



Red Hat JBoss Fuse Service Works 6.0

User Guide

This guide is for Red Hat JBoss Fuse Service Works users and developers.

Red Hat JBoss Fuse Service Works 6.0 User Guide

This guide is for Red Hat JBoss Fuse Service Works users and developers.

Red Hat Content Services

Legal Notice

Copyright © 2015 Red Hat, Inc..

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

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 teaches users how to utilize the Red Hat JBoss Fuse Service Works' graphical tools and web consoles.

Table of Contents

CHAPTER 1. RED HAT JBOSS FUSE SERVICE WORKS	4
1.1. WHAT IS RED HAT JBOSS FUSE SERVICE WORKS?	4
1.2. CORE FUNCTIONALITY	4
1.3. SYSTEM INTEGRATION	5
1.4. INTEGRATION USE CASE	5
1.5. CORE AND COMPONENTS	5
1.6. RED HAT JBOSS FUSE SERVICE WORKS FEATURES	6
1.7. COMPONENTS OF RED HAT JBOSS FUSE SERVICE WORKS	6
 CHAPTER 2. READ ME	 8
2.1. BACK UP YOUR DATA	8
2.2. RED HAT DOCUMENTATION SITE	8
 CHAPTER 3. JBOSS INTEGRATION AND SOA DEVELOPMENT	 9
3.1. JBOSS INTEGRATION AND SOA DEVELOPMENT	9
3.2. INSTALLING JBOSS DEVELOPER STUDIO INTEGRATION STACK	9
3.3. HELPFUL TIPS	11
3.4. RUNNING QUICKSTARTS FROM JBOSS DEVELOPER STUDIO	11
3.5. IMPORT PROJECTS FROM A GIT REPOSITORY IN JBOSS DEVELOPER STUDIO	12
3.6. SETTING A NEW RULES RUNTIME IN JBOSS DEVELOPER STUDIO	13
 CHAPTER 4. SETTING UP THE SERVER	 15
4.1. ADD JBOSS EAP SERVER	15
 CHAPTER 5. RULES	 17
5.1. DSL	17
5.2. DSL EDITOR	17
5.3. DSL EDITOR COMPONENTS	17
5.4. ADD A LANGUAGE MAPPING WIZARD	19
5.5. EDIT THE LANGUAGE MAPPING WIZARD	19
5.6. JBOSS RULES FLOW EDITOR	20
5.7. DIFFERENT TYPES OF CONTROL ELEMENTS IN FLOW PALETTE	20
5.8. DIFFERENT TYPES OF NODES IN FLOW PALETTE	21
5.9. RULE EDITOR	23
5.10. CONTENT ASSIST	23
5.11. CODE FOLDING	23
5.12. SYNCHRONIZATION WITH OUTLINE VIEW	23
5.13. RETE TREE VIEW	23
5.14. BREAKPOINTS	23
5.15. CREATING BREAKPOINTS	24
5.16. DEBUGGING	24
 CHAPTER 6. JBOSS DEVELOPER STUDIO BPEL PROJECTS	 26
6.1. BPEL	26
6.2. APACHE ODE	26
6.3. LOGGING INTO BPEL	26
6.4. CREATING A BPEL PROJECT	26
6.5. CREATING A BPEL PROCESS	29
6.6. CREATE A NEW SERVER RUNTIME	32
6.7. EDITING A BPEL PROCESS FILE	33
6.8. TABS SHOWN IN THE PROPERTIES VIEW	33
6.9. OBSERVING A BPEL PROCESS	34
6.10. ADDING A SERVICE TO A WSDL FILE	34

6.11. CREATING A DEPLOY.XML FILE	37
6.12. CREATING JBOSS BPEL SERVER	39
6.13. CREATING CORRELATION SETS	40
6.14. EXAMPLE BPEL COMPONENT IMPLEMENTATION	40
6.15. STRUCTURE OF A SWITCHYARD BPEL APPLICATION	41
6.16. WIZARDS	41
6.17. VIEWS	42
6.18. PROPERTY SECTION TABS	43
6.19. PROCESS PROPERTY SHEET TABS	44
6.20. DETAILS TAB OPTIONS	45
6.21. BPEL DESIGNER FEATURES	48
6.22. BPEL DESIGNER CONCEPTS	49
6.23. BPEL DEPLOYMENT DESCRIPTOR EDITOR PROPERTIES	50
6.24. DIALOGS	51
6.25. CHEAT SHEETS	52
6.26. CONTEXT MENU	53
CHAPTER 7. JBOSS DEVELOPER STUDIO GUVNOR TASKS	54
7.1. GUVNOR REPOSITORY EXPLORING PERSPECTIVE	54
7.2. CREATING A NEW GUVNOR CONNECTION	54
7.3. CONFIGURING THE GUVNOR CONNECTION WIZARD	55
7.4. GETTING LOCAL COPIES OF GUVNOR FILES	56
7.5. MANAGING GUVNOR RESOURCES	57
7.6. RESOURCE FROM GUVNOR WIZARD	58
7.7. GUVNOR REPOSITORIES VIEW	59
7.8. GUVNOR RESOURCE HISTORY VIEW	60
7.9. CONFIGURING THE GUVNOR RESOURCE HISTORY VIEW	60
7.10. GUVNOR PREFERENCES	60
7.11. GUVNOR REPOSITORY CONNECTION PREFERENCES	60
7.12. LOCAL GUVNOR REPOSITORY RESOURCE DECORATION PREFERENCES	61
APPENDIX A. REVISION HISTORY	62

CHAPTER 1. RED HAT JBOSS FUSE SERVICE WORKS

1.1. WHAT IS RED HAT JBOSS FUSE SERVICE WORKS?

Red Hat JBoss Fuse Service Works is a platform for developing enterprise application integration (EAI) and service-oriented architecture (SOA) solutions. It consists of a service component framework, business rules/complex event processing, life-cycle governance, runtime governance, and process automation. Red Hat JBoss Fuse Service Works is built on the same core as JBoss Fuse, and includes enterprise messaging, Apache Camel, and Apache CXF. Red Hat JBoss Fuse Service Works enables users to design, deploy, integrate, and orchestrate business services.

Red Hat JBoss Fuse Service Works can be used to integrate your major business systems into a cohesive infrastructure. Development is simplified with a transparent, lightweight service framework which uses Enterprise Integration Platform (EIP) technology. This allows developers to work with familiar technologies such as Apache Camel, BPEL, BPMN, or POJOs. To reduce the operational costs of production and maintenance, the platform utilizes an automatable, content-aware repository and service activity monitoring. These support the entire service life cycle.

[Report a bug](#)

1.2. CORE FUNCTIONALITY

Red Hat JBoss Fuse Service Works provides the following core functionality:

Enterprise Integration Pattern (EIP) Based Development

The versatile EIP framework is implemented in routing and transformation processes for faster and more efficient integration solutions.

High Performance Messaging

A high performance messaging broker supports messaging patterns such as publish-subscribe, point-to-point, and store-forward, and multiple cross language clients.

Service Development

The web services framework exposes integration assets as services and calls external services, supporting all major web services standards. It also supports RESTful calls.

Structured Service Development

A lightweight service development framework provides full life-cycle support for developing, deploying, and managing service-based applications.

Automatable Registry with Workflow

Manage the life-cycle of services from design, development, and deployment by defining, exposing, and enforcing rules or policies.

Business Transaction Monitoring

Capture service activity information, define and collect metrics, and define alerts and SLAs.

[Report a bug](#)

1.3. SYSTEM INTEGRATION

Integrating your major business systems into a cohesive infrastructure can be a challenge, especially when you have legacy applications. Red Hat JBoss Fuse Service Works provides a number of ways that enable you to integrate both new and legacy applications. Development is simplified with a transparent, lightweight service framework which uses the EIP technology. This allows developers to focus on higher order concepts while still working with familiar technologies such as Apache Camel, BPEL, BPMN, or POJOs. To reduce the operational costs of production and maintenance, the platform utilizes an automatable, content-aware repository and service activity monitoring. These support the entire service life cycle and development, QA, and production teams with runtime and design time visibility, monitoring, and alerting.

[Report a bug](#)

1.4. INTEGRATION USE CASE

Acme Equity is a large financial service. The company possesses many databases and systems. Some are older, COBOL-based legacy systems and some are databases obtained through the acquisition of smaller companies in recent years. It is challenging and expensive to integrate these databases as business rules frequently change. The company wants to develop a new series of client-facing e-commerce websites, but these may not synchronize well with the existing systems as they currently stand.

The company wants an inexpensive solution but one that adheres to the strict regulations and security requirements of the financial sector. What the company does not want to do is to have to write and maintain “glue code” to connect their legacy databases and systems.

Red Hat JBoss Fuse Service Works was selected as a middleware layer to integrate these legacy systems with the new customer websites. It provides a bridge between front-end and back-end systems. Business rules implemented with Red Hat JBoss Fuse Service Works can be updated quickly and easily.

As a result, older systems can now synchronize with newer ones due to the unifying methods of Red Hat JBoss Fuse Service Works. There are no bottlenecks, even with tens of thousands of transactions per month. Various integration types, such as XML, JMS and FTP, are used to move data between systems. Any one of a number of enterprise-standard messaging systems can be plugged into Red Hat JBoss Fuse Service Works providing further flexibility.

An additional benefit is that the system can now be scaled upwards easily as more servers and databases are added to the existing infrastructure.

[Report a bug](#)

1.5. CORE AND COMPONENTS

Red Hat JBoss Fuse Service Works provides an environment for easily applying SOA concepts to integrated applications. A SwitchYard application consists of components such as composite services and composite references. These provide service definitions and accessibility.

Along with SwitchYard, Red Hat JBoss Fuse Service Works is made up of a number of components including a rules-based router (Apache Camel), a web service framework (Apache CXF), and a message broker (Apache ActiveMQ).

[Report a bug](#)

1.6. RED HAT JBOSS FUSE SERVICE WORKS FEATURES

SwitchYard

SwitchYard is a lightweight service delivery framework providing full life-cycle support for developing, deploying, and managing service-oriented applications.

Business Process Execution Language (BPEL)

You can use web services to orchestrate business rules using this language. It is included with Red Hat JBoss Fuse Service Works for the execution of business process instructions.

Smooks

This transformation engine can be used in conjunction with Red Hat JBoss Fuse Service Works to process messages. It can also be used to split messages and send them to the correct destination.

JBoss Rules

This is the rules engine that is packaged with Red Hat JBoss Fuse Service Works. It can infer data from the messages it receives to determine which actions need to be performed.

[Report a bug](#)

1.7. COMPONENTS OF RED HAT JBOSS FUSE SERVICE WORKS

Red Hat JBoss Fuse Service Works ships with a number of components which enable its multi-functional capabilities.

Table 1.1. Red Hat JBoss Fuse Service Works Components

Component	Function
SwitchYard	Service delivery framework
JBoss Rules	Business rules engine with complex event processing
Design Time Governance	A service registry/repository
Runtime Governance	Service activity monitoring
JBoss Operations Network	Operations, administration, and management tools
JBoss EAP	A full JavaEE application server
Apache Camel	Rules Based Router
Smooks	Framework for processing XML and non-XML data using Java
ModeShape	Data Store
ActiveMQ/A-MQ Messaging	Messaging and Integration Patterns Server

Component	Function
Apache CXF	Services Framework

These components can be used in Red Hat JBoss Fuse Service Works to enable developers to build the required functionality using reliable and familiar tools. Some examples of how the components can be used are shown below:

Bean Services with CDI

SwitchYard leverages the power of Java EE6 and CDI to allow Java objects become services by adding an `@Service` annotation to your bean. Beans are automatically registered at runtime and references to other services can be injected as CDI beans using the `@Inject` annotation. Use CDI in your JSP and JSF applications to inject enterprise services into the web tier.

Declarative Transformation

With declarative transformation in SwitchYard, you can define the transformation and types to which it applies. SwitchYard automatically registers and executes the transformation. Choose from Smooks, Java, XSLT, JSON, and more.

Decision Services with JBoss Rules

Encapsulate business rules as decision services using the JBoss Rules component in SwitchYard. Each service has a well-defined contract with protocol binding details and marshaling details abstracted away by SwitchYard.

Smooks

This transformation engine can be used in conjunction with Red Hat JBoss Fuse Service Works to process messages.

Business Process Execution Language (BPEL)

You can use web services to orchestrate business rules using this language. It is included with Red Hat JBoss Fuse Service Works for the execution of business process instructions.

JBoss Rules

This is the rules engine that is packaged with Red Hat JBoss Fuse Service Works. It can infer data from the messages it receives to determine which actions need to be performed.

Testing

Comprehensive unit test support is provided to allow you to test services as you develop them.

[Report a bug](#)

CHAPTER 2. READ ME

2.1. BACK UP YOUR DATA



WARNING

Red Hat recommends that you back up your system settings and data before undertaking any of the configuration tasks mentioned in this book.

[Report a bug](#)

2.2. RED HAT DOCUMENTATION SITE

Red Hat's official documentation site is at <https://access.redhat.com/site/documentation/>. There you will find the latest version of every book, including this one.

[Report a bug](#)

CHAPTER 3. JBOSS INTEGRATION AND SOA DEVELOPMENT

3.1. JBOSS INTEGRATION AND SOA DEVELOPMENT

The **JBoss Integration and SOA Development** plug-in is provided to support JBoss Fuse Service Works in JBoss Developer Studio. It provides the following features:

- Creation of SwitchYard projects
- Adding SwitchYard capabilities to existing Maven based JBoss Developer Studio projects
- Configuration of SwitchYard capabilities
- A graphical editor for editing SwitchYard application configuration
- Java2WSDL
- XML catalog entries for SwitchYard configuration schema
- Integration supporting the SwitchYard Maven plug-in (org.switchyard:switchyard-plugin)
- Support for workspace deployment of SwitchYard projects

The JBoss Integration and SOA Development plug-in is provided by the JBoss Development Studio Integration Stack.

[Report a bug](#)

3.2. INSTALLING JBOSS DEVELOPER STUDIO INTEGRATION STACK

JBoss Developer Studio Integration Stack is not packaged as part of JBoss Developer Studio installations. These plug-ins must be installed independently through JBoss Central, as detailed in the procedure below.

Procedure 3.1. Install JBoss Developer Studio Integration Stack

1. Start JBoss Developer Studio.
2. In JBoss Central, select the **Software/Update** tab. Scroll through the list to locate **JBoss Developer Studio Integration Stack**. Select the check box next to **JBoss Integration and SOA Development** and click **Install**.

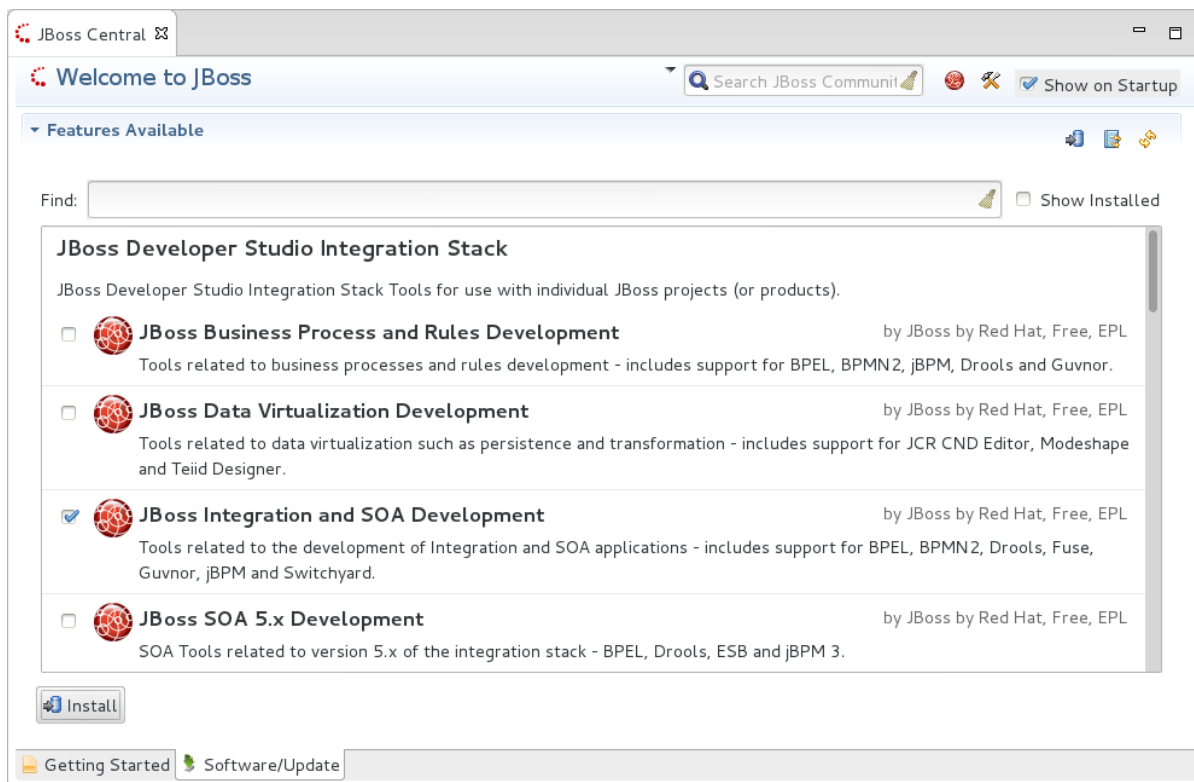


Figure 3.1. Find JBoss Developer Studio Integration Stack in JBoss Central Software/Update Tab

3. In the **Install** wizard, ensure the check boxes are selected for the software you want to install and click **Next**. It is recommended that you install all of the selected components.
4. Review the details of the items listed for install and click **Next**. After reading and agreeing to the license(s), click **I accept the terms of the license agreement(s)** and click **Finish**. The **Installing Software** window opens and reports the progress of the installation.
5. During the installation process you may receive warnings about installing unsigned content. If this is the case, check the details of the content and if satisfied click **OK** to continue with the installation.



