



Red Hat JBoss Fuse 6.0

Using the Fuse Developer Tools

Developing and Testing Applications

Red Hat JBoss Fuse 6.0 Using the Fuse Developer Tools

Developing and Testing Applications

JBoss A-MQ Docs Team

Content Services

fuse-docs-support@redhat.com

Legal Notice

Copyright © .

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide describes how to use Red Hat JBoss Fuse Plugins for Eclipse and Red Hat JBoss Fuse IDE, which provides developer tools designed to increase your productivity when designing, developing, and testing your integration applications.

Table of Contents

PART I. DEVELOPING APPLICATIONS	6
CHAPTER 1. DESIGN TIME TOOLING	7
1.1. THE ROUTE EDITOR	7
1.2. CREATING PROJECTS, ROUTES, AND TEST CASES	9
1.3. EXERCISING ROUTES	10
CHAPTER 2. CREATING A NEW FUSE PROJECT	12
OVERVIEW	12
PROCEDURE	12
RESOLVING MAVEN DEPENDENCY ERRORS	15
RELATED TOPICS	15
CHAPTER 3. CREATING A NEW CAMEL XML FILE	16
OVERVIEW	16
PROCEDURE	16
RELATED TOPICS	17
CHAPTER 4. EDITING A ROUTING CONTEXT IN THE ROUTE EDITOR	18
4.1. ADDING ROUTES TO THE ROUTING CONTEXT	18
4.2. ADDING PATTERNS TO A ROUTE	19
4.3. CONNECTING PATTERNS TO MAKE A ROUTE	20
4.4. CONFIGURING A PATTERN	21
4.5. REARRANGING PATTERNS ON THE CANVAS	22
4.6. REMOVING PATTERNS FROM A ROUTE	23
4.7. DISCONNECTING TWO PATTERNS	23
4.8. DELETING A ROUTE	24
4.9. ADDING BEANS AND CONFIGURATION	24
4.10. CONFIGURING THE ROUTE EDITOR	25
CHAPTER 5. CREATING A NEW APACHE CAMEL JUNIT TEST CASE	27
OVERVIEW	27
PROCEDURE	27
RELATED TOPICS	29
CHAPTER 6. THE SOURCE VIEW	31
CHAPTER 7. RUNNING ROUTES INSIDE RED HAT JBOSS FUSE PLUGINS FOR ECLIPSE	32
7.1. RUNNING ROUTES USING MAVEN	32
7.2. RUNNING ROUTES AS A LOCAL CAMEL CONTEXT	32
7.3. WORKING WITH RUNTIME PROFILES	33
CHAPTER 8. DEPLOYING PROJECTS TO A CONTAINER	39
8.1. USING A CONTAINER'S DEPLOY FOLDER	40
8.2. DEPLOYING A PROJECT TO A JMX CONNECTED CONTAINER	42
8.3. DEPLOYING A PROJECT TO A FABRIC CONTAINER	44
PART II. DEBUGGING	46
CHAPTER 9. DEBUGGING TOOLING	47
9.1. THE FUSE INTEGRATION PERSPECTIVE	47
9.2. INFRASTRUCTURE MONITORING	48
9.3. ROUTE DEBUGGING	49
9.4. JMS BROWSING	49

CHAPTER 10. OPENING THE FUSE INTEGRATION PERSPECTIVE	51
OVERVIEW	51
OPENING THE PERSPECTIVE	52
RELATED TOPICS	52
CHAPTER 11. THE FUSE JMX NAVIGATOR	53
11.1. VIEWING PROCESSES IN JMX	53
11.2. ADDING A JMX SERVER	54
CHAPTER 12. VIEWING A COMPONENT'S PROPERTIES	56
OVERVIEW	56
PROCEDURE	56
CHAPTER 13. BROWSING MESSAGES	57
OVERVIEW	57
PROCEDURE	57
RELATED TOPICS	57
CHAPTER 14. TRACING MESSAGES THROUGH ROUTES	58
14.1. CREATING TEST MESSAGES FOR ROUTE TRACING	58
14.2. ACTIVATING ROUTE TRACING	59
14.3. TRACING MESSAGES THROUGH A ROUTE	59
14.4. DEACTIVATING ROUTE TRACING	60
CHAPTER 15. MANAGING JMS DESTINATIONS	62
15.1. ADDING A JMS DESTINATION	62
15.2. DELETING A JMS DESTINATION	62
CHAPTER 16. MANAGING ROUTING ENDPOINTS	64
16.1. ADDING A ROUTING ENDPOINT	64
16.2. DELETING A ROUTING ENDPOINT	64
CHAPTER 17. EDITING RUNNING ROUTES	66
OVERVIEW	66
PROCEDURE	66
RELATED TOPICS	66
CHAPTER 18. MANAGING ROUTING CONTEXTS	68
18.1. SUSPENDING A ROUTING CONTEXT	68
18.2. RESUMING A ROUTING CONTEXT	68
18.3. SHUTTING DOWN A ROUTING CONTEXT	69
CHAPTER 19. MANAGING SERVERS	70
19.1. ADDING A SERVER	70
19.2. STARTING A SERVER	72
19.3. STOPPING A SERVER	72
19.4. DELETING A SERVER	73
PART III. WORKING WITH FABRICS	74
CHAPTER 20. THE FUSE FABRIC PERSPECTIVE	75
FABRIC NAVIGATOR	75
DIAGRAM VIEW	75
LOG VIEW	76
MESSAGES VIEW	76
PROPERTIES VIEW	76

JMX VIEWER	76
CHAPTER 21. SETTING UP A FABRIC ENVIRONMENT	77
OVERVIEW	77
OPENING THE FABRIC PERSPECTIVE	77
RELATED TOPICS	77
CHAPTER 22. SPECIFYING AND CONNECTING TO A FABRIC	78
22.1. ADDING FABRIC DETAILS	78
22.2. CONNECTING TO A FABRIC	79
22.3. DISCONNECTING FROM A FABRIC	80
22.4. EDITING A FABRIC'S DETAILS	80
22.5. DELETING A FABRIC'S DETAILS	81
CHAPTER 23. WORKING WITH FABRIC CONTAINERS	82
23.1. CREATING A NEW CHILD CONTAINER	82
23.2. CREATING A CONTAINER ON A REMOTE HOST	84
23.3. CREATING A NEW CONTAINER ON A CLOUD	86
23.4. STARTING A CONTAINER	90
23.5. CHANGING A CONTAINER'S PROFILES	91
23.6. STOPPING A CONTAINER	91
23.7. DELETING A CONTAINER	92
CHAPTER 24. WORKING WITH FABRIC PROFILES	94
24.1. CREATING A NEW PROFILE	94
24.2. DELETING A PROFILE	98
CHAPTER 25. WORKING WITH VERSIONS	99
25.1. CREATING A NEW VERSION OF A PROFILE	99
25.2. SETTING A CONTAINER'S VERSION	100
CHAPTER 26. CREATING A FABRIC IN THE CLOUD	101
26.1. ADDING CLOUD DETAILS	101
26.2. SPECIFYING FABRIC DETAILS	102
APPENDIX A. WIZARD FIELD REFERENCES	105
NAME	105
PROPERTIES	105
RELATED TOPICS	105
NAME	105
PROPERTIES	105
RELATED TOPICS	106
NAME	106
PROPERTIES	106
RELATED TOPICS	107
NAME	107
PROPERTIES	107
RELATED TOPICS	107
NAME	107
PROPERTIES	107
RELATED TOPICS	108
NAME	108
PROPERTIES	108
RELATED TOPICS	108
NAME	109

PROPERTIES	109
RELATED TOPICS	109
NAME	109
PROPERTIES	109
RELATED TOPICS	110
NAME	110
PROPERTIES	110
RELATED TOPICS	110
NAME	110
PROPERTIES	110
RELATED TOPICS	111
NAME	111
PROPERTIES	111
RELATED TOPICS	111
NAME	112
PROPERTIES	112
RELATED TOPICS	112
NAME	113
PROPERTIES	113
RELATED TOPICS	113
NAME	114
PROPERTIES	114
RELATED TOPICS	114
NAME	114
PROPERTIES	114
RELATED TOPICS	115
NAME	115
PROPERTIES	115
RELATED TOPICS	116
APPENDIX B. DEVELOPER TOOLS PREFERENCES	117
NAME	117
PROPERTIES	117
RELATED TOPICS	117
NAME	117
PROPERTIES	117
RELATED TOPICS	118
NAME	118
PROPERTIES	118
RELATED TOPICS	118
NAME	119
PROPERTIES	119
RELATED TOPICS	119

PART I. DEVELOPING APPLICATIONS

	7/22/11	EMJ
rearranged so that deployment is at the bottom	7/22/11	EMJ
smooshed a bunch of editing topics into one chapter	8/11/11	EMJ
dispersed the configuration stuff to more topical areas	8/15/11	EMJ
added intro material describing a basic work flow	1/4/12	JCM
version 2.1 updates		

The Red Hat JBoss Fuse Plugins for Eclipse, running inside Red Hat JBoss Developer Studio, simplify and streamline the process of developing integration applications by providing developer tools that are specifically designed to work with:

- Red Hat JBoss Fuse
- Red Hat JBoss A-MQ
- Apache Camel
- Apache CXF
- Apache Karaf

The developer tools streamline the process at all stages of application development:

1. Create a Maven project for your application.

The developer tools load all of the relevant Maven archetypes for creating integration projects using the Red Hat supported Apache projects.

2. Add new pieces of logic and functionality to an application.

The developer tools have a wizard for creating Apache Camel context files.

3. Edit the integration logic.

The developer tools have a visual route editor that makes editing Apache Camel routes as easy as dragging and dropping route components.

4. Test the application.

The developer tools include testing tools that provide the full gamut of testing capabilities including:

- creating JUnit test cases for Apache Camel routes
- JMX analysis of running components
- message tracing through Apache Camel routes

5. Deploy the application.

The developer tools can deploy applications to a number of containers.

CHAPTER 1. DESIGN TIME TOOLING

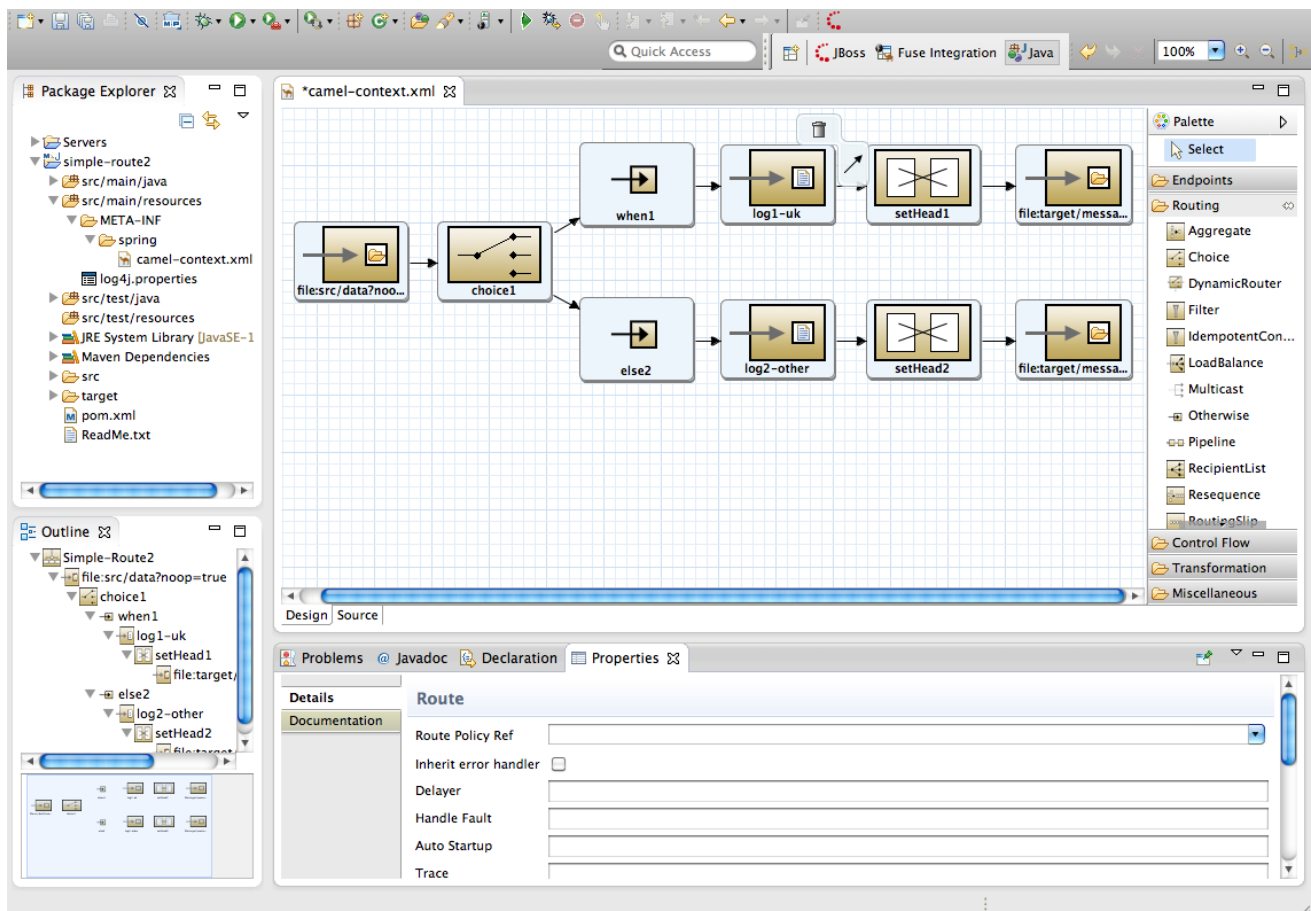
1.1. THE ROUTE EDITOR

Overview

You access the developer tools by opening the Java Perspective, and then selecting **File** → **New** → **Fuse Project** on the menu bar.

Creating a new Fuse project provides access to the route editor, the main design time tooling, shown in [Figure 1.1](#).

Figure 1.1. route editor



It consists of four main areas:

- Canvas—the large grid area on which routes are constructed
- Palette—the pane to the right of the canvas from which Enterprise Integration Patterns (EIPs) are selected
- Project view—the pane on the left side of the canvas, which can display multiple views of the active project. The pane defaults to **Package Explorer**, which displays the project's folders and files in an hierarchical tree format.



NOTE

We recommend that you drag **Outline** view from the upper right side of the workspace to the lower left side, beneath **Package Explorer**, to provide optimal space for the route editor. **Outline** view displays, in an outline of EIP icons, the contents of the current `<camelContext>` element in the routing context file.

- **Properties editor**—the editor in which you configure the selected node's properties. It opens in a tab in the pane below the canvas.

Canvas

The canvas is route editor's workbench and where you do most of your work. It displays a graphical representation of a route, which is made up of one or more connected EIP patterns (called nodes once they are placed on the canvas). Because the canvas displays only one route at a time, it delineates one route from another. You can add additional routes to your project, but each is displayed separately on its own slice of canvas.

Patterns are typically placed on the canvas by dragging them from the palette. Once on the canvas, a node's properties can be edited in the **Properties** editor. To do so, you need only select a node to open and populate the **Properties** editor with the properties that apply to that node. You can also edit a route's properties by clicking the canvas, which opens and populates the Properties editor with the properties that correspond to a single route in the routing context file.



NOTE

Clicking the **Source** tab at the lower, left of the canvas displays the XML code generated for the route, which you can then edit. The edits you make in **Source** view appear on the canvas when you switch back to **Design** View.

Palette

The Palette contains all of the patterns needed to construct a route. The Palette groups the patterns into categories according to function:

- **Endpoints**—patterns that start or end a route
- **Routing**—patterns that direct the flow of messages based on specified criteria
- **Control Flow**—patterns that behave like control functions in a programming language. For example, some define loops, some handle error conditions, some handle transactions, and so on.
- **Transformation**—patterns that change the contents of a message as it passes through a route
- **Miscellaneous**—patterns that control the environment in which a route executes. For example, the Threads pattern specifies the number of threads available.

Package Explorer

Package Explorer provides access to all of the components that make up a project. It provides the tools for managing, configuring, running, and debugging your project.

Outline view

Outline view provides a graphical outline of your route components. It enables you to switch easily between routes in a multiroute project. Clicking on a route in **Outline** view displays it on the route editor's canvas.

Outline view also provides a viewing portal when the route extends beyond the viewable area on the canvas. Sliding the view portal in Outline view over outlier nodes moves the canvas accordingly to bring those nodes into view.

Properties editor

You use the **Properties** editor to configure the route and each node in it. Each type of node has a set of properties that can or must be set. The Properties editor opens populated with fields that apply to the selected node.


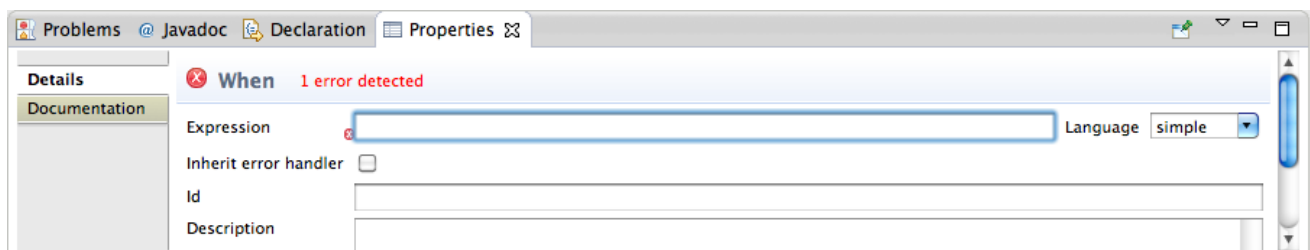
The **Properties** editor makes it clear which fields are required.  marks any required property that is not set, and the **Properties** editor displays an error message stating the number of required properties that still must be set, as shown in [Figure 1.2](#).

Figure 1.2. Properties editor's error reporting



The **Properties** editor's **Documentation** tab displays information about how to configure the selected node.

1.2. CREATING PROJECTS, ROUTES, AND TEST CASES

Overview

While you'll spend the majority of your time using the route editor, the developer tools include wizards that facilitate creating the following:

- Red Hat JBoss Fuse projects
- Apache Camel context files
- Apache Camel-specific JUnit tests

Fuse projects

The developer tools leverage the m2e plug-in's ability to execute Maven archetypes to create starting point projects for working on Apache Camel routes. The **New Fuse Project** wizard displays the Apache Camel Maven archetype options. Selecting one creates a project that is properly configured for developing Apache Camel applications.

For more information, see [Chapter 2, Creating a New Fuse Project](#).

Apache Camel context files

Apache Camel context files are the key piece of any project. They contain the routing rules that form the backbone of an Apache Camel application. The developer tools have a simple wizard that creates new Apache Camel context files.

For more information, see [Chapter 3, *Creating a New Camel XML file*](#).

Apache Camel test cases

The developer tools provide a wizard that generates JUnit code for testing your routes. The wizard makes it easy to select the assertions to test. It also simplifies the process of testing particular endpoints in a route.

For more information, see [Chapter 5, *Creating a new Apache Camel JUnit test case*](#).

Fuse messages

The developer tools provide a wizard that creates a message that you can use to test a route. You drop the message on Camel endpoints or ActiveMQ destinations to track its progress through the route. The wizard creates a skeletal message, which you can edit to customize the content of both the message body and the message header.

For more information, see [Section 14.1, “Creating test messages for route tracing”](#).

1.3. EXERCISING ROUTES

	7/22/11	EMJ
added entry for Tomcat deployment		
	09/09/12	EMJ
removed subscription restriction comments		

Overview

In addition to simplifying the process of creating and editing routes, the developer tools provide tools to make it easier to run your routes and deploy them into test environments. Using the developer tools you can:

- run a route using Maven
- run a route as a local Apache Camel context
- deploy into a running OSGi container

Using Maven

You can run your route with Maven, using Maven's standard goals. This method closely simulates how an Apache Camel project would run outside of the developer tools.

See [Section 7.3, “Working with runtime profiles”](#).

Running as a local Apache Camel context

You can run all routes in a context directly inside the developer tools without setting up a Maven runtime profile. You can run routes locally with or without running validation tests. Running a context directly inside the developer tools does not require that your project be a Maven project.

Deploying to OSGi

You can also deploy routes into a running OSGi container. The developer tools support deployment into these OSGi containers: Red Hat JBoss Fuse v6.x, Apache ServiceMix v4.x, and Apache Karaf v2.x.

See [Chapter 8, *Deploying Projects to a Container*](#).

CHAPTER 2. CREATING A NEW FUSE PROJECT

7/22/11

EMJ

Changed the archetype note to include blueprint and clarify that the limitation is just for the designer

7/12/12

EMJ

Removed comments about determining supported project types by their description

7/12/12

EMJ

Clarified some of the steps

OVERVIEW

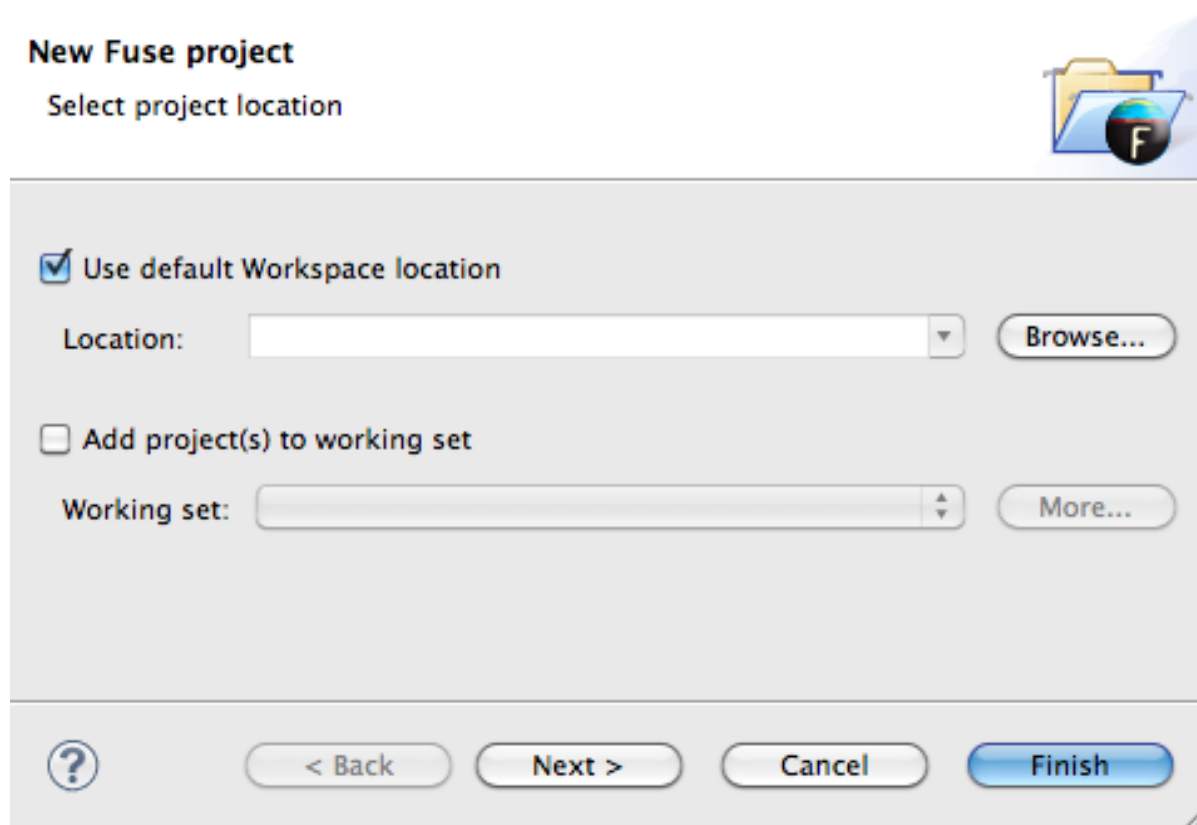
The developer tools use Maven archetypes to generate a project with all of the dependencies preconfigured. The archetypes also create the POM file needed to run and deploy your application, as well as sample code and data to get you started.

