates folder directly.
Then for the Events to Watch for parameter remove the MODIFY and DELETE actions. Since we are looking to add to the corporate database, in this example we are only interested in monitoring for new files arriving, not old ones being changed or removed:

Lastly, change the Poll Interval to 30 Seconds and then in the bottom left corner, click on the Validate button to ensure the trigger was set up correctly. Now click Apply to save these parameters. In the canvas the Trigger node will update to show it is a Directory Watch.

Save the Automation by selecting Menu > Save As and name the Automation "Incoming Building Footprints".

4) Log Message
Before we start the Automation, we need to add an Action so the Trigger protocol can parse the notifications onwards. Before processing the data we first want to check the Directory Watch trigger is working as expected. To do this we can send the incoming messages to a log file located on FME Server.

Select the Next Action node and set the Action to Log a message.
Click on the drop-down arrow for the Message parameter and select Event > Event as JSON because in this instance we want to record the entire incoming message from the Directory Watch protocol.

**TIP**

A trigger stores the incoming message event details as JSON however for the standard protocols FME is able to flatten the JSON down into its separate elements so the workspace or other subsequent action does not need to be able to handle this.

If your trigger contains information from a webhook that is buried in the JSON you can parse the entire message into a parameter to flatten using JSON transformers in FME Workbench.

Click Apply to save this Log configuration.

5) Start Automation

In order for FME Server to start watching the directory for incoming files, the Automation must be enabled. Select the Start Automation button in the top right corner. The button will turn red and an orange warning ribbon will appear across the canvas indicating that your Automation is currently running. This means FME Server is now checking that directory every 30 seconds for updates.

**FME Lizard says...**

As your Automation grows you may wish to turn off Guided mode, this can be done by selecting Hide
6) Test Automation

Now let's test the Automation. Locate the source Shapefile datasets in C:\FMEData2020\Data\Engineering\BuildingFootprints. Select a set of files (.dbf, .prj, .shp, .shx) and upload these files into the newly created Resources folder. There are two ways to do this.

You can use the file system (by copying the files to C:\ProgramData\Safe Software\FME Server\resources\data\BuildingUpdates) or use the FME Server web interface.

Check for updates in the log by selecting Menu > View Log File from the Incoming Building Footprints Automation.

If the log is not yet present, select the refresh button until it appears. You will see four CREATE messages from the Logger showing the individual file paths.
When you view this log file you might also notice reports of the Automation sending Jobs to FME Engine. This is because FME Server is actually performing the Log action using a simple FME Workspace.

FME Lizard says...

Remember, the Poll interval is set up to check the folder only once per 30 seconds - so if the Log file doesn't immediately appear, don't panic! Be patient and it will appear shortly.

Now we know how the Directory Watch Trigger works! We will see in subsequent exercises how to process this information.

CONGRATULATIONS

By completing this exercise you have learned how to:

- Create a new Automation
- Use Directory Watch to poll an FME Server Resource
- Test a Directory Watch trigger by reading a log
Automations and Workspaces

Workspaces are what make FME Server the ideal engine for data notifications. That's because the same functionality used to carry out spatial and tabular data transformations is also perfect for creating and transforming notification messages.

Message Transformation

A Simple Automation Setup

An Automation requires a minimum of two components. The simplest is where a trigger sends an incoming message to an action:

Using Automations you do not need to perform data processing tasks and could deliver notifications without even taking up an FME Server Engine. For example, someone sends an email to FME Server, which triggers an outgoing email in response.

However, that scenario does not include any transformation/restructuring of the message contents. If the message needs to be processed in some way then an FME workspace can be employed within FME Server.

A workspace is the foundation of FME. For Automations, it can be used to read an incoming message, extract spatial data from the message (regardless of format), carry out spatial transformations on that data, and then write the results in some way. The workspace can even read in extra data against which the message is to be processed.

It can also generate an outgoing message - possibly in response to some other spatial processing - and pass that on to an action.

This blend of live messaging with Spatial ETL is unique.

Let's look at two specific scenarios:

- A workspace runs in response to an incoming message
- A workspace runs and triggers an outgoing message

Workspaces Responding to Incoming Message

Let's think about this logically. An incoming message is received through a trigger. For a workspace to respond to an incoming message, there must be mechanisms for it to listen to that trigger and receive the information from it.

We already have terms for these mechanisms: **Triggers in Automations**
Yes, this scenario is set up by creating an Automation. The workspace action is a response to the trigger and receives the message content from it.

An example here is a trigger holds data - such as in the previous exercise - and goes on to submit a workspace that writes the incoming data to a database.

**Workspaces Triggering an Outgoing Message**

To continue the logic, a subsequent outgoing message is activated by the workspace action. For a workspace to cause that outgoing message. Again we already have terms for these mechanisms: *Actions in Automations*

This scenario is a little different, this time the workspace can trigger an action indirectly in an Automation by passing a message through an FME Server Topic. This means that the workspace can be run manually, or via another mechanism such as the Rest API call, that isn't listed as an Automation Trigger type but still enables the author to have a notification response triggered.

An example here is a workspace, started as a scheduled task, that sends a notification email to an administrator once complete.

**Full System**

Of course, the above diagrams show half-systems; i.e., the workspace either responds to a message (it is an Action) or it causes a message (through a Topic Trigger).

However, it is just as appropriate to have a workspace that is both an Action and a Trigger.
For example, a Trigger sends a message containing an emergency event and the coordinates of the client's position. A workspace runs in response to that emergency, processes the coordinates, and sends a new message on to an emergency response unit.

Notice that this setup requires one Trigger and two Action objects in FME Server. There is a trigger from a client, a workspace that acts as both an internal action and a trigger, and an external action pushing information back out to a client.

NEW

In FME 2020, a new FME Server Automations Writer has been added. It can be used to add additional output ports to any Workspace Action. This writer will allow for sending data from within the workspace directly to other Actions within an Automation and also for sending multiple output messages out of the Workspace.
**Workspaces as Actions**

In Automations Run Workspace is labelled an action since it reacts to an incoming message and therefore must be set up by being connected to a Trigger.

Within an Automation the Action protocol is called (as you might have guessed) *Run a Workspace* and falls under the Internal Action Category (orange node).

After selecting Run Workspace as the Action type, the user must first specify a Repository and workspace, the selected workspace is then examined and a list of its published parameters provided:
Having the parameters in a dialog like this means it is simple and easy to set up a workspace to run however you want it to in response to a trigger. Once created, whenever that Trigger receives a message from the sending client, the action will react by submitting that workspace to run on FME Server.

**NEW**

In FME 2020.0, you can now choose the *Run a Dynamic Workspace* action when building Automations. This action lets you run a workspace determined by the Output Keys of a previous Trigger or Action in your Automation. This lets you dynamically run workspaces in response to events earlier in the Automation. For example, you might run one workspace that reads in some data and determines which
workspace to run, and then outputs a key with the workspace Repository and name. The Run a Dynamic Workspace action can then be used to run the correct workspace.
Message Content

The ability to interpret and process message content is a key reason for using FME workspaces as actions. They can receive a message and then transform it in whatever way is required.

Passing Messages

A trigger passes messages to the workspace action. When it does, the message is written into a temporary JSON file. For the majority of protocols, the JSON elements are known and flattened within the Automations window so it is easy to retrieve the information downstream in the Automation without digesting the entire JSON event. However, under certain scenarios, you may wish to pass the entire event as JSON into a workspace.

The webhook protocol is a good example of this. Webhooks have a post URL which means they can be set up to receive messages from a third party application. In this instance, the JSON may contain multiple layers with attribute names not recognized by FME Server.

In order to retrieve this body, the full message can be passed into the Workspace for processing through a published parameter.

Interpreting the Message

An incoming message can be scanned and processed with a number of different transformers. If the messages are in JSON format there are transformers such as the JSONExtractor and JSONFlattener. Similarly there are XMLFlattener and XMLFragmenter transformers for XML content.

These transformers will convert the message from a JSON (or XML) string and into attributes that FME Workbench is able to process.

Here - for example - an author kickstarts the workspace using a Creator, and then has a ParameterFetcher to bring in the JSON message and finally a JSONFlattener to query and expose the attributes in the Topic message:
If the incoming message was a directory watch, then the JSON content may look like this:

```json
{
  "file.path": "/data/fmeserverdata/resources/data/Example.txt",
  "file.event": "CREATE",
  "source": "dirwatch",
  "time": "2019-05-06T18:21:22Z",
  "event.id": "f754c6b3-c75c-4e95-a49e-0a1db41decfc"
}
```

Notice how as well as including the updated file path it includes other information about the Automation event.