Figure 3.2. Warning Prompt for Installing Unsigned Content

6. Once installing is complete, you are prompted to restart the IDE. Click **Yes** to restart now and **No** if you need to save any unsaved changes to open projects. Note that changes do not take effect until the IDE is restarted.

Once installed, you may need to complete additional configuration actions before you can use the individual JBoss Developer Studio Integration Stack components. For plug-in specific configuration information, see the appropriate Red Hat JBoss product documentation available from <https://access.redhat.com/site/documentation> on the Red Hat Customer Portal.



IMPORTANT

The installation method for early releases of JBoss Developer Studio Integration Stack may vary from that given here. For instructions, see the appropriate Red Hat JBoss product documentation available from <https://access.redhat.com/site/documentation> on the Red Hat Customer Portal.

[Report a bug](#)

3.3. HELPFUL TIPS

Honor all XML schema locations

After installation, go to **XML** → **XML Files** → **Validation** → **Preferences** and ensure **Honor all XML schema locations** check box is cleared. This prevents erroneous XML validation errors from appearing on `switchyard.xml` files.

DTD warning

Importing SwitchYard quickstarts into JBoss Developer Studio results in non-fatal warnings regarding `log4j.dtd`. These can be safely ignored. To stop receiving the warning, ensure the file `log4j.xml` is closed before starting a project.

JavaSE-1.6 error message

When commencing a project, a warning may be displayed saying "Build path specifies execution environment JavaSE-1.6". To disable this warning, go to your Java preferences and ensure that JavaSE-1.7 JDK is checked to support JavaSE-1.6 environments.

[Report a bug](#)

3.4. RUNNING QUICKSTARTS FROM JBOSS DEVELOPER STUDIO

Overview

This topic demonstrates how to import a quickstart application to JBoss Developer Studio and then deploy it to a running application server.

Prerequisites

The JBoss Integration and SOA Development tools must be installed from the JBoss Developer Studio Integration Stack.

1. Open JBoss Developer Studio.
2. Click **File** → **Import** → **Maven** → **Existing Maven Projects**.
3. Select **Browse** and navigate to the quickstart directory, for example, `EAP_HOME/quickstarts/switchyard/bean-service` and then select **OK**.

The import tool scans the directory to locate the associated `pom.xml` file.

4. Click **Finish**.
5. The quickstart is listed in the Project Explorer view. You can expand the project to explore its contents.
6. In the Project Explorer view, right-click on the project's name and click **Run as** → **Run on server** → **EAP**.

Result

The quickstart application is deployed to the server and enabled by default.

[Report a bug](#)

3.5. IMPORT PROJECTS FROM A GIT REPOSITORY IN JBOSS DEVELOPER STUDIO

JBoss Developer Studio can be configured to connect to a central Git asset repository. The repository is where versions of rules, models, functions and processes are stored. This Git repository must already be defined by the KIE Workbench.

1. Start the Red Hat JBoss server (if not already running) by selecting the server from the server tab and click the start icon.
2. Select **File** → **Import** and navigate to the Git folder. Open the Git folder to select **Projects from Git** and click next.
3. Select the repository source as **URI** and click next.
4. Enter the details of the Git repository in the next window and click next.

Source Git Repository



Enter the location of the source repository.

Location

URI: Local File...

Host:

Repository path:

Connection

Protocol: ⌵

Port:

Authentication

User:

Password:

Store in Secure Store

?
< Back
Next >
Cancel
Finish

Figure 3.3. Git Repository Details

5. Select which branch you want to import in the next window and click next.
6. You are presented with the option to define the local storage for this project. Enter (or select) a non-empty directory, make any configuration changes and click next.
7. Import the project as a general project in the next window and click next. Name this project and click Finish.

[Report a bug](#)

3.6. SETTING A NEW RULES RUNTIME IN JBOSS DEVELOPER STUDIO

Setting this runtime provides an environment for new rules sets. It consists of a collection of jar files which are then utilized in rules creation. Once you have set up a runtime, you can go about adding and modifying rules in JBoss Developer Studio.

1. Download and unzip the runtime jars files located in the `jboss-brms-engine.zip` archive of the JBoss BRMS Deployable zip archive (available from [Red Hat Customer Portal](#)).

2. From the Red Hat JBoss Developer Studio menu, select **Window** and click **Preferences**.
3. Select **JBoss Rules** → **Installed JBoss Rules Runtimes** → **Runtime locations**.
4. Click **Add**; provide a name for the new runtime, and click **Browse** to navigate to the directory where the runtime is located.
5. Click **OK**, select the new runtime and click **OK** again. A dialog box indicates, if you have existing projects, that JBoss Developer Studio must be restarted to update the Runtime.

[Report a bug](#)

CHAPTER 4. SETTING UP THE SERVER

To make it possible for the tooling to manage a server, you need to add it to the Servers list. You can set up target runtime server using the **Servers** view. Once added, the server is displayed in the **Servers** view. Selecting a server in **Servers** view, enables you to start it, to stop it, or to delete its configuration. You can add multiple servers of the same type, as long as each uses a separate installation directory.

[Report a bug](#)

4.1. ADD JBOSS EAP SERVER

1. In JBoss Developer Studio, click the **Servers** view. If the **Servers** view is not visible, click **Window** → **Show View** → **Servers**.
2. If no servers have been previously created then the **Servers** view displays a new server hyperlink. Click this link to create a new server.

If there are one or more existing servers, right-click an existing server and click **New** → **Server**.

3. In the **Define a New Server** dialog, select a JBoss Enterprise Application Platform server from the **Select the server type** list.
4. The **Server's host name** and **Server name** fields are completed by default. In the **Server name** field, you can type a custom name by which to identify the server in the **Servers** view.
5. From the **Server runtime environment** list, select an existing server runtime environment for the application server type. Alternatively, to create a new runtime environment click **Add** and complete the fields and options as appropriate.



NOTE

If the **Server runtime environment** field is not shown, no server runtime environments exist for the selected application server type. To create a new server runtime environment without canceling the wizard, click **Next** and complete the fields and options as appropriate.

6. Click **Next**.
7. The server behavior options displayed vary depending on the selected application server type. To specify that the server life-cycle is to be managed from outside the IDE, select the **Server is externally managed** check box.

To specify that the server should be launched to respond to requests on all hostnames, select the **Listen on all interfaces to allow remote web connections** check box.

From the **location** list, select **Local**. Click **Next**.

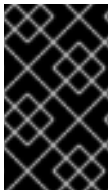
**NOTE**

The Expose your management port as the server's hostname option, which enables management commands sent by the IDE to be successfully received by the server, is bypassed for local servers regardless of whether the check box is selected.

8. To select applications to deploy with this server, from the Available list select the applications and click **Add**. Applications to be deployed are detailed in the Configured list.
9. Click **Finish** to create the server.

Result

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

**IMPORTANT**

You can create multiple servers that use the same application server. But a warning is displayed if you try to simultaneously run more than one server on the same host. This is because multiple running servers on the same host can result in port conflicts.

[Report a bug](#)

CHAPTER 5. RULES

5.1. DSL

A domain-specific language (DSL) is a set of custom rules, that is created specifically to solve problems in a particular domain and is not intended to be able to solve problems outside it. A DSL's configuration is stored in plain text.

[Report a bug](#)

5.2. DSL EDITOR

- In JBoss Rules, the DSL configuration is stored in `.dsl` files. These can be created selecting **File** → **New** → **Other** → **Drools** → **Domain Specific Language** from the projects context menu.

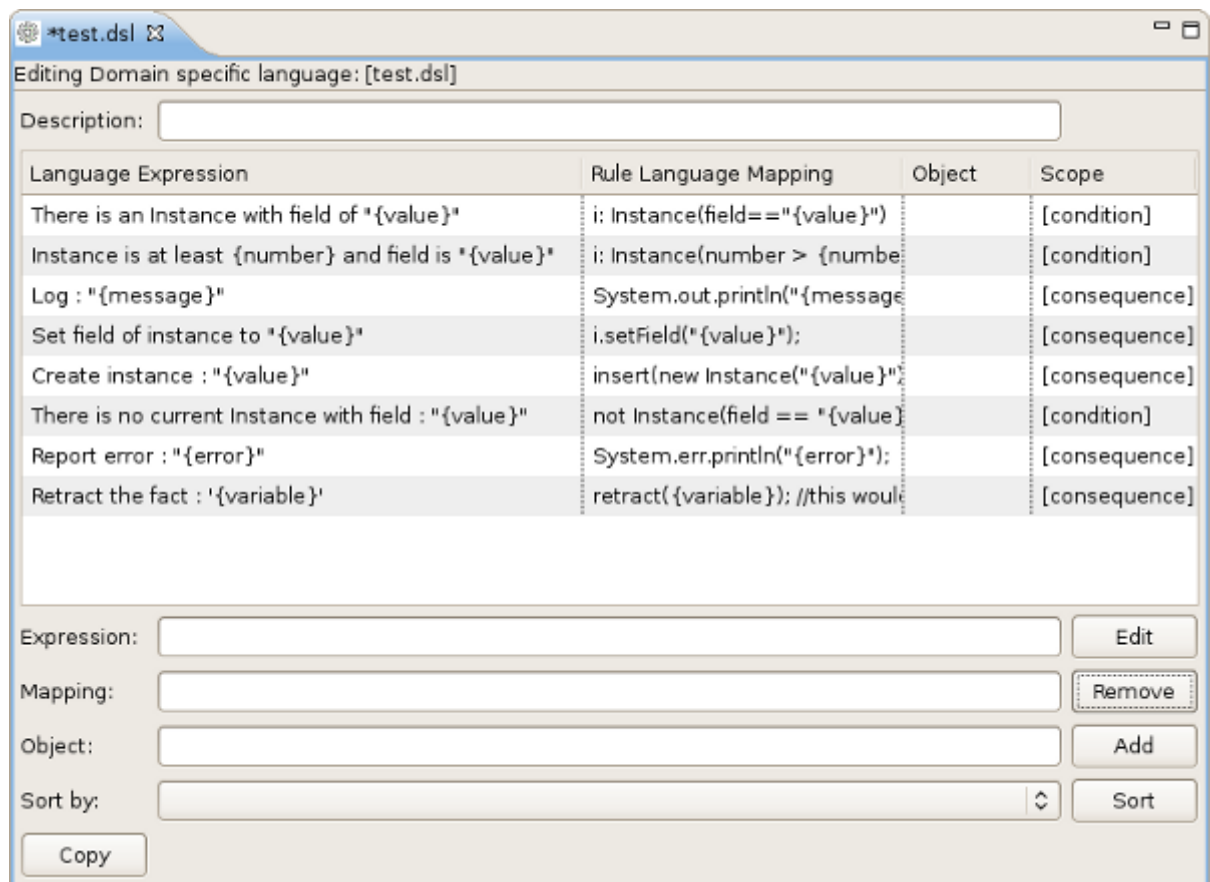


Figure 5.1. The DSL editor

[Report a bug](#)

5.3. DSL EDITOR COMPONENTS

Table 5.1. DSL Editor Components

Components	Description
Description	User's comments on a certain language message mapping.

Components	Description
Table of language message mappings	<p>The table is divided into 4 rows:</p> <ul style="list-style-type: none"> • <i>Language Expression</i> : expression you want to use as a rule • <i>Rule Language Mapping</i> : the implementation of the rules. The rule for this language expression is compiled by the rule engine compiler. • <i>Object</i> : name of the object • <i>Scope</i> : indicates where the expression is targeted, if its for the "condition" part of the rule ,"consequence" part, etc. <p>By clicking on some row's header you can sort the lines in the table according to the clicked row. By double clicking on the line the Edit language mapping is displayed.</p>
Expression	Shows the language expression of the selected table line (language message mapping).
Mapping	Shows the rule of language mapping for the selected table line (language message mapping).
Object	Shows the object for the selected table line (language message mapping).
Sort By	This option allows you to change the sorting order of the language message mappings. To do this select from the drop down list the method of sorting you want and click the Sort button.
Buttons	<ul style="list-style-type: none"> • <i>Edit</i> : Click this button to edit the selected line in the language message mappings table. • <i>Remove</i> : Click this button to delete the selected mapping line. • <i>Add</i> : Click this button to add new mapping lines to the table. • <i>Sort</i> : See the <i>Sort By</i> row above for more information. • <i>Copy</i> : Click this button to add new mapping lines to the table in which all the information is copied from the selected mapping line.

[Report a bug](#)

5.4. ADD A LANGUAGE MAPPING WIZARD

By adding this wizard, you can create language message maps. These allow you to control message flows. They can then be configured in JBoss Developer Studio.

Procedure 5.1. Task

1. Click the **Add** button and a dialog box appears.
2. Enter the details of your language mapping wizard into the relevant fields.

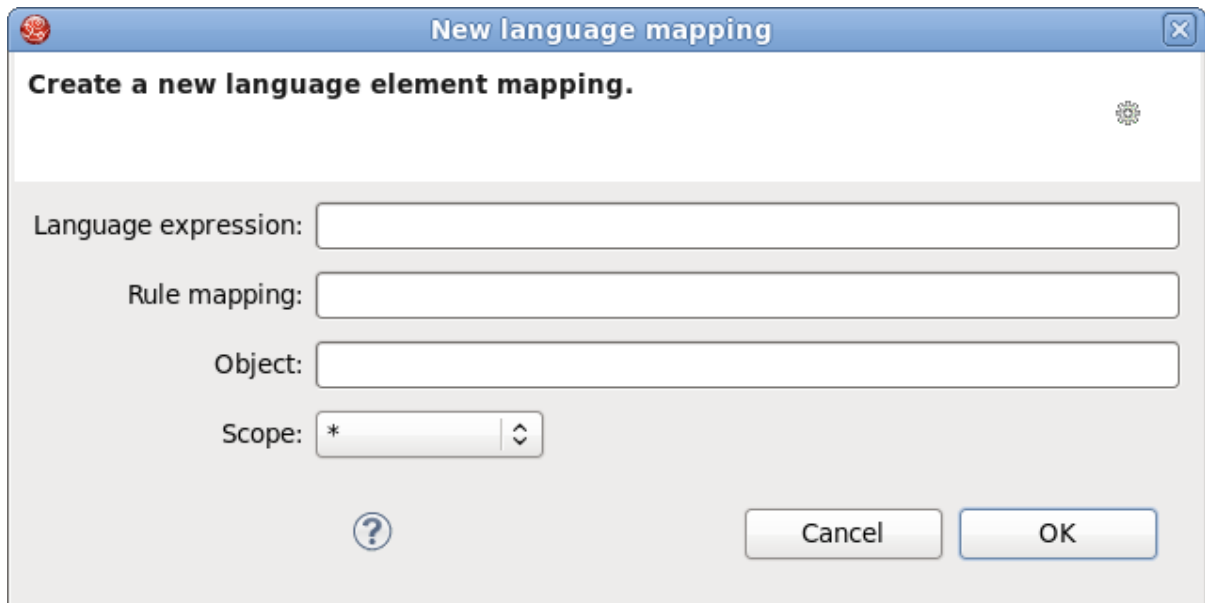


Figure 5.2. Dialog box

[Report a bug](#)

5.5. EDIT THE LANGUAGE MAPPING WIZARD

You can configure different fields in the Language Mapping Wizard to suit your needs. Any language message mapping can be modified through this process. You can edit the language expressions, rule mapping, objects and scopes of the instance.

1. This wizard can be opened by double-clicking a line in the table of language message mappings or by clicking the **Edit** button.

