FME® Server Training Manual

FME Server 2013 Edition

SAFE SOFTWARE

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

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

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

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

Much of the data used here originates from public domain data made available by the City of Austin and Travis County, Texas. The datasets can be found at:
ftp://ftp.ci.austin.tx.us/GIS-Data/Regional/coa_gis.html

Copyright

Revisions

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

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

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

www.safe.com

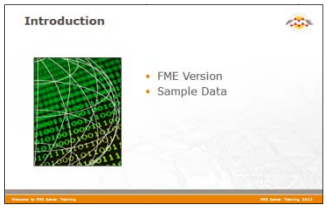
Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice.

Trademarks

Document Information

Document Name: FME Server Training Manual 2013
Version: FME Server 2013
Updated: November 2012-January 2013

Introduction



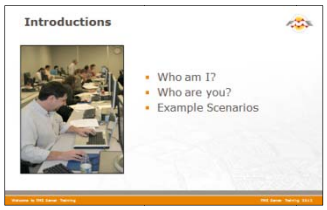
This document is the training manual for FME Server 2013

FME Version
This training material is designed specifically for use with FME2013. You may not have some of the functionality described if you use an older version of FME.

Sample Data
This sample data required for the examples and exercises in this document can be obtained from:

www.safe.com/fmedata

Introductions



Welcome to this FME Server training course.

Who Am I?
This is your instructor's chance to introduce her or himself.

Who Are You?
This is your chance to introduce yourself, and tell us about the following:

- Your organization
- Your planned use of FME Server

Exercise Scenarios
Throughout the training you won't be yourself!
You'll be a GIS professional in the emergency response team at the city of Interopolis.

The exercises build upon what you've learned in the content to reinforce your knowledge.

Each example and exercise has a single scenario: that there has been an earthquake and it's your job to manage and distribute the related spatial data. You're going to be busy!

Throughout this training you'll work closely with various departments, providing access to datasets they request to assist them in making decisions to deal with events linked to the earthquake.

Course Details

This is the workspace authoring training course for Safe Software's FME Server application.

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

It is comprised of the following sections:

1. Introduction to FME Server
2. Running Workspaces on FME Server
3. Passing Parameters
4. Download Services
5. Upload Services
6. Events and Notifications

The content may be spread over several days, or your instructor may choose particular sections to be covered.

Training Philosophy
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!

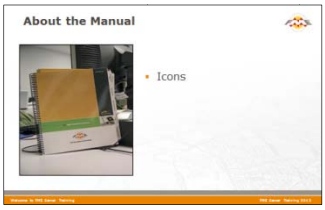
The training will introduce basic concepts and terminology, help you become an efficient FME Server user, and direct you to resources to help apply the product to your own needs.

Prerequisites
This training material is intended for persons with some prior experience of using FME Workbench. The content assumes a basic familiarity with the concepts and practices covered of FME Desktop; at 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

About the Manual



The FME Server training manual is yours to keep. It includes detailed material to help you remember your training.

During training we suggest you pay close attention to your instructor; they may cover the course content in a different order than the manuals, and will skip or add new content to better customize the course to your needs.

You'll find most of the instructor's PowerPoint slides reproduced in your manuals, as above. This will help you relate the manual content to the topic the instructor is discussing.

Icons
In the training manual you may see the following icons…

Tip: Additional advice to help apply the knowledge you have learned.

Caution: A warning where misuse of FME could lead to difficulties.

Example or Exercise: An example or exercise that the instructor will use to demonstrate a particular point or feature of FME. You may work through the example with the instructor or on your own.

Advanced Exercise: Extra challenges for students who are quick to finish and want to take an exercise a little bit further.

New: A feature new to, or significantly changed in, the most recent version of FME.

Q&A: Questions about the course content and how it is applied.

Important hints and tips appear in a box like this, to separate them from ordinary content.

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):

- Java Virtual Machine
- Apache Tomcat
- FME Desktop Version 2013
- FME Server Version 2013
- FLEXlm License Manager

A collection of sample datasets and workspaces are available in **C:\FMEData**

You may also find the sample data and workspaces on a disk or USB memory key included in your training pack.

Throughout the manual people from the city of Interopolis pop up to give you advice and to dispense FME-related wisdom.

Amenities
For in-person training, your instructor will explain the facilities and amenities available to you during the course.

Please respect other students' needs by keeping noise to a minimum when using a mobile phone or checking e-mail.

About Safe Software

Safe Software: Achieve total spatial data mastery!

History of Safe Software
Safe Software Inc. is the maker of FME® and the global leader in spatial data transformation technology that helps GIS professionals and organizations master their data interoperability challenges.

A common question is, “where did the names ‘Safe Software’ and ‘FME’ come from?”

Safe Software was founded way back in 1993 by Don Murray and Dale Lutz.

Back then, there was a format called the Spatial Archive and Interchange Format (SAIF). Since “SAIF” is pronounced “safe,” we decided to take the name Safe Software as a play on the SAIF format name.

FME officially stands for the Feature Manipulation Engine, and reflects its ability to transform data. By 2008, we had shortened it to “FME,” since by then most everyone knew what we were talking about, and to be honest, it was a mouthful.

Another common question is, “what happened to your hair?”, but that’s a story for another time!

Today, FME is used by thousands of customers in over 116 countries in a wide variety of industries. Our channel partner network has more than 100 value-added resellers across six continents.

The Safe Software Vision…

‘Our vision is to be the global leader in spatial data transformation, to knock down every last barrier to free and unrestricted spatial data use.’

The Safe Software Mission Statement…

‘Our mission is to help organizations use spatial data more easily and efficiently.’

Introduction to FME Server

What is FME Server?

FME Server is a powerful product for handling spatial data translations at the enterprise level.

FME Server Summarized
FME Server expands on the local processing tools of FME Desktop by providing a number of enterprise-level capabilities. In basic terms it runs workspaces and translates data in the same way as FME Desktop, but does so using a set of engines located on a server and accessed over a network such as the Internet.

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

- Start a workspace
- Structure a translation
- Deliver the output

The focus of this course is creating workspaces in a way that will take advantage of these capabilities.

Spatial Data Services

FME Server provides many spatial data services

Much of the FME Server networking capabilities are handled using what we call “Services”.

Basic Services
A number of basic services are provided with your FME Server installation and can be accessed interactively from the web interface.

Catalog Service	A service for determining what workspaces exist on a server, and what their parameters are
Job Submitter Service	A service for running a workspace and writing the output to a specified location
Data Download Service	A service for delivering the output of a translation as a zip file, via a download from a web page
Data Streaming Service	A service for delivering the output of a translation directly to an application
KML Network Link Service	A service for delivering the output of a translation directly into Google Earth
OGC Web Feature Service	A service for delivering the output of a translation as a WFS server
OGC Web Mapping Service	A service for delivering the output of a translation as a WMS server
Notification Service	A service for responding to notifications and issuing alerts to end-users

Additional Services
A number of additional services are also provided, to act as helpers in providing data or programmatic access to FME Server:

Data Upload Service	A service for handling uploads of source data
REST Service	A service for programmatic access via RESTful commands
Token Service	A service for creating and handling security tokens
Web Connection (SOAP) Service	A service for programmatic access via a SOAP interface

What Can I do with FME Server?

FME Server's flexibility allows it to be used in a wide number of scenarios

FME Server Uses
FME Server's services allow it to offer abilities such as the following:

- Web-based spatial data access: downloading and streaming
- Scalable data consolidation: loading and migration
- Online quality assurance: spatial data uploading and validation
- Server-based spatial data conversion: translation and transformation

FME Server Scenarios
Organizations may use FME Server to:

- Share and manage spatial data
  Sharing up-to-date data exactly where, when, and how people need it.

- Exchange data across a wide range of formats
  Scalable data loading and conversion in over 275 formats

- Transform data on-the-fly
  Automated and flexible transformation capabilities make collecting, processing, and delivering spatial data quick and efficient

- Improve efficiency and ROI
  A proven enterprise solution that helps to save time, improves process efficiency, and makes the impossible, possible.

With FME Server, organizations can address diverse spatial data requirements using a single enterprise solution.

FME Server Roles

Different roles have different interests in FME Server

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.

It's worth defining the terminology used for these roles, as it will be used quite often during this training course.

Author
An author is someone who creates translations (workspaces) 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 GIS analyst level, and would be an experienced FME user with a good understanding of published parameters and the source data formats being used.

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.

Because FME Server may be deployed in a number of ways, there are many types of end-user; the following are some examples:

- ***CAD Operator User***
  A CAD operator within your organization who wants to check out spatial data from your database warehouse for editing in CAD, and then upload the new drawing for Quality Assurance

- ***Corporate User***
  A business manager who wants to look at corporate infrastructure data in an application such as Google Earth or Virtual Earth

- ***Public User***
  Members of the public who want to download spatial data from your corporate website for personal use.

Developer

FME Server can be used as the back-end to 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 somewhat covered by these training materials.

Web developers may choose to include FME Server 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:

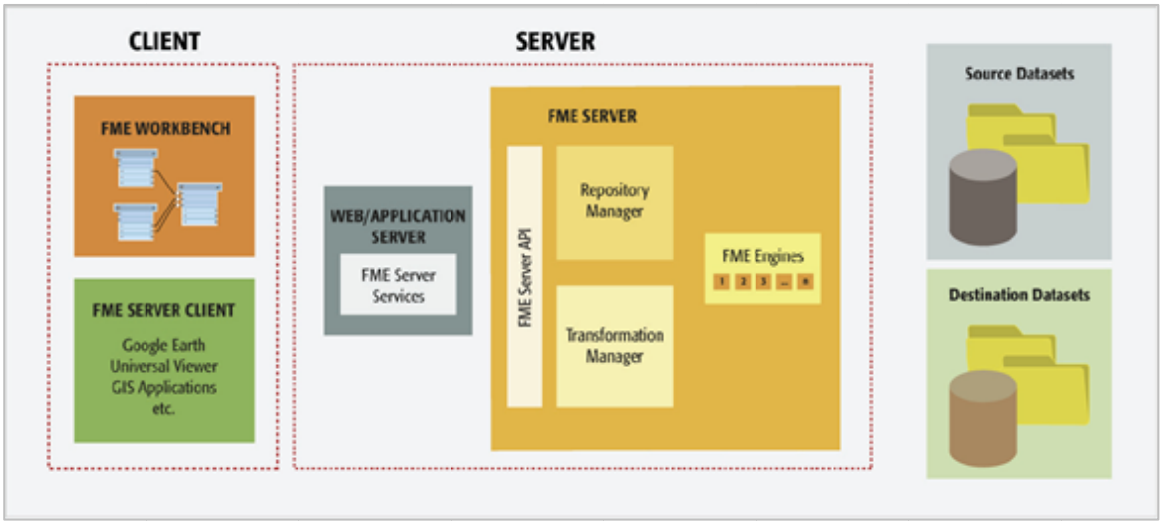
- Planning system architecture
- Installing prerequisite applications
- Installing and licensing FME Server
- Setting up FME Server services
- Monitoring FME Server services
- Troubleshooting
- Scaling FME Server

FME Workbench and FME Server

FME Workbench is the authoring environment for FME Server translations. It has functionality specifically for FME Server.

FME Server as a Model-Driven Architecture
FME Server is known as a Model Driven Architecture, because the specification for a translation is expressed as a model. 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 can be described as a *client-server pair*.



The Model-Driven Workflow
The four basic steps to creating and using a workspace on FME Server are:

- Author the workspace in FME Workbench.
- Publish the workspace from FME Workbench to FME Server.
- Run the workspace on FME Server.
- Maintain the workspace by downloading it from FME Server, making any required updates in FME Workbench, and re-publishing it back to FME Server.

Ms. Analyst says…

"Be aware that FME Workbench is part of the FME Desktop product and is not included as part of FME Server."

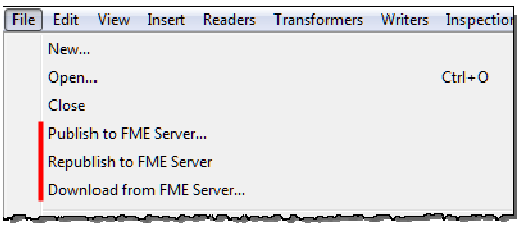
Workspace Management

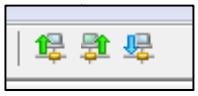
Because Workbench is a client of FME Server, workspaces may be transferred to and from an FME Server repository.

Transferring Workspaces
FME Workbench has the ability to:

- Publish a workspace to FME Server
- Republish a workspace to FME Server
- Download a workspace from FME Server

These three capabilities are accessed either through the menubar in FME Workbench…



…or the toolbar: 

Professor Lynn Guistic says…

"Published workspaces are held on FME Server in a file-based repository. Each FME Server may have multiple repositories, but any workspace can only belong to one of them.

All metadata related to the workspace is held separately in FME Server's repository database. This metadata includes information about source and destination datasets, workspace feature types, and published parameters."

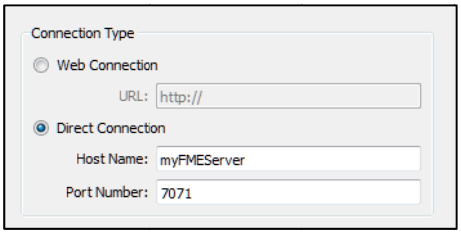
Publishing a Workspace

Once a workspace has been authored it is “published” to FME Server. A workspace is published using a simple wizard interface that connects using the FME Server API.

Connecting to Server

The first dialog in the publishing wizard is where a connection is defined.

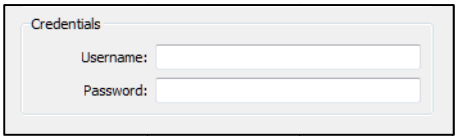
Workbench connects to FME Server using the FME Server API. It can connect within a local network (a Direct Connection) or through HTTP (a Web Connection).



A Direct Connection uses TCP/IP and requires the FME Server hostname and port number. The default port for TCP/IP is 7071. A Web Connection simply requires the URL of the FME Server.

HTTP is sometimes preferred in order to avoid blocking by a corporate firewall.

The dialog also provides fields in which to define connection credentials:



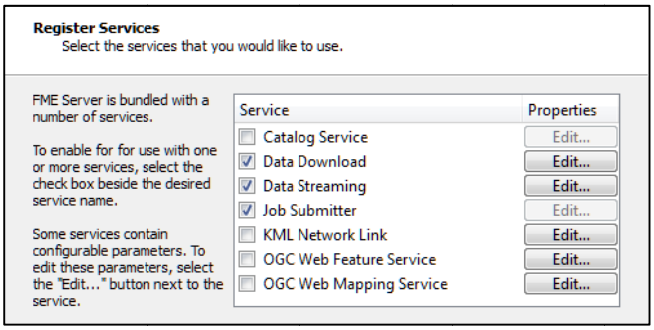
Repository Selection

The next dialog defines the repository for the workspace:

Either an existing repository can be used, or a new one created. The workspace name can also be edited, though it must always conclude with “.fmw”.

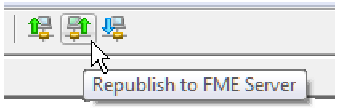
Workspace Registration

The final dialog defines which services the workspace is to be registered against. Job Submitter is a service that allows a workspace to be run on FME Server, with the output written to the location defined in the writer parameters.



Republishing a Workspace

Once a workspace has been published, the republish tool becomes active.
Further updates to the workspace (within the same session) can then be uploaded with a single click:

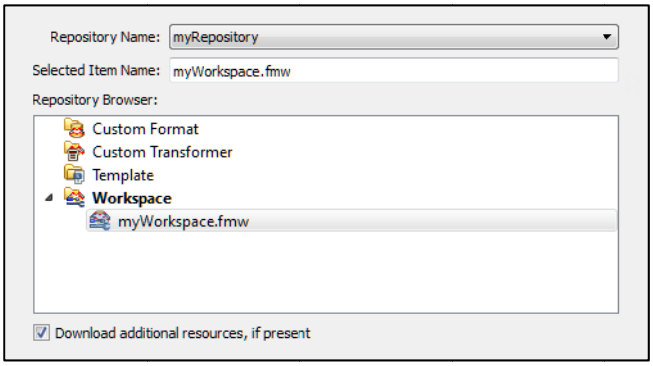


The same parameters are used as before. If you want to change any then you will have to go through the full publishing wizard.

Downloading a Workspace

Workbench can also “download” a workspace held in an FME Server repository, in order to make edits to it. 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 is *<user>/FME/My FME Server Workspaces*. The workspace – and resources – are then downloaded.

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

To avoid entering the same parameters every time a connection is made, set the parameters, click the Defaults button, and select 'Save as My Defaults'. Now these parameter values will be used automatically every time the connection dialog is opened.

Ms. Analyst says…

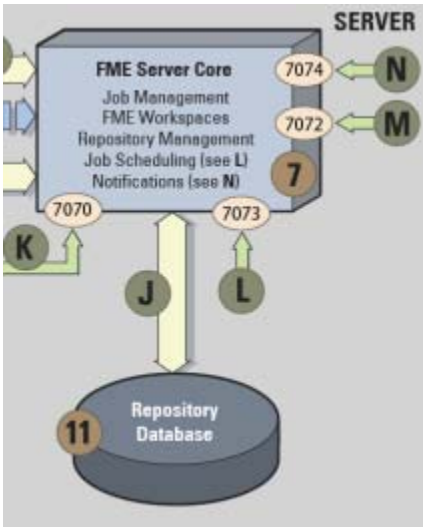
"Besides workspaces, it's also possible to publish/download FME custom transformers and custom formats to/from a server repository."

Repositories
Repositories are storage facilities for FME Server workspaces.

You can think of each repository as being like a folder on a file-system. Like a folder, it stores workspace files. Also like a folder, access rights can be granted to individuals and groups.

Although the workspaces are stored on the file-system (C:\apps\FMEServer\Server\repository) information about each workspace is stored in an internal database within FME Server. For interested experts, the FME Server core communicates with the database via JDBC over TCP/IP.

The contents of this database should **never** be edited directly. Querying the database is permissible, although extracting repository information is more commonly achieved using the REST API.



Example 1: Publishing a Workspace	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks
Overall Goal	Create a workspace, publish it to Server, and run it.
Demonstrates	FME Server workspace management
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example1Complete.fmw

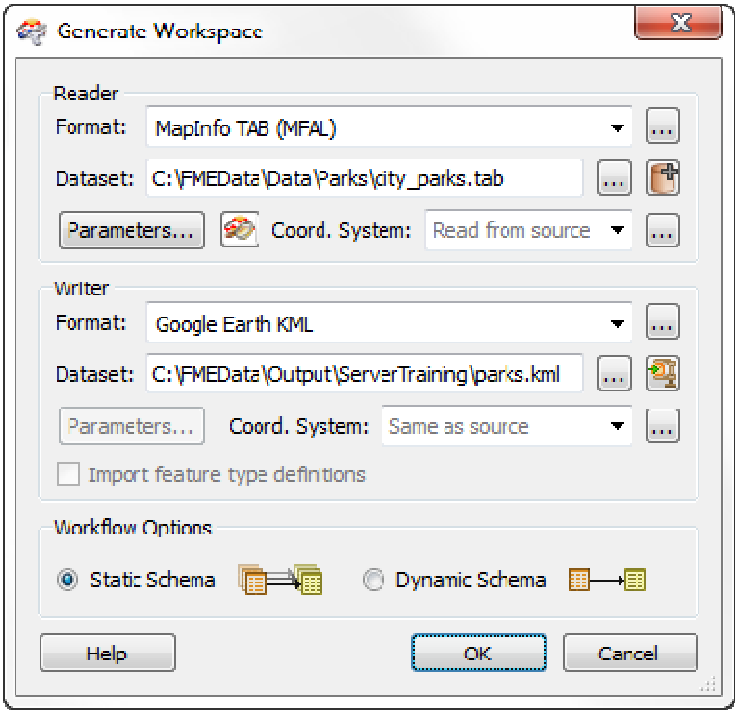
This exercise is just a simple one to refresh your memory on FME Workbench, and to get you started publishing workspaces.

1) Start Workbench
Start FME Workbench by choosing Start > All Programs > FME Desktop > FME Workbench

Select the Generate Workspace option in the start dialog (or press the shortcut, Ctrl+G).
Create a translation using the following parameters:

Reader Format MapInfo TAB (MFAL)
Reader Dataset *C:\FMEData\Data\Parks\city_parks.tab*

Writer Format Google Earth KML
Writer Dataset *C:\FMEData\Output\ServerTraining\parks.kml*



2) Save and Run Workspace
Save the workspace and then run it.

Check the output has been written (hint, right-click on the Writer feature type and choose Open Containing Folder) and then open it in Google Earth to ensure it looks correct.

3) Publish Workspace
Back in FME Workbench, choose File > Publish to Server from the menubar.

Publish to Server functionality is delivered through a wizard.

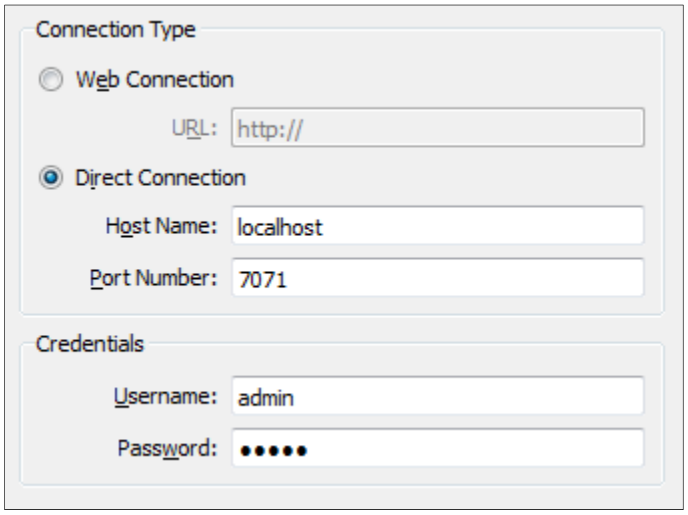
In the first dialog, enter connection parameters as described by your training instructor or system administrator. If you are unsure what to enter, and are using an FME Server installed on the same computer, common parameters will be:

Connection Type:
Direct Connection

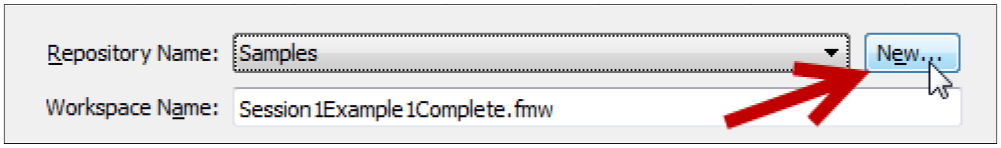
Host Name: localhost
Port Number: 7071

Credentials
Username: admin
Password: admin

Click **Next** to continue. If the credentials are correct you will move on to the next dialog in the wizard.



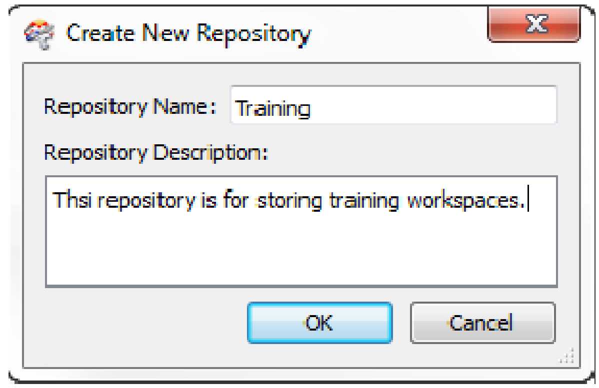
4) Publish Workspace - 2
Click the New button next to the Repository Name drop-down list. This will help us to create a new repository for training workspaces.



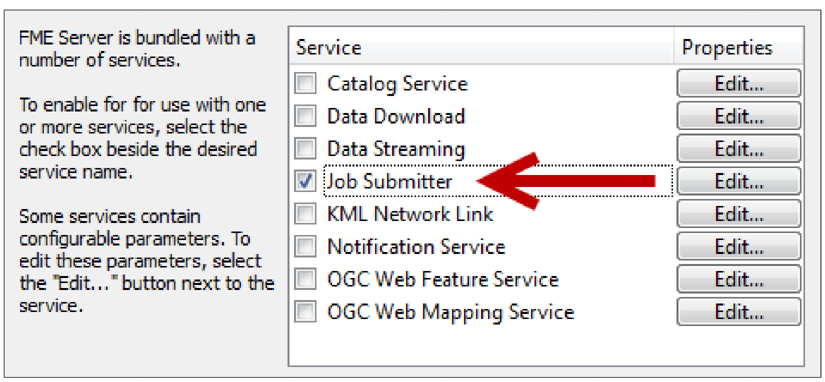
Enter Training as the name of the new repository and, optionally, enter a repository description.

Click **OK** to create the new repository, and then **OK** again to confirm acknowledgement of the response.

Now click **Next** to proceed to the final dialog.



5) Publish Workspace - 3
In the Register Services dialog, ensure that the Job Submitter service is checked. All other services can be left as they are, either checked or unchecked; the Job Submitter is all we are going to use here.



Click **Publish** to publish the workspace. After the workspace is transferred to Server, the log window will display a message reporting which workspace has been published to which repository and for which services.

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                              Publish Summary
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Server Host:            localhost
Server Port:            7071
Username:               admin
Repository:             Training
Target Name:            Session1Example1Complete.fmw
Resources Uploaded:     None
Services Registered:    Job Submitter
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Running a Workspace on FME Server

This section introduces the FME Server web interface

Once a workspace is uploaded to FME Server, it can be run using a server engine. There are many ways to start a workspace on FME Server, including:

- Simple job submission
- Running via URL
- Scheduled translations
- In response to events

For this section we'll look at simple job submission and running via a URL. To do this requires a workspace to be accessed via the FME Server web interface.

FME Server Web Interface
The FME Server web interface provides one access point to repositories and workspaces stored on FME Server. It is accessed via Start > All Programs > FME Server > User Web Interface, or by using the URL:

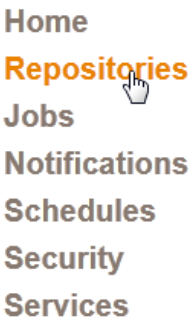
<hostname>/fmeserver/

For FME2013, the web interface has been completely overhauled and rewritten. Workspaces are now accessed via their repository, rather than the type of service. Additionally, there is now only one interface for both user and administrator access, whereas before there was a different interface for each.

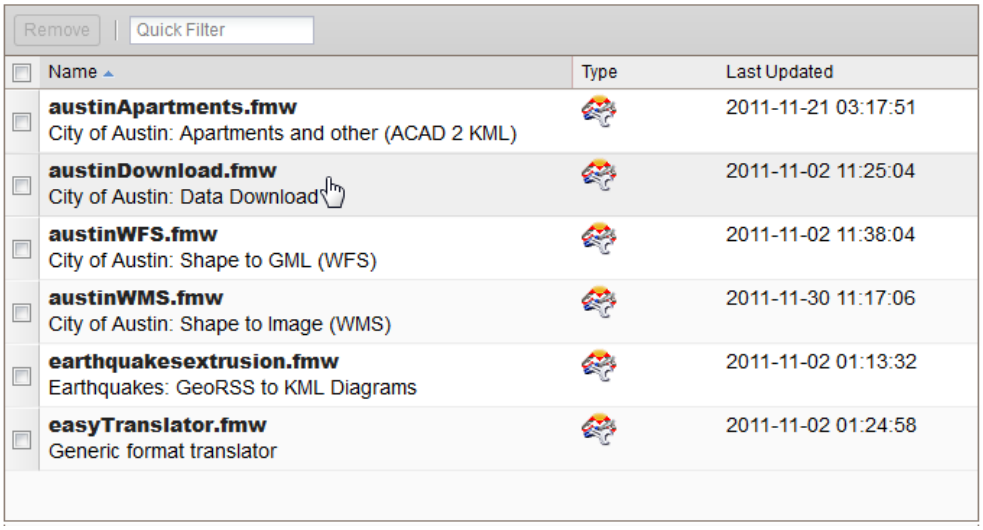
Job Submission
Job submission is when FME Server runs a workspace as-is. This is the closest to running a workspace on FME Desktop, as all inputs and outputs are as defined in the workspace (i.e. data is written out and not streamed or delivered in any other manner).

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

Jobs submission is carried out by navigating to the workspace to be run using the FME Server web interface. After accessing and logging in to the interface, the user must navigate to Repositories:



…then choose the repository in which the required workspace resides, and select that workspace.



A set of options are then listed (according to which services the workspace is registered against) and the user chooses Job Submitter > Run.

Running a Workspace via URL

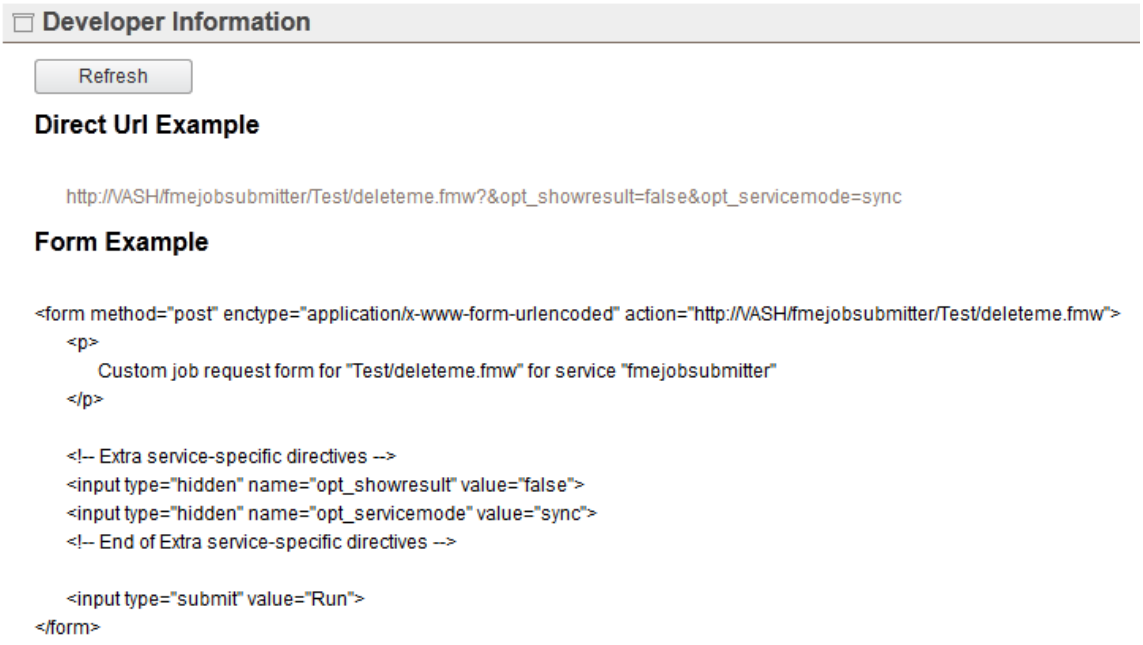
All job requests to an FME Server are a variation on an HTTP request. This makes running a workspace via a URL very simple, provided you know what form the request will take.

The easiest way to find that URL is again through the FME Server web interface.

This time a user would locate the Job Submitter options for their workspace, but click Configure instead:

The configure option opens up published parameters for editing (see the next session of the training course for more information) but also provides a section entitled Developer Information.

Developer Information includes both an example URL and example form. The URL method uses a HTTP GET request whereas the HTML form uses a HTTP POST request.



This information is a useful tool 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 application.

There are limits to the amount of data you can send in a GET request because URLs have length restrictions that vary depending on the browser being used. If you anticipate that your request parameters may include very long strings, use a POST request.

Example 2: Running a Workspace	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks
Overall Goal	Create a workspace, publish it to Server, and run it.
Demonstrates	FME Server workspace management
Starting Workspace	<uploaded to server>
Finished Workspace	n/a

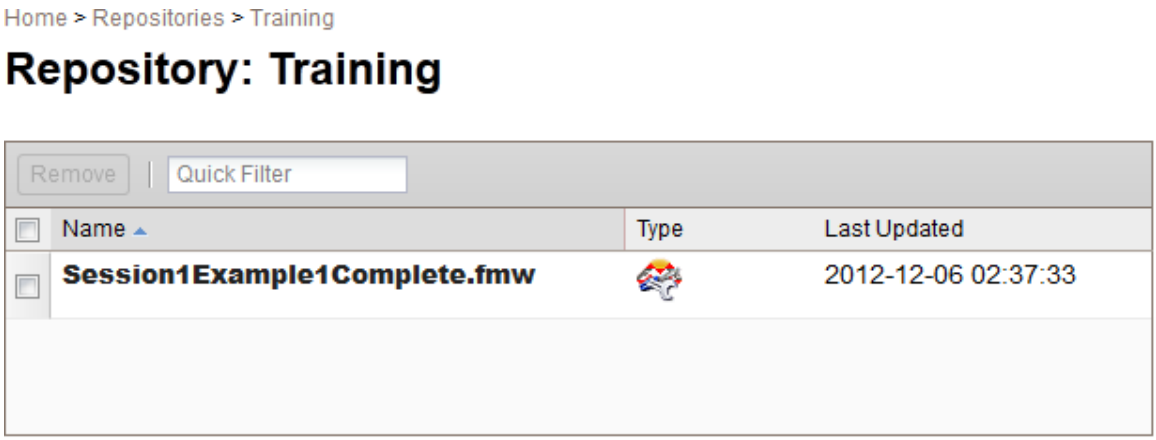
This exercise is takes the workspace you created in example 1 and runs it in a variety of different ways.

1) Open FME Server Web Interface
Open the web interface by choosing Start > All Programs > FME Server > Web User Interface

Log in to the interface using the credentials provided by your training instructor or system administrator. If you are unsure what to enter, try:

Username admin
Password admin

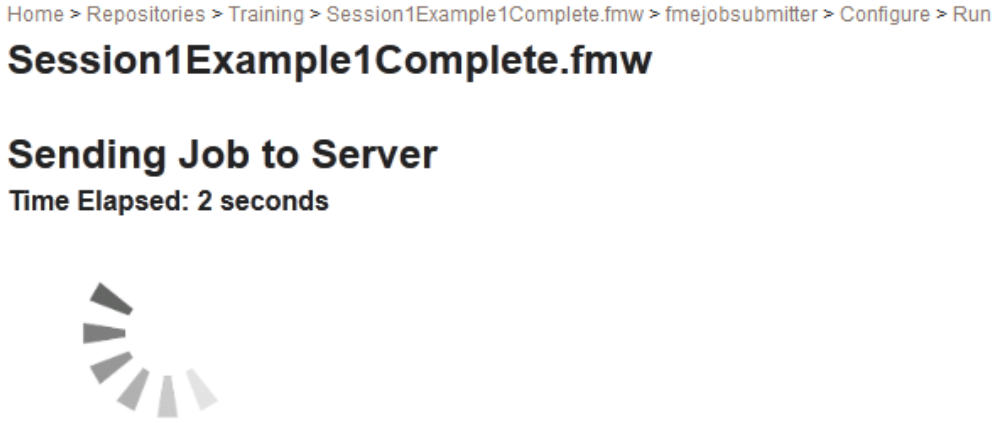
2) Navigate to Workspace
Click Repositories in the menu on the left-hand side of the interface. A list of repositories will become available. Click on the link for the Training repository. In here should be the workspace you published in Example 1.