When converted into FME attributes using the JSONFlattener transformer the result – as shown in Visual Preview – will look something like this:
Now the content is available to the workspace as a set of attributes and can be processed as required.

**Using the Message**

What you do with the message depends on your required setup. If the topic is merely a trigger, and the message is unimportant, it could be ignored. However, in most cases, the message content is important.

For the most part, Automations should do the work for you. If you want to use JSON-formatted messages in an Automations workflow, you can use the Webhook trigger.

More complex parsing of messages can be done within workspaces, for example, using JSON transformers. One useful example to consider is where the message contains an X/Y coordinate - for example the location of a person. Here, the X/Y coordinate could be converted into a point feature with the VertexCreator transformer, and from there, any number of FME transformers could be used to carry out spatial processing such as a geofence.
As a technical analyst in the GIS department, you have realized the overhead associated with pushing manual updates to your corporate database. Having read up about automations in FME Server, you think that it should be possible to set up a system that automates this process.

So far you have set up a system for added file notifications to be registered by FME Server. Now you must create a workspace to process these and publish it to FME Server. The workspace must then be triggered by the Directory Watch in Automations.

You may have noticed that the Log Action in Automations actually submits a workspace to process this request. Therefore we have already - perhaps unknowingly - covered how to set up this response. Now it's time to create a new workspace to fit in with your overall goal: to provide real-time updates to your corporate database.

---

**FME Lizard says...**

This exercise continues where Exercise 1 left off. You must have completed Exercise 1 to carry out this exercise, or upload the Begin Server Project to FME Server to start where the previous exercise left off.

---

1) Create Workspace

Start FME Workbench and begin with an empty workspace.

Select Readers > Add Reader from the menu bar. When prompted set the parameters as follows:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>Esri Shapefile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\Engineering\BuildingFootprints\update001.shp</td>
</tr>
<tr>
<td>Workflow Options</td>
<td>Single Merged Feature Type</td>
</tr>
</tbody>
</table>

It doesn't matter what Shapefile we use as the source right now; setting the source dataset in this step is only to satisfy the shapefile reader requirements. At runtime, the source dataset will be replaced by the file path recorded in the Directory Watch message.

By setting the Workflow options to Single Merged Feature Type this means it will be possible to process any source dataset (of the right format) and have it translated.

2) Add Writer

Having read the data from a Shapefile, we can now add it to the corporate database.
Select Writers > Add Writer from the menubar. When prompted set the parameters as follows:

<table>
<thead>
<tr>
<th>Writer Format</th>
<th>SpatiaLite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer Dataset</td>
<td>C:\FMEData2020\Data\Engineering\BuildingFootprints\building_footprints.sl3</td>
</tr>
<tr>
<td>Writer Parameters</td>
<td>Overwrite Existing Database: No</td>
</tr>
<tr>
<td>Add Feature Types</td>
<td>Table Definition: Manual</td>
</tr>
</tbody>
</table>

In the new feature type that is created, change the Table Name parameter to `building_footprints`:

![Feature Type Image]

Ensure that the Table Handling is set to "Create If Needed". Click OK to close the dialog and then connect the new feature type to the output port of the Shapefile Reader.

3) **Inspect Data** After adding the writer, click on the `building_footprints` feature type to bring up the popup menu. Then click the Inspect button to open the dataset in Visual Preview. There is already data in the `building_footprints.sl3` dataset, but we should take note of what the data looks like so we will know where it has changed once we update the dataset with the new data. The area within the red box will be where the new data will be added:
4) **Publish Workspace**

Save the workspace as RealTime-Ex2-Complete.fmw and publish it to the Training Repository in FME Server. We only need it to be run (not do anything special) so register it with the Job Submitter service only.

5) **Add Dataset to FME Server**

Since the purpose of this Automation is to update our database – let’s make sure that it is accessible in FME Server. To do this, we will upload the `building_footprints.sl3` SpatialLite database to FME Server’s shared resources.

Use the FME Server web interface to create a new folder **Output** in **Resources > Data** and upload the file located at `C:\FMEData2020\Data\Engineering\BuildingFootprints\building_footprints.sl3`
6) Edit Automation

Navigate to the Automations: Manage Automations page and select Incoming Building Footprints to open the Automation for editing. Before you can make any changes stop the Automation using the button in the top right corner. Instead of adding a new action node, simply select the Log node and change the Action parameter value to Run a workspace.

Select the Training Repository and workspace uploaded in the previous step. The parameters should now include one for the Source Esri Shapefile and the output database.

The source dataset needs to pick up the file path from the Directory Watch trigger. From the drop-down menu select file path found under the Directory folder. This drop down list is essentially the JSON flattened into its separate components.

For the output database browse the Resource/Data/Output folder to locate the SpatiaLite database uploaded in the previous step:
7) Add Filter
The Esri Shapefile Reader will only accept .shp file extension types, however the Directory Watch will pass a message to the workspace for every file uploaded. To prevent the Automation triggering database update workspaces that will fail add a Filter action so that only the file path containing .shp is parsed to the Run Workspace action.

Select the plus icon at the bottom of the canvas, drag an internal action (orange) onto the canvas.
Move the connection lines so that the Directory Watch enters this new Action node and the Run Workspace is connected to the Success Output port of this action.

Click on the action to configure the filter, set the Action to Filter messages. There are two parameter values required. Similar to how the Source dataset of the workspace was set, specify the File Path as the Key. In Contains, set the string to search for to .shp.

Click Apply to save the changes and restart the automation.

**FME Lizard says...**

*Instead of using a filter action we could have zipped up the update002.shp/.dbf/.shx/.prj files so that the Directory Watch was only triggered once. Much like the Log Action notice the Filter submits a FilterMessage.fmw workspace for each incoming message, therefore if you are handling a high volume of incoming data zipping files may be the preferred option.*

7) **Test Solution**

Now test the solution by putting update001, update002 or update003.shp (and other extensions) into the BuildingUpdates folder. If
these files already exist from an earlier exercise, delete them first, then re-add them. You will find that each dataset put into the folder is added to the SpatiaLite database.

Check the Completed Jobs page to confirm that the workspace was run.

7) Inspect the Output
In the FME Data Inspector, add a new dataset, and browse to the C:\ProgramData\Safe Software\FMEServer\resources\data\Output\tfolder and add the building_footprints.sl3 dataset. Depending on which update file you added, you should see one of the three buildings added to the dataset:

CONGRATULATIONS
By completing this exercise you have learned how to:
- Identify JSON elements from an incoming Trigger message
- Configure the Automation to run a workspace in response to a Trigger using part of this message
- Chain actions by passing an element of the incoming JSON through a filter
An Introduction to Email in Automations

It’s important to cover email notifications in some detail because they are one of the most commonly used protocol types on FME Server.

“Email” is a protocol that may be used by both Trigger and Action components in an Automations workflow. Email Triggers receive incoming emails from external clients, and Email Actions send emails out to an external account.

Email Protocols

In fact, email is not a single protocol. There are several methods of email transfer that exist. FME Server supports two of these email-related protocols: SMTP and IMAP.

SMTP (Simple Mail Transfer Protocol) is the ability to directly receive an email through an email server built into FME Server, or indirectly send an email through an SMTP service, for example, a Gmail account.

IMAP (Internet Message Access Protocol) is an indirect process that connects to an email account on a server elsewhere. When that account receives an email, the IMAP protocol passes it on to FME Server.
SMTP Trigger

SMTP Triggers are used when data is sent to FME Server via a direct email. FME Server receives an email and triggers an Action in response.

Such triggers are possible because FME Server includes a built-in email server as one of its components. However, this does require that the hostname and domain of your FME Server should resolve to a publicly accessible IP or DNS.

FME Lizard says …

The steps to set up the built-in email server for notifications are documented in the FME Server Reference Manual.

However, FME Cloud instances are automatically configured for email notifications, and have a public domain name too, so you don’t need to do any additional setup.

Creating an SMTP Trigger

Creating an SMTP Trigger is done in the Automations: Build Automation section of the FME Server web interface, by choosing the Email (SMTP) protocol as a new Trigger.

