JBoss Enterprise SOA Platform 5

BPEL Tools Reference Guide

This guide is for developers

Edition 5.3.1

Last Updated: 2017-10-27

JBoss Enterprise SOA Platform 5 BPEL Tools Reference Guide

This guide is for developers
Edition 5.3.1

David Le Sage
Red Hat Engineering Content Services
dlesage@redhat.com

Legal Notice

Abstract

This guide teaches developers to use JBDS' BPEL plug-in.

Table of Contents

PREFACE

CHAPTER 1. PREFACE

1.1. BUSINESS INTEGRATION

In order to provide a dynamic and competitive business infrastructure, it is crucial to have a service-oriented architecture in place that enables your disparate applications and data sources to communicate with each other with minimum overhead.

The JBoss Enterprise SOA Platform is a framework capable of orchestrating business services without the need to constantly reprogram them to fit changes in business processes. By using its business rules and message transformation and routing capabilities, JBoss Enterprise SOA Platform enables you to manipulate business data from multiple sources.

Report a bug

1.2. WHAT IS A SERVICE-ORIENTED ARCHITECTURE?

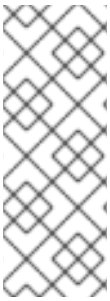
Introduction

A *Service Oriented Architecture* (SOA) is not a single program or technology. Think of it, rather, as a software design paradigm.

As you may already know, a *hardware bus* is a physical connector that ties together multiple systems and subsystems. If you use one, instead of having a large number of point-to-point connectors between pairs of systems, you can simply connect each system to the central bus. An *enterprise service bus* (ESB) does exactly the same thing in software.

The ESB sits in the architectural layer above a messaging system. This messaging system facilitates *asynchronous communications* between services through the ESB. In fact, when you are using an ESB, everything is, conceptually, either a *service* (which, in this context, is your application software) or a *message* being sent between services. The services are listed as connection addresses (known as *end-points references*.)

It is important to note that, in this context, a "service" is not necessarily always a web service. Other types of applications, using such transports as File Transfer Protocol and the Java Message Service, can also be "services."



NOTE

At this point, you may be wondering if an enterprise service bus is the same thing as a service-oriented architecture. The answer is, "Not exactly." An ESB does not form a service-oriented architecture of itself. Rather, it provides many of the tools than can be used to build one. In particular, it facilitates the *loose-coupling* and *asynchronous message passing* needed by a SOA. Always think of a SOA as being more than just software: it is a series of principles, patterns and best practices.

Report a bug

1.3. KEY POINTS OF A SERVICE-ORIENTED ARCHITECTURE

These are the key components of a service-oriented architecture:

1. the *messages* being exchanged
2. the *agents* that act as service requesters and providers
3. the *shared transport mechanisms* that allow the messages to flow back and forth.

Report a bug

1.4. WHAT IS THE JBOSS ENTERPRISE SOA PLATFORM?

The JBoss Enterprise SOA Platform is a framework for developing enterprise application integration (EAI) and service-oriented architecture (SOA) solutions. It is made up of an enterprise service bus (JBoss ESB) and some business process automation infrastructure. It allows you to build, deploy, integrate and orchestrate business services.

Report a bug

1.5. THE SERVICE-ORIENTED ARCHITECTURE PARADIGM

The service-oriented architecture (SOA) consists of three roles: requester, provider, and broker.

Service Provider

A service provider allows access to services, creates a description of a service and publishes it to the service broker.

Service Requester

A service requester is responsible for discovering a service by searching through the service descriptions given by the service broker. A requester is also responsible for binding to services provided by the service provider.

Service Broker

A service broker hosts a registry of service descriptions. It is responsible for linking a requester to a service provider.

Report a bug

1.6. CORE AND COMPONENTS

The JBoss Enterprise SOA Platform provides a comprehensive server for your data integration needs. On a basic level, it is capable of updating business rules and routing messages through an Enterprise Service Bus.

The heart of the JBoss Enterprise SOA Platform is the Enterprise Service Bus. JBoss (ESB) creates an environment for sending and receiving messages. It is able to apply “actions” to messages to transform them and route them between services.

There are a number of components that make up the JBoss Enterprise SOA Platform. Along with the ESB, there is a registry (jUDDI), transformation engine (Smooks), message queue (HornetQ) and BPEL engine (Riftsaw).

Report a bug

1.7. COMPONENTS OF THE JBOSS ENTERPRISE SOA PLATFORM

- A full Java EE-compliant application server (the JBoss Enterprise Application Platform)
- an enterprise service bus (JBoss ESB)
- a business process management system (jBPM)
- a business rules engine (JBoss Rules)
- support for the optional JBoss Enterprise Data Services (EDS) product.

Report a bug

1.8. JBOSS ENTERPRISE SOA PLATFORM FEATURES

The JBoss Enterprise Service Bus (ESB)

The ESB sends messages between services and transforms them so that they can be processed by different types of systems.

Business Process Execution Language (BPEL)

You can use web services to orchestrate business rules using this language. It is included with SOA for the simple execution of business process instructions.

Java Universal Description, Discovery and Integration (jUDDI)

This is the default service registry in SOA. It is where all the information pertaining to services on the ESB are stored.

Smooks

This transformation engine can be used in conjunction with SOA 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 SOA. It can infer data from the messages it receives to determine which actions need to be performed.