PROCEDURE

To create a Fuse project:

1. Select **File** → **New** → **Fuse Project** to open the **New Fuse Project** wizard shown in [Figure 2.1, “New Fuse Project Wizard's Project Location Page”](#).

Figure 2.1. New Fuse Project Wizard's Project Location Page



The wizard opens with the **Use default workspace location** option selected on the location page.

2. Specify the location where the data for the project will be stored.
 - o To use the default workspace select **Use default workspace location**.

- o To use an alternative location deselect **Use default workspace location** and specify a new location in the provided field.

Clicking  opens a file browser.

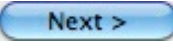
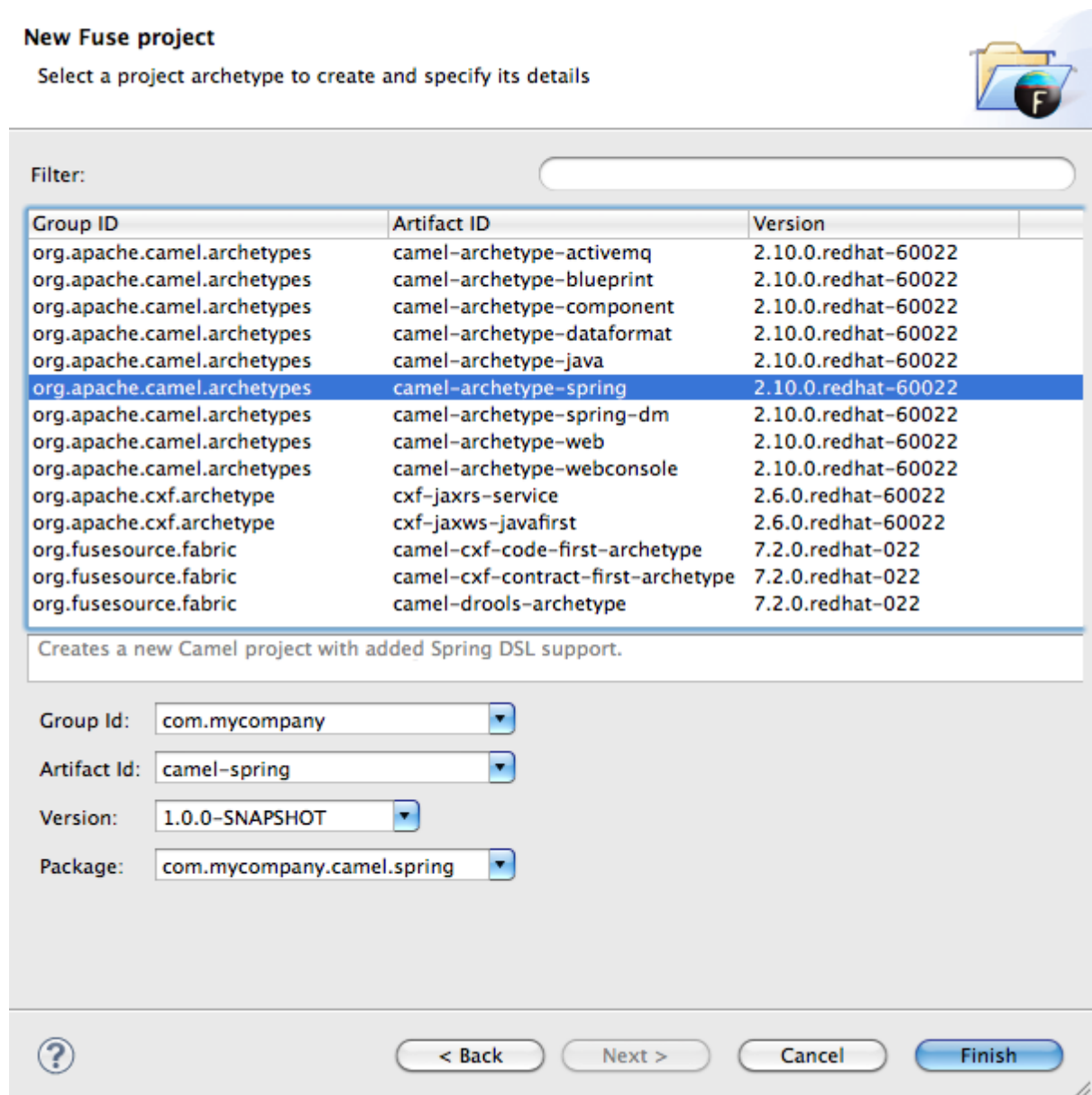
3. If you want to add the new project to an Eclipse working set, select **Add project(s) to working set** and enter the name of the working set.
4. Click  to open the **New Fuse Project** details page shown in [Figure 2.2](#).

Figure 2.2. New Fuse Project wizard's details page



5. Select a project type from the list.



NOTE

The route editor works with these project types:

- `camel-archetype-activemq`



Creates a new Apache Camel project that configures and interacts with Apache ActiveMQ.

- **camel-archetype-blueprint**

Creates a new Apache Camel project with support for OSGi blueprint.

- **camel-archetype-component**

Creates a new Apache Camel project for building a new component.

- **camel-archetype-component-scala**

Creates a new Apache Camel project for building a new component using Scala.

- **camel-archetype-dataformat**

Creates a new Apache Camel project for building a new data format.

- **camel-archetype-java**

Creates a new Apache Camel project using Java DSL.

You cannot edit Java DSL in the route editor.

- **camel-archetype-scala**

Creates a new Apache Camel project using Scala DSL.

- **camel-archetype-spring**

Creates a new Apache Camel project with added support for Spring DSL.

- **camel-archetype-spring-dm**

Creates a new Apache Camel web project that deploys the routes as a WAR.

- **camel-archetype-web**

Creates a new Apache Camel project with added support for Spring DSL.

- **camel-archetype-webconsole**

Creates a new Apache Camel project that deploys the Apache Camel Web Console, REST API, and your routes as a WAR.

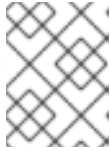
- **camel-cxf-code-first-archetype**

Creates a new Apache Camel project using Apache CXF code (Java) first.

- **camel-cxf-contract-first-archetype**

Creates a new Apache Camel project using Apache CXF contract (WSDL) first.

- **camel-drools-archetype**



Creates a new Apache Camel project that uses the Drools rule engine. Because this archetype is not yet OSGi ready with Spring and Camel, it is provided as a technical preview only. It is not ready for use in production environments.

6. Enter a group ID for the project in the **Group Id** field, or accept the default.

The developer tools use the group ID to form the first part of the dot-separated package name. For example, if you enter **Demo** for the group ID, it appears in the **Package** field as **Demo..**

7. Enter an artifact ID for the project in the **Artifact Id** field, or accept the default.

The developer tools use the artifact ID as the name of the project and to form the second part of the dot-separated package name. For example, if you enter **BigRoute** for the artifact ID, it appears in the **Package** field as **Demo.BigRoute**.

8. Enter a version for the project in the **Version** field, or accept the default.
9. If you want to change the package name generated for the artifacts, enter the new package name in the **Package** field.

10. Click **Finish** to create the Maven project, which contains starting point artifacts.

RESOLVING MAVEN DEPENDENCY ERRORS

You may encounter Maven dependency errors after you create a new Fuse project.

Though it can happen at other times, it more typically occurs when you create a project based on any of the supplied archetypes for the first time, but then cancel the project before the process finishes. Interrupting the process in this way often prevents all of the project's dependencies from downloading from the Maven repositories, which can take some time.

You can often easily resolve these dependency errors by updating Maven dependencies this way:

1. In **Package Explorer**, select the root project just created.
2. Right-click it to open the context menu.
3. Select **Maven** → **Update Project...**
4. In the **Update Maven Project** wizard, check **Force Update of Snapshots/Releases**.
5. Click **OK**.

In the bottom, right corner of the workbench, you may see the progress status bar churning as missing dependencies are downloaded from the Maven repositories.

RELATED TOPICS

[New Fuse Project](#)

CHAPTER 3. CREATING A NEW CAMEL XML FILE

7/22/11

EMJ

Changed note about location to clarify that it is for using Spring

7/20/12

EMJ

Fixed FIDEDOC-3 by adding location used by Blueprint

OVERVIEW

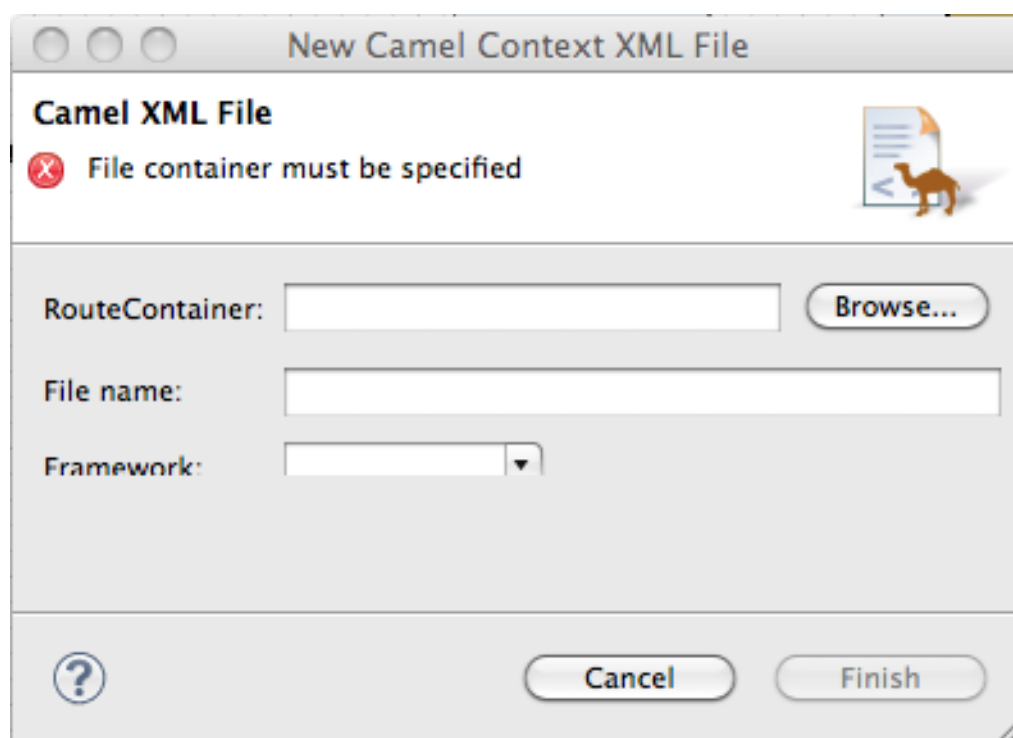
Apache Camel stores routes in an XML file that contains a `camelContext` element. The developer tools include a wizard that simplifies adding an Apache Camel context file to your project. It creates a new XML file that has all of the required namespaces preconfigured and a template `camelContext` element.

PROCEDURE

To add an Apache Camel context file to your project:

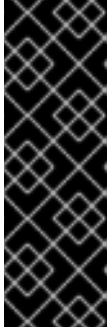
1. Select **File** → **New** → **Camel XML File** from the main menu to open the **Camel XML File** wizard, as shown in [Figure 3.1](#), “Camel XML File wizard”.

Figure 3.1. Camel XML File wizard



2. In **RouteContainer**, enter the location for the new file, or accept the default.

You can click  to search for an appropriate location.



IMPORTANT

The Spring framework and the OSGi Blueprint framework require that all Apache Camel files be placed in specific locations under the project's **META-INF** folder:

- Spring—*projectName* /src/main/resources/META-INF/spring/
- OSGi Blueprint—*projectName* /src/main/resources/META-INF/blueprint/

3. In **File Name**, enter a name for the new context file, or accept the default.

The file's name cannot contain spaces or special characters, and it must be unique within the JVM.

4. In **Framework**, accept the default, or select which framework the routes will use:

- **Spring**—for routes that will be deployed in Spring containers, non-OSGi containers, or as standalone applications
- **OSGi Blueprint**—for routes that will be deployed in OSGi containers
- **Routes**—for routes that you can load and add into existing **camelContexts**

5. Click **Finish**.

The new context file is added to the project and opened in the route editor.

RELATED TOPICS

[New Camel XML File](#)

CHAPTER 4. EDITING A ROUTING CONTEXT IN THE ROUTE EDITOR

8/11/11

EMJ

Added editor configuration topic

Developing an Apache Camel application typically consists of the following tasks:

1. [Adding](#) one or more routes to the routing context.
2. [Adding](#) a starting point pattern to a route.
3. [Adding](#) one or more endpoint patterns to a route.
4. [Adding](#) one or more processor patterns that represent how messages will be transformed and routed between the starting point and endpoint(s).
5. [Connecting](#) the patterns (referred to as nodes once they are placed on the canvas).
6. [Configuring](#) the details for each of the endpoints and processors that make up the route.
7. [Adding](#) any configuration beans to the context.

4.1. ADDING ROUTES TO THE ROUTING CONTEXT

Overview

The `camelContext` element within an xml context file creates a routing context. The `camelContext` element can contain one or more routes, but the route editor's canvas can display only one of the routes at a time. Therefore the canvas is each route's delineator, and each route displayed on the canvas maps to a `route` element in the generated `camelContext` element.

When you create a new Fuse Project, the developer tools create a `camelContext.xml` file that contains an empty `route` element within a `camelContext` element. You can view and edit the contents of the `camelContext.xml` file in the route editor's **Source** view. In **Design** view, the route editor presents an empty canvas, which represents the empty `route` element. You can drag patterns from the **Palette** and drop them onto the canvas to create a route. The design time tooling updates the empty `route` element with xml generated from the patterns you drop onto the canvas.

The developer tools provide three methods for adding a new route:

- From the menu bar, by opening the **Routes** menu, and then selecting **Add Route**
- In Design view, by right-clicking the canvas or a node to access the context menu, and then selecting **Add** → **Route**
- In Source view, by adding a `<route/>` element to the existing list within the `camelContext` element

Procedure

To add another route to the camelContext:

1. Select one of the methods for adding a route.

In **Design** view, a route icon appears in **Outline** view, and the **Properties** editor displays the list of the new route's properties for you to edit.

2. In the **Properties** editor, enter an ID (for example, Route2) for the new route in the route's **ID** field.

In **Outline** view, the new ID displays next to the new route icon.

3. In the **Properties** editor, enter a description of the route in the **Description** field.
4. On the menu bar, select **File** → **Save** to save the changes you made to the routing context file.



NOTE

To switch between multiple routes, select the route you want to display on the canvas by clicking it in **Outline** view or by selecting it from the **Routes** menu on the menu bar.

4.2. ADDING PATTERNS TO A ROUTE

Routes consist of a sequence of connected patterns, referred to as nodes once they are placed on the canvas. To add a pattern to a route you can either:

- [Drag the pattern from the palette](#)
- [Use the context menu](#)

Both methods provide access to all of the available patterns. Dragging patterns from the palette and dropping them onto the canvas is an easy and convenient way to add multiple patterns to the canvas quickly. Using a node's context menu, you can add and automatically connect a pattern to the node.

4.2.1. Drag and drop a pattern

Overview

The **Palette** lists all of the available patterns, grouped by function. For example, the Loop pattern is part of the **Control Flow** group.

You can select any pattern from the **Palette** and drag it to the canvas, which represents a single route.

Procedure

To drag a pattern onto a route:

1. In the **Palette**, locate the desired pattern.
2. Select the pattern, drag it to the canvas, and then release the mouse button.



NOTE

You need not place the pattern in its intended location. You can easily move any node on the canvas by dragging it to a new location.

The new pattern appears on the canvas and becomes the selected node. The **Properties** editor displays a list of the node's properties for you to edit and configure.

Related topics

Section 4.5, "Rearranging patterns on the canvas"
Section 4.4, "Configuring a pattern"
Section 4.6, "Removing patterns from a route"

4.2.2. Using the context menu

Overview

Nodes on the canvas have a context menu that includes an **Add** option for adding new patterns to a route.

Procedure

To add and connect a pattern to a node in a route:

1. In the canvas, select the node to which you want to connect the new pattern.
2. Right-click it to open the context menu.
3. Select **Add** to display the list of patterns that can be connected to the selected node. The patterns are grouped according to function.
4. Select the pattern to add to the route.

The new pattern is added to the route and automatically linked to the selected node. The new pattern becomes the currently selected node, and the **Properties** editor displays a list of the node's properties for you to edit and configure.

Related topics

Section 4.5, "Rearranging patterns on the canvas"
Section 4.4, "Configuring a pattern"
Section 4.6, "Removing patterns from a route"

4.3. CONNECTING PATTERNS TO MAKE A ROUTE

Overview


Until the patterns on the canvas (referred to as nodes) are connected, they do not constitute a route. The editor does not save a **route** element in the context file until all nodes are linked together to form a complete route. A complete route typically consists of a starting endpoint, a string of processing nodes, and one or more destination endpoints.

Connecting two nodes on the canvas is as simple as dragging a line from one to the other. Each node has a connector arrow. Selecting a node and dragging its connector arrow to a target node establishes a connection between the two nodes.

Procedure

To connect two nodes:

1. On the canvas, select the source node to display its connector arrow.

2. Click-drag the source node's connector arrow () to the target node.

The direction of the connection represents the direction messages flow between the nodes in the route.

3. While hovering over the target node, release the mouse button to drop the connector on it.

The route editor updates the **<route>** element with the xml generated from the connection. You can view the xml in **Source** view.

4. When you are done, save your work by selecting **File** → **Save** from the menu bar.

4.4. CONFIGURING A PATTERN

Overview

Most patterns require some explicit configuration. For example, an endpoint requires an explicitly entered **URI**.

The developer tools' **Properties** editor provides a form that lists all of the configuration details a particular pattern supports. The **Properties** editor also provides the following convenience features:

- validating that all required properties have values
- drop-down lists for properties that have a fixed set of values
- drop-down lists that are populated with the available bean references from the Apache Camel Spring configuration

Procedure

To configure a pattern:

1. On the canvas, select the node you want to configure.

The **Details** tab in the **Properties** editor lists all of the selected node's properties for you to edit. The **Documentation** tab describes the pattern and each of its properties.

2. Edit the fields in the **Properties** editor to configure the node.

3. When done, save your work by selecting **File** → **Save** from the menu bar.

Related topics

4.5. REARRANGING PATTERNS ON THE CANVAS

As your route grows and changes, you'll likely need to change how it is laid out on the canvas. Changing the layout does not change the underlying DSL, but it does make it easier to visualize the route.

The developer tools provide two ways to rearrange the patterns on the canvas:

- [dragging and dropping](#)
- [the context menu's](#)

4.5.1. Rearranging patterns by dragging them

Overview

You can select one or more nodes on the canvas and drag them to a new location. Each node's connections are retained and move along with it. However, the layout of the route will continue to change as it grows in scope and complexity.

Procedure

To drag a node to a new location on the canvas:

1. Determine which node or nodes you want to move.
2. Select one node, or select a group of nodes by dragging a box around them.
3. Drag the node or nodes to their new location, then release the mouse button.



NOTE

You can also rearrange the connectors between nodes to accommodate increasing numbers of patterns on the canvas. Clicking a connector joining two nodes on the canvas reveals a line segmented by a bendpoint. You can drag the bendpoint to another location on the canvas while leaving the two nodes in place. Dragging a bendpoint creates two new bendpoints on either side of it, further segmenting the connector. This feature makes it easy to organize the nodes in a complex route for visual clarity.

4.5.2. Automatically aligning patterns

Overview

The **Layout Diagram** command on the canvas' context menu automatically aligns all nodes on the canvas relative to the route's start point. By default, the route editor arranges the nodes in a route from left to right across the canvas, according to the order in which they are connected to each other. It moves the start point to the left edge of the canvas, then aligns the processing nodes and endpoints accordingly.

**NOTE**

You can change the default layout direction (**Right**). To do so, on the menu bar select **JBoss Developer Studio** → **Preferences** → **Fuse Plugins for Eclipse** → **Editor**, and then select **Down** from the **Choose the layout direction for the diagram editor**'s drop-down list

You can continue adding more patterns to your route after you execute the **Layout Diagram** command.

Procedure

To direct the developer tools to automatically align all nodes on the canvas:

1. Right-click the canvas to open the context menu.
2. Select **Layout Diagram**.

4.6. REMOVING PATTERNS FROM A ROUTE**Overview**

As you develop and update a route, you may need to remove one or more of the route's nodes. The **Remove** option on the nodes' context menu makes this easy to do. When you delete a node from the canvas, all of its connections with all other nodes on the canvas are also deleted.

**NOTE**

You can also remove a node by right-clicking it in **Outline** view and selecting **Remove**, or by selecting **Edit** → **Delete** from the menu bar.