Once the protocol type is selected there, is only one parameter that must be configured, that is the Email User Name. This does not need to be an existing user or email; in the above example, it is set up to be AuthoringCourse. Note that you do not need to specify @FMEServer-Hostname.com in the Email User Name parameter.
To construct the full email address, combine the Email Username with your FME Server hostname. Now, for this example, whenever an email is sent to AuthoringCourse@YourFMEServerHostname this Automation will receive an incoming message that contains the entire contents of that email, including the from and to address, and any attachments. This information is all processed by FME Server so it can be used to trigger an Action downstream in the workspace.

In the SMTP trigger, there is an Optional Parameter to specify the location to download any email attachments to. If this is not defined, the attachments will still be saved to a temporary location on FME Server.
**IMAP Triggers**

IMAP (Internet Message Access Protocol) is a variation on email for incoming notifications.

Instead of using the built-in email server, an IMAP Trigger connects to another email server and monitors it for incoming email. When a new email arrives in that account, the Automation is triggered.

---

**Creating an IMAP Trigger**

Like the SMTP protocol, IMAP Triggers are set up in the Automations: Build Automation section of the web interface by creating a new or adding to an existing Automation, but this time choosing the Email (IMAP) protocol:
This Trigger has significantly more parameters, most are for defining the IMAP (Email) server connection.

There are two important parameters that let you decide what and when to watch this Email Server. The Poll interval determines how often FME Server should check for emails and second, the Emails to fetch decides whether to fetch all unread emails or new emails only.

Like the SMTP Protocol there is a parameter to select an FME Server resource location in which to store any email attachments.

---

**FME Lizard says …**

*Most email servers support IMAP functionality, as do the majority of cloud-based email providers such as Gmail, Outlook.com, Yahoo!, etc; so it’s very easy to have FME Server scan a Gmail account (for example) for incoming mail, and then act on its contents. FME Server has the option to load templates for these popular email providers.*
Email Actions

Email Actions are designed to have FME Server send an email in response to a Trigger or Internal Action. The built-in email server in FME Server is only for incoming mail, as is the IMAP protocol, so messages need to be sent via an existing (external) SMTP email server.

Setting up an Email Action

Creating an Email Action is done on the Automations page of the web interface, and must connect to an existing Trigger. You can either create a new Automation and first set up a Trigger, or view existing Automations through the Manage page, where you can then extend these workflows to include an email notification.

There are many more parameters for outgoing mail because the full SMTP server connection parameters need to be defined. However, there is an option to load a template for some of the most common email services.
Email Details

Action: Send an email

Parameters

Load Template

SMTP Server

SMTP Server Port: 25

SMTP Account (optional): someone@example.com

SMTP Password (optional): ...

Connection Security: SSL/TLS

Email To

Email Cc (optional)

Email Bcc (optional)

Email From

Email Subject

Email Format: Text

Email Attachment (optional): ...

Email Body (optional):
Various fields for the email itself (From, To, Subject, Template) can be hard-coded or passed through to the Action from the previous connection in the Automation, whether that be a Trigger or Internal Action. One important parameter is the Email Format, which can either be plain text or HTML.

See the following sections for information on how to generate content for outgoing emails.
As a technical analyst in the GIS department, you were involved in a recent assignment to set up a Directory Watch solution for users to automatically update the corporate database.

Having learned that not all users are able to access the internal network where FME Server is hosted, you think that it should be possible to also set up a system that uses email-based automation to handle the same updates.

FME Lizard says...

*This exercise continues where Exercise 2 left off. You must have completed the previous chapter Exercises to carry out this exercise, or upload the Begin Server Project to FME Server to start where Exercise 2 left off.*

1) Create Resource Folder
The first step is to create another Resource folder where all the email attachments will be saved. Log into the FME Server web interface, navigate to Resources > Data > BuildingUpdates, and then create a new folder called Emails.

2) Update Automation
Next, Navigate to the Automations page, since we are triggering the same workspace we can expand on the existing Automation created in the previous exercises. Add a new Trigger by dragging and dropping a Trigger node (green) onto the canvas. Before configuring the Email Trigger add a connection from its output port to the Workspace Action.
The new Trigger can be created to use either the Email (SMTP) protocol or the Email (IMAP) protocol.

SMTP is easier to set up, but FME Server must reside on a server with a proper DNS record (all FME Cloud and Training machines will have this). IMAP is necessary when FME Server resides on an internal network or you would like to monitor an email account sitting on an external server.

**SMTP Protocol**

To use the SMTP protocol select Email (SMTP) as the Trigger. This will reveal the Email User Name parameter. Enter a name for receiving email, for example: *fmeshapeprocessing*

For the Download Attachments To parameter browse to the Emails folder created in step 1.
Clicking Apply will create an email address `fmeshapeprocessing@<hostname>` - for example:

<table>
<thead>
<tr>
<th>Host</th>
<th>Example Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FME Cloud</td>
<td><a href="mailto:fmeshapeprocessing@myfmeserver.fmecloud.com">fmeshapeprocessing@myfmeserver.fmecloud.com</a></td>
</tr>
<tr>
<td>Amazon AWS</td>
<td><a href="mailto:fmeshapeprocessing@ec1-23-456-789-012.compute-1.amazonaws.com">fmeshapeprocessing@ec1-23-456-789-012.compute-1.amazonaws.com</a></td>
</tr>
<tr>
<td>localhost</td>
<td>fmeshapeprocessing@IP Address</td>
</tr>
</tbody>
</table>

To quickly find your IP Address, navigate to Google in your browser and search "My IP". The search results will return your public IP address as the first result.

**IMAP Protocol**

To use the IMAP protocol select Email (IMAP) as the Trigger. This will open up a number of other parameters. Enter them according to your email account.

In case it is of use, the server information for Gmail, Outlook, and Yahoo! are as follows:

<table>
<thead>
<tr>
<th>IMAP Server Host</th>
<th>imap.gmail.com</th>
<th>imap-mail.outlook.com</th>
<th>imap.mail.yahoo.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Port</td>
<td>993</td>
<td>993</td>
<td>993</td>
</tr>
<tr>
<td>Connection Security</td>
<td>SSL/TLS</td>
<td>SSL/TLS</td>
<td>SSL/TLS</td>
</tr>
<tr>
<td>Verify SSL Certificates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

You will also need to check the settings in your email account to make sure IMAP is turned on. Regardless of the email provider, you should set these parameters as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poll Interval</td>
<td>30 Seconds</td>
</tr>
<tr>
<td>Emails to Fetch</td>
<td>New Emails Only</td>
</tr>
<tr>
<td>Download Attachments To</td>
<td>Data &gt; BuildingUpdates &gt; Emails</td>
</tr>
</tbody>
</table>

You could select any Resource folder for attachments to be saved to; but for this exercise series we have created a separate Emails folder to ensure we don’t choose the BuildingUpdates folder as this would cause the workspace to be triggered by both the Email and Directory Watch Trigger!
The Automation is not quite ready yet because the JSON message from the Email Trigger stores the incoming file path in an attribute named Email Attachment, however the Source dataset for the run workspace is set to pick up a value from the Directory Watch File Path attribute. Therefore in order for the workspace to be able to process the file path from both Triggers we need to create an attribute in each Trigger with a common name that can be used downstream in the workflow.

3) Create Custom Key
We will need to create an Output Key on both Triggers. First select the Email trigger and click on the Output Keys tab. Event Keys list all the standard output keys that come with that action, the second, custom section called User Keys.

To add a new User Key click on the plus icon. Set the key name to user.shapefile and for the Value select Email > Email Attachment in the Text Editor window.

Repeat this process in the Directory Watch trigger but this time set the Value to Directory > File Path.

Lastly return to the Workspace Action and change the Source Dataset from file.path to the user.shapefile attribute listed under the User Keys dropdown menu. Now, the workspace will be able to parse the Shapefile path to the workspace no matter what the trigger is.
User keys can only be used in the Automation they are defined in, and can be set to an element of the JSON from the Trigger or Action message. To create values that can be used across all workspaces you can create Global Keys by selecting the globe icon in the menu ribbon. These can only be set to plain text values, but an example of their use might be a value your organization commonly refers to, such as a UNC path to an external location where all data is stored.

4) Test Automation

Now let's test the Automation.

This time, because we connected the Email Trigger directly to the Workspace Action, let's zip up the four .shp, and associated file types, to add as an attachment.
Send an email with an attachment to the address created in the Email Trigger. When the email is received by FME Server (SMTP), or FME Server fetches it (IMAP), the Automation will send a message to the Workspace Action. (Remember that an IMAP publication only checks for an email at the specified poll interval, so the result might not be immediate!)

