



Red Hat JBoss Fuse 6.2

Tooling User Guide

Developing and Debugging Applications

Red Hat JBoss Fuse 6.2 Tooling User Guide

Developing and Debugging Applications

JBoss A-MQ Docs Team

Content Services

fuse-docs-support@redhat.com

Legal Notice

Copyright © 2015 Red Hat.

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 the Red Hat JBoss Fuse Tooling, which provides developer tools designed to increase your productivity when designing, developing, testing, and debugging your integration applications.

Table of Contents

PART I. INTRODUCING THE FUSE TOOLING USER INTERFACE	7
CHAPTER 1. JBOSS PERSPECTIVE	8
OVERVIEW	8
CANVAS	9
PALETTE	9
PROJECT EXPLORER	9
OUTLINE VIEW	10
PROPERTIES EDITOR	10
CHAPTER 2. DEBUG PERSPECTIVE	11
OVERVIEW	11
RELATED TOPICS	12
CHAPTER 3. FUSE INTEGRATION PERSPECTIVE	13
OVERVIEW	13
JMX NAVIGATOR	14
MESSAGES VIEW	14
CHAPTER 4. FABRIC8 PERSPECTIVE	15
FABRIC EXPLORER	15
DIAGRAM VIEW	15
LOG VIEW	16
MESSAGES VIEW	16
PROPERTIES VIEW	16
JMX VIEWER	16
PART II. DEVELOPING APPLICATIONS	17
CHAPTER 5. CREATING A NEW FUSE PROJECT	18
OVERVIEW	18
PROCEDURE	18
RESOLVING MAVEN DEPENDENCY ERRORS	21
RELATED TOPICS	21
CHAPTER 6. CREATING A NEW CAMEL XML FILE	22
OVERVIEW	22
PROCEDURE	22
RELATED TOPICS	23
CHAPTER 7. EDITING A ROUTING CONTEXT IN THE ROUTE EDITOR	24
7.1. ADDING ROUTES TO THE ROUTING CONTEXT	24
7.2. ADDING PATTERNS TO A ROUTE	25
7.3. CONNECTING PATTERNS TO MAKE A ROUTE	27
7.4. CONFIGURING A PATTERN	27
7.5. REARRANGING PATTERNS ON THE CANVAS	28
7.6. REMOVING PATTERNS FROM A ROUTE	29
7.7. DISCONNECTING TWO PATTERNS	29
7.8. DELETING A ROUTE	30
7.9. ADDING BEANS AND CONFIGURATION	30
7.10. CONFIGURING THE ROUTE EDITOR	31
CHAPTER 8. CREATING A NEW APACHE CAMEL JUNIT TEST CASE	33
OVERVIEW	33

PROCEDURE	33
RELATED TOPICS	35
CHAPTER 9. THE SOURCE VIEW	37
CHAPTER 10. RUNNING ROUTES INSIDE RED HAT JBOSS FUSE TOOLING	38
10.1. RUNNING ROUTES AS A LOCAL CAMEL CONTEXT	38
10.2. RUNNING ROUTES USING MAVEN	38
10.3. WORKING WITH RUNTIME PROFILES	39
CHAPTER 11. USING THE JBOSS FUSE SAP TOOL SUITE	45
11.1. INSTALLING THE JBOSS FUSE SAP TOOL SUITE	45
11.2. CREATE AND TEST AN SAP DESTINATION CONNECTION	46
11.3. CREATE AND TEST AN SAP SERVER CONNECTION	47
11.4. EXPORT SAP CONNECTION CONFIGURATION TO A FILE	49
11.5. CREATE A NEW SAP ENDPOINT	49
CHAPTER 12. GETTING STARTED WITH DATA TRANSFORMATION	52
12.1. FUSE TRANSFORMATION TOOLING	52
12.2. YOUR FIRST DATA TRANSFORMATION	52
PART III. DEBUGGING ROUTING CONTEXTS	64
CHAPTER 13. SETTING BREAKPOINTS	65
OVERVIEW	65
SETTING UNCONDITIONAL BREAKPOINTS	65
SETTING CONDITIONAL BREAKPOINTS	66
DISABLING BREAKPOINTS	66
DELETING BREAKPOINTS	67
RELATED TOPICS	67
CHAPTER 14. STARTING THE CAMEL DEBUGGER	68
OVERVIEW	68
PROCEDURE	68
WATCHING MESSAGE EXCHANGES PROGRESS THROUGH THE ROUTING CONTEXT	69
RELATED TOPICS	70
CHAPTER 15. STOPPING THE CAMEL DEBUGGER	71
OVERVIEW	71
TERMINATING THE CAMEL DEBUGGER BEFORE THE SESSION ENDS	71
RELATED TOPICS	71
CHAPTER 16. CHANGING VARIABLE VALUES	72
OVERVIEW	72
PROCEDURE	72
RELATED TOPICS	73
CHAPTER 17. ADDING VARIABLES TO THE WATCH LIST	74
OVERVIEW	74
PROCEDURE	74
RELATED TOPICS	75
CHAPTER 18. DISABLING BREAKPOINTS IN A RUNNING CONTEXT	76
OVERVIEW	76
DISABLING AND ENABLING BREAKPOINTS IN BREAKPOINTS VIEW	76
DISABLING AND ENABLING BREAKPOINTS ON THE RUNNING CAMELCONTEXT: CANVAS	77

DELETING AND RESETTING BREAKPOINTS ON THE RUNNING CAMELCONTEXT: CANVAS	78
RELATED TOPICS	79
PART IV. MONITORING AND TESTING APPLICATIONS	80
CHAPTER 19. JMX NAVIGATOR	81
19.1. VIEWING PROCESSES IN JMX	82
19.2. ADDING A JMX SERVER	82
CHAPTER 20. VIEWING A COMPONENT'S PROPERTIES	84
OVERVIEW	84
PROCEDURE	84
CHAPTER 21. BROWSING MESSAGES	85
OVERVIEW	85
PROCEDURE	85
RELATED TOPICS	85
CHAPTER 22. TRACING ROUTES	86
22.1. CREATING TEST MESSAGES FOR ROUTE TRACING	86
22.2. ACTIVATING ROUTE TRACING	87
22.3. TRACING MESSAGES THROUGH A ROUTE	87
22.4. DEACTIVATING ROUTE TRACING	88
CHAPTER 23. MANAGING JMS DESTINATIONS	90
23.1. ADDING A JMS DESTINATION	90
23.2. DELETING A JMS DESTINATION	90
CHAPTER 24. MANAGING ROUTING ENDPOINTS	92
24.1. ADDING A ROUTING ENDPOINT	92
24.2. DELETING A ROUTING ENDPOINT	92
CHAPTER 25. EDITING RUNNING ROUTES	94
OVERVIEW	94
PROCEDURE	94
RELATED TOPICS	94
CHAPTER 26. MANAGING ROUTING CONTEXTS	96
26.1. SUSPENDING A ROUTING CONTEXT	96
26.2. RESUMING A ROUTING CONTEXT	96
PART V. PUBLISHING APPLICATIONS TO A CONTAINER	98
CHAPTER 27. MANAGING SERVERS	99
27.1. ADDING A SERVER	99
27.2. STARTING A SERVER	102
27.3. CONNECTING TO A RUNNING SERVER	104
27.4. DISCONNECTING FROM A SERVER	106
27.5. STOPPING A SERVER	107
27.6. DELETING A SERVER	107
CHAPTER 28. PUBLISHING FUSE PROJECTS TO A SERVER	109
OVERVIEW	109
PUBLISHING FUSE PROJECTS AUTOMATICALLY WHEN RESOURCES CHANGE	110
PUBLISHING FUSE PROJECTS MANUALLY	112
VERIFYING THE PROJECT WAS PUBLISHED TO THE SERVER	113
RELATED TOPICS	114

PART VI. WORKING WITH FABRICS	115
CHAPTER 29. SETTING UP A FABRIC ENVIRONMENT	116
OVERVIEW	116
OPENING THE FABRIC PERSPECTIVE	116
RELATED TOPICS	116
CHAPTER 30. SPECIFYING AND CONNECTING TO A FABRIC	117
30.1. ADDING FABRIC DETAILS	117
30.2. CONNECTING TO A FABRIC	118
30.3. DISCONNECTING FROM A FABRIC	119
30.4. EDITING A FABRIC'S DETAILS	119
30.5. DELETING A FABRIC'S DETAILS	120
CHAPTER 31. WORKING WITH FABRIC CONTAINERS	121
31.1. CREATING A NEW CHILD CONTAINER	121
31.2. CREATING A CONTAINER ON A REMOTE HOST	122
31.3. CREATING A NEW CONTAINER ON A CLOUD	124
31.4. STARTING A CONTAINER	128
31.5. CHANGING A CONTAINER'S PROFILES	129
31.6. STOPPING A CONTAINER	129
31.7. DELETING A CONTAINER	130
CHAPTER 32. WORKING WITH FABRIC PROFILES	132
32.1. CREATING A NEW PROFILE	132
32.2. DELETING A PROFILE	135
CHAPTER 33. WORKING WITH VERSIONS	137
33.1. CREATING A NEW VERSION OF A PROFILE	137
33.2. SETTING A CONTAINER'S VERSION	138
CHAPTER 34. CREATING A FABRIC IN THE CLOUD	139
34.1. ADDING CLOUD DETAILS	139
34.2. SPECIFYING FABRIC DETAILS	140
APPENDIX A. WIZARD FIELD REFERENCES	143
NAME	143
PROPERTIES	143
RELATED TOPICS	143
NAME	143
PROPERTIES	143
RELATED TOPICS	144
NAME	144
PROPERTIES	144
RELATED TOPICS	145
NAME	145
PROPERTIES	145
RELATED TOPICS	145
NAME	146
PROPERTIES	146
RELATED TOPICS	146
NAME	146
PROPERTIES	146
RELATED TOPICS	147
NAME	147

PROPERTIES	147
RELATED TOPICS	147
NAME	147
PROPERTIES	147
RELATED TOPICS	148
NAME	148
PROPERTIES	148
RELATED TOPICS	148
NAME	148
PROPERTIES	149
RELATED TOPICS	149
NAME	149
PROPERTIES	149
RELATED TOPICS	150
NAME	150
PROPERTIES	150
RELATED TOPICS	151
NAME	151
PROPERTIES	151
RELATED TOPICS	152
NAME	152
PROPERTIES	152
RELATED TOPICS	152
NAME	152
PROPERTIES	153
RELATED TOPICS	153
NAME	153
PROPERTIES	153
RELATED TOPICS	154
APPENDIX B. TOOLING PREFERENCES	155
NAME	155
PROPERTIES	155
RELATED TOPICS	155
NAME	155
PROPERTIES	155
RELATED TOPICS	156

PART I. INTRODUCING THE FUSE TOOLING USER INTERFACE

Red Hat JBoss Fuse Tooling is an Eclipse-based IDE for creating and testing small- and large-scale integration applications. Its key features are a graphical routing editor, a full-featured Camel debugger for debugging locally running routing contexts, an integrated testing platform that uses JMX to communicate with the containers in a small-scale test environment, and tooling for using Fabric8 to test applications in large-scale, highly-distributed test environments.

CHAPTER 1. JBOSS PERSPECTIVE

OVERVIEW

By default, Red Hat JBoss Developer Studio starts up in **JBoss** perspective. To access the design time tooling, you select **File** → **New** → **Fuse Project** on the menu bar.

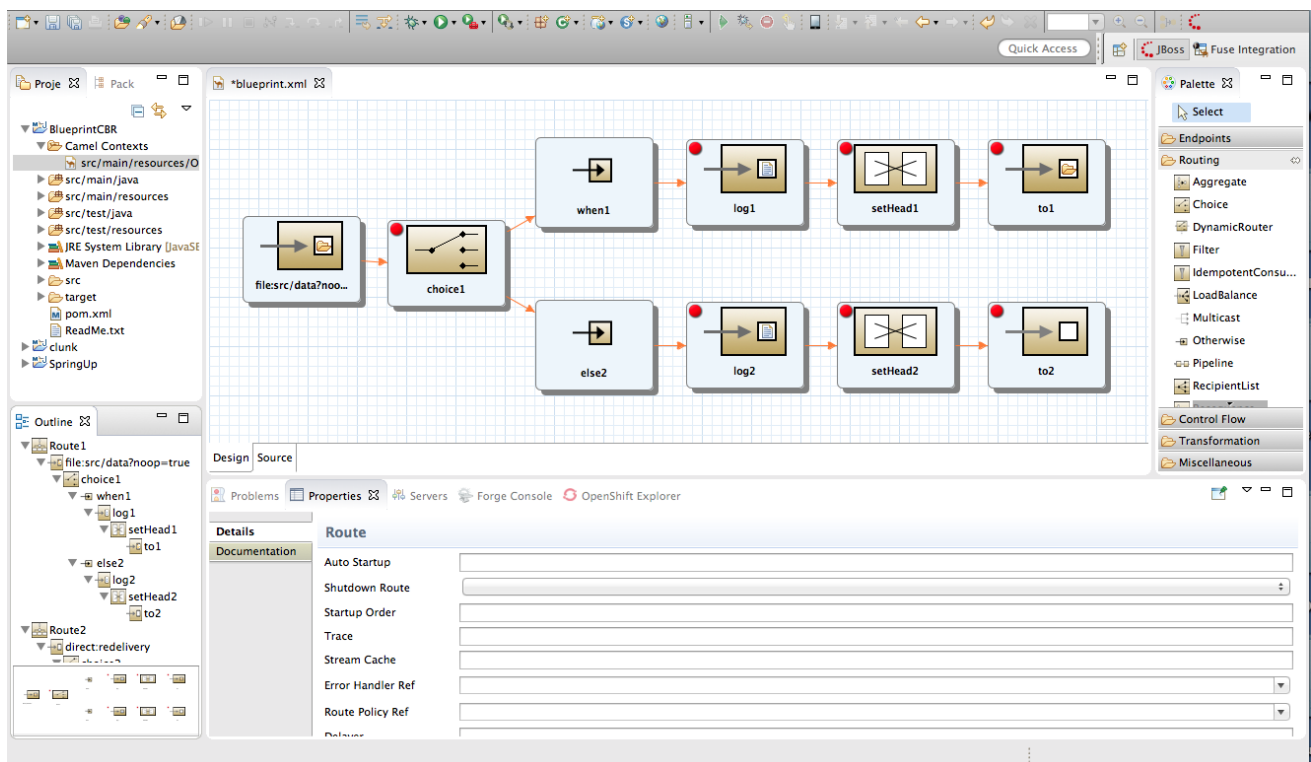


NOTE

Though you can also do all of your development in **Fuse Integration** perspective, **JBoss** perspective provides more space for the route editor to expand to accommodate complex routes.

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 **Project Explorer**, which displays the project's folders and files in an hierarchical tree format.



NOTE

Red Hat recommends you to close the **JMX Navigator** view and then drag **Outline** view from the upper right side of the workspace to the lower left side, beneath **Project 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:

- **Components**—endpoint patterns that start or end a route
- **Defined Endpoints**—endpoint nodes that have been defined in the routing context
- **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.

PROJECT EXPLORER

Project 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.


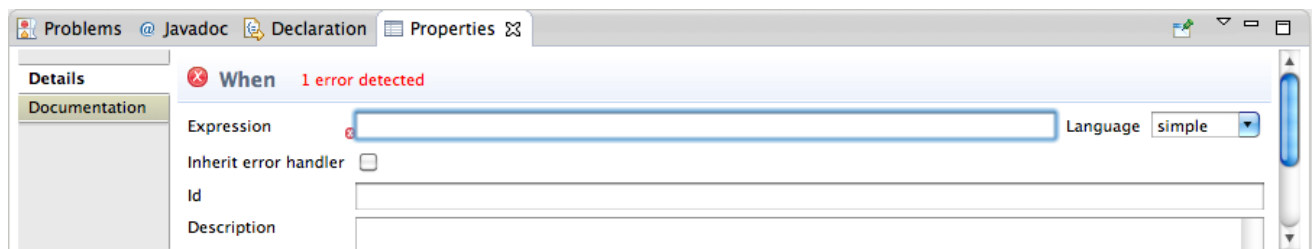
The **Properties** editor makes it clear which fields are required.  marks any required property that is not set, and the **Properties** editor displays an error message stating the number of required properties that still must be set, as shown in [Figure 1.2](#).

Figure 1.2. Properties editor's error reporting



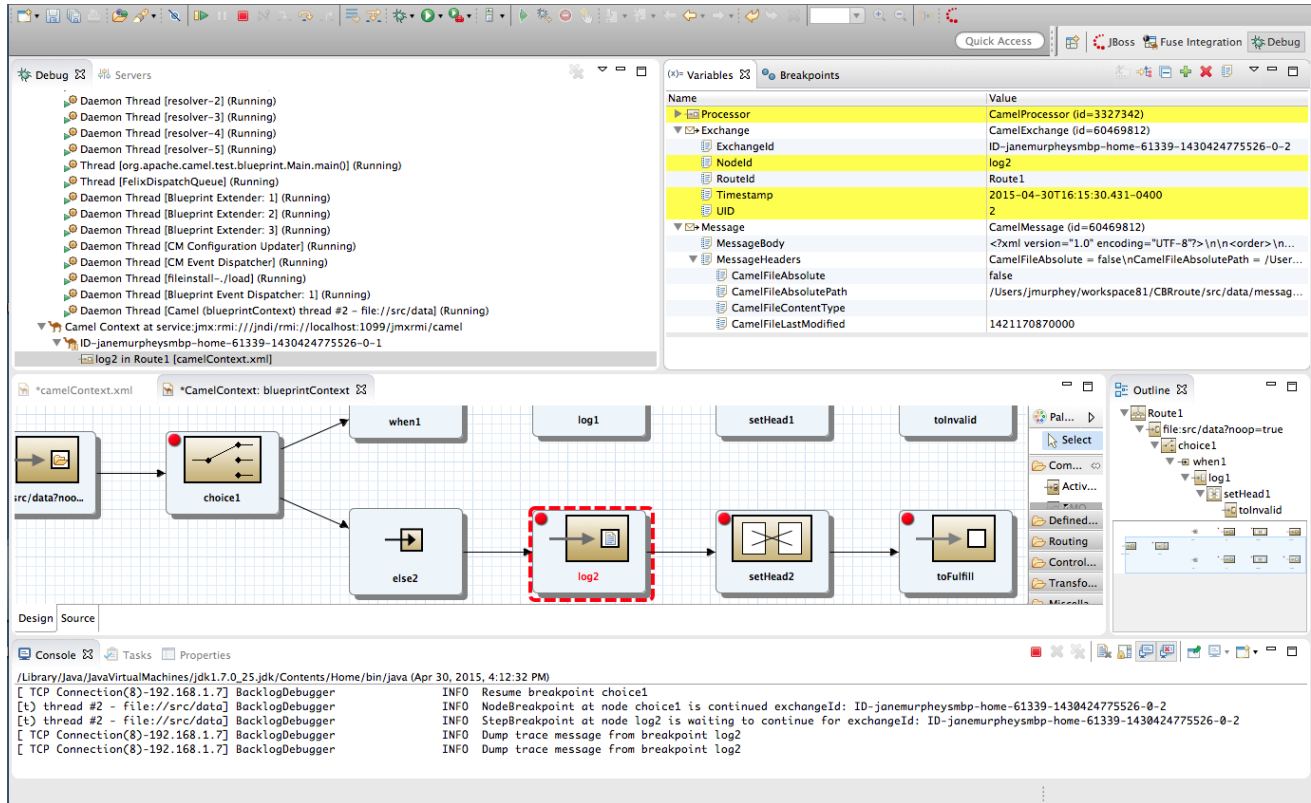
The **Properties** editor's **Documentation** tab displays information about how to configure the selected node.

CHAPTER 2. DEBUG PERSPECTIVE

OVERVIEW

The Debug perspective, shown in [Figure 2.1](#), provides access to all of the Camel debugger's functions. It contains all of the views used to monitor and debug a running routing context.

Figure 2.1. Debug perspective



- **Debug view**

For the running routing context, **Debug** view displays the debug stack.

You can switch between breakpoints within the same message flow, listed under the **Camel Context at service:jmx:rmi://jndi/rmi://localhost:1099/jmxrmi/camel** entry, to review and compare variable values in **Variables** view.

Messages flows are identified by their unique breadcrumb ID, and the breadcrumb ID of each subsequent message flow is incremented by 2. For example, in [Figure 2.1](#), “[Debug perspective](#)”, the breadcrumb ID for the first message flow is **ID-janemurpheysmbp-home-61339-11430424775526-0-1**, so the breadcrumbID for the second message flow would be **ID-janemurpheysmbp-home-61339-11430424775526-0-3**.

- **Variables view**



For each node in the routing context that has a breakpoint set, **Variables** view displays the value of the available variables when the breakpoint is hit. Each variable whose value changed since the preceding breakpoint is highlighted in yellow.

You can change the value of editable variables to check whether such changes produce the expected results and to test the robustness of your routing context.

You can also add variables to the watch list, so you can quickly and easily see whether their values change as expected at the expected point in the message flow.

- **Breakpoints** view

Displays a list of the breakpoints set in the routing context, and shows whether they are enabled or disabled. You can enable and disable individual breakpoints by checking (enabling) or unchecking (disabling) them. This enables you to temporarily focus on nodes in your routing context that are behaving problematically.

The  button skips over disabled breakpoints to jump to the next active breakpoint in the routing context. In contrast, the  button jumps to the next node of execution in the routing context, regardless of breakpoints.

- **Camel Context:** view

Displays the running **CamelContext : <CamelId>.xml** in graphical mode. For nodes set with breakpoints, it shows the type of breakpoint set and whether the breakpoint is enabled or disabled. When a breakpoint is hit, its corresponding node on the canvas is outlined in red.

To check a node's configuration, open Properties view and then select the node on the canvas in **Camel Context:** view.

- **Console** view

Displays the log output generated by the Camel debugger as it executes the routing context.

- **Properties** view

Displays the properties set for the node selected on the canvas in **CamelContext:** view.

RELATED TOPICS

[Part III, "Debugging Routing Contexts"](#)

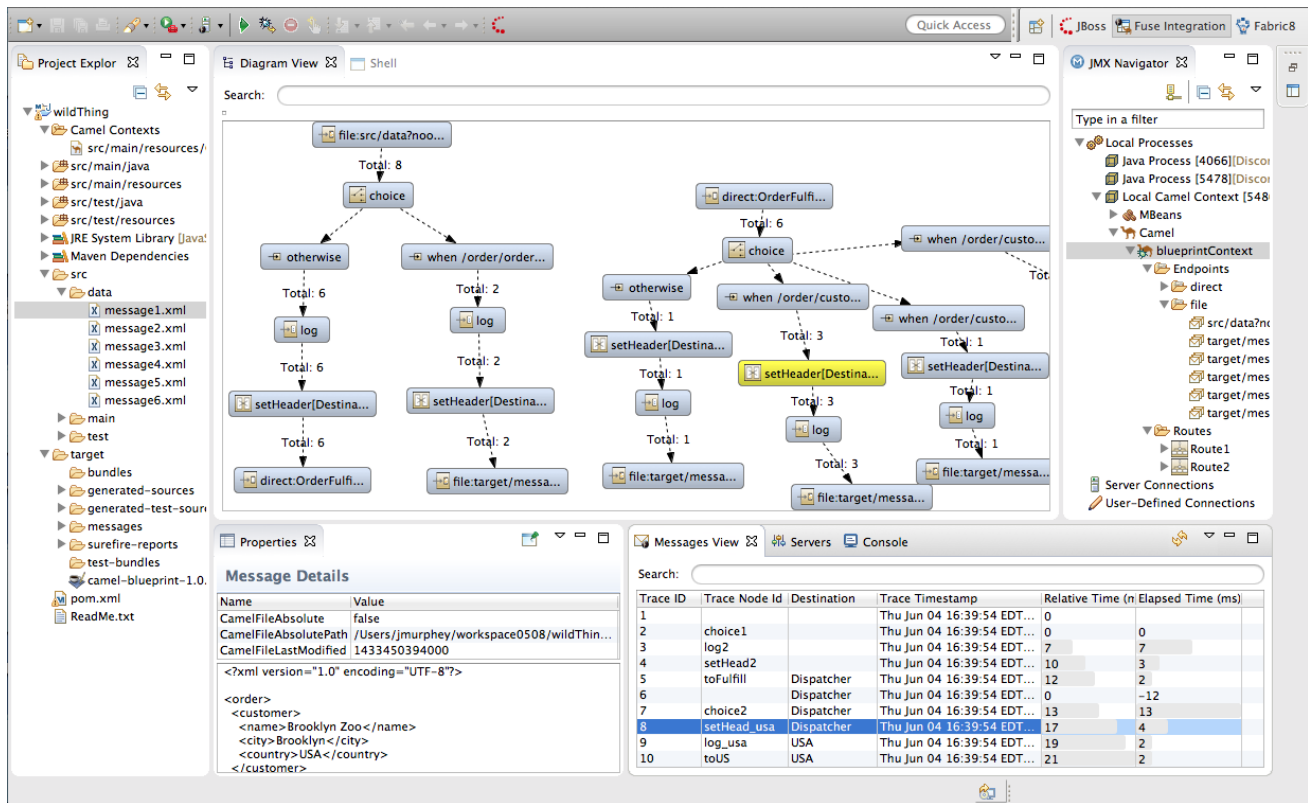
[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)

CHAPTER 3. FUSE INTEGRATION PERSPECTIVE

OVERVIEW

The **Fuse Integration** perspective, shown in [Figure 3.1](#), provides access to all of the tooling for monitoring and testing your integration application.

Figure 3.1. Fuse Integration perspective



The Fuse Integration perspective consists of eight main areas:

- **Project Explorer**—displays all of the projects known to the tooling. You can view all of the artifacts that make up each project. **Project Explorer** also displays all of the routing context `.xml` files for a project under its **Camel Contexts** node. This makes it easy for you to find and open the routing context file of any route included in a project.
- **JMX Navigator**—lists the JMX servers and the infrastructure they monitor. It allows you to browse JMX servers and the processes they are monitoring. It also identifies instances of Red Hat processes.
- **Diagram View**—displays a graphical tree representing the node selected in **JMX Navigator**. When you select a process, server, endpoint, or other node, **Diagram** view shows the selected node as the root and branches down to its children and grandchildren.

When you select a broker, **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, **Diagram** view displays all of the nodes in the route and shows the different paths that messages can take through it. It also displays timing metrics for each processing step in the route when route tracing is enabled.

- **Shell** view—Displays the command console of the connected container. You can control the container by entering commands in **Shell** view.
- **Messages View**—lists the messages that have passed through the selected JMS destination or through Apache Camel endpoints when route tracing is enabled.
- **Servers** view—displays a list of servers (Red Hat JBoss Fuse, Apache Karaf, or Apache ServiceMix) managed by the tooling. It displays their runtime status and provides controls for starting and stopping them and for debugging projects deployed on them.
- **Properties** view—displays the properties of the selected node.
- **Console** view— displays the console output for recently executed actions.

JMX NAVIGATOR

The **JMX Navigator** drives all monitoring and testing activities in **Fuse Integration** perspective. It determines which routes are displayed in **Diagram View**, the **Properties** viewer, and **Messages View**. It also provides menu commands for activating route tracing, adding and deleting JMS destinations, and starting and suspending routes. It is also the target for dragging and dropping messages onto a route.

By default, **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 browsing JMS destinations and for tracing messages that have passed through a route.