Procedure

To remove a node from a route:

1. Select the node you want to delete.
2. Right-click it to open the context menu.
3. Select **Remove**.

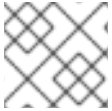
The node and all of its connections are deleted from the canvas.

Related topics

[Section 4.2, “Adding patterns to a route”](#)

4.7. DISCONNECTING TWO PATTERNS**Overview**

As you develop and update a route, you may need to disconnect some nodes. The **Remove** option on the connector context menu makes this easy to do.



NOTE

You can also delete a connector by selecting **Edit** → **Delete** from the menu bar.

Procedure

To disconnect two nodes:

1. Select the connector you want to delete.
2. Right-click it to open the context menu.
3. Select **Remove**.

The connector between the nodes is removed.

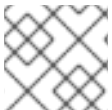
Related topics

[Section 4.3, “Connecting patterns to make a route”](#)

4.8. DELETING A ROUTE

Overview

In some cases you may need to delete an entire route from your routing context. The **Remove Route** option on the canvas' context menu makes this easy to do. When you delete a route, all of the nodes the route contains are also deleted.



NOTE

You can also remove a route from the **Routes** menu on the menu bar.

Procedure

To delete a route:

1. If the routing context contains more than one route, first select the route you want to delete in **Outline** view.
2. Right-click on the canvas to open the context menu.
3. Select **Remove Route**.

The route is removed from the canvas and from the camelContext.

4.9. ADDING BEANS AND CONFIGURATION

7/22/11

EMJ

added link to source editor topic and changed Spring Beans to beans

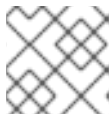
Overview

Many routing patterns rely on references to Java objects (beans) for configuration or for implementation details. You add the beans into the routing context file using the route editor's **Source** tab.

Procedure

To add beans to your routing context file:

1. Open your routing context file in the route editor.
2. Click the **Source** tab at the bottom of the route editor's canvas to switch to Source view, so you can edit the XML that defines the route.
3. Enter the **bean** elements needed by your route before the **camelContext** element.



NOTE

Use the **id** attribute to identify the bean, not the **name** attribute.



IMPORTANT

Do not edit the contents of the **camelContext** element. Red Hat JBoss Fuse Plugins for Eclipse overwrites the **camelContext** element when changes are made to the route diagram in **Design** view.

4. Save your changes by selecting **File** → **Save** on the menu bar.
5. Click the **Design** tab at the bottom of the route editor's canvas to return to Design view and the route diagram.

Related topics

[Chapter 6, The Source View](#)

[Configuring a Component](#) in the Endpoint Reference guide

4.10. CONFIGURING THE ROUTE EDITOR

Overview

Using the JBoss Developer Studio's preference panel, you can specify many aspects of the route editor's behavior:

- The default language to use for expressions in EIPs
- The method for labeling nodes on the canvas
- The direction in which patterns flow on the canvas when creating routes
- Whether the canvas displays a grid overlay

- Colors used for various diagrammatic components

Procedure

To configure the route editor:

1. On the menu bar, select **JBoss Developer Studio** → **Preferences** to open the **Preferences** dialog.
2. Select **Fuse Plugins for Eclipse** → **Editor** to show the **Editor** Preferences.
3. Select the expression language you prefer from the drop-down list.
4. Click the checkbox next to **If enabled the ID values will be use for labels if existing** to enable or disable using node IDs as labels.
5. Select, from the drop-down list, the direction in which you want the route editor to align the patterns in a route.
6. Click the checkbox next to **Show diagram grid** to enable or disable displaying the grid overlay on the canvas.
7. Click **Apply** to apply the changes to the **Editor** Preferences.
8. Select **Fuse Plugins for Eclipse** → **Colors** to show the **Colors** Preferences.
9. For each component whose color you want to change, click its color icon to open the color palette, and then select a color from the palette.
10. When done, click **Apply** and then **OK**, to close the **Preferences** dialog.

You can restore the route editor's original color scheme at any time by returning to **Fuse Plugins for Eclipse** → **Colors** and clicking **Restore Defaults**.

Related topics

[Editor](#)

CHAPTER 5. CREATING A NEW APACHE CAMEL JUNIT TEST CASE

11/16/2012

JCM

Added heads up for users to edit their JUnit test case to ensure it runs a valid test

OVERVIEW

A common way of testing routes is to use JUnit. The design time tooling includes a wizard that simplifies creating a JUnit test case for your routes. The wizard uses the endpoints you specify to generate the starting point code and configuration for the test.



NOTE

After you create the boilerplate JUnit test case, you need to modify it to add expectations and assertions specific to the route that you've created or modified, so the test is valid for the route.

PROCEDURE

To create a new JUnit test case for your route:

1. In **Package Explorer**, select the camel-context.xml file in your routing project.
2. Right-click it to open the context menu, and then select **New** → **Camel Test Case** to open the **New Camel JUnit Test Case** wizard, as shown in [Figure 5.1, “New Camel JUnit Test Case wizard”](#).

Figure 5.1. New Camel JUnit Test Case wizard

Camel JUnit Test Case

✖ Source folder name is empty.

Source folder:

Package: (default)

Camel XML file under test:

Name:

Which method stubs would you like to create?

setUpBeforeClass() tearDownAfterClass()
 setUp() tearDown()
 constructor

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

Alternatively, you can open the wizard by selecting **File** → **New** → **Camel Test Case** from the menu bar.

3. In **Source folder**, accept the default location of the source code for the test case, or enter another location.

You can click to search for a location.

4. In **Package**, accept the default package name for the generated test code, or enter another package name.

You can click to search for a package.

5. In **Camel XML file under test**, accept the default pathname of the routing context file that contains the route you want to test, or enter another pathname.

You can click to search for a context file.

6. In **Name**, accept the default name for the generated test class, or enter another name.

7. Select the method stubs you want to include in the generated code.

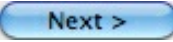
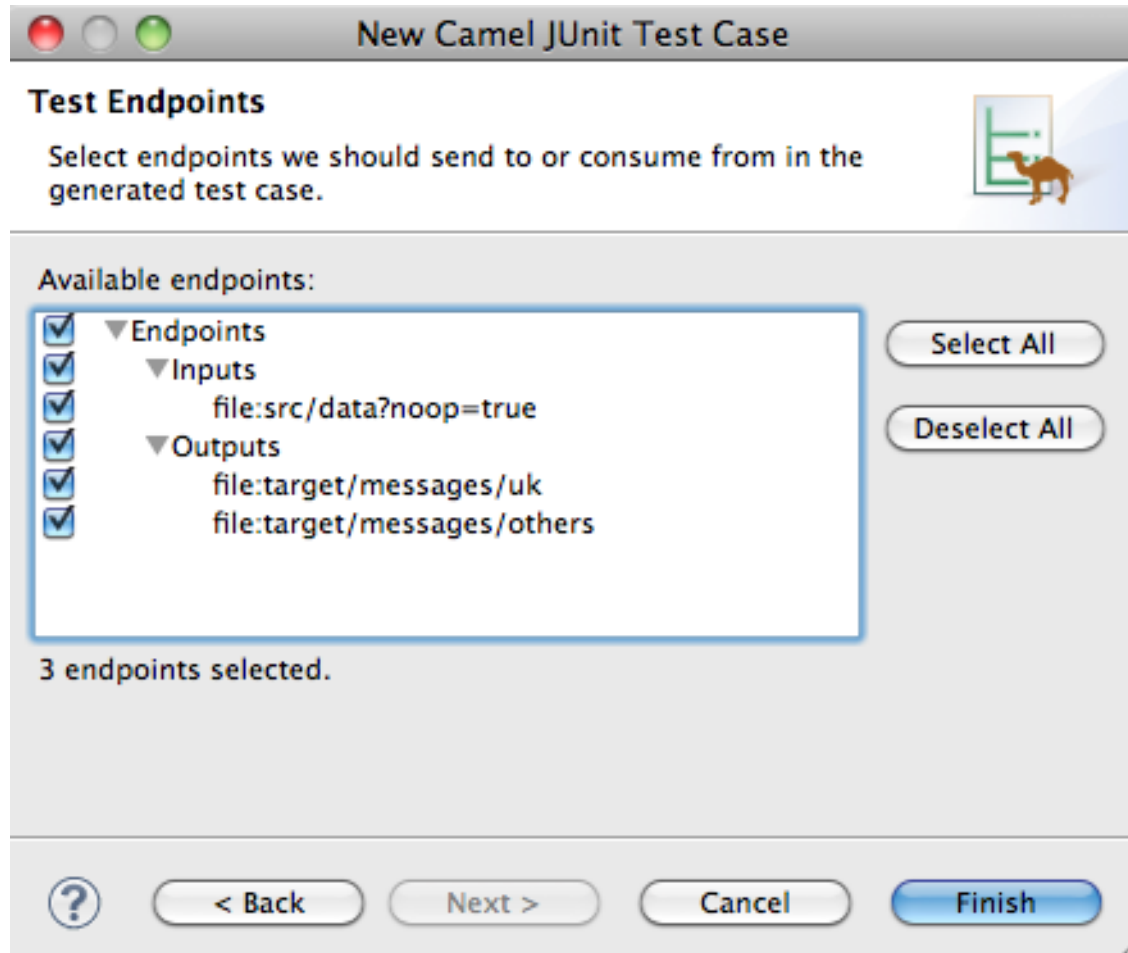

8. If you want to include the default generated comments in the generated code, check the **Generate comments** box.
9. Click  to open the **Test Endpoints** page. For example, [Figure 5.2, “New Camel JUnit Test Case page”](#) shows a route's input and output file endpoints selected.

Figure 5.2. New Camel JUnit Test Case page



10. Under **Available endpoints**, select the endpoints you want to test. Click the checkbox next to any selected endpoint to deselect it.
11. Click  .



NOTE

If prompted, add JUnit to the build path.

The artifacts for the test are added to your project and appear in **Package Explorer** under **src/test/java**. The class implementing the test case opens in the Java editor.

RELATED TOPICS

[New Camel JUnit Test Case](#)

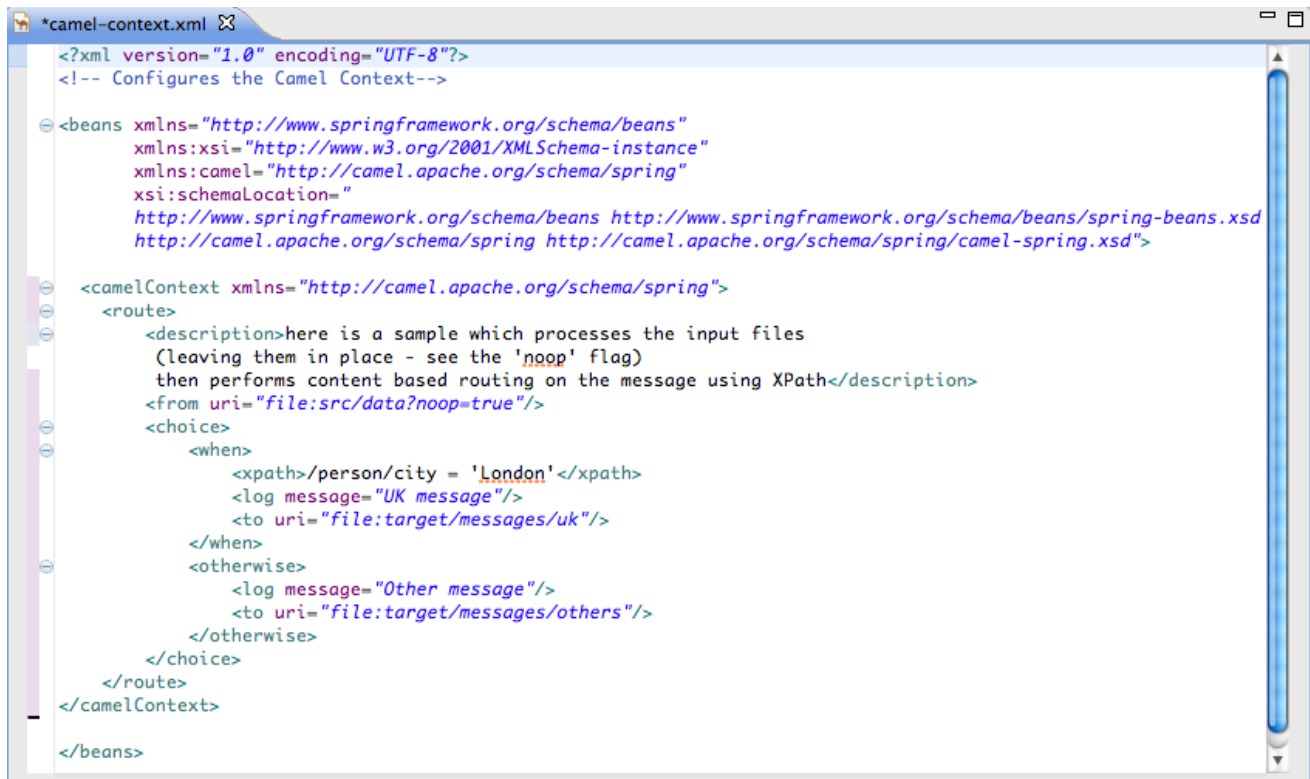
[Test Endpoints](#)

[chapter "Testing a Route with JUnit" in "Fuse IDEFuse Plugins for Eclipse Tutorials"](#)

CHAPTER 6. THE SOURCE VIEW

The developer tools' **Source** tab opens the route editor in Source view (Figure 6.1) to display the contents of an Apache Camel route file as XML.

Figure 6.1. Source view



```

*camel-context.xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Configures the Camel Context-->

<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:camel="http://camel.apache.org/schema/spring"
xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <description>here is a sample which processes the input files
      (leaving them in place - see the 'noop' flag)
      then performs content based routing on the message using XPath</description>
      <from uri="file:src/data?noop=true"/>
      <choice>
        <when>
          <xpath>/person/city = 'London'</xpath>
          <log message="UK message"/>
          <to uri="file:target/messages/uk"/>
        </when>
        <otherwise>
          <log message="Other message"/>
          <to uri="file:target/messages/others"/>
        </otherwise>
      </choice>
    </route>
  </camelContext>

</beans>

```

This view is useful for editing and adding any configuration, comments, or beans to the routing context file. The **Properties** editor can inspect the beans and use the information to populate the fields that require bean references.

While the route editor allows you to change the contents of the **camelContext** element in the routing context file, not all changes are preserved. Comments inserted into the **camelContext** element are lost upon switching to **Design** view.

CHAPTER 7. RUNNING ROUTES INSIDE RED HAT JBOSS FUSE PLUGINS FOR ECLIPSE

There are two basic ways to run your routes using the developer tools:

- [As a local camel context \(with or without tests\)](#)
- [Using Maven](#)

7.1. RUNNING ROUTES USING MAVEN

Overview

If the project containing your route is a Maven project, you can use the m2e plug-in to run your route. Using this option, you can execute any Maven goals, before the route runs.

Procedure

To run a route using Maven:

1. In **Package Explorer**, select the root of the project .
2. Right-click it to open the context menu, and then select **Run As** → **Maven build**.
 - The first time you run the project using Maven the **Edit Configuration** editor opens, so you can create a Maven runtime profile.

To create the runtime profile:

1. Make sure the route directory of your Apache Camel project appears in the **Base directory:** field.

For example on Linux the root of your project will be similar to `~/workspace/simple-router`.

2. In the **Goals:** field, enter `camel:run`.
3. Click **Apply** and then **Run**.
 - Subsequent Maven runs use this profile, unless you modify it between runs.

Results

The **Console** view displays the output from the Maven run.

Related topics

[Section 7.3.2, “Editing a Maven runtime profile”](#)

7.2. RUNNING ROUTES AS A LOCAL CAMEL CONTEXT

Overview

The simplest way to run an Apache Camel route is as a **Local Camel Context**. This method enables you to launch the route directly from **Package Explorer**'s context menu. When you run a route from the context menu, the developer tools automatically create a runtime profile for you. You can also create a custom runtime profile for running your route.

Your route runs as if it were invoked directly from the command line and uses Apache Camel's embedded Spring container. You can configure a number of the runtime parameters by editing the runtime profile.

Procedure

To run a route as a local Camel context:

1. In **Package Explorer**, select a routing context file.
2. Right-click it to open the context menu, and then select **Run As** → **Local Camel Context**.



NOTE

Selecting **Local Camel Context (without tests)** directs the Red Hat JBoss Fuse Plugins for Eclipse to run the project without performing validation tests, which may be faster.

Result

The **Console** view displays the output generated from running the route.

Related topics

[Section 7.3.1, "Editing a Local Camel Context runtime profile"](#)

7.3. WORKING WITH RUNTIME PROFILES

The Red Hat JBoss Fuse Plugins for Eclipse store information about the runtime environments for each project in *runtime profiles*. The runtime profiles keep track of such information as which Maven goals to call, the Java runtime environment to use, any system variables that need to be set, and so on. A project can have more than one runtime profile.

7.3.1. Editing a Local Camel Context runtime profile

Overview

A **Local Camel Context** runtime profile configures how Apache Camel is invoked to execute a route. A **Local Camel Context** runtime profile stores the name of the context file in which your routes are defined, the name of the **main** to invoke, the command line options passed into the JVM, the JRE to use, the classpath to use, any environment variables that need to be set, and a few other pieces of information.

The runtime configuration editor for a **Local Camel Context** runtime profile contains the following tabs:

- **Camel Context File**—specifies the name of the new configuration and the full path of the routing context file that contains your routes.
- **Main**—specifies the fully qualified name of the project's base directory, a few options for locating the base directory, any goals required to execute before running the route, and the version of the Maven runtime to use.
- **JRE**—specifies the JRE and command line arguments to use when starting the JVM.
- **Refresh**—specifies how Maven refreshes the project's resource files after a run terminates.
- **Environment**—specifies any environment variables that need to be set.
- **Common**—specifies how the profile is stored and the output displayed.

The first time an Apache Camel route is run as a **Local Camel Context**, Red Hat JBoss Fuse Plugins for Eclipse creates for the routing context file a default runtime profile, which should not require editing.

Accessing the Local Camel Context's runtime configuration editor


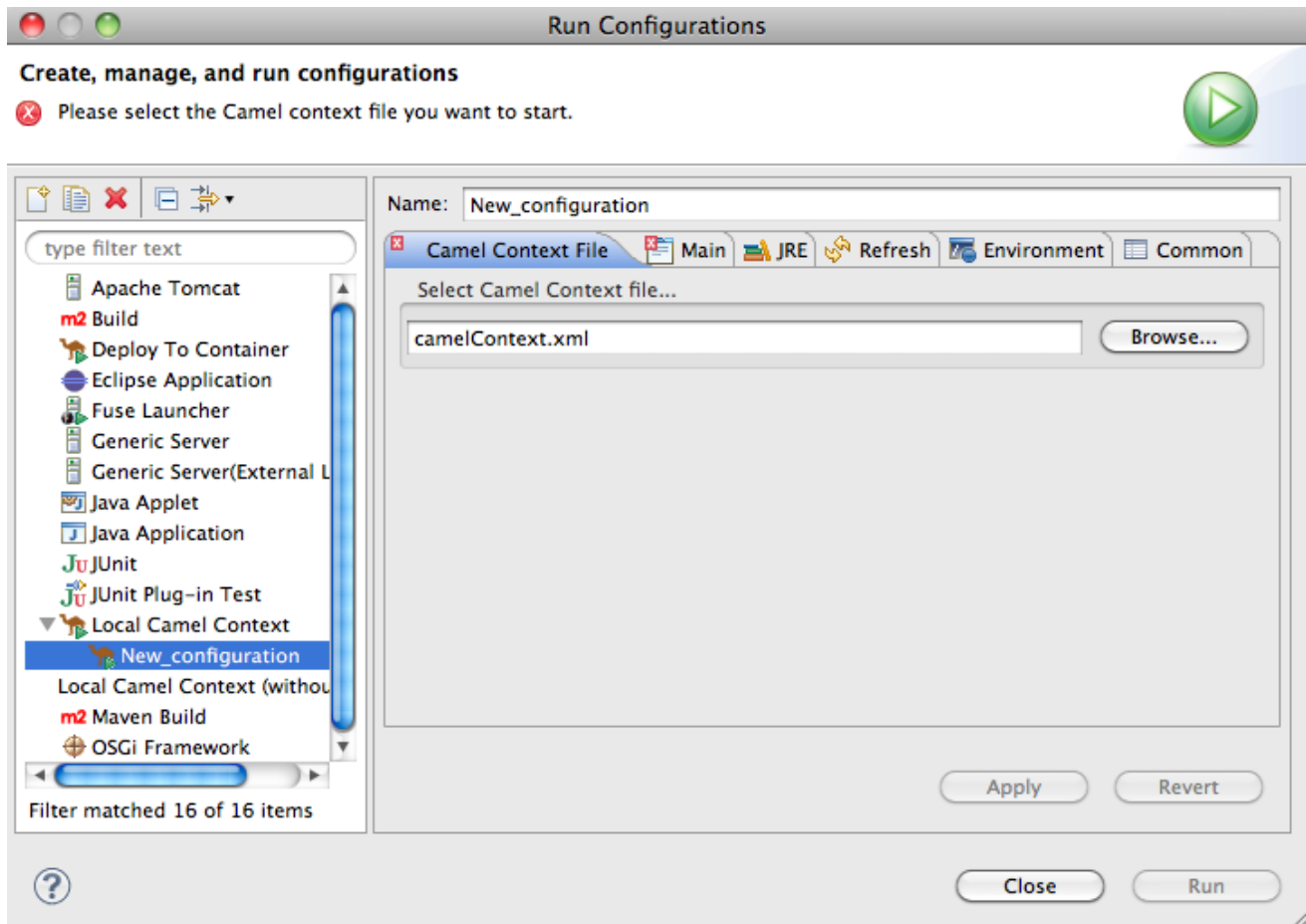
1. In **Package Explorer**, select the camelContext file for which you want to edit or create a custom runtime profile.
2. Right-click it to open the context menu, and then select **Run As... → Run Configurations** to open the **Run Configurations dialog**.
3. In the context selection pane, select **Local Camel Context**, and then click  at the top, left of the context selection pane.
4. In the **Name** field, enter a new name for your runtime profile.

Figure 7.1. Runtime configuration editor for Local Camel Context



Setting the camel context file

The **Camel Context File** tab has one field, **Select Camel Context file...** Enter the full path to the routing context file that contains your route definitions.