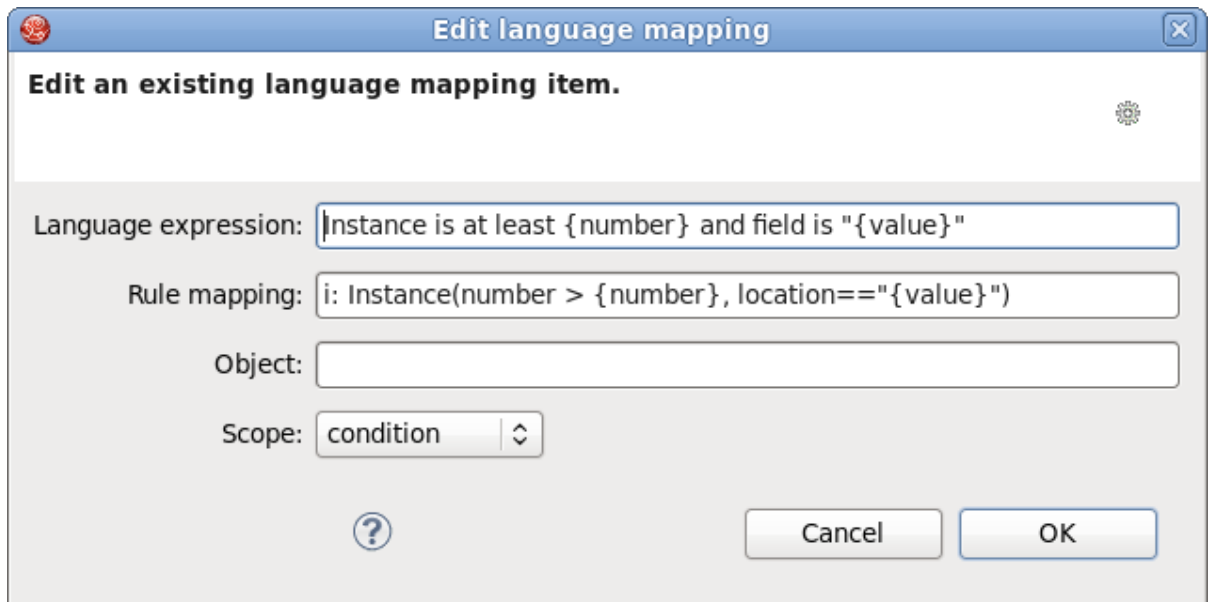


Figure 5.3. Editing options

2. To change the mapping, edit the appropriate options and click the **OK** button.

[Report a bug](#)

5.6. JBOSS RULES FLOW EDITOR

1. Ruleflows can only be set by using the graphical flow editor which is part of the JBoss Rules plug-in for JBoss Developer Studio. Once you have set up a JBoss Rules project, you can start adding ruleflows. Add a ruleflow file (.rf) by clicking on the project and selecting **File** → **New** → **Other** → **Flow File**.

By default these ruleflow files (.rf) are opened in the graphical Flow editor.



2. The Flow editor consists of a **palette**, a **canvas** and an **outline** view. Select the element you would like to create in the palette and then add it to the canvas by clicking on the preferred location.
3. Clicking on the **Select** option in the palette and then on the element in your ruleflow allows you to view and set the properties of that element in the **Properties** view.
4. The **Outline** view is useful for complex schemata where not all nodes are seen at one time. Using the **Outline** view allows you to easily navigate between parts of a schema.

[Report a bug](#)

5.7. DIFFERENT TYPES OF CONTROL ELEMENTS IN FLOW PALETTE

Table 5.2. Different Types of Control Elements in Flow Palette




Icon	Name	Description
	Select	Select a node on the canvas.





Icon	Name	Description
	Marquee	Selects a group of elements.
	Sequence flow	Joins two elements on the canvas

[Report a bug](#)

5.8. DIFFERENT TYPES OF NODES IN FLOW PALETTE

Table 5.3. Different Types of Nodes in Flow Palette

Icon	Name	Description
	Start Event	The start of the ruleflow. There can only have one start node. The Start Event cannot have incoming connections and should have one outgoing connection. Whenever the ruleflow process is started, the execution is started here and is automatically forwarded to the first node linked to this Start Event.
	End Event	A ruleflow file can have multiple End Events. This node should have one incoming connection and no outgoing connections. When an end node is reached in the ruleflow, the ruleflow is terminated (including other remaining active nodes when parallelism is used).
	Rule Task	Represents a set of rules. A Rule Task node should have one incoming connection and one outgoing connection. The RuleFlowGroup property is used to specify the name of the ruleflow-group representing the set of rules. When a Rule Task node is reached in the ruleflow, the engine starts executing rules that are part of the corresponding ruleflow-group. Execution automatically continues to the next node when there are no more active rules in this ruleflow-group.

Icon	Name	Description
	Gateway[diverge]	Allows you to create branches in your ruleflow. A Gateway[diverge] node should have one incoming connection and two or more outgoing connections.
	Gateway[converge]	Allows you to synchronize multiple branches. A Gateway[converge] node should have two or more incoming connections and one outgoing connection.
	Reusable Sub-Process	Represents the invocation of another ruleflow from this ruleflow. A subflow node should have one incoming connection and one outgoing connection. It contains the property processId which specifies the ID of the process that should be executed. When a Reusable Sub-Process node is reached in the ruleflow, the engine starts the process with the given ID. The subflow node continues only if the other subflow process has terminated its execution. The subflow process is started as an independent process, which means that the subflow process is not terminated if this process reaches an end node.
	Script Task	Represents an action that should be executed in this ruleflow. A Script Task node should have one incoming connection and one outgoing connection. It contains the property "action" which specifies the action that should be executed. When a Script Task node is reached in the ruleflow, it executes the action and continue with the next node. An action should be specified as a piece of (valid) MVEL code.

[Report a bug](#)

5.9. RULE EDITOR

The Rule editor works on files that have a `.dr1` (or `.rule` in the case of spreading rules across multiple rule files) extension. The editor follows the pattern of a normal text editor in JBoss Developer Studio, with all the normal features of a text editor.

[Report a bug](#)

5.10. CONTENT ASSIST

While working in the Rule editor you can get a content assistance the usual way by pressing **Ctrl+Space**. Content Assist shows all possible keywords for the current cursor position. It also suggests all available fields in a message.

[Report a bug](#)

5.11. CODE FOLDING

Code Folding is used in the *rule editor*. It allows you to hide and show sections of a file use the icons with minus and plus on the left vertical line of the editor.

[Report a bug](#)

5.12. SYNCHRONIZATION WITH OUTLINE VIEW

The **Rule editor** works in synchronization with the **Outline view** which shows the structure of rules, imports objects into the file and also performs file functions. The view is updated on save. It provides a quick way of navigating around rules by names in a file which may have hundreds of rules. The items are sorted alphabetically by default.

[Report a bug](#)

5.13. RETE TREE VIEW

The Rete Tree view shows you the current **Rete Network** for your `.dr1` file. Just click on the Rete Tree tab at the bottom of the **Rule editor**. Afterwards you can generate the current Rete Network visualization. You can push and pull the nodes to arrange your optimal network overview.

If you have a large number of nodes, you can select some of them with a frame and pull them together. You can zoom in and out the Rete tree in case not all nodes are shown in the current view. For this use the combobox or + and - icons on the toolbar.



NOTE

The Rete Tree view works only in JBoss Rules Rule Projects, where the JBoss Rules Builder is set in the project properties.

[Report a bug](#)

5.14. BREAKPOINTS

Whenever the execution of rules runs into a breakpoint, the execution is halted. Once the execution

stops, you can inspect the variables known at that point and use any of the default debugging actions to decide what should happen next (step over, continue, etc). The Debug view can be used to inspect the content of the working memory and agenda.

[Report a bug](#)

5.15. CREATING BREAKPOINTS

Create breakpoints to help monitor rules that have been executed. Instead of waiting for the result to appear at the end of the process, you can inspect the details of the execution at each breakpoint you set. This is useful for debugging and ensuring rules are executed as expected.

1. To create breakpoints in the Package Explorer view or Navigator view of the JBoss Rules perspective, double-click the selected `.dr1` file to open it in the editor.
2. You can add and remove rule breakpoints in the `.dr1` files in two ways:
 - o Double-click the rule in the **Rule editor** at the line where you want to add a breakpoint. A breakpoint can be removed by double-clicking the rule once more.



NOTE

Rule breakpoints can only be created in the consequence of a rule. Double-clicking on a line where no breakpoint is allowed does nothing.

- o Right-click the ruler. Select the **Toggle Breakpoint** action in the context menu. Choosing this action adds a breakpoint at the selected line or remove it if there is one already.
3. The **Debug perspective** contains a **Breakpoints view** which can be used to see all defined breakpoints, get their properties, enable/disable and remove them. You can switch to it by clicking **Window → Perspective → Others → Debug**.

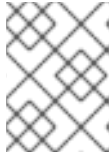
[Report a bug](#)

5.16. DEBUGGING

Once you have created breakpoints, you can use them to start debugging your rules processes when required. This allows you to monitor errors and fixes them automatically where possible. By using the debugging configuration with breakpoints, you can make sure your rules process runs as expected.

1. JBoss Rules breakpoints are only enabled if you debug your application as a JBoss Rules Application. To do this you should perform one of two actions:
 - o Select the main class of your application. Right-click on it and select **Debug As → JBoss Rules Application**.
 - o Alternatively, select **Debug As → Debug Configuration** to open a new dialog window for creating, managing and running debug configurations.

Select the **JBoss Rules Application** item in the left tree and click the **New launch configuration** button (leftmost icon in the toolbar above the tree). This creates a new configuration with a number of the properties already filled in based on main class you selected in the beginning. All properties shown here are the same as any standard Java program.

**NOTE**

Remember to change the name of your debug configuration to something meaningful.

2. Click the **Debug** button on the bottom to start debugging your application.
3. After enabling the debugging, the application starts executing and halts if any breakpoint is encountered. This can be a JBoss Rules rule breakpoint, or any other standard Java breakpoint. Whenever a JBoss Rules rule breakpoint is encountered, the corresponding `.dr1` file is opened and the active line is highlighted. The **Variables** view also contains all rule parameters and their value. You can then use the default Java debug actions to decide what to do next (resume, terminate, step over, etc). The debug views can also be used to determine the contents of the working memory and agenda at that time as well (the current executing working memory is automatically shown).

[Report a bug](#)

CHAPTER 6. JBOSS DEVELOPER STUDIO BPEL PROJECTS

6.1. BPEL

Business Process Execution Language (BPEL) is an OASIS-standard language for business rules orchestration. The BPEL component in SwitchYard is a pluggable container in that allows a WS-BPEL business process to be exposed as a service through an interface defined using WSDL.

[Report a bug](#)

6.2. APACHE ODE

Apache ODE ("Orchestration Director Engine") is a software component that is designed to execute BPEL business processes. It sends and receives messages to and from web services, manipulates data and performs error handling in the method prescribed in your process definition. To learn more about Apache ODE, visit the project website at <http://ode.apache.org/>.

[Report a bug](#)

6.3. LOGGING INTO BPEL

You can access the BPEL server through your browser. Once you have logged in, you can begin creating and modifying projects in the graphical user interface. You are also able to change your login details to something more secure than the default.

Procedure 6.1. Task

1. Access the BPEL console at the following URL: <http://localhost:8080/bpel-console>. A login screen appears.
2. Enter your credentials. The default username is `admin` with password `admin`.
3. To change your login details, access the **Settings** tab in the BPEL editor and click on **Users**. In this section there are fields to change your username and password.

[Report a bug](#)

6.4. CREATING A BPEL PROJECT

Create a BPEL project to orchestrate your business rules. These projects negate the need to configure each rule individually. You can create your BPEL project directly in the SwitchYard editor so you do not have to edit XML by hand. Once you have completed creating your project, you can apply additional configurations to customize your business rules.

1. First, select **File** → **New** → **Project** → **BPEL 2.0** → **BPEL Project** or **Legacy BPEL Project** from the menu bar. Then click the **Next** button.