Report a bug

1.9. FEATURES OF THE JBOSS ENTERPRISE SOA PLATFORM'S JBOSSESB COMPONENT

The JBoss Enterprise SOA Platform's JBossESB component supports:

- Multiple transports and protocols
- A listener-action model (so that you can loosely-couple services together)

- Content-based routing (through the JBoss Rules engine, XPath, Regex and Smooks)
- Integration with the JBoss Business Process Manager (jBPM) in order to provide service orchestration functionality
- Integration with JBoss Rules in order to provide business rules development functionality.
- Integration with a BPEL engine.

Furthermore, the ESB allows you to integrate legacy systems in new deployments and have them communicate either synchronously or asynchronously.

In addition, the enterprise service bus provides an infrastructure and set of tools that can:

- Be configured to work with a wide variety of transport mechanisms (such as e-mail and JMS),
- Be used as a general-purpose object repository,
- Allow you to implement pluggable data transformation mechanisms,
- Support logging of interactions.

IMPORTANT

There are two trees within the source code: **`org.jboss.internal.soa.esb`** and **`org.jboss.soa.esb`**. Use the contents of the **`org.jboss.internal.soa.esb`** package sparingly because they are subject to change without notice. By contrast, everything within the **`org.jboss.soa.esb`** package is covered by Red Hat's deprecation policy.

Report a bug

1.10. TASK MANAGEMENT

JBoss SOA simplifies tasks by designating tasks to be performed universally across all systems it affects. This means that the user does not have to configure the task to run separately on each terminal. Users can connect systems easily by using web services.

Businesses can save time and money by using JBoss SOA to delegate their transactions once across their networks instead of multiple times for each machine. This also decreases the chance of errors ocurring.

Report a bug

1.11. 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 synchronise well with the existing systems as they currently stand.

The company wants an inexpensive solution but one that will adhere 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.

The JBoss Enterprise SOA Platform 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 the JBoss Enterprise SOA Platform can be updated quickly and easily.

As a result, older systems can now synchronise with newer ones due to the unifying methods of SOA. 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 JBoss Enterprise SOA Platform 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

1.12. UTILISING THE JBOSS ENTERPRISE SOA PLATFORM IN A BUSINESS ENVIRONMENT

Cost reduction can be achieved due to the implementation of services that can quickly communicate with each other with less chance of error messages occurring. Through enhanced productivity and sourcing options, ongoing costs can be reduced.

Information and business processes can be shared faster because of the increased connectivity. This is enhanced by web services, which can be used to connect clients easily.

Legacy systems can be used in conjunction with the web services to allow different systems to "speak" the same language. This reduces the amount of upgrades and custom code required to make systems synchronise.

Report a bug

CHAPTER 2. INTRODUCTION

2.1. INTENDED AUDIENCE

This book is aimed at developers who wish to learn how to utilize the tools in JBoss BPEL. It explains how to create new projects, debug projects and how to use editors.

Report a bug

2.2. AIM OF THE GUIDE

This guide aims to give users an overview of how to use BPEL with projects. Users will be given steps to follow to create projects and edit those projects. They will also learn about different types of editors and how to deal with error messages.

Report a bug

2.3. INSTALLATION

Procedure 2.1. Task

1. Click **Help** -> **Install New Software...** -> **Add...** and insert name and location (for example, https://devstudio.jboss.com/updates/5.0/staging/soa-tooling/). Then click **OK**.
2. Choose *BPEL Editor*, then click **Next** and **Next**.
3. Accept the licence agreement.
4. Click **Finish** and restart your JBDS. (It isn't neccessary to restart computer.)
5. The JBoss BPEL Editor will be availible to you after restart.