The **Browse...** button accesses the **Open Resource** dialog, which facilitates locating the target routing context file. This dialog is preconfigured to search for files that contain Apache Camel routes.

Changing the command line options

By default the only command line option passed to the JVM is:

```
-fa context-file
```

If you are using a custom main class you may need to pass in different options. To do so, on the **Main** tab, click the **Add** button to enter a parameter's name and value. You can click the **Add Parameter** dialog's **Variables...** button to display a list of variables that you can select.

To add or modify JVM-specific arguments, edit the **VM arguments** field on the **JRE** tab.

Changing where output is sent

By default, the output generated from running the route is sent to the **Console** view. But you can redirect it to a file instead.

To redirect output to a file:

1. Select the **Common** tab.
2. Click the checkbox next to the **File:** field, and then enter the path to the file where you want to send the output.

The **Workspace...**, **File System...**, and **Variables...** buttons facilitate building the path to the output file.

Related topics

[Section 7.2, “Running routes as a local Camel context”](#)

7.3.2. Editing a Maven runtime profile

Overview

A Maven runtime profile configures how Maven invokes Apache Camel. A Maven runtime profile stores the Maven goals to execute, any Maven profiles to use, the version of Maven to use, the JRE to use, the classpath to use, any environment variables that need to be set, and a few other pieces of information.

The runtime configuration editor for a Fuse runtime profile contains the following tabs:

- **Main**—specifies the name of the new configuration, the fully qualified name of the project's base directory, a few options for locating the base directory, any goals required to execute before running the route, and the version of the Maven runtime to use.
- **JRE**—specifies the JRE and command line arguments to use when starting the JVM.
- **Refresh**—specifies how Maven refreshes the project's resource files after a run terminates.
- **Environment**—specifies any environment variables that need to be set.
- **Common**—specifies how the profile is stored and the output displayed.

The first time an Apache Camel route is run using Maven, you must create a default runtime profile for it.

Accessing the Maven runtime configuration editor


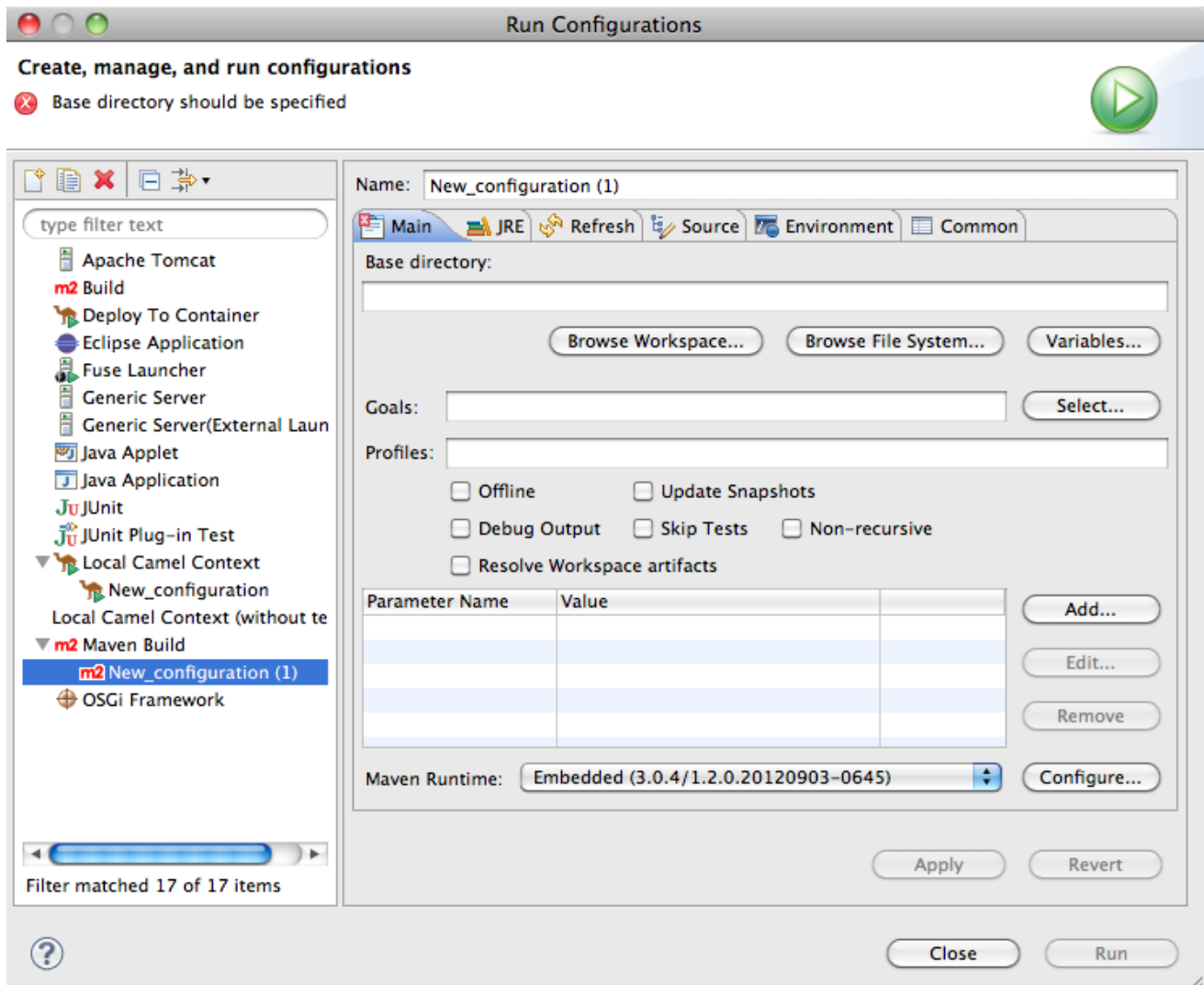
1. In **Package Explorer**, select the root of the project for which you want to edit or create a custom runtime profile.
2. Right-click it to open the context menu, and then select **Run As... → Run Configurations** to open the **Run Configurations dialog**.
3. In the context selection pane, select **m2 Maven Build**, and then click  at the top, left of the context selection pane.

Figure 7.2. Runtime configuration editor for Maven



Changing the Maven goal

The most commonly used goal when running a route is **camel:run**. It loads the routes into a Spring container running in its own JVM.

The Apache Camel plug-in also supports a **camel:embedded** goal that loads the Spring container into the same JVM used by Maven. The advantage of this is that the routes should bootstrap faster.

If your POM contains other goals, you can change the Maven goal used by clicking the **Configure...** button next to the **Maven Runtime** field on the **Main** tab. On the **Installations** dialog, you edit the **Global settings for <selected_runtime> installation** field.

Changing the version of Maven

By default, the Red Hat JBoss Fuse developer tools use m2e, which is embedded in Eclipse. If you want to use a different version of Maven or have a newer version installed on your development machine, you can select it by editing the **Maven Runtime** field on the **Main** tab.

Changing where the output is sent

By default, the output from the route execution is sent to the **Console** view. But you can redirect it to a file instead.

To redirect output to a file:

1. Select the **Common** tab.
2. Click the checkbox next to the **File:** field, and then enter the path to the file where you want to send the output.


The **Workspace...**, **File System...**, and **Variables...** buttons facilitate building the path to the output file.

Related topics

[Section 7.1, “Running routes using Maven”](#)

CHAPTER 8. DEPLOYING PROJECTS TO A CONTAINER

	8/11/11	EMJ
made a chapter with sections to include container config	7/13/12	EMJ
rewrote for clarity and accuracy	7/13/12	EMJ
merged configuration of hot deploy folder and hot deploy procedure into a single section		

The developer tools' **Deploy to...** tool() makes deploying applications into containers a one step process.

The developer tools support direct deployment into the following containers:

- Red Hat JBoss Fuse
- Apache ServiceMix
- Apache Karaf

There are three ways to deploy Fuse projects into a container:

- into a container's deploy folder

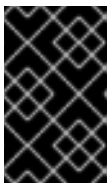
You copy the package generated by the developer tools into the container's deploy folder. This method has two drawbacks:

- The developer tools do not verify that the project builds a package that is compatible with the target container.
 - The developer tools do not provide feedback as to whether the deployment succeeds or fails.
- to a JMX connected container

The developer tools build the project, and, regardless of the packaging specified in the project's POM, installs the generated artifacts as a bundle. You can use the **Fuse JMX Navigator** to determine if the project deploys successfully.

- to a fabric profile

The developer tools build the project, packages the generated artifacts into a bundle, adds the bundle to the profile, and copies the bundle into the fabric's internal repository.



IMPORTANT

If the profile is not assigned to a running container the project will not be deployed into the fabric. You must assign the profile to a running container before you can see the project running.



NOTE

You can deploy a project by dropping it onto a connected container in **Fuse JMX Navigator**, or onto a fabric profile in **Fabric Navigator**.

8.1. USING A CONTAINER'S DEPLOY FOLDER

When the developer tools deploy a project into a container's deploy folder, they build the project as specified in the project's POM and copies the resulting package to a predetermined location. The developer tools assume two things:

- the specified location is a container's deploy folder
- the package generated by the project is compatible with the container

Before you can deploy a project to a container's deploy folder you must add the container's deployment information to the developer tools. Once the developer tools know the container's information, the container will appear as an option in the **Deploy To...** menu.

8.1.1. Configuring a Container's Deploy Folder

	7/22/11	EMJ
Added info about opening the container preferences		
	7/13/12	EMJ
changed deployment folder to deploy folder so it is consistent with the UI		
	7/13/12	EMJ
edited for clarity		

Overview

Before you can deploy applications into a container's deploy folder, you must provide the developer tools with the location of the container's deploy folder. The **Deploy Folders** preference panel enables you to configure one or more containers. You provide a name for the container and the full path to the container's deployment folder.

Accessing the Deploy Folder preference panel

You can access the deployment folders preference panel two ways:

- selecting **JBoss Developer Studio** → **Preferences** → **Fuse Plugins for Eclipse** from the main menu

This method opens the general Eclipse preference window. You need to select **Deploy Folders** from the list of preferences in the panel on the left side of the window.

- selecting **Deploy to...** → **Deploy Folder Configurations...** from the project's context menu

This method opens the preferences window and ensures that the **Deploy Folders** panel is selected.

Add a container's deploy folder

To add a container's deploy folder:

1. Open the **Deploy Folder** preference panel.
2. In the **Name** field, enter a name for the target container.

The name can be any unique string.

3. In the **Deploy Folder** field, enter the full path to the container's deployment folder.


The  button opens a file explorer.



IMPORTANT

The developer tools do not validate whether the selected folder is a valid deployment folder for a container.

4. In the **Description** field, enter a description for the container.
5. Click **Add** to add the new container to the table of configured containers.
6. Click **OK** to close the preferences window.

When add a container to the list, it appears in the context menus'  **Deploy to...** submenu.

Edit a container's deploy folder


To edit a container's deploy folder:

1. Open the **Deploy Folder** preference panel.
2. Select the container whose configuration you want to edit.
3. Edit the information that needs updating.




IMPORTANT

Changing the value of the **Name** field creates a new container configuration.

4. Click .
5. Click **OK** to close the preferences window.

Delete a container's deploy folder

To delete a container's deploy folder:

1. Open the **Deploy Folder** preference panel.
2. Select the container to be deleted.
3. Click .
4. Click **OK** to close the preferences window.

Related topics

Chapter 8, *Deploying Projects to a Container*

Deploy Folders

8.1.2. Deploying a Project to a Container's Deploy Folder

8/11/11

EMJ

Split into a section

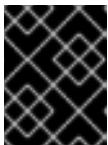
Overview

When the developer tools deploy a project to a container's deploy folder it builds the project and copies the results to the deploy folder configured for the container.

Procedure

To deploy a project to a container's deploy folder:

1. Select the routing context file that you want to deploy.



IMPORTANT


The project must be configured to package the results in a format that the target container can deploy.

2. Select  **Deploy to...** → *containerName* from the project's context menu.

Red Hat JBoss Fuse Plugins for Eclipse builds the project and copies the results to the selected container's deploy folder.



NOTE

If you have not already configured a container, you can do so by selecting  **Deploy to...** → **Deploy Folder Configurations...** from the context menu.

See [Section 8.1.1, "Configuring a Container's Deploy Folder"](#).

3. Check the **Console** view to see if your project was successfully built and copied to the target container.
4. Check the container's log files to see if the project was successfully deployed.

Related topics

[Section 8.1.1, "Configuring a Container's Deploy Folder"](#)

8.2. DEPLOYING A PROJECT TO A JMX CONNECTED CONTAINER

Overview

Deploying into a container connected to the developer tools' **Fuse JMX Navigator** provides a number of benefits over using a container's deploy folder. The chief benefits are that the project is automatically packaged into a bundle, and you can monitor the results of the deployment in the developer tools.

All of the supported containers that are connected to the **Fuse JMX Navigator** are listed in the **Deploy to...** menu. For information about how to connect a container to the **Fuse JMX Navigator**, see [Chapter 11, *The Fuse JMX Navigator*](#).

Procedure

To deploy a project to a JMX connected container:

1. Open the **Fuse Integration** perspective.
2. In **Fuse JMX Navigator**, expand the JMX server hosting the desired container.

Containers that are running on the local machine are listed in the special **Local Processes** node.
3. Double-click the entry for the desired container connect to it.
4. In **Project Explorer**, select the project you want to deploy.
5. Open the project's context menu.
6. Click **Deploy to...** → **container[xxx]**.
7. Check the **Console** view to see that your project builds successfully and is installed in the container.
8. In **Fuse JMX Navigator**, click **container[xxx]** → **Bundles** to populate the **Properties** view with the list of bundles installed in the container.
9. Use the search tool in Properties view to check that your project's bundle was installed.

When you start typing the name of your project the search tool will display the bundles whose names match the current string.

10. In **Fuse JMX Navigator**, open **container[xxx]**'s context menu.
11. Click **Refresh**.
12. Expand the **container[xxx]** node to view your project's nodes.

Now you can test and debug your application in a Red Hat JBoss Fuse environment.

Related topics

[Chapter 11, *The Fuse JMX Navigator*](#)

[Section 8.1.1, "Configuring a Container's Deploy Folder"](#)

[Chapter 4, Editing a routing context in the route editor](#)

[Chapter 14, Tracing messages through routes](#)

8.3. DEPLOYING A PROJECT TO A FABRIC CONTAINER

7/19/12

EMJ

rewrote for clarity and accuracy

Overview

When you are ready to test your application in a highly distributed environment, you can deploy it into a fabric using the **Deploy to...** tool. The **Deploy to...** tool builds your application, runs the tests, packages the application as a bundle, adds the bundle to a fabric profiles, and deploys the bundle into the fabric's internal Maven repository.

To complete the process, you need to ensure that the profile containing your application is deployed to at least one of the containers in the fabric. If the profile is not deployed to a container, the application will not be started.

Procedure

To deploy a project to a fabric container:

1. Open the **Fuse Integration** perspective.
2. In **Fabric Navigator**, expand the **Fabrics** node.
3. Double-click a fabric to connect to it.
4. Create a new profile in which you will install your project.

Follow the steps in [Section 24.1, "Creating a new profile"](#) up to [Step 6](#).

The new profile appears in the profile tree in **Fabric Navigator**.

5. In **Project Explorer**, select the project you want to deploy.
6. Open the project's context menu.
7. Select **Deploy to...** → **fabric** → **version** to see the list of available profiles.
8. From the list of profiles, select the profile that you created in [Step 4](#).
9. Verify that the application's bundle was added to the profile.

- a. In **Fabric Navigator**, click the profile into which the project was installed.

This will populate the **Properties** view with the profile's details.

- b. In **Properties** view, click the **Details** tab.
- c. Check that your project's bundle appears in the **Bundles** or **FABs** field.

10. If you do not already have a container deployed into the fabric for testing your application, create a new container in the fabric.

Follow the procedure appropriate for the container in [Chapter 23, Working with Fabric Containers](#).

The new container appears in **Fabric Navigator**, under the fabric's **Containers** node.

11. Assign the profile containing your application to a container in the fabric.

Follow the procedure in [Section 23.5, "Changing a Container's Profiles"](#).

12. Verify that the application was deployed to the container.

- a. In **Fabric Navigator**, double-click the new container to start it and to populate **Properties** view with its properties and profile information.
- b. In **Fabric Navigator**, click the container's **Bundles** node to view the list of bundles installed on it.

Related topics

[Chapter 23, Working with Fabric Containers](#)

[Chapter 24, Working with Fabric Profiles](#)

[Chapter 25, Working with Versions](#)

[Chapter 4, Editing a routing context in the route editor](#)

[Chapter 14, Tracing messages through routes](#)

PART II. DEBUGGING

	8/15/11	EMJ
reorged to group topics by user functions		
	8/17/11	EMJ
added new server docs		
	7/13/12	EMJ
collected all of the JMX explorer procedures in a single chapter		

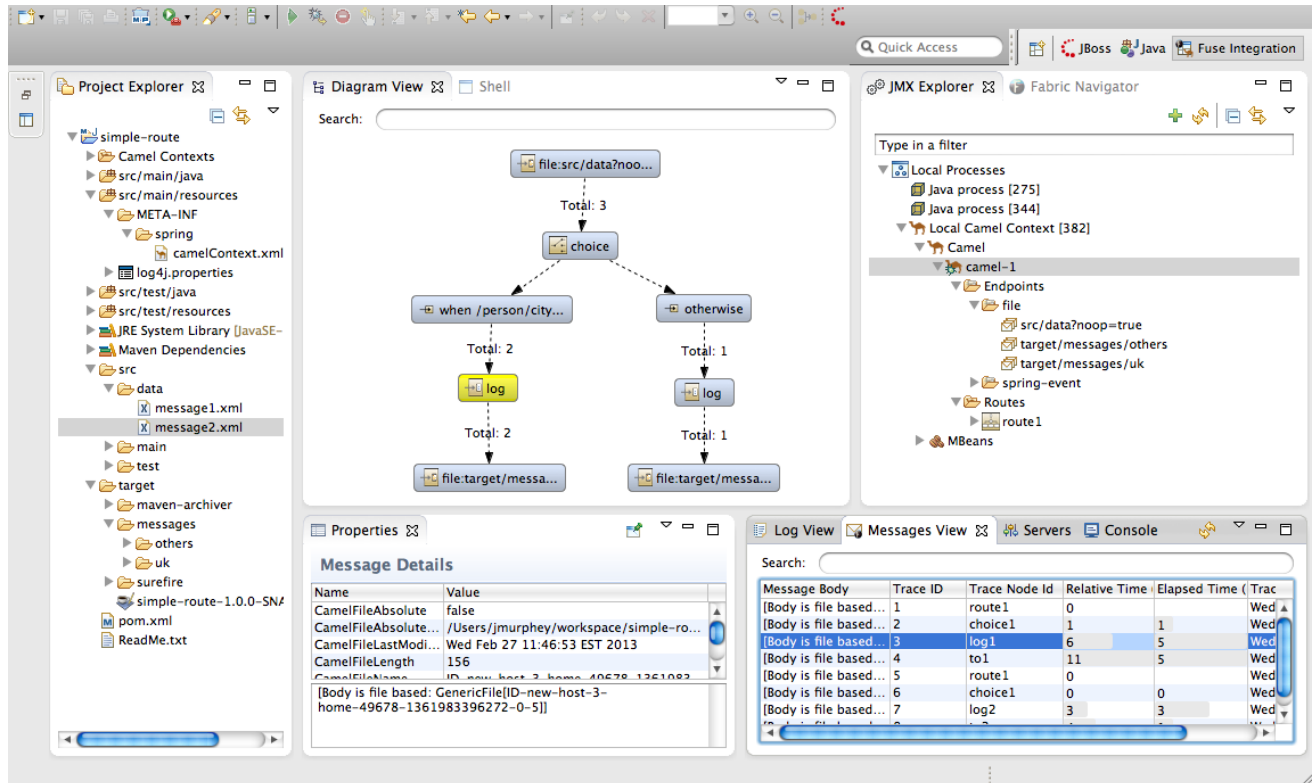
CHAPTER 9. DEBUGGING TOOLING

9.1. THE FUSE INTEGRATION PERSPECTIVE

Overview

The Fuse Integration perspective, shown in [Figure 9.1](#), is where you do most of your debugging.

Figure 9.1. Fuse Integration perspective



The Fuse Integration perspective consists of five main areas:

- **Fuse JMX Navigator**—lists the JMX servers and the infrastructure they monitor.
- **Diagram View**—provides a graphical representation of an item selected in the **Fuse JMX Navigator**. If the selected item is a route, Diagram view also displays timing metrics for each processing step in the route.
- **Messages View**—lists the messages that have passed through the selected JMS destination or Apache Camel endpoint.
- **Servers**—lists the Red Hat JBoss Fuse, Apache Karaf, or Apache ServiceMix containers defined in the debugging tooling environment.
- **Properties**—displays the properties of the selected item.

Fuse JMX Navigator

The **Fuse JMX Navigator** drives the debugging activities in the Fuse Integration perspective. It determines which routes are displayed in the **Diagram View**, the **Properties** viewer, and the **Messages View**. It also provides menu commands for activating route tracing, adding and deleting

JMS destinations, and starting and stopping routes. It is also the target for dragging and dropping messages onto a route.

By default, the **Fuse JMX Navigator** shows all of the Java processes running on your local machine. You can add JMX servers as needed to view infrastructure on other machines.

Messages view

The **Messages View** is used for JMS browsing and route tracing.

When a JMS destination is selected in the **Fuse JMX Navigator**, the view lists all of the messages sitting in the destination.

When a browsable Apache Camel endpoint is selected and route tracing has been activated, the view lists all of the messages that have passed through the endpoint since tracing was activated.

When a message in the **Messages View** is selected, its details are displayed in the **Properties** viewer. The message details include the message's body and all of its headers.

9.2. INFRASTRUCTURE MONITORING

Overview

A large portion of the Fuse Integration perspective is dedicated to visualizing the processes running in your infrastructure. In addition to displaying the running processes, the Fuse Integration perspective also displays all of the runtime metrics that are gathered from JMX MBeans.

Visualizations

The **Fuse JMX Navigator** and the **Diagram View** show you the processes deployed as part of your integration infrastructure.

The **Fuse JMX Navigator** arranges items in a tree and drives the rest of the operations on the Fuse Integration perspective.

Diagram View shows your infrastructure as a graph. It shows all of the nodes under the item selected in **Fuse JMX Navigator**.