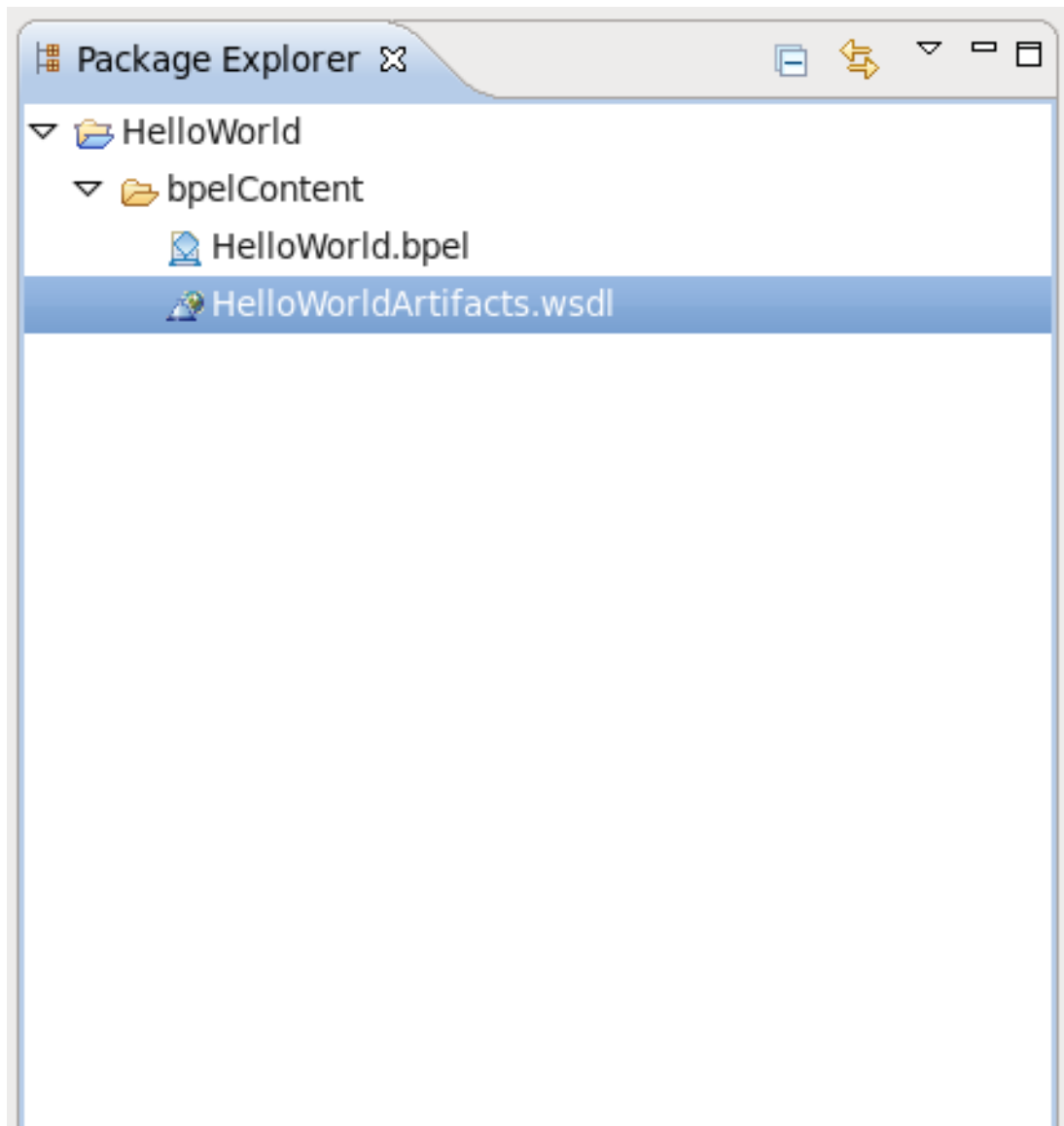


Figure 6.1. Diagram 1

2. Enter a project name in the Project Name field.

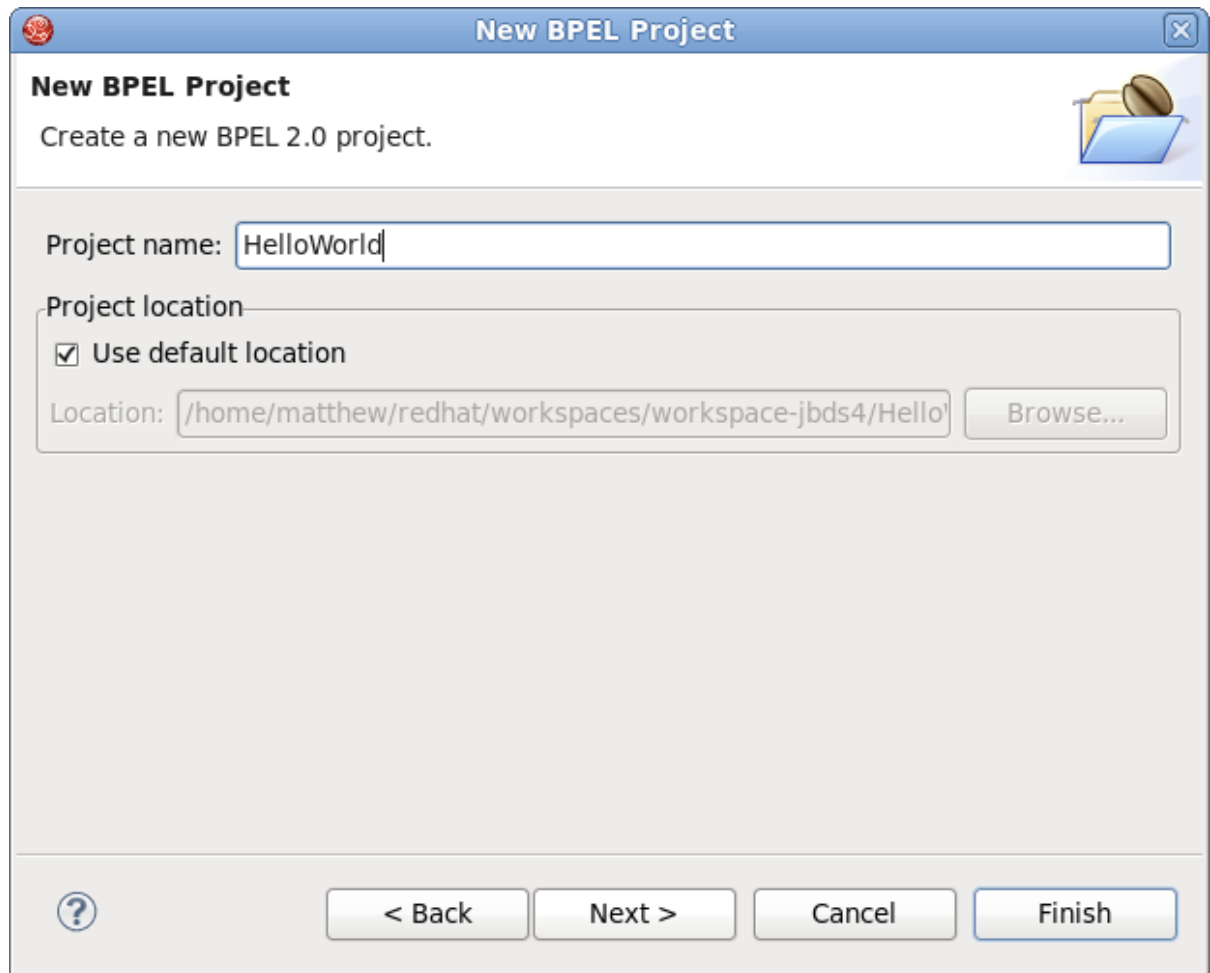


Figure 6.2. Diagram 2

3. Click the **Finish** button. The following screen appears.

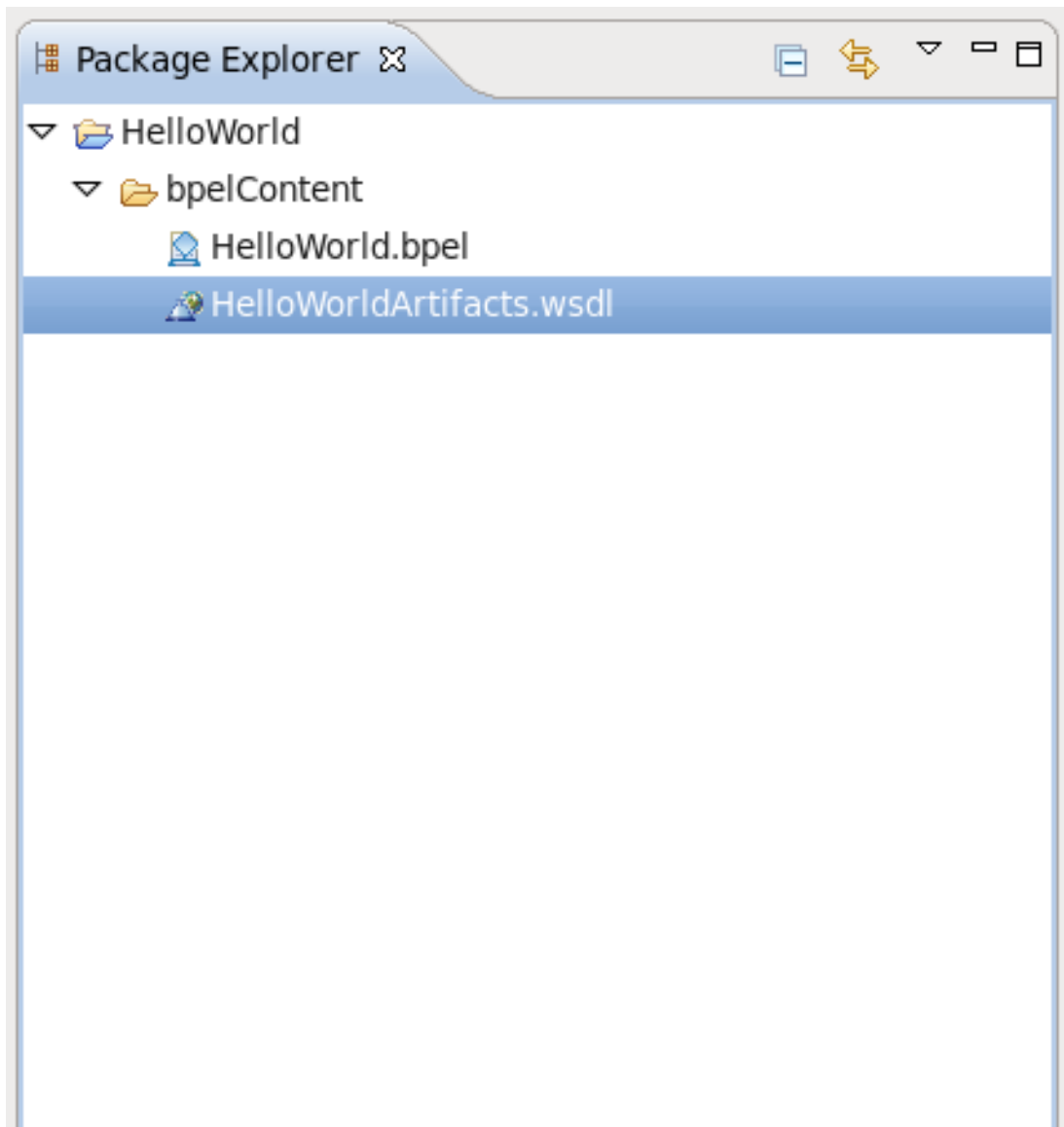


Figure 6.3. Diagram 3

4. You have now created a new project.

[Report a bug](#)

6.5. CREATING A BPEL PROCESS

Create a BPEL process to implement your projects. The wizard provides a simple way of setting the process defaults to provide maximum efficiency. Use it to customize values and WSDL information.

1. First, select **File** → **New** → **Others** → **BPEL 2.0** → **BPEL Process File** and click **Next**.

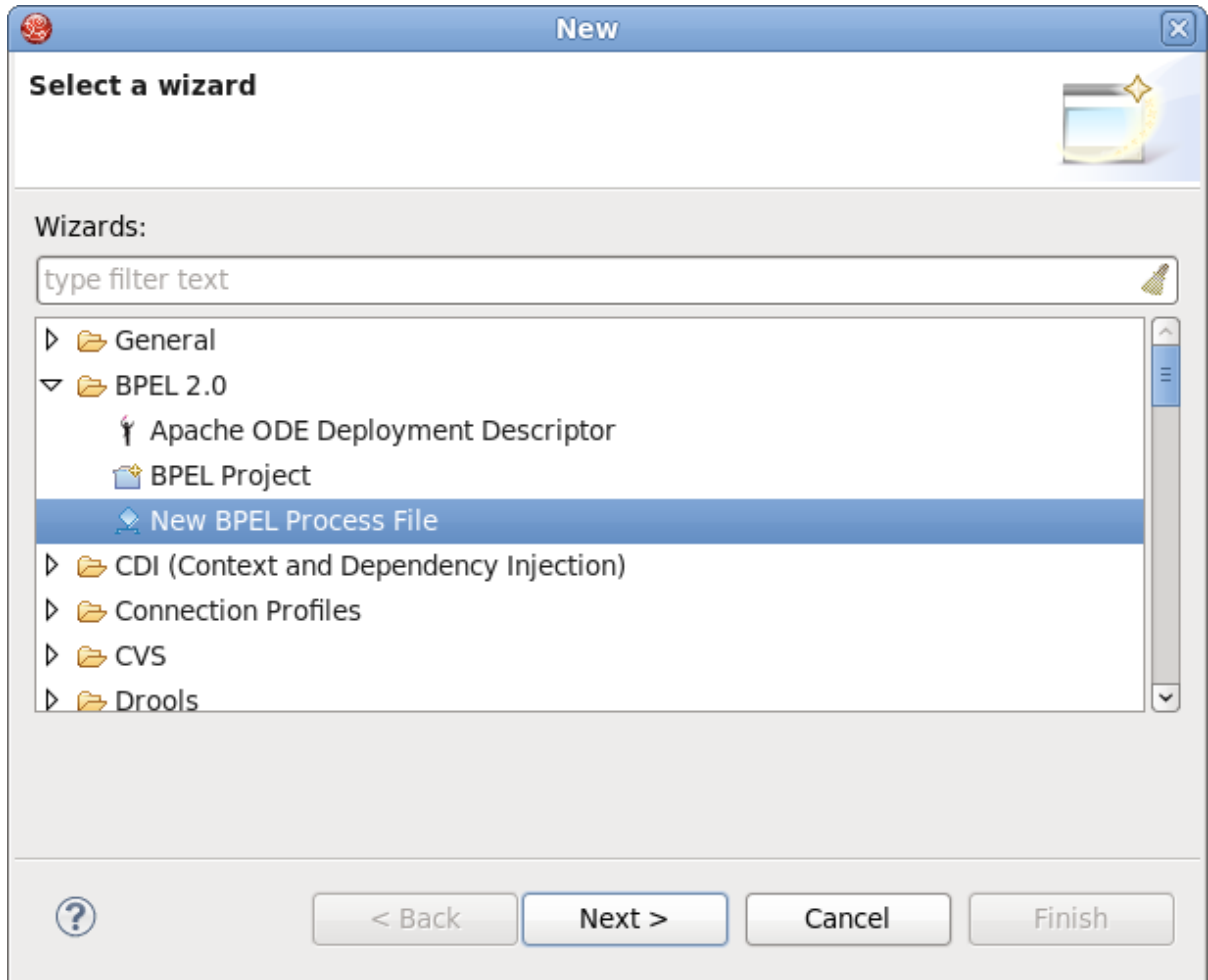


Figure 6.4. Diagram 1

2. From here you can choose to create a BPEL process from a template or a service description. The former is recommended.
3. Enter the following information:

Table 6.1. Fields and Values

Field	Value
BPEL Process Name	Enter a process name. For example, HelloWorld.
Namespace	Enter or select a namespace for the BPEL process.
Template	Select the appropriate template for the BPEL process. When you select the template, you can see information about it. Select Synchronous BPEL Process .

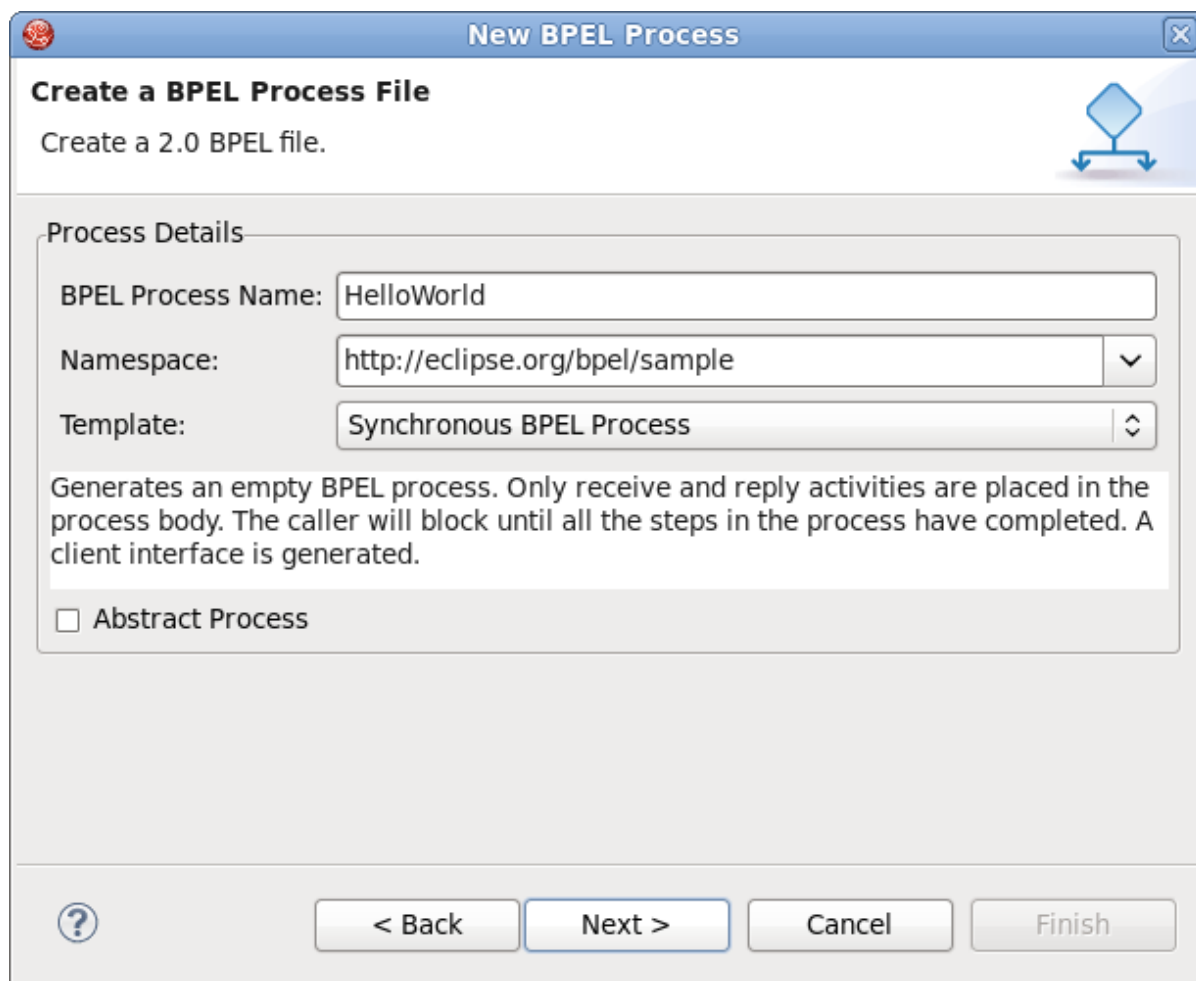


Figure 6.5. Diagram 2

4. Click **Next**. On this page, you can customize your WSDL service details using a template. Enter the following information:

Table 6.2. Fields and Values

Field	Value
Service Name	A WSDL service name for the BPEL process. The default name is HelloWorld .
Port Name	A WSDL port name for the BPEL process. The default name is HelloWorldPort .
Service Address	An address of the WSDL service for the BPEL process. The default value is http://localhost:8080/HelloWorld .
Binding Protocol	The binding protocol that you use in the WSDL. You can choose SOAP or HTTP. The default value is SOAP .

Create a WSDL File

Create a WSDL File for the BPEL Process

WSDL Details

Service Name: HelloWorld

Port Name: HelloWorldPort

Service Address: http://localhost:8080/HelloWorld

Binding Protocol: SOAP

< Back Next > Cancel Finish

Figure 6.6. Diagram 3

5. Click the **Next** button. On this page, you can select a folder for the process file from the projects in your workspace. If a folder is not selected, the default folder **HelloWorld/bpelContent** is used.
6. Click the **Finish** button. The process is complete.



NOTE

All of the files used in your BPEL project must be under the **bpelContent** folder of a BPEL project.

[Report a bug](#)

6.6. CREATE A NEW SERVER RUNTIME

It is necessary to create a server runtime in JBoss Developer Studio. This is so your applications have something to run on. You can use it to configure and deploy your applications.

Procedure 6.2. Task

1. Go to the **New Server** wizard.
2. Click on **Add**.

3. Fill in the name if you wish to do so (this step is optional because the name is preset).
4. Click on **Browse** and select the home directory.
5. Select one of the available configurations (the configuration you choose must have the BPEL engine available).
6. Click on **Finish**.

[Report a bug](#)

6.7. EDITING A BPEL PROCESS FILE

You can view a BPEL process file in the graphical interface. This way you are able to see all your BPEL files in front of you and modify them accordingly. By selecting the file from the palette, you are able to edit it.

1. Open the **Properties** view and **Palette** view by right-clicking the BPEL editor and selecting the **Show in Properties** or **Show Palette in Palette** view options.