Report a bug

CHAPTER 3. TASKS

3.1. CREATING A BPEL PROJECT

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

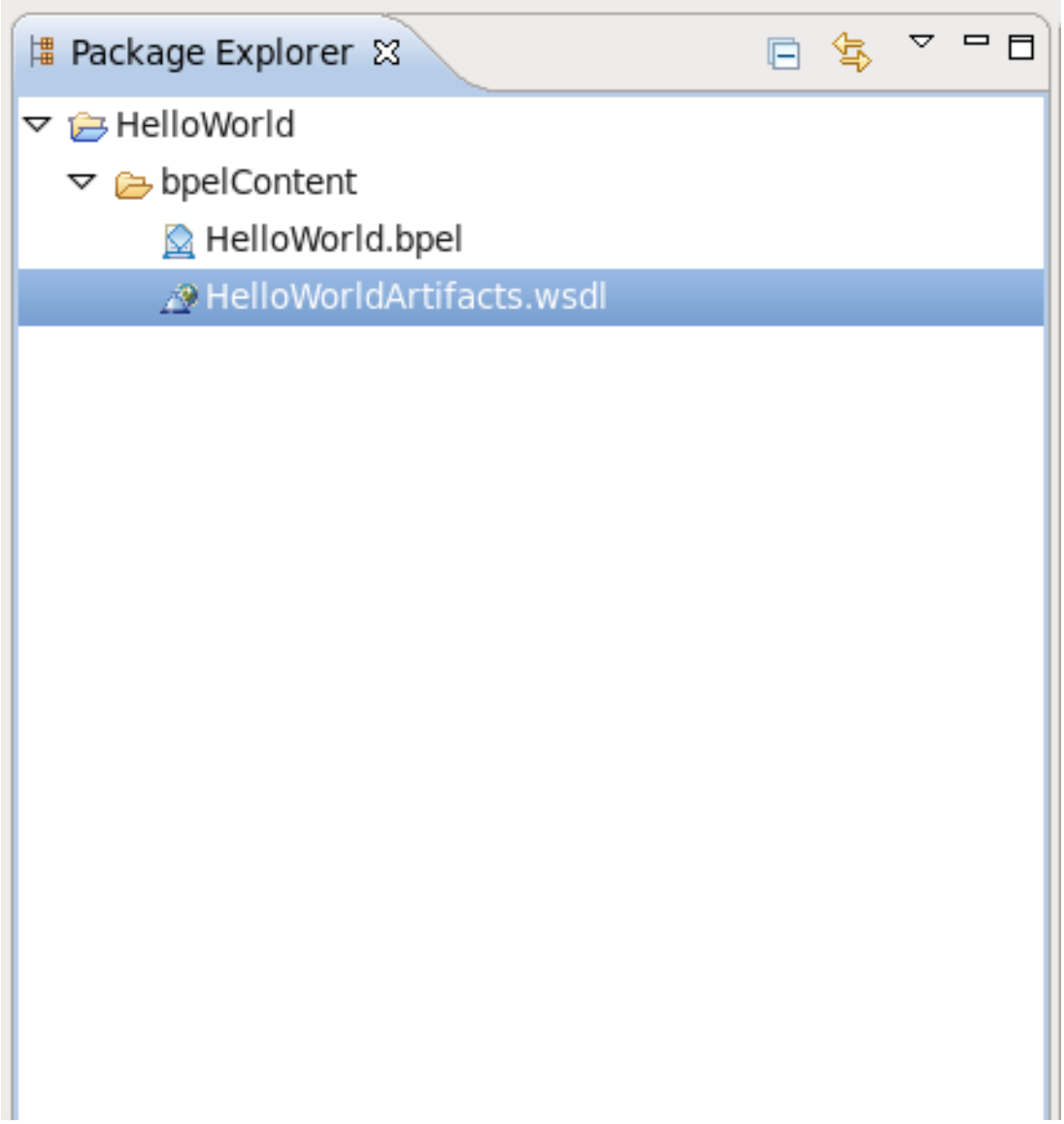


Figure 3.1. Diagram 1

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

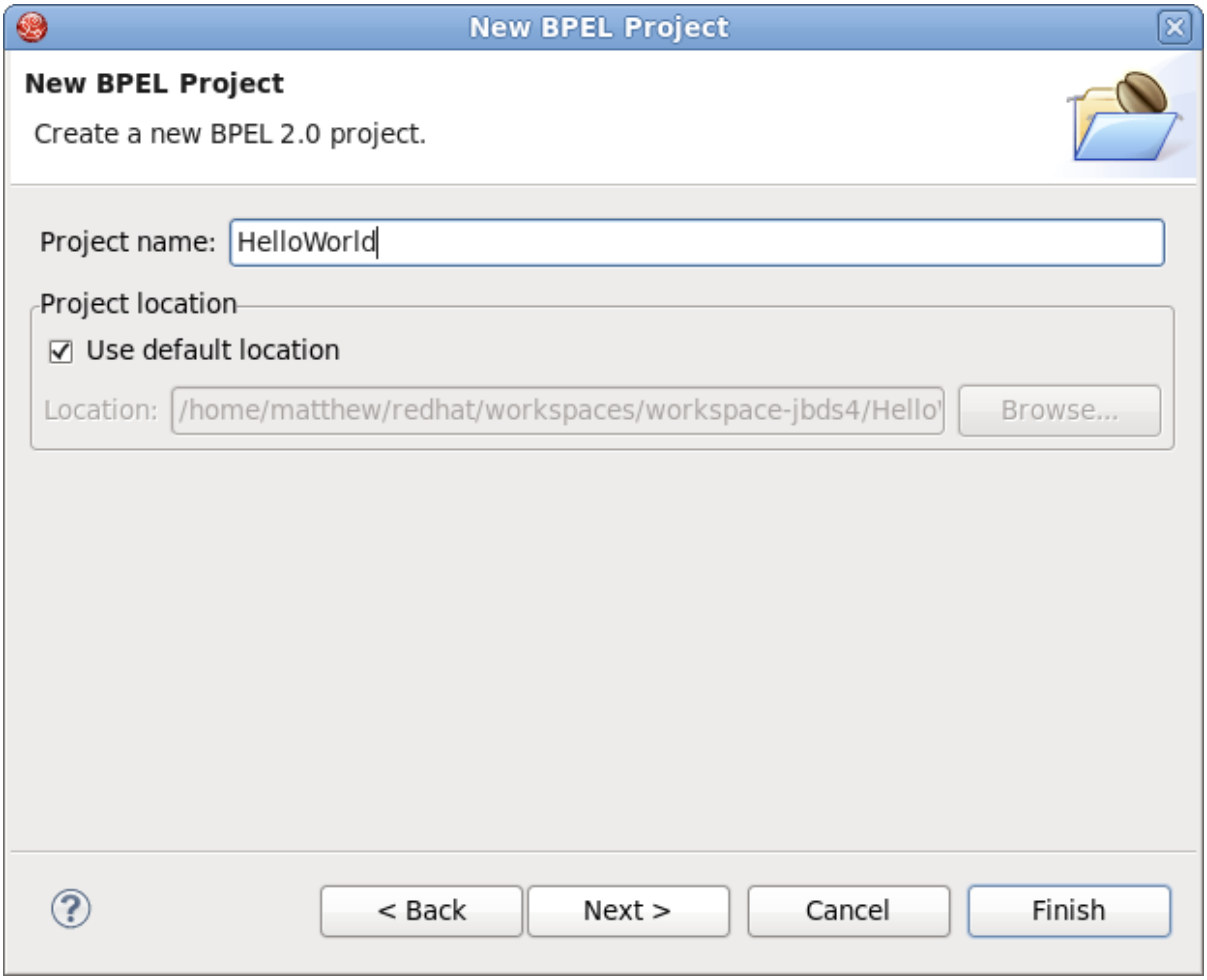