For more information see [Chapter 11, *The Fuse JMX Navigator*](#).

JMX metrics

JMX metrics are displayed in the **Properties** view of the Fuse Integration perspective. Depending on the type of process, the JMX metrics provide details such as:

- time running
- time to process a message
- number of messages processed
- processor load
- memory footprint

For more information see [Chapter 12, *Viewing a component's properties*](#).

9.3. ROUTE DEBUGGING

Overview

When debugging a route, it is helpful to see how a test message changes as it passes through the route. It is also helpful to trace the message's path through the route.

The developer tools' route tracing feature enables you to do both. The **Diagram View** displays a graphical representation of the route. The **Messages View** displays the messages processed by the node selected in **Fuse JMX Navigator**. You can also display the full contents of a message in the **Properties** view.

Live debugging

Debugging a route in the Fuse Integration perspective enables you to see how the route functions in its actual deployment environment. The route is hooked up to live endpoints that are managed by live brokers. When you drag a test message on to a route's endpoint, you're adding it to that endpoint.

You can ensure that the start endpoint is properly configured to receive messages. As the message passes through each processor in the route, you can check the entire message, including all of the transport headers, to see if it is being processed correctly. You can also check that the target endpoint is properly configured to pass results to the intended target destination.

Visualization

Diagram View displays your routes as graphs. Each node of the graph represents a step in the route and is identified by an icon representing the EIP pattern the node represents.

With route tracing activated, you can trace, in **Diagram View**, each node that has processed a message by scrolling through the message list in **Messages View**.

9.4. JMS BROWSING

Overview

A browser that inspects the messages on a JMS destination without consuming them is key to troubleshooting a distributed message system. Inspecting the messages can help you understand why messages are backing up or why they are being discarded too soon.

In addition to inspecting JMS destinations, the debugging tooling allows you to add messages to them. This feature enables you to add controlled test cases into the system for detailed analysis.

Viewing messages

Fuse JMX Navigator shows all of the JMS destinations deployed in your infrastructure. When you select a destination from the tree, a list of the messages sitting in it are listed in **Messages View**. **Messages View** displays the basic properties of the message.

Selecting one of the messages in **Messages View** displays all of that message's details, including the full message body, in the **Properties** viewer.

Adding messages

You can simply drag messages from **Project Explorer**, **Package Explorer**, or **Messages View** and drop them on any JMS destination displayed in **Fuse JMX Navigator** or in **Fabric Explorer**.

CHAPTER 10. OPENING THE FUSE INTEGRATION PERSPECTIVE

7/13/12

EMJ

added descriptions of all the views that make up the perspective

OVERVIEW

The Fuse Integration perspective provides access to all of the debugging tooling. It contains all of the views used to monitor and debug Fuse integration projects including:

- **Project Explorer**

A standard Eclipse view that shows all of the projects known to the developer tools. You can view and select the artifacts that make up each project.

In **Project Explorer**, the debugging tooling displays all of the camel context files for a project under the project's **Camel Contexts** node. This makes it easy for you to find and open the context file of any route in a project.

- **Fuse JMX Navigator**

A customized JMX view that allows you to browse JMX servers and the processes they are monitoring. The **Fuse JMX Navigator** automatically lists all of the JMX servers running on the local machine. It also can identify instances of Red Hat processes.

- **Diagram View**

Displays a graphical tree representing the node selected in the **Fuse JMX Navigator**.

When you select a process, server, endpoint, or other node in **Fuse JMX Navigator**, the graphical tree in **Diagram View** shows the selected node as the root and branches down to the children and grandchildren. For example, if you select a broker in **Fuse JMX Navigator** the tree in **Diagram View** displays up to three children: connections, topics, and queues. It also shows configured connections and destinations as grandchildren.

When you select a route, the graphical tree includes all of the nodes in the route and the path that a message can take through it. If you select a context node, **Diagram View** shows all of the routes in the context.

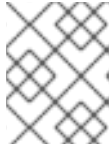
- **Shell**

Shows the command console of a container. You can control the connected container by entering commands into the shell.

- **Messages View**

Displays the list of messages sitting in the node selected in **Fuse JMX Navigator**. This view is populated only when a JMS destination or route endpoint is selected.

See [Chapter 13, *Browsing messages*](#) for more details.

**NOTE**

Route endpoints contain messages only when route tracing is activated. See [Section 14.3, “Tracing messages through a route”](#) for more information.

- **Servers**

Displays a list of the servers managed by the developer tools. It displays their runtime status and provides controls for stopping and starting them.


- **Console**

Displays the console output for recently executed actions.

- **Properties**

The **Properties** view displays the JMX properties for the node selected in **Fuse JMX Navigator**.

OPENING THE PERSPECTIVE

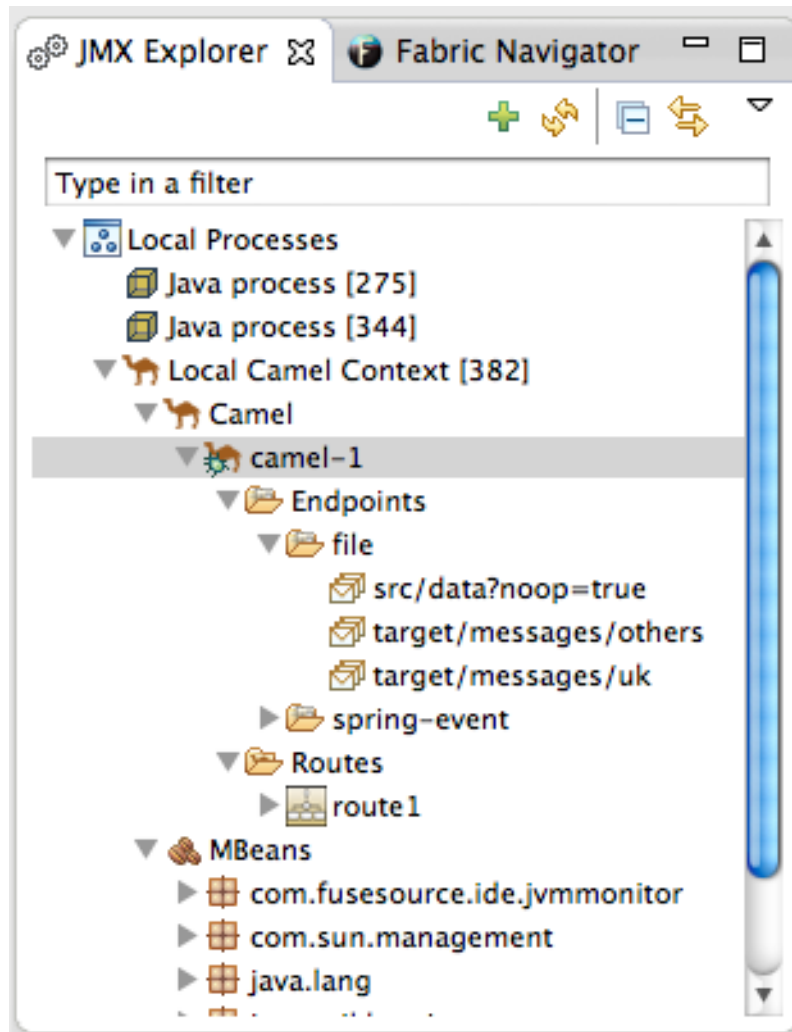
To open the **Fabric Integration** perspective, select **Window** → **Open Perspective** → **Fuse Integration**, or click  in the **Perspectives** tab, and select **Fuse Integration**.

RELATED TOPICS

CHAPTER 11. THE FUSE JMX NAVIGATOR

The **Fuse JMX Navigator**, shown in ???, displays all of the processes running in your application and drives all interactions with the debugging features. Other areas of the **Fuse Integration** perspective adapt to display information related to the node selected in the explorer. The explorer's context menu also provides the commands needed to activate route tracing and to add JMS destinations.

Figure 11.1. Fuse JMX Navigator



By default, the **Fuse JMX Navigator** discovers all of the JMX servers running on the local machine and lists them under the **Local Processes** tree. You can add other JMX servers using a server's JMX URL.

11.1. VIEWING PROCESSES IN JMX


Overview

The **Fuse JMX Navigator** lists all known process in a series of trees. The root for each tree is a JMX server.

The first tree in the list is a special **Local Processes** tree that contains all of the JMX servers running on the local machine. You must connect to one of the JMX servers to see the processes it contains.


Viewing processes in a local JMX server

To view information about processes in a local JMX server:

1. Expand the **Local Processes** entry in the **Fuse JMX Navigator**.
2. Double-click one of the top-level entries under **Local Processes** to connect to the JMX server.
3. Double-click it to open a connection.
4. Click the  icon that appears next to the entry to display the list of all components running in the JVM.

Viewing processes in alternate JMX servers

To view information about processes in an alternate JMX server:

1. [Add](#) the JMX server to the explorer.
2. Expand the server's entry in the **Fuse JMX Navigator** using the  icon that appears next to the entry to display the list of all components running in the JVM.

11.2. ADDING A JMX SERVER


Overview

The **Fuse JMX Navigator** lists all of the local JMX servers under the **Local Processes** branch of the tree. You may need to connect to specific JMX servers to see components deployed on other machines.

To add a JMX server you need to know the JMX URL of the server you want to add.

Procedure

To add a JMX server to the **Fuse JMX Navigator**:

1. Click  to the right of the **Fuse JMX Navigator** tab.
2. In the **Create a new JMX connection** wizard, select **Default JMX Connection**.
3. Click **Next>**.
4. Select the **Advanced** tab.
5. In the **Name** field, enter a name for the JMX server.

The name can be any string. It is used to label the entry in the **Fuse JMX Navigator** tree.

6. In the **JMX URL** field, enter the JMX URL of the server.
7. If the JMX server requires authentication, enter your user name and password in the **Username** and **Password** fields.
8. Click **Finish**.

The new JMX server is displayed as a branch in the **Fuse JMX Navigator** tree.

CHAPTER 12. VIEWING A COMPONENT'S PROPERTIES

OVERVIEW

The developer tools collect and display all of the JMX properties reported by Fuse components. This information can provide significant insight into what is happening in your integration application.

PROCEDURE

To see a Fuse component's properties:

1. Locate the node for the component in the **Fuse JMX Navigator**.

You may have to expand nodes on the tree to locate low-level components.

2. Select the Fuse component's node from the tree.
3. Open the **Properties** view.

The selected component's properties will be displayed in table format.

You can use the **Search** box in **Properties** view to locate specific properties based on value.

CHAPTER 13. BROWSING MESSAGES

8/15/11

EMJ

generalized for all endpoints and added list of things that can be browsed

OVERVIEW

A key tool in debugging applications in a distributed environment is seeing all of the messages stored in the JMS destinations and route endpoints in the application. The developer tools can browse the following:

- JMS destinations
- JMS routing endpoints
- Apache Camel routing endpoints
- SEDA routing endpoints
- Browse routing endpoints
- Mock routing endpoints
- VM routing endpoints
- DataSet routing endpoints

PROCEDURE

To browse messages:

1. Select the JMS destination or endpoint from **Fuse JMX Navigator**.

The list of messages is displayed in the **Messages View**.

2. Select an individual message to inspect from the list of messages in the **Messages View**.

Message details and content are displayed in the **Properties** view.

RELATED TOPICS

[Section 14.3, "Tracing messages through a route"](#)

CHAPTER 14. TRACING MESSAGES THROUGH ROUTES

Debugging a route usually involves solving one of two problems:

- a message was improperly transformed
- a message failed to reach its destination endpoint

Tracing one or more test messages through the route is easiest way to solve these problems.

The route tracing feature of the developer tools enable you to monitor the path a message takes through a route and see how the message is transformed as it passes from processor to processor.

Diagram View displays a graphical representation of the route, which enables you to see the path a message takes through it. For each processor in a route, it also displays the average processing time, in milliseconds, for all messages processed since route start-up and the number of messages processed since route start-up.

Messages View displays the messages processed by a JMS destination or route endpoint selected in the **Fuse JMX Navigator** tree. Selecting an individual message in **Messages View**, displays the full details and content of the message in the **Properties** view.

Tracing messages through a route involves the following steps:

1. [Creating test messages for route tracing](#)
2. [Activating route tracing](#)
3. [Tracing messages through a route](#)
4. [Deactivating route tracing](#)

14.1. CREATING TEST MESSAGES FOR ROUTE TRACING

Overview

Route tracing requires that messages have a specific structure. They must have a **message** root element in the namespace `http://fabric.fusesource.org/schema/messages`.

Procedure

To create a test message:

1. In **Project Explorer**, right-click the project to open the context menu.
2. Select **New** → **Fuse Message** to open the **Fuse Message File** wizard.
3. In the **RouteContainer** field, enter the path of the folder in which you want to store the message.

Use the **Browse...** button to help locate the proper folder.

4. In the **File name** field, enter a name for the message.
5. Click **Finish**.

The new message opens in the XML editor.

6. Fill in the contents (body and header text) of the message.
7. On the menu bar, click **File** → **Save** to save the test message.

Related topics

[Section 14.3, “Tracing messages through a route”](#)

[New Fuse Message](#)

14.2. ACTIVATING ROUTE TRACING

Overview

Before you can trace messages through a route, you must activate route tracing for the route's routing context.

Procedure

To start tracing on a routing context:

1. In the **Fuse JMX Navigator** tree, select the routing context on which you want to start tracing.

You can select any route in the context to start tracing on the entire context.

2. Right-click it to open the context menu, and then select **Start Tracing**.

If **Stop Tracing Context** appears on the context menu, tracing is already active.

Related topics

[Section 14.3, “Tracing messages through a route”](#)

[Section 14.4, “Deactivating route tracing”](#)

14.3. TRACING MESSAGES THROUGH A ROUTE

Overview

The best way to see what's happening in a route is to watch what happens to a message at each stop along the route. The debugging tooling provides a mechanism for dropping messages into a route and tracing the message's path through it.

Procedure

To trace messages through a route:

1. Create one or more test messages as described in [Section 14.1, “Creating test messages for route tracing”](#).
2. Activate tracing for the route's routing context as described in [Section 14.2, “Activating route tracing”](#).
3. Drag one of the test messages onto the route's starting point.
4. In the **Fuse JMX Navigator** tree, select the route being traced.

The **Messages View** is populated with a list of messages representing the message at each stage in the traced route.

5. Open **Diagram View** to see a graphical representation of the selected route.
6. In the **Messages View**, select one of the messages.

In **Diagram View**, the route step in which the selected message is sitting is highlighted. The **Properties** view displays the full details and content of the test message.

You can repeat this process as needed.

Related topics

[Section 14.1, “Creating test messages for route tracing”](#)

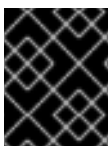
[Section 14.2, “Activating route tracing”](#)

[Section 14.4, “Deactivating route tracing”](#)

14.4. DEACTIVATING ROUTE TRACING

Overview

When you are finished debugging the routes in a routing context, you should deactivate tracing for the routing context.



IMPORTANT

Deactivating tracing stops tracing and flushes the trace data for all of the routes in the routing context. This means that you cannot review any past tracing sessions.

Procedure

To stop tracing for a routing context:

1. In the **Fuse JMX Navigator** tree, select the routing context for which you want to deactivate tracing.

You can select any route in the context to stop tracing for the context.

2. Right-click it to open the context menu, and then select **Stop Tracing Context**.

If **Start Tracing** appears on the context menu, tracing is not activated for the routing context.

Related topics

[Section 14.2, “Activating route tracing”](#)

[Section 14.3, “Tracing messages through a route”](#)

CHAPTER 15. MANAGING JMS DESTINATIONS

The developer tools' **Fuse JMX Navigator** allows you to easily add or delete JMS destinations from a running Red Hat JBoss Fuse A-MQ.



IMPORTANT

These changes are not persistent across broker restarts.

15.1. ADDING A JMS DESTINATION

Overview

When testing a new scenario, it is convenient to add a new JMS destination to one of your brokers. The developer tools provide a menu choice for adding a destination with a single click.

Procedure

To add a JMS destination to a broker:

1. In the **Fuse JMX Navigator** tree, select either the **Queues** child or the **Topics** child in the broker node where you want to add a destination.
2. Right-click it to open the context menu, and then select **Create Queue/Topic**.
3. In the naming dialog, enter a name for the destination.
4. Click **OK**.

The new destination appears in the tree.

Related topics

[Section 15.2, "Deleting a JMS destination"](#)

15.2. DELETING A JMS DESTINATION

Overview

When testing fail over scenarios or other scenarios that involve failure handling, it's convenient to be able to easily remove a JMS destination. The developer tools have a menu option that allows you to remove a destination with a single click.

Procedure

To delete a JMS destination:

1. In the **Fuse JMX Navigator** tree, select the JMS destination you want to delete.
2. Right-click it to open the context menu, and then select **Delete Queue/Topic**.

Related topics

[Section 15.1, “Adding a JMS destination”](#)

CHAPTER 16. MANAGING ROUTING ENDPOINTS

The **Fuse JMX Navigator** allows you to easily add or delete routing endpoints.



IMPORTANT

These changes are not persistent across router restarts.

16.1. ADDING A ROUTING ENDPOINT

Overview

When testing a new scenario, it is convenient to add a new endpoint to a routing context. The developer tools provide a menu choice for adding a routing endpoint with a single click.

Procedure

To add an endpoint to a routing context:

1. In the **Fuse JMX Navigator** tree, select the **Endpoints** child in the routing context node where you want to add an endpoint.
2. Right-click it to open the context menu, and then select **Create Endpoint**.
3. In the naming dialog, enter a URL defining the new endpoint.

See .

4. Click **OK**.

The new destination appears in the tree.

Related topics

[Section 16.2, “Deleting a routing endpoint”](#)

16.2. DELETING A ROUTING ENDPOINT

Overview

When testing fail over scenarios or other scenarios that involve failure handling, it is convenient to be able to easily remove an endpoint from a routing context. The developer tools have a menu option that allows you to remove an endpoint with a single click.

Procedure

To delete a routing endpoint:

1. In the **Fuse JMX Navigator** tree, select the endpoint you want delete.

2. Right-click it to open the context menu, and then select **Delete Endpoint**.

Related topics

[Section 16.1, “Adding a routing endpoint”](#)

CHAPTER 17. EDITING RUNNING ROUTES

07/23/12

EMJ

Streamlined the steps about saving the changes

OVERVIEW

If you want to experiment with changes to a running route, you can make the edits directly from the Fuse Integration perspective. The developer tools open the context file containing the route in the route editor. When your changes are saved, they take effect immediately.

The changes are made to the in-memory model of the running context, but are not persisted as part of their original project or any other project.



NOTE

If you want to save the changes permanently, you must use the **Save As** menu option.

PROCEDURE

To edit a running route:

1. In the **Fuse JMX Navigator** tree, select the context containing the route(s) you want to edit.
2. Right-click to open the context menu, and select **Edit Routes**.

Route editor displays the context and all of its routes.

3. Edit the route(s) as described in [Part I, “Developing Applications”](#).
4. When done, save the edited route.

- o Select **File** → **Save** to update the in-memory version of the routing context.

Changes take effect immediately.

- o >Select **File** → **Save As** to save the edited routing context in a new context file.

RELATED TOPICS

[Section 4.1, “Adding routes to the routing context”](#)

[Section 4.2, “Adding patterns to a route”](#)

[Section 4.4, “Configuring a pattern”](#)

[Section 4.3, “Connecting patterns to make a route”](#)

[Section 4.5, “Rearranging patterns on the canvas”](#)

[Section 4.6, “Removing patterns from a route”](#)

Section 4.7, “Disconnecting two patterns”

Section 4.8, “Deleting a route”

CHAPTER 18. MANAGING ROUTING CONTEXTS

The **Fuse JMX Navigator** enables you to easily suspend, shutdown, and resume routing contexts.

18.1. SUSPENDING A ROUTING CONTEXT

Overview

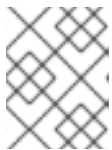
The developer tools enable you to suspend the operation of a routing context from **Fuse JMX Navigator**. Suspending a context gracefully shuts down all of the routes in the context, but keeps them loaded in memory so that they can easily be resumed.

If you want to kill all of the routes and free their resources, close the context. See [Section 18.3, “Shutting down a routing context”](#).

Procedure

To suspend a routing context:

1. In the **Fuse JMX Navigator** tree, expand the project's **Camel Contexts** node, and select the routing context you want to suspend.
2. Right-click it to open the context menu, and then select **Suspend Camel Context**.



NOTE

If **Resume Camel Context** appears on the context menu, the context is already suspended.

Related topics

[Section 18.3, “Shutting down a routing context”](#)

[Section 18.2, “Resuming a routing context”](#)

18.2. RESUMING A ROUTING CONTEXT

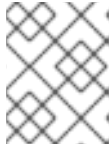
Overview

The developer tools enable you to resume a suspended routing context from **Fuse JMX Navigator**. Resuming a context restarts all of the routes in it, so that they can process messages.

Procedure

To resume a routing context:

1. In the **Fuse JMX Navigator** tree, expand the project's **Camel Contexts** node, and select the routing context you want to resume.
2. Right-click it to open the context menu, and then select **Resume Camel Context**.

**NOTE**

If **Suspend Camel Context** appears in the context menu, the context and its routes are running.

Related topics

[Section 18.1, “Suspending a routing context”](#)

18.3. SHUTTING DOWN A ROUTING CONTEXT

Overview

The developer tools enable you to shut down a routing context from **Fuse JMX Navigator**. Shutting down a routing context gracefully stops all of the routes in the context and releases all of the resources used by the routes. Once shut down, a routing context cannot be resumed.

If you want to stop the routes so that they can be restarted, suspend the context. See [Section 18.1, “Suspending a routing context”](#).

Procedure

To shut down a routing context:

1. In the **Fuse JMX Navigator** tree, expand the project's **Camel Contexts** node, and select the routing context you want to shut down.
2. Right-click it to open the context menu, and then select **Close Camel Context**.

Related topics

[Section 18.1, “Suspending a routing context”](#)

CHAPTER 19. MANAGING SERVERS

07/24/2012

EMJ

Added section about removing servers

The **Servers** panel enables you to deploy and manage servers in your Eclipse environment. The debugging tooling supports deploying the following servers:

- Red Hat JBoss Fuse 6.x Server
- Apache Karaf 2.x Server
- Apache ServiceMix 4.x Server

19.1. ADDING A SERVER

07/24/2012

EMJ

Added overview

07/24/2012

EMJ

Corrected procedure

Overview

To make it possible for the developer tools to manage a server, it needs to be added to the **Servers** list. Once added, the server will appear in the **Servers** view and in the **Servers** tree in both **Project Explorer** and **Package Explorer**.