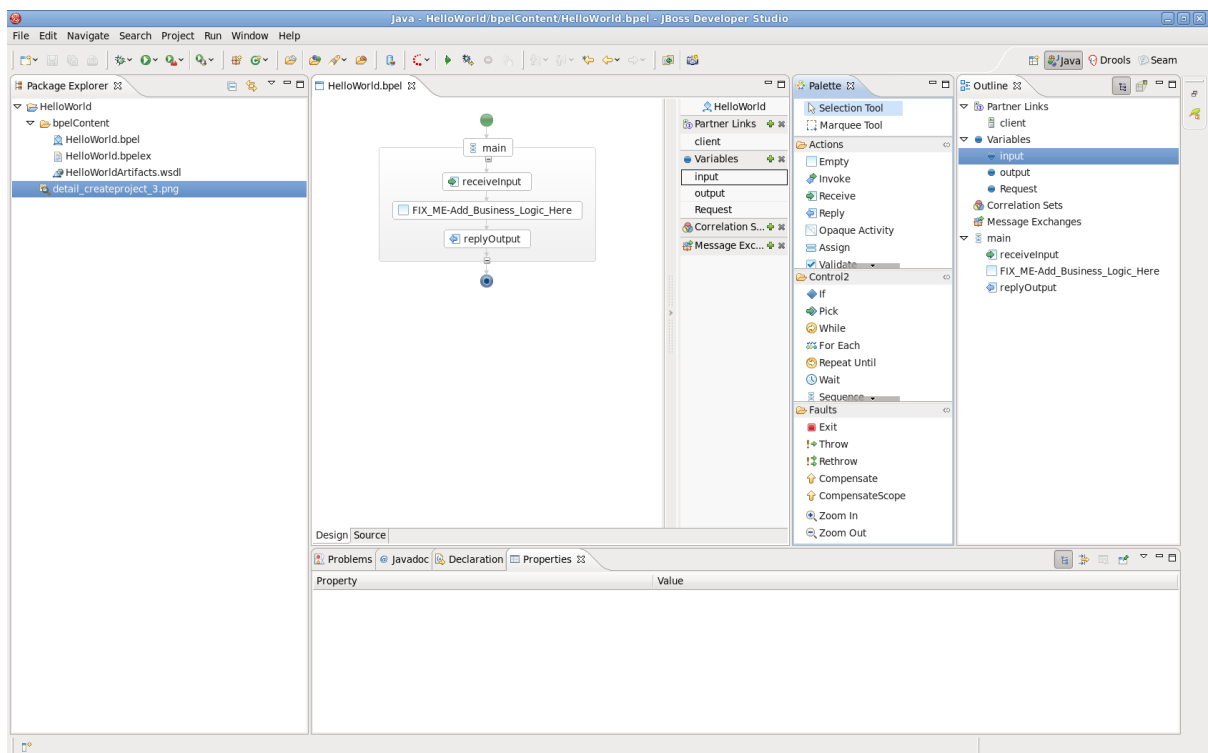


Figure 6.7. Diagram 1

2. In the **Palette** view, drag and drop your chosen BPEL element into the BPEL editor.
3. Switch to the **Properties** view to see information on the BPEL process.
4. The contents of the **Properties** view is automatically updated as elements are selected in the BPEL editor.

[Report a bug](#)

6.8. TABS SHOWN IN THE PROPERTIES VIEW

Table 6.3. Tabs Shown in the Properties View

Tab	Description
Description	Displays information about the element such as name, size, etc.
Details	Shows detailed and important information about the element. Most of the properties of an element are set in this section.
Join Behavior	Shows the Join Failure property of the element.
Documentation	Shows the documentation sub-element of an element.
Imports	Allows you to choose which documents are imported into the BPEL process definition.
Namespaces	Lets you edit the defined namespaces in the BPEL process document.

[Report a bug](#)

6.9. OBSERVING A BPEL PROCESS

By adding elements to a BPEL service, you are able to observe its output. This means you can monitor the progress of the process in the server window. If you see an error appear, you can halt the process and modify it to fix the problem.

1. Change the Empty element between elements `receiveInput` and `replyOutput` to `Assign`.
2. Add an `Assign` element between the `receiveInput` element and `replyOutput` element.
3. Click the `Assign` element in the BPEL editor to see the properties information in the Properties view.
4. Set its name in the Description tab as `assignHelloMesg`.
5. In the Details section of Properties view, click the `New` button to add a `copy` sub-element to the element. Assign "Variable to Variable" (`input:string` to `result:string`). An "initializer" popup dialog box appears. Click the `Yes` button.
6. Navigate down to the desired component and click it. The icon to the left of the component name indicates its type: a blue dot is the BPEL variable, an envelope is a message, an "e" is an XML element. Click the `New` button once more and select *Expression to Variable* (assign `concat($input.payload/tns:input, ' World')`) to `result:string`.

[Report a bug](#)

6.10. ADDING A SERVICE TO A WSDL FILE

The `HelloWorldArtifacts.wsdl` file is added to a service when you create a BPEL process file. A default service is already defined in this WSDL file, however you may wish to add your own service. This task shows you how to do this so that you are able to add the service that is most appropriate to your situation.

1. Open the file `HelloWorldArtifacts.wsdl` in the `HelloWorld` project.
2. Right-click the WSDL editor and select the **Add Service** option. A new service should appear in the editor. Name it `HelloWorldProcessService`. It has the Port named `NewPort`. Select it, right-click on it and rename it to `HelloWorldProcessPort` in the **Properties** view.

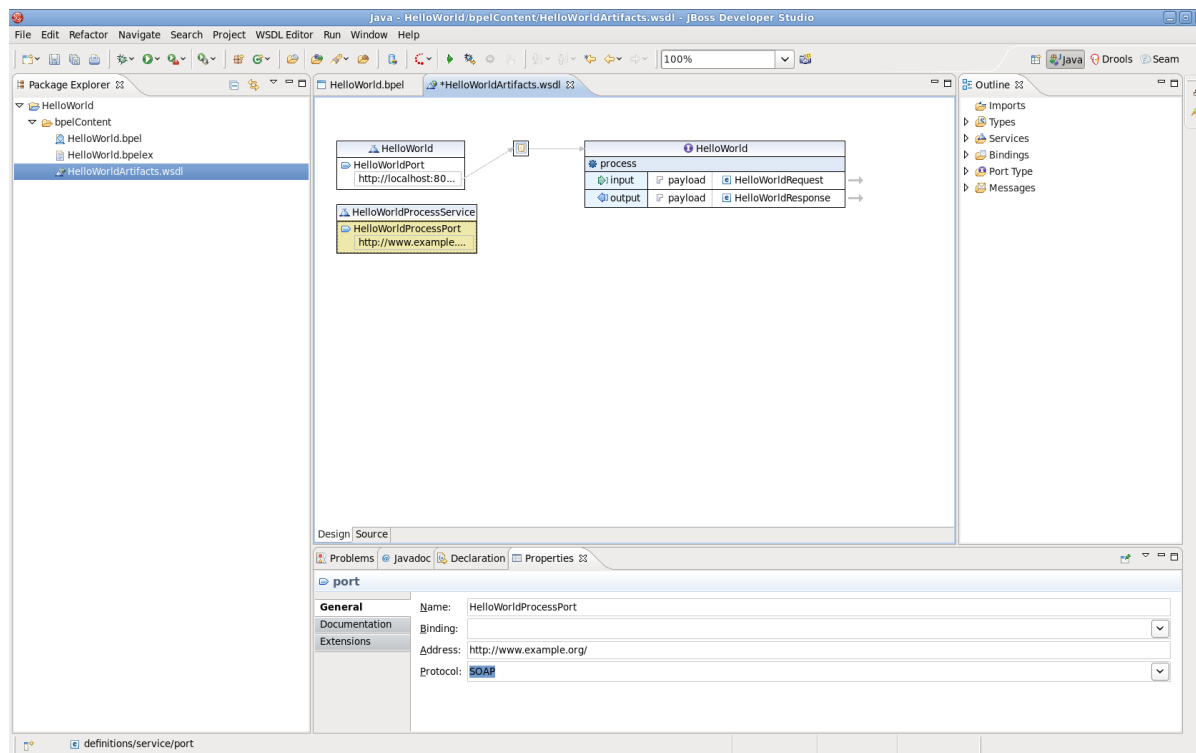


Figure 6.8. Diagram 1

3. Right-click in the whitespace of the WSDL editor and select the **Add Binding** option. A new Binding component appears in the editor. Name it `HelloWorldSOAPBinding`. Select it, and in the **General** tab of the **Properties** view and select `HelloWorld` as a port type in the **Port Type** field.
4. Click on the **Generate Binding Content** button to open the **Binding Wizard**.
5. In the wizard, select **SOAP** as the **Protocol**. Click the **Finish** button to close the wizard.

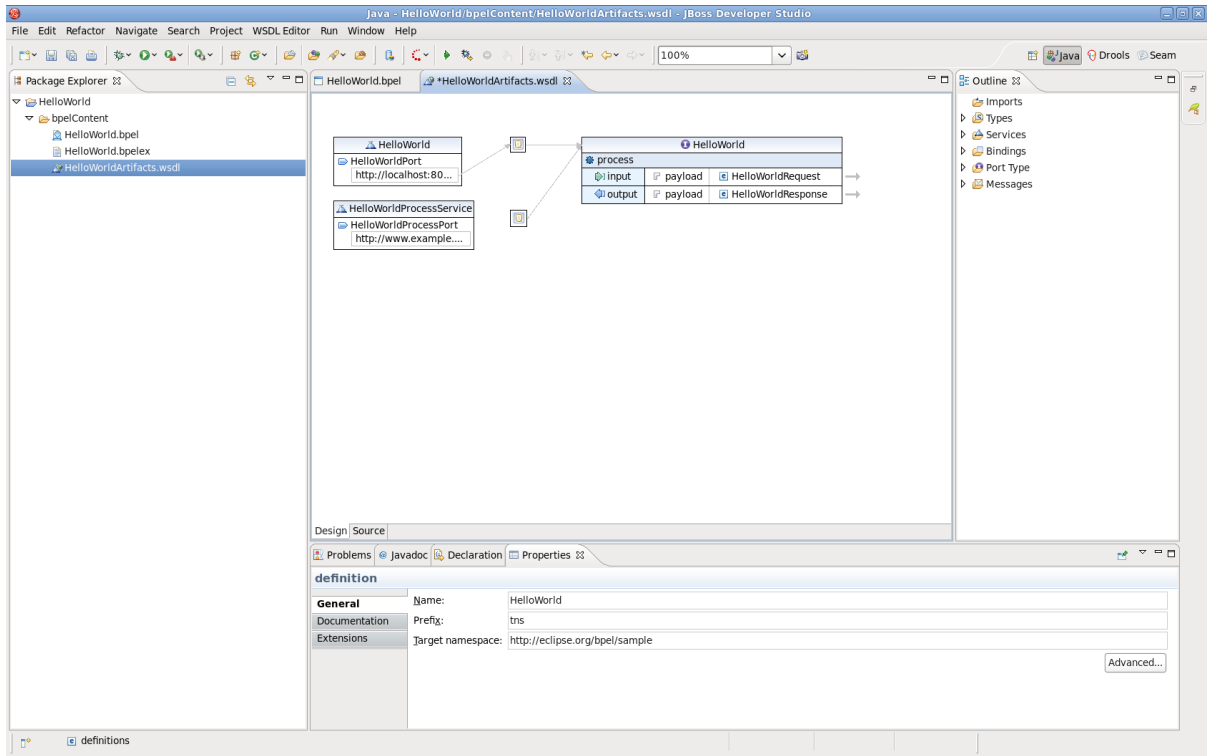


Figure 6.9. Diagram 2

6. Click the **HelloWorldProcessPort** property in the **General** section of the **Properties** view.
7. Select **HelloWorldSOAPBinding** in the **Binding** combobox.
8. Enter <http://localhost:8080/bpel/processes/HelloWorld?wsdl> in the **Address** field.

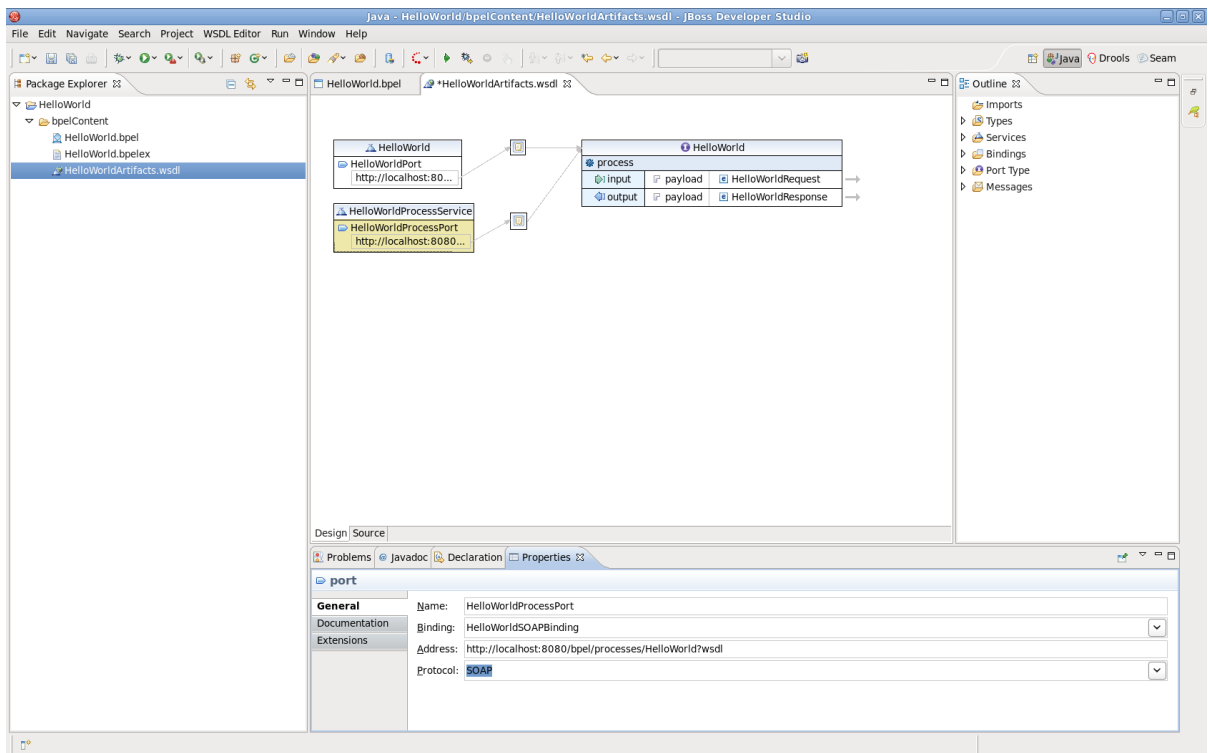


Figure 6.10. Diagram 3

[Report a bug](#)

6.11. CREATING A DEPLOY.XML FILE

Creating this file provides a basis for configuring BPEL processes. It is a deployment descriptor which you can use to enable events and modify existing ones. Add elements to it to dictate what your BPEL processes do.

1. To create a new `deploy.xml` file for deploying BPEL projects, select **File** → **New** → **Other** → **BPEL 2.0** → **BPEL Deployment Descriptor**. Click the **Next** button.

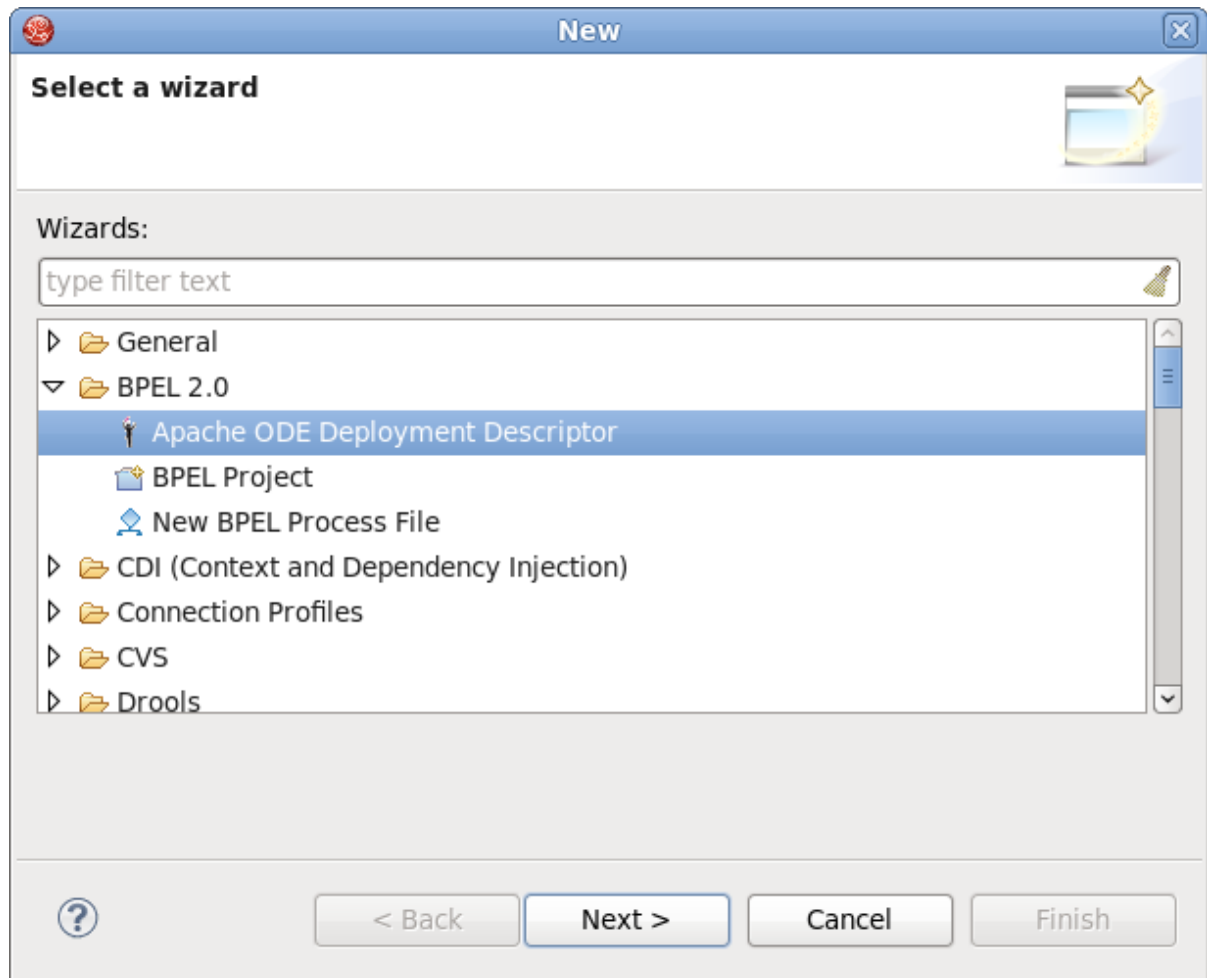


Figure 6.11. Diagram 1

2. On this page of the wizard, enter the BPEL Project. Do so by clicking the **Browse** button to select the BPEL project in your workspace that you want to deploy to the runtime.
3. Select the `bpelContent` folder in your new BPEL project for the `BPEL Project` field. Do not change the default file name which is `deploy.xml`.
4. Click on the **Finish** button to close the wizard and a new `deploy.xml` file is created.