CONGRATULATIONS

By completing this exercise you have learned how to:
- Create an Email Trigger
- Create an automation with multiple input triggers to a single action
- Create custom User Keys to configure a single action to handle multiple trigger types
Workspaces as a 'Trigger'

Workspaces and other Internal Actions are the only protocols that have both input and output ports. This means that although they are treated as an Action, since they need something to kick them off, they can also act as a trigger to subsequent actions downstream an Automation.

![Diagram of Workspaces as a 'Trigger']

Notifications from the Workspace are important for both sending messages containing content from the data being processed, and for reporting the status of a translation, such as whether it ran successfully or ended in failure.

Workspace Action Output Ports

By default, workspace nodes have two outports representing the workspace status. One for Success and the other for Failure, this allows you to build an action in response to one or both job statuses. Each output includes a set of Output Keys with additional information about the workspace that was run that can be used within Actions connected to either port.

Automations Writer can be used to add additional output ports to a Workspace Action simply by adding the writer into the workspace to be run. This will allow for setting up Actions to be triggered based on what happens within the workspace. The Automations Writer also allows for passing custom messages from within the workspace to Actions connected to any of the output ports.

Workspaces and FME Server Topics

Alternatively, Workspaces can be set up to trigger FME Server Topics to be used in an Automation. This is handy if the workspace is run from a process other than a listed protocol e.g. via an FME Server App, however you still want there to be an external response to the workspace. A workspace can be registered to notify a topic in one of two ways:

- Registering with the topic when it is published
- Using a transformer inside the workspace

![Diagram of Workspaces and FME Server Topics]
A Topic is a component on FME Server that acts as a mediator for messages and defines the message content. Think of it as a mix of the subject line for notifications and a trigger for them to occur. You can create a Topic trigger in the FME Server web user interface through Automations.

A Workspace can be linked to multiple topics, so each incoming message can trigger multiple actions to occur. Additionally, multiple workspaces can link to a single topic.
**Workspace Action Output Ports**

As mentioned in the previous section, all Workspace Actions will have two output ports by default (one for success and another for failed jobs). Each of these ports will have a set of Output Keys on them. These Output Keys allow for passing some basic information about the job that was run to any Actions connected downstream from that port.

For example, a user may want set up a workflow in which an email containing the job log file is sent to an administrator when a job fails. They could do this by connecting an Email Action to the failure port of the Workspace Action and use the `job.log` Output Key in the attachment field within the email configuration to pass in the path to the log file.

But what if you need to pass more detailed information to an Action within your Automation? For example: a summary of the number of features that were written out or a list of files to be processed by other Workspace Actions within an Automation.

**Using the Automations Writer**

This is where the Automations Writer is useful. The Automations writer can be added to any workspace to create additional output ports when the workspace is used in an Automation. The Automations writer can be added to a workspace in the same way any other writer is added. One output port will be created for each feature type added to the Workspace Canvas and all attributes on each feature type will be included as Output Keys within the Automation.

For example, this writer has two feature types (Output_One and Output_Two) with different sets of attributes on each. The image on the left shows how this looks within Workbench and the right shows the same Workspace used within an Automation:
The Automations Writer allows for lots of flexibility in what information gets passed out of a workspace. It can be used to create as many output ports as needed to allow for advanced filtering within the Automation to trigger specific Actions in response to anything that may happen within a Workspace. Multiple features can also be passed into any Automations Writer feature type to allow for triggering an Action multiple times.
Workspace Transformer as a Trigger

Instead of having a workspace trigger additional actions on job completion, it could instead send a notification through a transformer called the **FMEServerNotifier**.

In the example below, a workspace author is sending an FME Server notification when a feature fails the conditions of a Tester transformer:

![Diagram](image)

Notification properties are set in the transformer parameters. The parameters include those for connecting to FME Server, one for the topic to post to, and one for the message to be included.

Here the author intends to publish information to a topic called RoadInfo:

![Parameters](image)

There are two advantages to issuing a notification this way, over using the registration method:

- The workspace can issue a notification *during* a translation, rather than at the end of it.
- The workspace does not need to be run on FME Server to generate an FME Server notification. It will produce the same notification when run using FME Desktop.

The disadvantage is that you won't know whether the workspace completed successfully - or not - when the notification is issued.
Workspaces as an Action AND a Trigger

We've now looked at how to set up a workspace to be an Action (reacting to incoming messages) and how to set up a workspace as a Trigger (sending outgoing messages). A workspace can do each task individually, but when it is set as both an Action and a Trigger, the overall setup looks like this:

Unlike most Triggers/Actions in Automations the Run Workspace node has both an incoming and outgoing port meaning it can both receive messages from a Trigger, and continue the Automation by sending messages out to another action upon job completion.

In this scenario, the same workspace that receives an incoming notification also sends an outgoing notification. For example, details of a lightning strike are received via a LightningStrike topic. This trigger starts a workspace that processes the incoming information; for example, it determines which state/province/county the strike occurred in. The workspace then creates a new message and dispatches it to a WeatherAlert Action.
After configuring an Automation in FME Server to process building footprint updates with both the Directory Watch and Email Triggers, your supervisor is wondering if they can receive an email whenever the corporate database is updated. Using an external email server, you think that it is possible to configure the existing Automation in FME Server to satisfy this requirement. You also realize that you can use an Automations Writer in the workspace to include additional information about the updates made to the corporate database in the email that is sent.

FME Lizard says...

This exercise continues where Exercise 3 left off. You must have completed Exercise 3 to carry out this exercise.
Access to an SMTP Email Server is required for sending email in this exercise. Gmail, Outlook, and Yahoo! are examples of acceptable web-based solutions if you do not have access to an internal email server.

1) Open Workspace in Workbench
If you don't already have the workspace from Exercise 2 open in Workbench, download it from FME Server using the File > Download from FME Server option in Workbench. From the download from FME Server dialog, choose the Training Repository and select RealTime-Ex2-Complete.fmw to open it in Workbench for editing.

2) Create Summary Information for Job
Now, you are going to update this workspace to generate a summary of the number of records that were updated from the incoming Shapefile. Add an Aggregator transformer and connect it in parallel to the Shapefile Reader like this:
Open the Aggregator parameters dialog and set the Count Attribute to a name: NumberOfUpdates

3) Add Automations Writer
Go to Writers > Add Writer and select the FME Server Automations format. Set the Feature Type Definition to Automatic... and add the writer to the canvas. When prompted, set the Feature Type Name to JobSummary and click OK.

Connect the Automations Writer feature type to the output port of the Aggregator transformer.

4) Clean up Automations Writer Attributes
After connecting the Automations writer, you’ll notice that all the attributes from the Shapefile data are included on the writer. We don’t need all that information within the Automation. Open the Feature Type Properties on the writer, then go to User Attributes and select Manual under Attribute Definition.

Remove all the attributes except for the one we created earlier, NumberOfUpdates. Set the Type for NumberOfUpdates to int as this is a numeric value.

Click OK to apply that change. At this stage, there should be two attributes on the JobSummary Feature Type: NumberOfUpdates and fme_feature_type

5) Save and Republish Workspace
Save the workspace and Republish it to FME Server.

6) Update Workspace Action
Open the FME Server web interface and navigate to the Automations:Manage page. Click the Incoming Building Footprints and
stop the Automation to allow for editing. Click on the Workspace Action in the Automation to open its parameters. Click on the Refresh button in the top right of the Parameters window then click Apply to update the Workspace Action. A new output port should appear in the Automation.

7) Add External Action
The final step in the Automation is to add an email service External Action through which a response will be sent.

Select the plus icon in the bottom left and drag an External Action (blue) to the canvas. Connect this to the JobSummary output port of the Run Workspace node, which will now also act as a Trigger.

Double click on the node to configure it, set the action to send an email and set up your SMTP email server parameters.