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

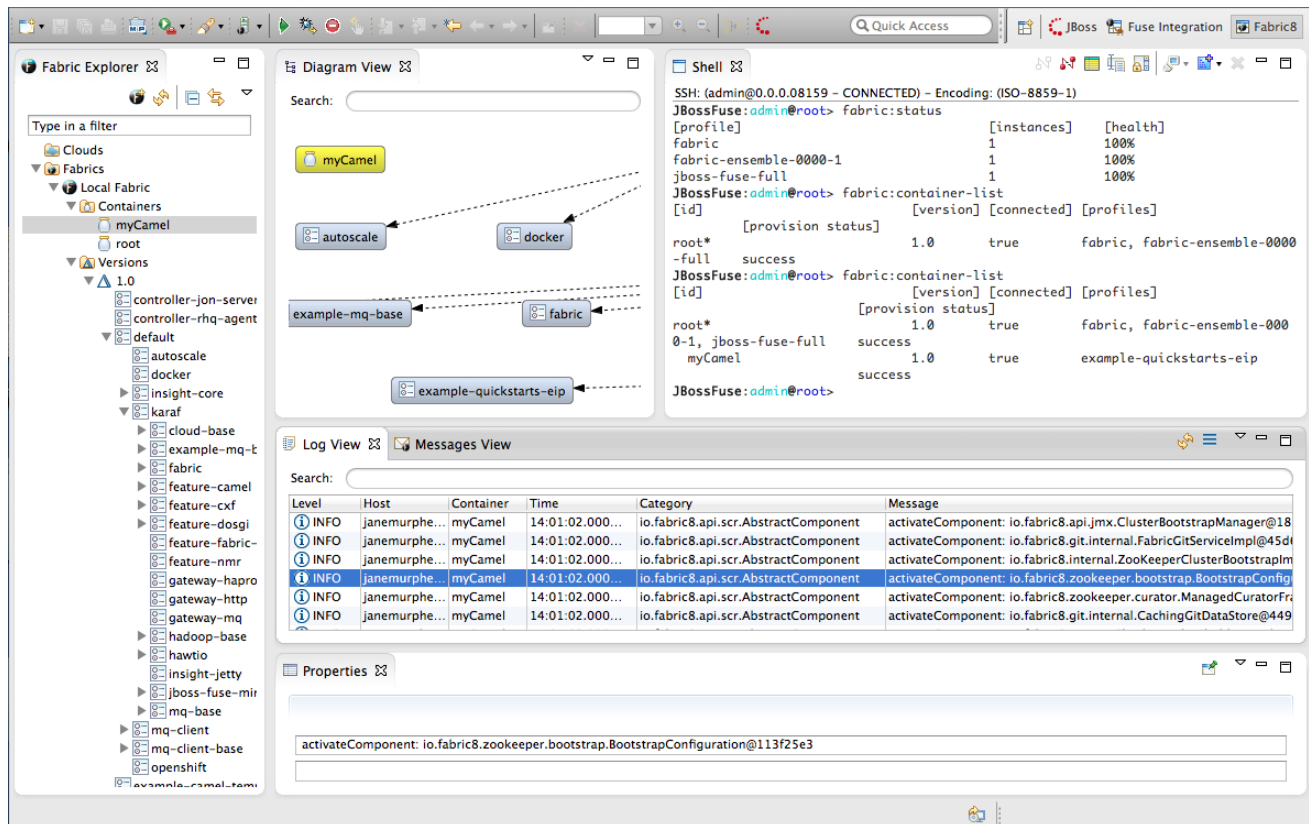
When route tracing is enabled, **Messages View** lists all of the messages that passed through the nodes in the route since tracing started. You can configure **Messages View** to display only the data in which you are interested and in your preferred sequence.

When a message trace in **Messages View** is selected, its details (message body and all message headers) are displayed in the **Properties** viewer. In **Diagram View**, the step in the route associated with the selected message trace is also highlighted.

CHAPTER 4. FABRIC8 PERSPECTIVE

The Fabric8 perspective, shown in [Figure 4.1](#), is where you access the fabric development and debugging tools.

Figure 4.1. Fabric8 perspective



The Fabric8 perspective consists of six main areas:

- **Fabric Explorer**—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 Explorer**.
- **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 Explorer**.

FABRIC EXPLORER

Fabric Explorer provides access to your fabrics. Once the tooling is connected to a fabric, **Fabric Explorer** 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 Explorer**. 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 Explorer**, **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.

PART II. DEVELOPING APPLICATIONS

We recommend that you develop your applications in [Chapter 1, *JBoss Perspective*](#) because it provides more room to expand the route editor's canvas when you are building complex routes. If it is not already open, click  on the right side of the JBoss Developer Studio tool bar.

CHAPTER 5. CREATING A NEW FUSE PROJECT

OVERVIEW

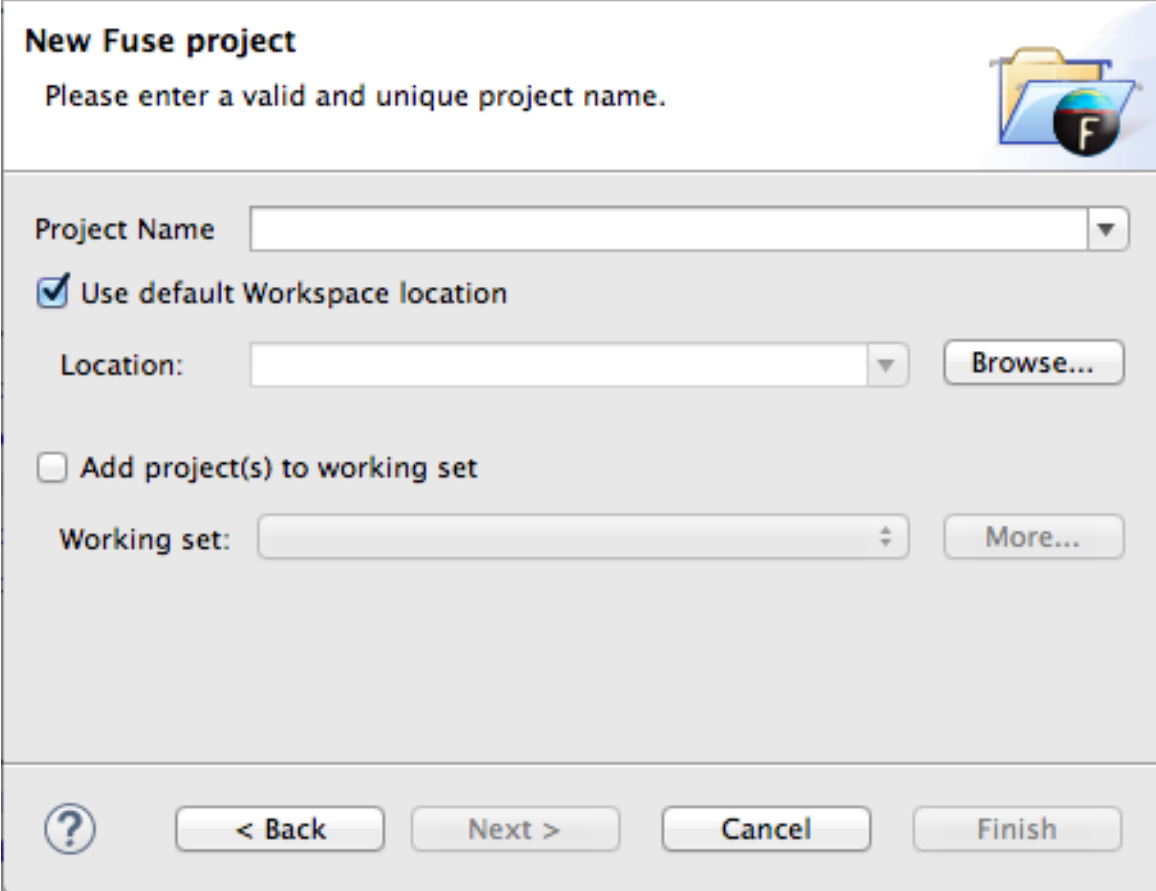
The tooling uses Maven archetypes to generate a project with all of the dependencies preconfigured. The archetypes also create the POM file needed to run and deploy your application, as well as sample code and data to get you started.

PROCEDURE

To create a Fuse project:

1. Select **File** → **New** → **Fuse Project** to open the **New Fuse Project** wizard shown in [Figure 5.1, “New Fuse Project Wizard's Project Location Page”](#).

Figure 5.1. New Fuse Project Wizard's Project Location Page



New Fuse project
Please enter a valid and unique project name.

Project Name

Use default Workspace location

Location:

Add project(s) to working set

Working set:

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

2. Enter the name of the Fuse project.
3. 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 clear **Use default workspace location** and specify a new location in the provided field.

Clicking  opens a file browser.


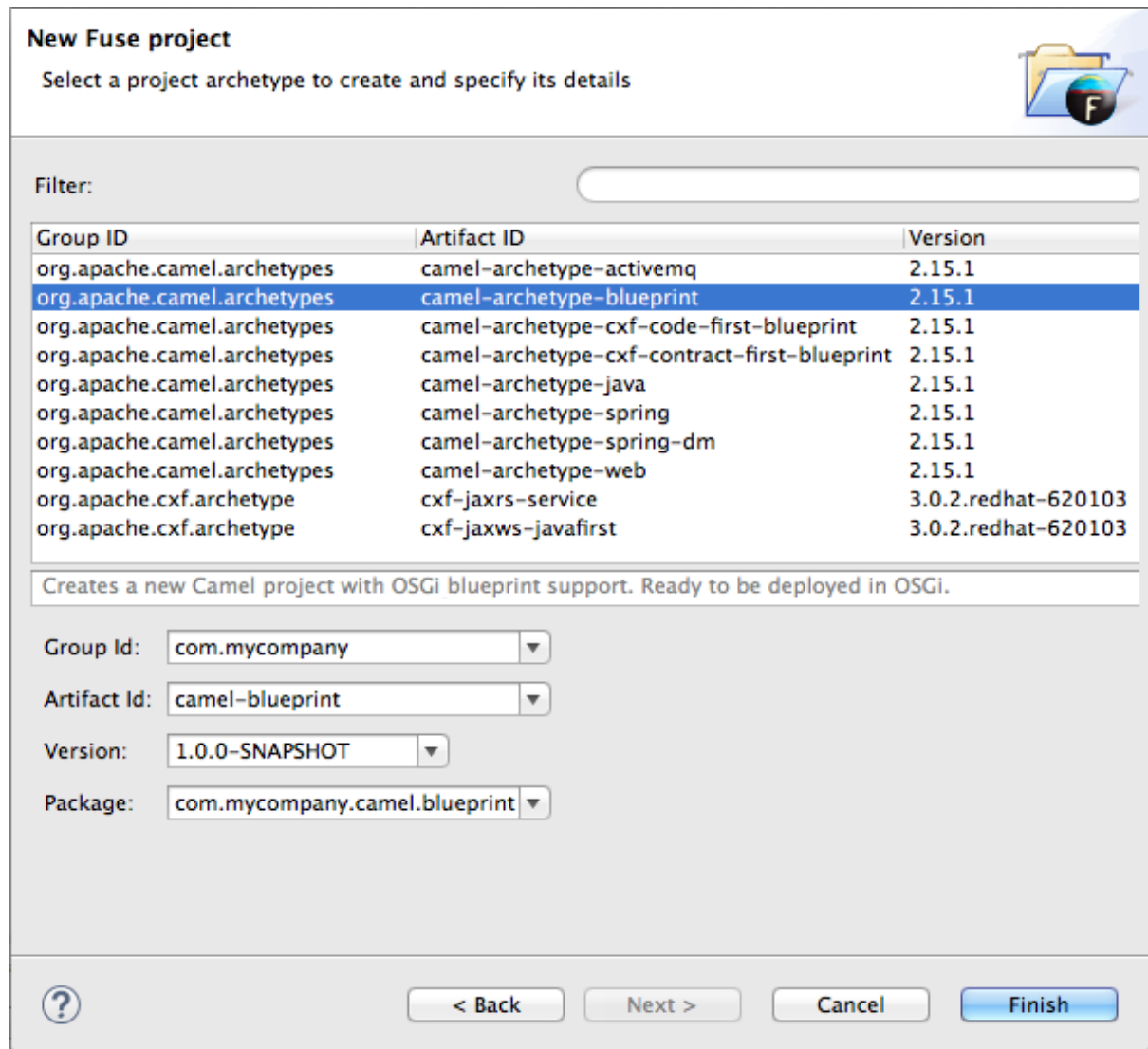
4. If you want to add the new project to an Eclipse working set, select **Add project(s) to working set** and enter the name of the working set.
5. Click  to open the **New Fuse Project** details page shown in [Figure 5.2](#).

Figure 5.2. New Fuse Project wizard's details page



6. 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 that is deployment-ready for OSGi.

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

Creates a new Apache Camel project with Apache CXF code-first example using OSGi Blueprint.

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

Creates a new Apache Camel project with Apache CXF contract-first example using OSGi Blueprint.

- **camel-archetype-java**

Creates a new Apache Camel project using Java DSL.

You cannot edit Java DSL in the route editor.

- **camel-archetype-spring**

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

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

Creates a new Apache Camel project with added support for Spring DSL that is deployment-ready for OSGi.

- **camel-archetype-web**

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

- **cxf-jaxrs-service**

Creates a simple CXF JAX-RS web application service using Spring configuration.

- **cxf-jaxws-javafirst**

Creates a project for developing a Web service starting from Java code.

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

The tooling uses 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** . .

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

The tooling uses 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**.

9. Enter a version for the project in the **Version** field, or accept the default.
10. If you want to change the package name generated for the artifacts, enter the new package name in the **Package** field.
11. 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 **Project 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](#)

[Red Hat JBoss Fuse Tooling Tutorials](#), To Create a New Route

CHAPTER 6. CREATING A NEW CAMEL XML FILE

OVERVIEW

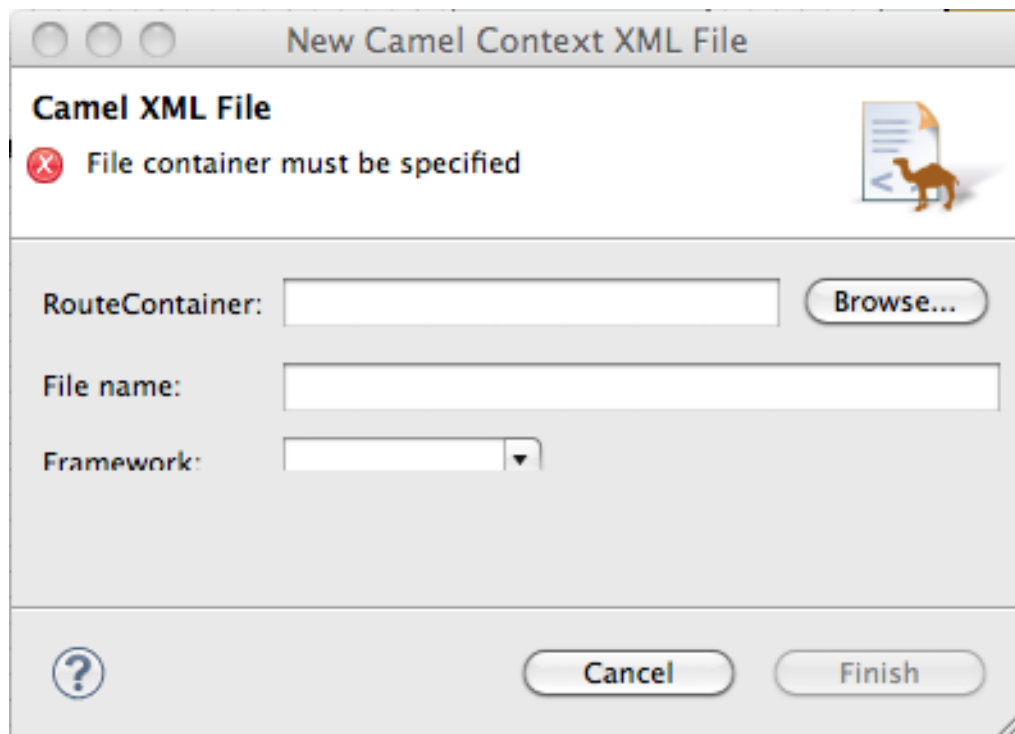
Apache Camel stores routes in an XML file that contains a `camelContext` element. The tooling includes a wizard that simplifies adding an Apache Camel context file to your project. It creates a new XML file that has all of the required namespaces preconfigured and a template `camelContext` element.

PROCEDURE


To add an Apache Camel context file to your project:

1. Select **File** → **New** → **Other...** → **Fuse Tooling** → **Camel XML File** from the main menu to open the **Camel XML File** wizard, as shown in [Figure 6.1, “Camel XML File wizard”](#).

Figure 6.1. 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** or **OSGI-INF** folder:

- Spring—*projectName* /src/main/resources/META-INF/spring/
- OSGi Blueprint—*projectName* /src/main/resources/OSGI-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 7. EDITING A ROUTING CONTEXT IN THE ROUTE EDITOR

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.

7.1. 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 tooling creates an example `camelContext.xml` file that is defined either as `camel-context.xml` (Spring) or `blueprint.xml` (Blueprint). 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 code generated from the patterns you drop onto the canvas.

The tooling provides 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.

Related topics

[Red Hat JBoss Fuse Tooling Tutorials](#), To Add a Content-Based Router

[Red Hat JBoss Fuse Tooling Tutorials](#), To Add Another Route to the CBR Routing Context

7.2. 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.

7.2.1. Drag and drop a pattern

Overview

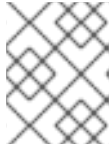
The **Palette** lists all of the available patterns, grouped by function. For example, the Loop pattern is part of the **Control Flow** group.

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 7.5, "Rearranging patterns on the canvas"
Section 7.4, "Configuring a pattern"
Section 7.6, "Removing patterns from a route"

7.2.2. 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 7.5, "Rearranging patterns on the canvas"
Section 7.4, "Configuring a pattern"
Section 7.6, "Removing patterns from a route"

7.3. 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.

7.4. CONFIGURING A PATTERN

Overview

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

The tooling's **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

7.5. 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 tooling provides two ways to rearrange the patterns on the canvas:

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

7.5.1. 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.

7.5.2. 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 **Preferences** → **Fuse Tooling** → **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 tooling to automatically align all nodes on the canvas:

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

7.6. 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 7.2, "Adding patterns to a route"](#)

7.7. 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 7.3, “Connecting patterns to make a route”](#)

7.8. 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.

7.9. ADDING BEANS AND CONFIGURATION

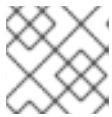
Overview

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

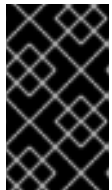
To add beans to your routing context file:

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. 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 9, The Source View](#)

[Apache Camel Development Guide](#), Part IV Programming EIP Components

7.10. CONFIGURING THE ROUTE EDITOR

Overview

Using the '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. In JBoss Developer Studio, click **Window** → **Preferences** to open the **Preferences** dialog.
2. Click **Fuse Tooling** → **Editor** to show the **Editor** preferences.
3. Select, from the drop-down list, the default language you want to use for expressions in EIP nodes.
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 in Routes Editor** to enable or disable displaying the grid overlay on the canvas.
7. Click **Apply** to apply the changes to the **Editor** preferences.
8. Click **Fuse Tooling** → **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 Tooling** → **Colors** and clicking **Restore Defaults**.

Related topics

Editor
Colors

CHAPTER 8. CREATING A NEW APACHE CAMEL JUNIT TEST CASE

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 **Project 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 8.1, “New Camel JUnit Test Case wizard”](#).

Figure 8.1. New Camel JUnit Test Case wizard

Camel JUnit Test Case

✖ Source folder name is empty.

Source folder:

Package:

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** → **Other...** → **Fuse Tooling** → **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.

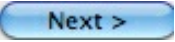
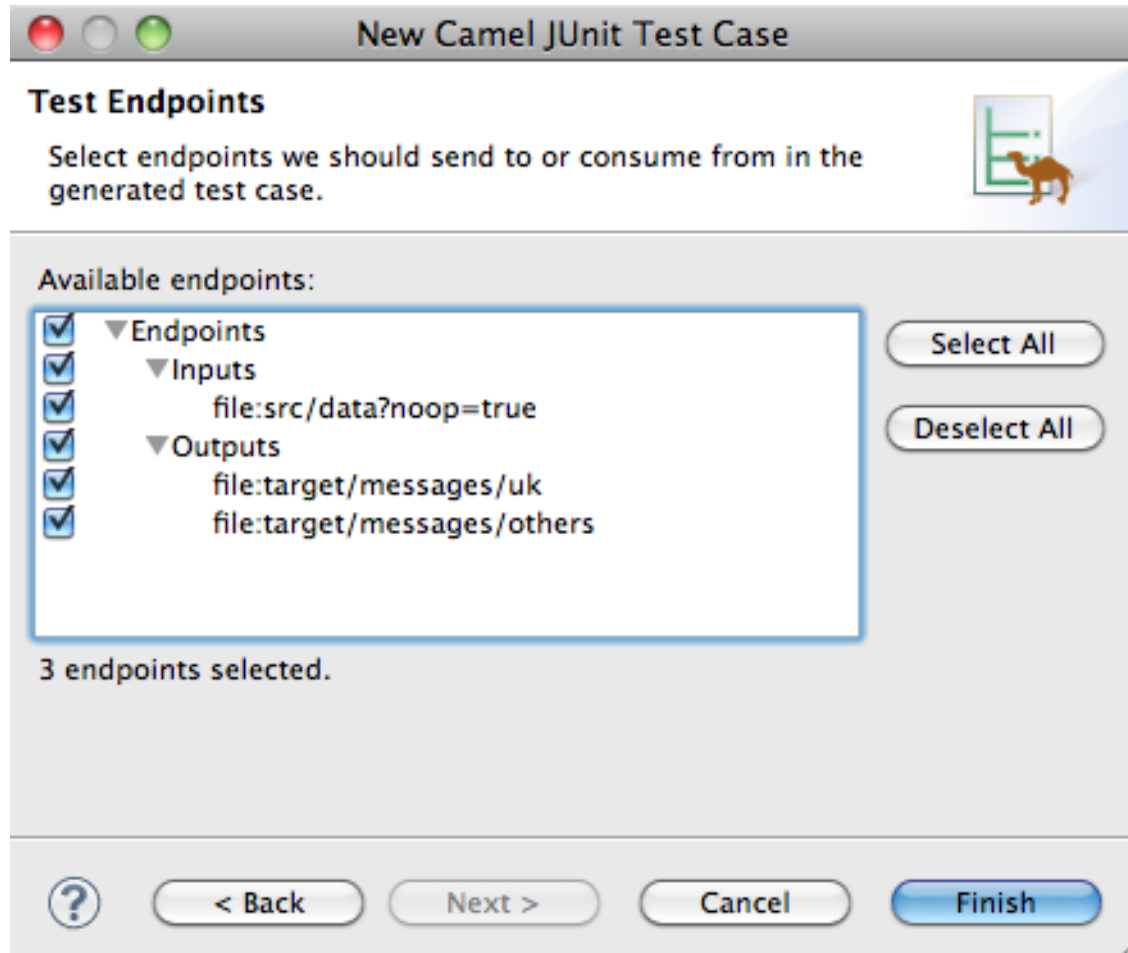

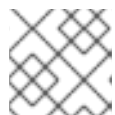
8. If you want to include the default generated comments in the generated code, check the **Generate comments** box.
9. Click  to open the **Test Endpoints** page. For example, [Figure 8.2, “New Camel JUnit Test Case page”](#) shows a route's input and output file endpoints selected.

Figure 8.2. 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 **Project 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](#)

[Red Hat JBoss Fuse Tooling Tutorials](#), To Test a Route with JUnit

CHAPTER 9. THE SOURCE VIEW

The tooling's **Source** tab opens the route editor in **Source** view (Figure 9.1) to display the contents of an Apache Camel route file as XML.