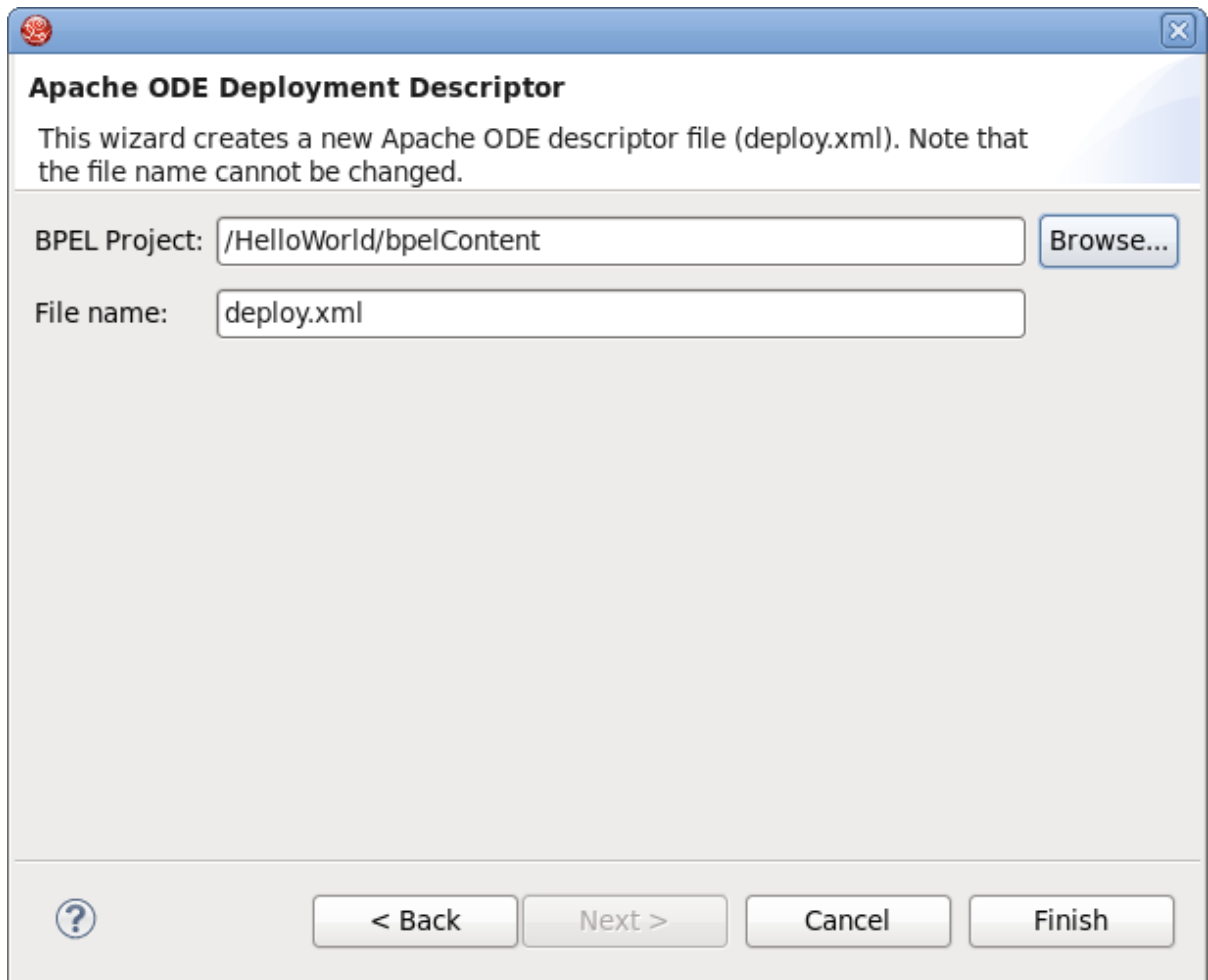


Figure 6.12. Diagram 2

5. Finally, double-click the `deploy.xml` file to open it in **ODE Descriptor Deployment Editor**. In the **Inbound Interfaces** section, click the **Associated Port** column and select `HelloWorldProcessPort` in the combobox. The **Related Service** and **Binding Used** columns should be automatically filled in. Save the changes to the `deploy.xml` file.

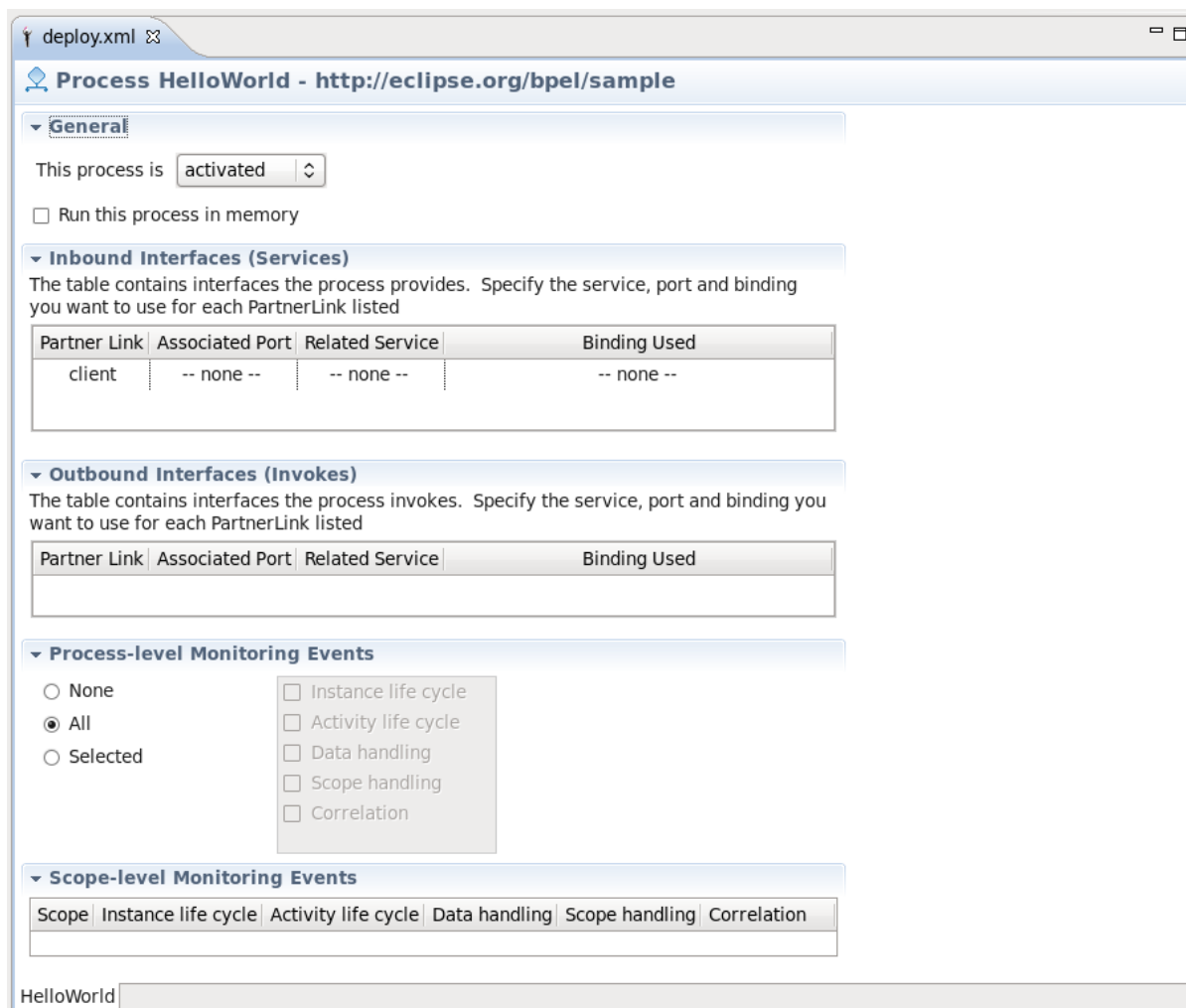


Figure 6.13. Diagram 3

[Report a bug](#)

6.12. CREATING JBOSS BPEL SERVER

It is necessary to add a server when you are setting up BPEL. Your BPEL processes call upon the server when they are being executed. Once it is set up, you are able to access the BPEL console in your web browser to monitor the processes.

1. Open the Servers view by selecting **Windows** → **Show View** → **Other** → **Server** → **Servers**.
2. Right-click the Servers view and select **New** → **Server** to open the New Server wizard.
3. Select **JBoss EAP 6** as a server type.
4. Click the **Next** button. On this page, input your JBoss EAP location. Then click **Next**.
5. Select **HelloWorld**, then click the **Add** button to add the project to the server. Finally, click the **Finish** button.
6. Start the server by right-clicking on the server and selecting the **Start** item.
7. Enter the link <http://localhost:8080/bpel-console/app.html> in your web browser to access the deployed processes.

[Report a bug](#)

6.13. CREATING CORRELATION SETS

Correlation sets provide a standardized set of properties that can be used for different messages. This way you do not have to set properties on each and every message. It allows for asynchronous messaging and normalization.

1. To create a correlation for a messaging activity, go to the dashboard tab **Correlation Sets** and click the plus button. Set a name for the set when prompted.
2. In **Properties** view, click the **Details** tab and then click the **Add** button. The **Select a Property** dialog box appears.
3. Enter a name for the new WSDL property and its type. (Either an XSD simple type or an XML Schema element.)
4. Click the **Browse** button to select a type. The **Type Selection** dialog box appears.
5. Click **New** in the **Aliases** section to create a new WSDL property alias.
6. Select either the **Message Type**, **XSD Simple Type** or XML scheme **Element** radio button and click **Browse** to select its type. Click **OK**.
7. A correlation can be assigned to a messaging activity (for example, Invoke, Receive, Reply). Select the activity, click **Add** on the **Correlation** property tab and choose the appropriate correlation set.

[Report a bug](#)

6.14. EXAMPLE BPEL COMPONENT IMPLEMENTATION

Here is an example of the BPEL component section of a SwitchYard configuration:

```
<sca:component name="SayHelloService">
  <bpel:implementation.bpel process="sh:SayHello" />
  <sca:service name="SayHelloService">
    <sca:interface.wsd1
interface="SayHelloArtifacts.wsd1#wsdl.porttype(SayHello)"/>
    </sca:service>
  </sca:component>
```



WARNING

The above example is for illustrative purposes only. It is best practice to edit configurations in the SwitchYard editor plug-in. Red Hat discourages you from manually editing the XML files.

The BPEL component contains a single *implementation.bpel* element that identifies the fully qualified name of the BPEL process. The component may also contain one or more service elements defining the WSDL port types through which the BPEL process can be accessed.

[Report a bug](#)

6.15. STRUCTURE OF A SWITCHYARD BPEL APPLICATION

The following image shows the structure of the *say_hello* SwitchYard BPEL quickstart:

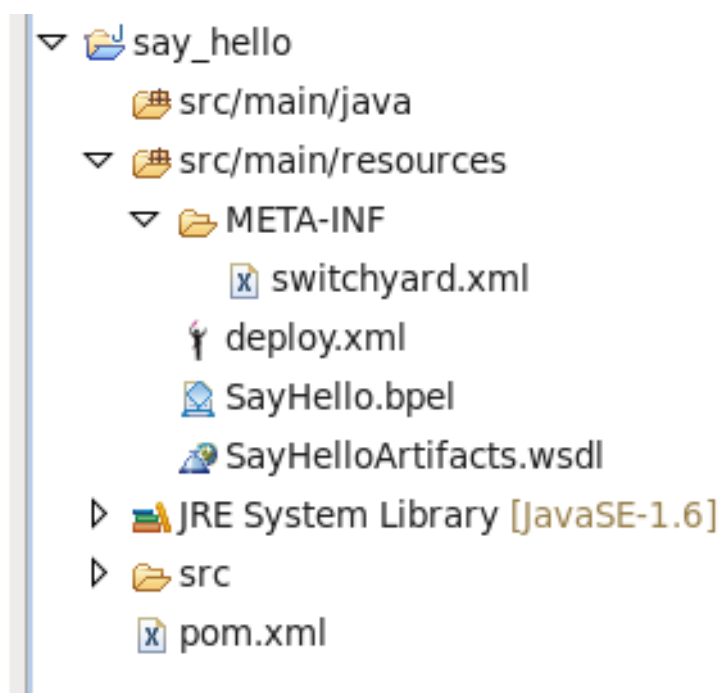


Figure 6.14. BPEL application structure

- As shown, the artifacts are structured within the *src/main/resources* folder.
- The `switchyard.xml` configuration file is located in the META-INF folder.
- The BPEL deployment descriptor (`deploy.xml`) and the BPEL process definition are located in the root folder.
- The WSDL interface definitions (and any accompanying XSD schemas) are located in sub-folders. The BPEL process and SwitchYard BPEL component configuration must define the correct relative path.

[Report a bug](#)

6.16. WIZARDS

Table 6.4. Wizards

Wizard name	Description
New BPEL Project wizard	Creates a faceted project which can be deployed to the JBoss Riftsaw runtime engine. It is available by selecting File → New → Other → BPEL 2.0 → BPEL Project . The <code>bpelContent</code> folder contains all the files necessary for your project.

Wizard name	Description
New BPEL Process File Wizard	Creates a BPEL process based on one of several templates defined by the wizard. The wizard assumes the new BPEL process is to be created in the current project of the Project Explorer or Navigator view. If a BPEL process of the same name already exists within the project, a warning message is displayed before any action is performed.
New BPEL Deployment Descriptor	Use this wizard to create a Deployment Descriptor file. This file is a manifest for the web service and is required if the BPEL process is to be deployed to a runtime engine. The BPEL Deployment Descriptor Editor opens once this wizard is completed.



IMPORTANT

BPEL artifacts must be contained somewhere within the **bpelContent** folder hierarchy if you intend to deploy the process. Complex projects may be organized into a folder hierarchy, but these folders must be contained within **bpelContent**.

The **Deployment Descriptor** file must be contained within the **bpelContent** folder and at the root of any folder hierarchy.

[Report a bug](#)

6.17. VIEWS

Table 6.5. Views

View	Description
Outline	The Outline view provides a structural layout of the BPEL process. You can view the process as either a hierarchical tree-structured outline or as a thumbnail view by pressing the associated button.