Figure 3.2. Diagram 2

3. Click the **Finish** button. The following screen will appear.

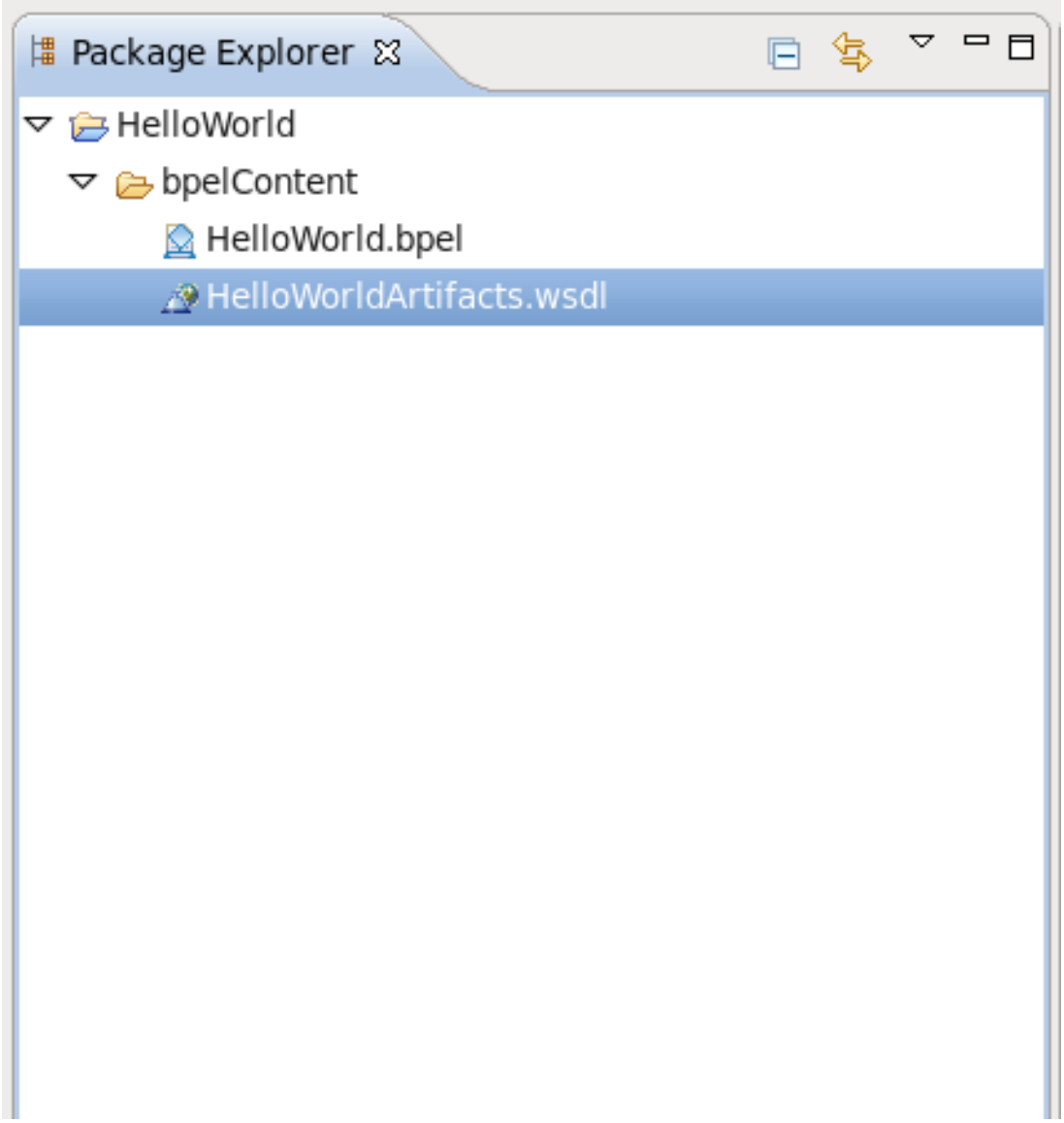


Figure 3.3. Diagram 3

4. You have now created a new project.

Report a bug

3.2. CREATING A BPEL PROCESS

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