Figure 9.1. 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>

```

The **Source** view is useful for editing and adding any configuration, comments, or beans to the routing context file. The Content Assist functionality helps you when working with configuration files. In the **Source** view, press **Ctrl+Space** to see a list of possible values that can be inserted into your project.

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 10. RUNNING ROUTES INSIDE RED HAT JBOSS FUSE TOOLING

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

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

10.1. 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 **Project Explorer**'s context menu. When you run a route from the context menu, the tooling automatically creates 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 **Project 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 tooling 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 10.3.1, “Editing a Local Camel Context runtime profile”](#)

[Red Hat JBoss Fuse Tooling Tutorials, To Run a Route](#)

10.2. 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 **Project Explorer**, select the root of the project .
2. Right-click it to open the context menu, and then select **Run As** → **Maven build**.
 - o The first time you run the project using Maven the **Edit Configuration and launch** 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**.

- o 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 10.3.2, “Editing a Maven runtime profile”](#)

10.3. WORKING WITH RUNTIME PROFILES

Red Hat JBoss Fuse Tooling 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.

10.3.1. 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**, creates for the routing context file a default runtime profile, which should not require editing.

Accessing the Local Camel Context's runtime configuration editor


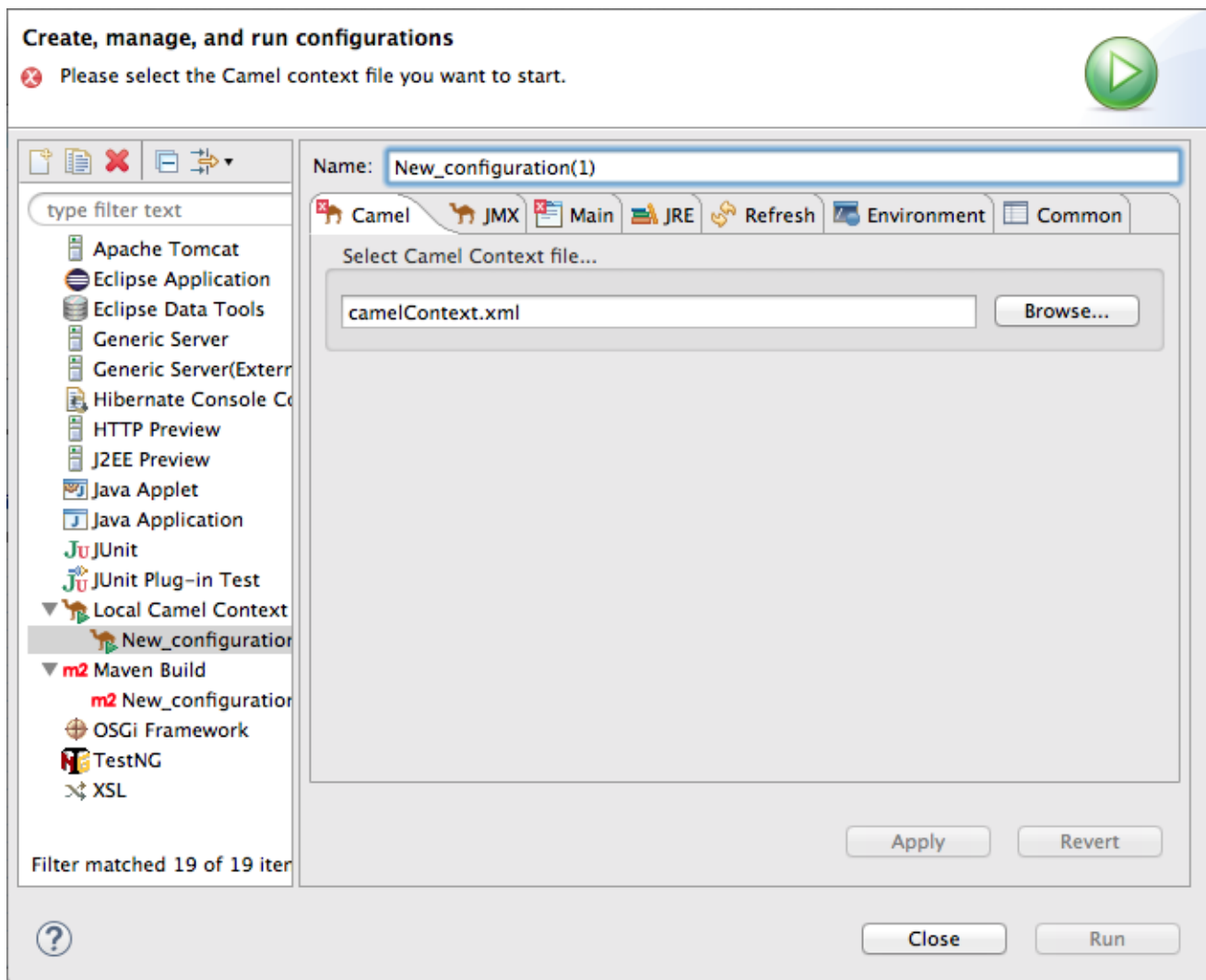
1. In **Project Explorer**, select the camelContext file for which you want to edit or create a custom runtime profile.
2. Right-click it to open the context menu, and then select **Run As...** → **Run Configurations** to open the **Run Configurations** dialog.
3. In the context selection pane, select **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 10.1. 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. In the **Standard Input and Output** pane, 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 10.1, “Running routes as a local Camel context”](#)

10.3.2. 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


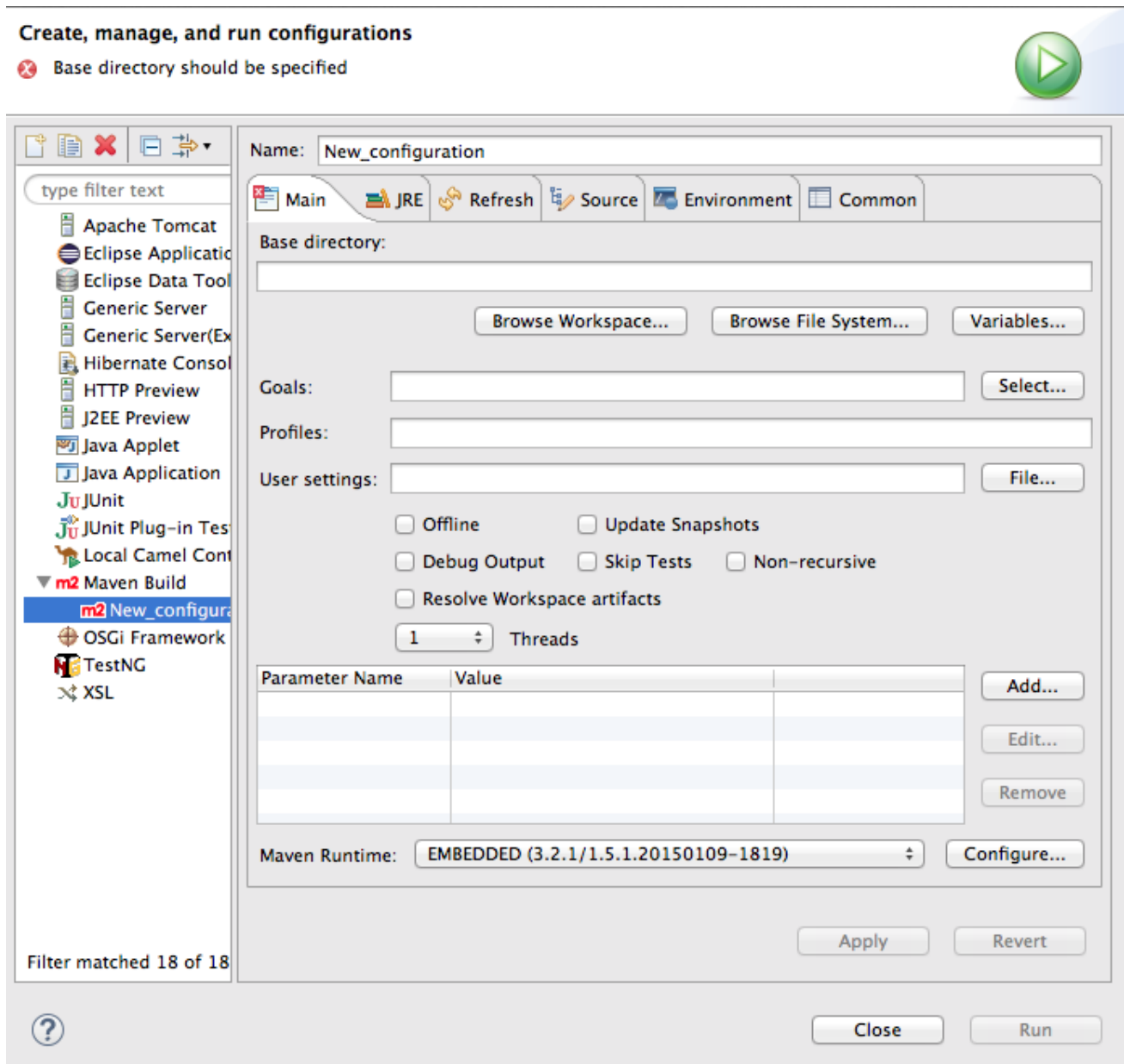
1. In **Project Explorer**, select the root of the project for which you want to edit or create a custom runtime profile.
2. Right-click it to open the context menu, and then select **Run As... → Run Configurations** to open the **Run Configurations** dialog.
3. In the context selection pane, select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

Figure 10.2. 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 Tooling for Eclipse uses 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 from the **Maven Runtime** drop-down menu 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 10.2, “Running routes using Maven”](#)

CHAPTER 11. USING THE JBOSS FUSE SAP TOOL SUITE

Abstract

The JBoss Fuse SAP Tool Suite makes it possible to integrate your Camel routes with a remote SAP Application Server. A variety of SAP components are provided to support Remote Function Calls (RFC) and the sending and receiving of Intermediate Documents (IDocs). Note that the SAP Tool Suite depends on the JCo and IDoc client libraries from SAP. *You must have an SAP Service Marketplace Account*, in order to install and use these libraries.

11.1. INSTALLING THE JBOSS FUSE SAP TOOL SUITE

Overview

The JBoss Fuse SAP Tool Suite provides a SAP Connections view for configuring SAP connections and a collection of SAP components for integrating your Camel routes with an SAP server. The suite is not installed by default, because it requires third-party JCo and IDoc client libraries, which are licensed separately by SAP.

Platform restrictions for SAP tooling

Because the SAP tool suite depends on the third-party JCo 3.0 and IDoc 3.0 libraries, it can only be installed on the platforms that these libraries support. For more details about the platform restrictions for SAP tooling, see [Red Hat JBoss Fuse Supported Configurations](#).

Prerequisites

Before you can install the JBoss Fuse SAP Tool Suite, you must download the requisites JCo and IDoc libraries from the following location:

- <http://service.sap.com/connectors>

Please note the following points:

- You will require an SAP Service Marketplace Account to download the JCo and IDoc client archives.
- Choose the appropriate JCo and IDoc libraries for your operating system.
- Only version 3.0.11 or greater of the JCo library is supported.
- Only version 3.0.10 or greater of the IDoc library is supported.
- For this installation procedure, you can leave the downloaded files in archive format. There is no need to extract the contents.

Procedure

To install the JBoss Fuse SAP Tool Suite into JBoss Developer Studio, perform the following steps:

1. In JBoss Developer Studio, select **File** → **Import** to open the **Import** wizard.

2. In the **Select** screen of the **Import** wizard, select **JBoss Fuse** → **Install JBoss Fuse SAP Tool Suite**, and click **Next**.
3. The **Install the JBoss Fuse SAP Tool Suite** screen opens, which displays the instructions for downloading the JCo and IDoc libraries from the SAP Service Marketplace. Click **Next**.
4. The **Select JCo3 and IDoc3 Archive to Import** screen opens. Next to the **JCo Archive File** field, use the **Browse** button to select the JCo archive that you downloaded from the SAP Service Marketplace. After selecting the JCo archive, the **Archive Version** and **Archive OS Platform** fields are automatically filled in, so that you can check whether the library you are installing has the correct version and OS platform.

Next to the IDoc3 Archive File field, use the **Browse** button to select the IDoc archive that you downloaded from the SAP Service Marketplace.

After selecting both archive files, click **Finish**.

5. A new **Install** wizard (for installing Eclipse plug-ins) opens automatically. This wizard displays the following to plug-ins to be installed:
 - JBoss Fuse SAP Tool Suite
 - SAP JCo3 and IDoc3 Libraries

Make sure that both of these plug-ins are selected. Click **Next**.



NOTE

The **SAP JCo3 and IDoc3 Libraries** plug-in is dynamically constructed from the selected JCo and IDoc libraries.

6. The **Install Details** screen allows you to review the plug-ins to be installed. Click **Next**.
7. The **Review Licenses** dialog opens. Select the **I accept** radiobutton option, and then click **Finish**.
8. If you encounter a **Security Warning** dialog (warning of unsigned content), click **OK** to ignore the warning and continue installing.
9. The **Restart Eclipse** dialog opens. Click **OK** to restart Eclipse.

11.2. CREATE AND TEST AN SAP DESTINATION CONNECTION

Overview

The JBoss Fuse SAP Tools Suite provides an SAP Connections view, which enables you to create and manage SAP destination connections and SAP server connections. This section describes how to create and test an SAP destination connection in the SAP Connections view.

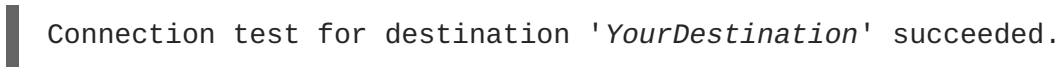
Procedure

To create and test an SAP destination connection, perform the following steps:

1. If you have not already added the **SAP Connections** view to your perspective, select **Window** → **Show View** → **Other** to open the **Show View** dialog. Select **SAP Connections** and click **OK**. The **SAP Connections** view opens on the left of your perspective (under **Project Explorer**).
2. In the **SAP Connections** view, right-click on **Destination Data Store** and select **New Destination**, to open the **Create Destination** dialog.
3. Enter a name for the destination in the **New Destination Name** field and click **OK**.
4. Select the newly created destination in the **Destination Data Store** tree, so that its properties show up in the **Properties** view.
5. In the **Properties** view, click on the **Basic** tab to configure the basic properties required to connect to an SAP destination. In this tab, fill in the following property fields to configure the connection:

- **SAP Application Server**
- **SAP System Number**
- **SAP Client**
- **Logon User**
- **Logon Password**
- **Logon Language**

These are standard SAP client connection properties. If you need more information about these settings, please consult the SAP documentation.

6. You are now ready to test the destination connection. In the **SAP Connection** view, right-click on the destination name and select **Test**.
7. The **Test Destination Connection** dialog opens. Click **Test**.
8. The dialog uses the current destination configuration settings to connect to the SAP server. If the test is successful, you will see the following message in the status area:
Connection test for destination '*YourDestination*' succeeded.
9. Click **Close**, to close the **Test Destination Connection** dialog.

11.3. CREATE AND TEST AN SAP SERVER CONNECTION

Overview

The JBoss Fuse SAP Tools Suite provides an SAP Connections view, which enables you to create and manage SAP destination connections and SAP server connections. This section describes how to create and test an SAP server connection in the SAP Connections view.

Procedure

To create and test an SAP server connection, perform the following steps:

1. If you have not already added the **SAP Connections** view to your perspective, select **Window** → **Show View** → **Other** to open the **Show View** dialog. Select **SAP Connections** and click **OK**. The **SAP Connections** view opens on the left of your perspective (under **Project Explorer**).
2. In the **SAP Connections** view, right-click on **Server Data Store** and select **New Server**, to open the **Create Server** dialog.
3. Enter a name for the server connection in the **New Server Name** field and click **OK**.
4. Select the newly created server connection in the **Server Data Store** tree, so that its properties show up in the **Properties** view.
5. In the **Properties** view, click on the **Mandatory** tab to configure the basic properties required for an SAP server connection. In this tab, fill in the following property fields to configure the connection:

- **Gateway Host**
- **Gateway Port**
- **Program ID**
- **Repository Destination**
- **Connection Count**

In the **Repository Destination** field, you can fill in one of the destination names from the **Destination Data Store** tree (see [Section 11.2, “Create and Test an SAP Destination Connection”](#)). An SAP server endpoint uses this destination to retrieve meta-data from a remotely hosted meta-data repository.

The other properties are standard SAP server connection properties. Please consult the SAP documentation for more information.

6. You are now ready to test the server connection. In the **SAP Connection** view, right-click on the server connection name and select **Test**.
7. The **Test Server Connection** dialog opens. Click **Server**.
8. The dialog uses the current server configuration settings to start up a server endpoint and connect to the gateway host. If the test is successful, you will see the following messages in the status area:

```
Server state: STARTED
Server state: ALIVE
```

If the test fails, the server status is reported as **DEAD**.

9. Click **Stop**, to shut down the test server.

10. Click **Close**, to close the **Test Server Connection** dialog.

11.4. EXPORT SAP CONNECTION CONFIGURATION TO A FILE

Overview

After creating some SAP destination connections (see [Section 11.2, “Create and Test an SAP Destination Connection”](#)) and SAP server connections (see [Section 11.3, “Create and Test an SAP Server Connection”](#)), you can export the SAP connection configuration to a file (either in Blueprint XML or Spring XML format). You will need this exported configuration data when you come to creating an SAP endpoint (the connection configuration is required in order to configure an SAP endpoint).

Procedure

To export SAP connection configurations to a file, perform the following steps:

1. In the **SAP Connections** view, right-click on **Sap Connection Configuration** and select **Export** to open the **Export SAP Connection Configuration** dialog.
2. Enter the location of the export file in the **Export Location** field, either directly or using the **Browse** button.
3. Select the **Export File Type** using the drop-down menu. The supported file types are: **Blueprint XML** or **Spring XML**. Select the file type that matches the file type you will use to create your Camel routes.
4. Click **Finish**.

11.5. CREATE A NEW SAP ENDPOINT

Overview

You can use the Components palette in the Camel editor to add SAP components to a route. But you must also remember to paste the requisite SAP connection configuration data into your Blueprint XML or Spring XML code.

Prerequisites

You must already have created some SAP destination connections and/or server connections, and have exported this configuration to a file of the appropriate type (Blueprint XML or Spring XML). See [Section 11.4, “Export SAP Connection Configuration to a File”](#).

Procedure

To create a new SAP endpoint, perform the following steps:

1. It is assumed that you already have a Fuse project and a Camel XML file to work with (which could either be in Blueprint XML or Spring XML format).
2. Open your Camel XML file in the Camel editor. If you have already installed the JBoss Fuse SAP Tool Suite, you should be able to see the SAP components under the **Components** palette in the Camel editor. The following SAP components are provided by the tool suite:

- **SAP IDoc Destination**
- **SAP IDoc List Destination**
- **SAP IDoc List Server**
- **SAP qRFC Destination**
- **SAP Queued IDoc Destination**
- **SAP Queued IDoc List Destination**
- **SAP sRFC Destination**
- **SAP sRFC Server**
- **SAP tRFC Destination**
- **SAP tRFC Server**

In the **Design** view of the Camel editor, drag one of these components onto the canvas to create a new SAP endpoint in the current **camelContext**.



NOTE

The SAP Netweaver component does not belong to the JBoss Fuse SAP Tool Suite. It is hosted in the Apache Camel project.

3. Switch to the **Source** view of the Camel editor, by clicking on the **Source** tab at the bottom of the canvas. In this view, you can see the XML source of the routes.
4. Open an exported SAP connection configuration file that you exported earlier (see [Section 11.4, “Export SAP Connection Configuration to a File”](#)), making sure that the type of the exported SAP connection configuration file matches the type of your Camel XML file (Blueprint XML or Spring XML).
5. The SAP connection configuration data is encoded as a **bean** element, with the **id** value, **sap-configuration**. Copy this **bean** element.
6. Return to the **Source** view of the Camel editor. Paste the **bean** element as a child of the root **beans** element in the Camel XML file.
7. You might not need all of the entries in the SAP connection configuration. Delete any unneeded destination connection and server connection entries (**entry** elements).



NOTE

Remember that server connections usually also depend on a destination connection (for retrieving meta-data from a remote repository). So, even if you only need a server connection, do not delete all of the destination connection entries.

8. When specifying an SAP endpoint URI, you must embed either a destination name or a server connection name in the URI format. For example, the **sap-srfc-destination** component has the following URI format:

```
sap-srfc-destination:destinationName:rfcName
```

To reference a particular destination, use the value of the relevant **entry** element's **key** attribute as the ***destinationName*** in this URI.

CHAPTER 12. GETTING STARTED WITH DATA TRANSFORMATION

12.1. FUSE TRANSFORMATION TOOLING

One of the challenges that comes with system and data integration is that the component systems often work with different data formats. You cannot simply send messages from one system to another without translating it into a format (or language) recognized by the receiving system. Data transformation is the term given to this translation.

The Fuse Transformation tooling provided with JBoss Fuse is a GUI to assist developers in implementing data translations as part of Camel routes.



WARNING

JBoss Fuse Transformation tooling is currently provided as a technical preview only, not suitable for production environments. For details on what "technical preview" means, see <https://access.redhat.com/solutions/21101>.

12.2. YOUR FIRST DATA TRANSFORMATION

In this tutorial you will learn how to use the tooling provided by JBoss Fuse to add a data transformation to a predefined Camel route. The Camel route used in this example directs messages from a source endpoint that produces XML data to a target endpoint that consumes JSON data. You will add a data transformation to map the XML data to JSON data in order for this route to work correctly.

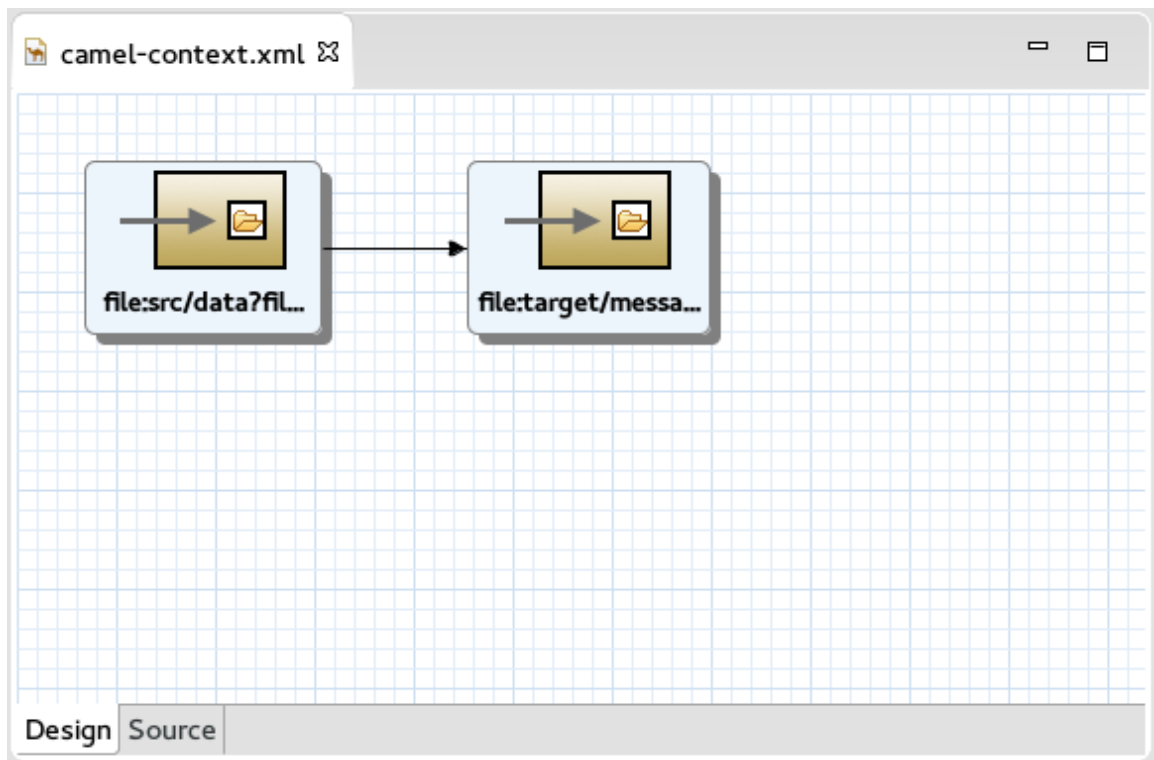
Prerequisites

- JBoss Fuse tooling is installed on JBoss Developer Studio. See [Install Red Hat JBoss Developer Studio Integration Stack](#).
- Maven is installed and configured correctly. See [Red Hat JBoss Fuse Maven Repositories](#).
- You have downloaded the [transformation quickstart applications](#) for use with this technical preview.

Procedure

1. **Import the starter quickstart application in JBoss Developer Studio.**
 - a. Select **File** → **Import**. Expand **Maven** and select **Existing Maven Projects**, then select **Next**.
 - b. You are prompted to select a Maven project to import. Select **Browse** and navigate to the root directory of the **starter** quickstart application, then select **OK**. Review the selection and select **Finish**. The starter application will appear in the **Project Explorer**.

- c. Expand the **starter** project in the **Project Explorer** and navigate to **starter/src/main/resources/META-INF/spring/camel-context.xml**. Open this file to view the Camel route. By default, the file is presented visually on the **Design** tab.



Select the **Source** tab to view the underlying XML. You can see that an XML file is produced from a source endpoint and a JSON file is consumed by a target endpoint.

2. Add a data transformation to the Camel route.

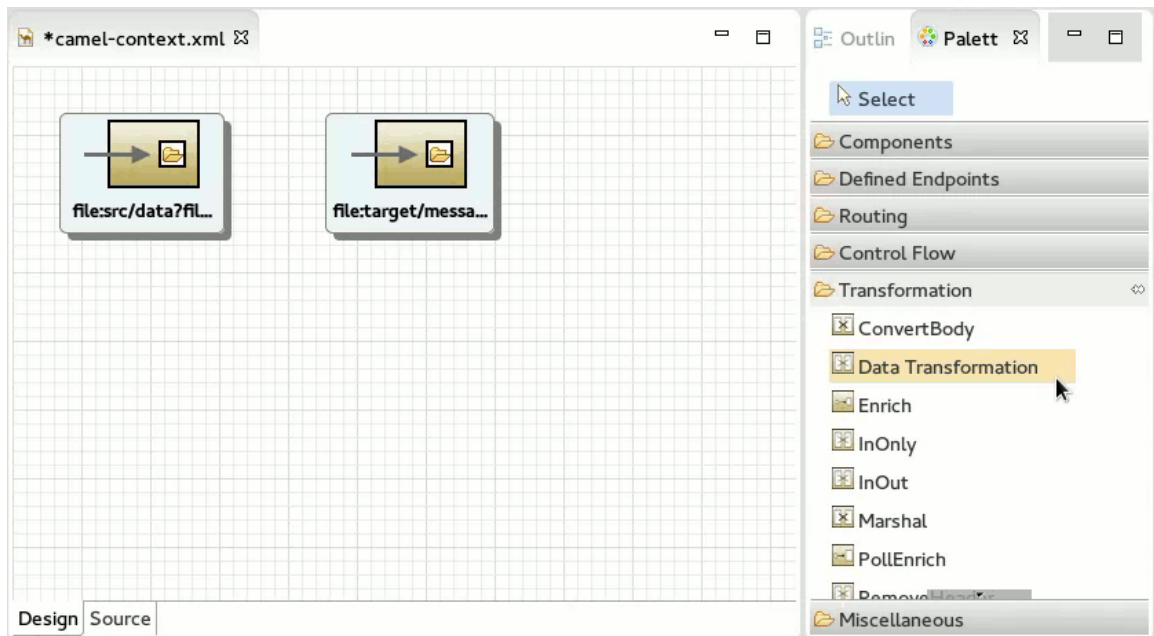
- a. Select the **Design** view for the **camel-context.xml** file.
- b. Remove the arrow connecting the two endpoints.



NOTE

Do not save your project at this point. If you save your project now, the endpoints will disappear.

- c. Click on the **Transformation drawer** in the **Palette**. Drag a **Data Transformation** endpoint onto the canvas.

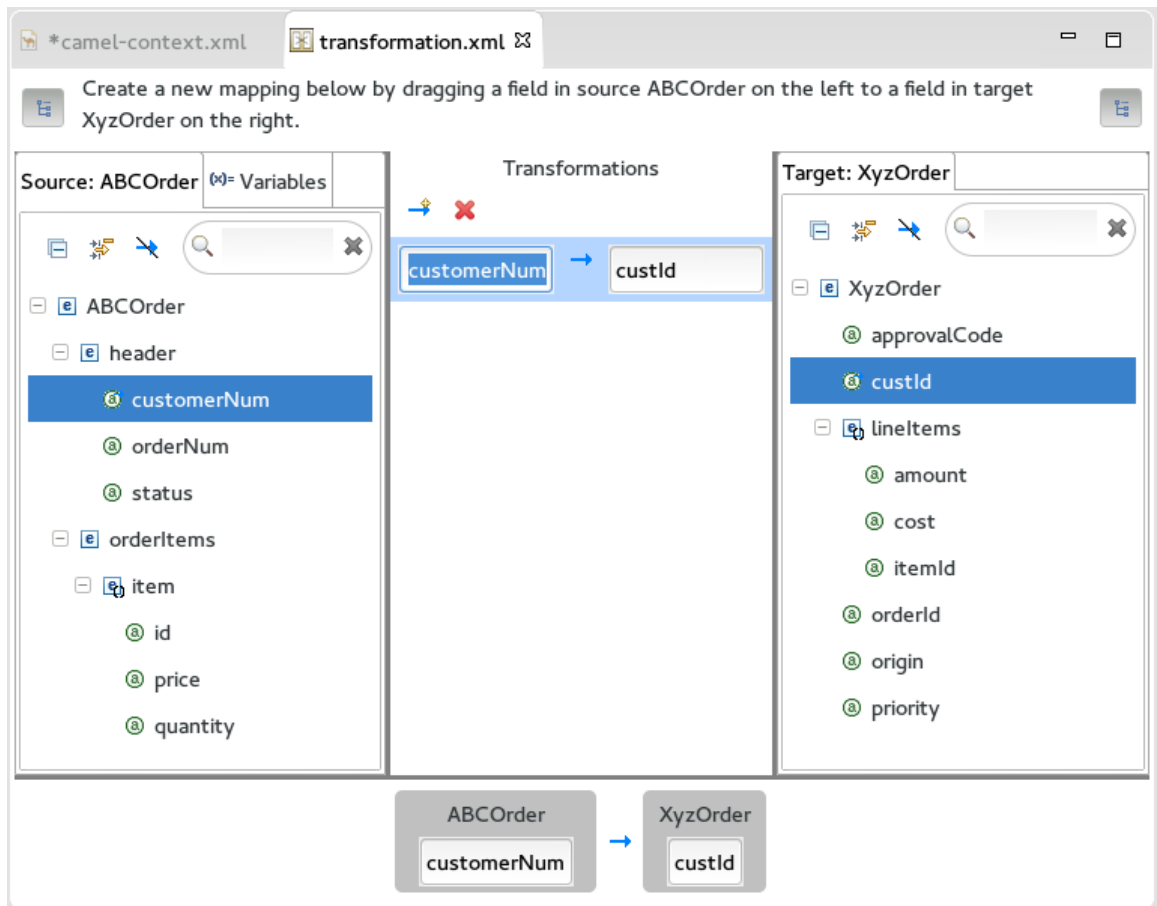


This will invoke the **New Transformation** wizard. If the **Palett** is not yet visible, select **Window** → **Show View** → **Other**, then expand **General** and select **Palett**.

- d. In the **New Transformation** wizard, set the following values (leaving all other default values) then select **Next**.

Field	Value
Transformation ID	xml2json
Source Type	XML
Target Type	JSON

- e. Now you are prompted to provide the source type definition. You can do this by supplying either an **XML Schema** (default) or an example **XML Instance Document**. Leave the default option and browse to select a **Source File**. Choose **abc-order.xsd**, then select **OK**. You are provided with a preview of the XML. Select **Next** to continue.
 - f. Now you are prompted to describe the target type definition in the same way. Browse to select a **Target File**. Choose **xyz-order.json**, then select **OK**. Because this is an example file, the **JSON Type Definition** is updated automatically to reflect this. Select **Finish**.
3. **You are now presented with the transformation editor with source fields on the left and target fields on the right. Map one set to the other.**
 - a. Expand all of the fields on both sides of the transformation editor.
 - b. Drag the source **customerNum** field from the left onto the target **custId** field on the right. The transformation (or mapping) will appear in the center.

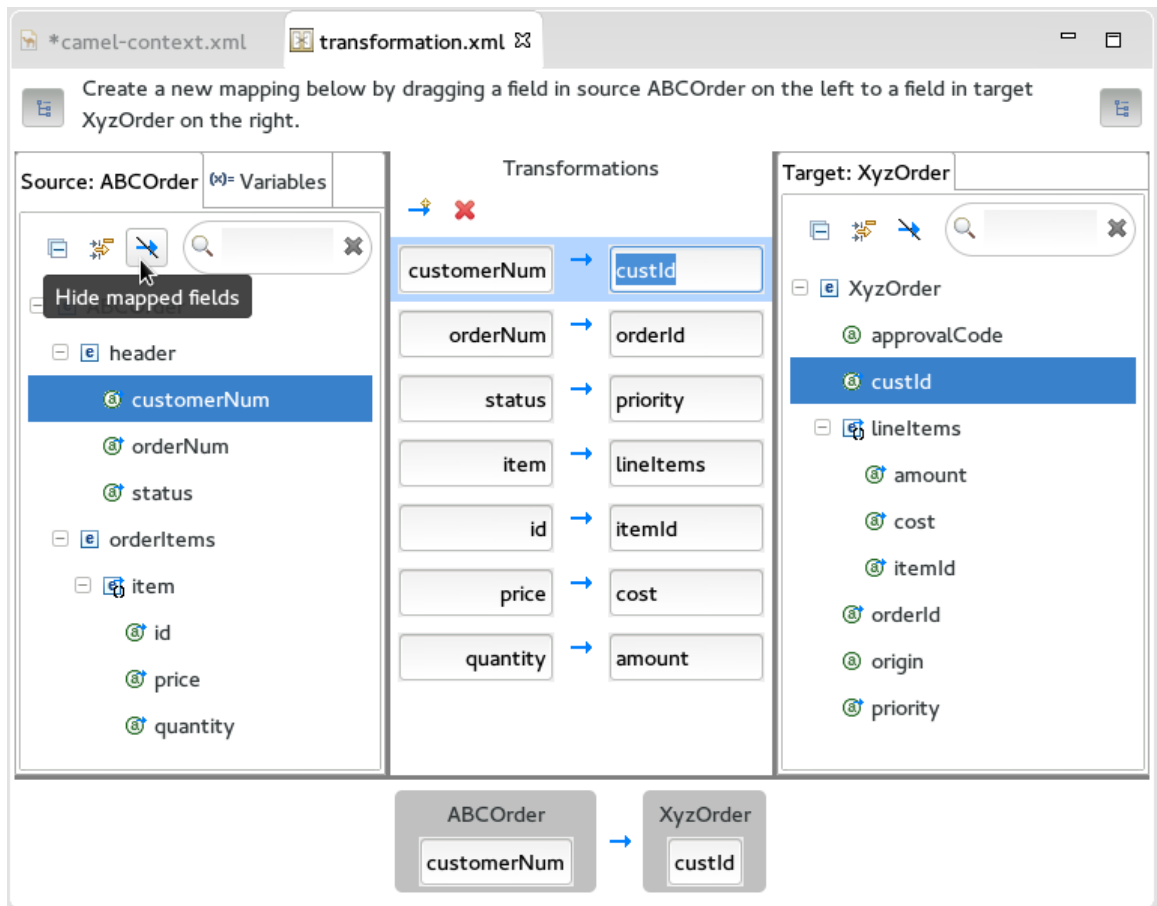


Continue to map the fields as follows:

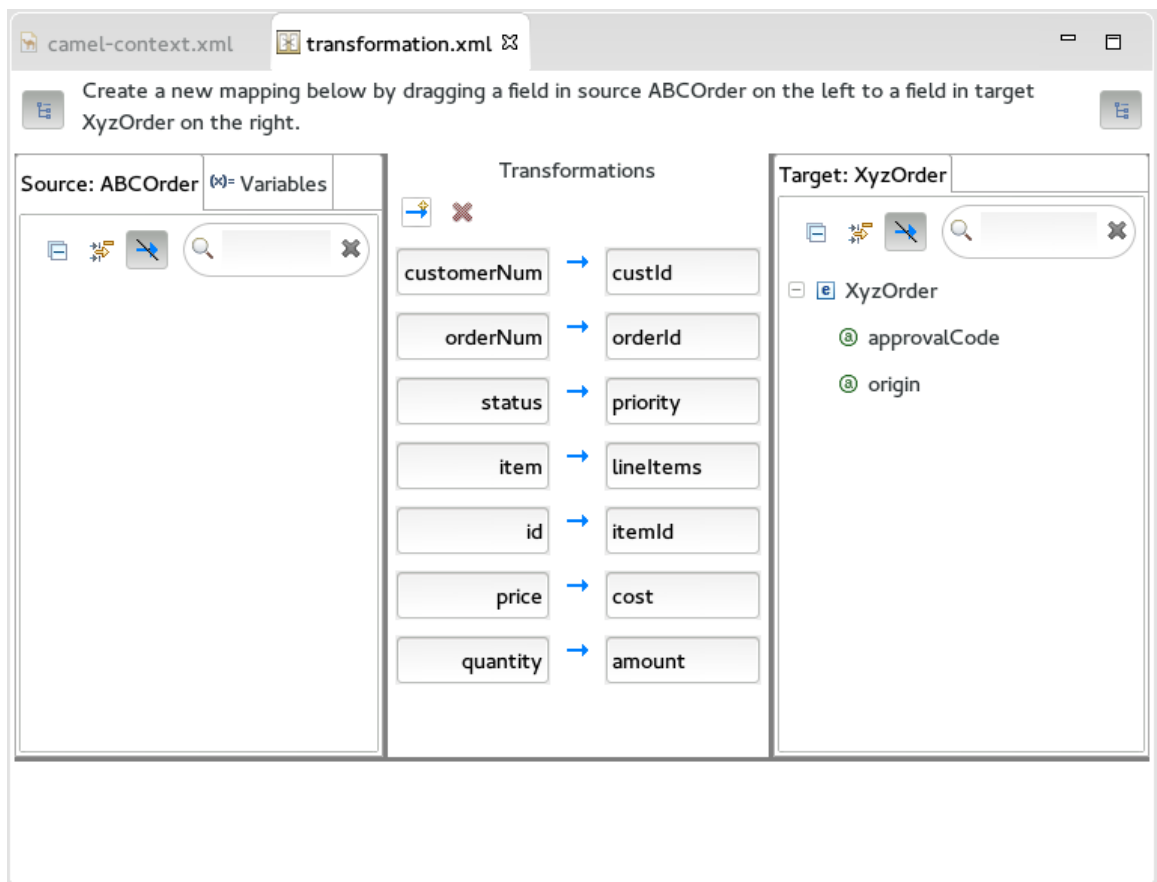
Source	Target
orderNum	orderId
status	priority
item	lineItems
id	itemId
price	cost
quantity	amount

4. Map any remaining fields.

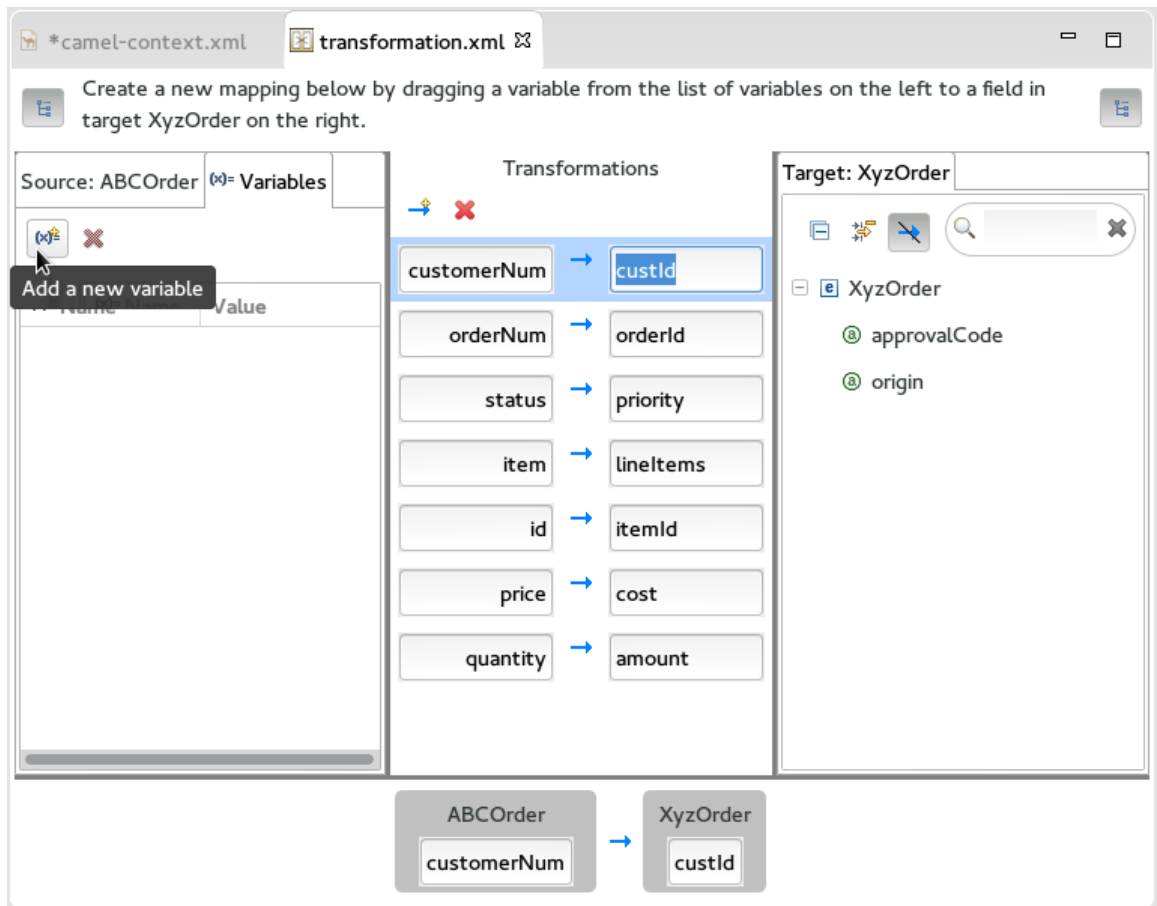
- a. Click on **Hide mapped fields** for both source and target to determine whether all fields have been mapped.



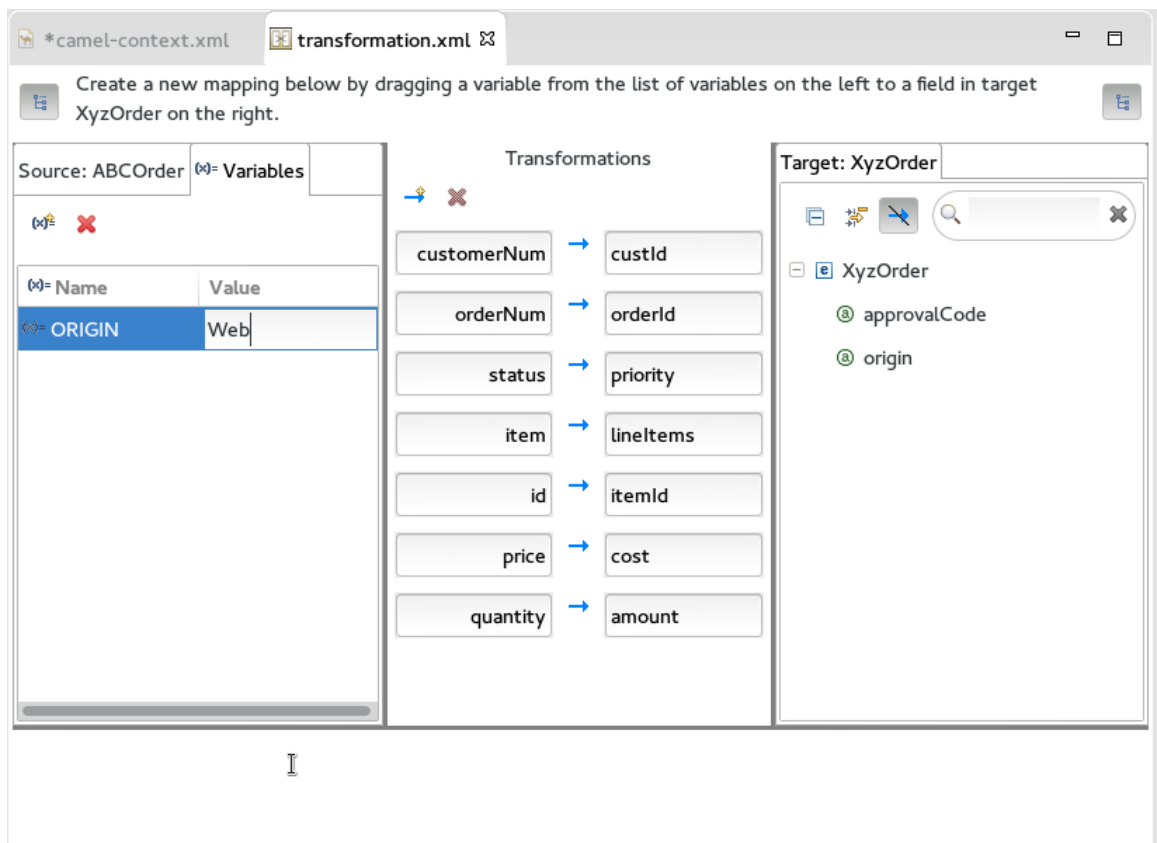
You will find that the target **approvalCode** and **origin** fields remain unmapped.



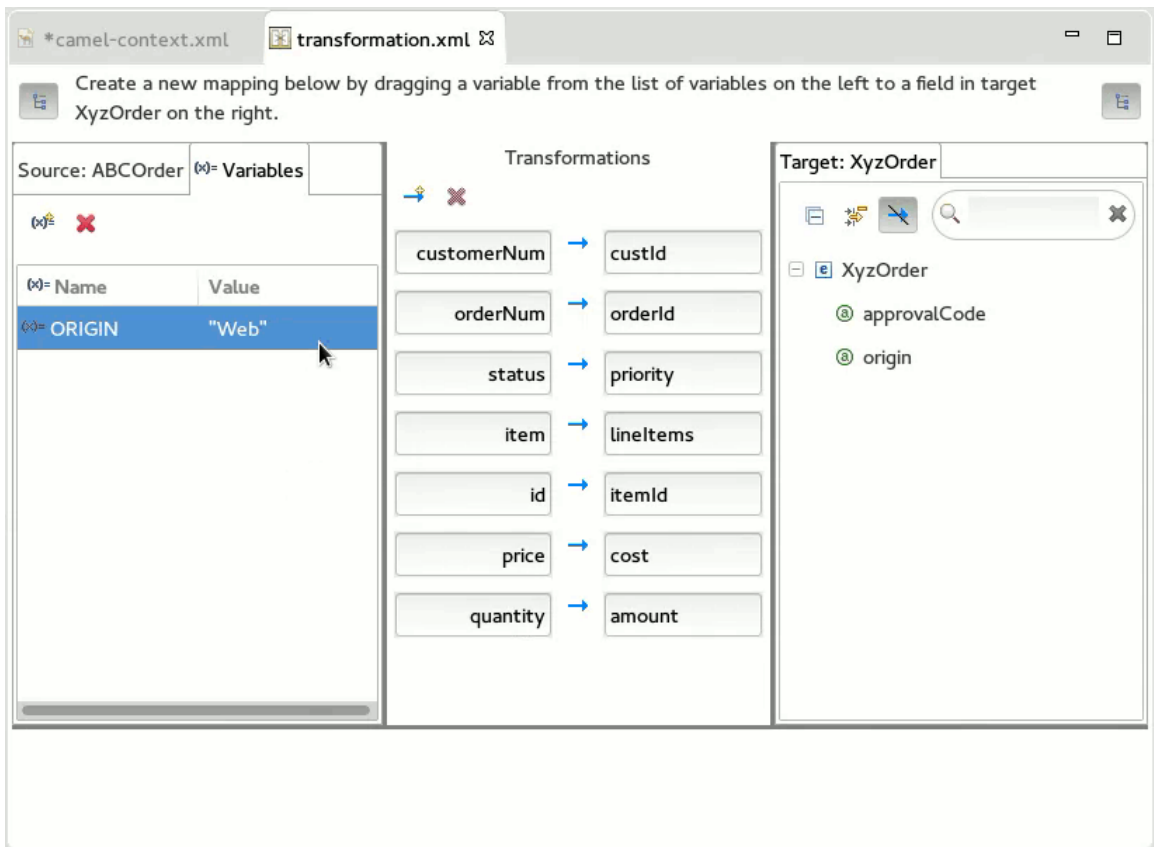
- b. Map the **origin** field to a constant variable: click on the **Variables** tab on the **Source** side. Select **Add a new variable**.



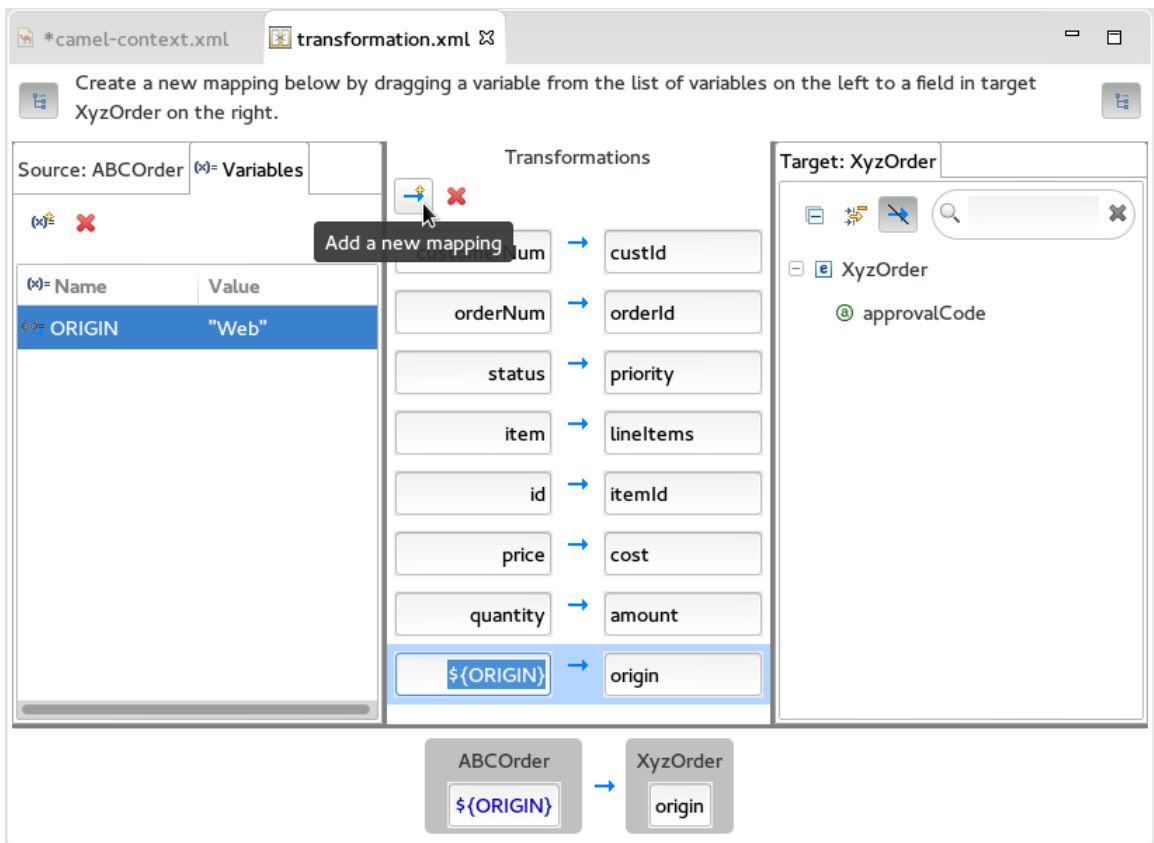
- c. You are prompted to **Enter a new variable name**. Enter **ORIGIN** and select **OK**. The variable is created and appears on the left in the **Variables** window.
- d. Select the default value and change it to **Web**.



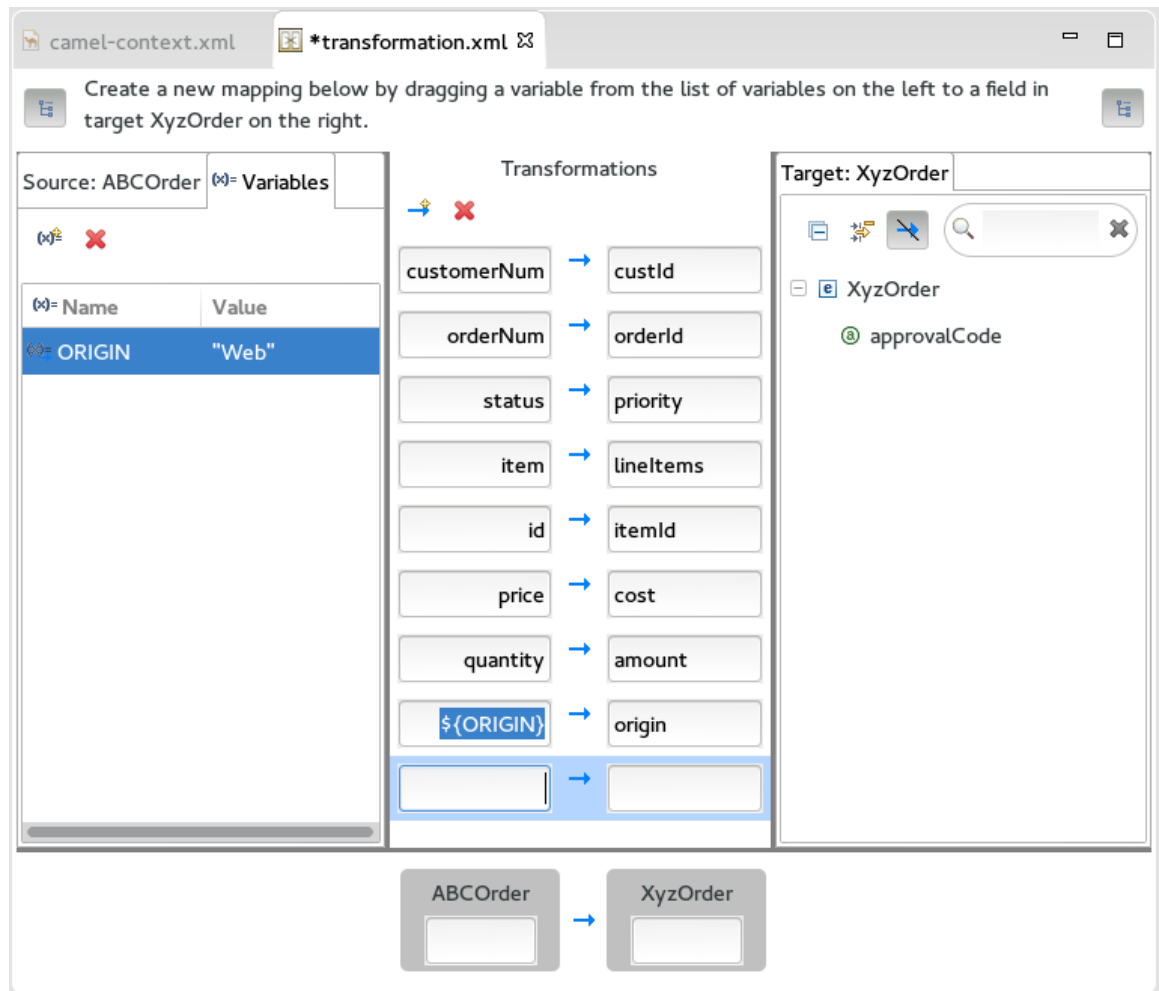
- e. Drag the new variable onto the target **origin** field.



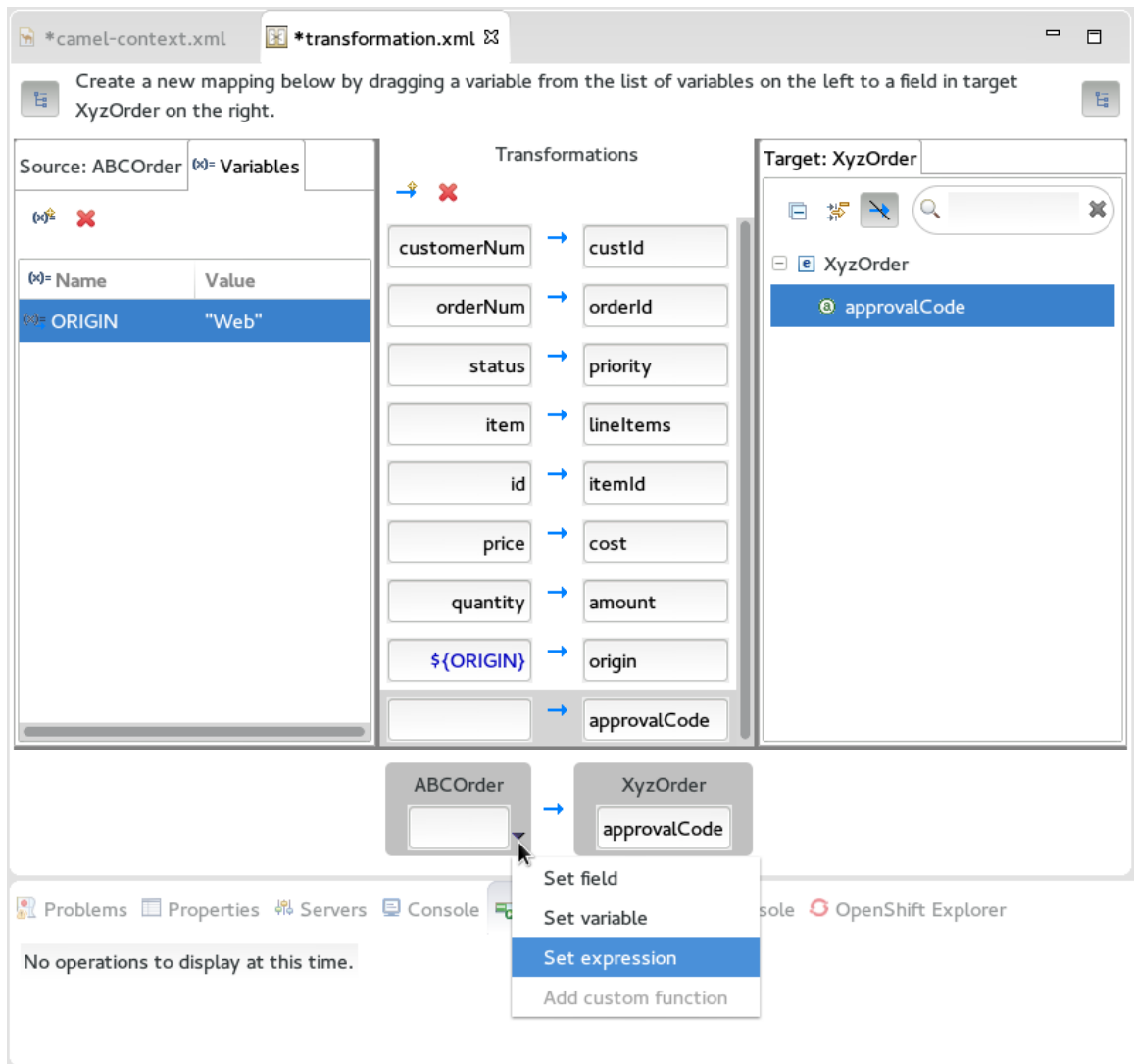
- f. Associate the **approvalCode** field with a dynamic evaluation of a Camel language expression: select **Add a new mapping**.



An empty transformation is added.



- g. Drag **approvalCode** from the target list into the target box of the empty transformation. You will notice that the selected transformation also appears in a window below. When you move your cursor over the source field in this view, an arrow appears. Select this arrow to open the menu from which you select **Set expression**.

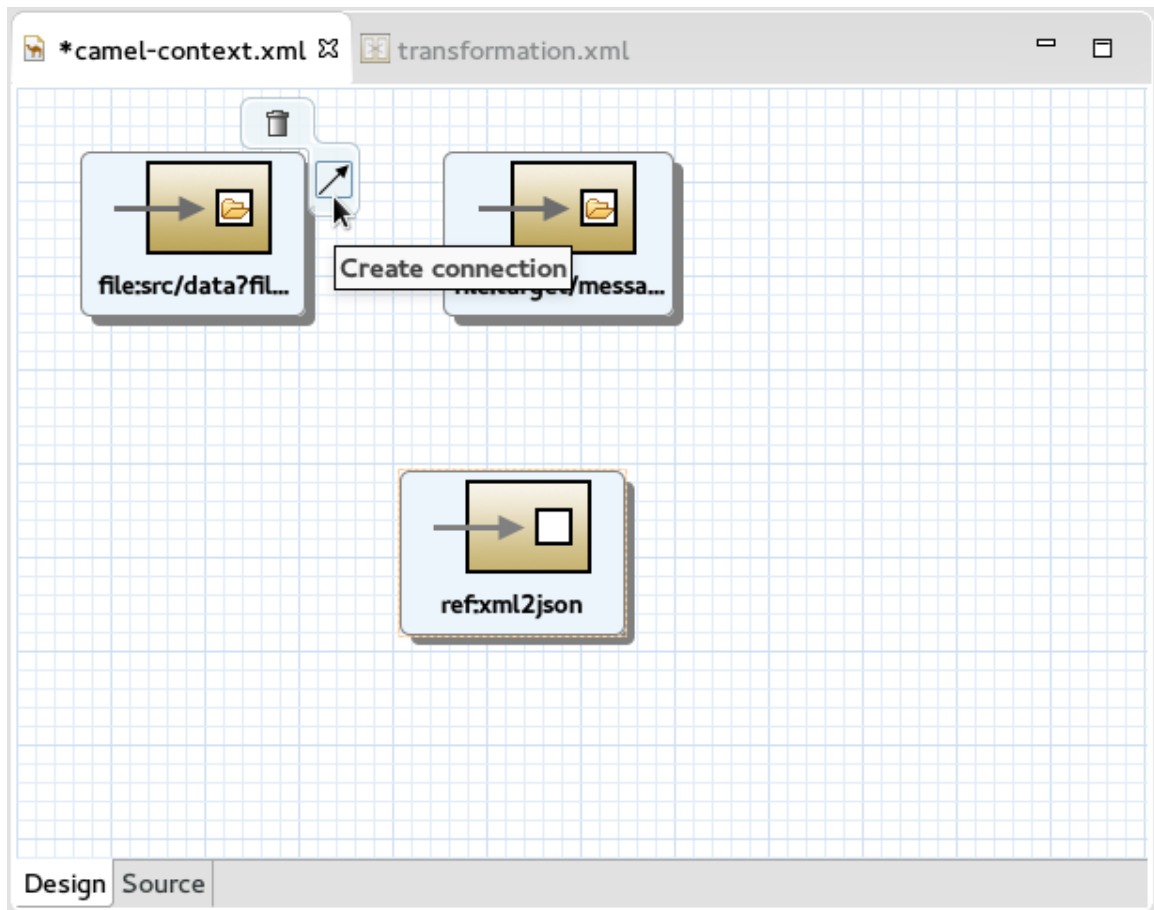


- h. You are prompted to select the expression language, and to enter an expression. Set the following values, then select **OK**:

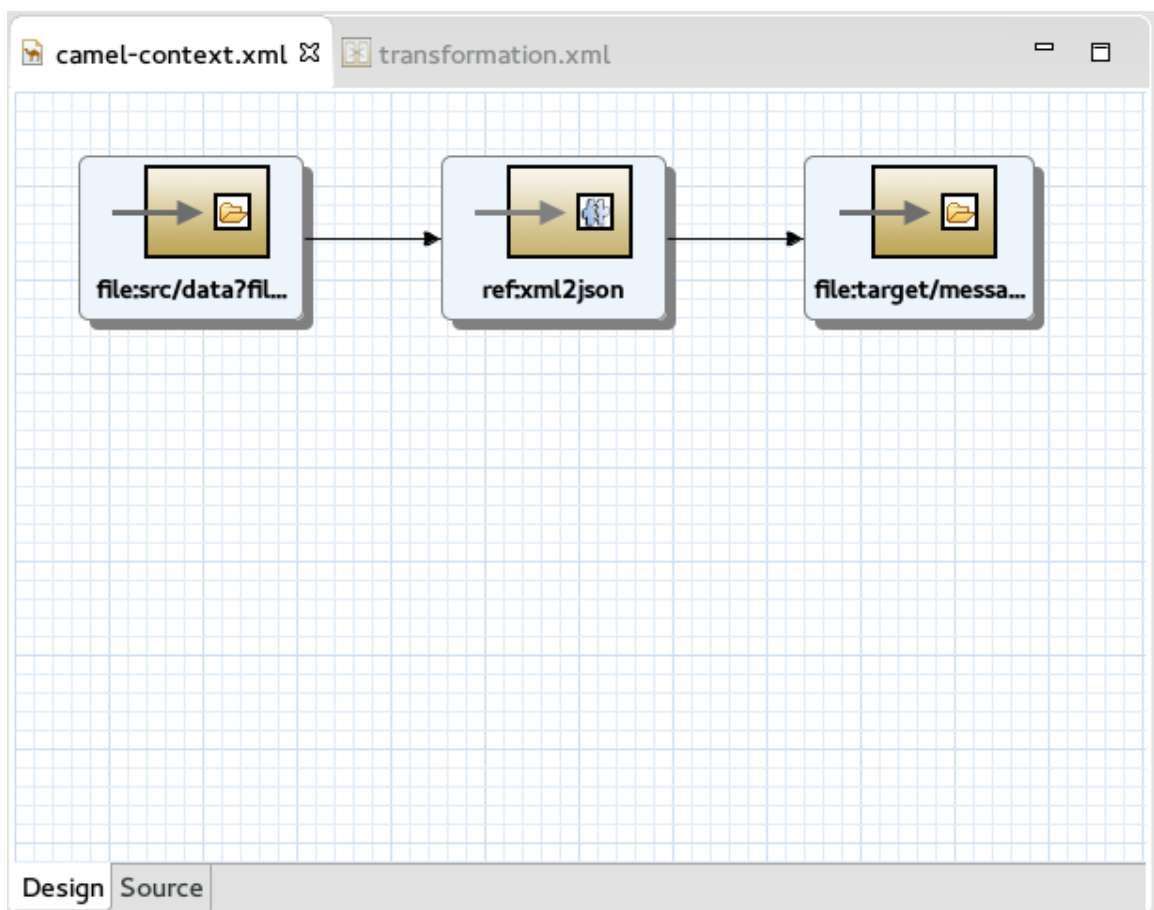
Field	Value
Language	Header
Expression	ApprovalID

5. Connect the endpoints.

- Select the `camel-context.xml` tab to focus on the Camel route again.
- When you hover your cursor over each endpoint, you are presented with an option to **Create connection**.



Create a connection from the source endpoint to the transformation endpoint and then from the transformation endpoint to the target endpoint. Save your changes. The endpoints are automatically arranged horizontally.



6. Test your data transformation.

- a. Select **File** → **New** → **Other**. Expand **Fuse Tooling** and select **Fuse Transformation Test**, then select **Next**. This will invoke the **New Transformation Test** wizard.
- b. In the **New Transformation Test** wizard, set the following values:

Field	Value
Package	example
Transformation ID	xml2json

- c. In the **Project Explorer** navigate to **starter/src/test/java/example** and then open the **TransformationTest.java** file.
- d. Add the following code to the **transform** method:

```
startEndpoint.sendBodyAndHeader(readFile("src/data/abc-order.xml"), "approvalID", "AUTO_OK");
```

- e. Ensure the **TransformationTest.java** file has focus, then select **Run** → **Run As** → **JUnit Test**.
- f. The output will be presented in the **Console** window. Here you can see both the source XML and the resulting target JSON.

Example 12.1. Source XML

```
<ABCOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:java="http://java.sun.com">
  <header>
    <status>GOLD</status>
    <customer-num>ACME-123</customer-num>
    <order-num>ORDER1</order-num>
  </header>
  <order-items>
    <item id="PICKLE">
      <price>2.25</price>
      <quantity>1000</quantity>
    </item>
    <item id="BANANA">
      <price>1.25</price>
      <quantity>400</quantity>
    </item>
  </order-items>
</ABCOrder>
```

Example 12.2. Target JSON

```
{"custId": "ACME-
```

```
123", "priority": "GOLD", "orderId": "ORDER1", "origin": "Web", "approvalCode": "AUTO_OK", "lineItems": [{"itemId": "PICKLE", "amount": 1000, "cost": 2.25}, {"itemId": "BANANA", "amount": 400, "cost": 1.25}]}
```

PART III. DEBUGGING ROUTING CONTEXTS

The Camel debugger includes many features for debugging routing contexts:

- Setting conditional and unconditional breakpoints on nodes in the route editor
- Autolaunching the debugger and switching to Debug Perspective
- Interacting with the running routing context:
 - Switch between breakpoints to quickly compare variable values of message instances
 - Examine and change the value of variables of interest
 - Add variables of interest to the Watch list to track them throughout the debug session
 - Disable and re-enable breakpoints on-the-fly
 - Track message flow graphically in the routing context runtime
 - Examine Console logs to track Camel and debugger actions



NOTE

Before you can run the Camel debugger, you must set breakpoints on the nodes of interest displayed on the route editor's canvas. Once that's done, you can run the Camel debugger on a project's routing context `.xml` file to find the logic errors in it and fix them. Invoking the Camel debugger runs the routing context in debug mode and opens the **Debug** perspective ([Chapter 2, *Debug Perspective*](#)).

CHAPTER 13. SETTING BREAKPOINTS

OVERVIEW

You can set breakpoints in either **JBoss** perspective or in **Fuse Integration** perspective, but **JBoss** perspective provides more room for expanding the route editor's canvas when building and modifying routes. To set breakpoints, your project's routing context `.xml` file must be open in the route editor's **Design View**.

The Camel debugger supports two types of breakpoints:

- Unconditional breakpoints—triggered whenever one is encountered during a debugging session
- Conditional breakpoints—triggered only when the breakpoint's specified condition is met during a debugging session



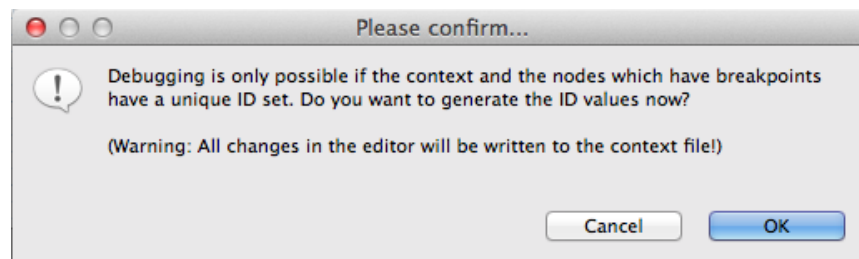
NOTE

You cannot set breakpoints on Consumer endpoints or on When or Otherwise nodes.



NOTE

The Camel debugger requires the `camelContext` element and each node for which a breakpoint is set to have a unique ID. You can set IDs manually in a node's **Properties** editor or directly in the CamelContext file in **Source** view, or you can let Fuse Tooling generate and apply them automatically. The **Please confirm...** dialog opens whenever you set a breakpoint on a node that does not already have a unique ID.




IMPORTANT

Make sure you save any changes you make to your routing context file before you run the Camel debugger on it. Otherwise, your changes will not be included in the debugging session.

SETTING UNCONDITIONAL BREAKPOINTS

With your routing context displayed on the canvas in **Design View**:


1. Select a node whose state you want to examine during the debugging session.
2. Click its  icon to set an unconditional breakpoint.
3. In the **Please Confirm...** dialog, click **OK** to have the Route Editor automatically generate and apply a unique ID to the `<camelContext>` element and to the node or nodes that need it, or click **Cancel** if you want to manually enter unique IDs in the `<camelContext>` element and to

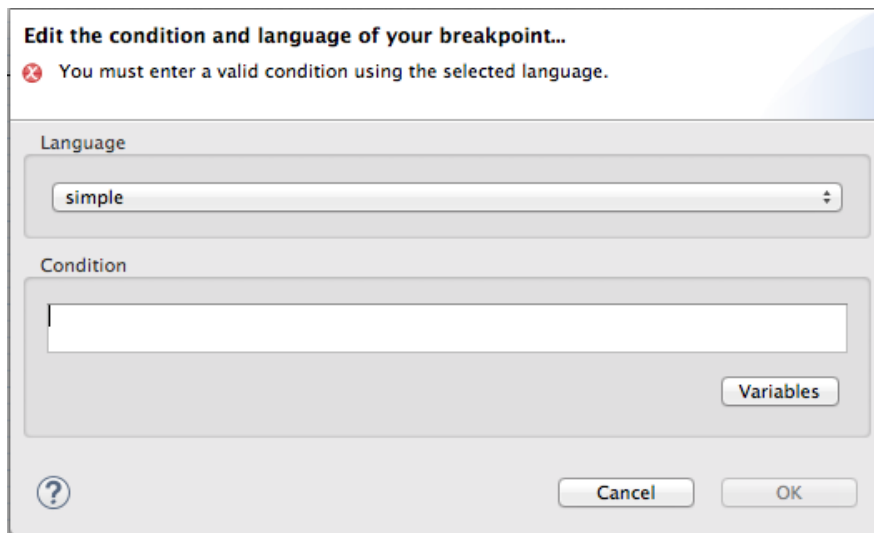
each node that needs one.

- Repeat these steps for each node on which you want to set an unconditional breakpoint.

SETTING CONDITIONAL BREAKPOINTS

With your routing context displayed on the canvas in **Design View**:

- Select a node whose state you want to examine during the debugging session.
- Click its  icon to set a conditional breakpoint and open the **Edit the condition and language of your breakpoint...** dialog.



- Click the **Language** drop-down menu and select the expression language to use to create the condition that will trigger the breakpoint.


Fuse Tooling supports twenty expression languages from which to choose. Except for **constant**, **method**, **property**, and **ref**, each of these languages provides variables for creating conditional expressions.


- Click the **Variables** drop-down menu and select in sequence one or more of the language's supported variables to create the condition for triggering the breakpoint. The variables you select appear in the **Condition** text box.


Alternatively, you can manually enter the expression directly into the **Condition** text box.

- Repeat steps [Step 1](#) through [Step 4](#) for each node on which you want to set a conditional breakpoint.

DISABLING BREAKPOINTS

You can temporarily disable a breakpoint, leaving it in place, then enable it again later. The  button skips over disabled breakpoints during debugging sessions.

To disable a breakpoint, select the node on the canvas and click its  icon. The breakpoint turns gray, indicating it has been disabled.

To enable a disabled breakpoint, select the node on the canvas and click its  icon. Depending on whether the disabled breakpoint is conditional or unconditional, it turns yellow or red, respectively, to indicate it has been re-enabled.





NOTE

You can also disable and re-enable breakpoints during debugging sessions. For details, see [Chapter 18, *Disabling Breakpoints in a Running Context*](#).

DELETING BREAKPOINTS

You can delete individual breakpoints or all breakpoints.

- Individual breakpoints—on the canvas, select the node whose breakpoint you want to delete, and click its  icon.
- All breakpoints—right-click the canvas, and select  **Delete all breakpoints.**

RELATED TOPICS

[Chapter 14, *Starting the Camel Debugger*](#)

[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)

CHAPTER 14. STARTING THE CAMEL DEBUGGER

OVERVIEW



NOTE

You must set breakpoints in your routing context file before you can start the Camel debugger.

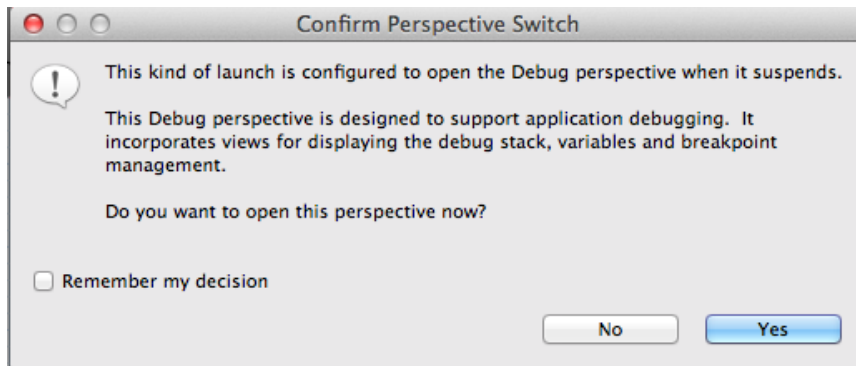
You run the Camel debugger on the routing context file, so be sure to save any changes you made to the file before you invoke the debugger. Otherwise your changes will not be included in the debugging session.

PROCEDURE

1. In **Project Explorer**, select the routing context file you want to debug.
2. Right-click it to open the context menu, and then select **Debug As... → Local Camel Context**.

Fuse Tooling builds the Camel route, starts up Apache Camel, starts the routing context, enables JMX, starts the route(s) in the routing context, adds the breakpoints to the nodes, and enables the Camel debugger.

The Camel debugger suspends execution of the routing context at the first breakpoint it encounters, and asks whether you want it to open the **Debug** perspective.



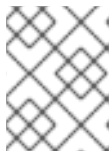
3. Click **Yes** to open **Debug** perspective.

Debug perspective opens with the routing context suspended at the first breakpoint in the running routing context.



IMPORTANT

Breakpoints are held for a maximum of five minutes, after which debugging automatically resumes, moving on to the next breakpoint or to the end of the routing context if there are no more breakpoints.



NOTE



To see the console output, you'll need to open **Console** view if it was not open when you switched perspectives.



NOTE

By default, **Debug** perspective opens displaying **Outline** view, which provides the means to switch between separate routes in a routing context. If your routing context contains a single route, closing **Outline** view will provide more space to expand the other views, making it easier to access and examine debugger output.

WATCHING MESSAGE EXCHANGES PROGRESS THROUGH THE ROUTING CONTEXT

Click  (Step Over) to jump to the next node of execution in the routing context. Click  (Resume) to continue execution at the next active breakpoint in the routing context.

The screenshot displays the Red Hat JBoss Fuse 6.2 IDE interface. At the top, the 'Servers' panel shows a list of running threads and processes, including 'log1 in Route1 [blueprint.xml]'. The 'Breakpoints' panel shows a table of breakpoints:

Name	Value
MinProcessingTime	0
AverageProcessingTime	0
MaxProcessingTime	0
TotalProcessingTime	0
Exchange	CamelExchange (id=1582720311)
ExchangeId	ID-dhpc-97-77-bos-redhat-com-59064-1415738575781-0-2
NodeId	log1
RouteId	Route1
Timestamp	2014-11-11T15:47:58.244-0500
UID	2
Message	CamelMessage (id=1582720311)
MessageBody	<?xml version="1.0" encoding="UTF-8"?>\n<person user="James">\n <firstName>Jan
MessageHeaders	CamelFileAbsolute = false\nCamelFileAbsolutePath = /Users/jmurphey/workspace/Blue
MessageId	ID-dhpc-97-77-bos-redhat-com-59064-1415738575781-0-2

The design canvas shows a Camel blueprint with a 'choice1' component that branches into two paths. The top path includes 'when1', 'log1' (highlighted with a red dashed box), 'setHead1', and 'to1'. The bottom path includes 'choice1', 'log2', 'setHead2', and 'to2'. The console window at the bottom shows the following log output:

```

/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Nov 11, 2014, 3:42:42 PM)
[TCP Connection(12)-10.18.97.77] BacklogDebugger INFO Step breakpoint choice1
[TCP Connection(12)-10.18.97.77] BacklogDebugger INFO Resume breakpoint choice1
[TCP Connection(12)-10.18.97.77] BacklogDebugger INFO Dump trace message from breakpoint log1
[TCP Connection(12)-10.18.97.77] BacklogDebugger INFO Dump trace message from breakpoint log1

```



RELATED TOPICS

[Chapter 15, Stopping the Camel Debugger](#)

[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)


CHAPTER 15. STOPPING THE CAMEL DEBUGGER

OVERVIEW

To stop the Camel debugger, click  on the menu bar once if the debugging session has ended. Otherwise you need to click  twice. Once to terminate the currently running node thread and once to terminate the **Camel Context** thread (both displayed in **Debug** view).



NOTE

Terminating the Camel debugger also terminates the Console but does not clear its output. To clear the output, click  (Clear Console) on **Console** view's menu bar.

TERMINATING THE CAMEL DEBUGGER BEFORE THE SESSION ENDS

Terminating the debugger before the debugging session ends may add extraneous entries to some of the project's folders in **Project Explorer**. You may see message lock files or **target/.CamelContextInDebug_XXXXXXXX_temp/camel-context.xml** lines in the root project's **src/main/resources/framework-INF/framework_name** folder. In **Fuse Integration** perspective, You may also see them in the project's **Camel Contexts** folder.

To clear them, right-click the root project, and then select **Refresh**.

RELATED TOPICS

[Chapter 14, Starting the Camel Debugger](#)

[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)

CHAPTER 16. CHANGING VARIABLE VALUES

OVERVIEW

When the Camel debugger hits a breakpoint, **Variables** view displays the values of all variables available at that point in the routing context. Some variables are editable, allowing you to change their value. This enables you to see how the application handles changes in program state.



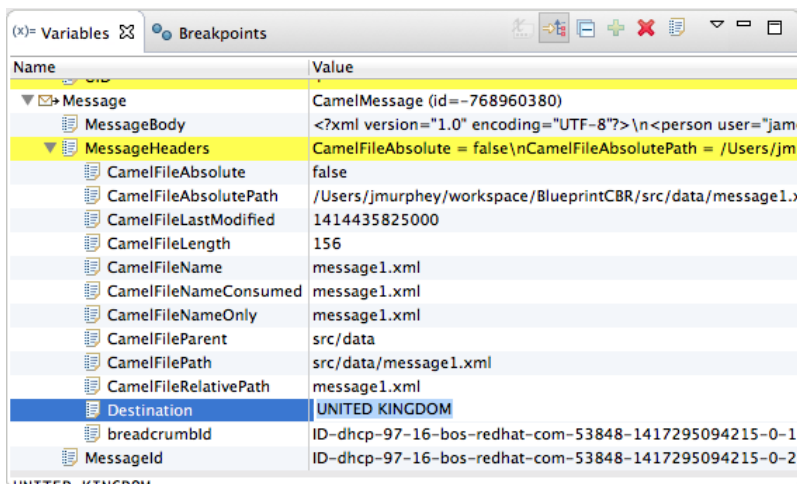
NOTE

Not all variables are editable. The context menu of those that are displays the option, **Change Value...**

PROCEDURE

To change the value of a variable:

1. If necessary, start up the debugger. See [Chapter 14, Starting the Camel Debugger](#).
2. In **Variables** view, select a variable whose value you want to change, and then click its **Value** field.



The variable's value field turns a lighter shade of blue, indicating it is in edit mode.



NOTE

Alternatively, you can right-click the variable to open its context menu, and select **Change Value...** to edit its value.

3. Enter the new value and then click **Enter**.

Console view displays an INFO level log entry noting the change in the variable's value (for example, **Breakpoint at node to1 is updating message header on exchangeId: ID-dhcp-97-16-bos-redhat-com-52574-1417298894070-0-2 with header: Destination and value: ChangedTarget**).

4. Continue stepping through the breakpoints and check whether the message is processed as expected. At each step, check **Debug** view, **Variables** view, and the **Console** output.

RELATED TOPICS

[Chapter 18, *Disabling Breakpoints in a Running Context*](#)

[Chapter 17, *Adding Variables to the Watch List*](#)

[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)

CHAPTER 17. ADDING VARIABLES TO THE WATCH LIST

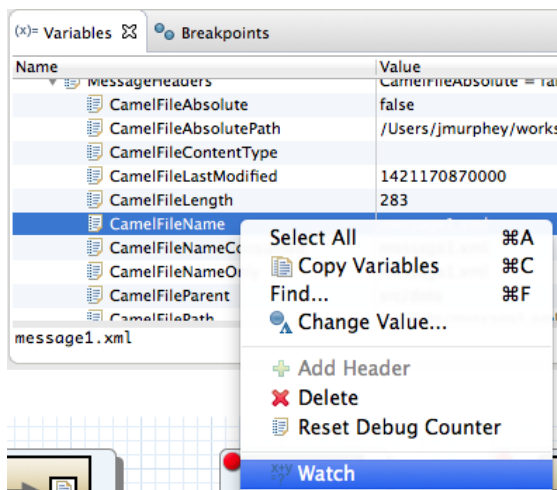
OVERVIEW

By adding them to the watch list, you can easily zero in on particular variables to see whether their values change as expected over the course of the routing context.

PROCEDURE

To add a variable to the watch list:

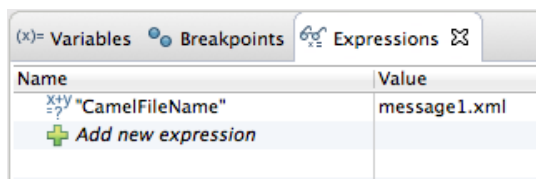
1. If necessary, start up the debugger. See [Chapter 14, Starting the Camel Debugger](#).
2. In **Variables** view, right-click a variable you want to track to open the context menu.



3. Select **Watch**.

A new tab, **Expressions**, appears next to the **Breakpoints** tab.

4. Click the **Expressions** tab to open it.



Expressions view displays the name of the added variable and its current value.

5. Repeat [Step 2](#) through [Step 4](#) to add additional variables to the Watch list.



NOTE

The variables you add will remain in the watch list until you remove them. To do so, right-click a variable in the list to open the context menu, and then click **Remove**.

6. With **Expressions** view open, step through the routing context to track how the value of each variable in the Watch list changes at each step in the routing context.

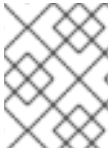
RELATED TOPICS

[Chapter 16, *Changing Variable Values*](#)

CHAPTER 18. DISABLING BREAKPOINTS IN A RUNNING CONTEXT


OVERVIEW

You can disable and re-enable breakpoints in a running routing context in either **Breakpoints** view or on the canvas of the running **CamelContext**: `<xxxxxx>`. You can delete and reset a deleted breakpoint only on the canvas of the running **CamelContext**: `<xxxxxx>`.



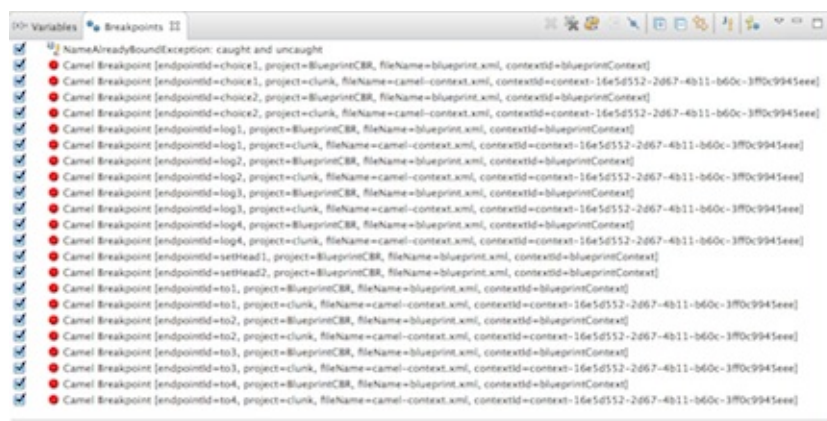
NOTE

The string `<xxxxxx>` represents the ID that you or the debugger assigned to the **camelContext** element in the project's routing context file.

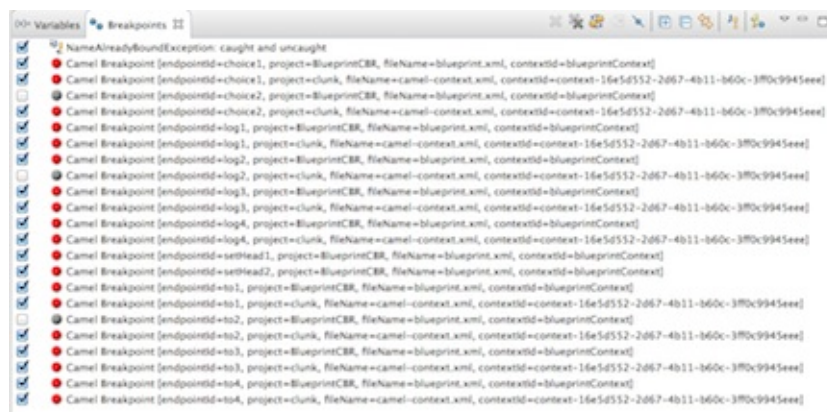
When a breakpoint is disabled, the  button causes the debugger to skip over it during the debugging session.

DISABLING AND ENABLING BREAKPOINTS IN BREAKPOINTS VIEW

Breakpoints view opens with all set breakpoints enabled.



To disable a breakpoint, clear its check box.



For each breakpoint you disable, **Console** view displays an INFO level log entry noting that it has been disabled (for example, **Removing breakpoint to3**). Likewise, for each breakpoint you re-enable, **Console** view displays an INFO level log entry noting that it has been enabled (for example, **Adding breakpoint to3**).

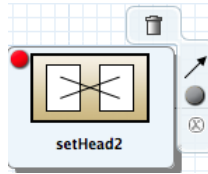
**NOTE**

To re-enable a disabled breakpoint, click its check box. **Console** view displays an INFO level log entry noting that the breakpoint has been added to the selected node.

DISABLING AND ENABLING BREAKPOINTS ON THE RUNNING CAMELCONTEXT: CANVAS

Make sure the routing context running in memory, **CamelContext**: <xxxxxx>, is active on the canvas.

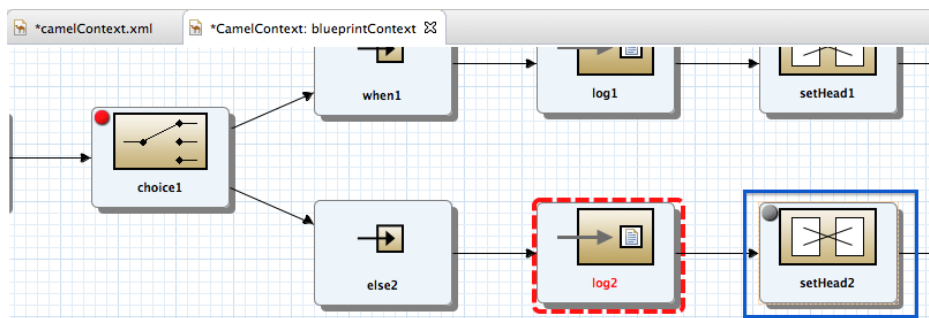
1. On the canvas, select the node with the breakpoint you want to disable; for example:

**NOTE**

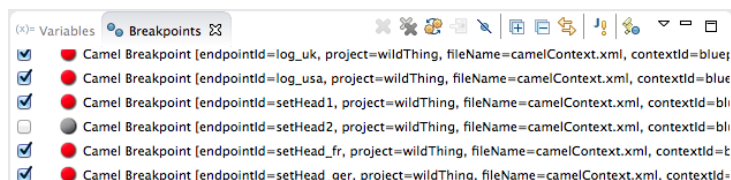
The ● icon denotes an unconditional breakpoint. A ● icon denotes a conditional breakpoint.

2. Click the ● icon in the node's side bar.

The ● icon replaces the ● icon in the node's upper left corner, indicating that the breakpoint is now disabled:



The node's check box in **Breakpoints** view is clear, also indicating that the breakpoint is disabled:

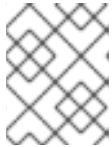


Console view contains a log of the event:


```

Console Tasks Properties
/Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (May 13, 2015, 3:34:29 PM)
[TCP Connection(9)-192.168.1.2] BacklogDebugger INFO Removing breakpoint setHead2
[⌘] thread #2 - file://src/data BacklogDebugger INFO NodeBreakpoint at node setHead2 is continued exchangeId:
[⌘] thread #2 - file://src/data BacklogDebugger INFO NodeBreakpoint at node toFulfill is waiting to continue
[TCP Connection(11)-192.168.1.2] BacklogDebugger INFO Resume breakpoint log2
[TCP Connection(11)-192.168.1.2] BacklogDebugger INFO Dump trace message from breakpoint toFulfill
[TCP Connection(11)-192.168.1.2] BacklogDebugger INFO Dump trace message from breakpoint toFulfill
[TCP Connection(11)-192.168.1.2] BacklogDebugger INFO Adding breakpoint log2

```



NOTE

To re-enable the breakpoint, select the node, and then click the  icon in its side bar.

DELETING AND RESETTING BREAKPOINTS ON THE RUNNING CAMELCONTEXT: CANVAS

You can delete and reset deleted breakpoints only on the running **CamelContext**: `<xxxxxx>` canvas.


1. On the **CamelContext**: `<xxxxxx>` canvas, select the node whose breakpoint you want to delete; for example, **setHead_usa**:

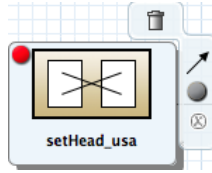
Note that the node's breakpoint is enabled in **Breakpoints** view.



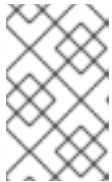
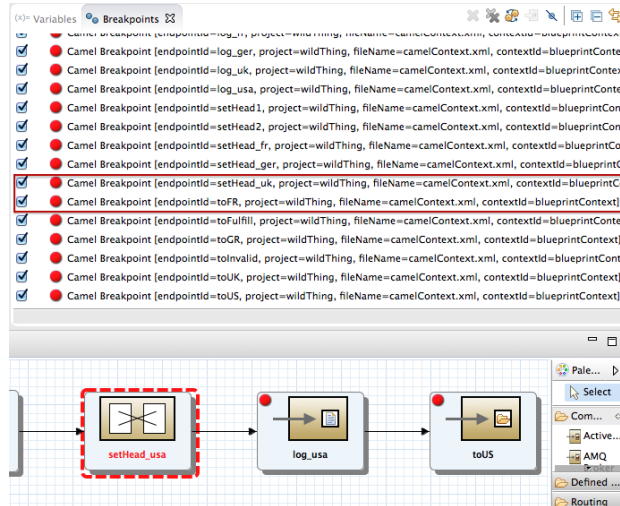
NOTE

In the above graphic, the dotted red line surrounding the **setHead_usa** node identifies the breakpoint on which the debugger is currently suspended.



2. Click the  icon in the node's side bar:



The breakpoint is removed from the node on the canvas and also from the list of breakpoints in **Breakpoints** view:



NOTE

To reset the breakpoint, select the node, and then click the  or  icon in its side bar, depending on whether you want to set an unconditional or conditional breakpoint.

RELATED TOPICS




[Chapter 13, *Setting Breakpoints*](#)

[the section called “Disabling breakpoints”](#)

[the section called “Deleting breakpoints”](#)

[chapter "To Debug a Routing Context" in "Tooling Tutorials"](#)

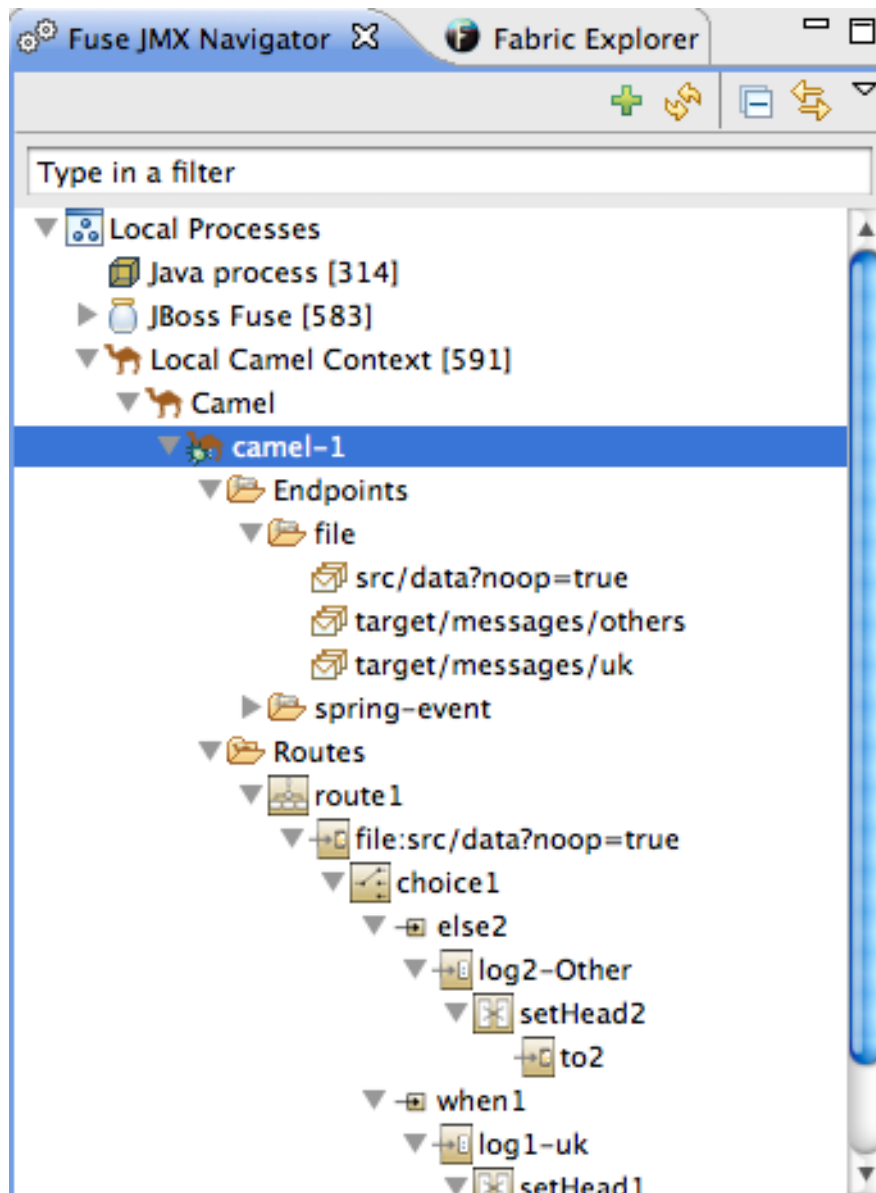
PART IV. MONITORING AND TESTING APPLICATIONS

To open **Fuse Integration** perspective ([Chapter 3, *Fuse Integration Perspective*](#)), click  on the right side of the JBoss Developer Studio tool bar. If the  icon is not available on the tool bar, click  and then select **Fuse Integration** from the list of available perspectives.

CHAPTER 19. JMX NAVIGATOR

JMX Navigator, shown in [Figure 19.1](#), displays all of the processes running in your application and drives all interactions with the monitoring and testing features. Other areas of the **Fuse Integration** perspective adapt to display information related to the node selected in JMX Navigator. JMX Navigator's context menu also provides the commands needed to activate route tracing and to add JMS destinations.

Figure 19.1. JMX Navigator



By default, **JMX Navigator** discovers all of the JMX servers running on the local machine and lists them under the lists them following categories:

- Local Processes
- Server Connections
- User-defined connections

You can add other JMX servers using a server's JMX URL.

19.1. VIEWING PROCESSES IN JMX


Overview

JMX Navigator lists all known processes 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 **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 **JMX Navigator**.
2. Expand the server's entry in **JMX Navigator** using the  icon that appears next to the entry to display the list of all components running in the JVM.

19.2. ADDING A JMX SERVER


Overview

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 **JMX Navigator**:

1. Click **New Connection**  on the **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 **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 **JMX Navigator** tree.

CHAPTER 20. VIEWING A COMPONENT'S PROPERTIES

OVERVIEW

The tooling collects and displays 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 .

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 21. BROWSING MESSAGES

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 tooling 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 .

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 22.3, “Tracing messages through a route”](#)

CHAPTER 22. TRACING 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 the easiest way to discover the source of these problems.

The route tracing feature of the tooling enables 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 **JMX Navigator** tree. Selecting an individual message in **Messages View** displays the full details and content of the message in **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](#)

22.1. CREATING TEST MESSAGES FOR ROUTE TRACING

Overview

Route tracing works with any kind of message structure. The **Fuse Message** wizard creates an empty message, leaving the structuring of the message entirely up to you.

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 **New File** wizard.
3. Enter or select the folder in which you want to store the message.
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 22.3, “Tracing messages through a route”](#)

[New Fuse Message](#)

[chapter “To Trace a Message Through a Route” in “Tooling Tutorials”](#)

22.2. 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 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** to start the trace.
If **Stop Tracing Context** is enabled on the context menu, then tracing is already active.

Related topics

[Section 22.3, “Tracing messages through a route”](#)

[Section 22.4, “Deactivating route tracing”](#)

22.3. 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 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 22.1, “Creating test messages for route tracing”](#).
2. Activate tracing for the route's routing context as described in [Section 22.2, “Activating route tracing”](#).
3. Drag one of the test messages onto the route's starting point.
4. In the 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 22.1, “Creating test messages for route tracing”](#)

[Section 22.2, “Activating route tracing”](#)

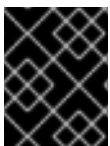
[Section 22.4, “Deactivating route tracing”](#)

[chapter "To Trace a Message Through a Route" in "Tooling Tutorials"](#)

22.4. 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 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.

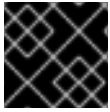
Related topics

[Section 22.2, “Activating route tracing”](#)

[Section 22.3, “Tracing messages through a route”](#)

CHAPTER 23. MANAGING JMS DESTINATIONS

The tooling's **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.

23.1. 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 tooling provides 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 23.2, "Deleting a JMS destination"](#)

23.2. 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 tooling has 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 23.1, “Adding a JMS destination”](#)

CHAPTER 24. MANAGING ROUTING ENDPOINTS

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



IMPORTANT

These changes are not persistent across router restarts.

24.1. ADDING A ROUTING ENDPOINT

Overview

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

Procedure

To add an endpoint to a routing context:

1. In the **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.
4. Click **OK**.

The new destination appears in the tree.

Related topics

[Section 24.2, “Deleting a routing endpoint”](#)

24.2. 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 tooling has a menu option that allows you to remove an endpoint with a single click.

Procedure

To delete a routing endpoint:

1. In the **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

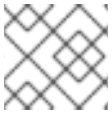
[Section 24.1, “Adding a routing endpoint”](#)

CHAPTER 25. EDITING RUNNING ROUTES

OVERVIEW

If you want to experiment with changes to a running route, you can make the edits directly from the Fuse Integration perspective. The tooling opens 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 **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 II, “Developing Applications”](#).
4. When done, save the edited route.

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

Changes take effect immediately.

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

RELATED TOPICS

[Section 7.1, “Adding routes to the routing context”](#)

[Section 7.2, “Adding patterns to a route”](#)

[Section 7.4, “Configuring a pattern”](#)

[Section 7.3, “Connecting patterns to make a route”](#)

[Section 7.5, “Rearranging patterns on the canvas”](#)

[Section 7.6, “Removing patterns from a route”](#)

[Section 7.7, “Disconnecting two patterns”](#)

Section 7.8, "Deleting a route"

CHAPTER 26. MANAGING ROUTING CONTEXTS

JMX Navigator enables you to easily suspend and resume routing contexts.

26.1. SUSPENDING A ROUTING CONTEXT

Overview

The tooling enables you to suspend the operation of a routing context from **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.

Procedure

To suspend a routing context:

1. In the **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 26.2, “Resuming a routing context”](#)

26.2. RESUMING A ROUTING CONTEXT

Overview

The tooling enables you to resume a suspended routing context from **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 **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 26.1, “Suspending a routing context”](#)

PART V. PUBLISHING APPLICATIONS TO A CONTAINER

To publish Fuse projects to a server container, you must first add the server and its runtime definition to the tooling's **Servers** list. Then you can assign Fuse projects to the server runtime and set the publishing options for it.

You manage servers and publish Fuse projects to them in **Fuse Integration** perspective (for an overview, see [Chapter 3, *Fuse Integration Perspective*](#)). To open **Fuse Integration** perspective, click  on the right side of the JBoss Developer Studio tool bar. If the  icon does not appear on the tool bar, click  and then select **Fuse Integration** from the list of available perspectives.

CHAPTER 27. MANAGING SERVERS

The **Servers** panel enables you to run and manage servers in your Red Hat Developer Studio environment. The supported servers are:

- Red Hat JBoss Fuse Server—versions 6.x
- Apache Karaf Server—versions 2.x, 3.0
- Apache ServiceMix Server—versions 4.5, 5.x



NOTE

For step by step instructions on how to publish a Camel project to Red Hat JBoss Fuse, see [Red Hat JBoss Fuse Tooling Tutorials](#), To Publish a Fuse Project to Red Hat JBoss Fuse.

27.1. ADDING A SERVER

Overview

For the tooling to manage a server, you need to add the server to the **Servers** list. Once added, the server appears in **Servers** panel, where you can connect to it and publish your Fuse projects.



NOTE

If adding a Red Hat JBoss Fuse 6.0 or 6.1 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 tooling to establish an SSH connection to the server.

Procedure

You can add a new server to the **Servers** panel in three ways:

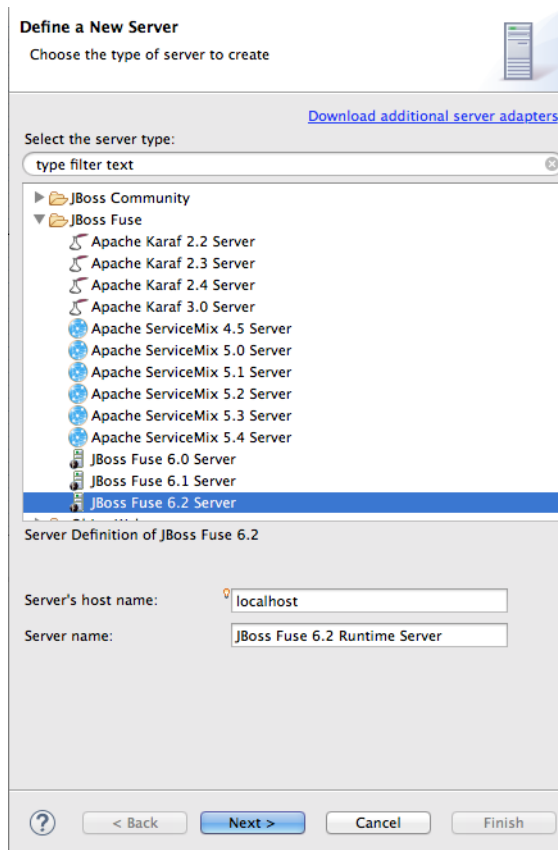
- In **Servers** panel, click the link **No servers are available. Click this link to create a new server....**



NOTE

This link appears in **Servers** panel only when no server has been defined.

- Right-click in **Servers** panel to open the context menu, then select **New** → **Server**.
 - On the menu bar, select **File** → **New** → **Other** → **Server** → **Server**.
1. In the **Define a New Server** dialog, expand the **JBoss Fuse** node to expose the list of available server options:



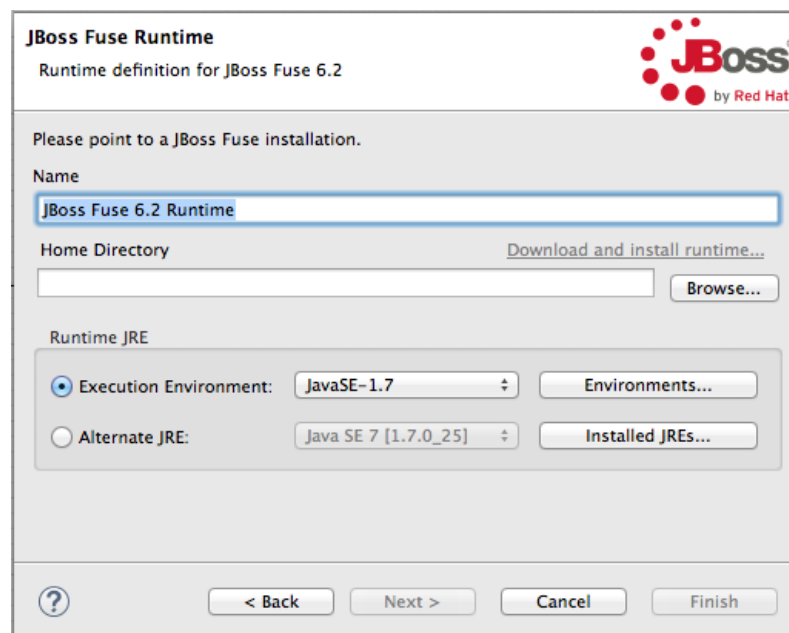
2. Click the server you want to add.
3. In **Server's host name**, accept the default (**localhost**).



NOTE

The address of **localhost** is **0.0.0.0**.

4. In **Server name**, accept the default, or enter a different name for the runtime server.
5. Click **Next >** to open the server's **Runtime definition** page:



**NOTE**

If the server is not already installed on your machine, you can install it now by clicking the link **Download and install runtime...** and following the site's download instructions. Depending on the site, you may be required to provide valid credentials before you can continue the download process.

6. Accept the default for **Name**.
7. In **Home Directory**, enter the path where the server runtime is installed, or click **Browse** to find and select it.
8. Select the runtime JRE from the drop-down menu next to **Execution Environment**.

If the version you want does not appear on the list, click the **Environments** button and select the version from that list. The JRE version you select must be installed on your machine.

**NOTE**

JBoss Fuse 6.2 requires either JRE versions 1.7 or 1.8.

9. Leave the **Alternate JRE** option as is.
10. Click **Next >** to save the server's runtime definition and open its **Configuration details** page:

11. Accept the default for **SSH Port (8101)**.

The runtime uses the SSH port to connect to the server's Karaf shell. If this default is incorrect for your setup, you can discover the correct port number by looking in the server's *installDir/etc/org.apache.karaf.shell.cfg* file.

12. In **User Name**, enter the name used to log into the server.

In the case of Red Hat JBoss Fuse, this is a user name stored in the Red Hat JBoss Fuse *installDir/etc/users.properties* file.



NOTE

If the default user has been activated (uncommented) in the `/etc/users.properties` file, the tooling autofills **User Name** and **Password** with the default user's name and password, as shown in [Step 10](#). The default user is already activated in JBoss Fuse 6.2, but not in JBoss Fuse 6.0 or 6.1.

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

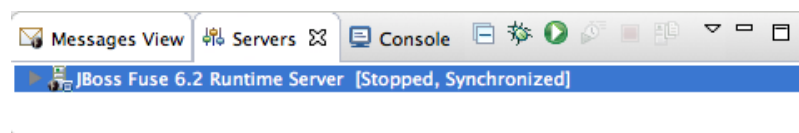
- **jaas:realms**—to list the realms
- **jaas:manage --index 1**—to edit the first (server) realm
- **jaas:useradd <username> <password>**—to add a user and associated password
- **jaas:roleadd <username> Administrator**—to specify the new user's role
- **jaas:update**—to update the realm with the new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

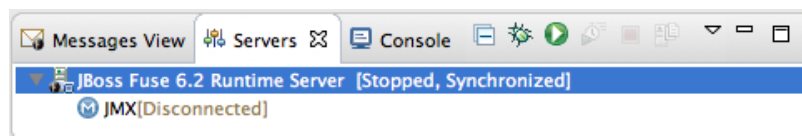
13. In **Password**, enter the password required for **User Name** to log into the server.

14. Click **Finish**.

The server runtime will appear in **Servers** panel. For example:



Expanding the server node exposes the server's JMX node:



Related topics

[Define a New Server](#)

[New Server Configuration](#)

[Add and Remove](#)


27.2. STARTING A SERVER

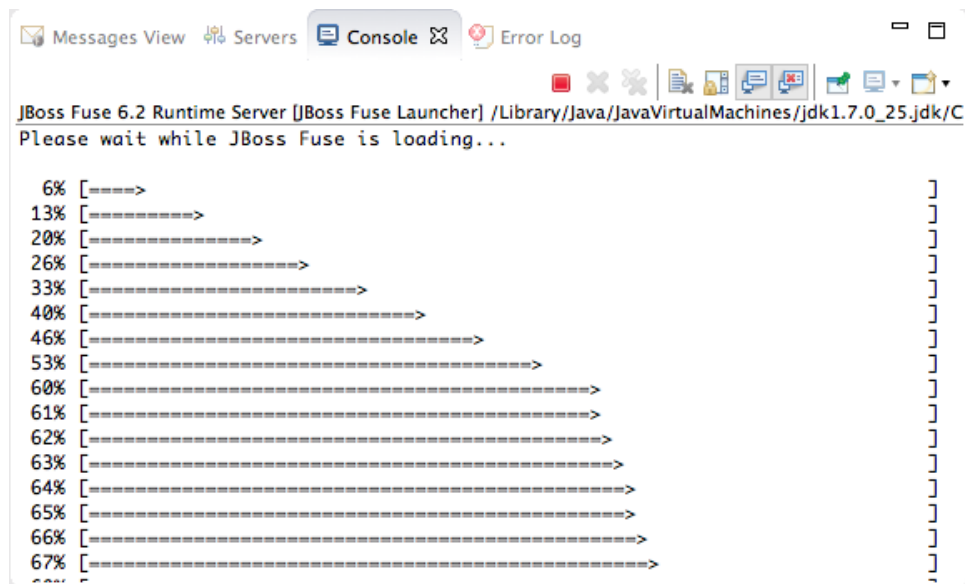
Overview

When you start a configured server, the tooling opens the server's remote management console in a **Shell** panel. This allows you to easily manage the container while testing your application.

Procedure

To start a server:

1. In the **Servers** panel, select the server you want to start.
 2. Click  .
- **Console** view opens and displays a message asking you to wait while the container is starting; for example:



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 27.1, “Adding a Server”](#).

- After the container has started up, **Shell** view opens to display the container's management console; for example:

```

SSH: (admin@localhost8159 - CONNECTED) - Encoding: (ISO-8859-1)

JBoss Fuse (6.2.0.redhat-099)
http://www.redhat.com/products/jbossenterprisemiddleware/fuse/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.