View	Description
Palette	<p>The primary editing, creation and viewing tools of the BPEL Designer are accessed from the Palette. The Palette can be docked either at the right or left edge of the BPEL Designer main window, or it can be detached and displayed in its own view.</p> <ul style="list-style-type: none"> • The Selection Tool is used to select individual activities in the editors drawing canvas. Multiple activities can be selected by holding the CTRL or SHIFT keys in combination with left mouse click. The Marquee Tool allows selection of groups of activities by dragging a selection rectangle around them. • BPEL activities are created by dragging icons from the labeled Actions, Controls and Faults palette sections (or drawers), onto the editor's drawing canvas. These sections can be collapsed and expanded by clicking on individual palette section titles. They can also be <i>pinned</i> to prevent them from collapsing if another section is expanded. • The tools at the bottom of the Palette are used to expand or shrink the drawing canvas.
Dashboard	<p>This panel is embedded in the BPEL Designer canvas and provides a quick overview of the BPEL elements that are defined for the currently selected activity or BPEL process. The process name appears at the top of the Dashboard. The main Dashboard area lists all of the Partner Links, Variables, Correlation Sets and Message Exchanges currently defined for the process. The green plus symbol and grey x symbol allow you to add and delete each of these elements. In-line editing of all element names works by selecting the name and then clicking again to enable the editor.</p>

[Report a bug](#)

6.18. PROPERTY SECTION TABS

Table 6.6. Property Sections

Name	Description
------	-------------

Name	Description
Description tab	The Description tab contains the activity name. Names must follow XML element naming conventions, limiting characters to letters, numbers and certain special characters only (spaces are not permitted).
Join Behavior tab	Join conditions are evaluated by the target activities of links. With the drop-down Expression language menu, enter an XPath expression that defines the condition of the join. The Suppress Join Failure behavior defined by the process or a containing scope can be overridden with the radio buttons at the bottom.
Correlation tab	The Correlation tab lists all correlations that are used by the currently selected Receive , Reply or Invoke activity. Correlations can be added to or removed from the activity through this tab.
Namespaces tab	Namespaces are URIs (Uniform Resource Identifiers) that uniquely identify a set of resources on the Internet. Shorthand aliases called prefixes are typically used in XML files to make them more readable. The Namespaces tab lists all of the namespace URIs and their prefixes in scope for the currently selected activity. Whenever you create a reference to an external property (an element defined in an XSD) whose namespace has not yet been assigned a prefix, the BPEL Designer prompts you to create a prefix. This can also be done through the Namespace tab of the Properties sheet for the property by clicking the Assign Prefix button.
Message Exchange tab	Message exchanges are used to associate a Reply activity with an inbound message activity and can be either a Receive , OnMessage or OnEvent . These are descriptive names given to a request-response conversation between two parties and must conform to XML element naming conventions.

[Report a bug](#)

6.19. PROCESS PROPERTY SHEET TABS

Table 6.7. Process Property Sheet Tabs

Name	Description
Description tab	The Description tab allows you to change the process name and its namespace URI.
Details tab	The Process Details tab allows you to select the default Expression and Query language. If you set Exit on Standard Fault to Yes , it causes the process to terminate if a WS-BPEL standard fault, other than a join failure, is encountered. Currently only XPath 1.0 is supported.
Join Behavior tab	The Process Join Behavior tab determines how the process handles the join failures. When set to Yes , any JoinFailure fault is ignored for all activities in the process. An activity is able override this value or inherit the value from its parent.
Imports tab	The Imports Detail tab lists all of the imported service interfaces (WSDL) and XML Schemas (XSD) used by the process. Additional WSDL and XSD files can be added to the imports on this page. After a new resource has been imported, you may assign a prefix to the namespace URI from the Namespaces tab. Imported resources must be located in the project root folder (bpelContent by default) or in a sub-folder.
Namespace tab	Here you can enter a name for your project.
Documentation tab	Click this tab to view relevant documentation pertaining to your process.

[Report a bug](#)

6.20. DETAILS TAB OPTIONS

Table 6.8. Details Tab Options

Name	Description
Partner Links	Partner Links help define the conversations between two services. They define the roles each partner plays in the conversation and the types of messages that can be exchanged between them. The Details tab allows you to choose the Expression language and Query language for selecting elements of aPartner Link .

Name	Description
Variables	Variables are used in BPEL to store inbound and outbound messages for examination and manipulation by the business logic. They can also be used to save intermediate results and the process state. The three kinds of variable declarations are messages types, XML Schema types and XML Schema elements. The Details tab allows you to define the variable declared type and its structure by selecting from known types. Once a variable type has been defined, the structure of the variable is shown. Clicking on the hyperlink opens the WSDL or XML Schema editor for the selected type or element.
Empty	The Empty activity is a placeholder for any undefined Basic Activity and is intended to eventually be replaced by a real activity before the process can actually be executed. If the BPEL engine encounters an Empty activity, it is ignored. The Details tab allows you to select one of four basic actions: Invoke , Receive , Reply and Assign . Hovering the mouse over one of the selection buttons displays a brief description of that activity.
Invoke	The Invoke activity requires a Partner Link name and an Operation as defined in the WSDL for that service. You can use the Quick Pick tree control on the right to select the Partner Link and Operation . For one-way invocations, specify only anInput Variable . For request-response invocations you must also specify an Output Variable . The check box labeled Use WSDL Message Parts Mapping provides an alternative to using variables for the request message.
Receive	A Receive activity requires a Partner Link name and an Operation as defined in the WSDL for this service. You can use the Quick Pick tree control on the right to select the Partner Link and Operation . A previously defined variable can be used to hold the message data, or the Use WSDL Message Parts Mapping check box can be set to store the incoming message in an anonymous WSDL message variable. The Create a new Process Instance check box, when enabled, causes the BPEL engine to start a new process. This starts a new conversation with a client.

Name	Description
Reply	A Reply activity requires a Partner Link name and an Operation as defined in the WSDL. You can use the Quick Pick tree control at the right to select the Partner Link and Operation . A previously defined variable can be used to provide the response message data, or the Use WSDL Message Parts Mapping check box can be set to use the data from the anonymous WSDL message variable.
Opaque	Opaque activities are only used in abstract processes and are meant as placeholders for other activities yet to be determined. When you drag and drop an Opaque activity onto the drawing canvas, the process is converted to a non-executable, abstract process.
Assign	The Assign section contains an array of variables including message options and management buttons. Additional type selection or data entry widgets appear below the From and To combo boxes, depending on the source and target item categories selected in the combo boxes. Initially these are controls for the selection of process variables, since the default combo box selection is Variable .
Validate	The Validate details tab contains a list of variables to be validated.
While and RepeatUntil	These activities have the same details tab, which allows you to specify an XPath expression to be evaluated for the conditional activity.
Link	The Link detail tab allows you to specify a condition that causes Flow synchronization to be satisfied and allow the target activity to continue. This is similar to the details tab of the other conditional activities.
Pick	The Pick tab allows you to specify whether the event creates a new process instance.
OnMessage	The OnMessage activity is used in Pick and event handlers. The Details tab allows you to specify the Partner Link , Operation and Message Type expected by the activity, and the process variable that contains the received message data.

Name	Description
OnAlarm	<p>The OnAlarm activity is used in either a Pick or event handler to handle timeouts while waiting for messages to arrive. This activity can be configured to wait for a certain period of time or until a specific date and time. The Details tab allows you to specify the Partner Link , Operation and Message Type expected by the activity, and the process variable that contains the received message data. Repeat conditions are only allowed for anOnAlarm in an event handler. This allows the activities enclosed in the activity to be executed repeatedly. Repeat duration is the amount of time the process waits before each repetition.</p>
ForEach	<p>ForEach allows you to specify a counter variable to be used for keeping track of the loop iterations. The Parallel execution check box executes all iterations in parallel. The Counter Values tab is where the starting and ending counter values are specified. The optional Completion tab allows you to specify the early termination condition for the loop.</p>
Wait	<p>The details tab of the Wait activity allows you set a delay (Duration) or specify a date and time to continue process execution.</p>
Scope	<p>The details tab for the Scope activity allows you to define whether the Scope is isolated.</p>
Throw	<p>The Throw activity invokes a fault handler in an enclosing Scope activity. Throw requires the name of either a standard BPEL fault, or the name of a user-defined fault message. A variable is used to hold the value of the fault data.</p>
CompensateScope	<p>The CompensateScope activity invokes a compensation handler in the Scope or the Invoke activity given by the name of the Target Activity.</p>

[Report a bug](#)

6.21. BPEL DESIGNER FEATURES

Table 6.9. BPEL Designer Features

Name	Description
<i>Drawing Canvas</i>	Contains the graphical representation of the BPEL process and is displayed when the Design tab at the bottom of the editor window is selected. Clicking on any of the activity names activates an in-line editor, allowing you to edit the process name. To finish editing, press the ENTER key or change focus by clicking on a different window control.
Source	This tab displays the XML (text) representation of the process. Any changes made in one view are shown in the other. The default layout of activities is top-to-bottom, but can be changed to horizontal layout from the context menu.
<i>Palette</i>	The primary editing, creation and viewing tools of the BPEL Designer are accessed from this tool.
<i>Dashboard</i>	Provides an overview of the BPEL process.
<i>Property Sheet</i>	Displays the properties of an activity when it is selected in the drawing canvas.
<i>Outline</i>	This panel provides a structural view of the BPEL process.

[Report a bug](#)

6.22. BPEL DESIGNER CONCEPTS

Table 6.10. BPEL Designer Concepts

Name	Description
Assign error	Hovering your mouse over this icon displays an error message.
Basic activities	Basic activities are represented on the drawing canvas as rounded rectangles containing an icon and the user-defined name of the activity. The Actions section of the Palette contains all of the basic activities. For example: Assign , Invoke and Receive .
Start and End	Every process has Start and End activities that act as placeholders for visualizing the beginning and end of the process flow.

Name	Description
Assign activity	The Assign activity allows you to manipulate variables and message contents that are defined in the process.
Invoke	The Invoke activity is used to send a message to an external service (one-way invocation) and, optionally, wait for a response (request and response). An Invoke can also define a compensation handler and a fault handler to handle exception conditions.
Receive	The Receive activity waits for a specific message type from a service client.
Reply	The Reply activity is used to respond to clients with a specific message type or a fault message.
Validate	The Validate activity is used to validate the values of variables against their XML Schema and WSDL data definitions. This includes the variable's data type as well as structure. If validation fails, the BPEL standard fault <code>invalidVariables</code> is thrown. Validation is typically performed just before sending messages to a partner or client, or after receiving a message to ensure the message contains all required data.
Wait	The Wait activity delays process execution for a certain amount of time, or until a given date and time. This is typically used to invoke an operation at a certain time. For example to update process state hourly or daily, or to collect some information from another service at a certain time.

[Report a bug](#)

6.23. BPEL DEPLOYMENT DESCRIPTOR EDITOR PROPERTIES

Table 6.11. BPEL Deployment Descriptor Editor Properties

Name	Description
Process selection tabs	Click on these tabs to display the configuration page for each process.
Initial process state	The process can be deployed in either an <i>active</i> , <i>inactive</i> or <i>retired</i> state.

Name	Description
Inbound interfaces selection	Select the WSDL port type that clients use to invoke this service.
Output interfaces selection:	Each invoked service (if any) requires you to select its port type.
Scope-level monitoring events	The BPEL engine can be configured to generate monitoring events for each Scope defined in the process.



NOTE

Before a BPEL project can be deployed to the runtime engine, you must create what is called a *deployment descriptor*. This is simply a manifest file, serialized as XML, that describes all of the BPEL processes and their interfaces to the BPEL engine. The *deployment descriptor* file must be created in the root folder of your project.

[Report a bug](#)

6.24. DIALOGS

Table 6.12. Dialogs

Name	Description
XPath expression editor (embedded control)	The XPath expression editor provides context-sensitive assistance in the form of pop-up proposals. The light bulb icon indicates that content assist is available by pressing the CTRL and SPACE keys simultaneously. The BPEL 2.0 specification provides for the definition of an XPath language version at the process level, as well as the activity level (for those activities that make use of XPath). However, only XPath 1.0 is supported by the BPEL Designer and the JBoss Riftsaw runtime engine.
Quick pick (embedded control)	Tree control is used in many property pages for selecting message parts, partner links and operations.

Name	Description
Type selection	<p>This dialog is displayed whenever the BPEL Designer requires you to select a message, message part, XML Schema type or XML element.</p> <ol style="list-style-type: none">1. Type Name: Used to limit the items displayed in the Matches (4) list. Only items that begin with the text in this filter are displayed.2. Show XSD Types: Can be used to limit where the editor searches for XSD files.3. Filter: Further reduces the number of matches according to types.4. Matches: Displays the items matching the selected filters. Selecting an item in this list updates the Type Structure (5) tree view.5. Type Structure: Displays the structure of the item selected in the Matches (4) list. Depending on the type of item requested, you may need to select an item from this tree control as well; the OK button being enabled is an indicated of a selection being required here.6. Add Schema: If the required XML Schema has not been resolved, you can add it to the process' imports by clicking this button.

[Report a bug](#)

6.25. CHEAT SHEETS

The Cheat Sheets option allows you to view tutorials. This helps you teach yourself how to use JBoss Developer Studio and its plug-ins. By playing around with them, you can gain a better understanding of how the studio works and what you can do with it.

Procedure 6.3. Task

1. Access the cheat sheet by clicking **Help** → **Cheat Sheets**.
2. The cheat sheet opens in a separate view as show below. Click on the **Click to begin** link to begin.

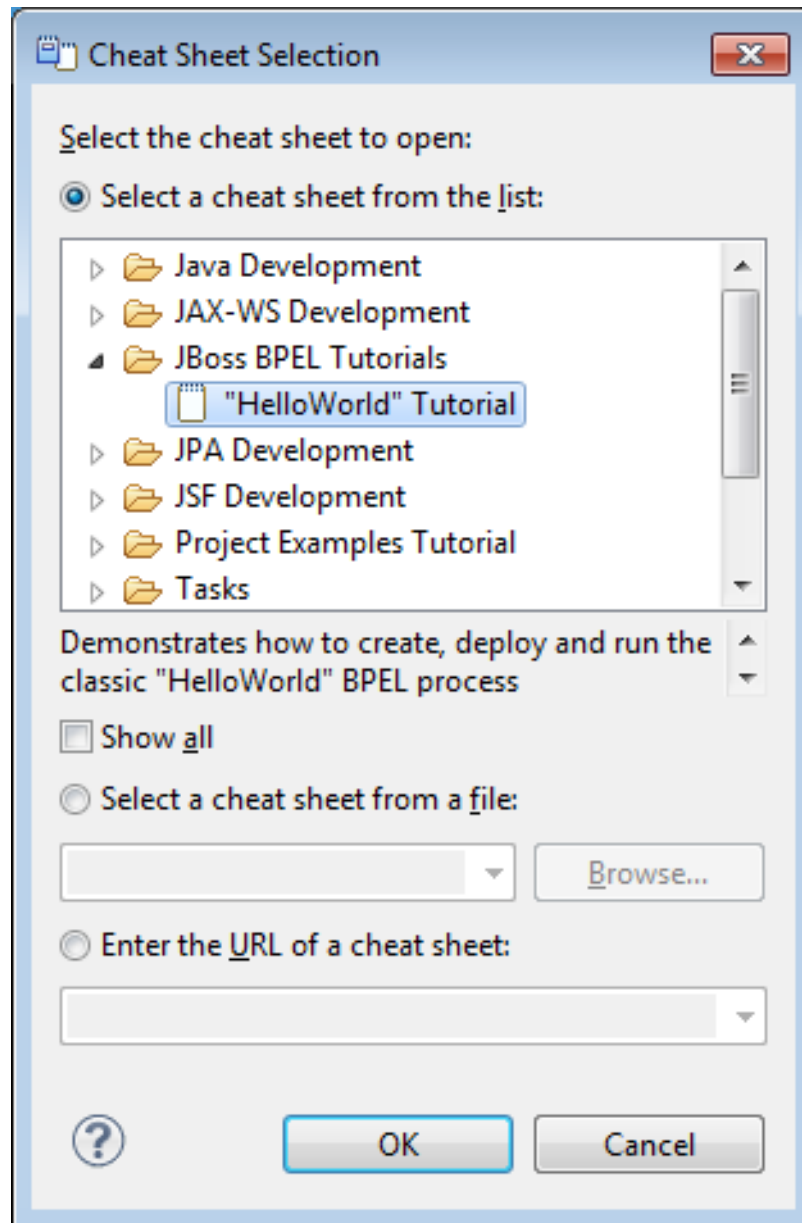


Figure 6.15. Diagram 1

3. You can now view tutorials for plug-in tools.

[Report a bug](#)

6.26. CONTEXT MENU

Access the context menu by right-clicking over an activity figure on the drawing canvas. A sub-menu appears. The items within the **Add** sub-menu appends a new activity after the current one while those within the **Insert Before** sub-menu inserts the new activity before the current one.

[Report a bug](#)

CHAPTER 7. JBOSS DEVELOPER STUDIO GUVNOR TASKS

7.1. GUVNOR REPOSITORY EXPLORING PERSPECTIVE

There are different perspectives which you can use to view your files in JBoss Developer Studio. You don't have to have only one perspective open at all times. You can switch between them to access different plug-ins in the same session.

1. In the JBoss Developer Studio workbench menu, select **Window** → **Open Perspective** → **Other** to view the perspective list.
2. Choose the perspective you wish to use. JBoss Developer Studio standard views such as **Properties** and **Resource Navigator** are useful.