In case it is of use, the server information for Gmail is as follows:

<table>
<thead>
<tr>
<th>SMTP Server Host</th>
<th>smtp.gmail.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Port</td>
<td>465</td>
</tr>
<tr>
<td>Connection Security</td>
<td>SSL/TLS</td>
</tr>
</tbody>
</table>
There is also the option to load a template for some other Email Servers, if you are unsure what port/host they use. Regardless of the email provider, you should set these parameters as follows:

<table>
<thead>
<tr>
<th>Email To</th>
<th>(An email you have access to check)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email From</td>
<td>Your account name (for example <a href="mailto:fmeshapeprocessing@myfmeserver.fmecloud.com">fmeshapeprocessing@myfmeserver.fmecloud.com</a>)</td>
</tr>
<tr>
<td>Email Subject</td>
<td>Building Footprints Database Updated</td>
</tr>
</tbody>
</table>

Next, set the Email Body. Click the drop-down arrow to the right of the value and select Text Editor. This will open a window that will allow you to add content that can also include values coming from elements within the Automation.

Set the Email Body to the following:

The Building Footprints database has been updated! {route.JobSummary.NumberOfUpdates} records were updated from a file called {route.JobSummary.fme_feature_type}

Note: To add the information coming from the Automations Writer, look at the panel on the left-hand side of the Text Editor and look under Workspace > JobSummary for the list of attributes available to use. Click on each one to add them to the text editor.

---

**FME Lizard says...**

*For the Email Body parameter click the drop-down arrow and then select Text Editor. This will open a pop-up window that allows you to write email content that also contains information from elements in the Automation. This might be useful for sending out Notifications to an Administrator on Job failure, where you can include the additional details about the Job, such as ID and Status Message in the email body, and the job log as an attachment.*

---

Before you Apply these parameters validate the Email Server configuration using the Validate button, if FME Server is unable to connect to the email server you can troubleshoot this before finding out after your automation is running.
Enable the Automation by selecting the Start Automation button.

**NOTE**

Depending on your Gmail security settings, you may need to create an app-specific password to allow FME Server to log into the account. See this article if you are noticing errors connecting to your account: [IMAP Publication or Email Subscription is not Reading Emails from Gmail](#)

8) Test Automation
Test the Automation by either sending an email with a zip file attachment of the Shapefile datasets (.dbf, .prj, .shp, .shx) from C:\FMEData2020\Data\Engineering\BuildingFootprints, or by adding this collection of files to the Directory Watch as in Exercise 1.

If the workflow was successful, you should receive an email back with a response!

**Building Footprints Database Updated**

fmeserver@gmail.com 4:23 PM (15 minutes ago)

The Building Footprints database has been updated!

CONGRATULATIONS

By completing this exercise you have learned how to:

- Add an Automations Writer to a workspace
- Set up an outgoing Email Action
- Trigger an Email notification through Automations
Authoring Job Chains

Workflow Management is a technique for controlling workspaces in sequence or branching with inbuilt logic. Part of this technique is being able to author workspaces that are "chained together" to run one after another.

What are Job Chains?

A chain of jobs is one that runs jobs in a particular sequence; either one after the other or in parallel. You can use Automations to implement this on FME Server.

Job Chains and Automations

Automations provide a way to handle workflow management within FME Server. To do this, you can simply create an Automation to handle triggering jobs in a particular sequence.

Setting up a job chain within an Automation is as simple as connecting two or more Run Workspace Actions to a Trigger. Depending on the desired behavior, you can connect the run workspace actions together in sequence or in parallel.

Run Jobs in Sequence

When actions are connected in sequence, they are connected together one after the other in a string like this:

When connected like this, these jobs will run sequentially. So, the first job will be submitted and FME Server will wait for it to run to completion. When it finishes, the next job will run to completion before the next action in the chain starts.

In this example, since the second workspace is connected to the Success port of the action for the first workspace, it will only run if the first job completes successfully. You can add additional actions connected to the failed ports if you wanted to handle those failures in some way. For example, if you wanted to send an email out if one of the jobs in this Automation failed to complete.

Run Jobs in Parallel

When actions are connected in parallel, they are connected to a single starting point (this can be a Trigger or another Action within the Automation) instead of in a row:
When connected like this, both these jobs will be submitted at the same time.

If there are at least two engines available on FME Server, then they will both run at the same time. If there is only one available engine, one job will run on that engine and the next job will wait in the queue until an engine becomes available. There is no guarantee as to which of these jobs will run first in this case (unless Job Queues have been configured for these workspaces to give them a priority).

**Conditional Processing**

There are two Action types that are particularly useful if you need to build some additional logic into your Automations. These are the Filter and Merge Actions.

The Filter can be used to check if a simple condition is true or false. In the example below, we are using a Filter to check if an email contained a .dgn file as an attachment. If that is true, we will run one workspace and if it's false, we will run a different workspace.
The Merge Action can be used to merge two parallel streams together. This is useful if you want to trigger several jobs to run at the same time (in parallel), and then wait for all of them to complete before moving onto the next action:

**TIP**

The Filter Action in Automations currently only supports very simple filters. If you need to configure more advanced filtering and the Filter Action isn't enough, you can add a Workspace Action instead and use that to check for more advanced conditions and use the Automations Writer to filter the data to different output ports.
You're a technical analyst in the GIS department of your local city. You have plenty of experience using FME Desktop, and your department has recently begun using FME Server.

A municipal election is about to happen, and Elections Interopolis have provided a dataset of new voting divisions in GML format. You've already created a workspace to translate these voting divisions to a SpatiaLite database format for use within the city and that data is being written to a resources folder on FME Server so everyone can use it.

Coincidentally, the planning department heard of this update. They have a workspace that assigns voting division IDs to each of the records in the city's address database for use in election planning and would like to have their workspace run automatically whenever there are any updates to the voting divisions.

You realize that you can chain these two translations together to be carried out consecutively and automatically whenever the Election Voting GML file is updated using an Automation.

1) Import Project
In the FME Server web interface, go to Projects > Manage Projects and import the Start Project for this Exercise from C:\FMEData2020\Projects\ServerAuthoring\RealTime-Ex5-Begin.fsproject

This will contain the two workspaces we want to run in sequence as well as input and output Resource folders for this workflow. Take a moment to familiarize yourself with the contents of this project and look at the two included workspaces using the View Workspace page.

2) Create Automation
In the FME Server interface, navigate to the Automations > Build page to start building a new Automation.

Save the Automation first and give it a name such as JobChaining.

3) Add Trigger
You want these jobs to start running whenever new data is added to the Election/Input folder, so set up a Trigger to handle that.

Click on the Trigger that is already placed in your Automation to configure it. Set the Trigger as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger</td>
<td>Directory Modified</td>
</tr>
<tr>
<td>Directory to Watch</td>
<td>Data &gt; Election &gt; Input</td>
</tr>
<tr>
<td>Events to Watch for</td>
<td>MODIFY</td>
</tr>
<tr>
<td>Poll Interval</td>
<td>30 Seconds</td>
</tr>
</tbody>
</table>
4) Add Workspace Action

Now we want to add an Action to run the first of the two workspaces we want to run in our job chain.

Click on the + button and drag an Action onto the canvas. Connect it to the Directory Watch Trigger and configure it as follows:

<table>
<thead>
<tr>
<th>Action</th>
<th>Run Workspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>Training</td>
</tr>
<tr>
<td>Workspace</td>
<td>RealTime-Ex5-WorkspaceA.fmw</td>
</tr>
</tbody>
</table>
5) Add Second Workspace Action

Note that a Workspace Action will always wait until the job has completed before moving onto the next Action within the Automation.

To set up the second workspace to run, simply add a second Action and connect it to the checkmark (or Action Succeeded) port on the Workspace Action we just added. Configure this one to run the RealTime-Ex5-WorkspaceB.fmw from inside the Training repository.

Your final Automation should look like this:

![Diagram of workspace chain]

6) Save and Start Automation

Save your Automation. Then click the Start Automation button to start the Automation.

7) Test the Automation

Now you can the Automation by uploading a modified version of the ElectionVoting.gml file to Resources.

Open the Resources page and navigate to Data > Election > Input. Click on Upload > Files and select the following file to upload:

C:\FMEData2020\Resources\ServerAuthoring\JobChaining\ElectionVoting.gml

Wait a minute or so and then View Triggered Jobs for your Automation, you should see that both chained workspaces successfully ran in order.

CONGRATULATIONS

By completing this exercise you have learned how to:
• Create a Job Chain within an FME Server Automation to run jobs in sequence
The Notification Service

Prior to FME Server 2019 incoming and outgoing message handling was built using the Notification Service. Although the overall concept is the same the way you put together these notifications was more challenging and the service was made up of three main components spread across multiple pages in the FME Server Web UI:

- **Publications**: Event handlers that listen for incoming notifications
- **Subscriptions**: Event handlers that dispatch outgoing notifications
- **Topics**: Subjects that describe what a notification is about

In Automations Triggers are equivalent to Publications and both External and Internal Actions are grouped as one under Subscriptions. In the Notification Service all messages had to pass through a Topic, this is no longer required and the different protocols can communicate directly with one another, however Topics are still one of the available protocols.