Open a browser to http://localhost:8181 to access the management console.

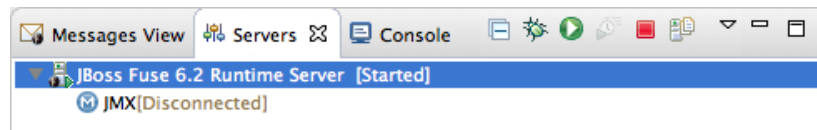
Create a new Fabric via 'fabric:create'
or join an existing Fabric via 'fabric:join [someUrls]'

Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown JBoss Fuse.

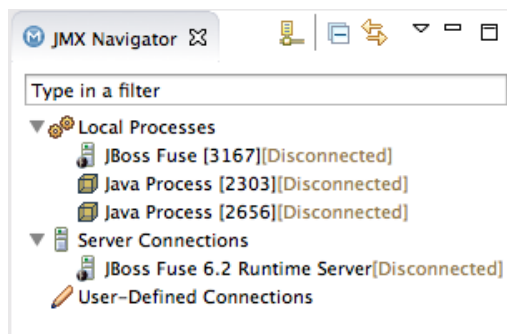
JBossFuse:admin@root>

```

- The running server appears in **Servers** panel:



- The running server also appears in **JMX Navigator** under **Server Connections**:



27.3. CONNECTING TO A RUNNING SERVER

Overview

After you start up a configured server, it appears in **Servers** panel and in **JMX Navigator**, under the **ServerConnections** node. You may need to expand the **Server Connections** node to see the server.

To publish and test your Fuse project application on the running server, you must first connect to it. You can connect to a running server either in **Servers** panel or in **JMX Navigator**.

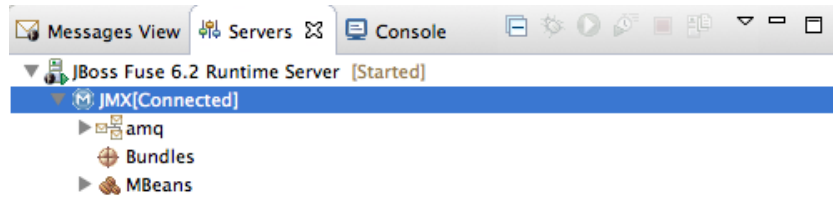


NOTE

Servers panel and **JMX Navigator** are synchronized with regards to server connections, so connecting to a server in **Servers** panel also connects it in **JMX Navigator**, and vice versa.

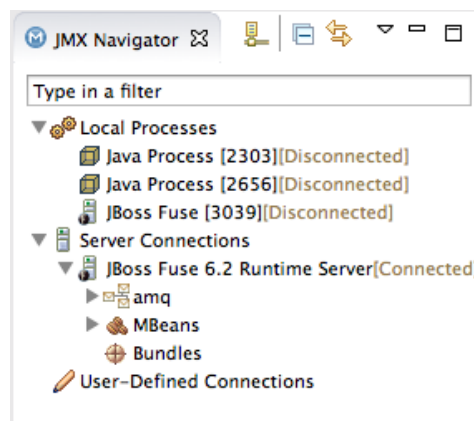
Connecting to a running server in Servers panel

1. In **Servers** panel, expand the server runtime to expose its **JMX[Disconnected]** node.
2. Double-click the **JMX[Disconnected]** node:



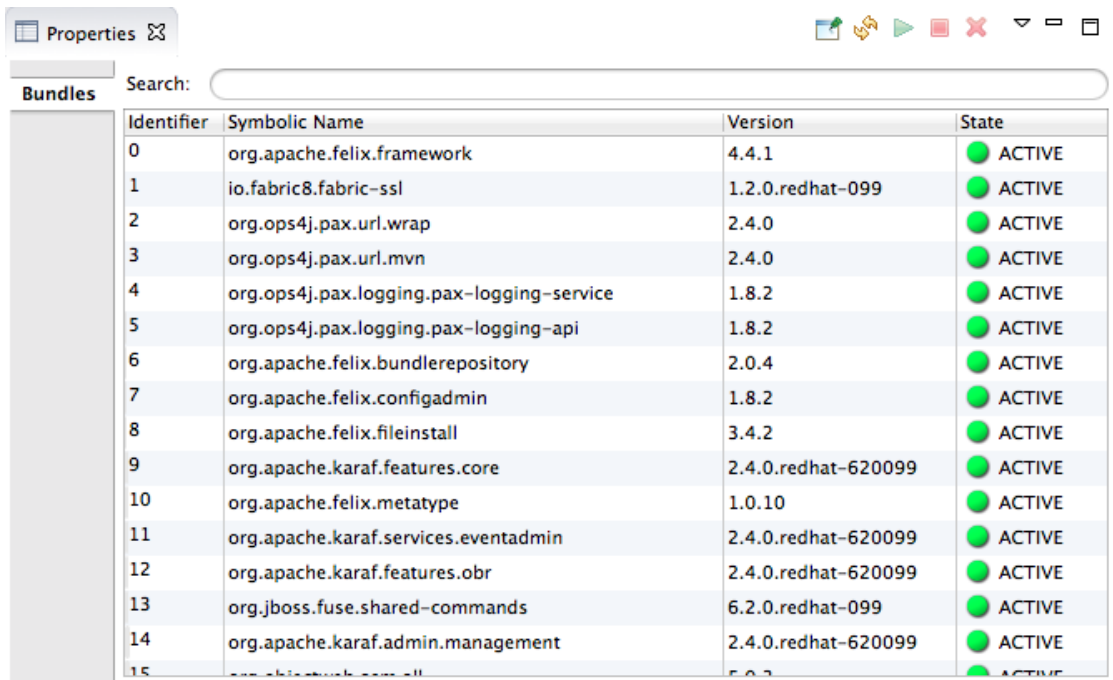
Connecting to a running server in JMX Navigator

1. In **JMX Navigator**, select the server to which you want to connect under the **Server Connections** node.
2. Double-click it:



Viewing bundles installed on the connected server

1. In either the **Servers** panel or **JMX Navigator**, expand the server runtime tree to expose the **Bundles** node, and select it.
2. The tooling populates **Properties** view with a list of bundles that are installed on the server:



Identifier	Symbolic Name	Version	State
0	org.apache.felix.framework	4.4.1	ACTIVE
1	io.fabric8.fabric-ssl	1.2.0.redhat-099	ACTIVE
2	org.ops4j.pax.url.wrap	2.4.0	ACTIVE
3	org.ops4j.pax.url.mvn	2.4.0	ACTIVE
4	org.ops4j.pax.logging.pax-logging-service	1.8.2	ACTIVE
5	org.ops4j.pax.logging.pax-logging-api	1.8.2	ACTIVE
6	org.apache.felix.bundlerepository	2.0.4	ACTIVE
7	org.apache.felix.configadmin	1.8.2	ACTIVE
8	org.apache.felix.fileinstall	3.4.2	ACTIVE
9	org.apache.karaf.features.core	2.4.0.redhat-620099	ACTIVE
10	org.apache.felix.metatype	1.0.10	ACTIVE
11	org.apache.karaf.services.eventadmin	2.4.0.redhat-620099	ACTIVE
12	org.apache.karaf.features.obr	2.4.0.redhat-620099	ACTIVE
13	org.jboss.fuse.shared-commands	6.2.0.redhat-099	ACTIVE
14	org.apache.karaf.admin.management	2.4.0.redhat-620099	ACTIVE
15	org.apache.karaf.features.api	2.4.0	ACTIVE

Using **Properties** view's **Search** tool, you can search for bundles by their **Symbolic Name** or by their **Identifier**, if you know it. As you type in the symbolic name or the identifier, the list updates, showing only the bundles that match the current search string.



NOTE

The **Symbolic Name** is the artifactId that you gave the project when you first created it. You can find the artifactId in the project's `pom.xml` file.

27.4. DISCONNECTING FROM A SERVER

Overview

When you are done testing your application, you can disconnect from the server without stopping it.

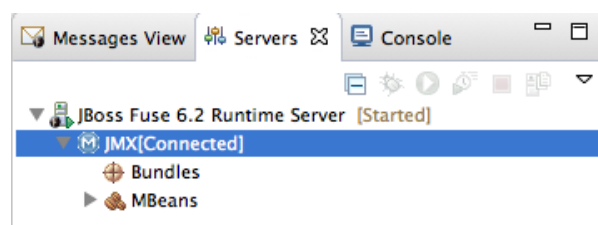


NOTE

Servers panel and **JMX Navigator** are synchronized with regards to server connections, so disconnecting from a server in **Servers** panel also disconnects it in **JMX Navigator**, and vice versa.

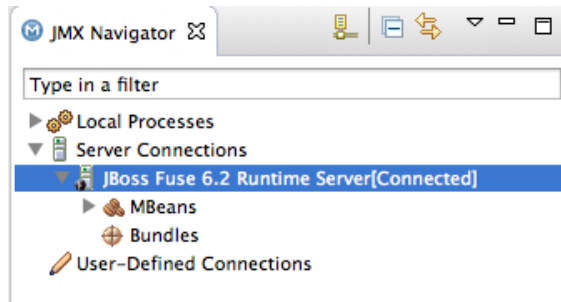
Disconnecting from a server in Servers panel

1. In **Servers** panel, expand the server runtime to expose its **JMX[Connected]** node.
2. Right-click the **JMX[Connected]** node to open the context menu, and then select **Disconnect**.



Disconnecting from a server in JMX Navigator

1. In **JMX Navigator**, select the server from which you want to disconnect.
2. Right-click it to open the context menu, and then select **Disconnect**.



27.5. STOPPING A SERVER


Overview

You can shut down a server in one of two ways:

- from **Servers** panel
- from the server's remote console in **Shell** view

Using Servers panel

To stop a server:

1. In **Servers** panel, 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**.

27.6. DELETING A SERVER

Overview

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

Deleting a server

1. In **Servers** panel, right-click the server you want to delete to open the context menu.

2. Select **Delete**.
3. Click **OK**.

Deleting the server's configuration

1. On the menu bar, click **JBoss Developer Studio** → **Preferences** → **Server**.



NOTE

On Linux and Windows machines, you access **Preferences** through the **Window** menu.

2. Expand the **Server** folder, and then select **Runtime Environments** to open the **Server Runtime Environments** page.
3. From the list, select the runtime environment of the server that you previously deleted from **Servers** panel, and then click **Remove**.
4. Click **OK**.

CHAPTER 28. PUBLISHING FUSE PROJECTS TO A SERVER

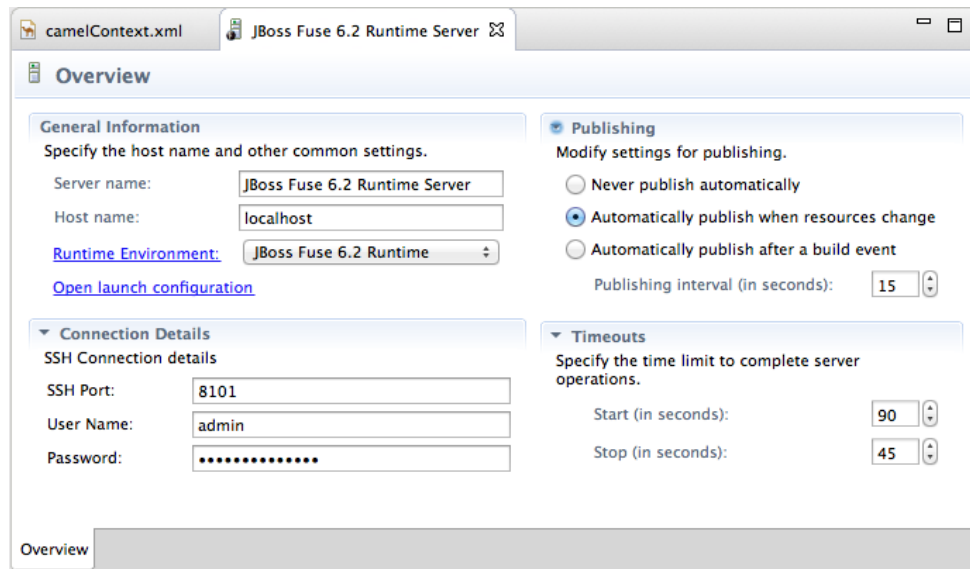
Abstract

You deploy Fuse projects into a server runtime using the Eclipse publishing mechanism. To do so, you must have defined and added the server to the **Servers** panel in **Fuse Integration** perspective (for details, see [Chapter 27, Managing servers](#). For a step-by-step demonstration, see [Red Hat JBoss Fuse Tooling Tutorials, To Publish a Fuse Project to Red Hat JBoss Fuse](#)

OVERVIEW

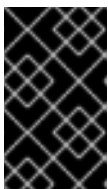
You can set up supported servers to publish assigned Fuse projects automatically or to publish them only when you manually invoke the publish command.

Each server runtime added to the **Servers** panel has its own **Overview** page that contains its configuration, connection, and publishing details:



You may need to expand **Publishing** to expose the server runtime publishing options and default settings:

- **Never publish automatically**—You must select this option to manually publish projects.



IMPORTANT

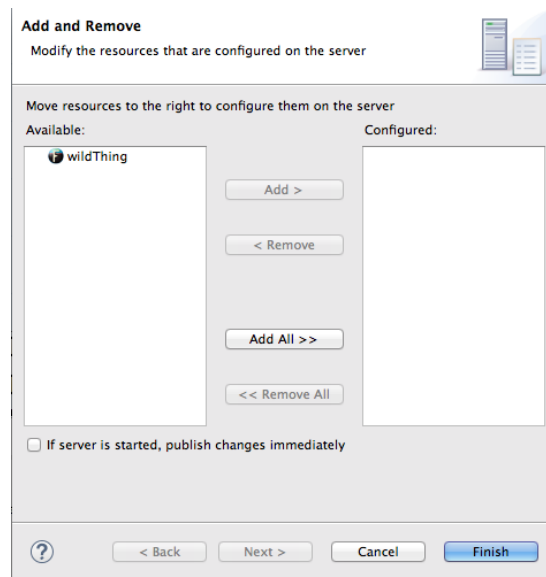
You must also disable the **If server started, publish changes immediately** option on the server's **Add and Remove** page (for details see, [the section called “Publishing Fuse projects manually”](#)).

- **Automatically publish when resources change**—[default] Enable this option to automatically publish or republish a Fuse project when you save changes made to it. How quickly projects are published depends on the **Publishing interval** (default is 15 seconds).
- **Automatically publish after a build event**—For Fuse projects, works the same as **Automatically publish when resources change**.

PUBLISHING FUSE PROJECTS AUTOMATICALLY WHEN RESOURCES CHANGE

The default publishing option for server runtimes is **Automatically publish when resources change**.

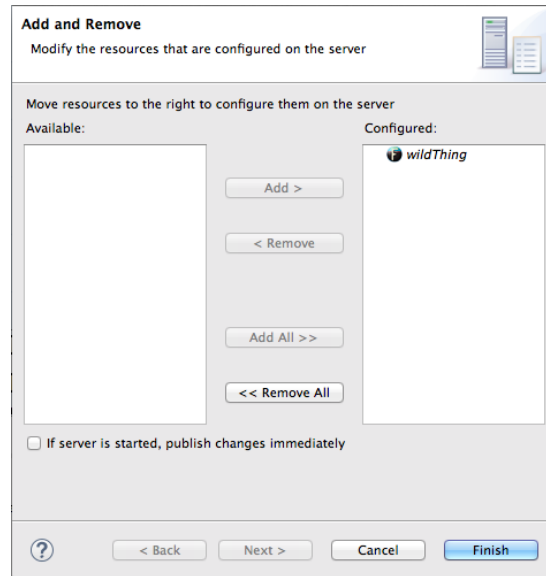
1. If necessary, start up the server runtime to which you want to publish a Fuse project. For details, see [Section 27.2, “Starting a Server”](#).
2. In the **Servers** panel, double-click the server runtime to open its **Overview** page.
3. Expand **Publishing**, and then select **Automatically publish when resources change**.
4. To increase or decrease the interval between publishing cycles, click the radio button next to **Publishing interval (in seconds)** up or down, as appropriate.
5. In the **Servers** panel, right-click the server runtime to open the context menu, and then select **Add and Remove**.



All resources available for publishing appear in the **Available** column.

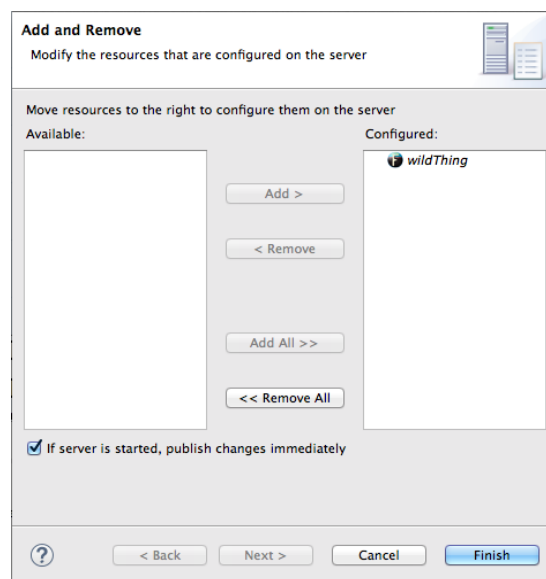
6. To assign a resource (in this case, the **wildThing** Fuse project) to the server runtime:
 - double-click it, or
 - select it, and click **Add >**.

The selected resource moves to the **Configured** column:



At this stage, the time at which the assigned resource would actually be published would depend on whether the server runtime was running and on the **Publishing interval** setting. However, if the server was stopped, you would have to manually publish the project after you started the server (for details, see [the section called “Publishing Fuse projects manually”](#)).

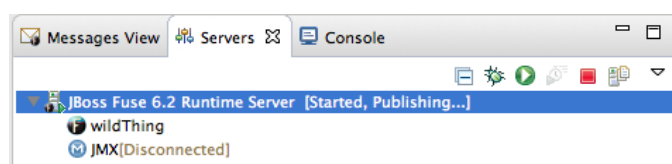
7. Click the **If server started, publish changes immediately** option to enable it:



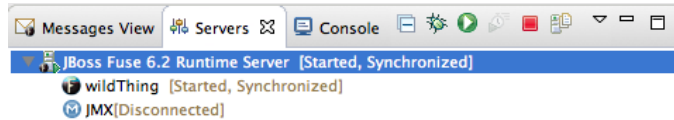
This option ensures that the configured project is published immediately once you click **Finish**. The **Automatically publish when resources change** option on the server runtime **Overview** page ensures that the configured project is republished whenever changes made to the local project are saved.

8. Click **Finish**.

The project appears in the **Servers** panel under the server runtime node, and the server runtime status reports [**Started, Publishing...**]:



When publishing is done, the status of both the server runtime and the project report **[Started, Synchronized]**:



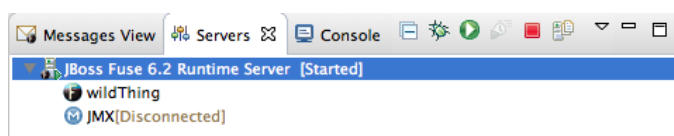
NOTE

For a server runtime, **Synchronized** means that all published resources on the server are identical to their local counterparts. For a published resource, **Synchronized** means that it is identical to its local counterpart.

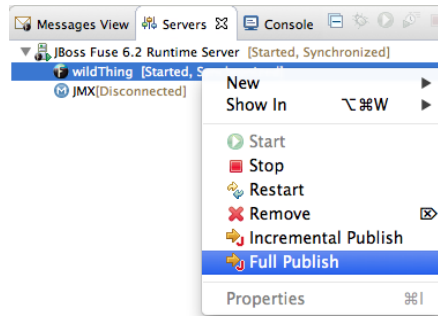
PUBLISHING FUSE PROJECTS MANUALLY

1. If necessary, start up the server runtime to which you want to publish a Fuse project. For details, see [Section 27.2, “Starting a Server”](#).
2. In the **Servers** panel, double-click the server runtime to open its **Overview** page.
3. Expand **Publishing**, and then select **Never publish automatically**.
4. If the Fuse project has already been assigned to the server runtime, make sure the **If server started, publish changes immediately** option is disabled.
 - a. In the **Servers** panel, right-click the server runtime to open the context menu.
 - b. Click **Add and Remove...** to open the server's **Add and Remove** page.
 - c. If the **If server started, publish changes immediately** option is enabled, disable it.
 - d. Skip to [Step 6](#).
5. If the Fuse project has not been assigned to the server runtime, assign it now.
 - a. Follow [Step 5](#) through [Step 6](#) in the section called “[Publishing Fuse projects automatically when resources change](#)”.
 - b. Do not enable the **If server started, publish changes immediately** option.
6. Click **Finish**.

The project appears in the **Servers** panel under the server runtime node, and the server runtime status reports **[Started]**:

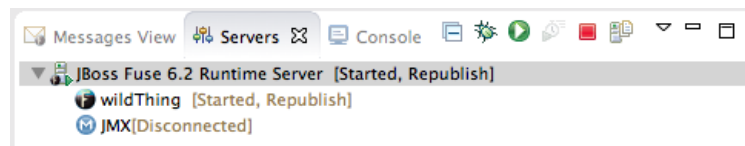


7. In the **Servers** panel, right-click the project's node (in this case, the Fuse project **wildThing**) to open the context menu:

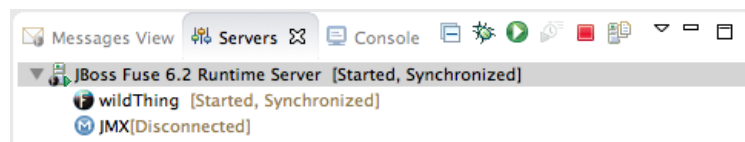


8. Select **Full Publish**.

During the publishing operation, the status of both the server runtime and the project report [**Started, Republish**]:



When publishing is done, the status of both the server runtime and the project report [**Started, Synchronized**]:



NOTE

The tooling does not support the **Incremental Publish** option. Clicking **Incremental Publish** results in a full publish.

VERIFYING THE PROJECT WAS PUBLISHED TO THE SERVER

After you have published a Fuse project to a server runtime, you can connect to the server and check that the project's bundle was installed on it.

1. Connect to the server runtime. For details see [the section called “Connecting to a running server in Servers panel”](#)
2. In the **Servers** panel, expand the server runtime tree to expose the **Bundles** node and select it.

The tooling populates **Properties** view with a list of bundles that are installed on the server:

 A screenshot of the 'Properties' view in the IDE, showing a table of installed bundles. The table has columns for Identifier, Symbolic Name, Version, and State.

Identifier	Symbolic Name	Version	State
259	io.fabric8.support.support-webapp	1.2.0.redhat-099	ACTIVE
260	io.hawt.hawtio-plugin-redhat-access	1.4.0.redhat-099	ACTIVE
261	io.fabric8.support.support-fabric8	1.2.0.redhat-099	ACTIVE
262	io.hawt.hawtio-redhat-fuse-branding	1.4.0.redhat-099	ACTIVE
263	javax.ws.rs.jsr311-api	1.1.1	ACTIVE
264	auth	0.0.0	ACTIVE
266	camel-blueprint	1.0.0.SNAPSHOT	ACTIVE

3. To find your project's bundle, either scroll down to the bottom of the list, or start typing the bundle's symbolic name in **Properties** view's **Search** box.

The bundle's symbolic name is the artifactId you gave the project when you created it. You can find the artifactId at the very beginning of the project's **pom.xml** file. To open it in the tooling's XML editor, click **pom.xml** in **Project Explorer**, and then click the **pom.xml** tab in the XML editor.

RELATED TOPICS

[Red Hat JBoss Fuse Tooling Tutorials](#), To Publish a Fuse Project to Red Hat JBoss Fuse

PART VI. WORKING WITH FABRICS

A fabric is a distributed configuration, provisioning, and clustering mechanism for developing, testing, and running 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.

For in-depth information on Fabric8, see the Fabric Guide and Cloud Computing with Fabric on the [Red Hat Customer Portal](#)

To open **Fabric8** perspective ([Chapter 4, Fabric8 Perspective](#)), click  on the right side of the JBoss Developer Studio tool bar, and then select **Fabric8** from the list of available perspectives.



WARNING

The Fabric8 Tooling feature has now been deprecated. Red Hat support for this feature will be discontinued from JBoss Fuse version 6.2. However, this feature will be removed completely after JBoss Fuse 7.0 release.

CHAPTER 29. SETTING UP A FABRIC ENVIRONMENT

The tooling provides an environment for developing and debugging fabric applications. Use the tooling 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.



NOTE

With Red Hat JBoss Fuse and the JBoss Server Extensions feature installed, you can start up Red Hat JBoss Fuse and create a fabric from inside the tooling. See [Chapter 27, *Managing servers*](#)

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 34, *Creating a Fabric in the Cloud*](#).

OPENING THE FABRIC PERSPECTIVE

To set up a fabric environment, you must switch to the Fabric8 perspective, by selecting **Window** → **Open Perspective** → **Other...** → **Fabric8**, or clicking  in the Perspectives tab, and selecting **Fabric8**.

RELATED TOPICS

[Chapter 30, *Specifying and Connecting to a Fabric*](#)

CHAPTER 30. SPECIFYING AND CONNECTING TO A FABRIC

Before you can view and edit your fabric, you must add it to the **Fabric Explorer** view. You can add multiple fabrics in Fabric8 perspective and connect to them concurrently.

30.1. ADDING FABRIC DETAILS

Overview

For the tooling 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 Explorer**, right-click **Fabrics** to open the context menu, and then select **Add Fabric details** to open the **Fabric Details** wizard, as shown in [Figure 30.1](#).

Figure 30.1. 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 **Jolokia URL**, and this name will appear in **Fabric Explorer**.

The default **Name** is Local Fabric.

3. In **Jolokia URL**, enter the url, in the form **http://hostname:port/jolokia/**, of the fabric to which you want to connect. This URL specifies the location of a *fabric registry agent*, whose default port is *8181*.

The default URL is `http://localhost:8181/jolokia/`.

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

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

You can 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 that was either specified for **User name** when the fabric was created or is defined in the Red Hat JBoss Fuse *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 either specified when the fabric was created or is the password of the first user defined in the Red Hat JBoss Fuse *installDir/etc/users.properties* file.

You can discover the Zookeeper password by issuing the command

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

7. Click **OK**.

The fabric's name appears in **Fabric Explorer** 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 30.2, "Connecting to a fabric"](#)

Related topics

[Section 30.5, "Deleting a fabric's details"](#)

30.2. CONNECTING TO A FABRIC


Overview


The tooling connects 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 Explorer**, 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 Explorer**.

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 30.1, “Adding fabric details”](#)

30.3. 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 tooling or the Red Hat JBoss Fuse console command line.

Procedure

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

The tooling closes the connection with the selected fabric.

30.4. 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 tooling.



IMPORTANT

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

Procedure

1. In **Fabric Explorer**, 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 30.1, “Fabric Details wizard”](#).
3. In **Name**, enter a new name for the fabric, if necessary.
4. In **Jolokia URL**, 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**.

30.5. DELETING A FABRIC'S DETAILS

Overview

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

Procedure

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

The tooling closes the connection with the selected fabric, and then deletes the fabric's details.

Related topics

[Section 30.1, "Adding fabric details"](#)

CHAPTER 31. WORKING WITH FABRIC CONTAINERS

31.1. CREATING A NEW CHILD CONTAINER

Overview

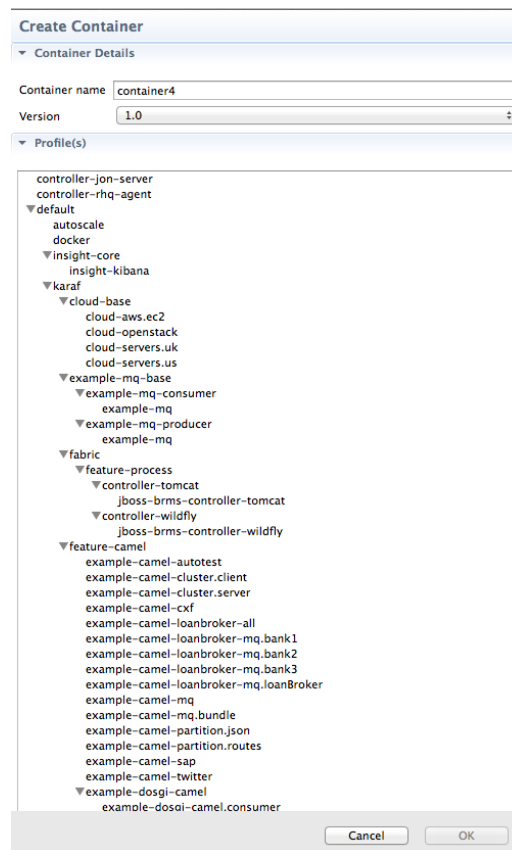
A child container is created on the server where its parent is.

Procedure

To create a new child container:

1. If necessary, in **Fabric Explorer**, 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 31.1. Create Child Container wizard

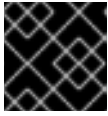


3. In **Container name**, enter a name for the new child container.

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 Explorer** as a node under **Containers**



NOTE

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

Related topics

[Section 32.1, “Creating a new profile”](#)

[Section 33.1, “Creating a new version of a profile”](#)

31.2. 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 Explorer**, 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 31.2. Create Container via SSH wizard

Create Container 3 errors detected

▼ Container Details

Container name

Version

Host

User name

Password

Path

Port

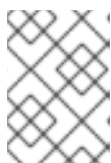
SSH retries

Retry delay

▼ Profile(s)