You can add multiple servers of the same type, as long as each uses a separate installation directory.



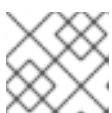
NOTE

If adding a Red Hat JBoss Fuse server, it's recommended that you edit its *installDir/etc/users.properties* file and add user information, in the form of **user=password,role**, to enable the developer tools to establish an ssh connection to the server.

Procedure

To add a new server to the **Servers** list:

1. From the main menu, select **File** → **New** → **Other** to open the **New** wizard.
2. Select **Server** → **Server** from the list.
3. Click **Next>**.
4. In the **Define a New Server** dialog, select the type of server you want to create from the list.
5. In **Server's host name**, accept the default (**localhost**) or enter the name of the host on which the server runs.



NOTE

Currently the debugger does not support servers running on remote machines.

6. In **Server name**, accept the default, or enter a different name for the server.
7. If the **Server runtime environment** drop-down list is present (when **Red Hat JBoss Fuse 6.x** is selected), accept the default or select another runtime environment from the list.
8. Click **Next>**.
9. If you have not defined a server of this type before, you must do so now in the **New ServerType Runtime** dialog.

- a. In **Installation directory**, enter the full path to the server's installation folder.

The  button opens a file browser.

- b. Click **Next>**.

Host Name defaults to the address of the machine running the server. For **localhost**, the address is **0.0.0.0**.

10. In **Port Name**, accept the default (**8101**), or enter the port number that the debugger will use to connect to the server.
11. In **User Name**, enter the user name that the debugger will use to connect to the server.

For a Red Hat JBoss Fuse server, this is a user name that is stored in the *installDir/jboss-fuse-6.0.0.redhat-xxx/etc/users.properties* file.

If one has not been set, you can either add one to that file using the format **user=password,role** (for example, **admin=admin,admin**), or you can set one using the `karaf jaas` command set:

- **jaas:realms**—to list the realms
- **jaas:manage --index 1**—to edit the first (server) realm
- **jaas:useradd <username> <password>**—to add a user and associated password
- **jaas:roleadd <username> admin**—to specify the new user's role
- **jaas:update**—to update the realm with the new user information

If a `jaas` realm has already been selected for the server, you can discover the user name by issuing the command `JBossFuse:karaf@root>jaas:users`.

12. In **Password**, enter the password associated with the user name that the debugger will use to connect to the server.

For a Red Hat JBoss Fuse server, this is the password set either in the *installDir/jboss-fuse-6.0.0.redhat-xxx/etc/users.properties* file or by the `karaf jaas` commands.

13. Click **Next>**.

The **New ServerType Add and Remove** page opens.

14. Move the resources you want the server to use from the **Available** column to the

Configured column.

15. Click **Finish**.

The new server appears in the list of servers in the **Servers** panel.

Related topics

Define a New Server
New Server Configuration
Add and Remove

19.2. STARTING A SERVER


	07/24/2012	EMJ
Added overview		
	07/24/2012	EMJ
Corrected procedure		

Overview

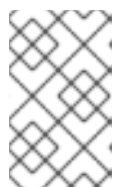
When you start a configured server, the developer tools open the server's remote management console in a **Shell** view. This allows you to easily manage the container while debugging your application.

Procedure

To start a server:

1. In the **Servers** panel, select the server you want to start.
2. Click  .

The **Console** view should open and display a message asking you to wait while the container is starting. A **Shell** view will open that displays the container's management console.



NOTE

If you did not properly configure the user name and password for opening the remote console, a dialog opens asking you to enter the proper credentials. See [Section 19.1, “Adding a Server”](#)

19.3. STOPPING A SERVER

	07/24/2012	EMJ
Added overview		
	07/24/2012	EMJ
Corrected procedure		


Overview

You can shutdown a server in one of two ways:

- from the **Server** view
- from the server's remote console

Using the server view

To stop a server:

1. In the **Servers** view, select the server you want to stop.
2. Click  .

Using the remote console

To stop a server:

1. Open the **Shell** view hosting the server's remote console.
2. Type **CTRL+D**.

19.4. DELETING A SERVER

Overview

When you are finished with a configured server, or if you misconfigure a server, it is easy to delete it's configuration.

Procedure

To delete a server:

1. In the **Servers** panel, select the server you want to delete.
2. Select **Delete** from the server's context menu.
3. Click **OK**.

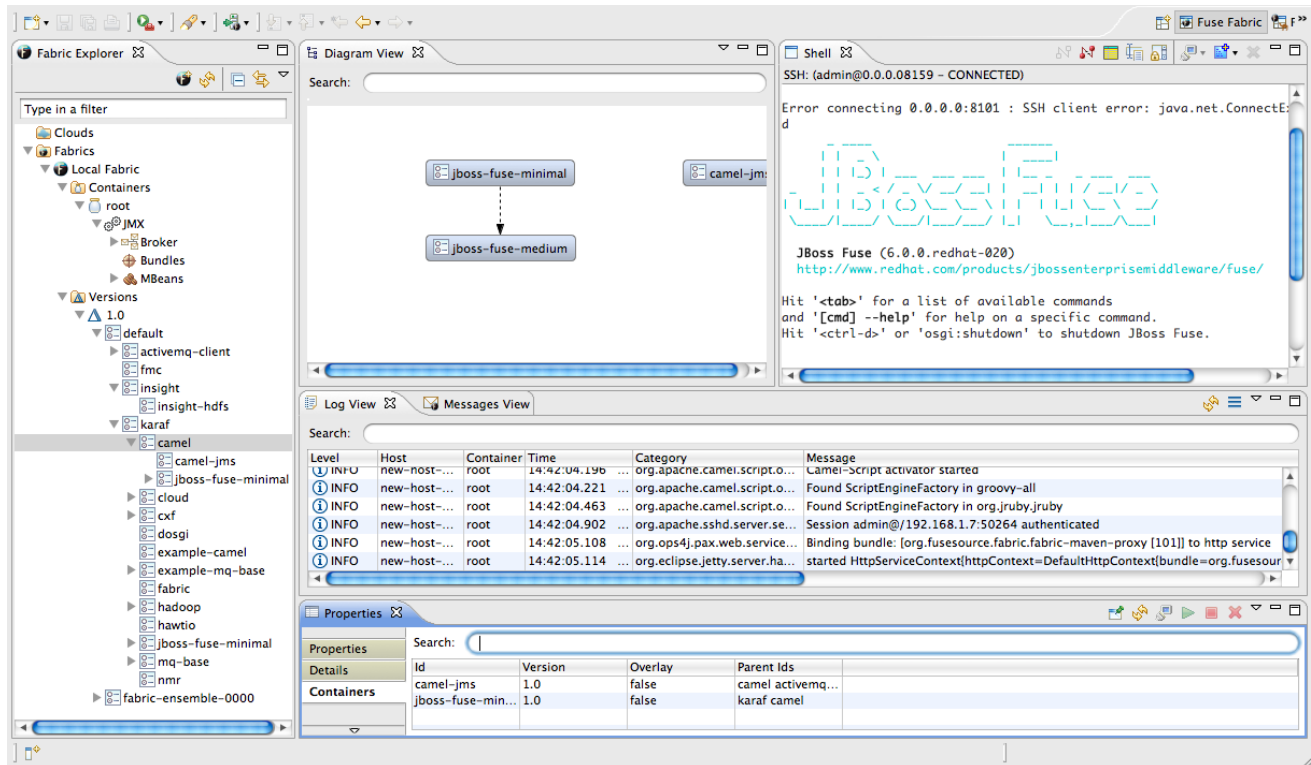
PART III. WORKING WITH FABRICS

A fabric is a distributed configuration, provisioning, and clustering mechanism for running Red Hat JBoss Fuse Plugins for Eclipse projects on multiple computers in an on-premises or public Cloud. You can view and edit fabrics, and the containers, versions, and profiles of which they consist. You can also create and delete containers and profiles, but you can only create versions.

CHAPTER 20. THE FUSE FABRIC PERSPECTIVE

The Fuse Fabric perspective, shown in [Figure 20.1](#), is where you access the fabric development and debugging tools.

Figure 20.1. Fuse Fabric perspective



The Fuse Fabric perspective consists of six main areas:

- **Fabric Navigator**—lists the fabrics and the containers, profiles, and versions of which they consist.
- **Diagram View**—provides a graphical representation of a node selected in **Fabric Navigator**.
- **Shell**— provides console access to any container running on the fabric.
- **Log view**—lists the log entries of the selected container or the selected JMX process.
- **Messages View**—lists the message instances that passed through the nodes in a selected route, after tracing was enabled on the route.
- **Properties view**—displays the properties of an object selected in **Fabric Navigator**.

FABRIC NAVIGATOR

Fabric Navigator provides access to your fabrics. Once the developer tools are connected to a fabric, **Fabric Navigator** provides access to all of the fabric's components (containers, profiles, and profile versions) enabling you to configure, run, and debug your project in the fabric environment.

DIAGRAM VIEW

Use **Diagram View** in conjunction with **Messages View** to track a message through the nodes of a

route for which tracing has been activated. In **Diagram View**, the node which corresponds to the message instance selected in **Messages View** is highlighted. **Messages View** displays metrics for the selected message instance, including the time at which it exited the highlighted node.

LOG VIEW

Log view displays the log entries of the container selected in **Fabric Navigator**. This feature enables you to see the log messages of any container running in a fabric.

Double-clicking a process in a container's JMX tree opens the process in a jconsole-like viewer, where you can browse its attributes, operations, notifications, and other information, all of which are presented in tabular format.

MESSAGES VIEW

The **Messages View** is used for tracing messages through a route in a fabric environment.

When an Apache Camel route for which route tracing has been activated is selected in **Fabric Navigator**, **Messages View** lists all messages that pass through the route from the time tracing was activated. **Messages View** also provides metrics for each message instance, making it easier to debug routes.

Use the view's **Refresh** button periodically to update its display of message traces.

PROPERTIES VIEW

For container nodes, the **Properties** view also provides the **Profiles** tab, for viewing the profiles assigned to the container, and the **Profile Details** tab, for viewing the details of each assigned profile.

The **Profiles** editor enables you to change the profiles associated with the selected container, and the **Profile Details** editor enables you to modify the configuration of the selected profile.

JMX VIEWER

Double-clicking a process in a container's JMX tree opens the process in a jconsole-like viewer, where you can browse its attributes, operations, notifications, and other information, all of which are presented in tabular format.

CHAPTER 21. SETTING UP A FABRIC ENVIRONMENT

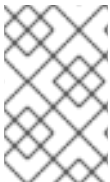
The developer tools provide an environment for developing and debugging fabric applications. Use the developer tools to build a mock-up of your actual fabric into which you can deploy your application to test and debug it.

OVERVIEW

To set up a fabric environment, you must have access to an existing fabric.

Setting up a fabric environment involves several basic steps:


- Providing the details for connecting to an existing fabric
- Connecting to the fabric
- Creating the containers that will host the various components of your distributed integration application
- Assigning each container one or more profiles, which provision containers by deploying and installing specified applications.



NOTE

For cloud applications, unless the cloud already has a fabric registry agent running on it, you must create one yourself, before you can include it to your fabric environment. For details, see [Chapter 26, *Creating a Fabric in the Cloud*](#).

OPENING THE FABRIC PERSPECTIVE

To set up a fabric environment, you must switch to the Fuse Fabric perspective, by selecting **Window** → **Open Perspective** → **Fuse Fabric**, or clicking  in the Perspectives tab, and selecting **Fuse Fabric**.

RELATED TOPICS

[Chapter 22, *Specifying and Connecting to a Fabric*](#)

CHAPTER 22. SPECIFYING AND CONNECTING TO A FABRIC

Before you can view and edit your fabric, you must add it to the **Fabric Navigator** view. You can add multiple fabrics in Fuse Fabric perspective and connect to them concurrently.

22.1. ADDING FABRIC DETAILS

Overview

For the developer tools to connect to a fabric, you must provide certain details about the fabric.

Procedure

To specify the details for connecting to a fabric:

1. In **Fabric Navigator**, right-click **Fabrics** to open the context menu, and then select **Add Fabric details** to open the **Fabric Details** wizard, as shown in [Figure 22.1](#).

Figure 22.1. Fabric Details wizard

2. In **Name**, enter the name of the fabric to which you want to connect. The name you enter identifies the fabric whose location you specify in **URLs**, and this name will appear in **Fabric Navigator**.

The default **Name** is Local Fabric.

3. In **URLs**, enter the url, in the form **hostname:port**, of the fabric to which you want to connect. This URL specifies the location of a *fabric registry agent*, whose default port is *2181*.

The default URL is localhost:2181.

4. In **User name**, enter the name used to log into the specified fabric.

This is the new user name specified when the fabric was created or is stored in Red Hat JBoss Fuse Plugins for Eclipse's *installDir/etc/users.properties* file. In that file, user information is specified using this format: **user=password,role** (for example, **admin=admin,admin**).

You can also discover the user name by issuing the command

JBossFuse:karaf@root>jaas:users, if the Jaas realm has been selected for the fabric.

5. In **Password**, enter the password required for **User name** to log into the specified fabric.

This is the password specified for the new user name when the fabric was created or is stored in Red Hat JBoss Fuse Plugins for Eclipse's *installDir/etc/users.properties* file.

6. In **Zookeeper Password**, enter the password required for logging into the specified fabric's zookeeper registry.

This is the password that was specified or generated when the fabric was created, or it is the user's password stored in Red Hat JBoss Fuse Plugins for Eclipse's *installDir/etc/users.properties* file.

You can also discover the Zookeeper password by issuing the command

JBossFuse:karaf@root>fabric:ensemble-password.

7. Click **OK**.

The fabric's name appears in **Fabric Navigator** as a node beneath **Fabrics**.



NOTE

Before you can view or modify a fabric's containers, profiles or versions, you must connect to the fabric. For details see [Section 22.2, "Connecting to a fabric"](#)

Related topics

[Section 22.5, "Deleting a fabric's details"](#)

22.2. CONNECTING TO A FABRIC


Overview

The developer tools connect to a fabric using the connection information you specified for it in the **Fabric Details** dialog.

Procedure

To connect to a fabric:

1. In **Fabric Navigator**, select the fabric to which you want to connect, then right-click it to open the context menu.
2. Select **Connect**.

An expand arrow () appears next to the fabric you selected in **Fabric Navigator**.

3. Click () to expand the fabric's tree and browse its containers, profiles, and versions.

From here, you can create and deploy new containers or modify existing ones by assigning them new profiles or modified versions of existing profiles.

Related topics

[Section 22.1, “Adding fabric details”](#)

22.3. DISCONNECTING FROM A FABRIC

Overview

Typically, you'd disconnect from a fabric without stopping its containers since they usually run application services. In this case, the containers will continue to run, unless stopped using the developer tools or the Red Hat JBoss Fuse console command line.

Procedure

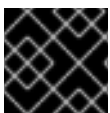
1. In **Fabric Navigator**, select the fabric from which you want to disconnect, then right-click it to open the context menu.
2. Select **Disconnect**.

The developer tools close the connection with the selected fabric.

22.4. EDITING A FABRIC'S DETAILS

Overview

When any of the details for connecting to your fabric change, you need to update the details in the developer tools.



IMPORTANT

Before you edit a fabric's details, disconnect from the fabric.

Procedure

1. In **Fabric Navigator**, select the fabric whose details you want to edit, and right-click it to open the context menu.
2. Select **Edit Fabric details** to open the **Fabric Details** dialog, as shown in [Figure 22.1, “Fabric Details wizard”](#).
3. In **Name**, enter a new name for the fabric, if necessary.
4. In **URLs**, enter the new url of the fabric, if necessary.
5. In **User name**, enter a name with which the new user will log into the selected fabric, if necessary.
6. In **Password**, enter the new password to use for logging into the selected fabric, if necessary.

7. Click **OK**.

22.5. DELETING A FABRIC'S DETAILS

Overview

To remove a fabric from **Fabric Navigator**, you delete its details.

Procedure

1. In **Fabric Navigator**, select the fabric whose details you want to delete, then right-click it to open the context menu.
2. Select **Delete Fabric details**.

The developer tools close the connection with the selected fabric, and then delete the fabric's details.

Related topics

[Section 22.1, "Adding fabric details"](#)

CHAPTER 23. WORKING WITH FABRIC CONTAINERS

7/19/12

EMJ

Added section about changing a containers profiles

23.1. CREATING A NEW CHILD CONTAINER

Overview

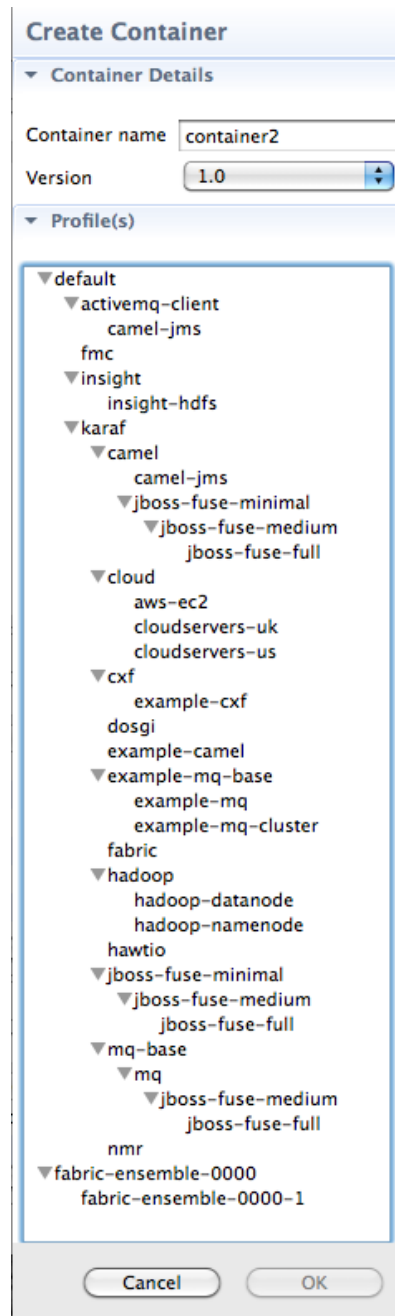
A child container is created on the server where its parent is.

Procedure

To create a new child container:

1. If necessary, in **Fabric Navigator**, expand the tree of the selected fabric in which you want to create the new child container.
2. Right-click the target parent container to open the context menu, and then select **Create a new child container**.

Figure 23.1. Create Child Container wizard

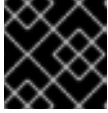


3. In **Container name**, enter a name for the new child container.

We recommend that you replace the default name with a meaningful name that identifies the container's role or function.

4. If multiple versions exist, you can select one from the **Version** drop-down list. Otherwise, accept the default value.
5. Under **Profiles**, click the checkbox next to the profile or profiles you want to assign to the new child container.

Profiles determine the function of containers; what applications they run. You can assign multiple profiles to a container, as long as the applications they install do not conflict with one another.

**IMPORTANT**

Do not assign a *base profile* (one ending in **-base**) to a container.

6. Click **OK**.

The new child container appears in **Fabric Navigator** as a node under **Containers**

**NOTE**

Selecting the new container in **Fabric Navigator** populates its **Profiles** and **Profile Details** pages in **Properties** viewer.

Related topics

[Section 24.1, “Creating a new profile”](#)

[Section 25.1, “Creating a new version of a profile”](#)

23.2. CREATING A CONTAINER ON A REMOTE HOST

Overview

To create a container on a remote host, the remote host machine must be ssh-enabled.

**NOTE**

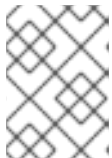
Creating a container via ssh on a remote Windows machine is not supported.


Procedure

To create a new container on a remote machine:

1. If necessary, in **Fabric Navigator**, expand the tree of the fabric for which you want to create the new remote container.
2. Right-click **Containers** to open the context menu, and then select **Create a new container via SSH**.

Figure 23.2. Create Container via SSH wizard

**NOTE**

An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

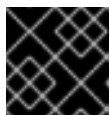
3. In **Container name**, enter a name for the new container.

We recommend that you replace the default name with a meaningful name that identifies the container's role or function.

4. If multiple versions exist, you can select one from the **Version** drop-down list. Otherwise, accept the default value.

5. In **Host**, enter the name or the IP address of the remote host.
6. In **User name**, enter the name of a user authorized to log into the remote host.
7. In **Password**, enter the user's password.
8. In **Path**, enter the path of the new container's location on the remote host.
9. In **Port**, accept the default port number (22), or enter a new port number to use for establishing an ssh connection on the remote host.
10. In **SSH retries**, accept the default number of retries (5), or enter the maximum number of retries to attempt at establishing an ssh connection on the remote host.
11. In **Retry delay**, enter the delay, in milliseconds, between retry attempts.
12. Under **Profiles**, click the checkbox next to the profile or profiles you want to assign to the container.

Profiles determine the function of a container; what applications it runs. You can assign multiple profiles to a container, as long as the applications they install do not conflict with one another.



IMPORTANT

Do not assign a *base profile* (one ending in **-base**) to a container.

13. Click **OK**.

The new container appears in **Fabric Navigator** as a container node under **Containers**

Selecting the new container in **Fabric Navigator** populates its **Profiles** and **Profile Details** pages in **Properties** viewer.

Related topics

[Section 24.1, "Creating a new profile"](#)

[Section 25.1, "Creating a new version of a profile"](#)

23.3. CREATING A NEW CONTAINER ON A CLOUD

Overview

To create a container on a cloud, you need to specify the cloud's connection details and then create the container.