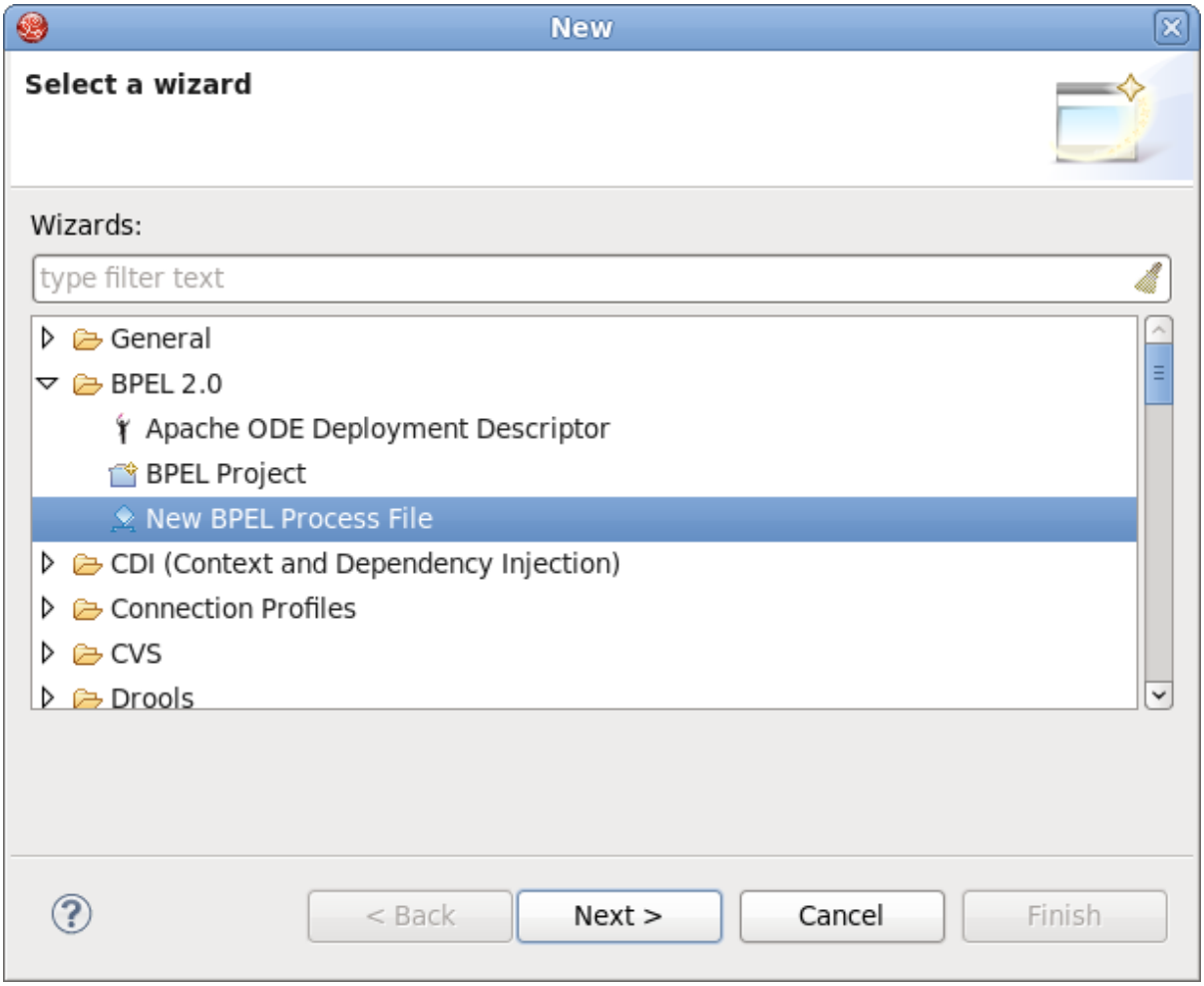


Figure 3.4. Diagram 1

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 3.1. 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 will see information about it. Select **Synchronous BPEL Process**.