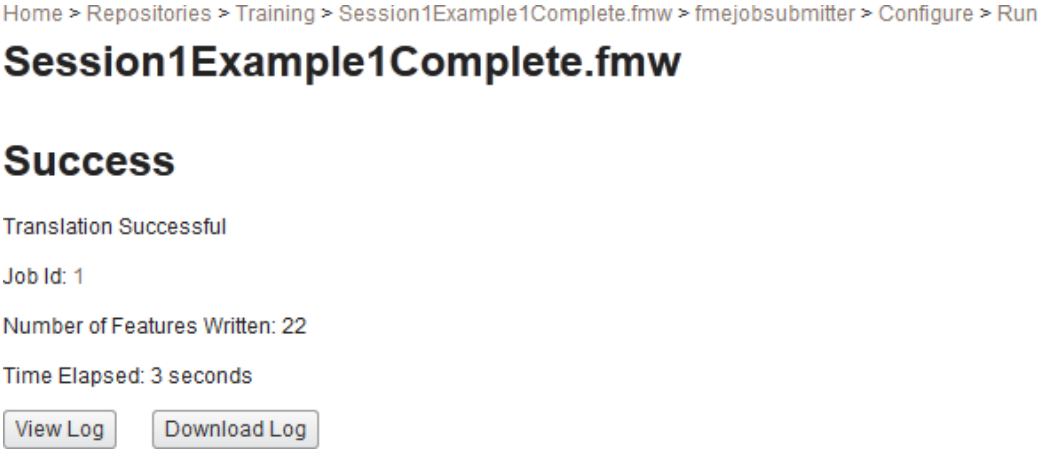
Click on the workspace to display the list of services against which it is registered.

3) Run Workspace
Run the workspace by clicking on the Run option under the Job Submitter service:

The workspace will start to run:



…and shortly afterwards will have been completed:



Click either of the two options to inspect the log file and ensure the translation was successful.

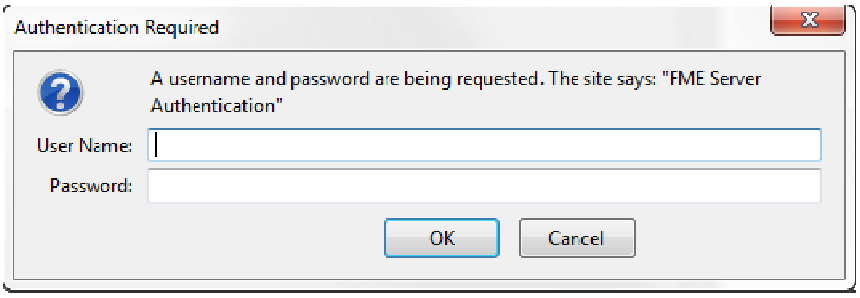
Browse the file system to locate the output data and ensure that the last-modified date/time of the file has been updated.

4) Get Developer Info
Back in the FME Server web interface, browse back to the list of services for the workspace. This time click Configure instead of Run.

In the subsequent dialog, click on the bar labelled “Developer Information” to expand that section of information:

Copy the URL listed under “Direct URL Example”.

5) Run Workspace
To run the workspace again, simply paste the copied URL into a web browser.
You will be prompted for your access credentials again, before the workspace will run:



Enter your username and password and then click **OK**. The workspace will now run again. Once more, you can confirm this by checking the last modified date/time of the output dataset.

Advanced Tasks
If you have time to continue this section, why not try these advanced tasks.

The first advanced task involves running the previous workspace using FME Workbench as the client (rather than a web browser).

The second advanced task involves creating your own web interface to run the workspace.

Task 1
Workspaces can also be triggered to run on FME Server using FME Workbench as the client. This is done using one of two transformers:

- FMEServerJobSubmitter
- FMEServerWorkspaceRunner

The likely use-case for this functionality is a workspace author who wishes to trigger a server job in response to a series of actions (or tests) that can be carried out locally. For example, the local workspace might test the date of a dataset to see if it has been updated recently; if so it will direct the FMEServerJobSubmitter to commence a workspace to operate on this data.

1) Start FME Workbench
Start FME Workbench and begin with an empty canvas.

You can close the existing workspace by:

- Using File > Close on the menubar
- Using the shortcut Ctrl+W
- Clicking the X character on the tab labelled Main

Place a Creator transformer by clicking on the canvas and typing “Creator”.

2) Place FME Server Transformer
Compare the FMEServerJobSubmitter and FMEServerWorkspaceRunner transformers. Ask your instructor if you cannot see the differences between how they work.

Choose one of these transformers and place it in the workspace, connected to the Creator transformer.

3) Set Parameters
Open the parameters for your chosen transformer. Again the parameters are defined by a wizard. Work through the wizard’s dialogs, filling in your credentials and selecting the workspace from the previous examples as the one to be run.

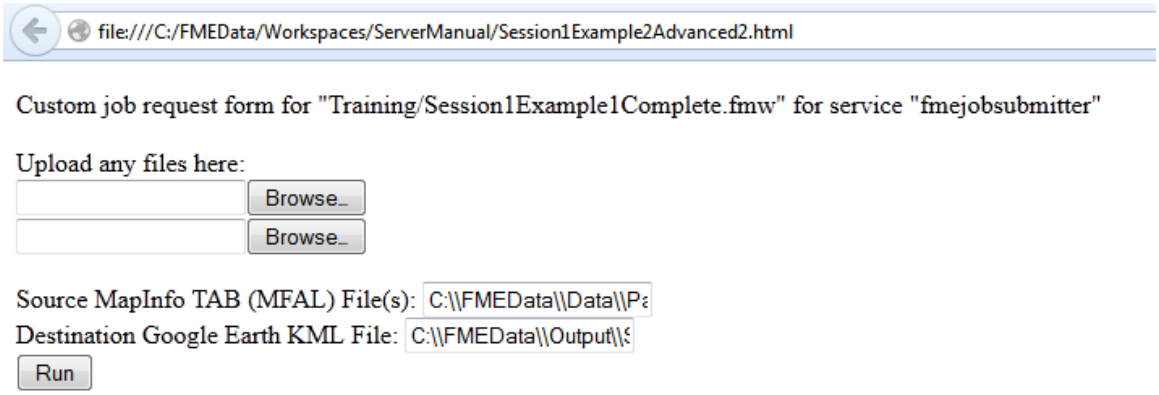
If there are parameters you do not understand, then just leave their default values.

4) Run the Workspace
Connect Logger transformers to the SUCCEEDED and FAILED output ports, so you can tell if the process worked correctly, and then run the translation.

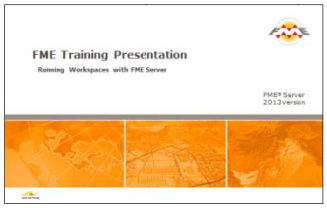
Again, check the date/time of the output dataset to prove that the workspace ran correctly.

5) Check Source Data
If you have the time, use the Directory and File Pathnames reader (instead of the Creator transformer) to read the date/time of the City_Parks dataset. Use a Tester to check if the file has todays date and, if so, run the workspace on FME Server.

Task 2
Use the developer information on the configuration page, and a HTML editor, to create a simple web page for users to run the workspace.



Module Review



This module introduced you to running workspaces on FME Server.

What You Should Have Learned from this Module
The following are key points to be learned from this module.

Theory

- FME Server is a Model Driven Architecture – where workspaces are the models
- FME workspaces are stored on FME Server in repositories
- The Job Submitter service is for running a workspace “as-is”

FME Skills

- Publish a workspace from FME Workbench to FME Server, creating a new repository if necessary
- Run a workspace on FME Server either through the web interface or via a URL
- Run a workspace on FME Server using a transformer in FME Workbench

Parameters

Parameters are used to control a workspace. They can be pre-defined by the author or set at run-time by the user.

FME Parameters

Every component of a translation – workspace, reader, writer, feature type, and transformer – has a set of parameters to control how it operates.

These are parameters that *directly* control a translation. There are parameters in the Navigator window of Workbench and – for FME Server in particular – there are other parameters to control Services or Jobs.

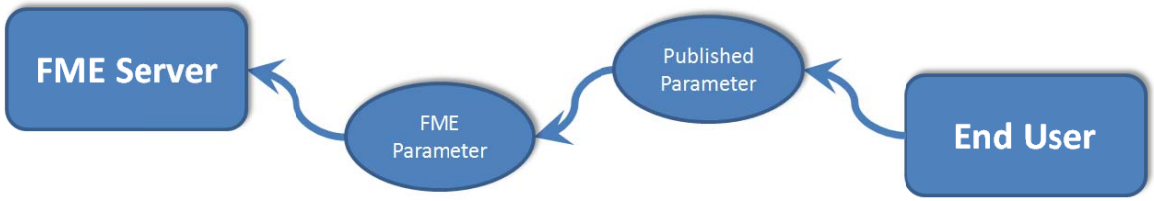
All of these could be called **FME Parameters**. They will usually be set by a workspace author for the express purpose of defining how the workspace will operate.

User Parameters

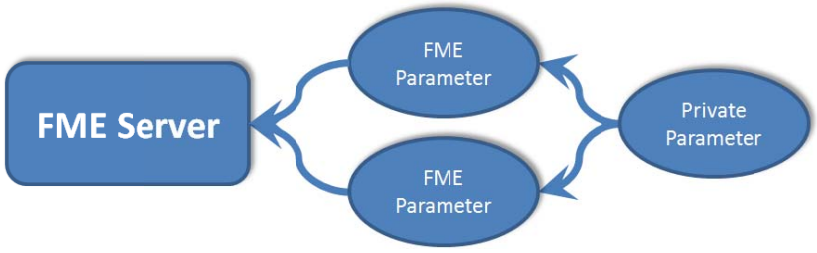
An alternate form of parameter is called **User Parameters**. These are created by the workspace author and are usually linked to an FME parameter to set it implicitly; so these parameters may be said to *indirectly* control a translation.

User parameters have one of two states: **public** or **private**.

Public (or Published) parameters are used to accept input from the end-user:

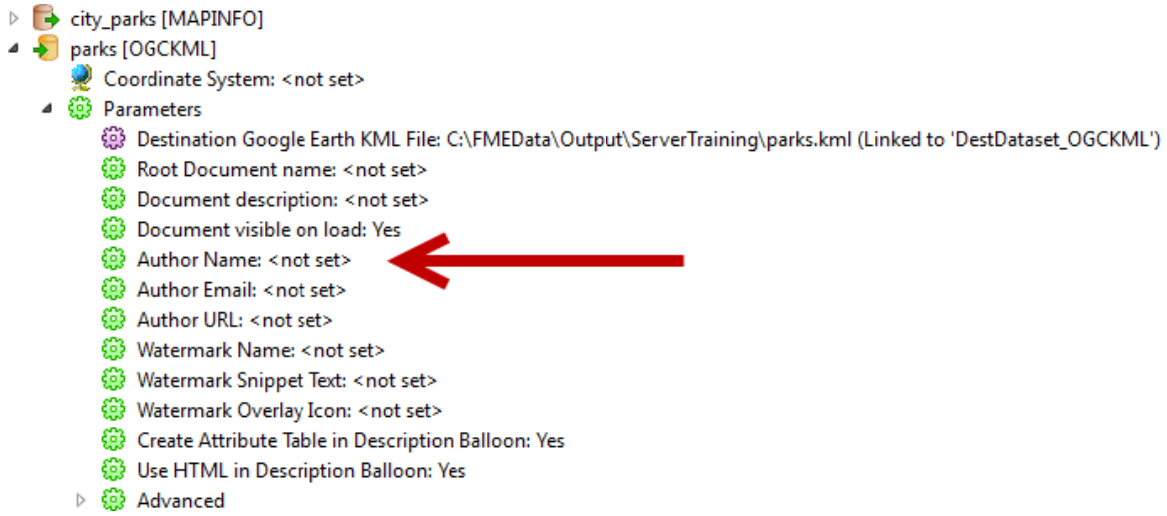


Private parameters are used by the workspace author to share a single value amongst several FME parameters:

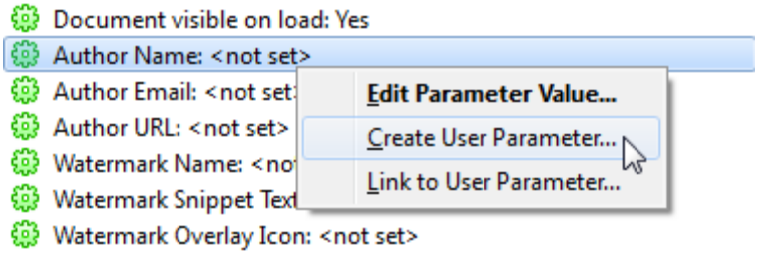


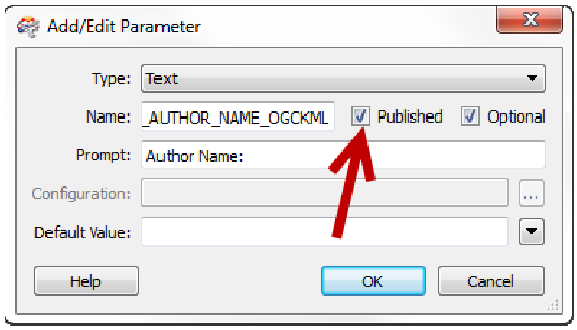
For example, the KML writer has a parameter that defines the name of the author (in this case it means the name of the user, i.e. the person running the workspace and creating the KML data).

This is the FME Parameter:



By right-clicking on this FME parameter, the author can choose to create (and link to) a User Parameter:

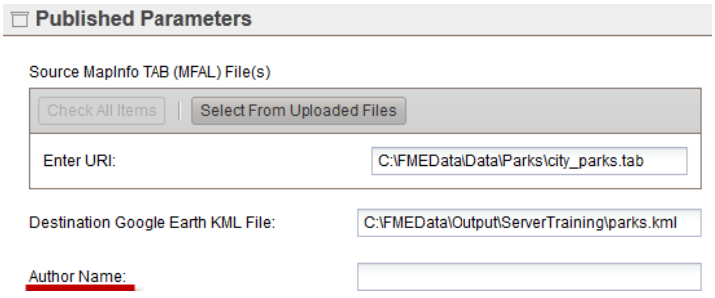




The author gets to choose whether to make this published or private:

With this user parameter, the author may decide to make it private if he wishes to share its value with multiple FME parameters; for example if there were metadata requiring the same information.

Alternatively, he would make it published to have the end user enter a value. When he does this, and publishes the workspace to FME Server, the user parameter appears in the configuration dialog.



Did you notice the source and destination dataset parameters in the above dialog? They are automatically published by FME for all readers and writers.

Developers and Parameters
Developers who create their own interface to FME Server will usually wish to submit information to a server workspace via published parameters. To do this without hard-coding information in their scripts, they will need to be able to retrieve a list of parameters for any particular workspace.

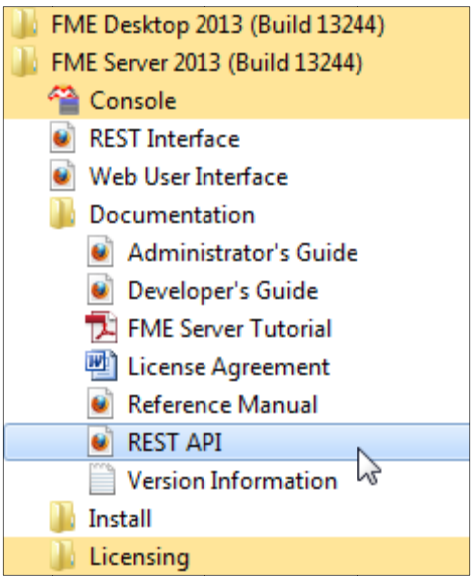
Fortunately, when a workspace is published to FME Server, part of the information that is written to the repository database is a list of these published parameters.

Furthermore, the FME Server REST API includes a set of methods for retrieving all of this information. With these methods a custom web form can be made to present workspace parameters to the end-user, without hard-coding them.

REST Repository Methods
Some repository methods are as follows:

`/fmerest/repositories.<format>`	Retrieve a list of repositories
`/fmerest/repositories/<repository>.<format>`	Retrieve a list of workspaces in a particular repository
`/fmerest/repositories/<repository>/<workspace>/parameters.<format>`	Retrieve a list of published parameters for a workspace

For a full list and further information, see the FME Server REST API documentation, available through the Start menu:



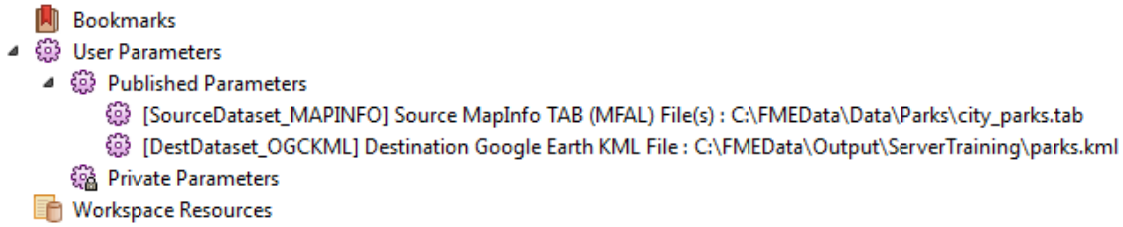
Example 3: User Parameters	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	Create and publish some user parameters
Demonstrates	Published Parameters
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example3Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example3Complete.fmw C:\FMEData\Workspaces\ServerManual\Example3AdvancedComplete.fmw

The workspace created in chapter 2 is to be made available for public use online. However, it is important that the end-users have the ability to set values for some of its parameters.

1) Open Workspace
Start Workbench (if necessary) and open the workspace from Example 1. Alternatively open
C:\FMEData\Workspaces\ServerManual\Example3Begin.fmw

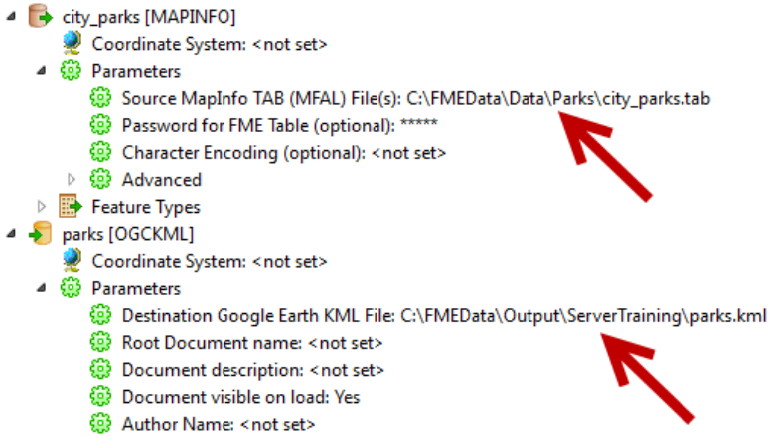
2) Delete Pre-defined Parameters
First of all, it's not necessary to prompt users for the source data or destination data locations, so these published parameters can be deleted.

To do this, expand the list of Published Parameters in the Workbench navigator window.



Select each one of these parameters in turn, and press the delete key to delete them.

Note that this action doesn't remove the reader and writer parameters, or their values, it merely ensures that the end-user is not prompted to enter a new value for them.



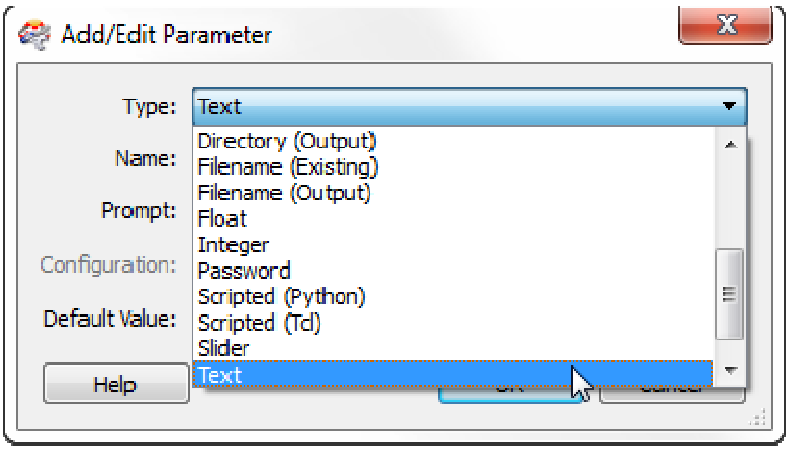
3) Create User Parameter

Now the user should be prompted to enter values for various parameters on the KML writer. The parameters are ones which control fields in the KML output document. They are:

Author Name
Author Email
Author URL

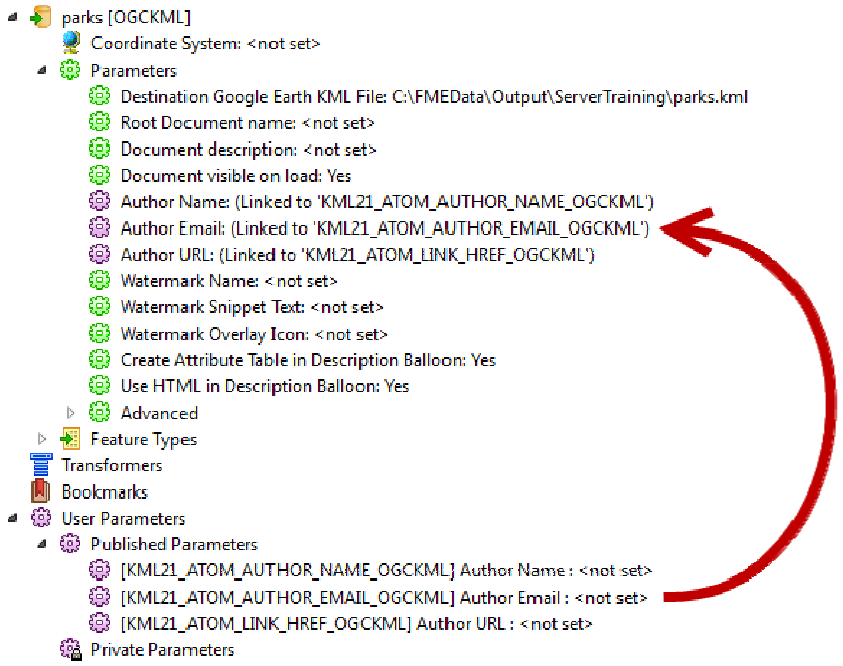
Locate and right-click the Author Name parameter. Choose the option to “Create User Parameter” This will open a dialog in which to define the new user parameter.

Examine the different types of parameters available by clicking on the drop-down list.



Select (or keep) the type of Text, and simply click OK to accept the rest of the parameters. Repeat this action to create user parameters for the Author Email and Author URL fields too.

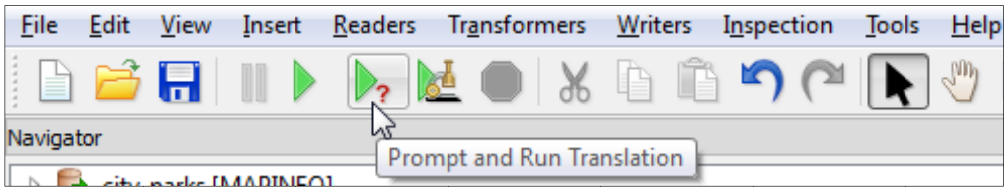
Notice how three user (published) parameters are created and linked automatically to the three FME parameters:



4) Save and Run Workspace (on Desktop)
Save the workspace.

Before publishing it back to FME Server, it's considered good practice to run it from Workbench first to ensure it works correctly.

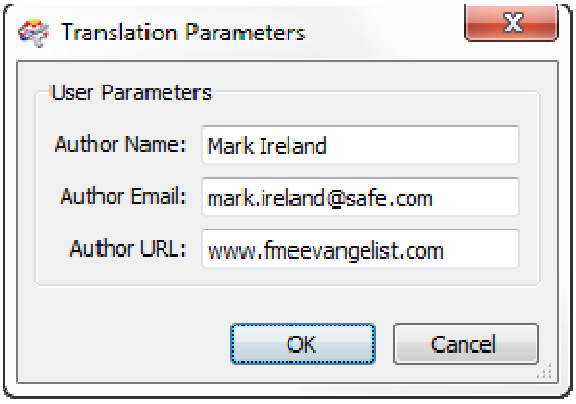
Instead of using the default **Run Translation** button, run the workspace by using the **Prompt and Run Translation** button.



When developing workspaces for FME Server it's always best to test them using Prompt and Run, as it confirms the state of the parameters as seen by the end-user. It is recommended you adopt this technique throughout this training course.

Now you (as the end-user) will be prompted to enter values for the published parameters.

The workspace will run once you press OK, and your values passed to the relevant FME parameters.



You can open the output dataset in a text editor to prove that the information was written as entered:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- Generated by FME 2013 Beta (Build 13244) -->
3  <kml xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:atom=
   "http://www.w3.org/2005/Atom" xmlns="http://www.opengis.net/kml/2.2">
4  <Document>
5  <name>parks</name>
6  <atom:author>
7  <atom:name>Mark Ireland</atom:name>
8  <atom:email>mark.ireland@safe.com</atom:email>
9  </atom:author>
10 <atom:link href="www.fmeevangelist.com"/>
11 <visibility>1</visibility>
```

5) Publish Workspace
If the workspace ran correctly to completion, then it can be published to FME Server.

Select **File** > **Publish to FME Server** from the menubar.
Enter the same connection details as before, and then publish the workspace to the Training repository.

Once more, register the workspace with the Job Submitter service and click **Publish**.

Check the log window in FME Workbench to confirm the workspace was published successfully.

Interestingly, you might notice that FME Server has actually uploaded data with the workspace this time around. This only occurs now because the published parameter for source data has been deleted.

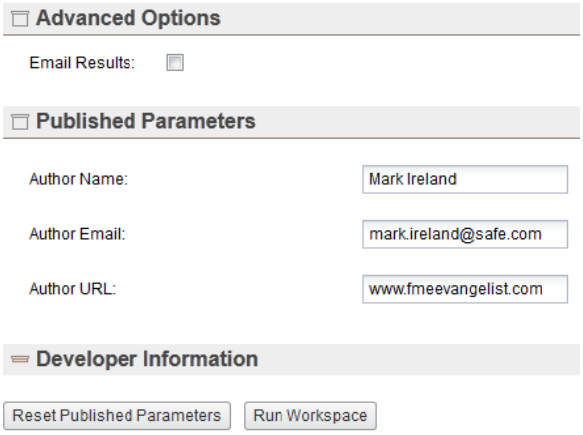
As long as there is a published parameter, FME will assume there is no chance of a problem; should the source data be missing, the end-user still has the ability to specify the proper location.

However, when there is no published parameter, missing data would have serious consequences on the ability to run the workspace. Therefore it uploads the data automatically to ensure that the workspace is always able to be run.

6) Run Workspace (on Server)
Open the FME Server web interface and log in as in previous examples.

Navigate to Repositories > Training > <workspace>

Click the Configure option under the Job Submitter service:



Notice how the configuration page has fetched the parameters published in the workspace, including Author Name and Author Email, and has ignored the parameters that were deleted.

Enter your name and other information into the fields. Unless you unchecked the "optional" box when creating the parameters, none of these are compulsory.

Click **Run Workspace** to run the translation.
Again, check the output dataset to ensure the translation ran correctly.

In this example the destination dataset is defined using a local path. This is allowable because FME Server is operating on the same machine as FME Desktop, and that you are logged in to.

In a real-world, distributed server, environment shared paths (such as a Windows UNC path) are more likely. That way FME Server can write to directories that are not local to the installation.

Advanced Task
If you have time to continue this section, why not try this advanced task.

1) Create Parameter
Back in Workbench, locate the advanced KML writer parameter called Force Geometry Orientation. This parameter controls whether KML geometry is output with a left-hand (counter-clockwise) orientation, a right-hand (clockwise) orientation, or neither.

Create a published parameter from this FME parameter. Note that the type is Choice, and that the three choices of value are “none”, “left”, and “right”.

Republish the workspace.

2) Check Developer Information
In the FME Server web interface, open the configuration dialog for the workspace (Job Submitter service) and click the Developer Information header to expose the URL examples.

Copy the Form Example HTML and create a web page from it. Notice that the Force Geometry Orientation parameter is a text field, rather than a drop-down list of values. This is because the form example HTML is relatively unsophisticated code. Compare this to the parameter in the “Published Parameters” section of the FME Server interface, which was developed to actually show this as a drop-down list.

If you have the time (and the skills in web development), create a web page for submitting a workspace that uses a drop-down list for the Force Geometry Orientation parameter.

Also, perhaps, consider implementing validation on the Author Email and Author URL parameters to ensure that the user has entered valid values for these fields.

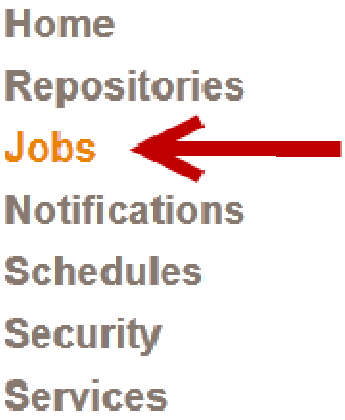
Would you validate these in the web form or inside FME Workbench? What are the advantages of each method?

Jobs

Querying the jobs history

Rather than checking the output dataset, another way to test whether the translation worked is to look at the jobs history. This will tell you whether the translation succeeded or failed, and shows other general information such as the engine name, workspace name, time started, and time ended.

The Jobs Menu
The FME Server jobs history is accessed by clicking Jobs in the menu on the left-hand side of the FME Server web interface.



The dialog has tabs to display jobs that have been completed, jobs that are waiting in the queue, and jobs that are currently running.

Queued Jobs
Queued jobs are those that have been submitted to FME Server that are still sitting in the queue and waiting to be assigned to an engine. Jobs will be queued when there is not an engine available with which to process them, or the specified engine for that job is already busy.

The job queue is controlled by a part of the FME Server core called the Transformation Manager. It sorts jobs by priority first and then by order of submission.

One other feature of the job queue is “time to live” (ttl). This feature ensures that a job does not become “stale” by removing it from the queue if its queuing time has exceeded its “time to live”.

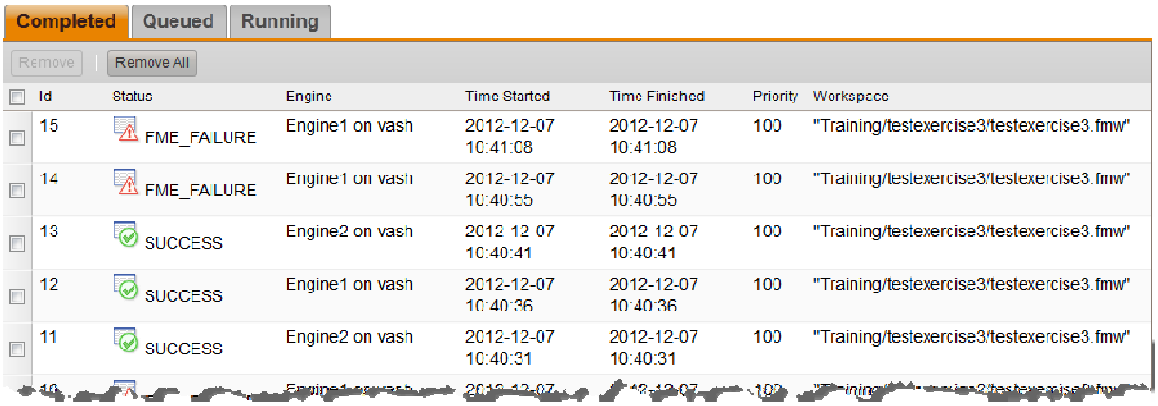
Running Jobs
Running jobs are those that have been assigned to an engine and that are currently being processed by that engine.

Once a job has finished then it is added to the Completed Jobs list and the next job in the queue is run.

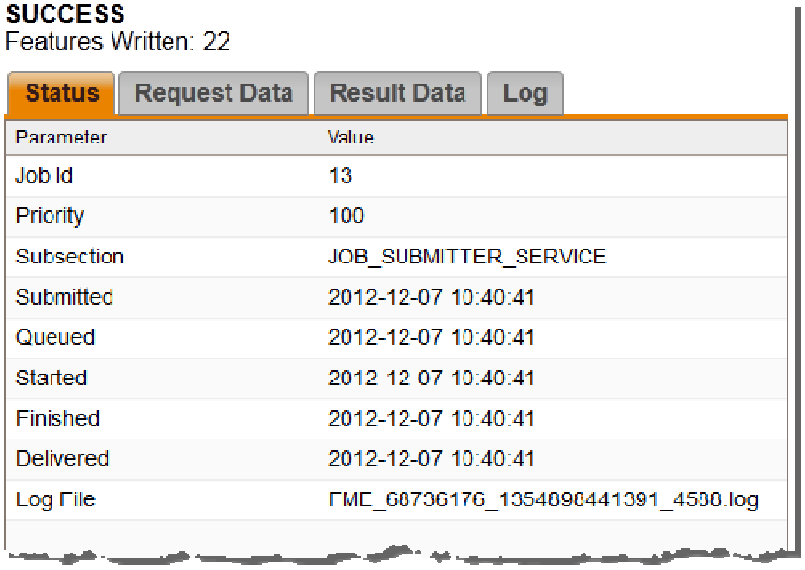
The Queued and Running Jobs pages are useful for authors/administrators who wish to know if a particular job is being run as planned.

Completed Jobs

Completed jobs are those that have been processed already. This page shows a list of previously run jobs, their status, and other information that is useful to help debug problems with.



Clicking on a particular job returns more detailed information about it:



These tabs let you find:

- General information on the status of the job
- The request sent to FME Server to start the job
- The information returned by FME Server at the completion of the job
- The FME log file for the workspace translation

Of these, the Result Data and the Log are the most useful for diagnosing failure of a workspace to run.

Example 4: Job Status	
Scenario	FME user; City of Interopolis, Planning Department
Data	None
Goal	Check Job Status of completed workspace
Demonstrates	Job section of FME Server web interface
Starting Workspace	None
Finished Workspace	None

Instead of checking the output dataset, let's check the job status for example 3.

1) Open FME Server Web Interface

Open the FME Server web interface and log in as necessary.

Click Jobs in the main menu to open the Jobs dialog.

Locate the previously run job in the Completed jobs list. Click on it to reveal more detailed information.

2) Download Log File

Click the tabbed labelled Log. Use the download option to download the log file. Open it in a text editor and check to ensure the data was output as planned, and to see how many features were written.