- controller-jon-server
- controller-rhq-agent
- ▼ default
 - autoscale
 - docker
 - example-quickstarts-jms
 - ▼ insight-core
 - insight-kibana
 - ▼ karaf
 - ▼ cloud-base
 - cloud-aws.ec2
 - cloud-openstack
 - cloud-servers.uk
 - cloud-servers.us
 - ▼ example-mq-base
 - ▼ example-mq-consumer
 - example-mq
 - ▼ example-mq-producer
 - example-mq
 - ▼ fabric
 - ▼ feature-process
 - ▼ controller-tomcat
 - jboss-brms-controller-tomcat
 - ▼ controller-wildfly
 - jboss-brms-controller-wildfly
 - ▼ feature-camel
 - example-camel-cluster.client
 - example-camel-cluster.server
 - example-camel-cxf
 - example-camel-loanbroker
 - example-camel-mq
 - example-camel-mq.bundle

Cancel OK

**NOTE**

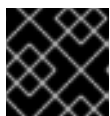
An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

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 password for **User name**.
8. In **Path**, accept the default (`~/containers/`), or 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 (`1`), or enter the maximum number of retries to attempt at establishing an ssh connection on the remote host.
11. In **Retry delay**, accept the default (`1`), or 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 Explorer** as a container node under **Containers**

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

Related topics

[Section 32.1, "Creating a new profile"](#)

[Section 33.1, "Creating a new version of a profile"](#)

31.3. CREATING A NEW CONTAINER ON A CLOUD

Overview

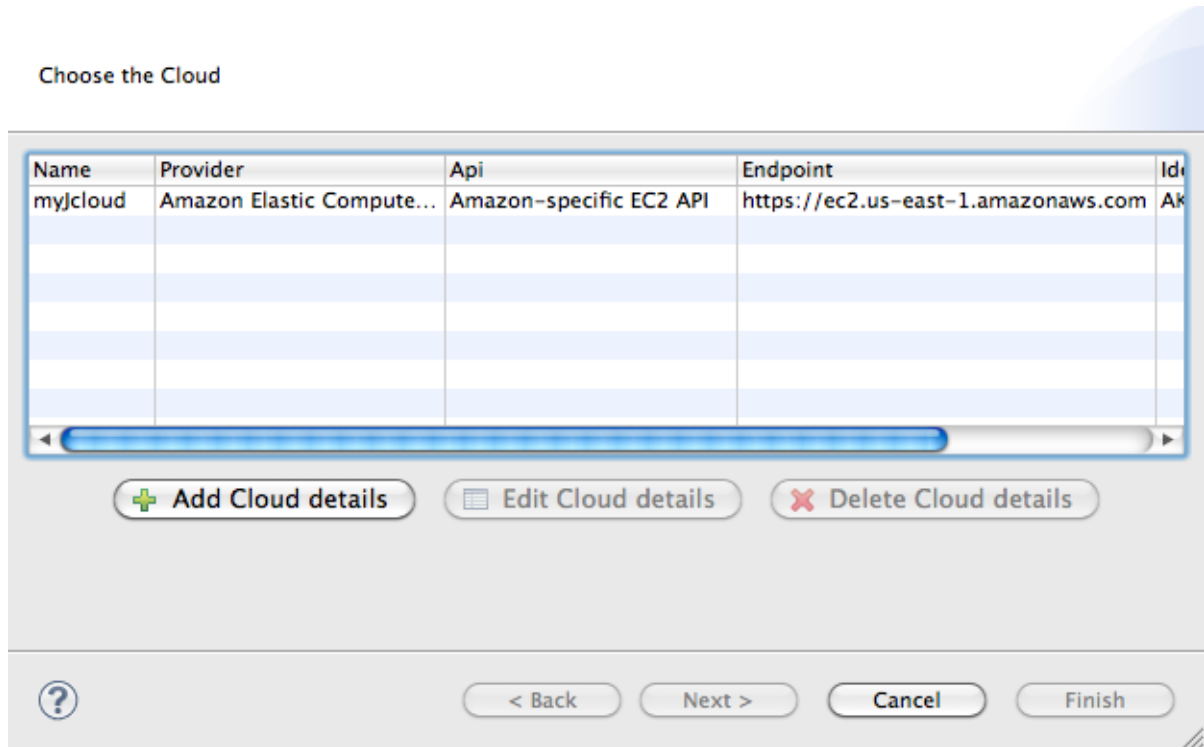
To create a container on a cloud, you need to specify the cloud's connection details and then create the container.

Procedure

To create a new container on a cloud:

1. In **Fabric Explorer**, if necessary, expand the tree of the fabric for which you want to create the new container on the cloud. Right-click **Containers** to open the context menu.
2. Select **Create a new container on the cloud** to open the **Create Container in a Cloud** wizard.

Figure 31.3. Create Container in a Cloud/Choose the Cloud page



NOTE

Unless the connection details for a cloud have already been added to the tooling, 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 34.1, "Adding cloud details"](#).

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

Figure 31.4. New Containers Details page

New Containers Details

✖ Create a container in the cloud... 3 errors detected

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

Container name:

Version:

Group:

User:

Password:

Zookeeper Password:

Location ID:

Hardware ID:

OS Family:

OS Version:

Image ID:


Clear Defaults

▼ Profile(s)

- controller-jon-server
- controller-rhq-agent
- ▼ default
 - autoscale
 - docker
 - example-quickstarts-jms
 - ▼ insight-core
 - insight-kibana
 - ▼ karaf
 - ▼ cloud-base
 - cloud-aws.ec2
 - cloud-openstack
 - cloud-servers.uk
 - cloud-servers.us
 - ▼ example-mq-base
 - ▼ example-mq-consumer

? < 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**, accept the default (*fabric*), or enter the group in which to start up the new container.
7. In **User**, accept the default (*admin*), or enter the name of another user who is authorized to log into the new cloud container.

This is the user that was either specified when the fabric was created or is defined in the Red Hat JBoss Fuse `installDir/etc/users.properties` file (see [Section 30.1](#), “Adding fabric details”).

8. In **Password**, enter the password required for **User** to log into the new cloud container.

This is the password that was specified for **User** when the fabric was created or is defined for the user in the Red Hat JBoss Fuse `installDir/etc/users.properties` file.

9. In **Zookeeper Password**, accept the default generated password for logging into the cloud container's zookeeper registry, or enter the zookeeper password that was used when the fabric was created.

You can discover the zookeeper password by issuing the command `JBossFuse:karaf@root>fabric:ensemble-password`.

10. 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.

11. 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**.

12. In **OS Version**, optionally select from the drop-down list the version for the operating system you selected.

13. 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 tooling starts an instance of the image in the cloud, and then installs a fabric agent and provisions the new agent with the selected profiles. Depending on the cloud provider, instantiation and provisioning could take some time.

14. 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.

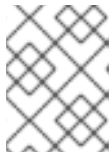


IMPORTANT

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

15. Click **OK**.

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



NOTE

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

Related topics

[Chapter 34, *Creating a Fabric in the Cloud*](#)

[Section 32.1, “Creating a new profile”](#)

[Section 33.1, “Creating a new version of a profile”](#)

31.4. 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 Explorer**, 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 tooling started the container, click **Containers** in **Fabric Explorer**, and open the **Properties** view. You should see something similar to [Figure 31.5](#), where **mycamelExample** was started and provisioned successfully.

Figure 31.5. 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 31.1](#):

Table 31.1. Container startup status

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

Related topics

[Section 31.6, “Stopping a container”](#)

31.5. 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.



IMPORTANT

The profile or profiles you select will override the entire list of profiles currently deployed in the selected container.

Procedure

To change the profiles deployed to a container:

1. Select the container.
2. In the **Properties** view, select the **Profiles** tab.
3. From the list, select the profile or profiles you want to deploy to the container by highlighting them.



NOTE

To select multiple profiles, hold down the **CTRL** key while you select them.

31.6. 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 Explorer**, 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 tooling stopped the container, click **Containers** in **Fabric Explorer**, and open the **Properties** view. You should see something similar to [Figure 31.6](#), where **myRoute** was stopped successfully.

Figure 31.6. 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 31.1](#), “Container startup status”.

Related topics

[Section 31.4, “Starting a container”](#)

[Section 31.7, “Deleting a container”](#)

31.7. 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:


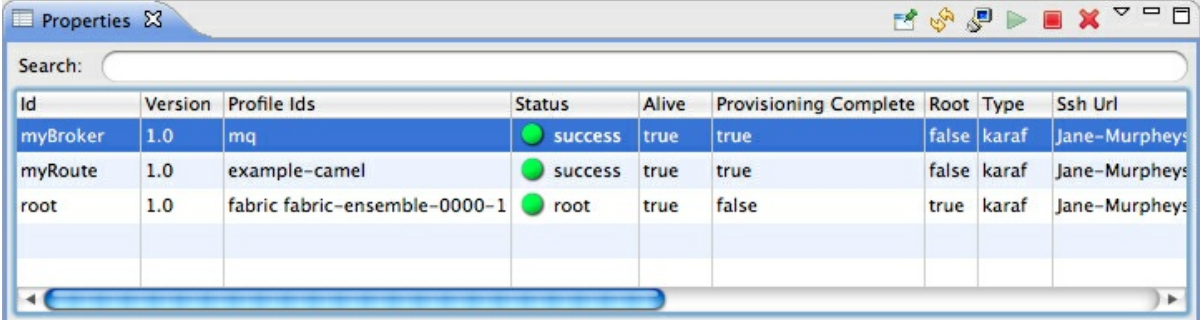



1. If necessary, in **Fabric Explorer**, expand the tree of the fabric whose container you want to delete.
2. Click **Containers** to populate **Properties** view with a list of the fabric's containers.
3. In **Properties** view, select the container you want to delete, and then click  on the right-hand side of the toolbar.

Figure 31.7. Container selected for deletion in Properties view

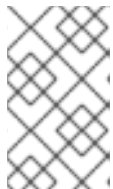


Id	Version	Profile Ids	Status	Alive	Provisioning Complete	Root	Type	Ssh Url
myBroker	1.0	mq	 success	true	true	false	karaf	Jane-Murphey
myRoute	1.0	example-camel	 success	true	true	false	karaf	Jane-Murphey
root	1.0	fabric fabric-ensemble-0000-1	 root	true	false	true	karaf	Jane-Murphey

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

The tooling displays 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.

5. In **Fabric Explorer**, 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 31.6, “Stopping a container”](#)

[Section 31.1, “Creating a new child container”](#)

[Section 31.2, “Creating a container on a remote host”](#)

[Section 31.3, “Creating a new container on a cloud”](#)

CHAPTER 32. 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 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.

32.1. CREATING A NEW PROFILE

Overview

The tooling provides 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

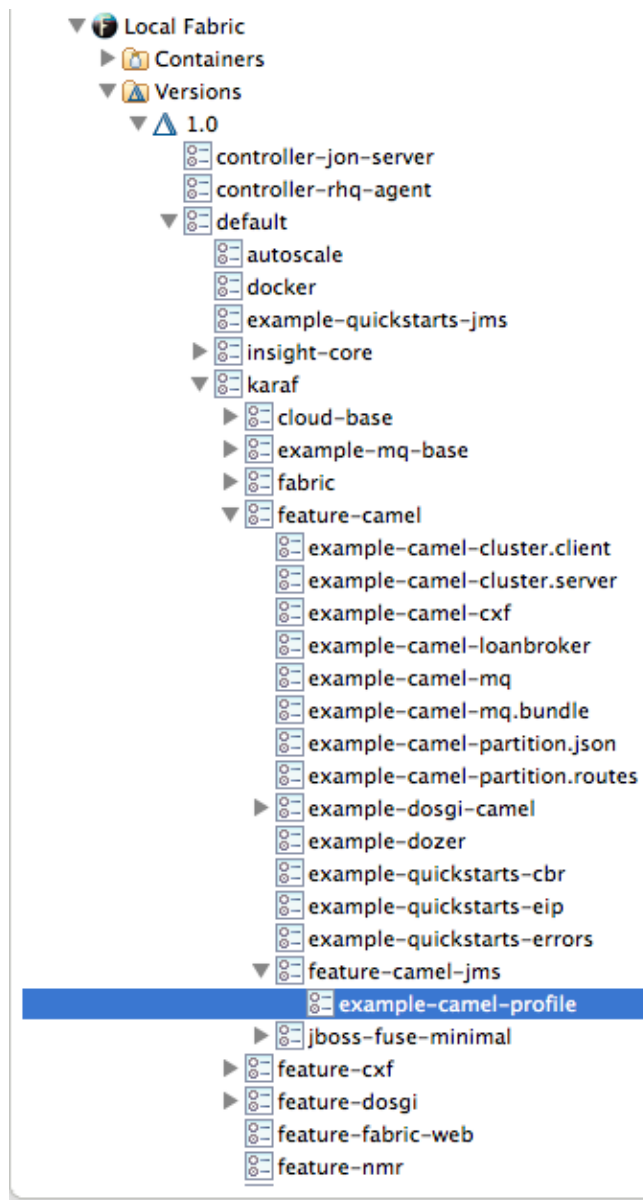
This procedure creates a brand new profile. If you want to create a new version of an existing profile to incrementally test changes to deployed containers, see [Chapter 33, Working with Versions](#)

Procedure

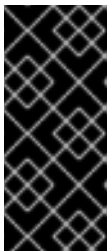
To create a new profile:

1. In **Fabric Explorer**, 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 32.1](#).

Figure 32.1. 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 selected profile is the parent of the new profile, the new profile inherits the selected profile's configurations, properties, features, repositories and bundles as well as those of the selected profile's ancestors. However, a child's configurations override an ancestor's when their configuration files have the same name.

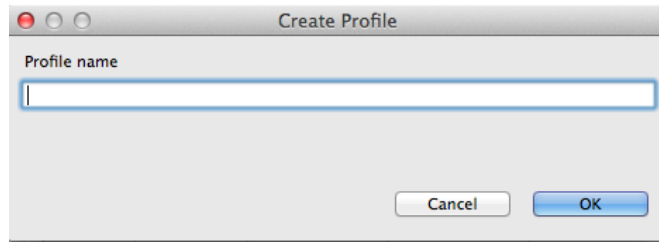


NOTE

Neither configuration files nor properties are visible within the tooling, but you can view them in the Red Hat JBoss Fuse Management Console. For details, see the Red Hat JBoss Fuse Management Console documentation at https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/.

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

Figure 32.2. Create Profile dialog



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

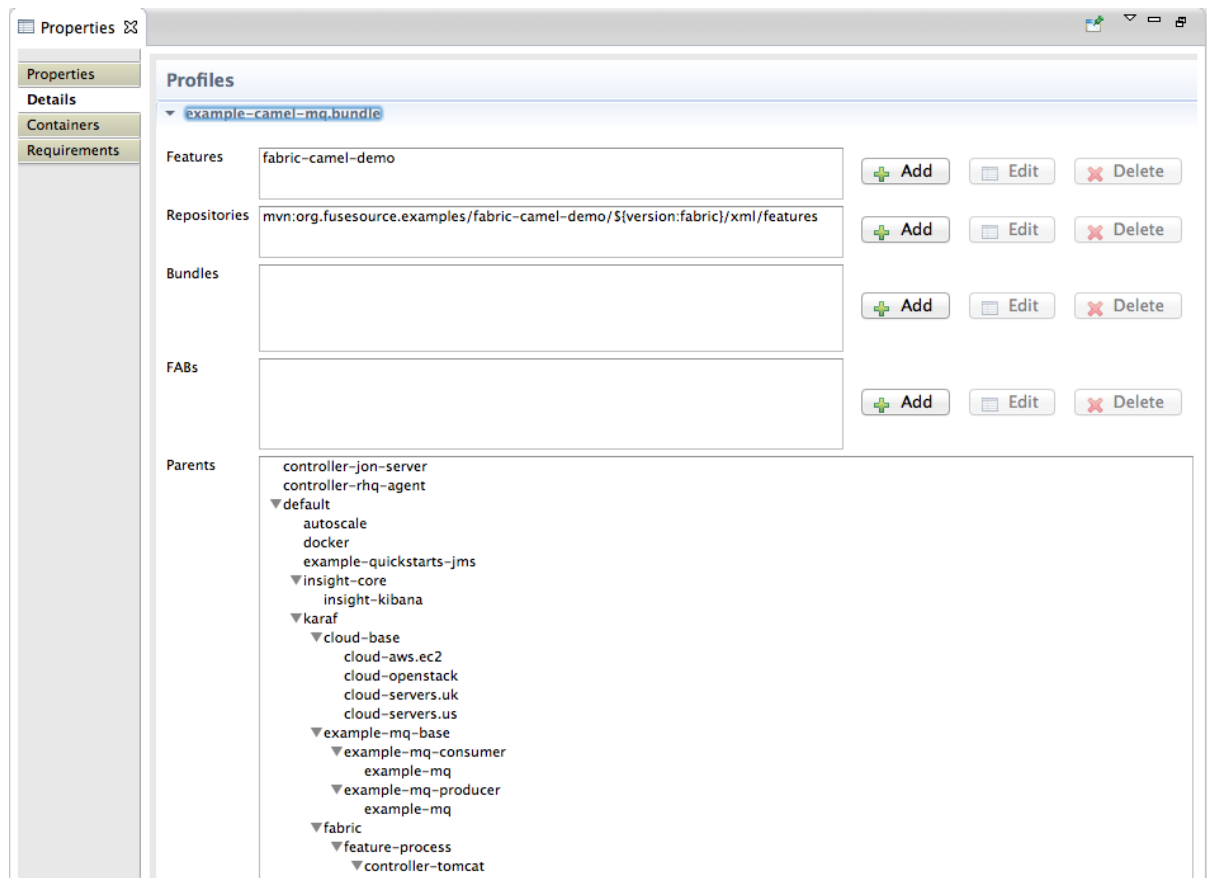
Use a name that is descriptive of the new profile's function.

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

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

7. In **Fabric Explorer**, 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 32.3. 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 [Deploying](#)

Features.

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.

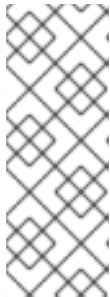


NOTE

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 Explorer**. 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 is provisioned, your project is wiped out.

You can now assign the new profile to a container.

10. In **Fabric Explorer**, 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 33.1, "Creating a new version of a profile"](#)

32.2. 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 Explorer**, 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 33. 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.

33.1. 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 Explorer**, 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 32.1](#), “Creating a new profile”.
3. Select **Versions**, 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 tooling sorts 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 tooling copies all of the base version's profiles into the new version and displays the new version in **Fabric Explorer**, directly under the base version.

Modifying profiles in the new version

To modify a profile in the new version:

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

The tooling populates 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.

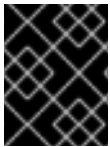
Related topics

[Section 32.1, "Creating a new profile"](#)

33.2. 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 Explorer**, 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 34. 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 tooling 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

34.1. ADDING CLOUD DETAILS

Overview

To connect to a cloud, the tooling needs 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 Explorer**, right-click **Clouds** to open the context menu, and then select **Add Cloud details**.

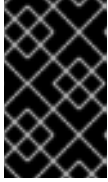
Figure 34.1. Cloud Details wizard



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 tooling autofills **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 tooling connects to the cloud whose access details you specified and loads the provider's images. Once that's done, the name you entered in [Step 2](#) appears as a node under **Clouds** in **Fabric Explorer**.

Related topics

[Section 31.3, "Creating a new container on a cloud"](#)

34.2. SPECIFYING FABRIC DETAILS

After you've added the access information that the tooling needs 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 Explorer**, 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 34.2. Fabric Details page



NOTE

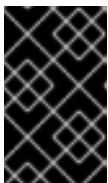
An error icon () marks required fields that lack a valid value, and the dialog banner displays the number of errors detected.

You may have to wait a few seconds while the tooling retrieves 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 user name, or enter a different user name to use for logging into the new fabric registry container.
7. In **Password**, enter the password required for **User** to log into the specified fabric.
8. In **Zookeeper Password**, accept the default, generated password for logging into the cloud fabric's zookeeper registry, or enter a different password.
9. In **Location ID**, select from the drop-down list the ID of the region in which the machines you want to use are located.
10. 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.

11. 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**.

12. In **OS Version**, optionally select from the drop-down list the version for the operating system you selected.
13. 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.
14. In **Maven proxy URI**, accept the default URI, or enter the URI to the Maven proxy repository you want to use.
15. Click **Finish**.

The tooling creates the container, then provisions 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 Explorer** 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 34.1, "Adding cloud details"](#)

[Section 31.3, "Creating a new container on a cloud"](#)

APPENDIX A. WIZARD FIELD REFERENCES

NAME

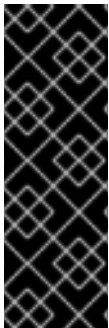
New Camel XML File — Creates a new XML file for editing routes with the route editor

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 or to be added to into existing camelContexts .



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 6, 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
Project Name		Specifies the name of the new project.
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 5, 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 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.

Name	Default	Description
Artifact ID		Specifies the artifact ID Maven uses when generating the new project. This ID is mapped to the POM file's artifactId element.
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 5, 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
Parent Folder	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 22.1, “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 8, 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
------	---------	-------------

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

RELATED TOPICS

[Chapter 8, *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 27.1, "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, <code>0.0.0.0</code>	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.
Password		[Optional] Specifies the password the authorized user must enter to access the server remotely.

RELATED TOPICS

[Section 27.1, “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 the server is configured to use.

RELATED TOPICS

[Section 27.1, “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	<i>Local Fabric</i>	Specifies the name of the fabric to connect to.
Jolokia URL	<i>https://localhost:8181/jolokia</i>	Specifies the url, in the form of https://hostname:port/jolokia, 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 30.1, “Adding fabric details”](#)

[Section 30.4, “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.

Name	Default	Description
Profiles		Specifies the profile(s) to install in the new container.

RELATED TOPICS

[Section 31.1, “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>~/containers</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.

Name	Default	Description
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 31.2, “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	<i>fabric</i>	Specifies the name of the group in which to start up the new container.
User	<i>admin</i>	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.

Name	Default	Description
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 31.3, “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 33.1, “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).
Owner		Specifies the name of the account owner.

RELATED TOPICS

[Section 34.1, “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	<i>Cloud Fabric</i>	Specifies the name of the new fabric.
Container name	<i>Registry</i>	Specifies the name of the fabric registry container.

Name	Default	Description
Group	<i>fabric</i>	Specifies the name of the group in which to start up the fabric registry container.
User	<i>admin</i>	Specifies the name of a user authorized to login to the fabric registry container.
Password		Specifies the user's password for logging into the the fabric.
Zookeeper Password	*****	Specifies the zookeeper password for logging into the cloud fabric's zookeeper registry.
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.
Maven proxy URI	<i>http://repo.fusesource.com/nexus/content/groups/ea</i>	Specifies the Maven proxy repository to use.

RELATED TOPICS

[Section 34.2, "Specifying fabric details"](#)

APPENDIX B. TOOLING PREFERENCES

NAME

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

PROPERTIES

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

Table B.1. Editor Properties

Name	Default	Description
Language	Simple	Specifies the default expression language used by the route editor.
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 7.10, "Configuring the route editor"](#)

NAME

Colors — Sets the default colors for diagrammatic components of Route editor.

PROPERTIES

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

Table B.2. Colors Properties

Name	Description
Route Editor Diagram Grid Color	Specifies the default color for the diagram grid.
Route Editor Diagram Connection Color	Specifies the default color for the connection in the diagram.

Name	Description
Route Editor Diagram Text Color	Specifies the default color for the text in the diagram.
Route Editor Diagram Figure Background	Specifies the default color of EIP node background.
Route Editor Diagram Figure Foreground	Specifies the default color of EIP node foreground.

RELATED TOPICS

[Section 7.10, “Configuring the route editor”](#)