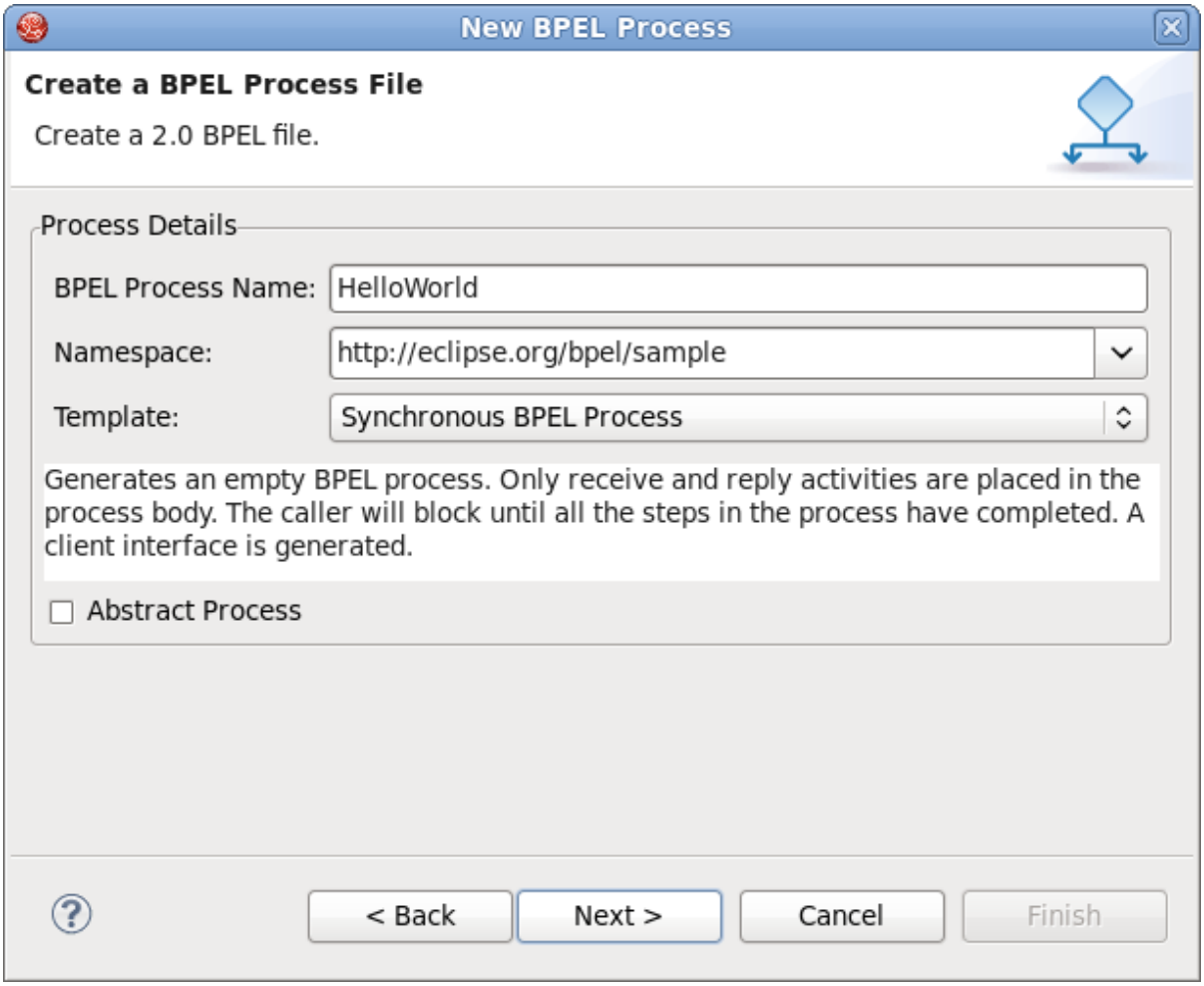


Figure 3.5. Diagram 2

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

Table 3.2. 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**.

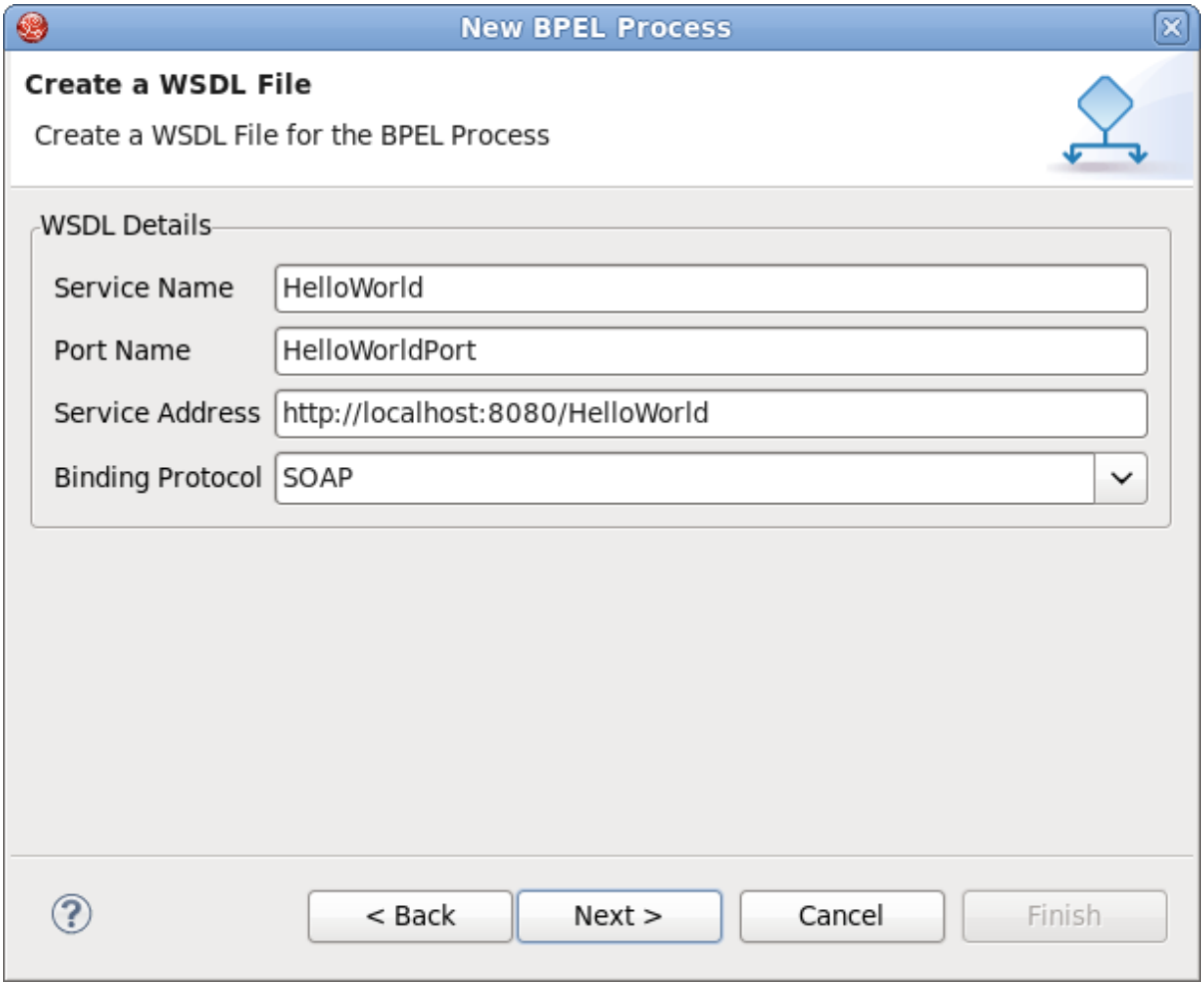
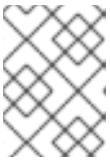