```
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|                               Features Written
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|city_parks                                                                  22
|==============================================================================
|Total Features Written                                                      22
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|                            Features Read Summary
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|city_parks                                                                  22
|==============================================================================
|Total Features Read                                                         22
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|                           Features Written Summary
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|city_parks                                                                  22
|==============================================================================
|Total Features Written                                                      22
|=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
|Translation was SUCCESSFUL with 0 warning(s) (22 feature(s) output)
|FME Session Duration: 0.1 seconds. (CPU: 0.1s user, 0.0s system)
|END - ProcessID: 4588, peak process memory usage: 65832 kB, current process memory usage: 65684 kB
```

Advanced Task

Open up the REST API documentation and look for a method to extract a list of completed jobs. The name of the core component that controls queuing is a hint to where to find this information.

Generic Readers and Writers

Format flexibility is achieved through FME's Generic Writer and Generic Reader

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.

Since authors often wish to create a solution that works for as many users as possible, they should not always tie their workspaces to a single format. Format-flexible workspaces ensure as many end-users as possible can use them, and can be created using FME's Generic Reader and Writer.

Generic Writer

The generic writer includes a parameter that defines output format. This way a workspace may be created that can write to any format set by that parameter. By publishing the parameter the user gets to choose the output format at run-time.

There are a couple of points to keep in mind when using the writer:

- The parameter for output format is automatically published, but only as a text field that accepts FME's short form name for any format. Best practice decrees this should usually be replaced by a new Choice with Alias parameter; to reduce the list to a reasonable set of choices, with descriptive format names.

- Each writer format has its own specific parameters, and these may still need to be set when a generic writer is used. For example, a specific template file might need to be used when AutoCAD is chosen as the output format. This can be achieved by adding a writer of the same format and setting the parameters in that writer. The Generic writer will inherit the parameters of this dummy writer, even if no features are connected to it.

Generic Reader

Less commonly used, the generic reader works similarly but includes a parameter for input format. This way a workspace may be created that can read any format chosen by the user at run-time.

Unlike the generic writer, the input format parameter is not published by default. When the input format parameter is unset, the reader will "guess" at the format from the file extension. Caution is required when the extension is one shared by multiple formats. For example, the *.mdb* extension is used by ESRI Geodatabase, GeoMedia Access Warehouse and others.

When using the generic reader with FME Server, attention must still be given to the Source Dataset parameter. This parameter needs to be set to the type of file or files FME would expect to see. For example, out of all the files in a Shape dataset, only the .shp file names need to be passed to the Source Dataset parameter, either by the user or programmatically.

Custom Parameters

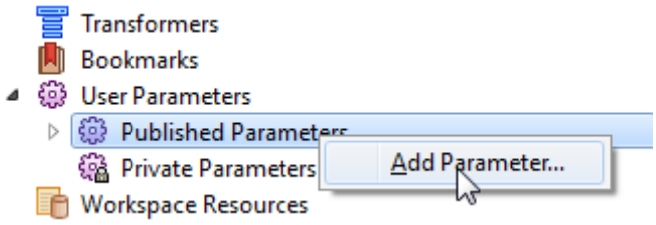
Custom user parameters provide a way to accept user input, without necessarily tying it to an FME parameter

Besides creating a user parameter from an existing FME one (as we did in the previous example), another way to accept user input is to make a new user parameter from scratch.

We sometimes call these custom user parameters, as they aren't automatically linked to an FME parameter.

Creating a Custom Parameter

Custom parameters are created by right-clicking the User Parameters section of the Navigator window and choosing "Add Parameter":



Linking Custom Parameters

Once a custom parameter is created, it can be linked to an FME parameter. Remember, linking a user parameter to an FME parameter provides indirect control over a translation.

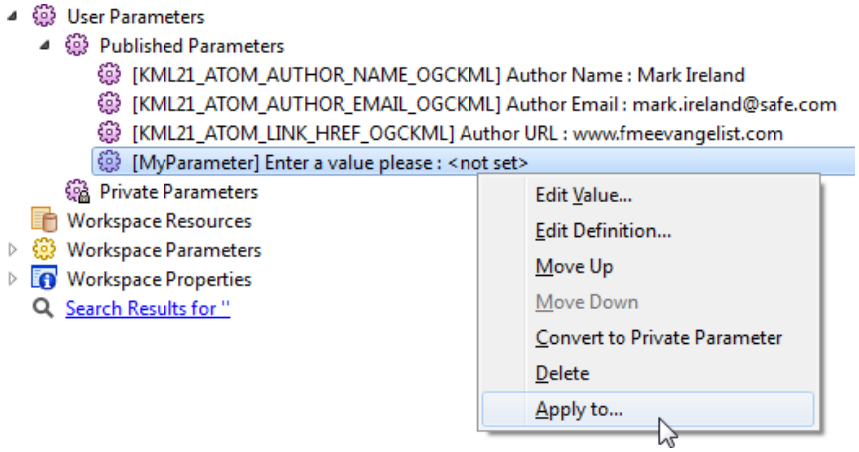
There are two ways to create this link:

Start from FME parameter

1. Right-click the FME parameter
2. Choose *Link to User Parameter*
3. Choose the custom parameter to be linked to

Start from custom parameter (below)

1. Right-click the custom parameter
2. Choose *Apply to…*
3. Select the FME parameters to apply it to



Parameter Types

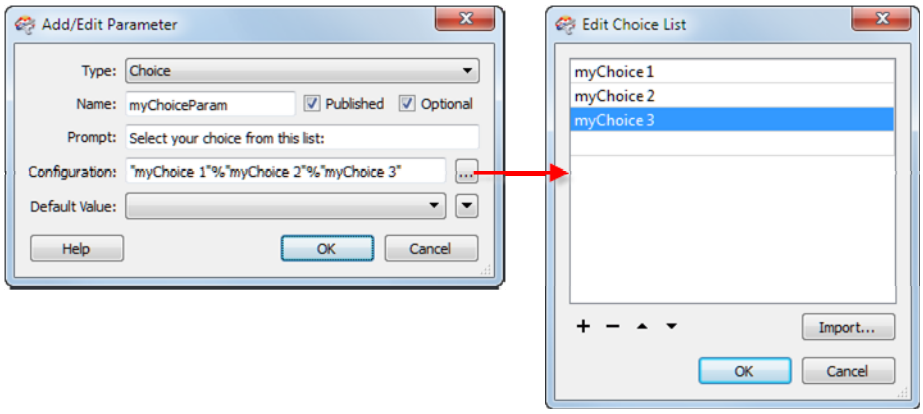
There are many types of parameter available. The most commonly used are:

Choice:	A single choice from a pick list of values
Choice with Alias:	A single choice from a pick list of alias values, each with a hidden true value
Choice with Alias (Multiple):	Like Choice with Alias but allows multiple choices
Coordinate System Name:	The selection of a coordinate system
Filename (Existing):	The selection of a single, existing file
Float:	A floating point number
Integer:	An integer (number with zero decimal places)
Text:	A text entry field and the default parameter type

Choice Parameters

The Choice type of parameter is one of the most used in FME Server workspaces. A Choice parameter is defined as a series of entries, in a list, that the user gets to choose from at run time.

Defining a choice includes a dialog – reached through the Configuration button – that lets the workspace author define the list of choices.



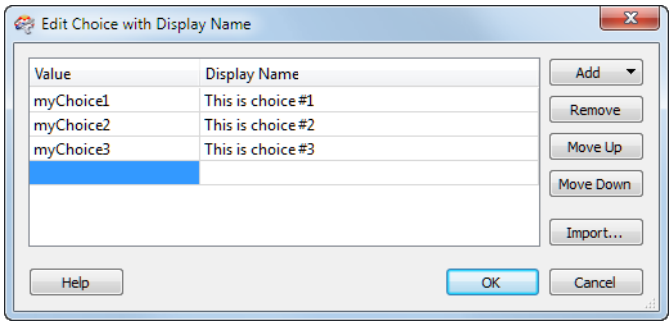
A choice parameter is the one commonly used for unlinked parameters, with a common set of choices being a simple Yes/No.

There are variations on this theme – including “Choice (Multiple)” and “Choice or Text” – but the most important is almost certainly the parameter type called “Choice with Alias”.

Choice with Alias Parameters

In a Choice with Alias parameter, the configuration dialog is a lookup table of values.

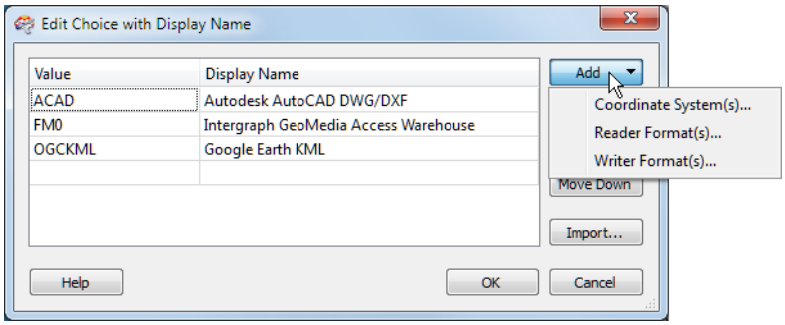
The end-user selects a "Display Alias" from the list and the lookup table used to determine what true value is returned. It's a way to let the user select one value, but substitute it with another.



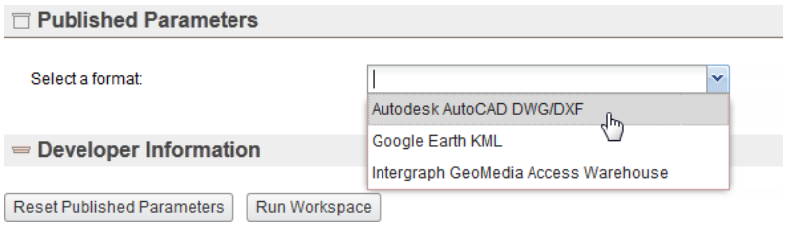
The reason this is important is that the input expected by some FME parameters is terse, and not descriptive. A lookup table like this allows the user to pick from a more descriptive alias, but still return the correct (non-descriptive) value to the FME parameter.

For example, rather than pick ACAD, OGCKML, or FM0 from a list of formats, the user can pick from the more descriptive AutoCAD DWG, Google Earth KML, and Intergraph GeoMedia; but still have the correct keyword passed to FME.

In fact, for ease of authoring, the Choice with Alias configuration dialog actually contains shortcuts to every FME Reader, Writer, and Coordinate System; so that a value and display alias can be added more easily, and without having to look up the correct values.



In FME Server the selection will now look something like this:



The parameter type Choice with Alias (Multiple), has the same lookup table structure as Choice with Alias, but allows the user to select multiple values from the list, not just one. This is most useful for letting users choose multiple feature types from a list.

Example 5: Choice with Alias Parameter	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	Create a Choice with Alias parameter to define output format
Demonstrates	Choice with Alias Parameters
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example5Complete.fmw

The workspace created in previous examples works well enough, but it limits users to a single output format; KML. What is needed now is a workspace that will allow users to select which format of data they wish to write.

So, we will rewrite the workspace, this time using a Generic format and a Choice with Alias parameter to let the user choose which format to write.

1) Generate Workspace
Start Workbench (if necessary). Visit the Start tab and select the option to **Generate workspace** (or use the shortcut key, Ctrl+G)

Create a new workspace using the following settings:
Reader Format: *MapInfo TAB (MFAL)*
Reader Dataset: *C:\FMEData\Data\Parks\city_parks.tab*

Writer Format: *Generic (Any Format)*
Writer Dataset: *C:\FMEData\Output\ServerTraining*

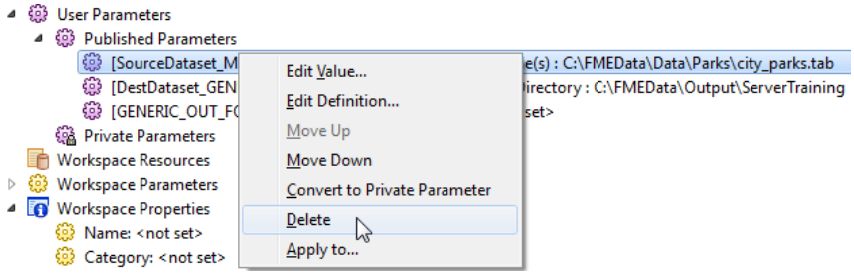
Workflow Options: *Static Schema*

As a writer, the generic format that allows selection of output format through a parameter.

Save the workspace locally at
<My Documents>\My FME Server Workspaces\Example5.fmw

2) Remove Parameters
By default published user parameters are created for the source and destination datasets, and the output format. Because this example doesn't need the source and destination parameters, and because a new format parameter will be created, these can all be removed.

In the Navigator window, expand the list of published parameters. Right-click each of these parameters in turn and choose Delete.

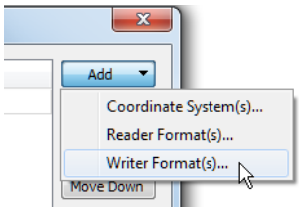


3) Create Custom Parameter
Still in the Navigator window, now right-click on the Published Parameters section, and choose Add Parameter.

Set the options as follows:

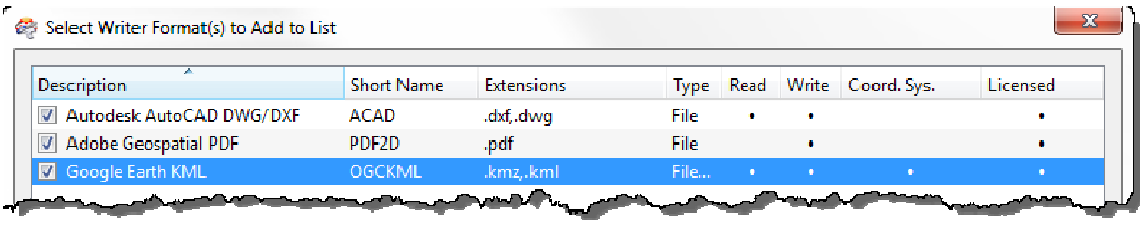
Type	Choice with Alias
Name	OUTPUT_FORMAT
Published	Yes
Optional	No
Prompt	Output Format:

Now click the button next to the configuration field. In the dialog that opens, click the Add button and select Writer Formats:



Search for, and then add, the following formats:

PDF2D	Adobe Geospatial PDF
ACAD	Autodesk AutoCAD DWG/DXF
OGCKML	KML

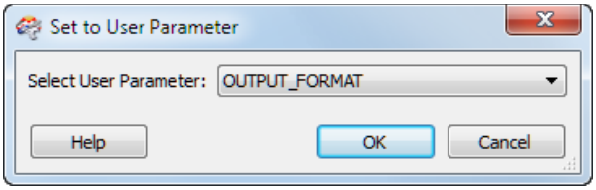


Click **OK**, then **OK** again and once more to close these dialogs.

4) Link Custom Parameter
The user parameter must now be linked to the FME parameter for output format.

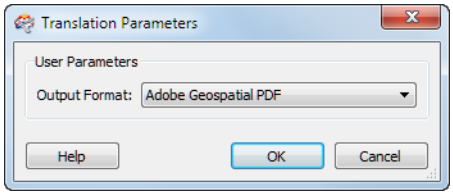
On the Navigator, locate the writer parameter ServerTraining [GENERIC] > Parameters > Output Format. Right-click the parameter and choose Link to User Parameter.

Select OUTPUT_FORMAT from the list provided, as the user parameter to link to. Click **OK**.



5) Run Workspace on FME Desktop
Save the workspace. It is now ready to test using Prompt and Run.

Select **File** > **Prompt and Run Translation** from the menubar. The translation parameters dialog will open and show an output format parameter. Select Adobe Geospatial PDF (notice the alias is being displayed) as the output format. Click **OK** to run the workspace.

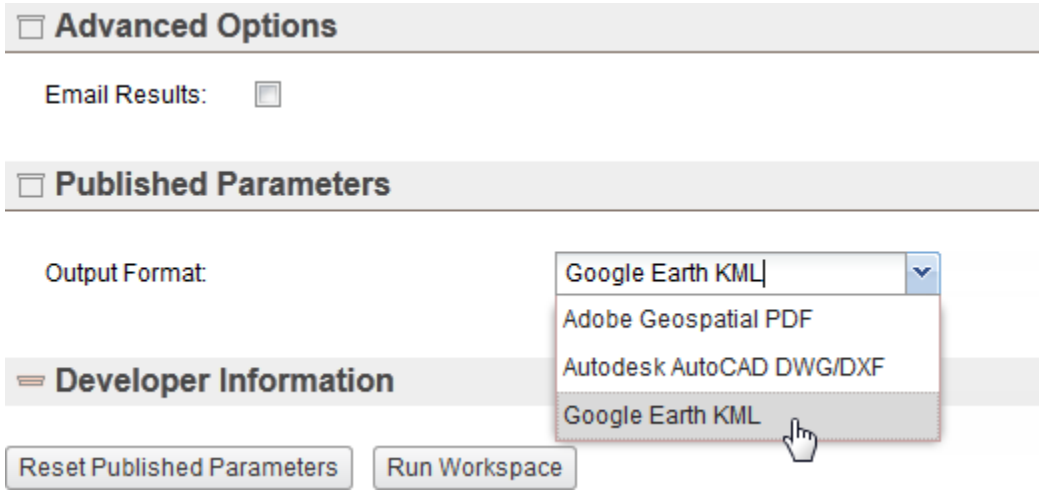


The output will be written to:
C:\FMEData\Output\ServerTraining\ServerTraining.pdf

6) Publish to FME Server
Use File > Publish to FME Server to publish the workspace.
Publish it to the Training repository and register it with the Job Submitter service.

7) Run Workspace on FME Server
Open the FME Server web interface and navigate to the published workspace.
Select the Job Submitter > Configure option.

Notice how the FME Server interface prompts the user to select the output format.

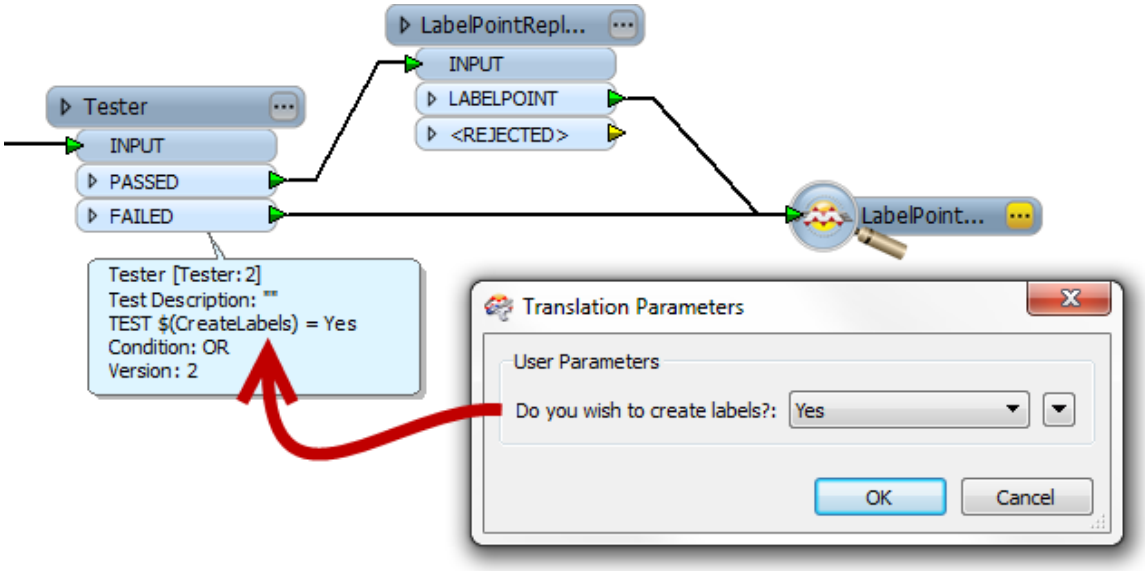


Choose a format and run the workspace to ensure it is working correctly.

Unlinked Custom Parameters
Although the main purpose of a user parameter is to indirectly control an FME parameter, an alternative scenario exists that doesn't include linking.

The scenario is when the author doesn't want to give the user control over how a particular transformation works, but instead to decide whether to use that transformation at all.

Take this example workspace:



Here the author has given the end-user the choice of whether to create labels for their data or not.

This is achieved through a custom user parameter. The parameter's value is tested to see if it is "Yes" or "No" and the features routed accordingly through the workspace. If the answer is "Yes" then labels are created. If the answer is "No" then the label-creating transformer is bypassed.

In this way a user parameter can be used to control large sections of workspace, rather than being tied to one particular FME parameter.

Shared Parameters
As mentioned, it is possible to link the same user parameter to multiple FME parameters.

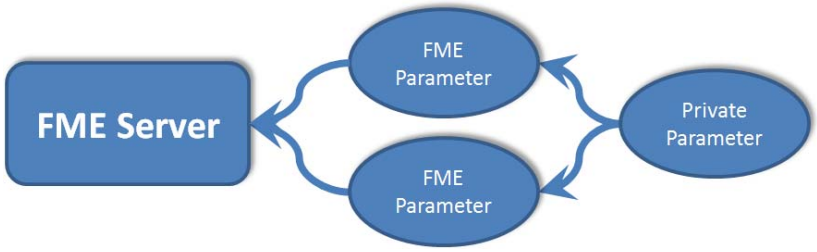
This is called a Shared Parameter, as each FME parameter to which a user parameter is linked, will share the value of that user parameter at runtime.

A good use case is a database password parameter: there might be several FME parameters in a workspace, all of which need the same database login information. If a separate parameter were created for each, then they would all need to be set with the same information.

However, by sharing the same user parameter with each FME parameter, the information only needs to be entered once.

Shared parameters may be either published or private.

Shared published parameters will allow the user to supply a single value to multiple FME parameters, whereas a private parameter allows the author to supply a single value to multiple FME parameters.



For example, consider a workspace containing 1,000 transformers; 10 of which are Snappers that all use the same tolerance, but that the user is not required to set.

To change the tolerance it would be madness to track down each Snapper and change it manually. So, the author creates a private shared parameter linked to each Snapper; that way a change of tolerance is made by simply changing the user parameter value.

But - since this is a private parameter - the user is not prompted to enter a value at run-time.

Private Parameters have an obvious role for FME Server, since a workspace author won't want to prompt the end user for more parameters than is necessary. By making parameters private they won't show up on the Server web interface.

Scripted Parameters – covered in a later section – are always private parameters. They are useful when the value returned needs to be processed in some way before being passed to FME.

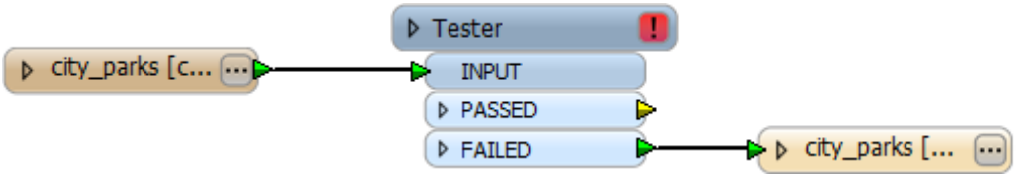
Example 6: Reusing Parameters	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks (MapInfo TAB)
Overall Goal	Create a Choice with Alias parameter to define output format
Demonstrates	Reusing a parameter. Redirecting data according to a parameter.
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example6Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example6Complete.fmw C:\FMEData\Workspaces\ServerManual\Example6AdvancedComplete.fmw

A useful enhancement to the Generic workspace will be to style the output when KML is selected.

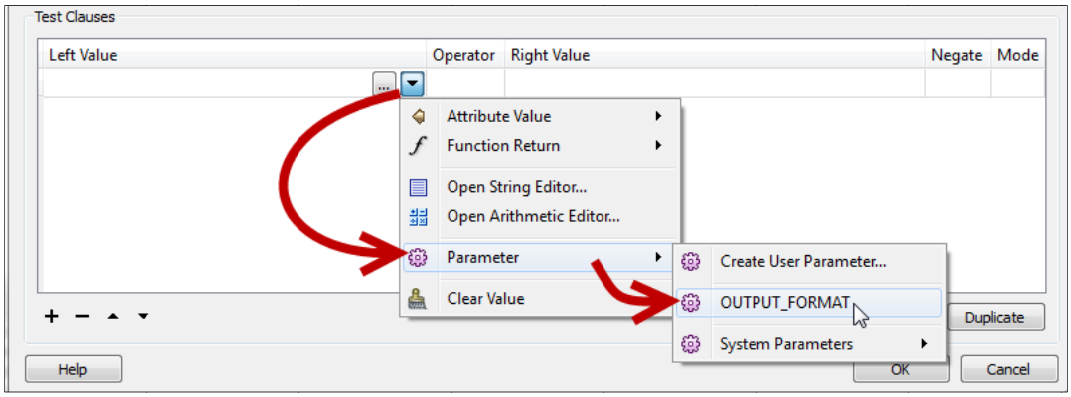
Here, we don't need to create a separate (unlinked) parameter, because we already have the output format parameter to tell us whether KML has been selected.

1) Open Workspace
Open the workspace from the previous example, or use the new starting workspace.

2) Add Tester Transformer
Add a Tester transformer between the reader and writer feature types. The output port of the Tester to be connected is the FAILED port.

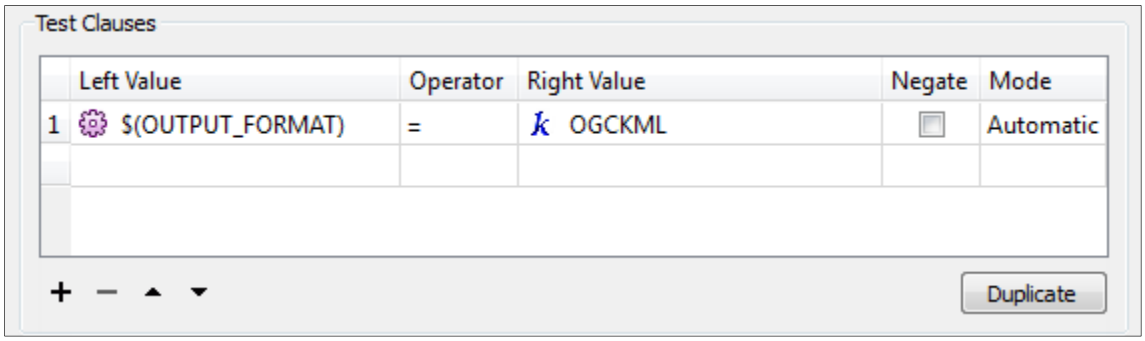


3) Set Tester Parameters – 1
Open the Tester parameters dialog. We want to test whether the KML format has been selected so, for the Left Value, select Parameter > OUTPUT_FORMAT from the drop-down menu:

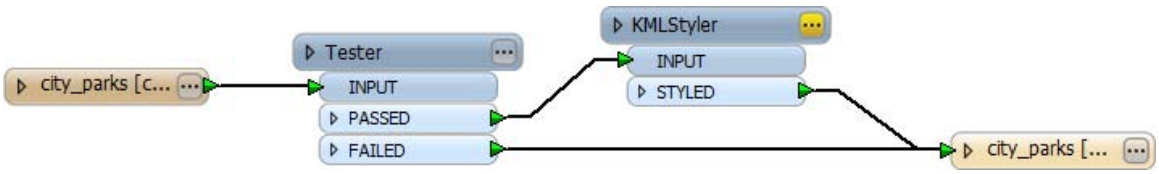


4) Set Tester Parameters – 2
Now set the Tester operator to equals (=).

Finally, enter OGCKML into the Right Value field. Remember this, not the alias, is what is returned by this choice parameter.



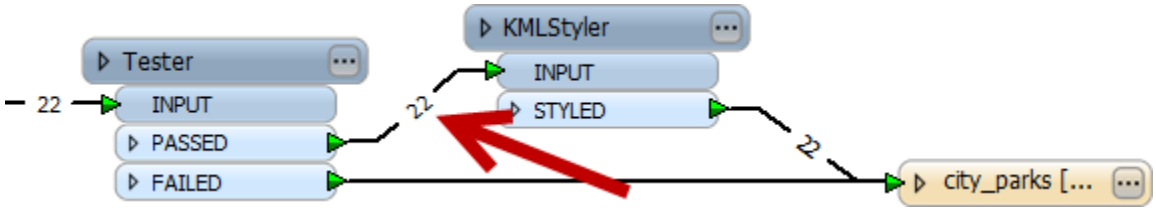
5) Add KMLStyler Transformer
Add a KMLStyler transformer connected between the Tester:PASSED port and the writer feature type.



6) Set KMLStyler Parameters
Open the KMLStyler parameters dialog and set an outline color and a fill color. Since these are park features, something in green would be fairly conventional.

7) Run Workspace on FME Desktop
Save the workspace. Select **File** > **Prompt and Run Translation** from the menubar to run it
This time be sure to select “Google Earth KML” as the format to write.

If the workspace is operating correctly, you should see the features output the Tester transformer through the PASSED port and then pass through the KMLStyler.



Try running the workspace with a different output format to ensure the KMLStyler is bypassed as it should be. Then publish the workspace to FME Server and run it there, to ensure it still operates correctly.

Advanced Task
Consider example 3, and how we prompted the user to enter values for various parameters on the KML writer:

- Author Name
- Author Email
- Author URL

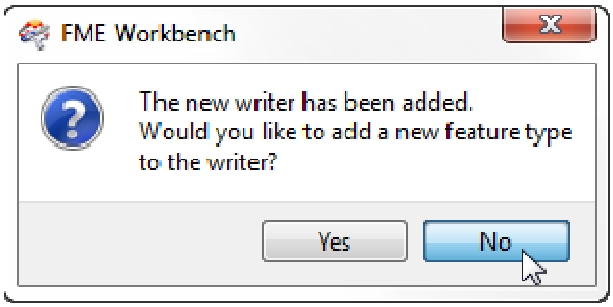
Since we are using the Generic Writer these parameters don't exist, and so cannot be set the same way. If we wished to set them we would need to use a dummy writer.

1) Add KML Writer
Use Writer > Add Writer on the menubar to add a new writer. When prompted set:

Writer Format: *Google Earth KML*
Writer Dataset: *C:\FMEData\Output\ServerTraining\parks.kml*

Although the output Dataset is not really important (because it will never be written to) we still need to set something to avoid being prompted for a value at run time.

When prompted to add a new feature type, click **No**.

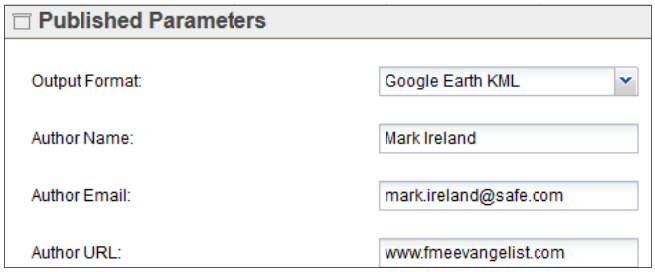


2) Publish KML Writer Parameters
Now a KML writer exists, the three author parameters can be located and published. It doesn't matter that this writer receives no data; the Generic Writer will use these parameters if it is notified to write KML data.

Publish the three parameters as you did in example 3. At the same time remove the Destination Google Earth KML File published parameter, which was created when the KML writer was added.

3) Run Workspace
Now run the workspace using File > Prompt and Run. Enter values for the Author parameters. Open the output KML dataset in a text editor to verify that the chosen values were picked up and written by the Generic writer.

Now publish the workspace to FME Server and run it there to prove that the translation still operates as expected.



Very Advanced Task
Consider this limitation of the workspace and its impact on the FME Server interface: you can't set/use one published parameter based on the value of another.

For example, there's no method to expose the KML Author Email parameter only when the format is set to KML. The parameter still appears even when the selected format is PDF (for example).

To do this would require a custom web form.

So, create a custom web form that only exposes the Author parameters when the user chooses KML as the output format.

Dataset Parameters

Each reader has a source dataset parameter, and each writer a destination dataset parameter.

Each Reader and Writer in a workspace has a source or destination dataset parameter. Sometimes – for example with databases – there are actually multiple parameters to define a dataset.

For file and folder-based datasets, there is usually only a single parameter: the location of the file to be read or written.

Source Datasets
Wherever your data is stored, FME Server must be able to access it to use it!

If you have file-based data stored on a different server you need to use network paths to let FME Server access the data.

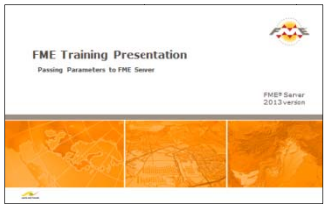
Microsoft's Universal Naming Convention (UNC), for example *\\ComputerName\SharedFolder* specifies a common syntax to describe the location of a network resource. This method is a good way to define shared directories or mapped drives for your source data in Windows.

Destination Datasets
Again, wherever data needs to be written, FME Server must have access to it.
A UNC path is also a way to define a shared folder for FME Server to write to.

Published Dataset Parameters
If a source data parameter is published, then the end-user will be able to alter the location of the source data. Similarly, if the destination data parameter is published, they will be able to alter where output data is written to.

It's important to note that FME creates most workspaces with the dataset parameters published by default; therefore they must be unpublished if the end-user is to be prevented using them.

Module Review



This module introduced you to passing parameters to an FME Server workspace.

What You Should Have Learned from this Module
The following are key points to be learned from this module.

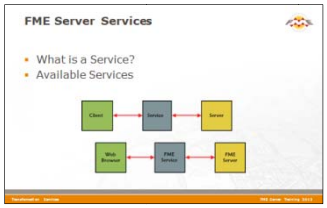
Theory

- Components of an FME workspace are controlled by parameters. When the end-user is to be given control of a component, its parameters should be *published*.
- The Jobs section of the FME Server web interface is used for discovering the status of a particular workspace translation
- Format-independent workspaces can be created with the Generic Format reader and writer.
- Custom Parameters are those created from scratch, rather than from an existing FME parameter

FME Skills

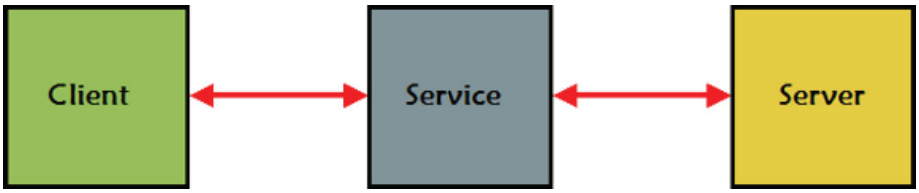
- The ability to create and delete published parameters inside an FME workspace
- The ability to browse the FME Server web interface to check the status of a job
- The ability to create custom User Parameters, link them to FME parameters, and use them independently
- The ability to use the Generic Writer format

FME Server Services

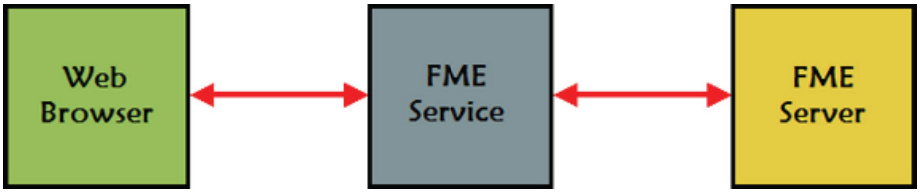


Many standard operations using FME Server take place using something called a 'service'. FME Server provides a wide range of services to carry out common tasks.

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.



For FME Server, a service lets you send specific types of requests to FME Server and provide results 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.

Professor Spatial F.M.E., E.T.L. says...
'Although the concept sounds complicated, a service is 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:

OGC Web Feature Service (WFS)	OGC Services
OGC Web Mapping Service (WMS)	
Data Download Service	Transformation Services
Data Streaming Service	
Job Submitter Service	
KML Network Link Service	
Catalog Service	Utility Services
Data Upload Service	
Token Service	
Web Connection (SOAP) Service	
REST Service	
Notification Service	Notification Services

Remember that services can communicate in both directions. Transformation and OGC services – for example Data Download – are primarily for Server to deliver data to the end user.

Utility services – such as Data Upload – are generally for communication TO the server. In some cases this will be communication from the end-user, but in most cases these are facilities that an author or developer will be using in a way that's hidden from the user.

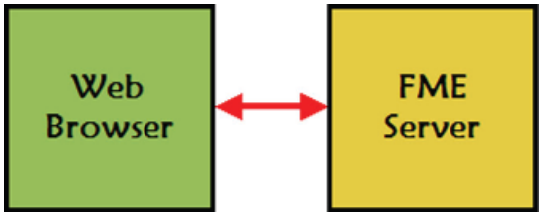
The Notification Service is one that is used equally for passing information into and out of FME Server. Incoming messages notify FME Server to take some action, whereas outgoing messages alert an end-user that some sort of event has occurred.

Workspaces and Services

Workspaces published to FME Server can be registered against a number of services

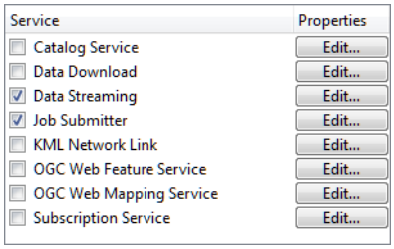
When a workspace is published to FME Server it can be registered with a particular service. The services it can be registered with are either transformation or OGC services.

Basically, these services allow the workspace to communicate to the end-user's client (e.g. web browser) in a number of different ways.



The differences between some of these services and the output they produce can be subtle, so it's important to match your requirements to the correct service carefully.

Registering with a Service
The Job Submitter service is selected in the FME Server publishing wizard, whenever a workspace is published. However, the same dialog has other service options.



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.

When a workspace is not registered against a service, you can run it as a regular translation using the FME Server Console or the *FMEServerJobSubmitter* transformer in Workbench.

When a workspace is registered against a service, it can be used by that service and it's still available for use as a regular translation using FME Server Console.

It's important to understand that a workspace may be registered against one service, many services, or no services at all.

OGC Services

OGC services are Transformation Services for handling workspaces in adherence to open GIS standards

The Open Geospatial Consortium (OGC®) promotes interoperability through the creation of interface specifications known collectively as OpenGIS®. There is a whole array of data specifications; from the well-known, such as KML, to the lesser-known, such as GeoRSS. FME supports many of these OGC specifications.

Many OGC specifications – such as the KML format – are treated as a special case in FME because they are related to a particular format (FME reader/writer)

Therefore, in FME Server, OGC Web Services refers to two specifications without a particular reader or writer: WFS (Web Feature Service) and WMS (Web Mapping Service).

Web Feature Service
WFS is a specification for handling vector data over a web connection.

Any FME workspace with a GML format writer can be published to FME Server as a WFS service. However, WFS has a number of keywords that must be handled using published parameters:

- bboxWest, bboxSouth, bboxEast, and bboxNorth
- destCoordSystem
- FEATURE_TYPES
- REQUEST_VERSION

These parameters are case-sensitive, and must be defined exactly as shown.

- bboxWest, bboxSouth, bboxEast, and bboxNorth are used – as with other services – to spatially define the area of data to be read.

- destCoordSystem defines the output coordinate system required

If no Coordinate System or Bounding Box is submitted with the WFS query, then default values are used. Defaults can be configured using the FME Server web interface.

- FEATURE_TYPES specifies the feature types (also referred to as layers or tables) required in the output

- REQUEST_VERSION defines the version of WFS the request is using—valid values are 1.0.0 and 1.1.0

Web Mapping Service
WMS is a specification for handling raster and SVG data over a web connection.

More specifically, WMS is a specification for handling pictorial representations of map data. WCS (Web Coverage Service) is required for true raster formats, such as GeoTIFF and DTED.

Any FME workspace with a WMS-compliant format writer can be uploaded to FME Server as a WMS service. The following formats are compliant and tested with FME Server:

- GIF
- JPEG
- PNG
- SVG

Like WFS, WMS supports keywords that are handled using published parameters.

Any FME workspace with a GML format writer can be published to FME Server as a WMS service. However, WMS has a number of keywords that must be handled using published parameters:

- bboxWest, bboxSouth, bboxEast, and bboxNorth
- destCoordSystem
- FEATURE_TYPES
- OUTPUT_FORMAT
- TRANSPARENT
- BGCOLOR
- WIDTH
- HEIGHT

Again, the parameters are case-sensitive and must be named exactly as shown.

The parameters different to WFS are:

- OUTPUT_FORMAT is used to define the format of the output data
- TRANSPARENT specifies the transparency of the raster image
- BGCOLOR defines the raster background color
- WIDTH/HEIGHT are used to denote the size of the raster image

Transformation Services

Transformation services are tools for handling workspaces

Transformation services are those that run a workspace and communicate the output to the user in a variety of ways.

The four transformation services are:

- Job Submitter Service
- Data Download Service
- Data Streaming Service
- KML Network Link Service

Job Submitter Service
So far, the only service used in this course has been the Job Submitter service, which is a way to instruct an FME Engine to run a translation and write the output to the location specified within the workspace.

Data Download Service
The Data Download service instructs FME to run a workspace and write the output to a temporary location. The end-user is provided a link through which the output can be downloaded from that location.

Data Streaming Service
The Data Download service causes a workspace to be run and the output to be streamed directly to the end-user's computer. The action taken on that data depends on the computer's configuration, but in general it will be opened in the default application for that type of data.

KML Network Link Service
Rather the deliver the output data, a KML Network Link Service sends a shortcut to the data streaming service for that data. Google Earth can use that shortcut to pull the data into its display, and can be set up to do so at regular intervals, ensuring the data shown on screen never becomes stale.

Data Download Service

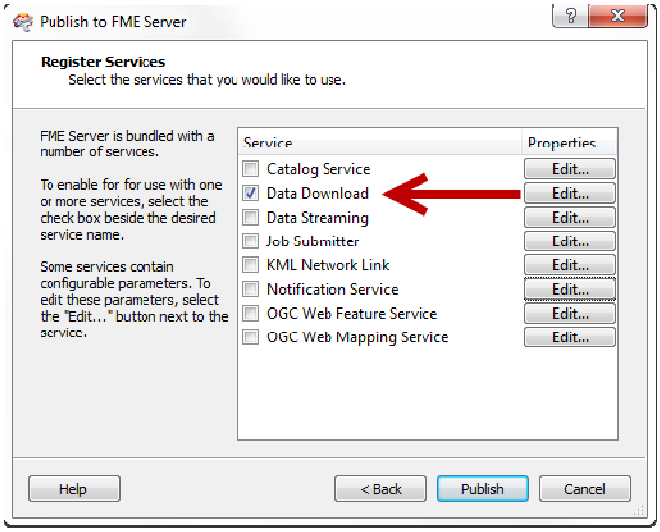
Data Download services are the ideal tools for allowing your end-users to quickly obtain spatial data in the format of their choice.

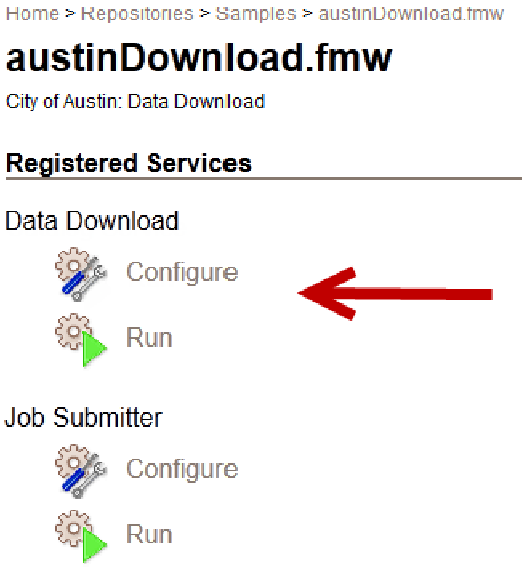
What is the Data Download Service?
The Data Download service lets authors and administrators distribute data to end-users in the format and coordinate system of their (the end-user's) choice.

The Data Download service overrides the destination dataset defined in a workspace, so instead of the output getting written to its intended location, it is instead written to a web-accessible directory.

The end-user is then provided with a link to that location, from which to download the data.

Creating a Data Download Service
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 options become available in the FME Server Web interface for that particular workspace:

Example 7: Data Download	
Scenario	FME user; City of Interopolis, Planning Department
Data	Interopolis Database
Overall Goal	Create a data download service
Demonstrates	Data Download services
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example7Complete.fmw

This workspace will be created specifically for a Data Download service.

1) Start Workbench
Start Workbench (if necessary) and use the Generate Workspace tool to create a workspace as follows:

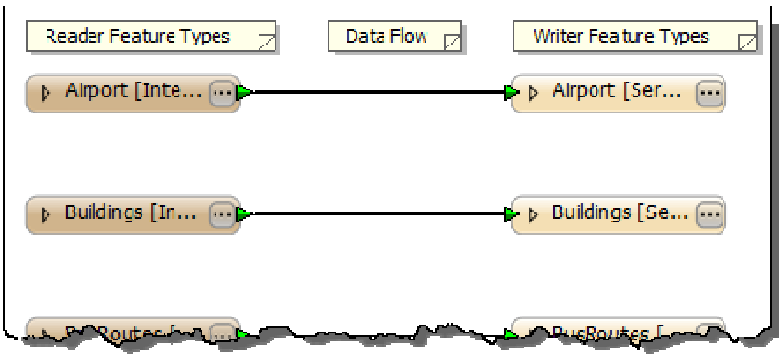
Reader Format: *Autodesk MapGuide Enterprise SDF*
Reader Dataset: *C:\FMEData\Data\DemoData\InteropolisDatabase.sdf*

Reader Parameters
Remove Schema Qualifier Yes

Writer Format: *Generic (Any Format)*
Writer Dataset: *C:\FMEData\Output\ServerTraining*

When prompted, leave all feature types selected to add them to the workspace.

***NB**: If all feature types are prefixed with the word "Default" – for example Default.Airport – then you neglected to check the reader parameter Remove Schema Qualifier.*



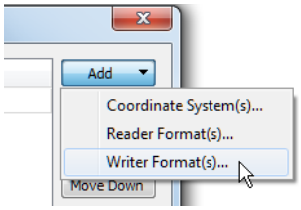
2) Remove Published Parameters
As in previous examples, remove any existing published parameters.

3) Create Custom Parameter
As in previous examples, create a new published parameter for format selection.

Set the options as follows:

Type	Choice with Alias
Name	OUTPUT_FORMAT
Published	Yes
Optional	No
Prompt	Output Format:

In the configuration dialog, again click the Add button and select Writer Formats:



This time search for, and then add, the following formats:

ACAD	Autodesk AutoCAD DWG/DXF
GML	GML (Geography Markup Language)
MAPINFO	MapInfo TAB (MFAL)
OGCKML	KML
PDF2D	Adobe Geospatial PDF
SHAPE	Esri Shape

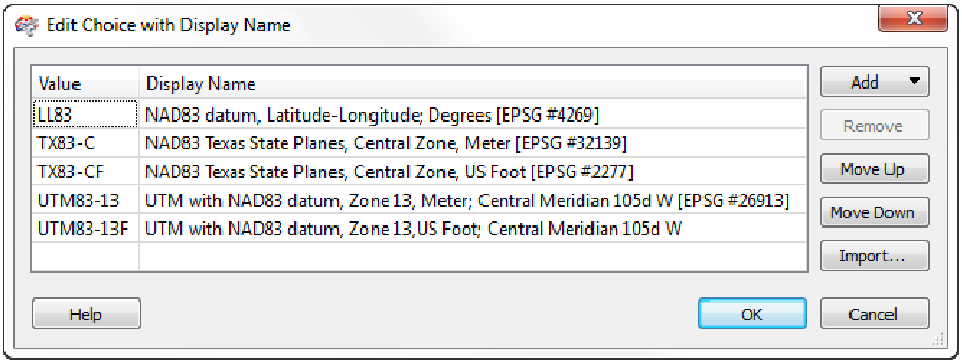
Once more, link this user parameter to the FME parameter by right-clicking ServerTraining [GENERIC] > Parameters > Output Format, choosing Link to User Parameter, and selecting OUTPUT_FORMAT from the list provided, as the user parameter to link to.

4) Create Custom Parameter
Create a second custom parameter. This time set the options to:

Type	Choice with Alias
Name	OUTPUT_COORD
Published	Yes
Optional	No
Prompt	Output Coordinate System:

In the configuration dialog, this time select Add > Coordinate System(s) from the menu. Search for and add the following coordinate systems:

LL83
TX83-C
TX83-CF
UTM83-13
UTM83-13F



5) Link Custom Parameter
Now the coordinate system user parameter must be linked to the appropriate FME parameter.

On the Navigator, locate the writer parameter ServerTraining [GENERIC] > Coordinate System.

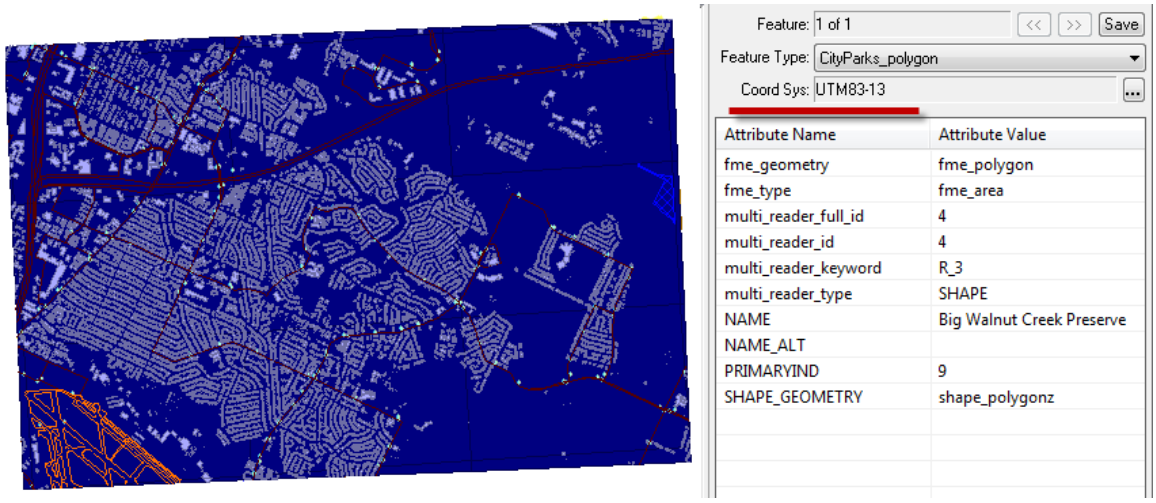
Right-click the parameter and choose Link to User Parameter. Choose OUTPUT_COORD from the subsequent dialog and click **OK**.

6) Run Workspace on FME Desktop
Save the workspace. It is now ready to test using Prompt and Run.

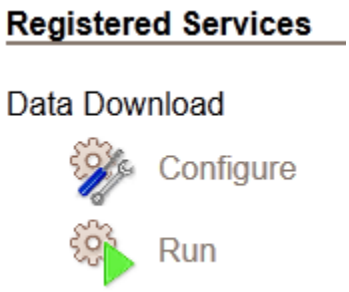
Select **File** > **Prompt and Run Translation** from the menubar. The translation parameters dialog will open. Select an output format and coordinate system and click **OK** to run the workspace.

Check the output folder (C:\FMEData\Output\ServerTraining) and inspect the datasets to ensure the translation is correct.

***Hint**: Coordinate system can be inspected in the Coord Sys window in the FME Universal Viewer:*



7) Publish to FME Server
Use File > Publish to FME Server to publish the workspace. Publish it to the Training repository and register it with the **Data Download** service.



8) Run Workspace on FME Server
Open the FME Server web interface and navigate to the published workspace. Select the Data Download > Configure option.

Once more select a format and coordinate system and run the workspace.

The translation will run and, when complete, display a link from which to download the output.

Advanced Task
As an advanced task, create a web form that allows the user to select the different values for the various published parameters, and then run the workspace.

The KML output format is not compatible with the Coordinate System parameter, because KML can only be output in a Lat-Long coordinate system; so make the form disable the coordinate system setting when the format parameter is set to KML.

Layer Selection and Handling

Layer (Feature Type) flexibility can be achieved through the Feature Types to Read parameter.

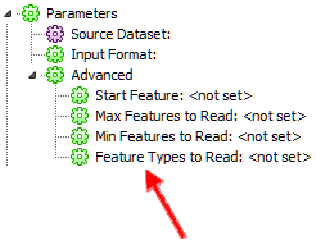
There is no single solution flexible enough to meet every possible method by which users might wish to select data or features. Every organization implementing FME Server lets users select data in different ways, and the use of multiple methods is probably the way to be most flexible.

Spatial data is often organized in layers (groups, classes, categories, feature types) and one way users will wish to choose data is on a layer-basis.

Selecting Layers

Allowing users to select data by layer is the simplest solution available in an FME Workspace, although in many situations users will wish to select data not just by layer, but by a mix of criteria.

Allowing choice by layer alone can be handled easily if the layers correspond to Feature Types in the source data. Each reader in FME has a parameter called Feature Types to Read.



When the parameter is published, the workspace will accept a list of layers as a parameter. This parameter can be linked to a Multiple Choice with Alias custom parameter, which in itself can be populated with a list of clearly-named layers.

Grouped Layers

On some occasions the user does not need to be presented with a full list of layers. Instead they might need to see “groups” of layers.

For example, what is presented to the end-user as “Water” features may actually be made up of multiple source feature types such as Lakes, Rivers, and Streams.

This type of scenario can be achieved in an FME Server workspace using scripted parameters.

Scripted Parameters

Scripted parameters are a way to incorporate a Python or Tcl script into the generation of a parameter value. This script can include references to other user parameters, meaning one parameter value can be calculated based on the value of another.

Scripted parameters are always private parameters because the parameter is set by the script not by the user. Parameters referred to in the scripts are generally published parameters that accept input from the user.

So, user input can be accepted from one parameter, and this input processed using a script, in order to produce the actual information required by the workspace.

Example 8: Feature Types to Read	
Scenario	FME user; City of Interopolis, Planning Department
Data	Interopolis Database
Overall Goal	Create a data download service
Demonstrates	Feature Types to Read
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example8Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example8Complete.fmw C:\FMEData\Workspaces\ServerManual\Example8AdvancedComplete.fmw

This example continues to develop the workspace from the previous example. Now it adds the ability for the user to select which layers to read from the source data.

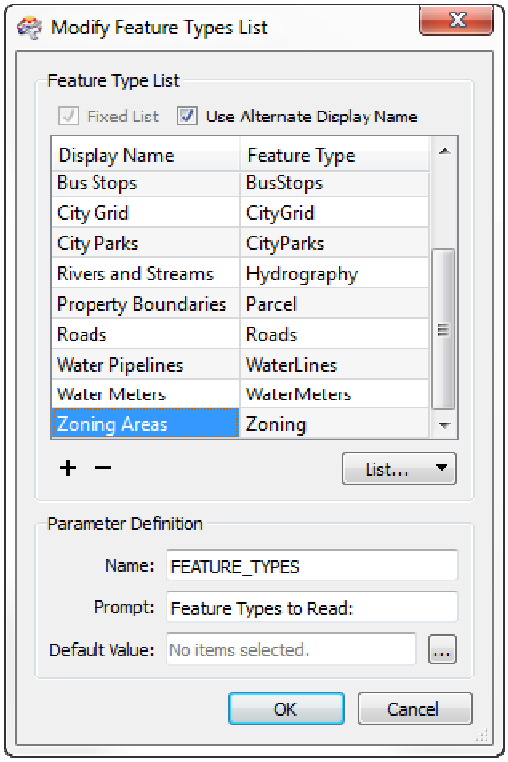
1) Start Workbench
Start Workbench (if necessary) and open either the workspace from the previous example or the Begin workspace for this example.

2) Publish Feature Types to Read
Locate the Feature Types to Read parameter for the SDF reader. Right-click and choose Create User Parameter.

Notice that this particular parameter has its own unique dialog box. This is so the list of layers can be manipulated manually.

Check the box for “Use Alternate Display Name” and enter some suitable names for the different feature types; for example “Bus Routes” instead of “BusRoutes”.

Click **OK** to accept the definition.



3) Run Workspace on FME Desktop
Save the workspace and run it within FME Workbench to prove that the Feature Types to Read parameter works correctly.

If the parameter is working as it should, only layers selected here should appear in the output dataset. The feature counts in the workspace will also indicate the success of this operation.

Advanced Task
The feature types in the workspace might be considered in groups as follows:

CityParks	Environment	Airport	Transportation
Hydrography		Roads	
Buildings	Properties	WaterLines	Utilities
Parcel		WaterMeters	
BusStops	Public Transit	CityGrid	Civic Info
BusRoutes		Zoning	

If this were the case then the user can be presented with a list of groups instead of each layer individually. This can be achieved by using the same Alternate Display Name, for each of the Feature Types to Read.

1) Modify Published Parameter
Modify the Feature Types to Read parameter to use Environment, Properties, etc as the Display Name, instead of the original feature type names.

2) Run Workspace
Save and run the workspace – both on Desktop and Server – to prove that it works correctly.

Data Streaming Service

Data Streaming services return the results of an FME workspace directly into a supported application.

What is the Data Streaming Service?
The main difference between a data streaming service and a data download service is that a data download service returns an HTML, XML, or JSON format that contains a link to the data. A data streaming service returns the data itself.

Therefore the URL for the service can be posted into a web browser, and the data will be automatically downloaded and opened in the application associated with that file type.

Alternatively, the URL can be used directly as the source for a client application. 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.

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.

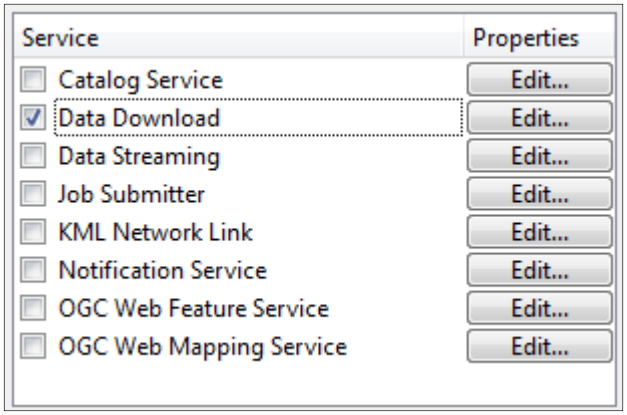
What Formats can be Streamed?
You can use any workspace with the data streaming service, provided it writes data in a format that is file-based or folder-based.

If an output dataset is comprised of more than one file, the service automatically creates a compressed 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
- JSON
- GeoJSON
- KML
- PDF
- HTML

Creating a Data Streaming Service
As with other services, a workspace becomes available for data streaming use when the author registers it against the Data Streaming service when publishing it to FME Server.



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.

Workspaces can store the MIME type of the output datasets and transmit that information to the Data Streaming service.

Therefore, the end-user's system will know what sort of action to take with the streamed file – for example, open a KML file in Google Earth – without being prompted by the end-user.

You can set only one MIME type per workspace. Allowing more than one MIME type to be set does not make sense because only one file should be returned when using the Data Streaming service. Otherwise, the data is zipped and the MIME type is not used.

KML Network Link Service

The KML Network Link service is a special service whose output is intended solely for use in a KML browser, such as Google Earth.

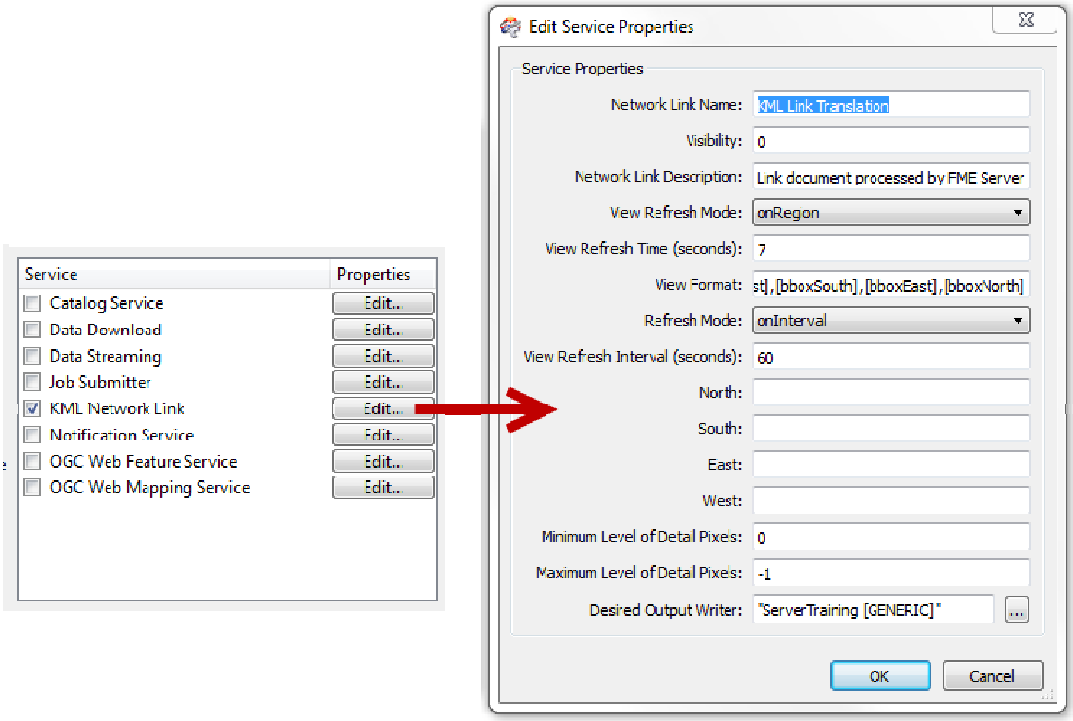
What is the KML Network Link Service?
A KML Network Link service is when the output from an FME Server translation is a simple KMZ (zipped KML) file that contains solely a network link to reference the workspace on the server.

Whenever a KML browser (such as Google Earth) opens this dataset it follows the link, triggering FME Server to run the translation and return the true output 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 the browser, which writes it to a cache.

Creating a KML Network Link Service
As with other services, a workspace becomes available for use as a KML Network Link service when the author checks the KML Network Link checkbox in the FME Server Publishing Wizard. The edit button provides a whole series of different parameters that control the refresh rate when the data is opened in Google Earth.



Example 9: Data Streaming and KML Network Links	
Scenario	FME user; City of Interopolis, Planning Department
Data	Land parcels (MapInfo MIF/MID)
Overall Goal	Deliver property boundaries via data streaming to KML
Demonstrates	Data Streaming, KML Network Links
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example9Complete.fmw

This exercise starts out as a simple requirement to stream property polygon boundaries to KML.

1) Start Workbench
Start Workbench (if necessary) and begin with a blank workspace.
Use Readers > Add Reader to add a reader as follows:

Reader Format: MapInfo MIF/MID
Reader Dataset: *C:\FMEData\Data\Properties\parcel_K24.mif*

Workflow Options Single Merged Feature Type

The merged feature type is important here because there are multiple source files, each with a different name, and we want to set up the workspace to read any of them.

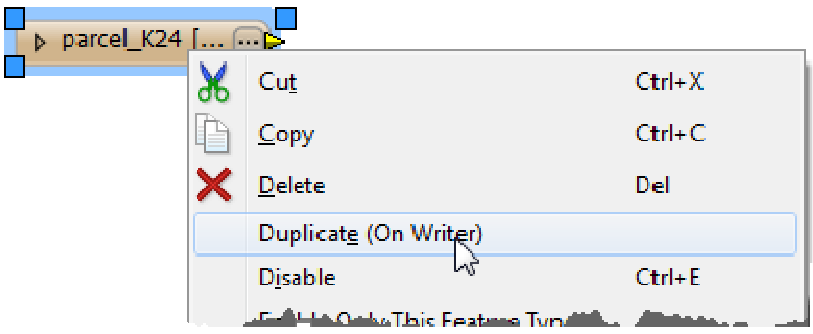
2) Add Writer
Now use Writers > Add Writer to add a writer as follows:

Writer Format: Google Earth KML
Writer Dataset: *C:\FMEData\Output\ServerTraining\Properties.kml*

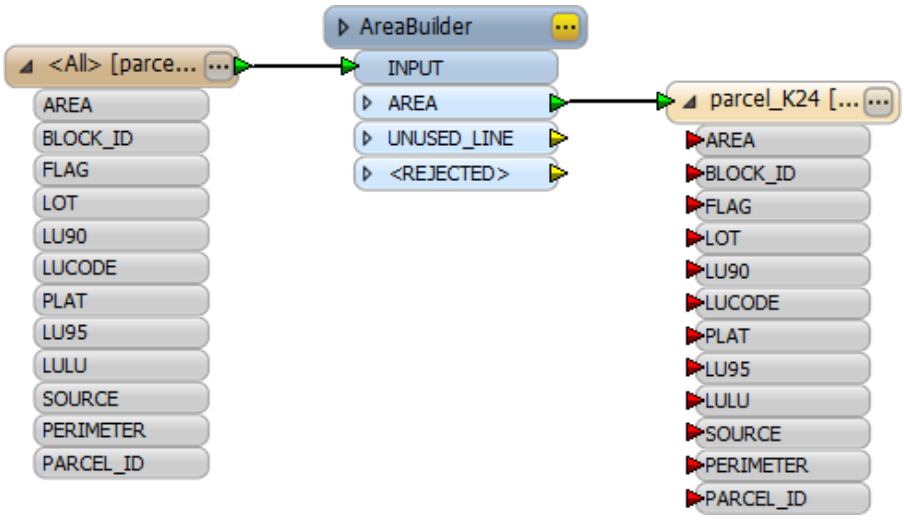
Workflow Options Static Schema

When prompted to add a Feature Type to the workspace, choose No.

3) Add Writer Feature Type
Right-click on the Reader feature type and choose Duplicate (On Writer):



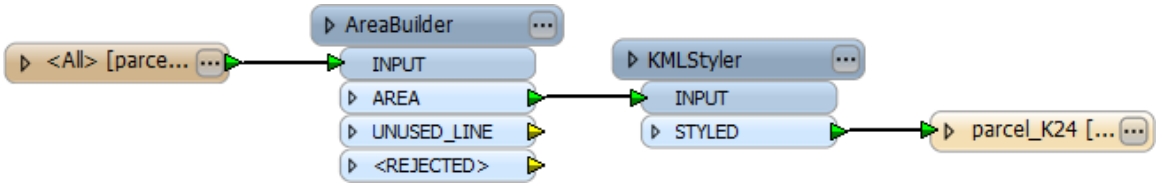
4) Add AreaBuilder Transformer
Add an AreaBuilder transformer between the reader and writer feature types. The purpose of this transformer is to turn line features in the source data into polygon features in the output.



5) Set AreaBuilder Parameters
Open the AreaBuilder parameters dialog. Turn the parameter for Snapping Type to “End Point Snapping” and set the snapping tolerance to 5. The units here are in feet.

Leave the remaining parameters as they are and click **OK** to exit the dialog.

6) Add KMLStyler Transformer
Optionally add a KMLStyler transformer to provide outline and fill colors to the data.



7) Check Published Parameters
Check the published parameters. Notice there are parameters for the source and destination dataset and for Feature Types to Read.

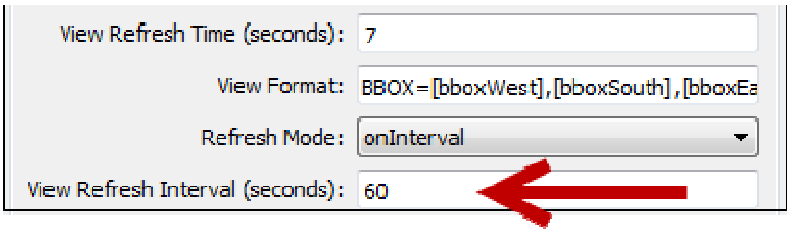
Because this is a streaming workspace – i.e. writing directly to an application – we aren’t going to need the destination dataset published parameter, so delete it. Also delete “Feature Types to Read” because there are no separate layers inside a MIF file.

Leave the published parameter for source dataset because we wish the user to be able to select which tile of data is to be translated.

8) Run Workspace on FME Desktop
As usual, run the workspace locally to ensure it operates correctly.

9) Publish to FME Server
Save the workspace and then publish it to FME Server.

In the Register Services dialog, register the workspace with the Data Streaming and KML Network Link services. Then click the Edit button next to KML Network Link. Change the View Refresh Interval parameter to 10. This will ensure the data is refreshed every 10 seconds.



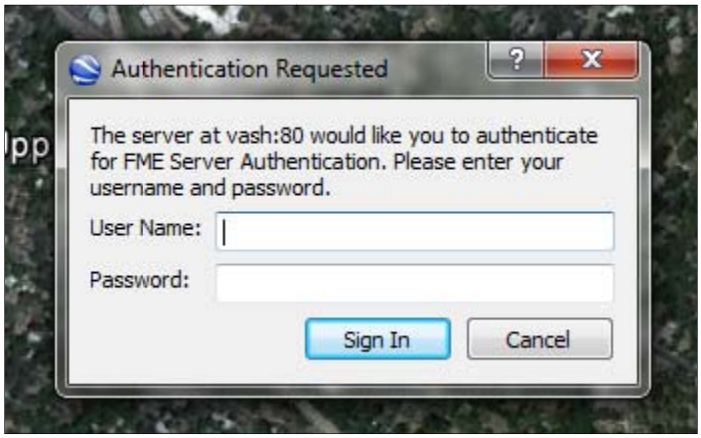
10) Run Workspace on FME Server - 1
Open the FME Server web interface and browse to the workspace. Choose the Data Streaming Run option. The workspace will run and the data is opened in Google Earth (or whichever application is the default for KML datasets)

11) Run Workspace on FME Server - 2
Now browse to the workspace and run it using the KML Network Link run option.

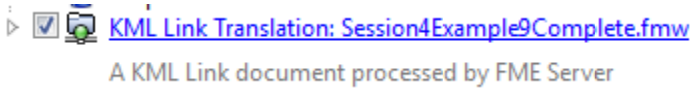
Again the data will open in Google Earth, but is not turned on by default. Check the box in the Places window to activate it.

This time it will ask for a username and password. This is because the workspace needs to be run and therefore authenticated.

Enter your FME Server user name and password and then click **Sign In.**



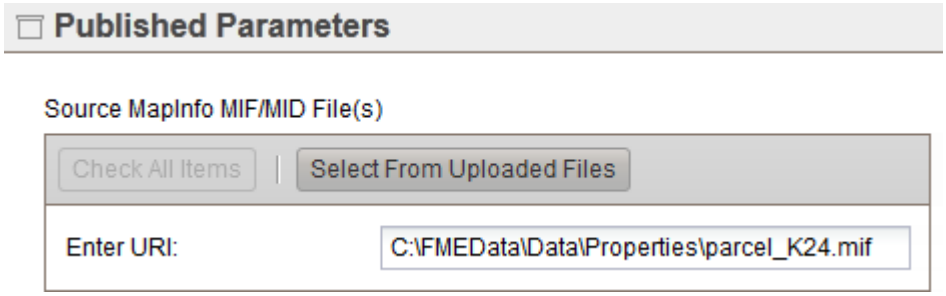
The data is now retrieved and the Places window shows the network link.



The icon on this link will update momentarily as the data is updated every ten seconds. Check the Jobs History on FME Server. You will see the job is being run every time Google Earth requests an update:

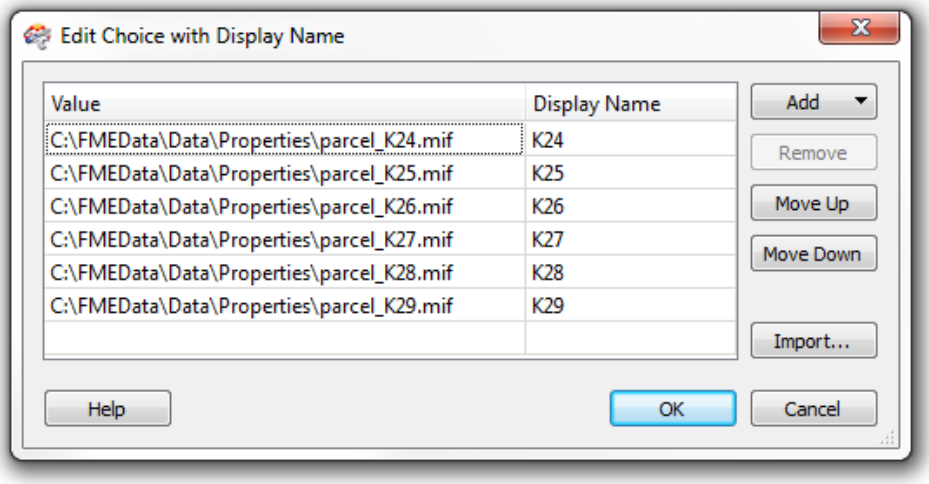
Time Started
2012-12-11 14:11:06
2012-12-11 14:10:56
2012-12-11 14:10:46
2012-12-11 14:10:35
2012-12-11 14:10:25
2012-12-11 14:10:15
2012-12-11 14:10:04

Advanced Task
If you use the Configure option in FME Server for this workspace you'll notice that the Source dataset parameter is a text-entry field.



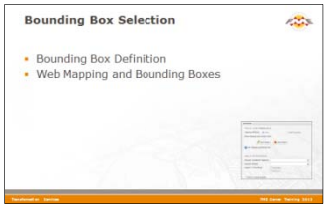
It would be rather annoying to the user to have to type in a file name each time. So, as an advanced task, try to improve this file-selection method.

The simplest way to do this would be to use choice with alias parameter, where the choice of tile name is replaced by the matching file. This could be used in a scripted parameter which is then linked to the FME source dataset parameter.



You might also want to try using the Choice with Alias (Multiple) parameter to select multiple tiles. In that scenario you will definitely need to use a scripted parameter to turn the list of choices into a string that the source dataset parameter will understand.

Bounding Box Selection

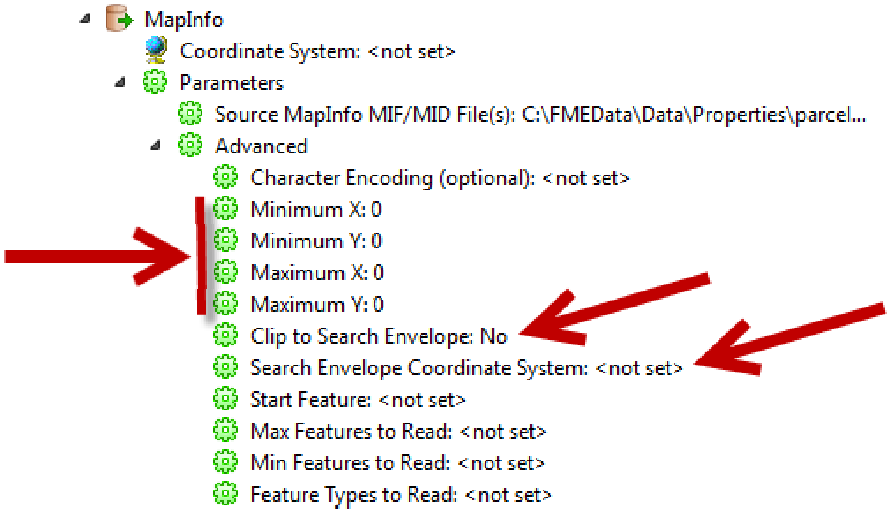


Bounding Boxes provide a way to limit the data being read and processed

Although most workspaces are designed around FME feature types, this may not be how the end-users see the data, or how they need to select it. An installation that provided selection only by layer (or even a group of layers) would be very limiting indeed.

Besides layers, other methods of data selection revolve around spatially defining an area of interest. A bounding box is the simplest way to spatially define such an area.

Bounding Box Definition
All FME readers have a set of parameters to define the bounding box (envelope) of data that is being read. This is the most efficient method of selecting data, particularly if the source format has a spatial index, and applies to both vector and raster datasets.



If these search envelope parameters are published, then the end-user is able to enter values that define the extent of the data he/she is interested in.

Besides the actual coordinates, there are two related parameters: “Clip to Search Envelope” and “Search Envelope Coordinate System”.

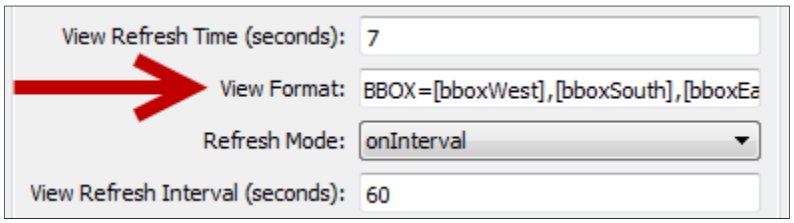
Clip to Search Envelope causes features that overlap the bounding box to be clipped and the outer part discarded. This will be the most common use for FME Server.

The Search Envelope Coordinate System means that the bounding box can have its own coordinate system, separate from the reader coordinate system. So the bounding box coordinates can be defined in a general coordinate system like Lat./Long, even if the data is UTM.

Web Mapping and Bounding Boxes
Bounding box parameters are particularly useful when a web mapping application (for example Google Earth) is requesting data via a Data Streaming service. The workspace can restrict itself to features within the current web map view, provided the application supplies this information.

One example of this is a Google Earth KML network link. To retrieve only the data that is necessary, the network link takes the bounding box of the current view in Google Earth and passes it to the workspace.

In fact, when you published the workspace in the previous example, you might have noticed how Google Earth passes these back:



Of course, for this to work requires the workspace to contain these exact parameters:

- bboxWest
- bboxEast
- bboxNorth
- bboxSouth

In fact, WFS and WMS services also require the presence of these bounding box parameters in the workspace, if they are to operate efficiently.

Ms. Analyst says…

"You might previously have used a Clipper transformer to restrict an area of interest, but that is not the most efficient method. Using reader parameters ensures the data is reduced to size at the earliest possible opportunity and therefore gives a better performance."

Example 10: Clipping to a Bounding Box	
Scenario	FME user; City of Interopolis, Planning Department
Data	Land parcels (MapInfo MIF/MID)
Overall Goal	Deliver property boundaries via data streaming to KML
Demonstrates	Data Streaming, KML Network Links
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Session5Example10Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Session5Example10Complete.fmw

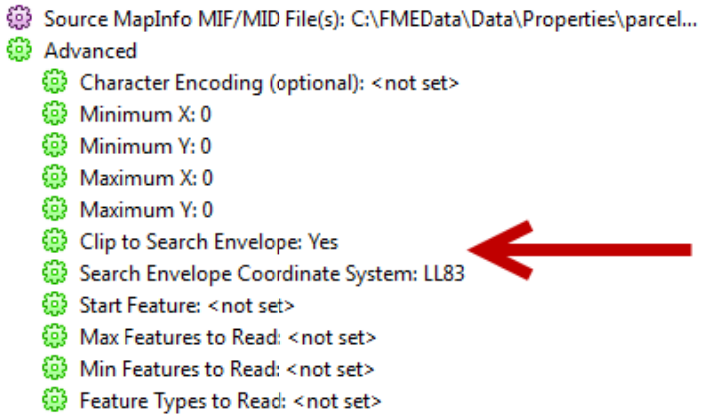
1) Open the Workspace
Start Workbench if needed, and open the workspace from the previous example, or use the Starting Workspace as provided.

2) Parameterize Search Envelope - 1
Locate the search envelope parameters for the reader.

Manually set “Clip to Search Envelope” to Yes, and “Search Envelope Coordinate System” to LL83.

These are parameters that need to be hard-coded and not be available to the user.

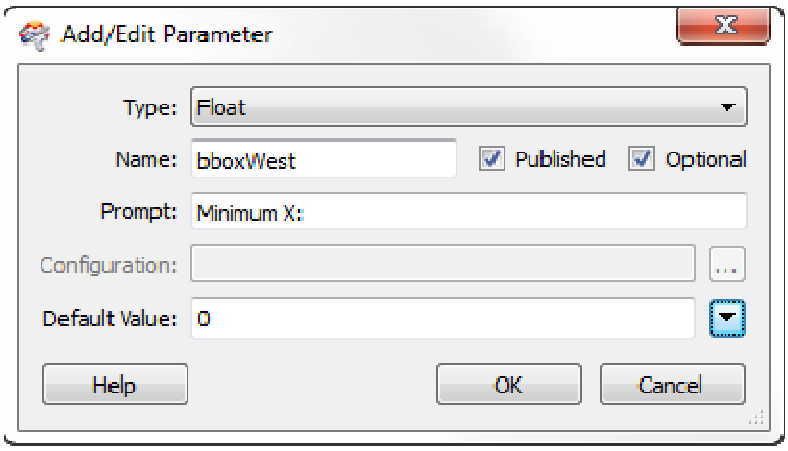
Also set the Reader’s Coordinate System to LL83, as this is required for the clip coordinate system to work.



3) Parameterize Search Envelope - 2
Publish the four parameters for minimum X, minimum Y, maximum X, and maximum Y. Ensure they have the same names as Google Earth will use to provide values (bboxWest, bboxSouth, bboxEast, bboxNorth).

By default the parameters will be given the type “Float”, which is correct for these values.

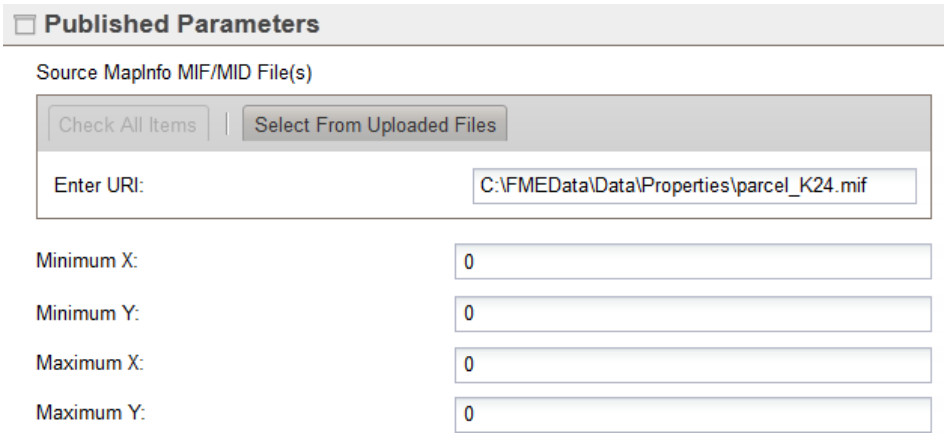
NB: *In the release version of FME 2013, you will also need to create a parameter called BBOX. This bug is fixed in 2013-SP1 or newer.*



4) Save and Publish Workspace
Save the workspace and publish it to FME Server. Register it with the Data Streaming and KML Network Link services.

5) Run Workspace on FME Server
Locate the workspace on FME Server in the training repository. Choose Data Streaming > Configure.

Notice how the bounding box parameters are now made available for the end user to restrict the data being read. A web mapping application might use this by having the user draw a bounding box, and then passing the extents of that box to these parameters.

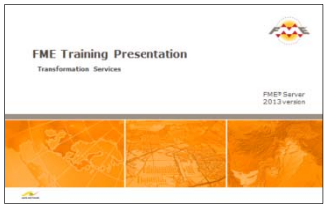


Try running the workspace. If all parameters are left to zero, then ALL of the data will be returned.

6) Run Workspace on FME Server - 2
Now run the workspace on FME Server as a KML Network Link. When the data is opened in Google Earth, zoom in and pan around the data for 20-30 seconds while the data is reloaded.

Check the Job History in FME Server. Click on one of the jobs to examine its request from Google Earth. The request will include the clip envelope being passed to FME from Google Earth.

Module Review



This module introduced you to FME Server's Transformation and OGC Services.

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
- OGC Services are used for OpenGIS standard outputs
- Transformation Services are used for delivering output data to the end user in a number of different ways
- 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

FME Skills

- The ability to design a workspace for use with, and register it to, the FME Server Data Download service
- The ability to use the Feature Types to Read parameter for layer selection
- The ability to design a workspace for use with, and register it to, the FME Server Data Streaming service
- The ability to use Bounding Box parameters for area selection

Utility Services

Utility Services are management functions that aren't involved in data distribution

Whereas Transformation and OGC Services are primarily ways for Server to deliver data to the end user, Utility Services provide functionality to send data and queries to FME Server. These services basically provide programmatic access to FME Server for carrying out administrative tasks such as managing users or retrieving information about the contents of a repository.

Data Upload Service
The Data Upload Service provides a method for uploading resources used by a workspace. These resources are primarily source datasets, but can also include custom formats, custom transformers, and other similar supplementary files.

The Data Upload service can be used by either the workspace author or the end-user in order to provide the required workspace resources, although only the author has management access to the datasets on the server.

Catalog Service
The Catalog Service allows creation of an inventory of resources on an FME Server. It is a way for FME authors to be given access to these resources through a web page or application.

Token Service
The Token Service is for generating security credentials that can be used as proof of access rights instead of a username and password. This way the end user has a single sign-on for all aspects of FME Server and does not need to enter their password repeatedly.

REST Service
The REST Service is an API that allows access to FME Server through RESTful HTTP requests. The service allows programmatic access to most of FME Server's functionality, including the ability to run a workspace, manage job queues, and schedule requests.

Because this service allows programmatic access to the list of repositories, the list of workspaces within a repository, and the list of published parameters for any workspace in a repository, it is frequently used by web developers to create custom pages and forms whose content is derived dynamically from queries to FME Server.

For example, a developer would be able to create *ad hoc* forms containing a list of workspace parameters, without knowing in advance what workspaces exist or what parameters they contain.

Web Connection (SOAP Communication) Service
The Web Connection Service provides a mechanism for connecting to FME Server through the Simple Object Access Protocol (SOAP). This, as an example, is how FME Workbench connects to and communicates with FME Server.

The Web Connection Service is intended for use by developers and is not a user-oriented service.

Data Upload Service

The FME Server web interface allows the end-users of a workspace to upload data to FME Server.

When the source data for a workspace does not ordinarily lie within a place accessible to the FME Server (either on a local disk or on another computer via a UNC path) then the data must be supplied by an upload to the server.

Data can be uploaded by either the workspace author or the end user. Uploaded data is stored in a temporary location in the repository file system on FME Server and removed automatically after a set period of time.

Author Uploads

If source datasets are not to be provided by the end-user, then they must be provided by the workspace author when publishing the workspace.

When to Upload Data

Uploading data *is* useful in the following scenarios:

- When small file-based resources are used in the workspace (e.g. by transformers)
- When shared paths or drives aren't available

When Not to Upload Data

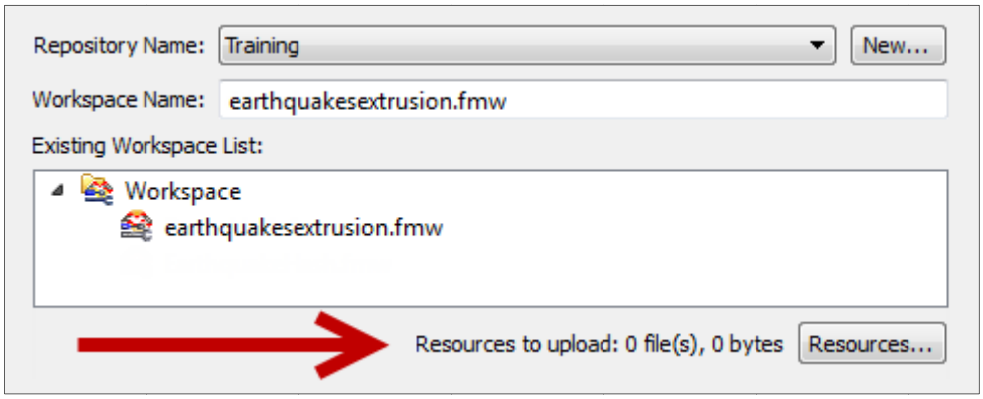
Uploading data is *not* useful in the following scenarios:

- When large file-based datasets are being used
- When the resources are meant to be fixed and unchanging

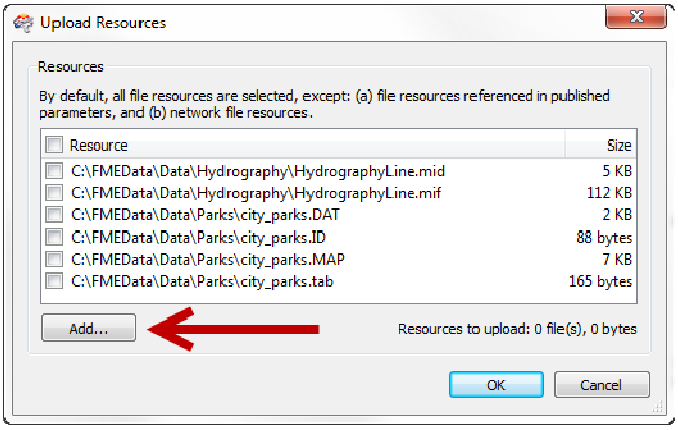
Selecting Data to Upload

As mentioned, whenever a workspace is published to FME Server from Workbench there is always an option to upload the source data – and any other required files – with it.

The option appears on the dialog for repository selection and workspace naming:



By default, all source datasets are listed in the Upload Resources dialog (although not necessarily selected) but the author can add more files using the Add button:

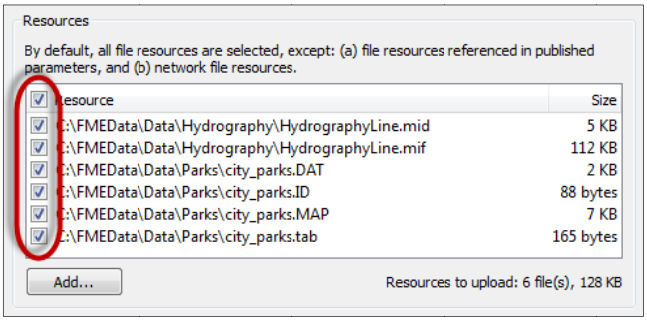


Author Uploads and Published Parameters
Although the Upload Resources dialog lists all source resources, whether they are also selected depends upon whether the Source Dataset parameters are published or not.

When Dataset Parameters Are Published
When dataset parameters ARE published then datasets are not automatically selected for upload. It's assumed that the data will be available to FME Server anyway.

But, even if the datasets listed are not available, the workspace is still capable of being run because the published parameters allow the end-user to upload their own data.

When Dataset Parameters Are Not Published
When dataset parameters are NOT published (in which case the workspace author must have made the decision to remove them) then datasets ARE automatically selected for upload.



The reasoning for this behaviour is that, since the end user can't supply the data, the author must.

Of course, the author can choose to uncheck the box and not upload that file, provided the source data is going to be made available in another manner; for example on a shared directory.

Custom Resources
Resources such as Custom Transformers and Custom Formats are not uploaded in the same way as other resources. They need to be properly installed by opening them in FME Workbench and publishing them to FME Server – just as if they were a workspace.

Example 11: Author Uploads	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property Boundaries (KML)
Overall Goal	Create an online service for converting KML datasets
Demonstrates	Data Upload by Author
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example11Complete.fmw

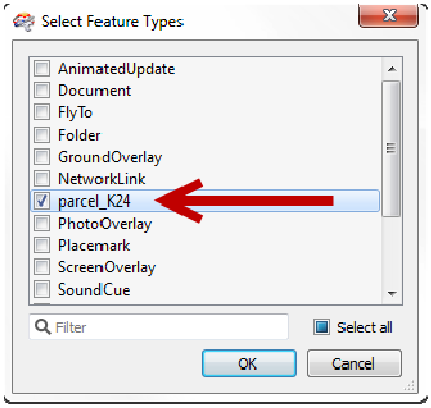
The project here is to create an online service for the planning department to convert KML datasets. Initially the service is to convert a KML dataset of property boundaries.

As a first step the author decides to create a fixed workspace that simply reads a single KML. He will upload the data when the workspace is published, and remove the published parameters as the end-user is not expected to supply their own data.

1) Start Workbench
Start Workbench if necessary and use the Generate Workspace dialog to create a translation with the following parameters:

Reader Format Google Earth KML
Reader Dataset *C:\FMEData\Data\Properties\Properties.kml*

Writer Format Generic (Any Format)
Writer Dataset *C:\FMEData\Output\ServerTraining*

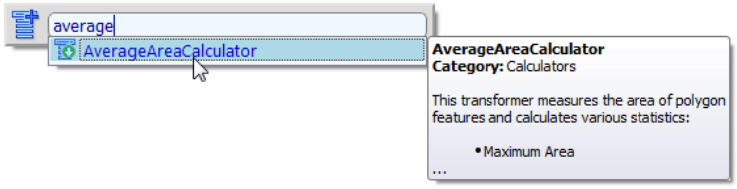


When prompted, the only Feature Type to include is "parcel_K24", the layer on which the data is stored:

If any excess Feature Types appear in the workspace (either Reader or Writer) they can be deleted.

2) Add AverageAreaCalculator Transformer
The planning department need to calculate the average size of each property in the dataset. Luckily a Custom Transformer exists on the FME Store to carry out this exact task.

Use Quick Add to add an AverageAreaCalculator transformer.



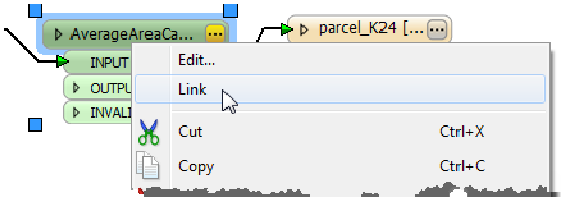
It will be automatically downloaded from the FME Store and added to the workspace:

3) Add AverageAreaCalculator Transformer

Connect the AverageAreaCalculator transformer between the Reader and Writer feature types.

Right-click the transformer and choose the option “Link”.

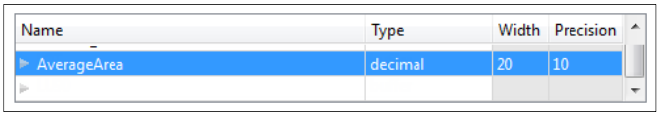
This will make the transformer a linked (rather than embedded) version.

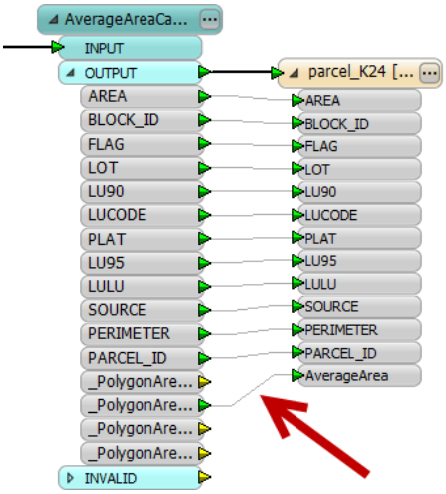


4) Update Schema

The custom transformer creates a number of different attributes but, at the moment, the schema does not permit these to be written out.

Add a new attribute to the writer Feature Type schema called AverageArea. It should be of type “decimal”

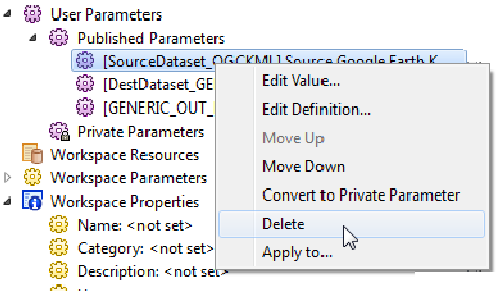




Connect _PolygonAreaAverage from the AverageAreaCalculator to this new attribute:

5) Delete Published Parameters

The end-user does not need to specify the source dataset, nor – since this will be a Data Download service – do they need to specify the output location.



Therefore, use the Navigator window to delete the source and destination dataset published parameters:

Leave the Generic Output Format parameter in place.

6) Publish Workspace to FME Server
Now save the workspace and then publish it to FME Server.

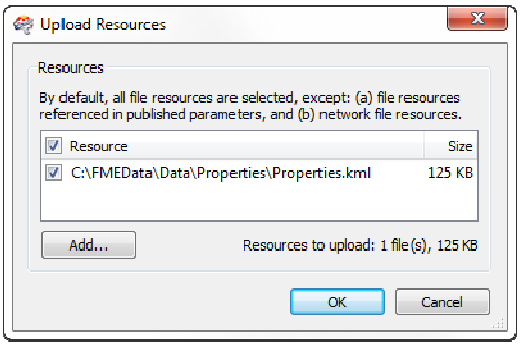
The first dialog will be a warning that the Linked custom transformer will need to be published too. The important part to note is that it needs to be in the same repository as the workspace. Click **Yes** to continue.

In the Repository selection dialog, select the Training repository.

Notice that the "Resource to Upload" label already shows 1 file selected. That's because we removed the published parameter and so the KML source data is automatically selected.

Register the workspace with the Data Download service and click Publish to complete publishing it to FME Server.

Notice that the log window reports not just the workspace that was published, but also the KML dataset that was uploaded too.



7) Publish Custom Transformer to FME Server
Now open the Custom Transformer in Workbench. You can do this easily enough by using Right Click > Edit on the instance of the transformer in the current workspace.

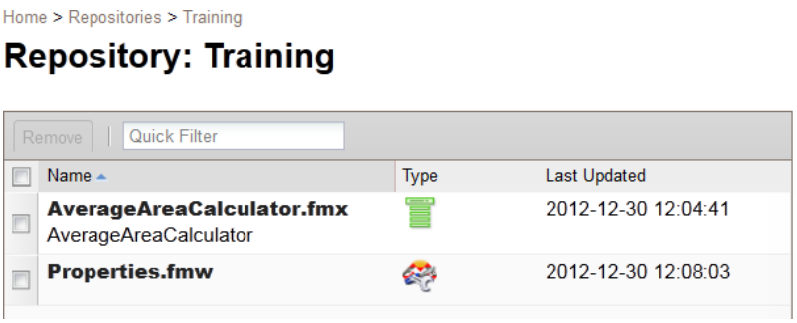
Now publish the Custom Transformer to FME Server. Be sure to use the same repository (Training) as was used in the workspace.

The only service it should – or can – be registered with is the Catalog Service. A custom transformer like this cannot be used with any Transformation Service and so they will not be shown.

Although it seems redundant to upload the AverageAreaCalculator, when it could be used embedded into the workspace instead, there is a benefit to doing so. Once the transformer is published it can be used in any other workspaces that need it, without being either embedded or re-published.

8) Locate Workspace
In the role of user, open the FME Server web interface and browse to the Training repository.

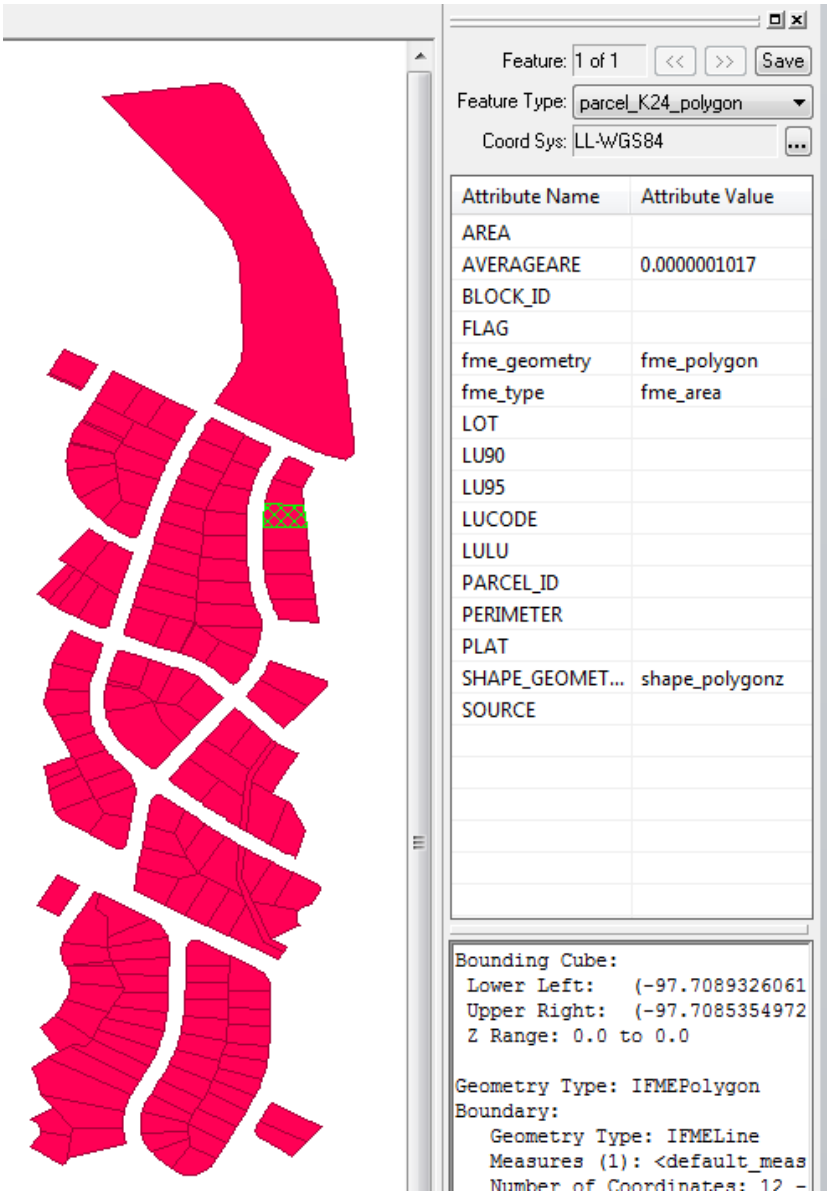
You will see both the workspace and its custom transformer:



9) Run Workspace
Choose Data Download > Configure and manually enter SHAPE as the format to output.
Notice that there are no parameters in which to enter/upload a source dataset.

Click **Run Workspace**.
The workspace will be run and a link provided through which to download the output.

Click on the Data Download URL to download the data. Open it for inspection in the FME Universal Viewer. You should find the data is a series of polygons, each of which possesses an attribute called AverageArea (or AVERAGEARE, since the data was written to Shape).



User Uploads

When a workspace author wishes for the source datasets for a translation to be provided by the end user, then the user must have a mechanism for providing the data to FME Server.

This mechanism is for the end-user to upload the source data when running the workspace.

When to Upload Data

User uploads *are* useful in the following scenarios:

- When only the user possesses the data; for example when uploading data to be examined for quality assurance or data validation
- When end-users have more up-to-date datasets than the workspace author; for example the user is contributing data to a data warehouse or store.

When Not to Upload Data

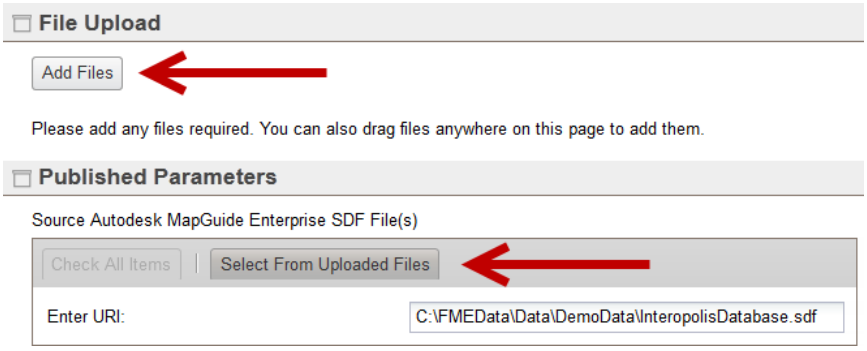
User uploads are *not* useful in the following scenarios:

- When the source data is fixed and the workspace is only ever intended to be used for that particular dataset.

User Uploads and Published Parameters

Whenever a workspace published to FME Server has a source dataset published parameter, it's assumed that the user will supply the data.

In the FME Server web interface, the user can upload data in the configuration dialog for any workspace. They can either use the Add Files button to open a file selection dialog, or can drag-and-drop files directly onto the interface.



Once files have been uploaded they can be selected in any published parameter that accepts filenames as input.

Of course, the author can still choose to upload data, as a standby or default for when the user does not wish to supply files.

If the end-user is required – or allowed – to supply data, then the published parameters in the workspace should be left in place. A parameter to prompt the user for source data will allow them to upload files for that purpose.

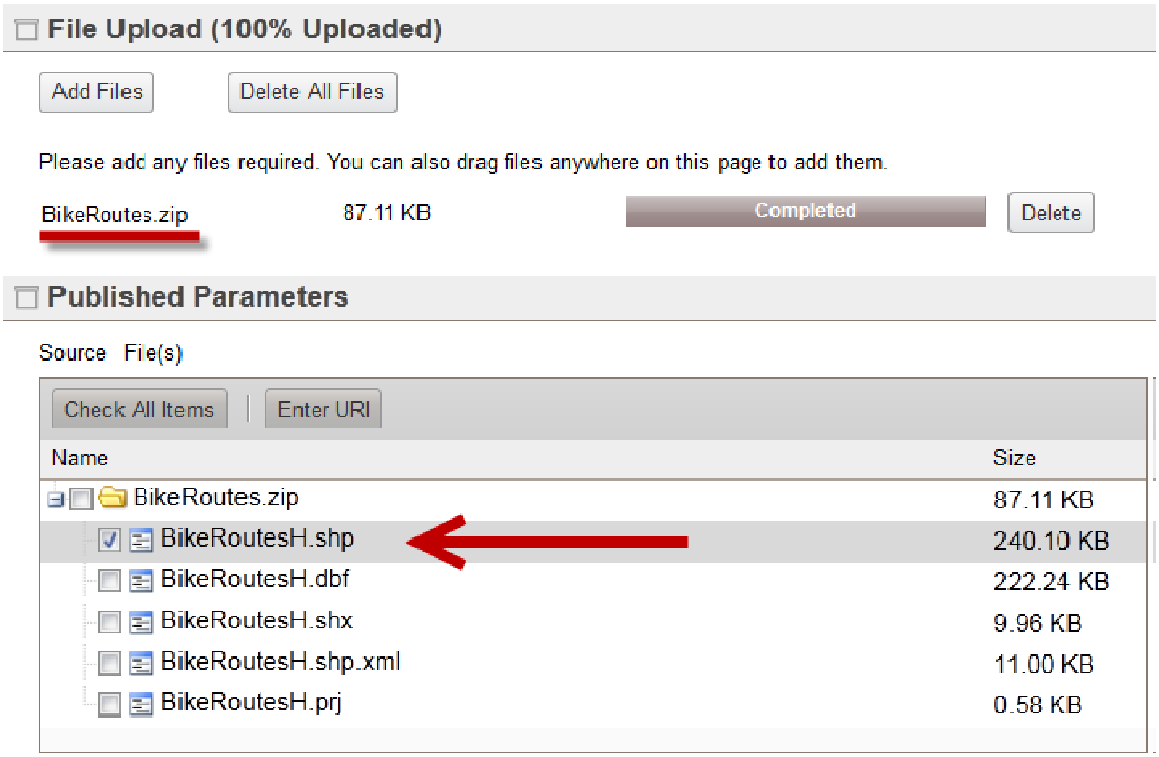
Remember, when the end-user is NOT required to supply data, then the published parameters in the workspace should be removed. If there are no parameters to prompt the user to supply data, then they will not be able to do so.

Multi-File Datasets
It's critical that the user makes the full source dataset accessible to FME Server. Some datasets are made up of multiple files, and the user must supply them all.

If not all of the required files are uploaded, the workspace will not run.

For example, if the source is a Shape dataset, the end-user must upload **all** files that make up the dataset; that is, ***.shp***, ***.shx***, ***.dbf***, and (optionally) ***.prj*** files

The easiest method is to upload the entire set of files in a zip file. Then the individual source dataset can be selected from within the zip file.



Example 12: User Uploads	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property Boundaries (KML)
Overall Goal	Create an online service for converting KML datasets
Demonstrates	Data Upload by End User
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example12Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example12Complete.fmw C:\FMEData\Workspaces\ServerManual\Example12AdvancedComplete.fmw

Example 11 must be completed in order to carry out Examples 12, 13, and 14; otherwise the Custom Transformer is not available on the FME Server.
If you haven't uploaded the Transformer, be sure to right-click on it and choose embed so that an external copy is not required.

The planning Department is pleased with the previous workspace, but now wish to upload the KML dataset themselves so they can always run the workspace on the latest data.

To do this simply requires user uploads to be activated for the workspace.

1) Start Workbench
Start Workbench and open the workspace from the previous example (or open the pre-defined Starting Workspace).

2) Publish Dataset Parameter
To give the users the option to upload their own data, all that needs to be done is publish the Source Dataset parameter.

In the Navigator window, locate the parameter called Source Google Earth KML File or URL and right-click on it. Choose the option to Create User Parameter.

In the Add/Edit parameter dialog, simply click **OK** to accept the default values.

3) Publish Workspace
Publish the workspace back to FME Server (File > Republish may be enough).

4) Run Workspace
In the FME Server web interface, browse to the Training repository and locate the published workspace. Select Data Download > Configure.

Because the source dataset parameter is published, you should now be able to upload data.

Use the Add Files button to add Properties.kml instead of using the uploaded version. Now, in the Published Parameters section, click Select From Uploaded Files and select Properties.kml

Click Run Workspace to run the workspace. Inspect the output. It should be as before.

5) Check Log
In the Jobs section of the interface, open the log for the previous translation. You should see that the data comes from a local copy of the file, not the one defined in the workspace:

```
Opening the XML reader with source dataset
'C:\apps\FMEServer\Server\Upload\Training\Properties.fmw\7CAE3A0B8CA3D20
0092238A5600CE9BF\Properties.kml'
```

Advanced Tasks
The Planning Department do mention a couple of tidy-ups they would like if you have time.

1) Restrict the list of output formats to a smaller number. By now you should know how to do this with a parameter of the type “Choice with Alias”. Pick a few formats you think a typical planning department user could handle!

2) Calculate the area in square feet, instead of degrees. You could do this by reprojecting the data before measuring it (Reprojector transformer). TX83-CF (EPSG:2277) is the coordinate system to use for this data.

Dynamic Layer Handling

Layers can be read dynamically rather than fixing them in the workspace

When a user uploads the source data for a translation – or when the source data is a database outside the author's control – then the feature types provided can differ from translation to translation.

In this scenario it is not good practice to tie a workspace to specific table names or to specific schemas. This would require maintenance to the workspace for every change to the source data.

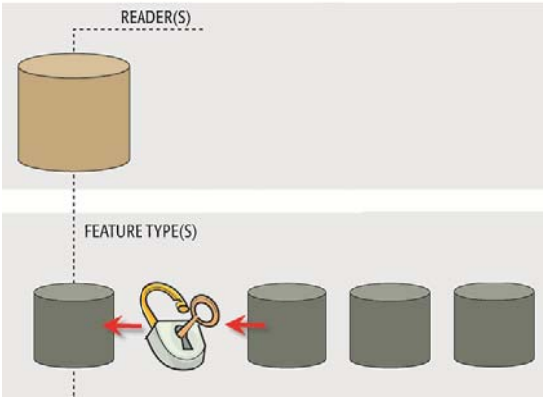
A flexible workspace needs to be able to process any table, even if it didn't exist at the time the workspace was initially created. This is achieved using a Dynamic workspace.

If a workspace contains a Dynamic Schema, it will be able to read and write any feature type regardless of what has been pre-defined.

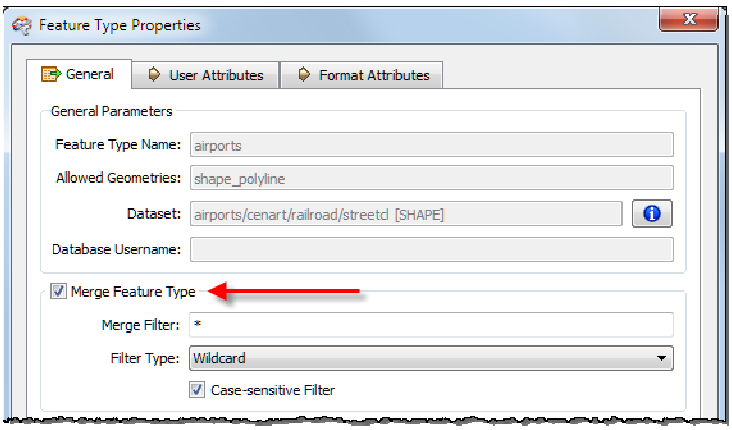
Reading Layers Dynamically
Whenever a new workspace is created, or a reader added to FME, there are options to make the schema dynamic.

On the reader side, a dynamic schema unlocks the reader and allows data from any feature types not already pre-defined in the workspace to pass.

Whenever a dynamic reader is added, the Feature Types to Read parameter is automatically published. This is because FME assumes a dynamic schema means the list of feature types will be uncertain.



This functionality is what lets FME Server users choose the layers they want to read.



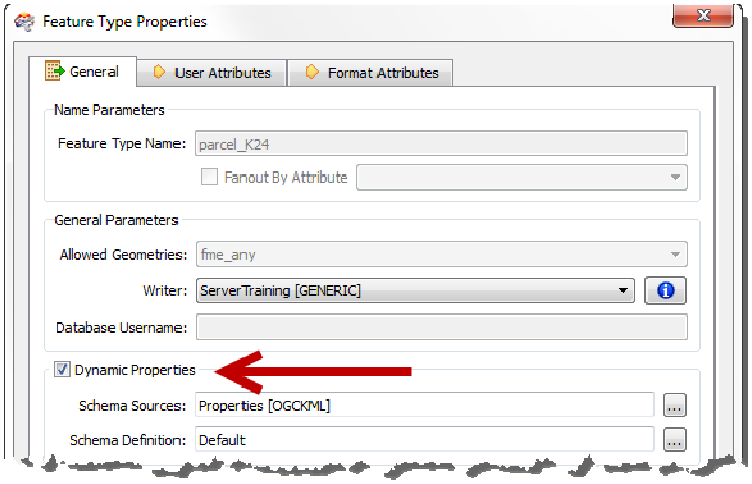
Looking closer at how this works, in a dynamic reader the properties of the source feature type have the Merge Feature Type option checked, and the Merge Filter set to the * wildcard.

Writing Layers Dynamically
If only data reading is dynamic, then the output will always be on the same feature types, regardless of what was contained in the input.

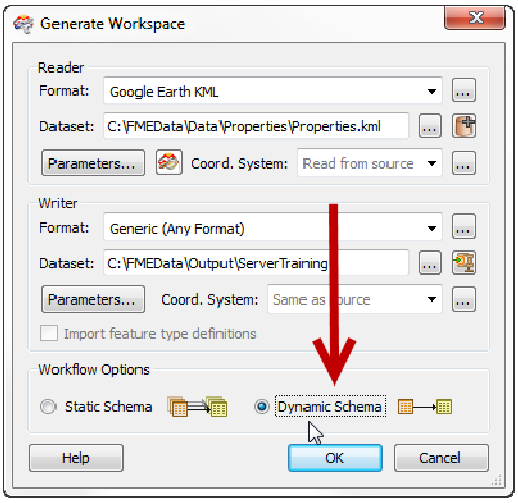
If the workspace author requires the same feature type names to appear in the output, as were in the input, then a dynamic writer is required.

With a dynamic writer it isn't necessary for the feature types being read to exist in the writer; the data can be written to any feature type in the workspace and will be given the correct name.

In a Dynamic Writer (signified by setting the "Dynamic Properties" setting in the Feature Type Properties dialog), the output schema is by default a copy of what is being read.



Creating Dynamic Workspaces
Although the dynamic properties can be set manually for a reader and writer, it's more common to use the Dynamic Schema option directly when generating the workspace:



Example 13: Dynamic Layers	
Scenario	FME user; City of Interopolis, Planning Department
Data	Property Boundaries (KML)
Overall Goal	Create an online service for converting KML datasets
Demonstrates	Dynamic Layer Handling
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example13Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example13Complete.fmw

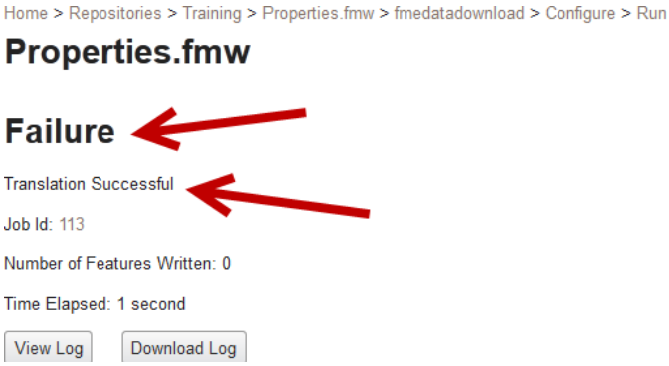
The planning Department says that they tried to run the workspace on a properties dataset but it failed to produce any output. Let's see why….

1) Run Workspace
In the FME Server web interface, browse to the workspace and choose Data Download > Configure.

The planning team were using the file: C:\FMEData\Data\Properties\Properties2.kml
So use the Add Files option to upload that dataset, select that as the file to use, and then press **Run Workspace** to run the translation.

Strangely FME Server reports the translation was a failure and yet successful!

Let's try and find out how this could be.



2) Check Log
Click the View Log or Download Log button to examine the log of the translation.

You should see that zero features were written. Although there is no error message, there is an interesting statistic:

```
STATS |Unexpected Input Remover(TestFactory):
Tested 428 input feature(s)
               -- 0 feature(s) passed and 428 feature(s) failed
```

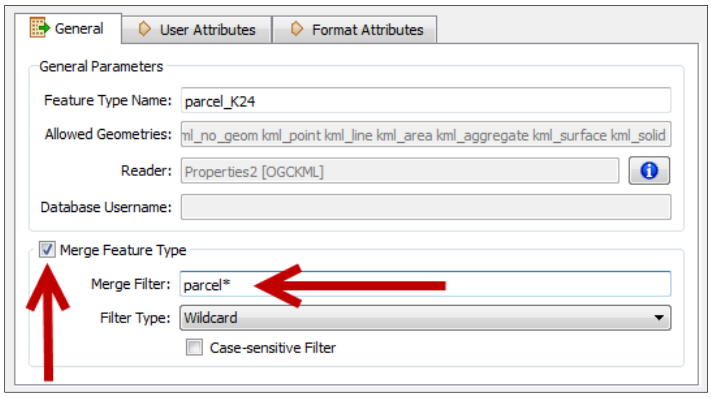
The Unexpected Input Remover is designed to filter out data whose Feature Type is undefined in the workspace. So, apparently, the problem is that the end user is trying to read data with a different feature type to the original dataset.

Inspect the source data – Properties2.kml – in the FME Universal Viewer to prove that this is the case.

3) Start Workbench
Start Workbench and open the workspace from the previous example (or open the pre-defined Starting Workspace).

Open the Feature Type Properties dialog for the reader feature type. Put a check mark in the Merge Feature Type parameter.

Set the Merge Filter to something that will let similar data through, for example parcel*



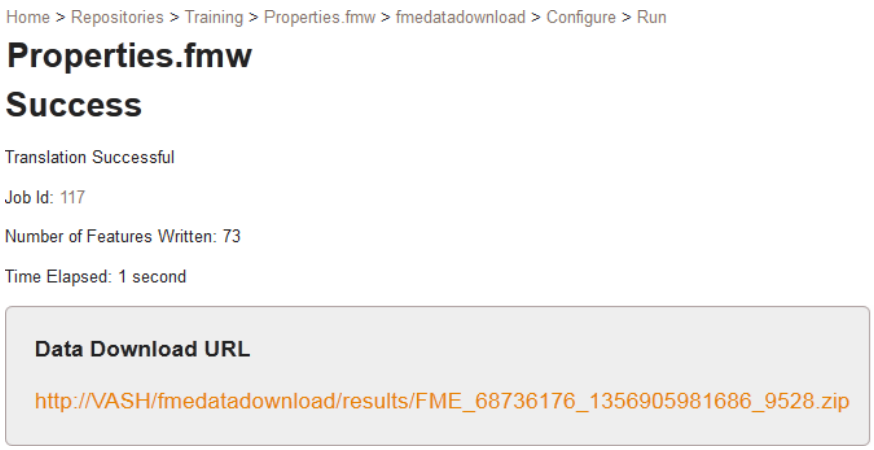
4) Run Workspace
Run the workspace (within FME Workbench) to prove that it now produces output.

Inspect the output dataset. Notice that it is still producing an output called parcel_k24.shp, even though the tile being read is now parcel_k25

To fix this, open the Feature Type Properties dialog for the writer feature type. Put a check mark in the Dynamic Properties parameter. Click **OK** to close the dialog.

5) Re-Run Workspace
Re-run the workspace. Notice that the output is now called parcel_k25.shp
It now works because the name is being created dynamically from the source schema.

6) Re-Publish Workspace
Re-publish the workspace to FME Server. Run it again – using both Properties.kml and Properties2.kml – to prove that the process now works for both datasets.



Catalog Service

The Catalog service is a mechanism for giving FME authors access to FME resources

What is the Catalog Service?
The Catalog service has two major functions: to retrieve a list of catalogued resources, and to create a shortcut (.fmc) file to a catalogued resource.

The purpose of these functions is to enable web developers to create an online inventory of resources that an FME author can browse and download.

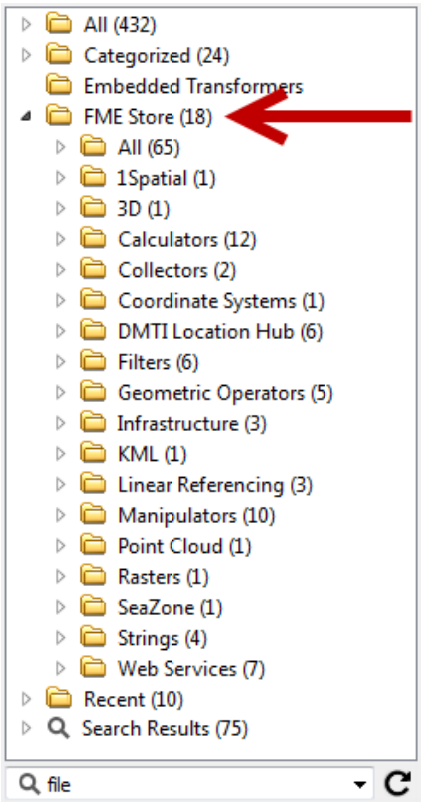
The FME Store
The FME Store is an excellent example of how the Catalog Service can be used.

Resources are uploaded to an FME Server installation and registered with the Catalog Service. This Server needs no engines, because items published to it will never be run – just made available for download.

The Transformer Gallery in FME Workbench queries the catalog and presents a list of the available Custom Transformers.

When a resource on the FME Store is selected, then the resource is downloaded (via a .fmc file) and installed locally:

```
Requesting catalog details for 'AreaUnitConverter'...
Sending request to FME Store...
Successfully sent request.
Retrieved catalog details.
Connecting to FME Store...
Successfully connected.
Downloading 'AreaUnitConverter.fmx'...
Successfully downloaded file.
Successfully downloaded 'AreaUnitConverter.fmx' from FME Server
Disconnecting...
Successfully disconnected.
```



Example 14: Catalog Service	
Scenario	FME user; City of Interopolis, Planning Department
Data	None
Overall Goal	Check contents of repository via the Catalog Service
Demonstrates	Catalog Service
Starting Workspace	None
Finished Workspace	None

Let's check the catalog for the previous custom transformer uploaded in the previous exercise.

1) Open Web Browser
Open a web browser. In the navigation toolbar enter the URL:

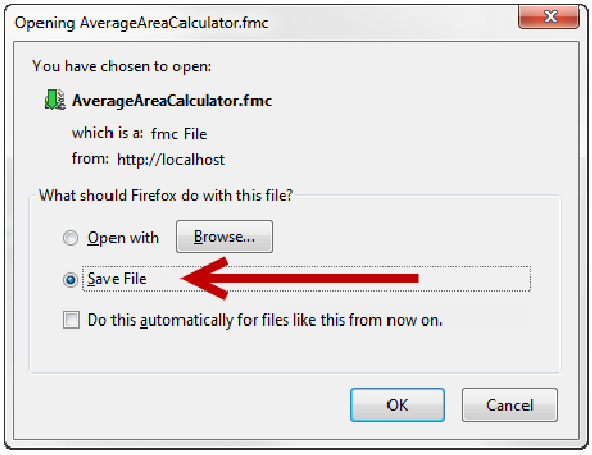
```
http://localhost/fmecatalog/catalog
```

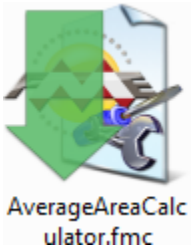
This will return a list of all items, in all repositories – *where you have permission to query it*.

2) Download Custom Transformer Catalog
`http://localhost/fmecatalog/Training/AverageAreaCalculator.fmx`

Notice that this will cause a .fmc file to be retrieved from the catalog service.

Choose the option to Save File





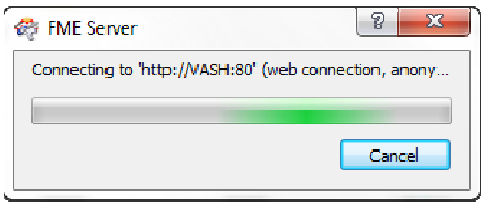
3) Inspect File Contents
Locate the saved file in a file browser.
It will have a unique FME-style download icon.

Open the file in a text editor to inspect its contents.

Notice that it is an XML file. Importantly, its contents include the transformer name, the transformer description, and the location of the transformer on FME Server.

4) Double-Click File
Double-click the fmc file. A connection will be made to FME Server – you will need to enter your access credentials – to download the resource.

You will be prompted for a location to save the downloaded custom transformer.



Token Service

The Token Service helps implement an alternative method of security credentials

What is the Token Service?
The Token Service is for generating security credentials in the form of a single key (token) that can be used as proof of access rights instead of a username and password.

The benefit of this method is that the end user has a single sign-on for all aspects of FME Server and does not need to enter their password repeatedly for each operation.

Token security is intended for use with custom web applications. The REST web interface allows (in fact, requires) its use, but the basic FME Server web interface does not support this type of security.

Tokens can be generated dynamically via a HTTPS call, and will have a set expiration time.

Example 15: Token Service	
Scenario	FME user; City of Interopolis, Planning Department
Data	None
Overall Goal	Create a Security Token
Demonstrates	Token Service
Starting Workspace	None
Finished Workspace	None

Let's create a security token, and then use it to access the REST API.

1) Open Web Browser
Open a web browser. In the navigation toolbar enter the URL:

```
http://localhost/fmetoken/service/generate?user=<yourusername>&password=
<yourpassword>&expiration=1&timeunit=day
```

This will generate a token for you to use for a single day. If this technique fails then use the URL:

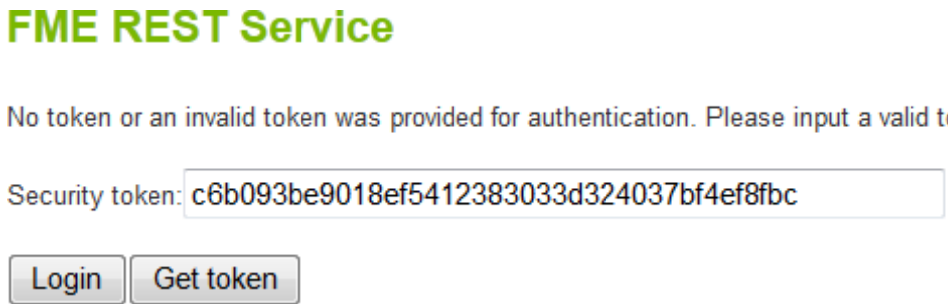
```
http://<servername>/fmetoken
```

…to access the token service directly.

2) Open REST Web Interface
Open the REST web interface. You can use the shortcut on the start menu or open a web browser and enter the URL:

```
http://<servername>/fmerest
```

3) Enter Token
When prompted, enter the token you generated and click **Login**. You now have access to the REST API through the web interface.



Advanced Task
If you have the time and experience, why not try creating a web interface that accepts username and password in a form and uses them to create a token.

REST Service

The REST Service is the foremost service for use by web developers

What is the REST Service?
The FME Server REST Service provides a simple, open Web interface directly to the key functionality of FME Server.

With this service a web developer is capable of creating dynamic, easy maintainable, highly customizable web applications, for running all kinds of Spatial ETL processes online.

This URL-driven approach also makes it easier to integrate FME Server with other applications and web services such as ArcGIS Server, Google Maps, Bing Maps, OpenLayers, and other web API's.

REST resources and operations are accessible either through a hierarchy of URI's, or in the FME Server REST web interface.

The REST interface is accessed either through a shortcut on the Start menu, or with the URL:

`http://<servername>/fmerest`

URI Structure
All URIs for the rest service should be issued with a security token as a parameter, otherwise they will fail to be processed; for example:

```
http://localhost/fmerest/services?token=c6b093be9018ef5412383033d324037bf4ef8fbc
```

URIs can also be suffixed with .<format> where <format> is one of either xml, json, or html. This causes the response to be in the selected format; for example:

```
http://localhost/fmerest/repositories.json?token=.....
```

The REST web interface is basically a way to access and navigate through the REST API, where all the responses are in HTML format.

A list of repositories as returned by the REST API in three different formats: XML, JSON, HTML

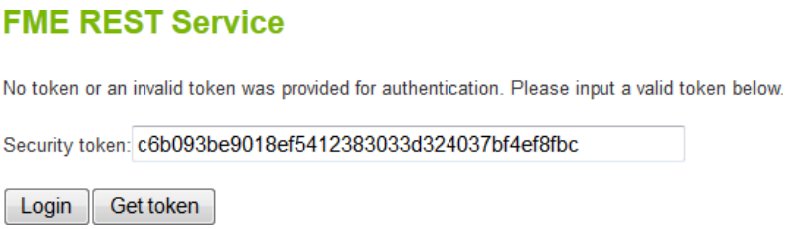
REST Methods
The REST API has methods for handling many aspects of FME Server. Some of the key ones are:

Repositories	Creating a repository Deleting a repository Retrieving a list of repository contents
Workspaces	Running Registering/Deregistering with a Service Retrieving a list of Readers/Writers Retrieving a list of Feature Types Retrieving a list of Published Parameters
Services	Retrieving a list of Services Creating a Service Deleting a Service Retrieving a list of Service Resources
Jobs	Retrieving a list of Jobs (Running, Queued, Completed) Cancelling a job Deleting a job from the history Retrieving a list of job results
FME Resources	Uploading a Resource Downloading a Resource Deleting a Resource
Scheduling	Creating a scheduled job Deleting a scheduled job Retrieving a list of scheduled jobs
Notifications	Creating a Topic/Subscriber/Publisher Deleting a Topic/Subscriber/Publisher Updating a Topic/Subscriber/Publisher Retrieving a list of Topics/Subscriber/Publisher
Logs	Retrieving a list of Log Files Viewing a Log File Downloading a Log File

Example 16: REST Interface and API	
Scenario	FME user; City of Interopolis, Planning Department
Data	None
Overall Goal	Carry out actions using the REST interface and API
Demonstrates	REST Service
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example16AdvancedComplete.fmw

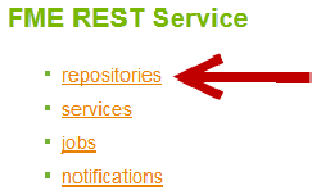
Let's use the REST API to locate, query, and run the planning department workspace from the previous examples.

1) Open REST Web Interface
Open the REST Web Interface and log in with your security token.



2) List Repositories
Click repositories in the REST interface.

Notice how this returns a list of repositories, in HTML format. Check the current URL to see what method is being used, and how the security token has been incorporated.

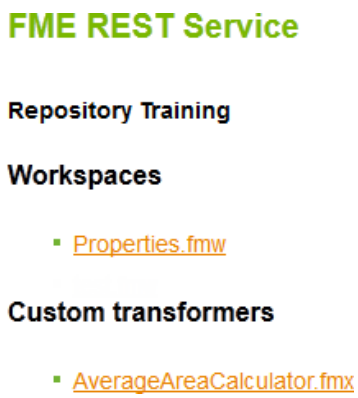


Try clicking the XML and JSON links on the left-hand side of the interface, to see how the data looks in these formats.

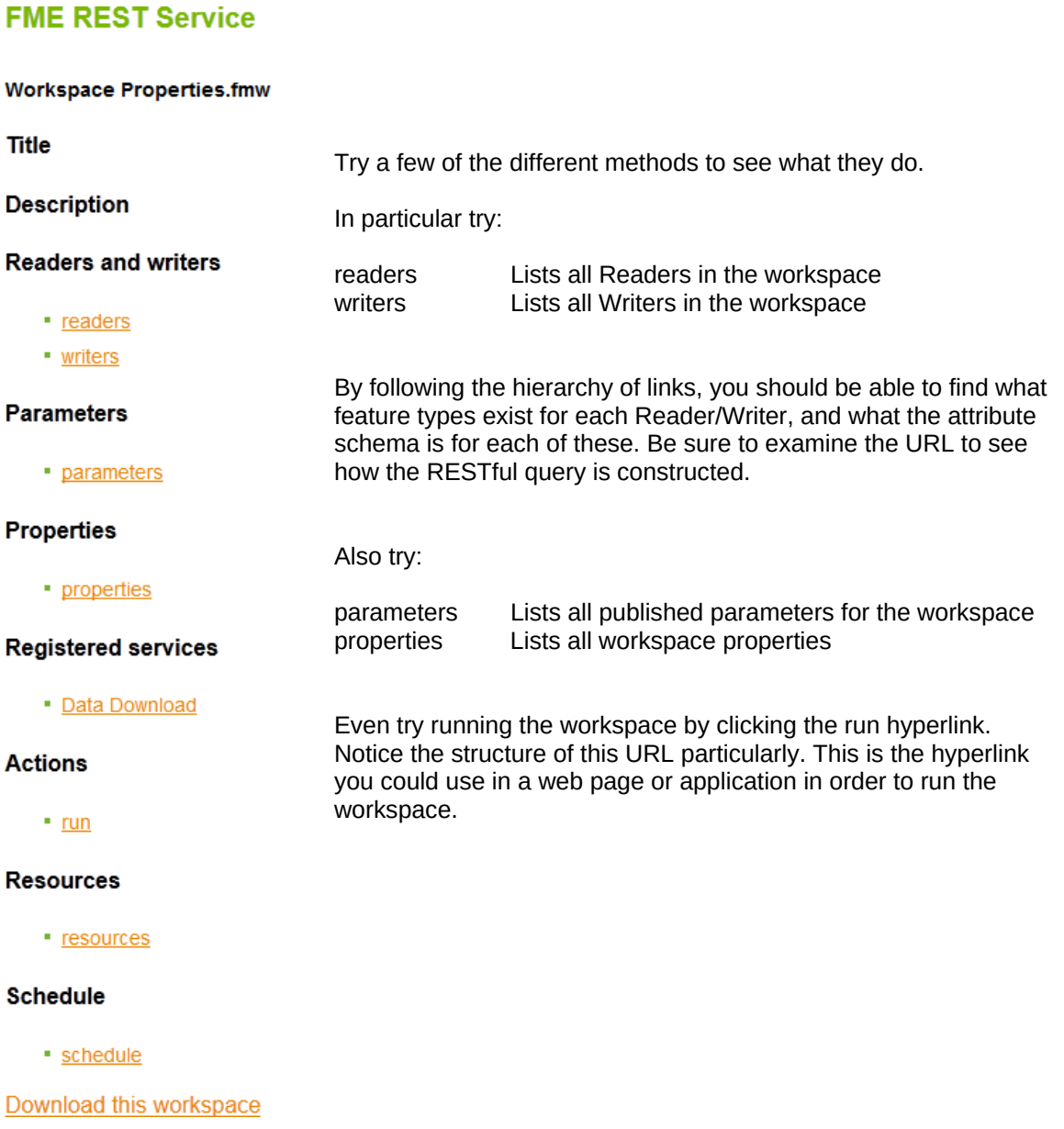
Check the web browser's URL to see how the format has been incorporated.

3) Select Repository
Back in the HTML interface, click on the Training repository to select it. Notice that it now lists all of the different FME resources available in that repository.

Again, check the current URL to see what method is being used, and perhaps try the XML and JSON links to return the information in that format.



4) Select Workspace
Back in the HTML interface, click on the workspace (Properties.fmw) to select it.
Notice that there are a whole series of links to different methods that can be used on a workspace.



Try a few of the different methods to see what they do.

In particular try:

readers	Lists all Readers in the workspace
writers	Lists all Writers in the workspace

By following the hierarchy of links, you should be able to find what feature types exist for each Reader/Writer, and what the attribute schema is for each of these. Be sure to examine the URL to see how the RESTful query is constructed.

Also try:

parameters	Lists all published parameters for the workspace
properties	Lists all workspace properties

Even try running the workspace by clicking the run hyperlink. Notice the structure of this URL particularly. This is the hyperlink you could use in a web page or application in order to run the workspace.

Advanced Tasks
If you have the time and web skills, why not try putting together a web page that queries a repository and returns a list of workspaces. Each workspace should be a hyperlink to the RESTful URL that runs the workspace.

You may find the FME Server REST Playground contains useful materials to help you with this task. Visit: http://fmeserver.com/userweb/sharper/playground/index.html

Alternatively, try using the REST API inside an FME workspace by following these instructions:

1) Start Workbench
Start FME Workbench and begin with a blank canvas.

2) Add Creator Transformer
Add a Creator transformer to start off the workspace.

3) Add HTTPFetcher Transformer
Add a HTTPFetcher transformer after the Creator.

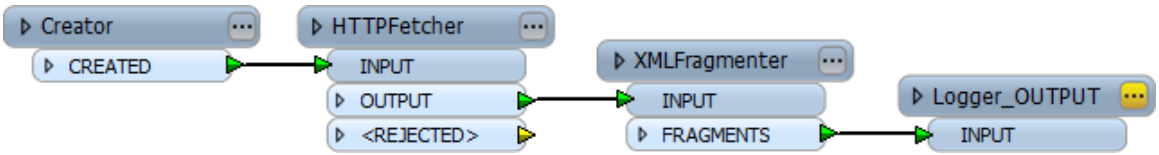
Open the parameters dialog. In the Target URL field enter a query to your FME Server's REST API. For example:

```
http://localhost/fmerest/repositories/Samples.xml?token=<YourToken>
```

Add a Logger or Inspector transformer after the HTTPFetcher and run the translation to see what information is returned. It should be a list of workspaces in the Samples repository.