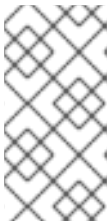
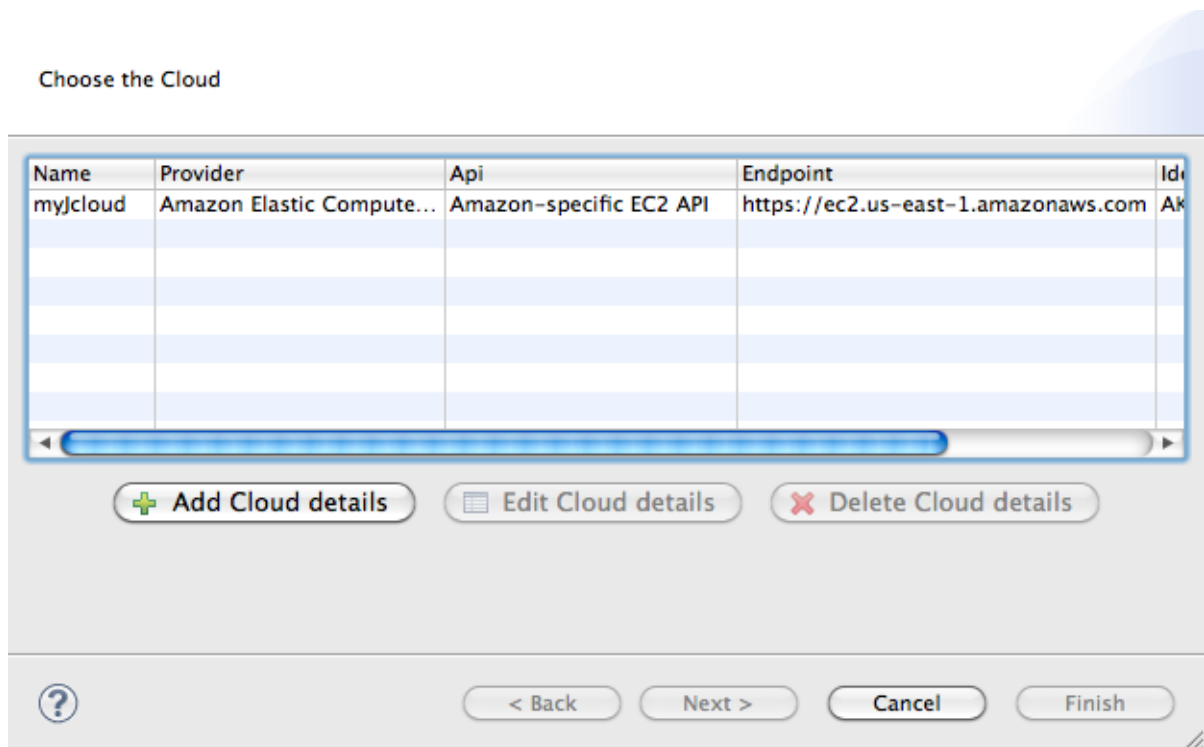
Procedure

To create a new container on a cloud:

1. In **Fabric Navigator**, right-click anywhere, except on **Clouds** or **Fabrics**, to open the context menu.

2. Select **Create a new container on the cloud** to open the **Create Container in a Cloud** wizard.

Figure 23.3. Create Container in a Cloud/Choose the Cloud page



NOTE

Unless the connection details for a cloud have already been added to the developer tools, this dialog opens devoid of selections. You can add a cloud's connection details now by clicking **Add Cloud details** and following the procedure in [Section 26.1, "Adding cloud details"](#).

3. Select a cloud from the list, and click **Next** to open the **New Containers Details** page.

Figure 23.4. New Containers Details page

New Containers Details

Create a container in the cloud... 2 errors detected

Please select one of image or OS Family/Version to create the container...

Container name: container2

Version: 1.0

Group: fabric

User: admin

Location ID: [Error icon]

Hardware ID: [Error icon]

OS Family: [Dropdown]

OS Version: [Dropdown]

Image ID: [Dropdown]

Clear Defaults


Profile(s)

- default
 - activemq-client
 - camel-jms
 - fmc
 - insight
 - insight-hdfs
 - karaf
 - camel
 - camel-jms
 - jboss-fuse-minimal
 - jboss-fuse-medium
 - jboss-fuse-full
 - cloud
 - aws-ec2
 - cloudservers-uk
 - cloudservers-us

? < Back Next > Cancel Finish



NOTE

An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

4. In **Container name**, enter a name for the new container.

We recommend that you replace the default name with a meaningful name that identifies the container's role or function.

5. If multiple versions exist, you can select one from the **Version** drop-down list. Otherwise, accept the default version.
6. In **Group**, enter the group in which to start up the new container.
7. In **User**, enter the name of the user who is authorized to login to the new container.

8. In **Location ID**, select from the drop-down list the region in which the machines you want to use are located.
9. In **Hardware ID**, select from the drop-down list the ID of the hardware archetype to use.

Hardware archetypes specify the amount of memory and compute power available for running applications on machine images.



IMPORTANT

The next three fields provide two methods for specifying the software configuration (operating system, application server, and applications) you want to use. Either select an OS Family (and optionally a version), or select an Image ID.

10. In **OS Family**, select from the drop-down list the family name of the operating system you want to use.

If you select an **OS Family**, do not also select an **Image ID**.

11. In **OS Version**, optionally select from the drop-down list the version for the operating system you selected.
12. If you selected an **OS Family**, skip this step. Otherwise, in **Image ID**, select from the drop-down list the ID of the image template to use for the software configuration.

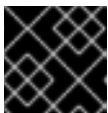


NOTE

The developer tools start an instance of the image in the cloud, and then install a fabric agent and provisions the new agent with the selected profiles. Depending on the cloud provider, instantiation and provisioning could take some time.

13. Under **Profiles**, select the profile or profiles you want to assign to the container.

Profiles determine the function of containers—what applications they run. You can assign a container multiple profiles, as long as the applications they install do not conflict with one another.



IMPORTANT

Do not assign a *base profile* (one ending in **-base**) to a container.

14. Click **OK**.

The new container appears in **Fabric Navigator** as a container node under **Containers**



NOTE

Selecting the new container in **Fabric Navigator** populates its **Profiles** and **Profile Details** pages in **Properties** viewer.

Related topics

[Section 24.1, “Creating a new profile”](#)

[Section 25.1, “Creating a new version of a profile”](#)

[Section 26.1, “Adding cloud details”](#)

23.4. STARTING A CONTAINER

Overview

When you create a container, it starts up automatically and continues to run until stopped by some event, planned or otherwise. You can manually restart a stopped container.

Procedure

To start a container:

1. If necessary, in **Fabric Navigator**, expand the tree of the fabric whose container you want to start.
2. Select the container and right-click it to open the context menu.
3. Click **Start Container**.



NOTE

To check whether the developer tools started the container, click **Containers** in **Fabric Navigator**, and open the **Properties** view. You should see something similar to [Figure 23.5](#), where **mycamelExample** was started and provisioned successfully.




Figure 23.5. Container status in Properties view

Id	Version	Profile Ids	Status	Alive	Provisioning Complete
myRoute	1.0	example-camel	● stopped	false	true
mycamelExample	1.0	example-camel	● success	true	true
root	1.0	fabric fabric-ensemble-0000-1	● root	true	false

The status indicators in the Status column of the **Properties** view indicate the startup state of the container, as shown [Table 23.1](#):

Table 23.1. Container startup status

Icon	Meaning
●	Container stopped

Icon	Meaning
	Container started; analysis and provisioning in progress
	Container started and provisioned successfully
	Failure occurred during startup or provisioning

Related topics

[Section 23.6, “Stopping a container”](#)

23.5. CHANGING A CONTAINER'S PROFILES

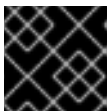
Overview

A container's profiles determine what the container is hosting. If you want to change what is deployed in a container or how it is configured, you change the profiles deployed to it.

Procedure

To change the profiles deployed to a container:

1. Select the container.
2. In the **Properties** view, select the **Profiles** tab.
3. Select the profile to be deployed to the container.



IMPORTANT

Your selection will override the list of profiles currently deployed to the container.



NOTE

You can select multiple profiles by holding the **CTRL** key

23.6. STOPPING A CONTAINER

Overview

There are times when you might want to stop then restart a container; for example, to recycle a hung container, force a container to reprovision itself, or to reclaim CPU cycles.

Procedure

To stop a container:

1. If necessary, in **Fabric Navigator**, expand the tree of the fabric whose container you want to stop.
2. Select the container and right-click it to open the context menu.
3. Click **Stop Container**.



NOTE

To check whether the developer tools stopped the container, click **Containers** in **Fabric Navigator**, and open the **Properties** view. You should see something similar to [Figure 23.6](#), where **myRoute** was stopped successfully.

Figure 23.6. Container status in Properties view

Id	Version	Profile Ids	Status	Alive	Provisioning Complete
myRoute	1.0	example-camel	● stopped	false	true
mycamelExample	1.0	example-camel	● success	true	true
root	1.0	fabric fabric-ensemble-0000-1	● root	true	false

For information on the meaning of the status indicators that might appear in the Status column of the **Properties** view, see [Table 23.1](#), “Container startup status”.

Related topics

[Section 23.4, “Starting a container”](#)

[Section 23.7, “Deleting a container”](#)

23.7. DELETING A CONTAINER

Overview

When a container becomes obsolete—no longer used in the fabric integration project—you may want to delete it. Before you do so, make sure no other container in the project depends on it. Deleting a container terminates the container and its processes and removes its configuration from the fabric.



NOTE

To delete a container, it must be running in the fabric.

Procedure

To delete a container:

1. If necessary, in **Fabric Navigator**, expand the tree of the fabric whose container you want to delete.
2. Click **Containers** to populate **Properties** view with a list of the fabric's containers.


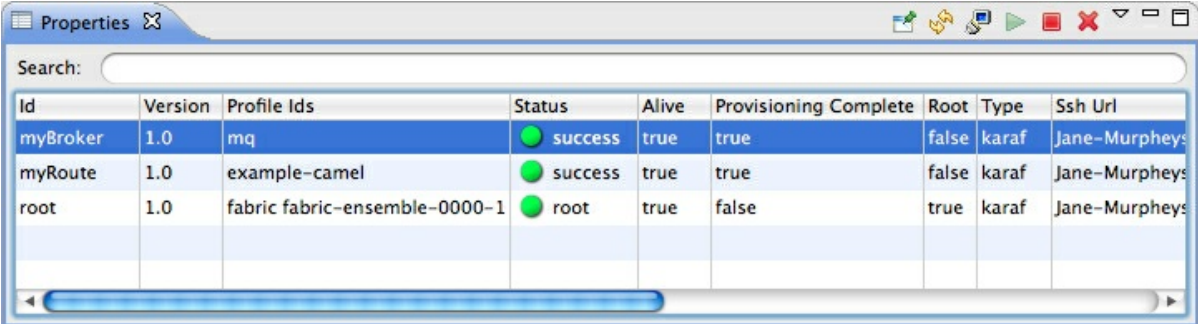
- In **Properties** view, select the container you want to delete, and then click  on the right-hand side of the toolbar.

Figure 23.7. Container selected for deletion in Properties view



Id	Version	Profile Ids	Status	Alive	Provisioning Complete	Root	Type	Ssh Url
myBroker	1.0	mq	success	true	true	false	karaf	Jane-Murphey:
myRoute	1.0	example-camel	success	true	true	false	karaf	Jane-Murphey:
root	1.0	fabric fabric-ensemble-0000-1	root	true	false	true	karaf	Jane-Murphey:

- In the **Destroy Container(s)** dialog, click **OK** to delete the selected container.

The developer tools display a message that removal is in progress. In **Properties** view, the container's status briefly changes to **stopped**, and then the container disappears from the list.

- In **Fabric Navigator**, open the fabric's context menu, and click **Refresh** to update **Containers** and remove the deleted container from the tree.



NOTE

If you have the Red Hat JBoss Fuse console open in **Shell** view, you can verify that the container and its configuration have been removed from the fabric by entering at the command line **fabric:container-list**.

Related topics

[Section 23.6, “Stopping a container”](#)

[Section 23.1, “Creating a new child container”](#)

[Section 23.2, “Creating a container on a remote host”](#)

[Section 23.3, “Creating a new container on a cloud”](#)

CHAPTER 24. WORKING WITH FABRIC PROFILES

Profiles are applied to fabric containers, and they determine the function and behavior of their containers. A profile consists of code and configurations, which specify the type of service or application a container runs. The code may come from OSGi bundles, Fab Application Bundles (see, [Fab Deployment Model](#) in the Red Hat JBoss Fuse Deploying into the Container guide), and Karaf features pulled from repositories, also specified in the profile. Defined hierarchically, profiles can be set up to inherit from other profiles, building on their parents' definitions.

24.1. CREATING A NEW PROFILE

Overview

The developer tools provide access to numerous profiles, each of which imparts specific functionality and behaviors to the containers to which it's assigned. But you may want to create new profiles that inherit from these base profiles, so you can override, for example, configuration details. You may also want to create your own profiles to deploy your own code. To do so, you create a new profile based on an existing one, and then change its configuration and properties to generate the specific behaviors you want to deploy to the target containers.



NOTE

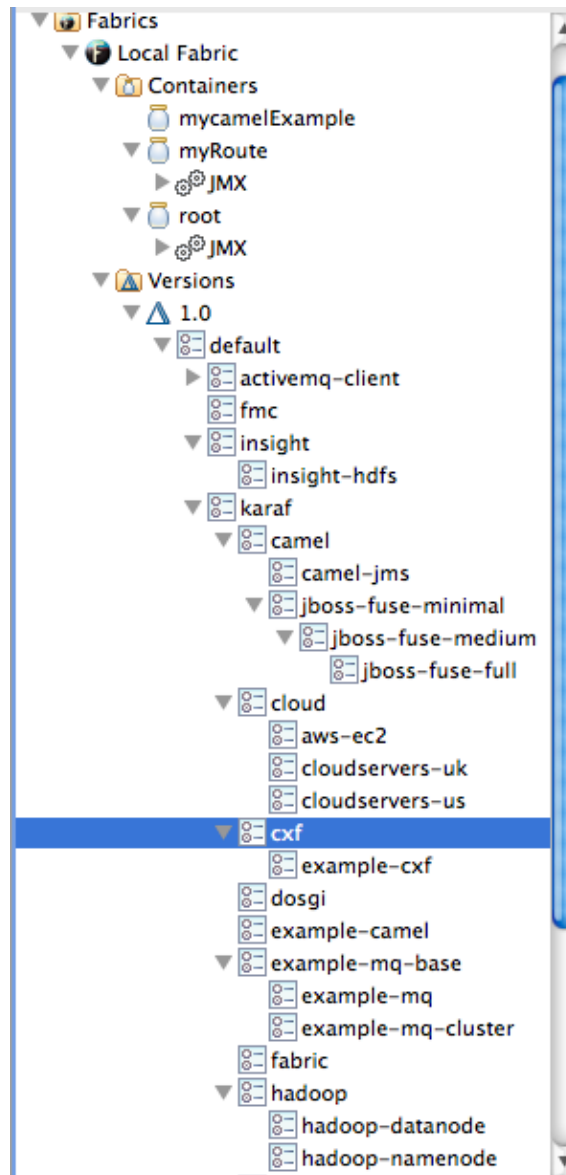
This procedure creates a brand new profile. If you want to create a new version of an existing profile to incrementally test changes to deployed containers, see [Chapter 25, Working with Versions](#)

Procedure

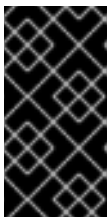
To create a new profile:

1. In **Fabric Navigator**, expand the fabric in which you want to create a new profile.
2. Expand the fabric's tree to the level of the profiles, as shown in [Figure 24.1](#).

Figure 24.1. Accessing the profiles



3. Select the profile on which you want to base the new profile, and right-click it to open the context menu.



IMPORTANT

Because the base profile is the parent of the new profile, the new profile inherits the base profile's configurations, properties, features, repositories and bundles as well as those of the base profile's ancestors. However, a child's configurations override an ancestor's when their configuration files have the same name.

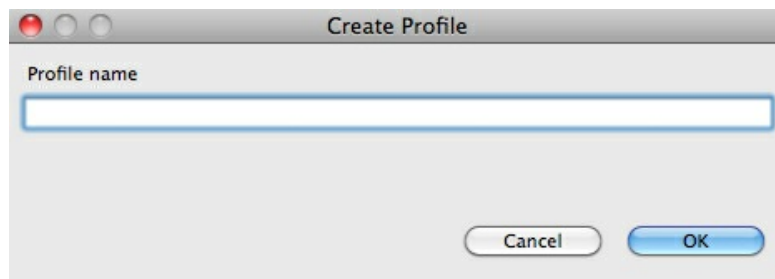


NOTE

Neither configuration files nor properties are visible within the developer tools, but you can view them in the Red Hat JBoss Fabric Management Console. For details, see the Red Hat [JBoss Fabric Management documentation](#).

4. Select **Create a new Profile**.

Figure 24.2. Create Profile dialog



5. In **Profile name**, enter a name for the new profile.

Use a name that is descriptive of the new profile's function.

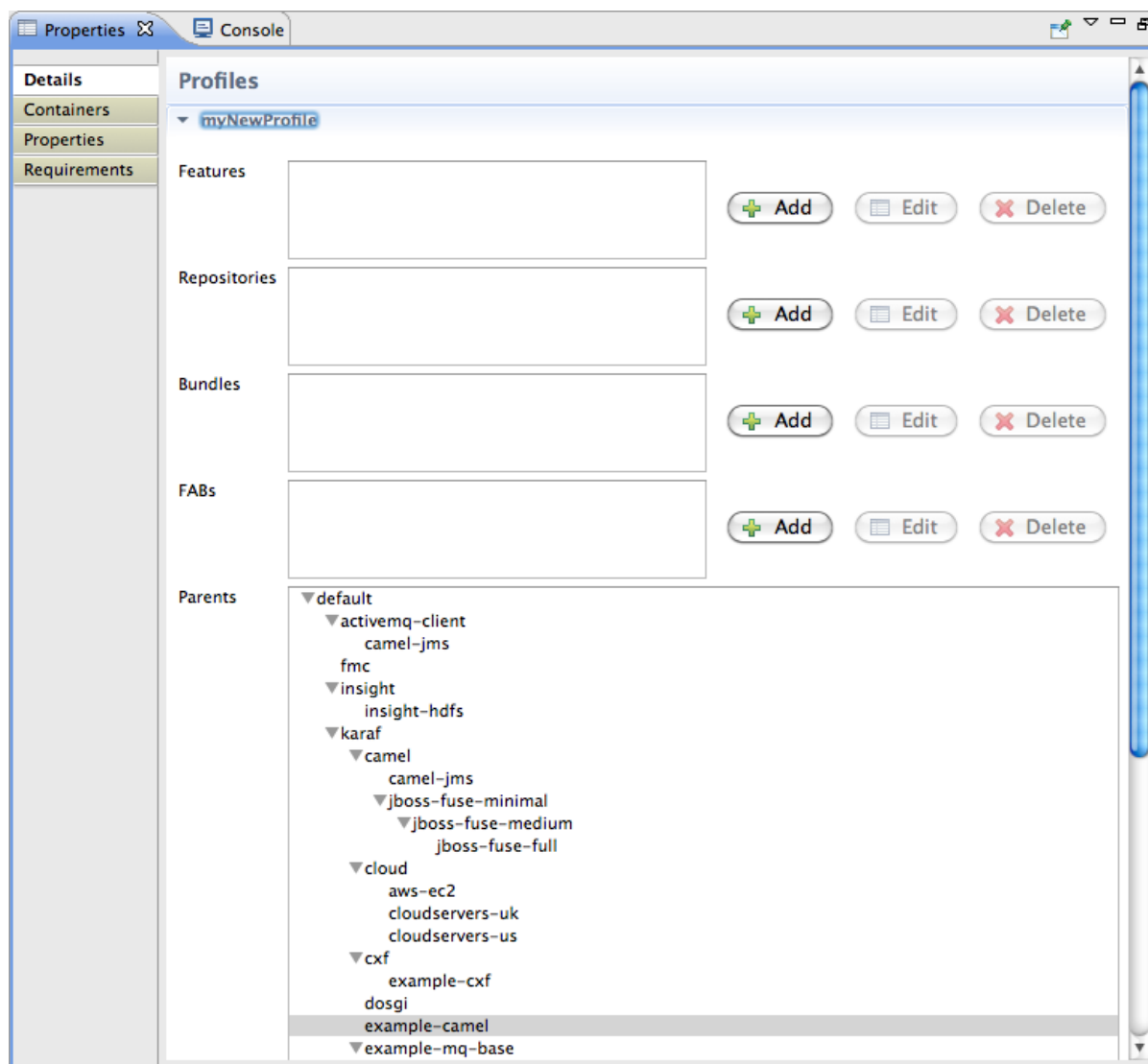
6. Click **OK** to create and save the new profile.

The new profile is added to the list of profiles in **Fabric Navigator**, under its parent profile.

7. In **Fabric Navigator**, select the new profile to populate the **Properties** editor with the new profile's properties information and profile details.

8. In the **Properties** editor, click the **Details** tab.

Figure 24.3. Profiles page



On the **Profiles** page, you can add, delete, or edit features, the repositories from which features are downloaded, and bundles. You can also add or remove parent profiles. Profiles can have multiple parents, and they inherit the qualities of each

Features are files that aggregate references to interdependent or related bundles that together define a feature. Typically, you'd use bundles only if the feature they define is not already included in a features file. For more information on bundles and features, see [Fab Deployment Model](#) in the Red Hat JBoss Fuse Deploying into the Container guide.

Child profiles inherit everything (configurations, properties, features, repositories, and bundles) from their immediate parents and from all of their parents' ancestors. When the child's and any ancestor's configuration file have the same name, the child's configuration file replaces the ancestor's file. Otherwise, configurations are cumulative.

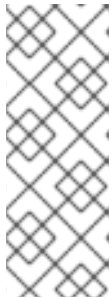


NOTE

The **Profiles** page displays only the features, repositories, and bundles added by the child profile, not those contributed by the profile's parents or other ancestors. The **Parents** pane shows (highlighted) the profile's immediate parents only.

9. Make the changes you want to the new profile to configure it.

Changes are automatically saved as you make them.



NOTE

In **Fuse Integration** perspective, you can easily add a Fuse project to the new profile by dragging it from **Project Explorer** and dropping it onto the new profile in **Fabric Navigator**. When you do, the fabric tooling builds the project, runs all tests, installs the project in the new profile, and uploads the profile into the fabric's internal Maven repository. Because profiles are stored in a fabric's internal repository, they cannot be shared across other fabrics.



IMPORTANT

If you drop your project on a fabric container, rather than on a profile, the next time that container gets provisioned, your project is wiped out.

You can now assign the new profile to a container.

10. In **Fabric Navigator**, select the container to which you want to assign the new profile.

The fabric tooling populates the **Properties** editor with the container's properties information and profile details.

11. Open the container's **Details** tab, and then select the new profile in the **Profiles** list.

Repeat [Step 10](#) and [Step 11](#) for each container to which you want to assign the new profile.