Figure 3.6. Diagram 3

5. Click the **Next** button. On the 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** will be 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

3.3. CREATE A NEW SERVER RUNTIME

Procedure 3.1. 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

3.4. EDITING A BPEL PROCESS FILE

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.

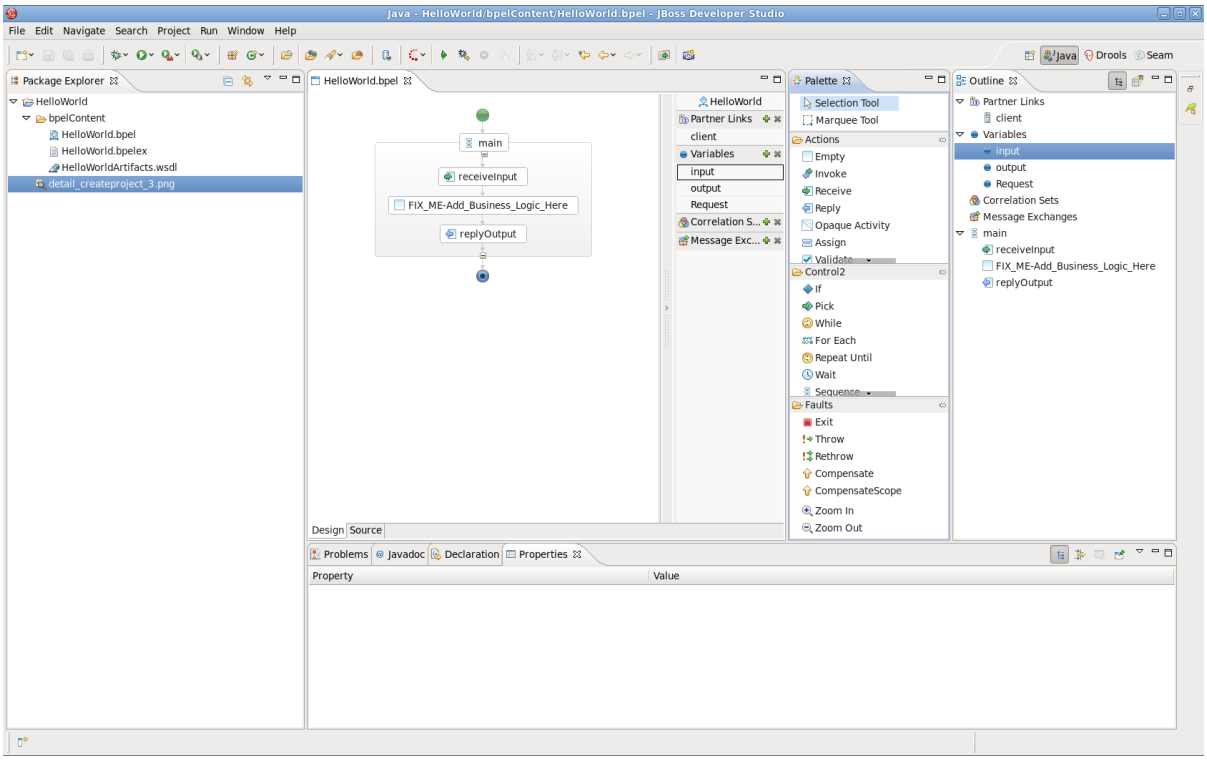


Figure 3.7. Diagram 1

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

3.5. TABS SHOWN IN THE PROPERTIES VIEW

Table 3.3. 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 will be imported into the BPEL process definition.
Namespaces	Lets you edit the defined namespaces in the BPEL process document.

Report a bug

3.6. OBSERVING A BPEL PROCESS

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 will appear. Click on 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

3.7. 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, if you want to add your own service, follow the steps below:

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.