Whether you are new to FME Server or just running through this training course as a refresher, moving forward we recommend using Automations to build any real-time workflows. If you would like more information on the Notification Service please refer to the 2018 Server Authoring Course which will run through the same series of exercises we have just completed in the Notification Service.
Troubleshooting for Administrators

This section shows a few basic troubleshooting techniques in case of an emergency.

Directory Watch Trigger is not started by Adding Files

If you are having troubles triggering an Automation that contains the Directory Watch trigger when new files are added, then the following may be of help:

- Ensure that the account running the FME Server Windows Services has permissions to access the directory.
- CREATE and MODIFY notifications are only sent when there is a change in file size - this might be an issue if you are overwriting the file, or deleting and re-adding within the set Poll Interval.

FME Server Fails to Receive Email

If you are unable to receive email on FME Server then the following suggestions may be of help:

- Ensure you carry out the post-installation configuration steps as noted in the reference manual. Without this only local mail could be delivered.
- Is the FME Server’s email SMTP port open? Typically this will be port 25 or 465.
- Does your FME Server have a DNS name that is available on an internet name server or your local DNS setup? Email cannot be delivered to it if it cannot be found!
- For IMAP, is the FME Server’s IMAP port open? Typically this will be port 993.

FME Server Fails to Send Email

If you are unable to send email on FME Server then the following suggestions may be of help:

- Have you created and correctly configured the Email Action?
- Is the FME Server’s email SMTP port open? Typically this will be port 25 or 465.
- Does the FME Server have an internet connection?
- Are you using a Google account with 2-Step Verification? You will need to generate an app password and replace the SMTP password with this 16 digit app password code.

Email Uses Defaults

If an email sent by FME Server uses default parameters instead of those you had configured, then the following suggestions may be of help:

- If you are sending an email using the FMEServerNotifier transformer ensure that the Content parameter is set to an attribute that contains the email message information in JSON format (use the custom transformer FMEServerEmailGenerator).
- If you are sending an email from a completed workspace using the Notification Service and Topics to Notify (Success) ensure the Notification Writer is set to the Text File writer which is writing the email information.

Cannot Connect to WebSockets Server
If you cannot connect to a WebSockets server, then the following may be of help:

- Ensure the FME Server’s WebSockets port (default 7078) is open.
- Ensure you are using the correct stream_id for sending and receiving between your applications.
Module Review

This module introduced you to FME Server’s Automations.

What You Should Have Learned from this Module

The following are key points to be learned from this module.

Theory

- Real-time systems are message-driven and involve small amounts of information.
- Automations handle individual messages.
- Topics are keywords that define a notification subject.
- Triggers are FME Server objects that listen for and respond to notifications from clients
- Actions are FME Server objects that push notifications to FME Server or external clients
- Incoming Notification messages are received from triggers, and outgoing messages are sent to actions
- Protocols are the method of communication (for example, email, FTP, Amazon S3, etc.)
- Workspaces can be run in a series of translations using a technique called chaining within Automations

FME Skills

- The ability to create an Automation, with Triggers, Internal and External Actions
- The ability to use the Directory Watch protocol as a trigger
- The ability to use the Email protocol as a trigger and an action
- The ability to use workspaces to process incoming notification messages
- The ability to chain jobs using Automations

Further Reading

For further reading why not check out...

- This article on Getting Started with Automations which includes a series of example automations.
Questions

All notification setups must have which of these:

Triggers AND Actions Triggers OR Actions Or both Triggers OR Actions, but never both None of the above
In FME Server Automations it is necessary for all of the components to be used in a setup. This is because the system is designed to allow FME Server to react to incoming Notifications, if you designed an Automation that only contained an incoming Trigger the Automation could not be started because the incoming message is not being passed forward to anything. Likewise if you only had an Action in the Automation, this can never send a notification if it does not receive an event to kick it off. For this reason all components are required to Start an Automation.

Tell me, which one of these statements is correct:

SMTP and IMAP can both be used as either a Trigger and/or a Action protocol SMTP can be used as both a Trigger and a Action; IMAP can only be used for a Trigger SMTP can only be used for a Action; IMAP can only be used for a Trigger See the table under Automation Protocols for the full list of which protocols and whether they can be a trigger or action

I want my workspace to send me an email when it is run, so I know when people are using it to download data. When I publish it, what should I register it to?

The Notification Service The Data Download Service The Email (SMTP) Protocol The Workspace Subscriber Protocol

I'm setting it up for people to download data, so I register it as a Data Download service. It's as simple as that. To get a notification I just have to pick a topic to trigger in the Data Download settings. Of course, to get an email I must set up an Email Action connected to that topic - but that has nothing at all to do with how I register the workspace!

I've got a workspace that reads 50,000 features, transforms them, and writes them out. If I want to send a single notification that the features have been read, which combination of transformers would be of most use?

Creator/FeatureWriter/FMEServerNotifier Creator/FMEServerJobSubmitter Creator/FeatureReader/FMEServerNotifier FeatureHolder/Sampler/FMEServerNotifier I'm sending a notification (not running a job) so I use the FMEServerNotifier. I already have a reader so I don't need a transformer to read or write data. However I do need a Sampler transformer to reduce the number of features down to one; otherwise I'll send 50,000 notifications. The FeatureHolder ensures the notification is not triggered until all features have been read. It would look like this:

Which of the Receiver transformers has a parameter to stop it running continuously? Select all that apply.

SQSConnector WebSocketReceiver JMSReceiver TCPIPReceiver

The SQSConnector has the ability to switch to a number of messages to read, and the TCPIPReceiver has the option to close the connection once the publishing client disconnects.
Message Streaming

Message streaming is a real-time technique like Automations. However, where Automations receive/send one-off messages, message streaming involves a continuous flow of information. This example from the FME Server Playground shows a (simulated) live feed of buses, airplanes, and boats in the San Francisco Bay area:

Instead of a workflow being run once for each message, handling a message stream involves creating a workspace that will connect to the stream and run continuously receiving messages as soon as they are sent. Because of this reduced overhead, this technique can process data at a much faster rate than Automations. For our purposes, “continuous” means that messages arrive at the FME Server at a faster rate than Automations could handle; say more than one message per second.

Why Use a Message Stream?

When a workspace is involved as an action for processing messages, Automations operates by starting and running the workspace on demand, in response to an incoming trigger.

However, problems occur when the average message interval is less than the time taken for the Automation to start and run a workspace. A message interval of one second or less is the threshold at which the Automations service starts to struggle.

In a message stream, the workspace used for message processing is constantly running and doesn’t need to be started each time. Because of this reduced overhead, it can process data at a much faster rate than Automations.
When used in this way, we call it high capacity message streaming, as thousands of messages can be processed every second.
Elements of a Message Stream

Like Notifications, Message Streams can come into or out of FME:

However, rather than using Automations with workspace readers and writers, message streams are handled by transformers. A "receiver" transformer acts as an action, listening to a message stream and responding when a message is received. A "sender" transformer acts as a trigger, creating a message and sending it directly to an open message stream.

Transformers and Protocols

A number of transformers can handle message streams, each of which is tied to a specific protocol.

<table>
<thead>
<tr>
<th>Transformer</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>KafkaConnector</td>
<td>Apache Kafka</td>
</tr>
<tr>
<td>AzureQueueStorageConnector</td>
<td>Azure Storage Service Queues</td>
</tr>
<tr>
<td>JMSReceiver</td>
<td>JMSSender</td>
</tr>
<tr>
<td>KinesisReceiver</td>
<td>KinesisSender</td>
</tr>
<tr>
<td>SNSSender</td>
<td>Amazon Simple Notification Service (SNS)</td>
</tr>
<tr>
<td>SQSConnector</td>
<td>Amazon Simple Queue Service (SQS)</td>
</tr>
<tr>
<td>TCPIPReceiver</td>
<td>TCPIPSender</td>
</tr>
<tr>
<td>TweetStreamer</td>
<td>Twitter</td>
</tr>
<tr>
<td>WebSocketReceiver</td>
<td>WebSocketSender</td>
</tr>
<tr>
<td>PythonCreator</td>
<td>PythonCaller</td>
</tr>
</tbody>
</table>

Receiver Transformers

All receiver transformers in this list are designed to listen continuously to a message source and emit features only when a message arrives. Even then, the transformers will go on listening and awaiting more messages. Therefore, a workspace containing any of these transformers will run continuously and not need to be stopped or started for each message.