Related topics

[Section 25.1, “Creating a new version of a profile”](#)

[Section 8.3, “Deploying a Project to a Fabric Container”](#)

24.2. DELETING A PROFILE

Overview

When a profile is inactive—no longer assigned to any container in the fabric—you can safely delete it.

Procedure

To delete an inactive profile:

1. In **Fabric Navigator**, expand the fabric which contains the profile you want to delete.
2. Expand the fabric's tree to the level of versions, and then expand the version that contains the target profile.
3. Select the target profile, and right-click it to open the context menu.
4. Select **Delete Profile**.

CHAPTER 25. WORKING WITH VERSIONS

Versioning provides the means for safely rolling out upgrades of the profiles deployed and running in containers across a fabric. Using versioning, you can make incremental updates to existing profiles, and then test them out on small groups of containers. When you've completed and tested the final versions, you can confidently deploy them to their target containers in the fabric.

25.1. CREATING A NEW VERSION OF A PROFILE

Overview

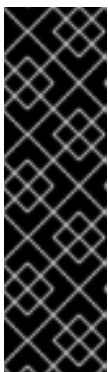
A version is simply the collection of all profiles associated with a fabric. When you create a new version, you are copying all of the profiles from the most recent version—the base version—into the new version. Once that's done, you can upgrade the profiles in the new version by modifying them.

Creating a new version

To create a new version:

1. In **Fabric Navigator**, select the fabric in which you want to create a new version.
2. Expand the fabric's tree to the level of Versions, shown in [Section 24.1, "Creating a new profile"](#).
3. Select a version, and then right-click it to open the context menu.
4. Select **Create Version**.
5. In the **Create Version** wizard, enter a new version or accept the default value.

The default version is always a minor increment of the highest-numbered version. So, if the highest-numbered version is 1.3, the new version will default to 1.4, and all of the profiles in version 1.3 will be copied into version 1.4.



IMPORTANT

Version names are important! The developer tools sort version names based on the numeric version string, according to major.minor numbering, to determine the version on which to base a new one. You can safely add a text description to a version name as long as you append it to the end of the generated default name like this: **1.3<.description>**. If you abandon the default naming convention and use a textual name instead (for example, **Patch051312**), the next version you create will be based, not on the last version (**Patch051312**), but on the highest-numbered version determined by dot syntax.

6. Click **OK**.

The developer tools copy all of the base version's profiles into the new version and display the new version in **Fabric Navigator**, directly under the base version.

Modifying profiles in the new version

To modify a profile in the new version:

1. In **Fabric Navigator**, expand the newly created version.
2. Select a profile that you want to modify.

The developer tools populate the **Properties** editor with the profile's details for you to edit.

3. In the **Properties** editor, click the **Details** tab to access the **Profiles** page.
4. Make the changes you want to the new profile to configure it.

Changes are automatically saved as you make them.



NOTE

In **Fuse Integration** perspective, you can easily add a Fuse project to the new profile by dragging it from **Project Explorer** and dropping it onto the new profile in **Fabric Navigator**. When you do, the developer tools build the project, run all tests, install the project in the new profile, and upload the profile into the fabric's internal Maven repository. Because profiles are stored in a fabric's internal repository, they cannot be shared across other fabrics.

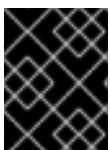
Related topics

[Section 24.1, "Creating a new profile"](#)

25.2. SETTING A CONTAINER'S VERSION

Overview

After you've created a new version, you can assign it to selected containers. This feature enables you to easily roll out upgraded profiles in a controlled and orderly way.



IMPORTANT

Setting a container's version immediately provisions the container with the profiles from the selected version.

Procedure

To set a container's version:

1. In **Fabric Navigator**, select the container whose profiles you want to change.
2. Right-click it to open the context menu, and then select **Select Version**.

The **Select Version** submenu lists all available versions.

3. Click the version you want to assign the selected container.

The container is immediately provisioned with the new version of any profile assigned to it.

CHAPTER 26. CREATING A FABRIC IN THE CLOUD

To create a fabric on a cloud, you need to

- Have established an account with a cloud provider
- Provide the developer tools the account details for connecting to the cloud
- Specify the details required for creating the fabric and the machine image on which it will run

26.1. ADDING CLOUD DETAILS

Overview

To connect to a cloud, the developer tools need the relevant access and authorization information. This information was provided to you or a system administrator by the cloud provider, when the account was set up.

Procedure

To add a cloud's access details:

1. In **Fabric Navigator**, right-click **Clouds** to open the context menu, and then select **Add Cloud details**.

Figure 26.1. Cloud Details wizard

The screenshot shows a 'Cloud Details' dialog box with the following fields and their states:

- Name:** Text input field with a red error icon (X) to its left.
- Provider name:** Dropdown menu.
- Api name:** Dropdown menu.
- Endpoint:** Text input field.
- Identity:** Text input field with a red error icon (X) to its left.
- Credential:** Text input field with a red error icon (X) to its left.
- Owner:** Text input field.

At the bottom of the dialog are two buttons: 'Cancel' and 'OK'.



NOTE

An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

2. In **Name**, enter a name for the cloud.



IMPORTANT

The next three fields provide two methods for specifying the provider's information. You select either a provider's name, or you select both an API and its endpoint.

3. In **Provider name**, select the name of the cloud provider from the drop-down list.

If you select a provider name, the developer tools autofill **Api name** and **Endpoint** with the selected provider's information, so you can skip to [Step 6](#).

4. In **Api name**, select from the drop-down list the name of the API you want to use.

This list displays only APIs that have been installed on your local machine or environment.

5. In **Endpoint**, enter the uri of the web service that exposes the API you selected in [Step 4](#).

If you have an on-premises, private cloud installed, you need to get this uri from the cloud administrator.

6. In **Identity**, enter the account identifier supplied by the cloud provider.

For example, for Amazon Elastic Compute Cloud (EC2) accounts, you'd enter the Access Key ID.

7. In **Credential**, enter the account password supplied by the cloud provider.

For example, for Amazon Elastic Compute Cloud (EC2) accounts, you'd enter the Secret Access Key.

8. In **Owner**, for EC2 only, enter the id assigned to a custom image.

This option is EC2-specific and applies only to custom images. To query or use a custom image, you must enter the owner id assigned to it by EC2; otherwise, the image will not appear in the list of instances.

If you entered valid access details, the developer tools connect to the cloud whose access details you specified and load the provider's images. Once that's done, the name you entered in [Step 2](#) appears as a node under **Clouds** in **Fabric Navigator**.

Related topics

[Section 23.3, "Creating a new container on a cloud"](#)

26.2. SPECIFYING FABRIC DETAILS

After you've added the access information that the developer tools need to connect to a cloud, you can create a fabric on the cloud. This procedure assumes that you have already added the cloud details.

Overview

Creating a fabric on a cloud involves creating on the cloud a container that contains a fabric registry and specifying the machine image on which you want to create and run it.

Procedure


To create a fabric on a cloud:

1. In **Fabric Navigator**, right-click **Fabrics** to open the context menu, and then select **Create Fabric in the cloud**.
2. On the **Create Fabric in the cloud** wizard's **Choose the Cloud** page, select the cloud from the list, and then click **Next** to open the wizard's **Fabric Details** page.

Figure 26.2. Fabric Details page



NOTE

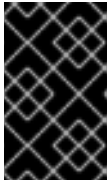
An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

You may have to wait a few seconds while the developer tools retrieve the location, hardware, and image IDs from your cloud provider to populate the wizard's corresponding fields.

3. In **Fabric Name**, accept the default name, or enter a different name for the new fabric.
4. In **Container name**, accept the default name, or enter a different name for the new container that will host the fabric registry.
5. In **Group**, accept the default name, or enter a different name for the group in which to start up the new fabric registry container.

6. In **User**, accept the default name, or enter a different user name to use for logging into the new fabric registry container.
7. In **Location ID**, select from the drop-down list the ID of the region in which the machines you want to use are located.
8. In **Hardware ID**, select from the drop-down list the ID of the hardware archetype you want to use.

Hardware archetypes specify the amount of memory and compute power available for running applications on machine images.



IMPORTANT

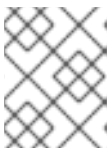
The next three fields provide two methods for specifying the software configuration (operating system, application server, and applications) you want to use. Either select an OS Family (and optionally a version), or select an Image ID.

9. In **OS Family**, select from the drop-down list the family name of the operating system you want to use.

If you select an **OS Family**, do not also select an **Image ID**.

10. In **OS Version**, optionally select from the drop-down list the version for the operating system you selected.
11. If you selected an **OS Family**, skip this step. Otherwise, in **Image ID**, select from the drop-down list the ID of the image template to use for the software configuration.
12. In **Maven proxy URI**, accept the default URI, or enter the URI to the Maven proxy repository you want to use.
13. Click **Finish**.

The developer tools create the container, then provision it with all of the bundles, features, and profiles required to create a fabric registry. This process can take up to ten minutes. When it's done, the container specified in [Step 4](#) appears in **Fabric Navigator** as a node under **Fabrics**.



NOTE

It's a good idea to access your cloud account and verify that the new fabric registry container is up and running.

Related topics

[Section 26.1, "Adding cloud details"](#)

[Section 23.3, "Creating a new container on a cloud"](#)

APPENDIX A. WIZARD FIELD REFERENCES

NAME

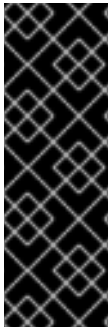
New Camel XML File — Creates a new XML file for editing routes with the route editor

PROPERTIES

[Table A.1](#) describes the properties you can specify.

Table A.1. New Camel XML File properties

Name	Default	Description
RouteContainer		Specifies the folder into which the new file will be placed.
File Name	camelContext.xml	Specifies the name of the new routing context file.
Framework	Spring	Specifies whether the context file is intended to be used in a Spring container or in an OSGi container that supports Blueprint.



IMPORTANT

The Spring framework and the OSGi Blueprint framework require that all Apache Camel files be placed in specific locations under the project's **META-INF** folder:

- Spring—*projectName* /src/main/resources/META-INF/spring/
- OSGi Blueprint—*projectName* /src/main/resources/META-INF/blueprint/

RELATED TOPICS

[Chapter 3, Creating a New Camel XML file](#)

NAME

New Fuse Project — Creates a new Fuse project

PROPERTIES

[Table A.2](#) describes the properties you can specify.

Table A.2. New Fuse IDE project: select project location properties

Name	Default	Description
Use default Workspace location	Enabled	Specifies whether to create the new project in the default workspace.
Location		Specifies the full path to an alternate workspace when the default workspace location option is disabled.
Add project(s) to working set	Disabled	Specifies whether to add the new project to one or more established working sets.
Working set		Specifies the working sets in which to add the new project when the working set option is enabled.

RELATED TOPICS

[Chapter 2, Creating a New Fuse Project](#)

NAME

New Fuse Project — Creates a new Fuse project

PROPERTIES

[Table A.3](#) describes the properties you can specify.

Table A.3. New Fuse IDE project: select project archetype and specify its details properties

Name	Default	Description
Archetype		Specifies the Maven archetype to use to create the new project.
Group ID		Specifies the group ID Maven uses when generating the new project. This ID is mapped to the POM file's groupId element.
Artifact ID		Specifies the artifact ID Maven uses when generating the new project. This ID is mapped to the POM file's artifactId element.

Name	Default	Description
Version	<i>1.0.0-SNAPSHOT</i>	Specifies the version name Maven uses when generating the new project. You can accept the default version name, or enter a new one. This value is mapped to the POM file's version element.
Package	<i><groupId>.<artifactID></i>	Specifies the package name for the generated artifacts. You can accept the default package name, or enter a new one.

RELATED TOPICS

[Chapter 2, Creating a New Fuse Project](#)

NAME

New Fuse Message — Creates a new message to use in route tracing

PROPERTIES

[Table A.4](#) describes the properties you can specify.

Table A.4. Fuse Message File properties

Name	Default	Description
RouteContainer	The selected folder	Specifies the folder into which the new message file will be placed.
File Name	message.xml	Specifies the name of the new message file.

RELATED TOPICS

[Section 14.1, “Creating test messages for route tracing”](#)

NAME

New Camel JUnit Test Case — Configures the JUnit artifacts created to test a Apache Camel route.

PROPERTIES

[Table A.5](#) describes the properties you can specify.

Table A.5. New Camel JUnit properties

Name	Default	Description
Source Folder	<i>project/src/test/java</i>	Specifies the folder into which the JUnit code is placed.
Package	Name of the package when the project was created	Specifies the package name for the JUnit code.
Camel XML file under test	Selected context file, or blank if other than a context file is selected	Specifies the Camel XML file containing the routes to be tested.
Name	classname based on the selected context file (for example, CamelContextXMLTest), or empty if none selected	Specifies the name of the JUnit class.
Method stubs		Specifies the JUnit method stubs to include in the generated test class.
Generate comments		Specifies whether to generate comments in the new test class.

RELATED TOPICS

[Chapter 5, Creating a new Apache Camel JUnit test case](#)

NAME

Test Endpoints — Lists the endpoints that can be tested by a JUnit test case.

PROPERTIES

[Table A.6](#) describes the properties you can specify.

Table A.6. Test Endpoint properties

Name	Default	Description
Available endpoints	all	Specifies the endpoints to test in the Camel JUnit test case.

RELATED TOPICS

[Chapter 5, Creating a new Apache Camel JUnit test case](#)

NAME

Define a New Server — Defines a new server instance.

PROPERTIES

[Table A.7](#) describes the properties you can specify.

Table A.7. New Server properties

Name	Default	Description
Server type		Specifies the type of server to define.
Server's host name		Specifies the name of the machine running the server.
Server name		Specifies the name of the server.
Server runtime environment	Selected server's runtime	Specifies the runtime environment of the server.

RELATED TOPICS

[Section 19.1, "Adding a Server"](#)

NAME

New Server Configuration — Configures access details for the server.

PROPERTIES

[Table A.8](#) describes the properties you can specify.

Table A.8. Server Configuration details properties

Name	Default	Description
Host Name	For localhost, 0.0.0.0	Specifies the address of the machine running the server.
Port Name	8101	Specifies the port used to contact the server.
User Name		[Optional] Specifies the name of a user authorized to access the server remotely.

Name	Default	Description
Password		[Optional] Specifies the password the authorized user must enter to access the server remotely.

RELATED TOPICS

[Section 19.1, “Adding a Server”](#)

NAME

Add and Remove — Specifies the resources available to a server.

PROPERTIES

[Table A.9](#) describes the properties you can specify.

Table A.9. Add and Remove properties

Name	Description
Available	Specifies the resources that are available to the server.
Configured	Specifies the resources that server is configured to use.

RELATED TOPICS

[Section 19.1, “Adding a Server”](#)

NAME

Fabric Details — Specifies information needed to connect to a fabric.

PROPERTIES

[Table A.10](#) describes the properties you can specify.

Table A.10. Fabric Details properties

Name	Default	Description
Name	Local Fabric	Specifies the name of the fabric to connect to.

Name	Default	Description
URLs	localhost:2181	Specifies the url, in the form of <i>hostname:port</i> , of the specified fabric.
User name		Specifies the name of a user that has login privileges.
Password		Specifies the password of the specified user.
Zookeeper Password		Specifies the password for accessing the fabric registry.

RELATED TOPICS

[Section 22.1, “Adding fabric details”](#)

[Section 22.4, “Editing a fabric's details”](#)

NAME

Create Child Container — Specifies information needed to create a container on the local host.

PROPERTIES

[Table A.11](#) describes the properties you can specify.

Table A.11. Create Child Container properties

Name	Default	Description
Container name	Container#	Specifies the name of the new container.
Version	The highest-numbered version according to major.minor numbering	Specifies the version of the profiles to use.
Profiles		Specifies the profile(s) to install in the new container.

RELATED TOPICS

[Section 23.1, “Creating a new child container”](#)

NAME

Create Container via SSH — Specifies information needed to create a container on a remote host.

PROPERTIES

[Table A.12](#) describes the properties you can specify.

Table A.12. Create Container via SSH properties

Name	Default	Description
Container name	Container#	Specifies the name of the new container.
Version	The highest-numbered version according to major.minor numbering	Specifies the version of the profiles to use.
Host		Specifies the name or the IP address of the remote host.
User name		Specifies the name of a user authorized to login to the remote host.
Password		Specifies the password of the specified user.
Path	<i>/usr/local/fusesource/container</i>	Specifies the path of the new container's location on the remote host.
Port	22	Specifies the port number to use for establishing an ssh connection on the remote host.
SSH retries	1	Specifies the maximum number of retries to attempt at establishing an ssh connection on the remote host.
Retry delay	1	Specifies the delay, in milliseconds, between retry attempts.
Profiles		Specifies the profile(s) to install in the new container.

RELATED TOPICS

[Section 23.2, "Creating a container on a remote host"](#)

NAME

Create Container in a Cloud — Specifies information needed to create a container in a cloud.

PROPERTIES

[Table A.13](#) describes the properties you can specify.

Table A.13. Create Container in a Cloud:New Containers details properties

Name	Default	Description
Container name	Container#	Specifies the name of the new container.
Version	The highest-numbered version according to major.minor numbering	Specifies the version of the profiles to use.
Group		Specifies the name of the group in which to start up the new container.
User		Specifies the name of a user authorized to login to the new container.
Location ID		Selects the region in which the machines you want to use are located.
Hardware ID		Selects the hardware archetype to use.
OS Family		Selects the general operating system to use.
OS Version		Selects the version of the operating system you selected.
Image ID		Selects a specific image template to use for the software configuration.
Profiles		Specifies the profile(s) to install in the new container.

RELATED TOPICS

[Section 23.3, “Creating a new container on a cloud”](#)

NAME

Create Version — Specifies the identifier for the new version of the fabric's profiles.

PROPERTIES

[Table A.14](#) describes the properties you can specify.

Table A.14. Create Version properties

Name	Default	Description
Create a new version for rolling upgrades to Profiles	<i>The highest-numbered version according to major.minor numbering</i>	Specifies the identifier for the new version of fabric's profiles.

RELATED TOPICS

[Section 25.1, “Creating a new version of a profile”](#)

NAME

Cloud Details — Specifies information needed to connect to a cloud.

PROPERTIES

[Table A.15](#) describes the properties you can specify.

Table A.15. Cloud Details properties

Name	Default	Description
Name		Specifies the name of the cloud to connect to.
Provider name		Selects the cloud provider.
Api name		Selects the API to use on the specified cloud.
Endpoint		Specifies the cloud endpoint uri.
Identity		Specifies the account identifier (supplied by the cloud provider).
Credential		Specifies the account password (supplied by the cloud provider).

Name	Default	Description
Owner		Specifies the name of the account owner.

RELATED TOPICS

[Section 26.1, “Adding cloud details”](#)

NAME

Create Fabric in the Cloud — Specifies information needed to create a fabric in a cloud.

PROPERTIES

[Table A.16](#) describes the properties you can specify.

Table A.16. Create Fabric in a Cloud: Fabric details properties

Name	Default	Description
Fabric name	Cloud Fabric	Specifies the name of the new fabric.
Container name	Registry	Specifies the name of the fabric registry container.
Group	fabric	Specifies the name of the group in which to start up the fabric registry container.
User	admin	Specifies the name of a user authorized to login to the fabric registry container.
Location ID		Selects the region in which the machines you want to use are located.
Hardware ID		Selects the hardware archetype to use.
OS Family		Selects the general operating system to use.
OS Version		Selects the version of the operating system you selected.

Name	Default	Description
Image ID		Selects a specific image template to use for the software configuration.
Maven proxy URI	<i>http://repo.fusesource.com/nexus/content/groups/ea</i>	Specifies the Maven proxy repository to use.

RELATED TOPICS

[Section 26.2, “Specifying fabric details”](#)

APPENDIX B. DEVELOPER TOOLS PREFERENCES

NAME

Deploy Folders — Configures the hot deployment folders for the containers into which routes can be deployed.

PROPERTIES

[Table B.1](#) describes the properties you can specify.

Table B.1. Deployment folder properties

Name	Description
Name	Specifies the name the developer tools use to identify the container.
Deploy Folder	Specifies the full path the container's hot deploy directory.
Description	Specifies a brief description of the container.



IMPORTANT

The developer tools do **not** validate that the provided path is a valid deployment directory/folder for a container.

RELATED TOPICS

[Chapter 8, *Deploying Projects to a Container*](#)

[Section 8.1.1, "Configuring a Container's Deploy Folder"](#)

NAME

Editor — Sets the default value for some route editor properties.

PROPERTIES

[Table B.2](#) describes the properties you can specify.

Table B.2. Editor Properties

Name	Default	Description
Language	Simple	Specifies the default expression language used by the route editor.

Name	Default	Description
ID values	Enabled	Determines whether ID values are used to label EIP nodes on the route editor's canvas.
Layout direction	Right	Specifies the direction in which routes are laid out on the route editor's canvas.
Diagram grid	Enabled	Determines whether a grid overly is displayed in the background on the route editor's canvas.

RELATED TOPICS

[Section 4.10, "Configuring the route editor"](#)

NAME

Fuse JMX Navigator — Configures the monitoring of local JMX servers and how information is displayed in the **Fuse Integration** perspective's **Properties** view

PROPERTIES

[Table B.3](#) describes the properties you can specify.

Table B.3. Monitor Properties

Name	Default	Description
Period to update	1000	Specifies the update frequency, in milliseconds, for the local JMX server's statistics.
Timeline	Disabled	Determines whether the graphs displayed on the Timeline tab have legends.
Threads	Enabled	Determines whether stack traces are considered when displaying information in the Threads tab.
Memory	Enabled	Determines whether stack traces are considered when displaying information in the Memory tab.

RELATED TOPICS

[Chapter 11, *The Fuse JMX Navigator*](#)

[Section 11.1, “Viewing Processes in JMX”](#)

NAME

Tools — Configures the monitoring of local JMX servers and how information is displayed in the **Fuse Integration** perspective's **Properties** panel

PROPERTIES

[Table B.4](#) describes the properties you can specify.

Table B.4. Tools Properties

Name	Default	Description
Auto detect...	3000	Specifies the number of milliseconds the Fuse JMX Navigator scans JVMs on the local host to see if they are still running.
Memory	50	Specifies the maximum number of classes included in the heap histogram displayed on the Memory tab.

RELATED TOPICS

[Chapter 11, *The Fuse JMX Navigator*](#)

[Section 11.1, “Viewing Processes in JMX”](#)