```
`_creation_instance' has value `0'
`_headers{0}.name' has value `Date'
`_headers{0}.value' has value `Mon, 31 Dec 2012 20:16:00 GMT'
`_headers{1}.name' has value `Content-Type'
`_headers{1}.value' has value `text/xml;charset=UTF-8'
`_headers{2}.name' has value `Content-Length'
`_headers{2}.value' has value `1502'
`_headers{3}.name' has value `Server'
`_headers{3}.value' has value `Apache-Coyote/1.1'
`_http_status_code' has value `200'
`_url_contents' has value `<?xml version="1.0" encoding="UTF-8"?><serviceResponse>
`fme_feature_type' has value `Creator'
`fme_geometry' has value `fme_undefined'
`fme_type' has value `fme_no_geom'
```

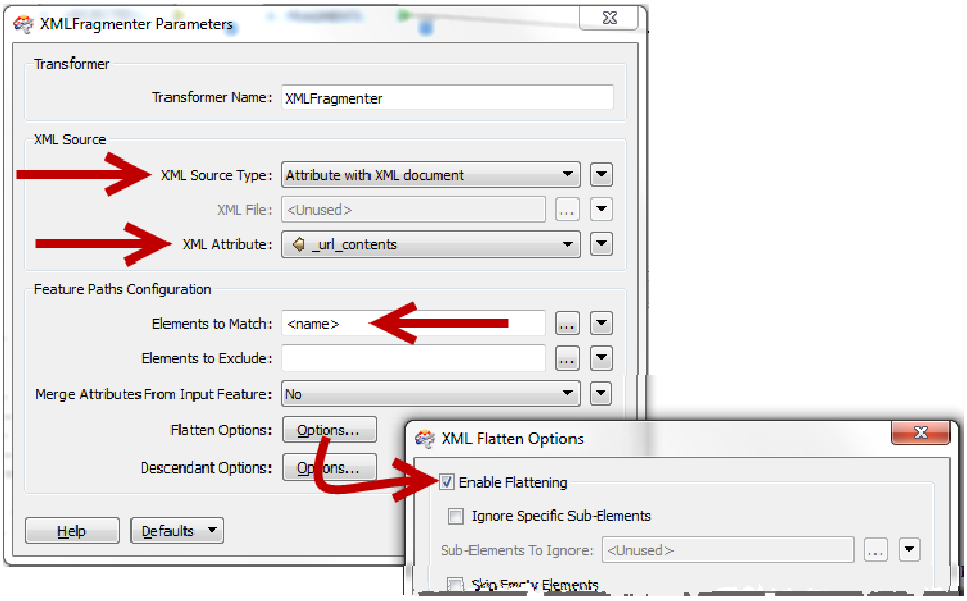
4) Add XMLFragmenter Transformer
Add an XMLFragmenter transformer after the HTTPFetcher.



5) Set Parameters
Open the parameters dialog.

Set the Source Type to be an Attribute and the attribute to be _url_contents (the contents retrieved by the HTTPFetcher).

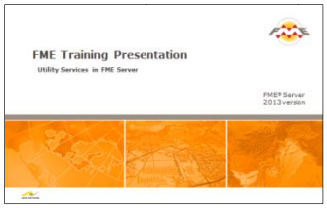
Set Elements to Match to <name> and, under Flatten Options, put a checkmark against Enable Flattening.



6) Re-Run Workspace
Re-run the workspace. Now you have retrieved a list of workspaces in XML format, and exploded that list to create a separate feature for each.

```
Feature Type: 'XMLFragmenter_FRAGMENTS'
Attribute(string)          : `fme_type' has value `fme_no_geom'
Attribute(encoded: utf-16): `name' has value `easyTranslator.fmw'
Attribute(encoded: utf-16): `xml_fragment' has value ``<?xml version="1.0" encoding="UTF-16"?><name>easyTranslator.fmw</name>'
Attribute(encoded: utf-16): `xml_id' has value `id-name-1.3.3.6.3'
Attribute(encoded: utf-16): `xml_matched_element' has value `name'
Attribute(encoded: utf-16): `xml_parent_child_pos' has value `'
Attribute(encoded: utf-16): `xml_parent_id' has value `'
Attribute(string)          : `xml_type' has value `xml_no_geom'
Geometry Type: Unknown (0)
```

Module Review



This module introduced you to FME Server's Utility Services.

What You Should Have Learned from this Module
The following are key points to be learned from this module.

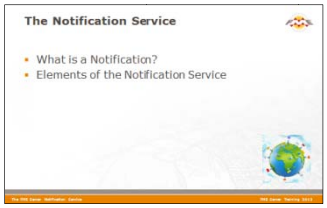
Theory

- Utility Services provide programmatic access to FME Server for carrying out administrative tasks
- Data can be uploaded to FME Server by either the workspace author or end-user
- Dynamic reading and writing means that the source data can contain any layers (feature types) even if they aren't pre-defined in the workspace
- The Catalog Service allows creation of an inventory of resources on an FME Server.
- The Token Service is for generating security credentials that can be used as proof of access rights instead of a username and password.
- The REST Service is an API that allows access to FME Server through RESTful HTTP requests.

FME Skills

- The ability to upload data when publishing a workspace, or prompt the end-user to upload data at run-time
- The ability to handle source feature types using a dynamic translation
- The ability to use the FME Server Catalog service
- The ability to use the FME Server Token service
- The ability to use the FME Server REST service

The Notification Service



The Notification Service is a way for data to be pushed to and from FME Server in the form of short messages.

What is a Notification?
A notification is a message (sometimes called an "alert") that informs someone or something that a particular event has happened.

For example, a person might be alerted when a database record has been updated, or a software component might be alerted when a user has submitted data to be processed.

Notifications on FME Server can occur in two different forms. Incoming notifications alert FME Server to an event that has taken place elsewhere. Outgoing notifications alert users to an event that has taken place on FME Server.

In this way, FME Server can take action in response to an event it is notified about, or a user can take action in response to an event that FME Server notifies them about.

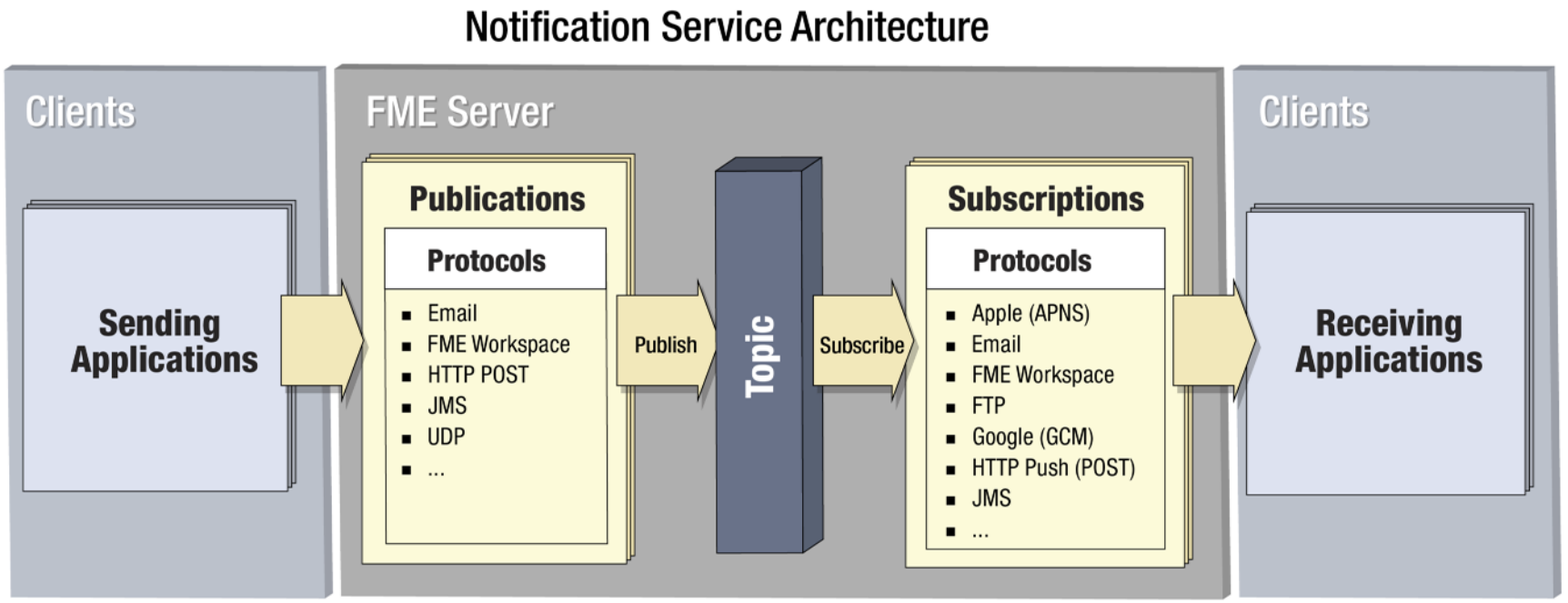
The **Notification Service** is the part of the FME Server architecture that handles incoming and outgoing notifications.

Elements of the Notification Service
The Notification Service includes a number of different elements. The diagram on the following page shows them as they are arranged.

- Clients — An external user or system that sends or receives a notification
- Publications — Objects that listen for incoming notifications
- Subscriptions — Objects that dispatch outgoing notifications
- Protocols — Methods by which FME Server can receive or send notifications
- Topics — A subject that describes what the notification is about

Although the diagram shows a continuous process, from a sending client through FME Server to a receiving client, it is not necessary for all of these components to be used in a particular setup.

If the system is designed for FME Server to only receive notifications, then only the publications side of the diagram is relevant. Likewise, if the system is intended for FME Server to only send notifications, then only the subscriptions side of the diagram applies.



- Clients — An external user or system that sends or receives a notification
- Publications — Objects that listen for incoming notifications
- Subscriptions — Objects that dispatch outgoing notifications
- Protocols — Methods by which FME Server can receive or send notifications
- Topics — A subject that describes what the notification is about

Clients
A client is a user or system that sends or receives a notification. The client may be a physical person, or may just be a component in a computer system.

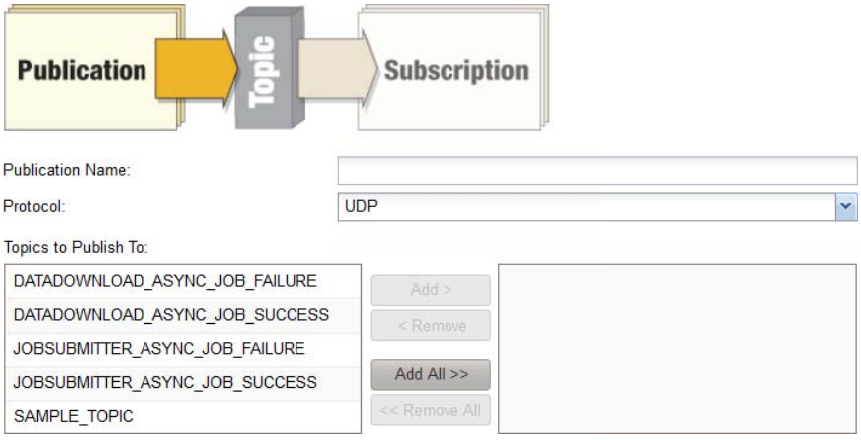
For example, a database update might cause a trigger to send a notification to FME Server, in which case the database system is the client. But a client could also be a person who, for example, triggers a notification by sending an email to FME Server. Since they are sending information you might call these clients "**Publishers**".

Likewise, FME Server can send a notification to either another component in the system (including sending a notification back to itself, in order to trigger another FME action), or it can send a notification to an end-user, for example in the form of an email. Since they are receiving information, you might call these clients "**Subscribers**".

Publications
Publications are components that receive incoming notifications from a client and pass them on to a particular topic.

Publications are created in the FME Server web interface under the Notifications section, by selecting the Publications tab and clicking on the New button:

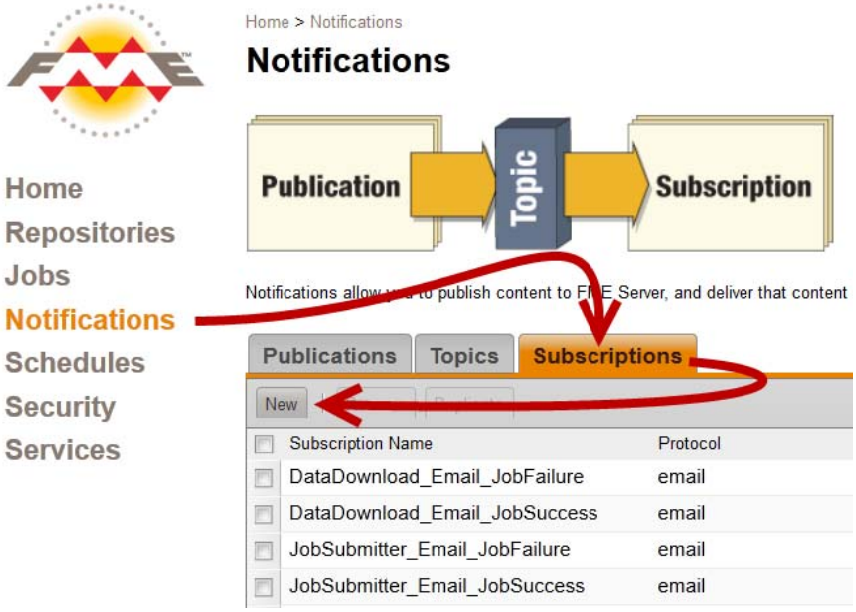
A publication is tied to a particular protocol and to a set of topics, so you will need to select these when creating the publication:



Being able to select multiple topics means that a single notification can trigger multiple responses on FME Server, and alert multiple subscriptions.

Subscriptions
Subscriptions are components that are triggered by a particular notification Topic and in response send output notifications to a client.

Subscriptions are created in the FME Server web interface under the Notifications section, by selecting the Subscriptions tab and clicking on the New button.



Again, a subscription is tied to a particular protocol and set of topics, so you will need to select these when creating the Subscription.

As shown above, FME Server includes a set of pre-defined subscriptions. These are triggered by the success or failure of an FME workspace and in response will send emails to any subscribers.

Protocols

A protocol is a method by which notification communications take place between FME Server and a client. In other words it is the method by which FME Server receives notifications and sends notifications.

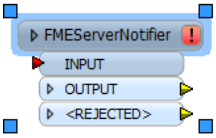
Protocols are pre-defined objects in the FME Server architecture and have no dialog to create them in the web interface. However, a number of fields are made available to configure them when a Publication or Subscription chooses to make use of that protocol.

There are several different protocols available; some of them are only for use on Publications, some are only for Subscriptions, and some of them can be used with both notification types.

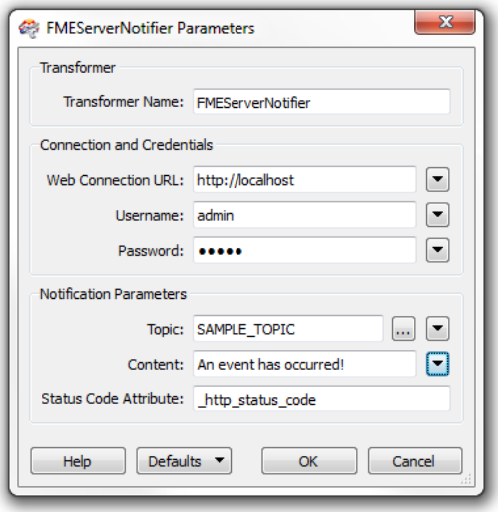
Name		Description	Publications	Subscriptions
Apple Push Notification		Alerting Apple mobile devices		✓
Email		Communication via email	✓	✓
FTP		Writing to an FTP site		✓
Google Cloud Messaging	NEW	Alerting Android mobile devices		✓
JMS	NEW	Communication with a Java Message Service	✓	✓
Logger		Output to a simple log file		✓
Push		Pushing notifications via HTTP requests		✓
REST API		Posting via a HTTP Post or the REST interface	✓	
UDP		Communication via a User Datagram Protocol port	✓	

FME Workspaces can also act as a form of protocol.

On the publication side, an FME Workspace can trigger a topic through the FMEServerNotifier transformer:



On the subscription side, an FME workspace can be triggered by a topic.



The workspace may just run as per usual, or it may itself act as a publisher and trigger a different topic.

In this way, a workspace can be triggered to produce output, which is then brought to the attention of an end-user through a Subscription.

It is this capability that makes FME Server a very powerful engine for spatial data notifications.

Topics

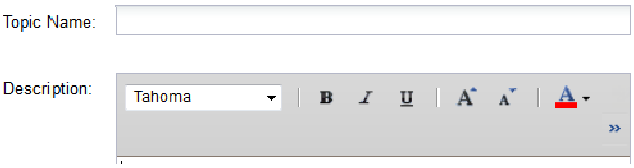
A notification topic is a keyword that identifies the subject that the notification concerns.

For example, a topic could be named “DatabaseUpdate” to indicate that related notifications are concerned with updates made to a particular database.

It would be expected that publishers that trigger the topic would be responding to a database update, and that subscribers alerted by the topic would have subscribed to receive alerts on database updates.

Topics can be defined within a workspace, either when it uses an FMEServerNotifier transformer or when the workspace is published to FME Server.

However, topics are usually defined using the FME Server Web Interface:



In the web interface only a topic name and description is required. Topics are joined to Publications and Subscriptions in their creation dialog, not this one.

Topic Relationships

The relationship between topics and Publications/Subscriptions is many-to-many (M:M).

A single Topic can be published to by multiple Publications. This scenario allows the topic to be triggered by multiple different protocols. For example, the “DatabaseUpdate” topic could be notified by either an email, or via JMS or UDP.

The reverse is also true, and a single Publication can also publish to multiple topics. This way it can cause FME Server to take several actions upon receipt of a single notification.

Similarly, a single Topic can alert multiple Subscriptions to an event. Again, this allows the topic to alert subscribers via multiple different protocols. For example, some subscribers could be alerted to a database update via email, others through a mobile device (Apple or Android).

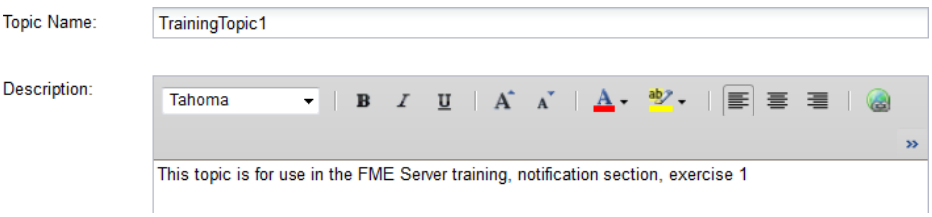
And, as you might expect, a single Subscription can be triggered by multiple topics. This way a subscriber can be notified about multiple events – for example database updates, emergency alerts, and weather events – all using the same method (protocol).

Example 17: Notification Subscriptions	
Scenario	FME user; City of Interopolis, City Maintenance
Data	None
Overall Goal	Issue a basic notification alert to subscribers
Demonstrates	Notification subscriptions
Starting Workspace	None
Finished Workspace	None

This example creates basic Subscribers and posts to them using the REST API to publish to a topic.

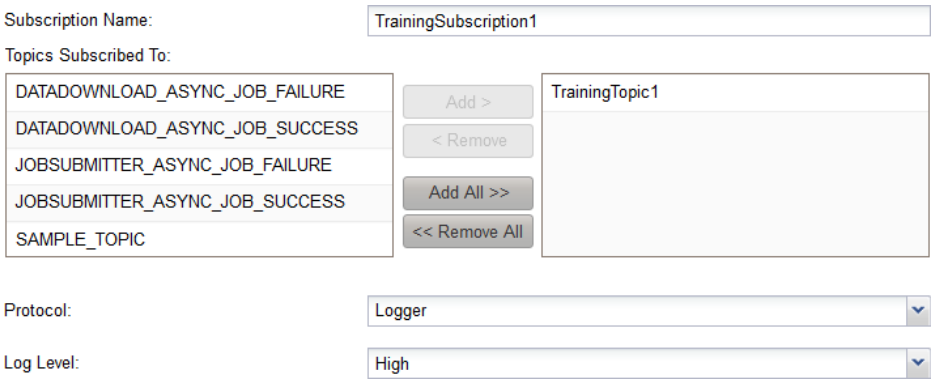
1) Open FME Server Web Interface
Open the FME Server web interface (if necessary) and log in.
Browse to Notifications and click the Topics tab

2) Create Topic
Click **New** to start creating a new topic.
When prompted, create a topic called TrainingTopic1 and (optionally) enter a description.



3) Create Subscription
Click the Subscriptions tab, and then click **New** to start creating a new subscription.

When prompted, create a Subscription called TrainingSubscription1.
Select the topic TrainingTopic1 as the single topic to subscribe to.
Select Logger as the protocol and select the Log Level as High.

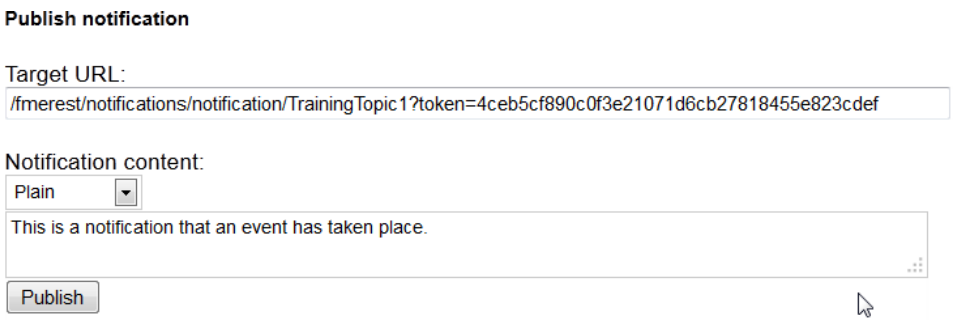


We have now created a Subscription that is tied to a particular topic and waiting for a notification.

4) Open REST Web Interface
The simplest way to publish to a topic is to use the FME Server REST API.
Open the FME Server REST web interface. Create and/or enter your security token to log in.

Browse to notifications > topics > TrainingTopic1
Notice that there are some fields by which a topic can be published to.

Select Plain as the notification content type.
Enter some plain text as notification content.



Click Publish to publish the notification.
We have now published a notification to the topic, which will pass it on to the Subscription.

5) Open Log
In a file browser, browse to <FME Server>\Logs\notifier\subscribers\logger
Open the most recent log file. It should report the success of the notification:

```
Thu-20-Dec-2012 03:47:48 PM   INFORM   800005 : onNotification:
notification { topic : TrainingTopic1, message { subscriber_content :
This is a notification that an event has taken place. } }
```

6) Create Subscription
Back in the FME Server web interface, browse back to the Subscriptions tab and again click New to start creating a new subscription.

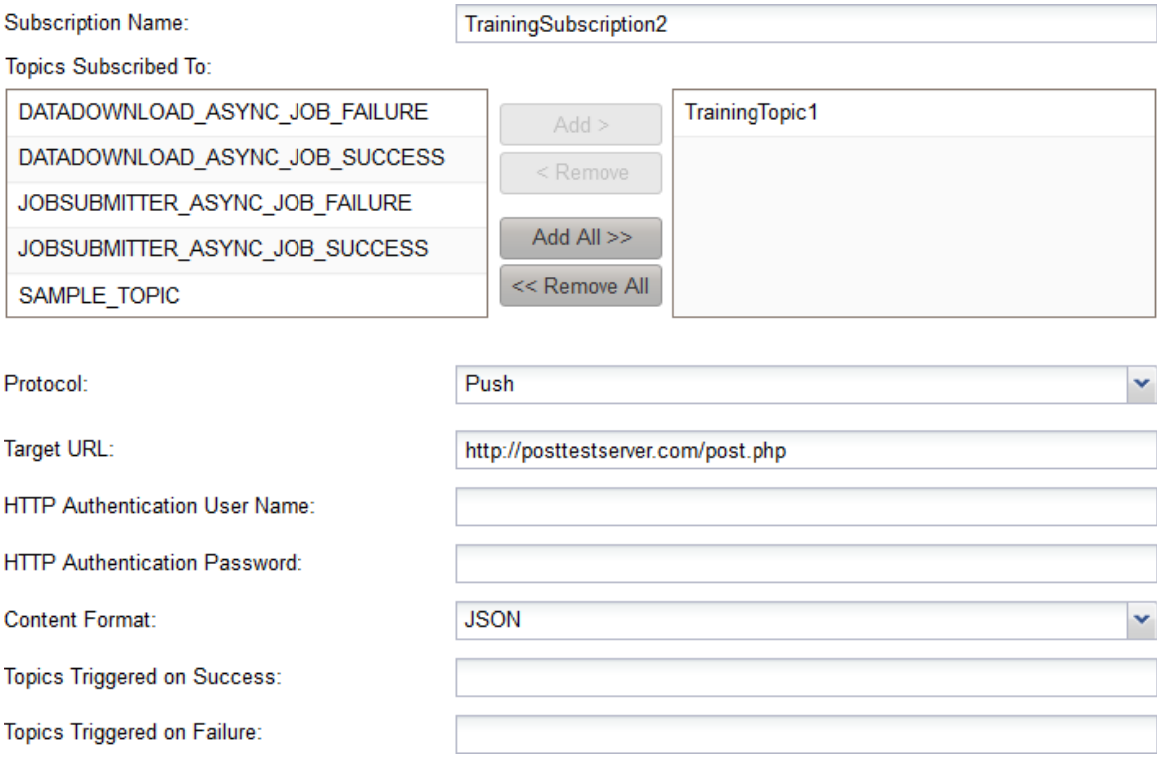
When prompted, this time create a Subscription called TrainingSubscription2
Select TrainingTopic1 as the topic again, but this time set the Protocol to Push.

The Push protocol will send information to a http site with a POST command.

For the target URL enter: http://posttestserver.com/post.php
This is a web site that can be used to test post commands.

Set the content format to either JSON or XML (it doesn't matter for this example) and leave the rest of the fields empty.

Click OK to create the Subscription.



7) Open REST Web Interface
Again, open the FME Server REST web interface and use it to post a notification to the TrainingTopic1 topic.

8) Visit posttestserver.com
In a web browser visit http://posttestserver.com/data/2013

Navigate through the folders to the current date, and locate the most-recent entries. Click to open it. You should see your notification successfully posted.

```
Time: Thu, 20 Dec 12 13:57:16 -0800
Source ip: 216.36.189.163

Headers (Some may be inserted by server)
UNIQUE_ID = UNOJvNBx6hIAACsDwnwAAAAC
CONTENT_LENGTH = 42
CONTENT_TYPE = text/plain; charset=UTF-8
HTTP_HOST = posttestserver.com
HTTP_CONNECTION = close
HTTP_USER_AGENT = Apache-HttpClient/4.0.3
REMOTE_ADDR = 216.36.189.163
REMOTE_PORT = 52167
GATEWAY_INTERFACE = CGI/1.1
REQUEST_METHOD = POST
QUERY_STRING =
REQUEST_URI = /post.php
REQUEST_TIME = 1356040636

No Post Params.

== Begin post body ==
This is a second notification of an event.
== End post body ==
```

Notifications and Workspaces

Workspaces are what make FME Server the ideal engine for Spatial Data notifications

Technically speaking, FME Server can receive and send notifications without having a workspace intervene. However, a workspace is the key functionality of FME. With a workspace you can read spatial data from with a message, in almost any format, carry out spatial transformations on that data, and then pass on the results to a set of subscribers.

This blend of live messaging with Spatial ETL is unique.

Workspaces as a Publication

When used as a publisher, a workspace is notifying FME Server of an event that has taken place. Usually this will be used to pass on information as to the status of a particular translation.

For example, a user might carry out a set of intensive translations that take several hours to run. He can use the workspace to publish to a particular topic when it is complete, to alert the user as to the status and result of the translations.

Note that the workspace doesn't need to be run on FME Server; it can equally publish to a topic whether it is run on FME Desktop, within an FME Objects application, or from the command line.

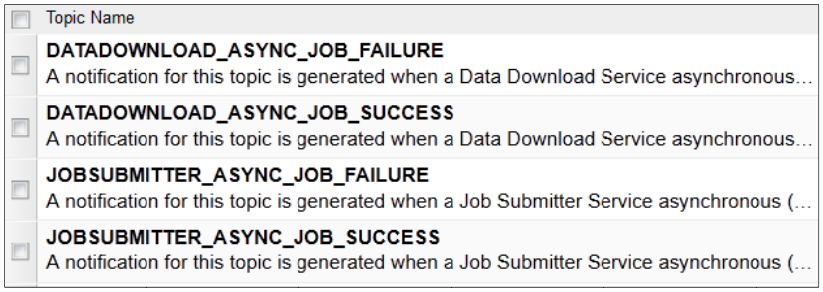
Publishing from a Workspace

Workspaces can publish to a topic in one of two ways.

Firstly a workspace can use the FMEServerNotifier transformer to issue an alert. The transformer parameters include user credentials required to connect to FME Server, and notification parameters such as the topic to post to and the message to be included.

Secondly, a workspace will publish to a topic whenever it is run on FME Server as either a Data Download (in asynchronous mode) or a Job Submitter service. That's because there are pre-defined topics for these scenarios. Therefore a Subscription can listen for these topics and issue a notification based on the result of the workspace.

There are two topics each for Data Download and Job Submitter, one each for a successful translation and a failed translation.



Therefore a system administrator could subscribe to these topics to listen for when a workspace has failed.

Example 18: Publishing from a workspace	
Scenario	FME user; City of Interopolis, City Maintenance
Data	Earthquakes
Overall Goal	Publish notifications from a workspace
Demonstrates	Notification subscriptions
Starting Workspace	C:\FMEData\Workspaces\ServerManual\Example18Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example18Complete.fmw C:\FMEData\Workspaces\ServerManual\Example18-2Complete.fmw

This example uses a workspace to publish a notification to subscribers, to inform them that a dataset has been updated.

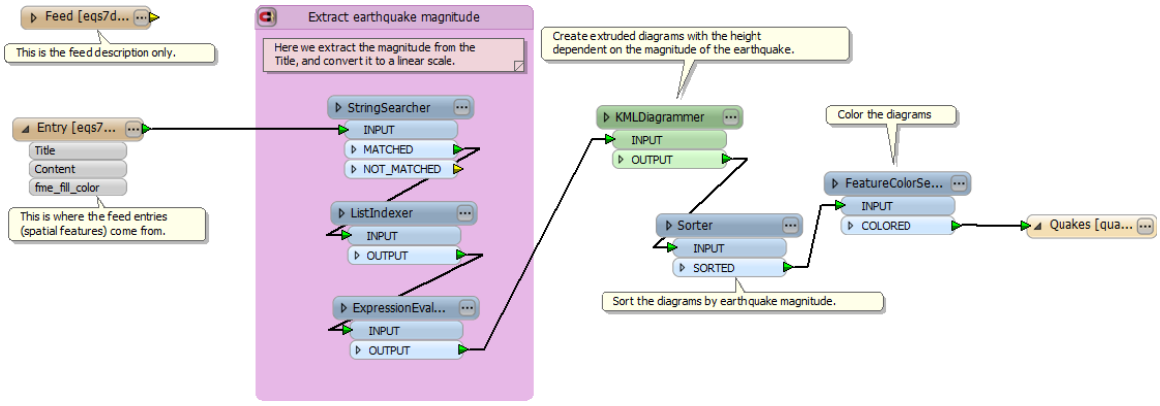
1) Start FME Workbench

Start FME Workbench. The starting point for this example already exists as a sample on FME Server, so either download it from there or open the starting workspace as shown above.

Repository Samples
Workspace earthquakesextrusion.fmw

2) Run Workspace

The workspace reads a feed of the latest earthquakes to have occurred, from the USGS web site. It then creates a three-dimensional representation in KML for viewing in Google Earth.



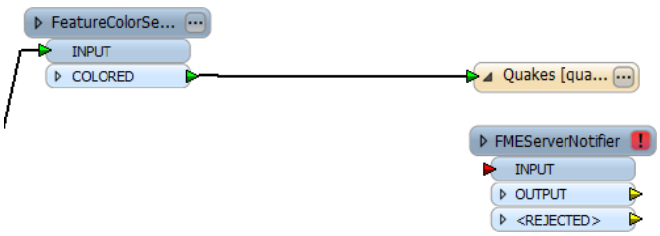
Run the workspace and inspect the output in Google Earth, to see what it is doing.

3) Add FMEServerNotifier Transformer

Add an FMEServerNotifier transformer, but don't connect it yet.

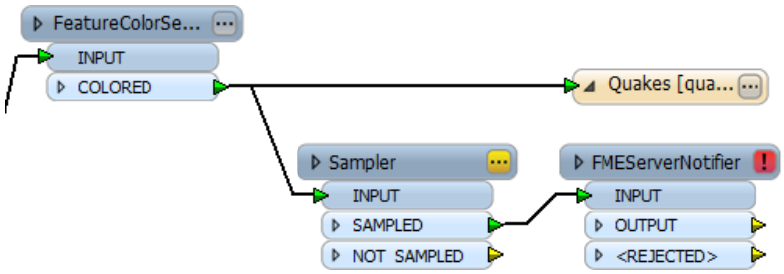
If the transformer were connected into the main workflow then it would create an alert for every earthquake.

However, we wish to create only one notification for the entire dataset.

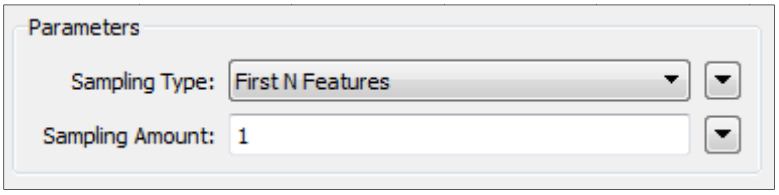


4) Add Sampler Transformer
A Sampler transformer will ensure only a single feature triggers the notification.

Add a Sampler transformer. Make a second connection from the last transformer in the current workflow (a FeatureColorSetter) to the Sampler. Connect the Sampler to the FMEServerNotifier.



5) Set Sampler Parameters
Open the Sampler parameters dialog and set it up to sample only the first feature in the workflow.

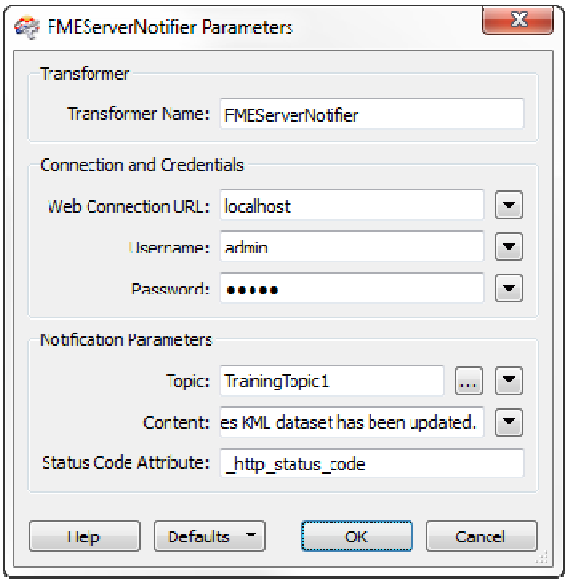


6) Set FMEServerNotifier Parameters
Open the FMEServerNotifier parameters dialog.

Fill in the connection and credential details for your FME Server installation.

Click on the Topic "browse" button. You will be presented with a list of topics.
Select TrainingTopic1

For the Content field simply enter a fixed string such as: "Notification: The Earthquakes KML dataset has been updated".



7) Run Workspace
Run the workspace. Check the Notification log files, and the posttestserver.com contents (see previous example), to ensure that the workspace properly triggered the topic and issued the correct notification.