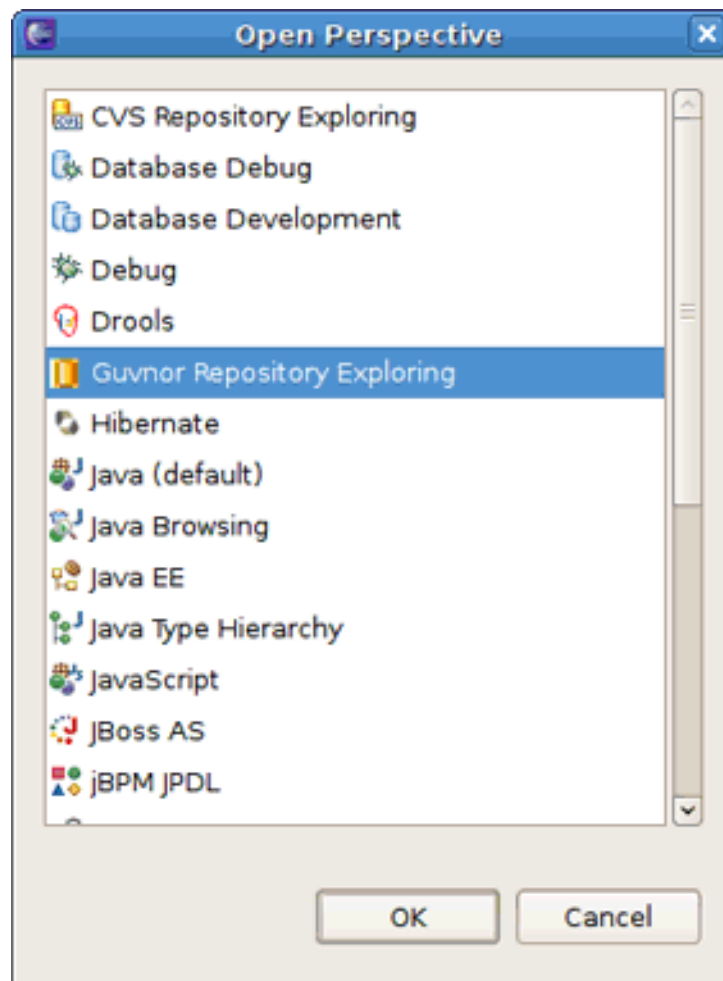


Figure 7.1. Perspective list

[Report a bug](#)

7.2. CREATING A NEW GUVNOR CONNECTION

JBoss Developer Studio comes with a wizard for creating Guvnor connections. Adding new connections allows you to configure repositories and access Guvnor files. You can modify these in JBoss Developer Studio.

1. Open a new connection.
2. Start the **Guvnor Connection wizard** in one of the following ways:

- o Select **File** → **New** → **Other** → **Guvnor** → **Guvnor repository location** within JBoss Developer Studio.
- o Go to the **Guvnor Repositories** view using the drop-down menu.
- o Use the menu button to select a new connection. This is added to JBoss Developer Studio automatically.

[Report a bug](#)

7.3. CONFIGURING THE GUVNOR CONNECTION WIZARD

Once you have created a new Guvnor connection, you can configure it. Most of the fields have default values already filled out, but you are able to change these to be more applicable. You can list your Guvnor repository URLs here.

1. Default values appear in the **Location**, **Port** and **Repository** fields. You can edit these fields using their corresponding text boxes.
2. To input a Guvnor repository URL into the respective fields, drag and drop it into the **Location** field.



IMPORTANT

When using JBoss Developer Studio tooling, you cannot access Guvnor data if this Guvnor is deployed on a secured HTTP (HTTPS).

3. The EGT calls the Guvnor repository at various times, such as when it is determining if any resource updates are available. If you use session authentication, the authentication dialog appears at different times during the JBoss Developer Studio workbench session, depending on what actions you take. We recommend saving the authentication information in the JBoss Developer Studio keyring.

Click on **save username** and **password** to save your details into the JBoss Developer Studio workbench's keyring file.



NOTE

If the authentication information is not stored in the keyring, then the EGT uses a session authentication. This means that the credentials supplied are only used during the lifetime of the JBoss Developer Studio workbench instance.



NOTE

If an authentication failure error occurs, you can retry the same operation and supply different authentication information.



NOTE

The JBoss Developer Studio keyring file is distinct from keyring files found in some platforms such as Mac OS X and many forms of Linux. Thus, sometimes if you access a Guvnor repository outside the EGT, the keyring files might become outdated, and you are unexpectedly prompted for your credentials in JBoss Developer Studio. Enter your username and password to continue.

[Report a bug](#)

7.4. GETTING LOCAL COPIES OF GUVNOR FILES

You can import Guvnor files on your hard drive into JBoss Developer Studio. This is where you can edit them and view their details. The file tree is displayed in the Resource Navigator so you are able to see all the information pertaining to the file you have selected.

1. To get local copies of Guvnor repository resources, you can drag-and-drop them from the Guvnor Repositories View. You can also perform this action in the Guvnor Wizard.
2. Local copies of Guvnor repository files are created. The EGT sets an association between the local copy and the master file in the repository. You should not alter this information. The association allows for operations such as update and commit in synchronization with the master copy held in the Guvnor repository.

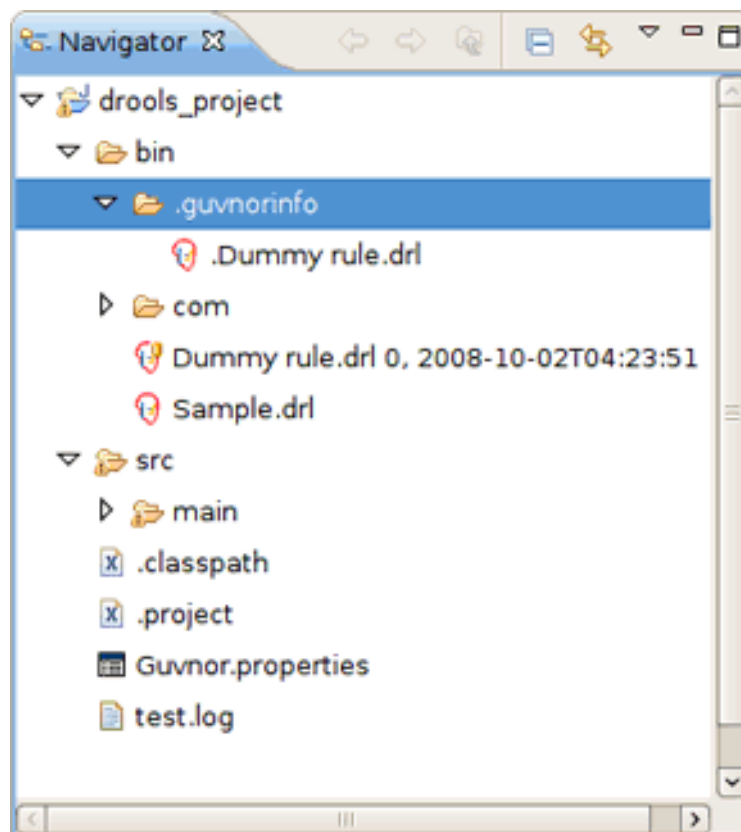


Figure 7.2. Diagram

The EGT decorates local resources associated with Guvnor repository master copies. This decoration appears in JBoss Developer Studio views conforming to the JBoss Developer Studio Common Navigator framework, such as the JBoss Developer Studio **Resource Navigator** and the **Java Package Explorer**. The **Dummy rule.drl** file with the decoration

in the **Resource Navigator**. The Guvnor icon decorator is on the top right of the file image, and the Guvnor revision details are appended to the file name. You can change the location of these in the **Local Guvnor Repository Resource Decoration Preferences**.

3. You can see that the **Dummy rule.drl** file is associated with a Guvnor repository resource, and the local copy is based on revision 0, with a **02-10-2008, 4:21:53** date/time stamp. However, the file **Sample.drl** is not associated with a Guvnor repository file.
4. The EGT adds a property page to the standard JBoss Developer Studio properties dialog. Use it to view the specific Guvnor repository, the location within the repository, the version (date/time stamp) and the revision number.

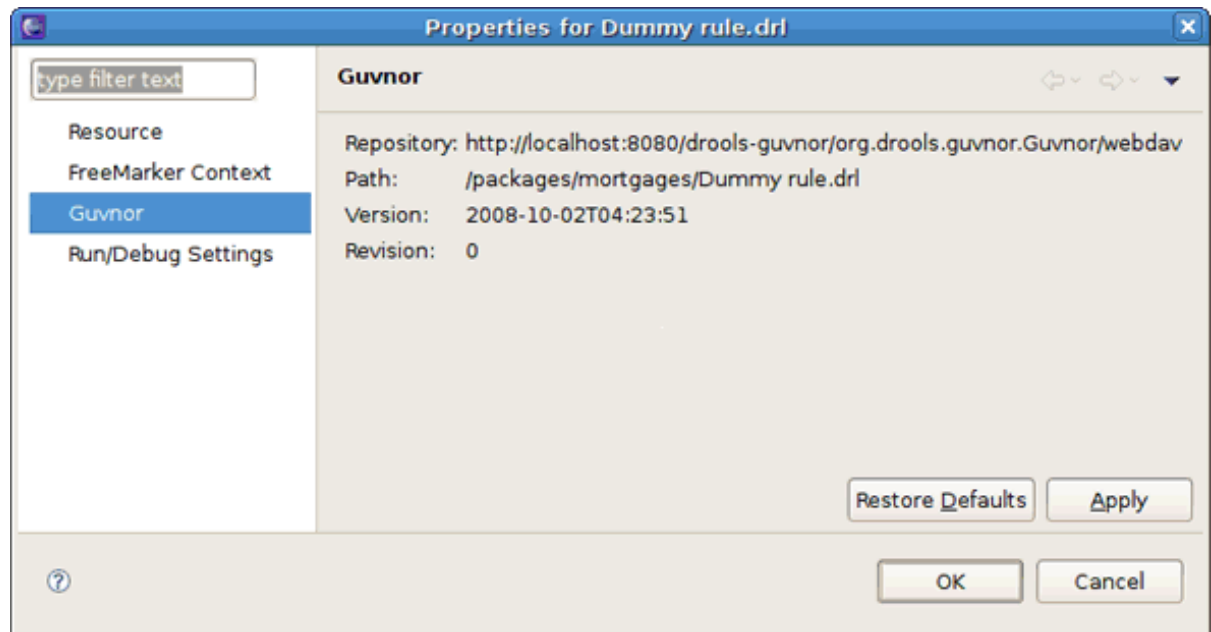


Figure 7.3. Properties

[Report a bug](#)

7.5. MANAGING GUVNOR RESOURCES

The EGT provides a number of actions (available through the Guvnor context menu on files) for working with files, both those associated with Guvnor repository master copies and those not associated. The actions are:

Update Action

Available for Guvnor resources that are not synchronized with the Guvnor repository master copies. These resources may not have synchronized due to local changes and/or changes to the master copy. Performing the Update action replaces the local file contents with the current contents from the Guvnor repository master copies.

Add Action

Available for local files that are not associated with a Guvnor repository master copy. Choosing the Add action launches the Add in Guvnor wizard. The first page of the wizard asks for the selection of the target Guvnor repository and provides the option of creating a new Guvnor repository connection. Once the target Guvnor repository is chosen, the wizard asks for the folder location to add the selection files. Clicking on the Finish button adds the selected files to the Guvnor repository and creates an association between the local and Guvnor repository files.

**NOTE**

The snapshot folder in the Guvnor repository is read-only for EGT, and hence not visible as a candidate location in this wizard. The Guvnor repository web administration tools must be used to add snapshot content.

**NOTE**

The wizard does not allow existing Guvnor repository files to be overwritten. Another target location must be chosen.

Commit Action

Enabled for Guvnor repository associated files that have local changes. The Commit action writes the changes back to the associated Guvnor repository files and update the association for the new revision created.

If a local change is based on an older revision of a file than is currently in the Guvnor repository, the Commit action asks whether you wish to overwrite the current version in the Guvnor repository with the local content. When such conflicts occur, you should use the JBoss Developer Studio Guvnor version tools and the JBoss Developer Studio standard tools to determine the differences and merge content based on the current version.

Show History Action

Populates the Guvnor Resource History View with revision history for the selected file.

Compare with Version Action

Enabled for singular Guvnor repository associated files. This action first opens a wizard asking for the version for comparison (with the local file contents). Once the revision is selected, the action opens the JBoss Developer Studio Compare editor (in a read-only mode) which shows the differences in the two versions.

Switch to Version Action

Enabled for singular Guvnor repository associated files. It first prompts for the selection of a version. Once the version is selected, the Switch to Version action replaces the local file contents with those from the revision selected.

Delete Action

Enabled for one or more Guvnor repository associated files. After the action has been confirmed, the Delete action removes the files in the Guvnor repository and deletes local metadata for the Guvnor repository association.

Disconnect Action

Enabled for one or more Guvnor repository associated files. Removes local metadata from the Guvnor repository association.

[Report a bug](#)

7.6. RESOURCE FROM GUVNOR WIZARD

This wizard allows you to copy files to your local workspace from the Guvnor repository. Once added, you can access them within Guvnor. They are listed as resources which you can call on.



1. Open the wizard by selecting **File** → **Import** → **Guvnor** → **Resource from Guvnor** or **File** → **New** → **Other** → **Guvnor** → **Resource from Guvnor**.
2. You are prompted to enter the source Guvnor repository. You can also create a new Guvnor repository connection.
3. Once the source Guvnor repository is chosen, the wizard prompts for the resources to be copied.
4. Finally, choose the target location for the files in the local workspace.
5. On completion the wizard copies the selected files from the Guvnor repository to the local workspace. If a file with the same name already exists in the destination, you are prompted to rename it.

[Report a bug](#)

7.7. GUVNOR REPOSITORIES VIEW

The **Guvnor Repositories** view contains tree structures representing the contents of Guvnor repositories. You can perform the following actions under the resources in the **Guvnor Repositories** view:

Table 7.1. Guvnor Repositories View Actions

Action	Description
Create a new Guvnor repository connection	Use the wizard to create a new connection by selecting File → New → Other → Guvnor → Guvnor repository location in JBoss Developer Studio.
Remove a Guvnor repository connection	Click Delete () to remove a repository connection.
Refresh Guvnor repository resources.	Use the Refresh context menu item to reload the tree for the selected node.
"Drill-into" functionality	Use icons on the toolbar such as the Home , Go Back and Go Into buttons to navigate deeply nested structures.
Go Home button ()	Returns the tree to the top-level structure.
Selecting a file in the Guvnor repository	Doing this causes the JBoss Developer Studio Properties view to display the details of that file.

[Report a bug](#)

7.8. GUVNOR RESOURCE HISTORY VIEW

The **Guvnor Resource History** view displays details about the revision history of the files you have selected. The **Guvnor Resource History** view is populated by **Show History** actions in either the local **Guvnor** context menu or in the context menu for a Guvnor repository file in the Guvnor Repositories View.

[Report a bug](#)

7.9. CONFIGURING THE GUVNOR RESOURCE HISTORY VIEW

This enables you to see the revision history of your project. If you need to know when part of your project was changed, you are able to go through its history and see when and what modifications were made. This simplifies keeping track of changes.

1. Select **Show History** in the local **Guvnor** context menu. Alternatively, select it in the context menu for a Guvnor repository file in the Guvnor Repositories View.
2. The **Guvnor Resource History** view is refreshed to display the revision history.
3. To open a JBoss Developer Studio read-only editor with the revision contents, double-click on a revision row or select the **Open (Read only)** option from the context menu.
4. Click on the **Save As** option when a file is open in a read-only editor to save a local writable copy of the contents. (This does not associate the file created with its Guvnor source.)

[Report a bug](#)

7.10. GUVNOR PREFERENCES

The EGT provides a preference page in the **Guvnor** category. The preferences cover two categories:

- Guvnor repository connections
- Local Guvnor repository resource decorations

[Report a bug](#)

7.11. GUVNOR REPOSITORY CONNECTION PREFERENCES

There are two preferences that can be set for Guvnor repository connections. These are used when creating new connections:

Table 7.2. Guvnor Repository Connection Preferences

Preference	Description
Default Guvnor repository URL template	Simplifies the process of creating additional similar connections by changing part of the field, such as the host name.

Preference	Description
Authentication information	Defines whether or not authentication information should be saved in the JBoss Developer Studio platform key-ring by default. As with the Guvnor repository URL template, this decision can be determined when creating the connection.

[Report a bug](#)

7.12. LOCAL GUVNOR REPOSITORY RESOURCE DECORATION PREFERENCES

This category of preferences provided by the EGT can be used to define how the decoration of local resources associated with Guvnor repository resources is presented. Since the Guvnor repository is not a substitute for a SCM, and since SCM tools in JBoss Developer Studio tend to decorate local resources, it is useful to be able to control how the EGT decorates its local resources to avoid conflicts with SCM packages.

In the **File Decoration** section of the preference page, you can choose the location (Top right, Top left, Bottom right, Bottom left) of the decoration icon, or you can choose not to display it. In the **Text** section, you can format the Guvnor metadata which is appended to the file names.

- Whether to show an indicator (>) when the local file has changes not committed back to the Guvnor repository
- Whether to show the revision number
- Whether to show the date/time stamp

Any changes to these preferences take effect immediately upon clicking the **Apply** and then **OK** buttons.

[Report a bug](#)

APPENDIX A. REVISION HISTORY

Revision 6.0.0-27 Version number updated from 6 to 6.0 for consistency.	Monday February 23 2015	B Long
Revision 6.0.0-26 Updated for Red Hat JBoss Fuse Service Works 6.0	Tuesday January 13 2015	Nilam Shendye
Revision 6.0.0-25	Wednesday December 17 2014	Anshu Mahajan