Sender Transformers

All sender transformers in this list will emit a message for each feature that enters. They don't, by themselves, keep a workspace running continuously and will shut down when incoming data is exhausted. However they will keep their connection open if a receiver is still running; therefore a continuous flow of outgoing messages requires both a receiver and sender.

Connector Transformers
Connector transformers have an Action parameter that allows for setting them up to function as a sender or a receiver and a Mode parameter to allow them to stream data as it is received.

**Python Transformers**

The PythonCreator and PythonCaller aren't specifically designed to connect to particular message protocols, but they can be made to do so and are particularly useful for protocols not otherwise supported in FME.
Message Streaming Architecture

A message streaming architecture that both receives and sends messages looks like this:

- A stream of messages is read into the workspace via one of the available transformers, for example, the JMSReceiver
- Each message is processed by any of the available FME transformers, according to the needs of the project
- A stream of messages is sent out of the workspace via one of the available transformers, for example, the TCPIPSender

Although the diagram shows a continuous process, it is not necessary for all of these components to be used in a setup. If the system is required to only receive messages, then only a Receiver transformer is needed. Likewise, if the system is intended to only send messages, then only a Sender transformer is required.

If both receiving and sending messages is required, then all components are necessary. However, it's still possible to split those actions up across several workspaces.

Databases

The Database components in this diagram are optional but are very useful. Messages will usually need to be processed against some other datasets (for example overlaid against a geofence), and a database is the quickest solution for reading and writing data.

Data read from a database is intended to be used to process the incoming message. For example, perhaps the message represents a point feature (maybe a vehicle location) that is used to filter against database data (maybe traffic conditions).

Data written to the database is usually to record a stream of message information. For example, perhaps each incoming message represents a point feature (a lightning strike) that needs to be written to a database for a historical record.
As a technical analyst in the GIS department, you deal with spatial data. Sometimes you need to process that data in real-time, and sometimes that data can arrive in significant quantities and at high speed.

In one such case, the city has been given access to real-time information about emergency calls.

*By emergency calls we mean the equivalent of 911 calls in North America, 999 in the UK, 112 in most of Europe, and 000 in Australia.*

Of course, these calls can arrive at a tremendous rate and at unknown intervals. If the city wishes to respond to any of these, and even if they wish to just record a history of the calls, you must implement a message streaming setup in FME Server.

1) Open Workspace

We don't have access to a real-time stream of emergency phone calls for this training course, so we will have to generate our own.

Open the workspace C:\FMEData2020\Workspaces\ServerAuthoring\DataStream-Ex1-Begin.fmw
Notice that the workspace generates a stream of events. A random number of events are generated, at random times, and at random locations. Additionally, random severity and event type attributes are generated.

Each event is wrapped up into a JSON format message. All that we need to do is push that message out as a stream.

FME Lizard says...

This workspace is just generating "events". Those events could be lightning strikes, vehicle locations, traffic accidents, or even UFO sightings! For this exercise, we'll pretend they are emergency phone calls. In real life you would be connecting to an existing stream of data and wouldn't need to generate one in this way.

2) Add WebSocketSender Transformer

Connect a WebSocketSender transformer to the output port of the JSONTemplater. Inspect the parameters and set them as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSocket Server URL</td>
<td>ws://localhost:7078</td>
</tr>
<tr>
<td>Verify SSL Certificates</td>
<td>No</td>
</tr>
<tr>
<td>Connection Preamble</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>ws_op: &quot;open&quot;,</td>
</tr>
<tr>
<td></td>
<td>ws_stream_id: &quot;EmergencyEvents&quot;</td>
</tr>
<tr>
<td>Data To Transmit</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>ws_op: &quot;send&quot;,</td>
</tr>
<tr>
<td></td>
<td>ws_msg: @Value(EventMessage)</td>
</tr>
</tbody>
</table>

As you can see, these parameters open a WebSocket connection (to an EmergencyEvents stream) and send information (the EventMessage attribute). Save the parameters and then save the workspace.

TIP

Note that the Data To Transmit parameter uses @Value() instead of fme_get_attribute(), which is what will appear if you use the FME Feature Attributes drop-down to construct the JSON. Make sure you use @Value().

3) Create Workspace

Now we have the ability to generate a stream of data we will create the workspace that is to process the data. Start FME Workbench and begin with a blank canvas (don't close the stream generator workspace, as we'll need that as well in a moment).

In the blank canvas add a Creator transformer and follow it with a WebSocketReceiver. Inspect the WebSocketReceiver transformer parameters and set them as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSocket Server URL</td>
<td>ws://localhost:7078</td>
</tr>
<tr>
<td>Verify SSL Certificates</td>
<td>No</td>
</tr>
</tbody>
</table>
| Connection Preamble | `{  
  ws_op: "open",
  ws_stream_id: "EmergencyEvents"
}` |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Attribute</td>
<td>IncomingMessage</td>
</tr>
</tbody>
</table>

Save the changes and add a Logger transformer after the WebSocketReceiver.

4) Publish Workspaces

Let's test what we have by publishing the workspaces and running them on FME Server.

Publish each workspace in turn. In both cases register it with the JobSubmitter service. There are no datasets or other parameters we need to worry about.

5) Run Workspace

Log in to the FME Server web interface, locate the data stream generator workspace, and run it. The dialog in response will look like this:

![DataStreamGenerate.fmw](image)

The workspace will run for a long time, and we can leave it to do so. Leave this page by clicking the Run Workspace button on the main menu and - within the Run Workspace page - locate the processing workspace. Now run that.

Again the response will report that the workspace is running, and will continue to do so.

6) Check Jobs and Cancel

Navigate to the Jobs page and click the tab labeled Running. You will see the two jobs:

<table>
<thead>
<tr>
<th>Completed</th>
<th>Queued</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Workspace</th>
<th>Repository</th>
<th>Username</th>
<th>Status</th>
<th>Started</th>
<th>Source Name</th>
<th>Source Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>DataStreamGenerate.fmw</td>
<td>Training</td>
<td>admin</td>
<td>Green</td>
<td>Today at 12:01:19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DataStreamProcess.fmw</td>
<td>Training</td>
<td>admin</td>
<td>Green</td>
<td>Today at 12:02:31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let the jobs run for a minute or two. Then choose each of them and click the Cancel button to cancel them:

Once canceled, go to the Completed jobs tab. You'll see the two canceled jobs:

Click on the processing workspace job and check the log. You should see messages in the log like this:

These messages from the Logger prove that the WebSocketReceiver is acting as expected and receiving messages from the message stream.

FME Lizard says...

You've proved that you can create a workspace to process a message stream, which is the important part of this exercise. But if you have the time, let's see what improvements we can add to make the result more realistic.

7) Add JSONFlattener
The first thing to do with incoming messages is to extract information as attributes. Because the incoming data is JSON format, add a JSONFlattener transformer to the processing workspace, replacing the Logger after the WebSocketReceiver.
Inspect the JSONFlattener's parameters and set the attribute IncomingMessage as the JSON Document to process.

Under Attributes to Expose manually enter:

- EventID
- EventLocation.EventXCoord
- EventLocation.EventYCoord
- EventSeverity
- EventType

You will now have the information from the message available as a set of attributes in the workspace.

8) Add VertexCreator
Now add a VertexCreator transformer. Set it up to use the X/Y attributes to create a true point feature:
9) Add Reader
The public transportation team within the city has learned you are working with this emergency data. They wish to be alerted immediately if there is an emergency event within 200 meters of a transit station. Let's show them how easy it is to set this up.

Firstly we need the transit station data, so select Readers > Add Reader and add the following:

<table>
<thead>
<tr>
<th>Reader Format</th>
<th>Esri Geodatabase (File Geode Open API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader Dataset</td>
<td>C:\FMEData2020\Data\CommunityMapping\CommunityMap.gdb</td>
</tr>
</tbody>
</table>

When prompted (or in the parameters dialog) ensure that only the TransitStations table is selected.

10) Filter Data
Now let's filter the emergencies.