```
== Begin post body ==
Notification: The Earthquakes KML dataset has been updated.
== End post body ==
```

Instead of the Sampler transformer, a Creator could have been used instead to create a trigger feature. Either way, the FMEServerNotifier transformer is useful because it allows us to issue a notification to FME Server from a workspace that is being run on FME Desktop.

However, this method has the drawback that a notification will be issued without any guarantee that the workspace has run to completion. The writer could still fail for some reason, leaving the user being alerted to a new dataset that doesn't exist.

So, for notifications that ought to check for workspace status before being issued, and that are intended to run on FME Server, built-in Topics exist that are triggered by workspace success and failure.

Example 19: Using a workspace as a Subscriber	
Scenario	FME user; City of Interopolis, City Maintenance
Data	Geohash Coordinates
Overall Goal	Receive notification of an earthquake, process the data and pass it on
Demonstrates	Notification subscription by a workspace
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Example19Complete.fmw

This example uses a workspace to receive a notification of an event, and then publish a notification to subscribers, to inform them of where the event has taken place.

Let's assume that an earthquake monitoring service issues alerts using a GeoHash coordinate. We'll create an FME workspace to process that incoming GeoHash, convert it into a real geometry, and issue a notification to subscribers including that information.

1) Start FME Workbench
Start FME Workbench. Use the Generate Workspace dialog to create a translation as follows:

Reader Format Geohash (Beta)
Reader Dataset *C:\FMEData\readme.txt*

Writer Format SpatiaLite
Writer Dataset *C:\FMEData\Output\ServerTraining\Earthquakes.sl3*

The selected source (readme.txt) isn't a Geohash format file, but that doesn't matter. In FME Server the data will come from the notification and therefore the source will be ignored. Geohash works for a notification because it is a custom format wrapped around the Text File reader.

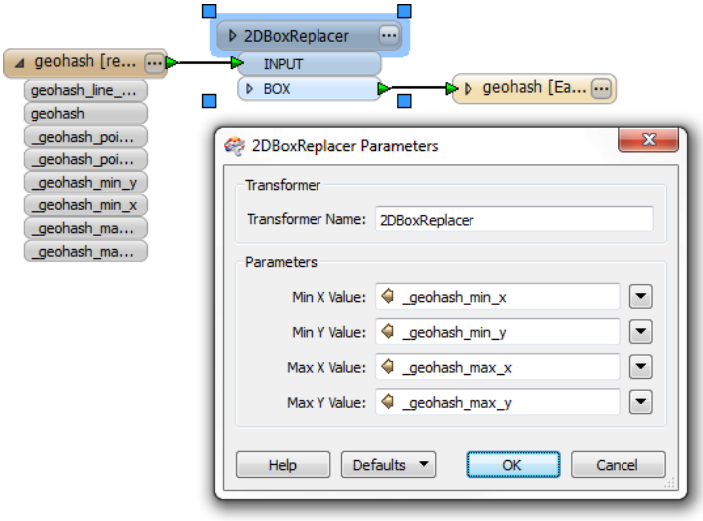
2) Add 2DBoxReplacer
Geohash coordinates can be decoded not just into a single point, but into a rectangular area of interest. Four attributes are provided by the reader to transmit this information and we will use them to create a proper geometry.

Add a 2DBoxReplacer transformer between the reader and writer feature types.

Open the parameters dialog.

Set the parameters to be set by the attribute values:

_geohash_min_x,
_geohash_min_y,
_geohash_max_x,
_geohash_max_y

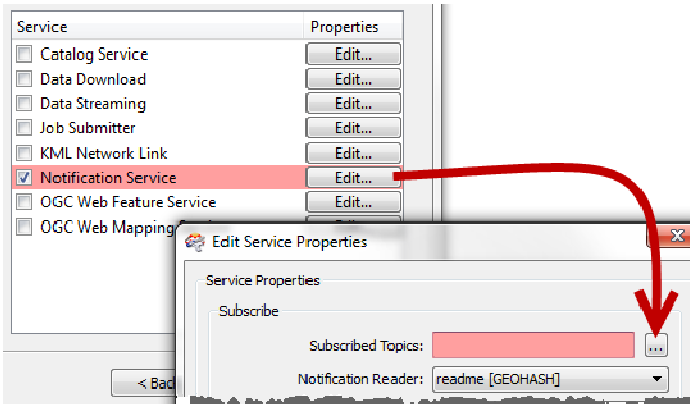


3) Save and Publish Workspace
Save the workspace and then publish it to FME Server.

An initial warning dialog will suggest the Geohash reader needs an extra file (custom format) to be uploaded. However, since the custom format is already installed on FME Server, this warning can be ignored.

Save the workspace to the Training repository as EarthquakeHash.fmw

Register it with the Notification Service. Click the Edit button for the service properties and then click the browse button for Subscribed Topics.

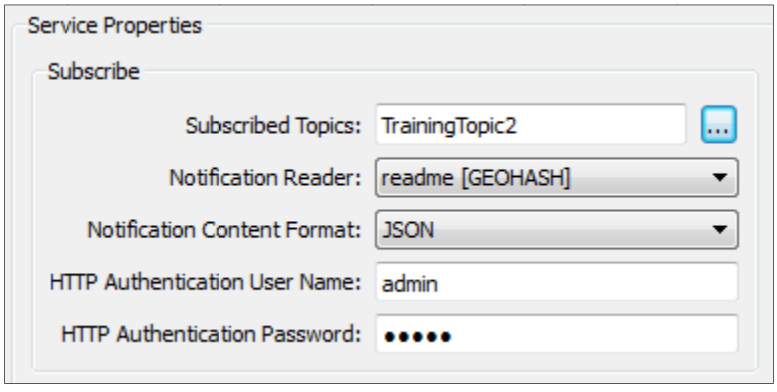


In the following dialog, click the New Topic button to create a new topic named: TrainingTopic2

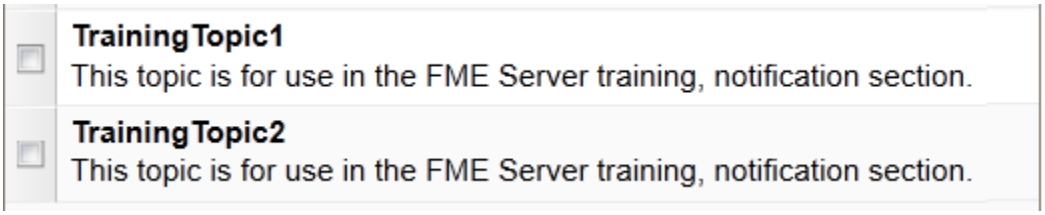
Back in the dialog “Edit Service Properties”, set the Notification Reader to the GEOHASH reader. Fill in the Authentication parameters as usual.

The Notification Content Format can be left as JSON. In this case it does not matter because it is a Geohash coordinate being treated as plain text.

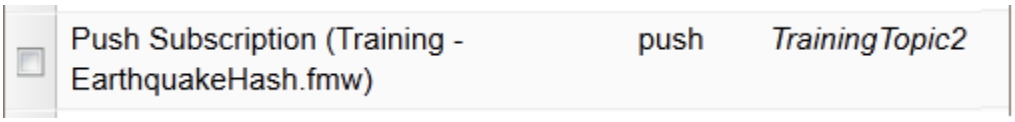
The section for Publish parameters does not need to be filled in, as the workspace is a Subscription only and not a Publisher.



4) Examine Setup
In the FME Server web interface, browse to the Notifications section and click on the Topics tab. You will see the newly created topic, TrainingTopic2:



Click the Subscriptions tab and you will see a Subscription created for the published workspace that is triggered by the TrainingTopic2 topic:



The Subscription is a "push" type, whose target URL will point to the workspace itself, similarly to:

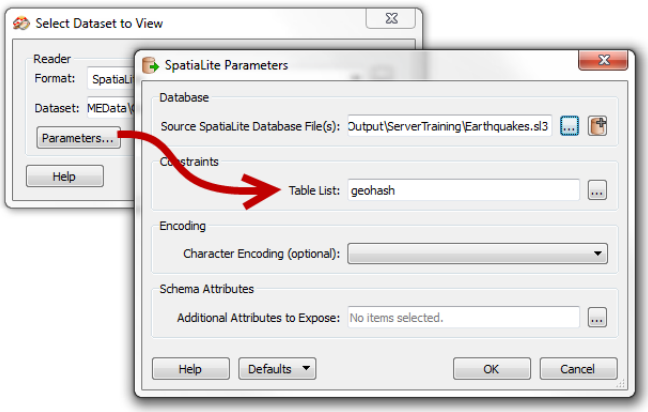
`http://<myFMEServer>/fmenotification/Training/EarthquakeHash.fmw`

5) Trigger Topic
Open the FME Server REST web interface. Log in and browse to the topic TrainingTopic2.

Enter "ezs42" into the Notification Content field (this is a valid Geohash coordinate) and click Publish. A pop-up dialog will notify you that the publication has succeeded.

6) Check Output
Check the output. It should be located at: *C:\FMEData\Output\ServerTraining\Earthquakes.sl3*
Open the dataset inside the FME Universal Viewer to ensure it contains valid data. It should show the bounding box of the defined point, taking the precision of its geohash into account.

When inspecting SpatiaLite data be sure to click the Parameters button and choose the table to be inspected, otherwise no data will be read into the Viewer!



7) Re-Trigger Topic
Trigger the topic again, this time using a geohash of “ep131”.

Refresh the data in the FME Universal Viewer and check the contents. There should now be two different geohash areas displayed in the main window.

There are now two features because the workspace is set up to not overwrite the existing SpatiaLite database (if you only see one feature then check the writer parameter in your workspace and then the FME Server job history if necessary).

Adding to (rather than overwriting) a dataset is a very useful technique for preserving notification data. This way you can keep a record of all notifications over time. Any file-based format that has the ability to add data – usually signified by an “Overwrite – Y/N” parameter – can be used, as could a database format.

For the ultimate in web-based solutions, try writing notification data to a Google Fusion table!

Advanced Task
NB: *For this task you need access to either an email server or cloud-based email such as Gmail.*

So far this workspace responds to a notification, and then transforms the incoming data and writes it out, but it doesn't alert any end users to the event. To do this requires the workspace to also act as a Publication, as well as a Subscription.

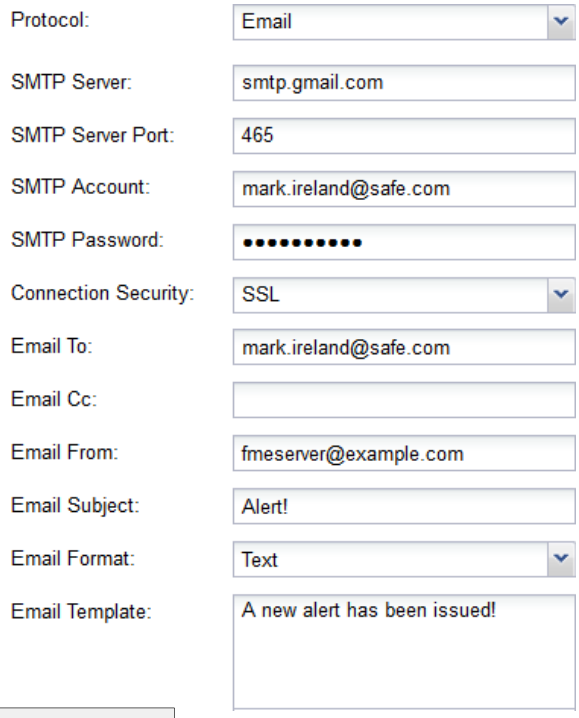
For this advanced task create a new topic (say TrainingTopic3) and a new Subscription (TrainingSubscription3). The Subscription should subscribe to the new topic and be of the type Email.

When prompted enter details of a known email server or account

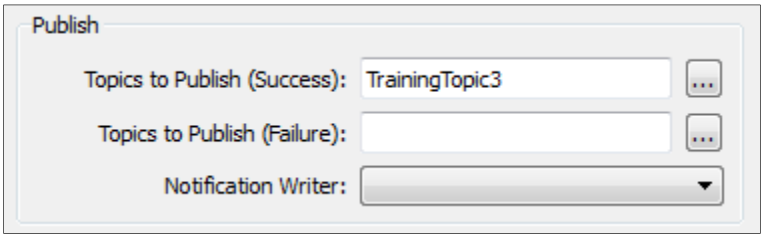
The settings for configuring an Email notification using the Gmail SMTP Server are:

SMTP Server	smtp.gmail.com
SMTP Server Port	465
SMTP Account	<gmail address>
SMTP Password	<gmail password>
Connection Security	SSL

Then enter the email To/From and a Subject/Content



If you set the Email Format to HTML then the template can contain HTML content!



Publish the workspace back to FME Server. This time, add a topic to publish to (success) which will be TrainingTopic3.

Back in the REST web interface, publish a geohash to TrainingTopic2. This will (via the workspace) cause an email to be sent out with an alert.

If the email fails to arrive, check the log file at: <FMEServer>\Logs\notifier\subscribers\email

If it reports that gmail.smtp.com is unknown, then log in manually at mail.google.com as this may help clear the problem.

Very Advanced Task
If you have time, add a new Writer to the workspace to create proper content for the email. Republish the workspace selecting this as the "Notification Writer".

Check the FME Server reference manual for info on the Email template language and keywords.

Spatial Selection by Polygon

Besides Bounding Boxes, features can also be selected spatially by a polygon.

Previous examples of allowing spatial selection involved bounding boxes. A more complex solution than a bounding box is selection of an area of interest by a polygon.

The polygon may be drawn on a web map or selected from a predefined set of areas such as administrative boundaries. For example, the user may select a State or Province and want to download the data from a set of layers within that State or Province.

Passing FME a bounding box as a set of parameters is simple enough, but passing a complex shape that way would be tricky. The solution is the *FeatureReader* transformer.

The FeatureReader Transformer
The *FeatureReader* transformer acts like a reader within a workspace, but because it is a transformer it can work on information that has already been part-transformed by the workspace.

The *FeatureReader* is activated by an incoming **Initiator** feature. It is a combination of the geometry of this feature, and parameters in the transformer, that tell FME what data to read.

For example, the initiator feature might be a polygon, and the transformer parameters instruct FME to read only the features within that polygon.

Alternatively the parameters might use an attribute on the initiator feature to perform a simple (non-spatial) query, to decide what features to read.

The transformer also has a “Feature Types to Read” type of parameter, which can even be defined as an attribute on the incoming initiator feature.

Terminology: The Initiator feature can be thought of as representing a query and in fact early terminology called it the “Query” feature.

For FME Server, there are two probable scenarios for creating an initiator feature:

- A *Creator* transformer creates an Initiator feature. By publishing the geometry parameter of the *Creator* the workspace can accept a polygon as a reader parameter.
- A true FME reader reads a polygon selected from a list on FME Server and passed as a parameter; for example a state or province boundary. The polygon is passed to the *FeatureReader* as the Initiator feature.

The FeatureReader replaces both the ArcSDEQuerier and the OracleQuerier transformers.

Example 20: Spatial Selection by Polygon	
Scenario	FME user; City of Interopolis, Planning Department
Data	Geohash, Roads (MIF/MID)
Overall Goal	Create a workspace for polygon clipping
Demonstrates	Querying data with the *FeatureReader* transformer
Starting Workspace	Create from scratch
Finished Workspace	C:\FMEData\Workspaces\ServerManual\Session5Example6Complete.fmw

In the previous examples, a Geohash reader was used to define an area of interest. Now that area can be used to select features within that area – for example features that fall inside the sphere of influence of an earthquake.

1) Start Workbench
Start Workbench (if necessary). Similarly to the previous example, create a workspace with:

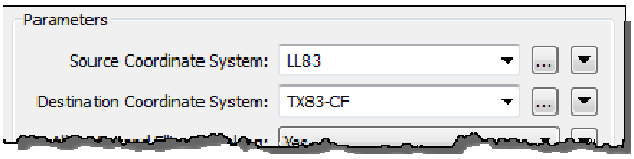
Reader Format Geohash (Beta)
Reader Dataset *C:\FMEData\readme.txt*

Writer Format SpatiaLite
Writer Dataset *C:\FMEData\Output\ServerTraining\AffectedFeatures.sl3*

Again, the selected source (readme.txt) isn't a Geohash format file, but that doesn't matter.

2) Add Reprojector Transformer
Add a Reprojector transformer.

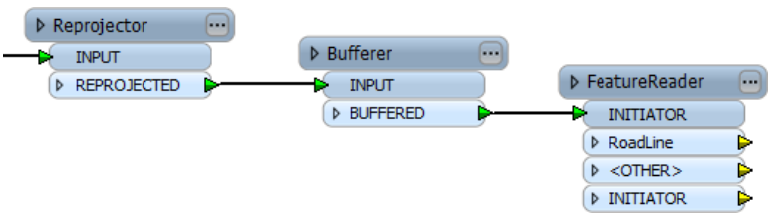
Open the parameters dialog and set it to reproject the incoming geohash feature from LL83 to TX83-CF (EPSG:2277)



3) Add Bufferer Transformer
Add a Bufferer transformer.

Open the parameters dialog and set it to buffer the geohash feature by 5000feet.
This will represent the area affected by the earthquake (it was small and very localized)!

4) Add FeatureReader Transformer
Add a FeatureReader transformer.



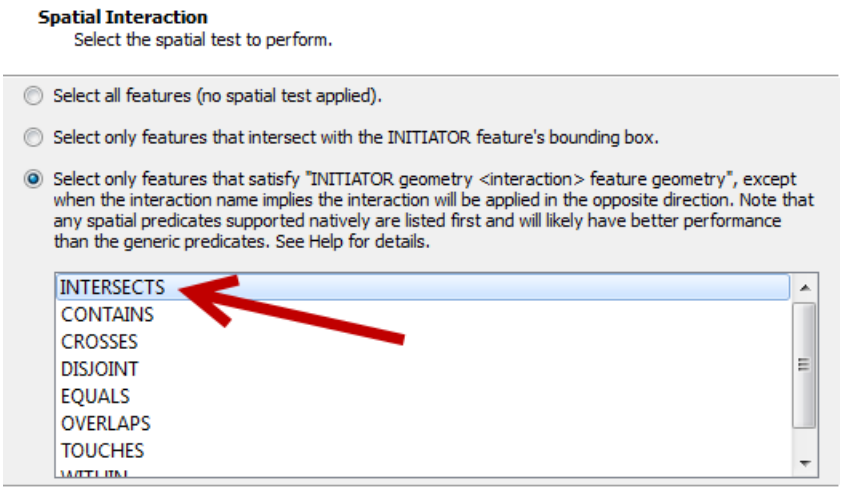
5) Set Parameters
Open the FeatureReader parameters dialog. In fact, it is a series of dialogs (wizard) rather than just one dialog.

For the data to query, choose the dataset:

Reader Format MapInfo MIF/MID
Reader Dataset *C:\FMEData\Data\Roads\RoadLine.mif*

For the Query Operation, leave it as “Query the feature types specified on the previous page”.

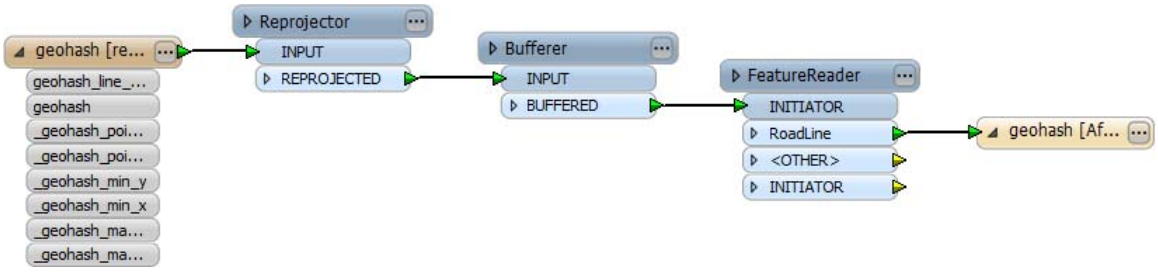
For the Spatial Interaction, select the third option (to compare geometries) and choose the action INTERSECTS.



For the Advanced Options use “Keep Result Attributes” and “Keep Result Geometry”.

Click **Finish** to close the wizard.

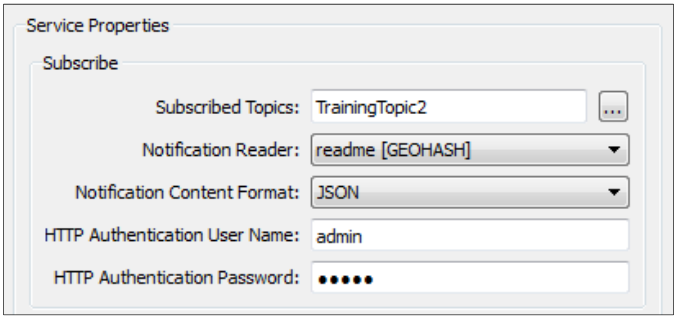
6) Tidy Schema
Connect the transformers between the Reader and Writer feature types. Now remove all user attributes from the writer feature type (they won’t be needed).



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

Notice how the files for the Roads data are automatically uploaded. As we are using a local server there is no need to do this, so uncheck the boxes (you will get a warning).

Register the workspace with the Notification Service and choose the subscribed topic to be TrainingTopic2. Select the Geohash reader as the Notification Reader.



8) Trigger Notification
Open the FME Server REST API web interface.

As before, log in and browse to the topic TrainingTopic2.

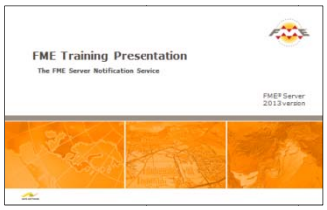
Enter “9v6kpv” into the Notification Content field (this is a valid Geohash coordinate) and click Publish. A pop-up dialog will notify you that the publication has succeeded.

9) Check Output
Check the output. It should be located at: *C:\FMEData\Output\ServerTraining\AffectedFeatures.sl3*

Open the dataset inside the FME Universal Viewer to ensure it contains valid data. It should show all of the road features within the earthquake affected area.

This has demonstrated how the data being read can be restricted to a particular spatial area, through use of the FeatureReader transformer.

Module Review



This module introduced you to FME Server's Notification Services.

What You Should Have Learned from this Module
The following are key points to be learned from this module.

Theory

- A notification is a message (sometimes called an "alert") that informs someone or something that a particular event has happened. Incoming notifications alert FME Server to an event that has taken place elsewhere. Outgoing notifications alert users to an event that has taken place on FME Server.
- Incoming notifications are received from Publishers or Publications
- Outgoing notifications are sent to Subscribers or Subscriptions
- Protocols are the method of communication (e.g. email, SMS, JMS, etc)
- Topics are keywords that define a notification subject
- Selection of source data by complex area can be achieved using the FeatureReader transformer

FME Skills

- The ability to create Subscriptions to send notifications to
- The ability to trigger notifications from either the REST API or a workspace
- The ability to use a workspace as a subscriber to receive information
- The ability to use the FeatureReader transformer to deliver data for a particular spatial boundary

Course Wrap-Up

Product Information and Resources

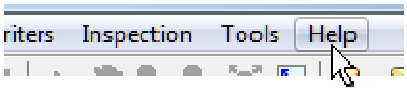
Although your FME training is now at an end, there is a good supply of expert information available for future assistance.

Safe Software Web Site
Our 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.

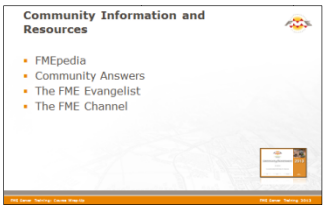
Safe Support Team
Behind FME are passionate, fun, and knowledgeable experts, ready to help you succeed, with a support and services philosophy built on the principle of knowledge transfer.

Safe Software Blog
The Safe Software blog ("It's All About Data") provides thoughts on spatial data interoperability from the folks who make it their passion.

FME Manuals and Documentation
Use the Help function in FME Workbench to access context-sensitive help, and various manuals and documentation for FME Desktop.



Community Information and Resources



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

FMEpedia
This encyclopaedia is our FME community website.
It provides one-stop access to all FME community resources. You'll find a wealth of FME information, including tips, tricks, and FAQs. Information is provided by both Safe staff and FME users.

fmepedia.safe.com

FME Talk
The FME community group (FME Talk) has 1,500 members who post FME related messages and questions and share in answering other users' questions.

FME Community Answers

The FME Evangelist
The FME Evangelist delivers inside news about cutting edge examples and the latest developments in FME functionality.

fmeevangelist.com

The FME Channel
This FME YouTube channel is for those demos that can only be properly appreciated through a screencast or movie.

www.youtube.com/user/FMEchannel

Course Feedback

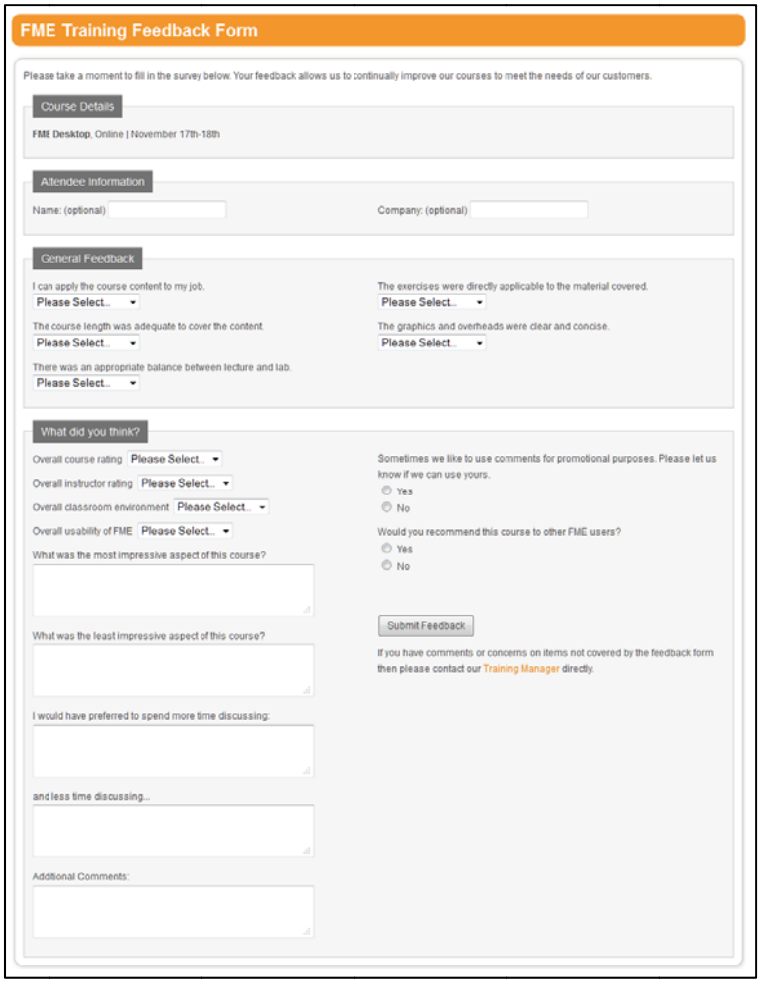
The format of this training course undergoes regular changes prompted by comments and feedback from previous courses.

There's one final Q+A to go – and this time you'll be telling us the answers!

Safe Software Inc. greatly values feedback from training course attendees. This is your chance to tell us what you really think about how well we're meeting your training goals.

The current course structure has been determined by attendee comments and we appreciate your feedback more than ever.

You'll be sent a link to a feedback form after your course.



Certificates

All FME training course attendees receive a certificate.

Congratulations!

With the presentation of your certificate of achievement, you have now officially completed the FME Server 2013 Training Course.

Thank You

Thank you for attending this FME training course.

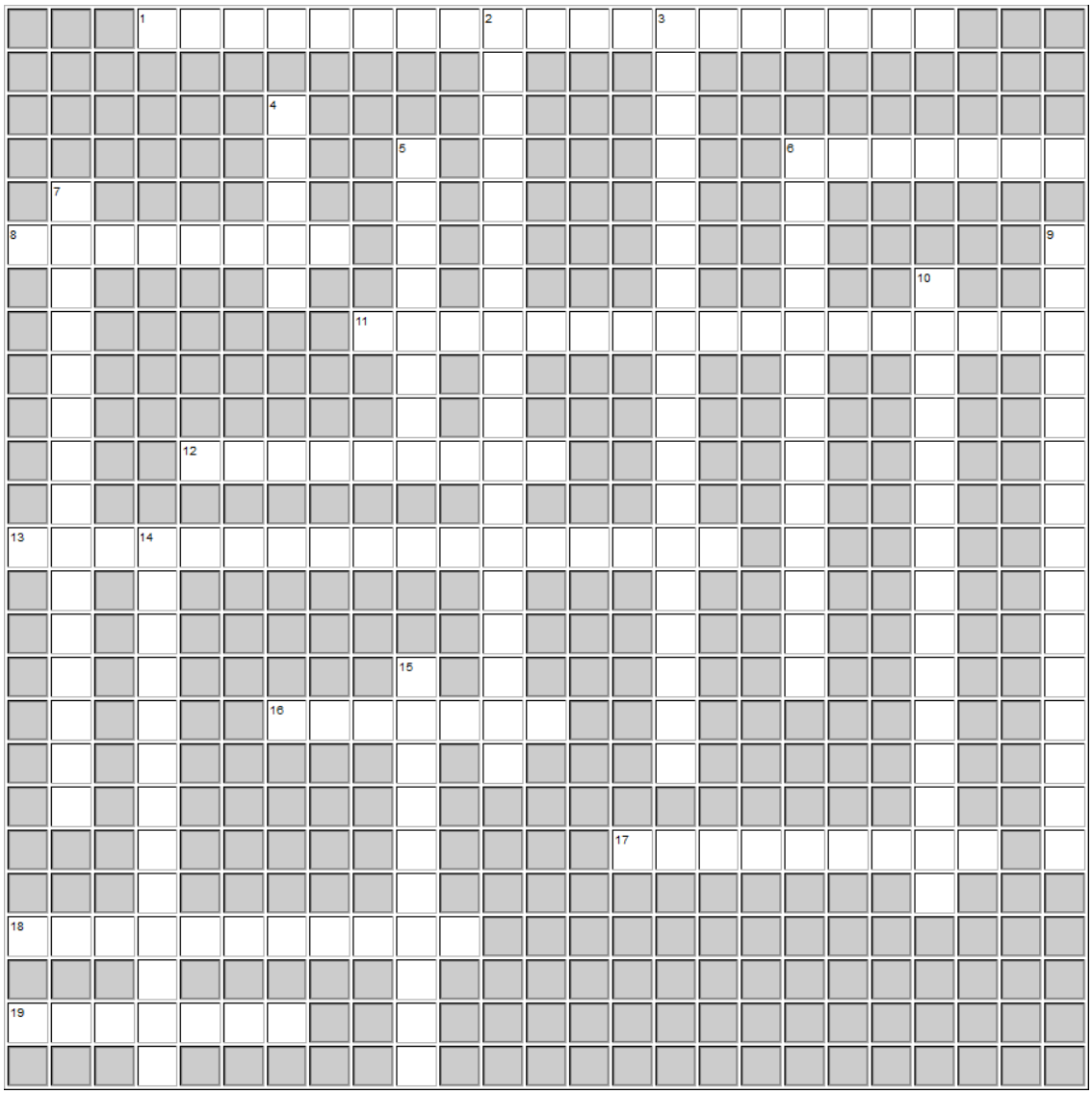
All of us funky dudes at Safe Software Inc. wish you good luck with your use of FME.

Congratulations!

Judge GIS says…

"By reading this far you have been convicted of first degree FME training.
I hereby sentence you to carry out this cryptic crossword!

Be warned – it may well be hard labor!"



Across

1. Pressing F11 helps you do this expression (3-3-6-7)
6. New reader implies where you would find a church bell. In Europe! (7)
8. Is it a bird? Is it a plane? Is it a typo? No! Just an heroic new format for 2012 (8)
11. In short, this new format might be called a Felid (6-7-4)
12. This rank transformer is nominally changed for 2012 (9)
13. Eccentric firemen veto fires. Right way to communicate information (3-6-8)
16. Traffic cop issuing a ticket, say, now supports raster data (7)
17. Ape cult I'd press to copy a TestFilter clause (9)
18. This amalgamated transformer sounds like the ingredient for making paint dry (5-6)
19. This new format helps you go straight to where the treasure is buried (6-1)

Down

2. Felonious bagels got mixed up with new cloud format (6-6-6)
3. A 2012 technique creates multiple procedures, and they won't even meet at infinity! (8-10)
4. The nom-de-plume for a transformer, or a European rock strata (5)
5. A universal truth embraces on a newly supported projection (8)
6. An interrogator in the queue initially has no ports (6-7)
7. Sounds like a collector of boats; either concave or convex (4-11)
9. Abnormal city divorce rate allows security to harmonize (6-9)
10. It's data all. Thoughts on spatial data interoperability (3-3-5-4)
14. One tiny typo and this new transformer drops element 47 from the periodic table (6-7)
15. Father Christmas, say, got deprecated for 2012 (4-6)