First, add a Bufferer transformer to the TransitStation feature type and buffer the features by 200 meters. Be sure to set the End Cap Style and Corner Style parameters to Round.

Secondly, add a PointOnAreaOverlayer to assess whether an emergency falls inside one of these buffers. The workspace will now look like this:
At the moment there is one big problem that stops this from working. The PointOnAreaOverlayer transformer is a Group-Based transformer, sometimes called a “blocker.” It will hold on to features until it has finished being fed them, before outputting any data. In our case we want to make it Feature-Based; i.e., it will process each message at once.

So, inspect the PointOnAreaOverlayer parameters and set Areas First to Yes:

![Parameters](image)

This tells the transformer that all area features (buffered stations) will be first to arrive; therefore any point features (message locations) can be processed immediately.

However, we have to ensure that the transit features will arrive first. Therefore inspect the transformer parameters for the Creator transformer and set Create at End to Yes:

![Creator Parameters](image)

Now, all being well, the transit features will arrive first at the PointOnAreaOverlayer transformer.

Finally, add a Tester transformer after the PointOnAreaOverlayer. Set up the test to check for overlaps > 0 (i.e. where the message location falls inside a transit station buffer). Connect some Logger transformers to the Tester output ports and name them Tester_Passed and Tester_Failed:

![Tester](image)

Note that, if there were other parameters (for example the transit team were only interested in Event Types 7, 8, 9, and 10) you could add them to this Tester as well.

11) Publish Workspaces

Now publish the two workspaces again (you may or may not have to upload the TransitStation Geodatabase along with the
workspace) and run them using the same process as before, but leave it for a few minutes longer, as it can take a while for one of the random events to fall inside a transit station buffer.

Once stopped, check the logs and you should see that messages falling within 200 meters of a transit station are logged under the Tester_Passed header (you can use Ctrl+F to search for them, but note that the log might go to multiple pages).

FME Lizard says...

If you want to adjust the settings to get a result quicker, then go ahead. For example, you might set the buffer size to 500 meters instead of 200, or you might reduce the interval time on the message generator. Feel free to make whatever parameter changes you like to test the setup. You could even bypass the Decelerator transformer (in the data-stream creation workspace) to see how fast FME can deal with the incoming messages! However, if you do that, be sure to start the processing workspace first, else the generator might finish by the time you do get the processor started!

12) Add Writer

The messages that are being received are not all being used by the transit team, but we should probably keep a record of them. So - back in the processing workspace in FME Workbench - select Writers > Add Writer from the menubar. Use the following parameters to add a database Writer:

<table>
<thead>
<tr>
<th>Writer Format</th>
<th>SpatiaLite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writer Dataset</td>
<td>C:\FMEData2020\Output\Training\EventMessages.sl3</td>
</tr>
<tr>
<td>Writer Parameters</td>
<td>Advanced : Features Per Transaction = 1</td>
</tr>
<tr>
<td>Add Feature Type(s)</td>
<td>Table Definition: Automatic</td>
</tr>
</tbody>
</table>

In the newly added feature type, change the name to events and close the dialog. Connect the feature type to the VertexCreator output port (i.e. we're recording all events, not just the filtered ones):
The attributes are added automatically but include a few we don't need. So open up the properties dialog again for the feature type and click the User Attributes tab. Change it from Automatic to Manual and delete the attributes:

- _creation_instance
- incomingmessage
- eventlocation_eventxcoord
- eventlocation_eventycoord

Notice that the attributes were automatically renamed (to lower case and removing disallowed characters) to match SpatiaLite requirements.

If you publish and run the workspace (you may need to set the SpatiaLite database output to be written to a Resources folder) now you should be able to see - while the workspace is still running - the results being added to the database. You can inspect the file in the FME Data Inspector to prove this.

13) Create Automation
The filtered messages are important to the transit team, but at the moment they are going nowhere. We should set up a way to inform them.
We could add another messaging transformer, such as the WebSocketSender, JMSSender, SQSConnector, or even a Tweeter. That would make the processing workspace a "pure" messaging workspace.

On the other hand, the outgoing messages are nothing like the same rate as the incoming messages. With the parameters as described in this exercise, there is only a transit message once every minute. So, we can create a "hybrid" solution by setting output messages to be sent via Automations.

Go to the FME Server web interface and navigate to the Automations: Build page.

Add an FME Server Topic Notified Trigger to the canvas and create a new Topic called Emergency Transit Messages:

Now add a Log action connected to that topic. There are various protocols we could realistically use for sending a message (email springs to mind) but for the purposes of this exercise use the "Log a message" action is appropriate. Set the Formatted Message using the drop-down menu to Event as JSON (under the Event sub-menu). Then start the Automation.

14) Add FMEServerNotifier Transformer
Back in the processing workspace in FME Workbench, remove any Logger transformers at the end of the workspace. Add an FMEServerNotifier transformer connected to the Tester:Passed port:
Inspect the transformer parameters and set it up to send a message to the EmergencyTransitMessages topic. Set the message content to be whatever you like. You could use the text editor dialog to create something out of the available attributes (it can be plain text, it doesn't have to be JSON or XML).

15) Publish and Run Workspaces

Re-publish and set the workspaces running again. Navigate to View Log File under the Menu drop-down in your Automation to find the results as recorded by the Log Action. An event should look something like this in the log:

```
{ "automation.id": "88893f75-5895-4855-a1bf-e1975d2719c5", "EventSeverity": "5.2", "EventType": "8", "EventLocation": "{"EventYCoord": "548573.687", "EventXCoord": "491851.971""}, "EventID": "33", "time": "2020-08-06T14:52:40-07:00", "source": "topic", "event.id": "b6e9df7b-424c-49b4-a02b-e169343ab6f8", "automation.name": "EmergencyTransitMessages" }
```

**CONGRATULATIONS**

By completing this exercise you have learned how to:

- Send and receive messages via WebSockets
- Publish and run message-streaming workspaces
- Cancel message-streaming workspaces and check their log files
- Extract attributes from JSON messages
- Use transformers to transform and filter a message according to its content
- Set up workspaces to handle group-based transformers in a real-time scenario
- Record incoming messages into a database
- Set up a hybrid system with message streaming and automations
Module Review

This module introduced you to FME Server’s Message Streaming capabilities.

What You Should Have Learned from this Module

The following are key points to be learned from this module.

Theory

- Message Streams are set up using transformers and involve a workspace that runs continuously receiving/sending messages

FME Skills

- The ability to create a workspace for handling a message stream

Further Reading

For further reading why not check out...

- This blog article on JMS message streaming at 125,000 messages per hour!
Questions

Writing to a database in a High Capacity Message Streaming setup requires that the transaction interval is set to what value?

<Not Set>  One (1)  Infinity (∞) Setting the transaction interval to one means that each message is committed as it arrives. Any other value (in this list at least) would probably mean the data is never committed until the workspace was manually closed.
Course Wrap-Up

Although your FME training is now at an end, there is a good supply of expert information available for future assistance.
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.
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 web site.

FME Manuals and Documentation

For FME Server documentation, click the Documentation item in the FME Server web interface menu:
For FME Desktop use the Help function in FME Workbench to access help and other documentation. Alternatively, look on our website under the Support section.
Community Information and Resources

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

The FME Community

The FME Community is 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.

Forums

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. Join the conversation to see how the community helps each other with their 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!

Safe Software Webinars

Learn new ideas for transforming your data via over a hundred upcoming and recorded webinars.
The Safe Software blog provides technical information about FME, articles about customers' use cases, and general thoughts on spatial data interoperability.
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 movies, including recordings of most training and tutorials.
Feedback and Certificates

The format of this training course undergoes regular changes prompted by comments and feedback from previous courses.

Course Feedback

FME Lizard says...

*There’s one final set of questions – and this time you’ll be telling me if the answers are correct or not!*

Safe Software greatly values feedback from training course attendees, and our feedback form is your chance to tell us what you really think about how well we’re meeting your training goals.

You can fill in the feedback form now, but you’ll also be reminded by email shortly after your course. Safe Software’s partners who carry out training may ask that you fill in a separate form, but you can also use the official Safe Software form if you wish.

Certificates

FME Lizard says...

*In order to prove you have taken this training course, a certificate will be emailed automatically to anyone who was logged on for the duration of Safe Software hosted courses.*
Thank You

Thank you for attending this FME training course.
All of us at Safe Software wish you good luck and lots of fun as you conjure up exciting new FME magic.
Thank You