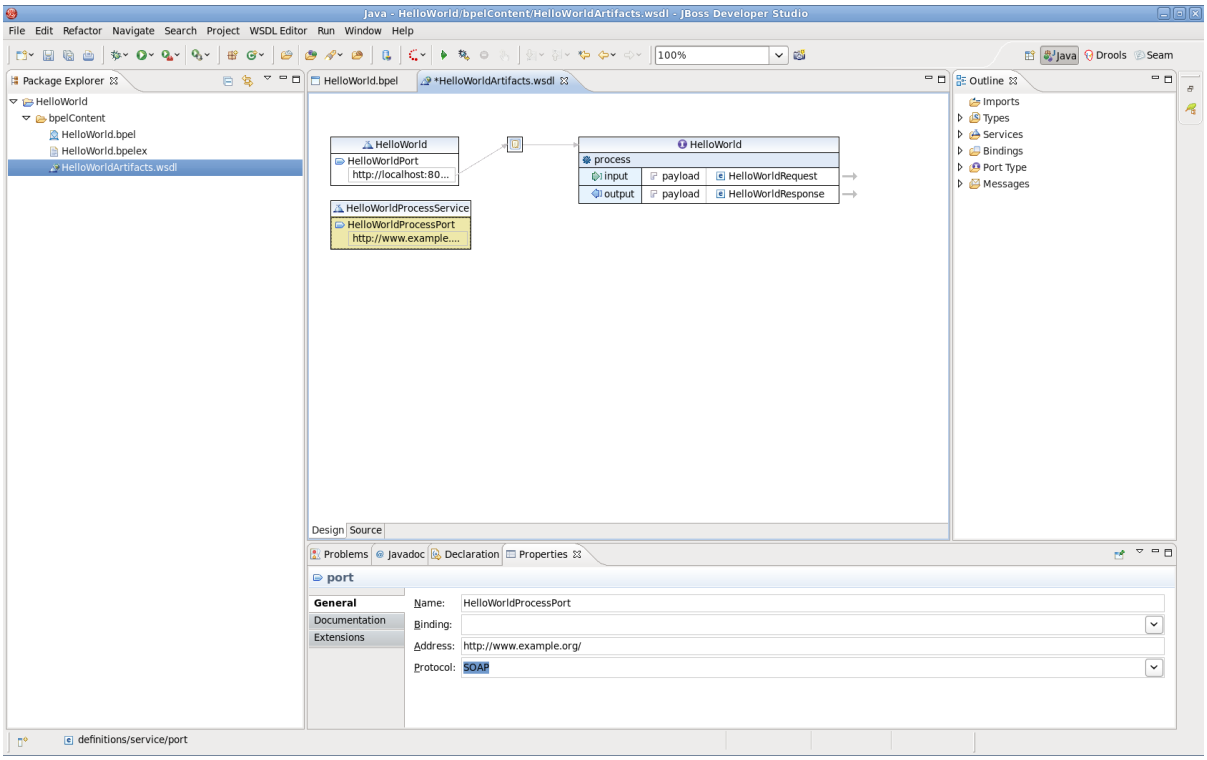


Figure 3.8. Diagram 1

3. Right-click in the whitespace of the WSDL editor and select the **Add Binding** option. A new Binding component will appear 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 **PortType** 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.

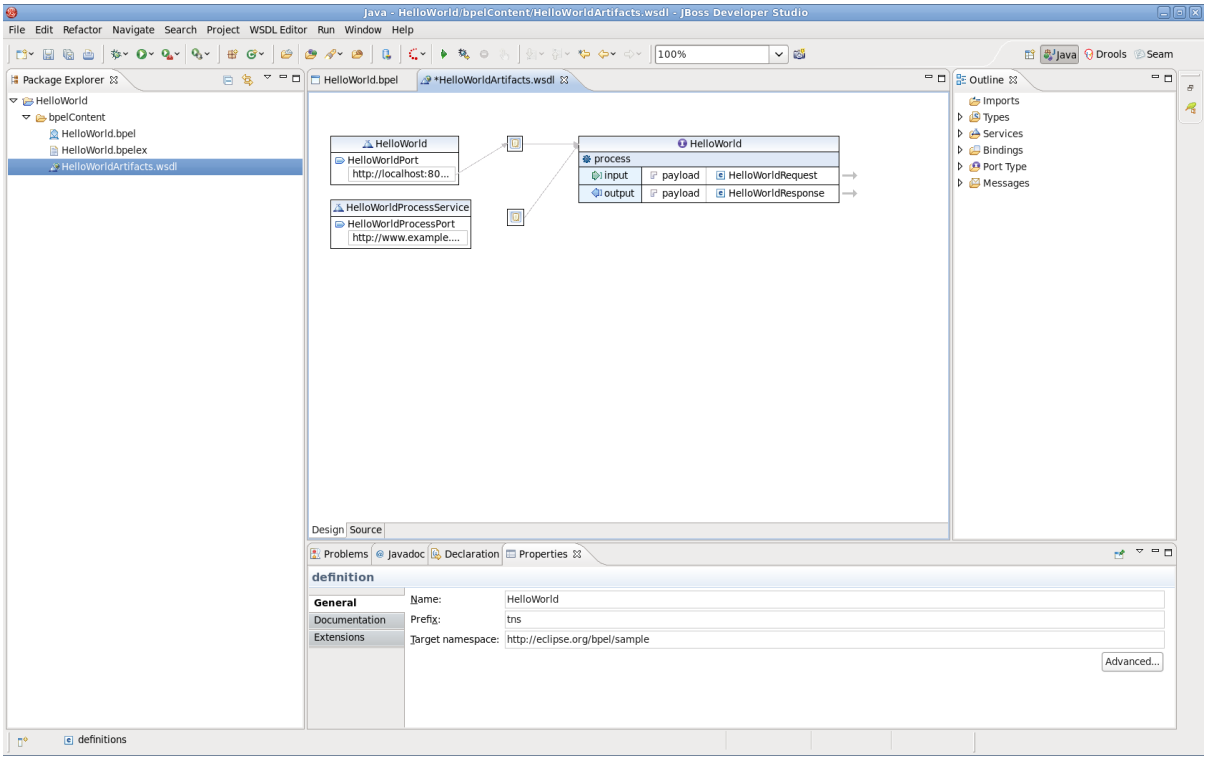


Figure 3.9. Diagram 2

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.

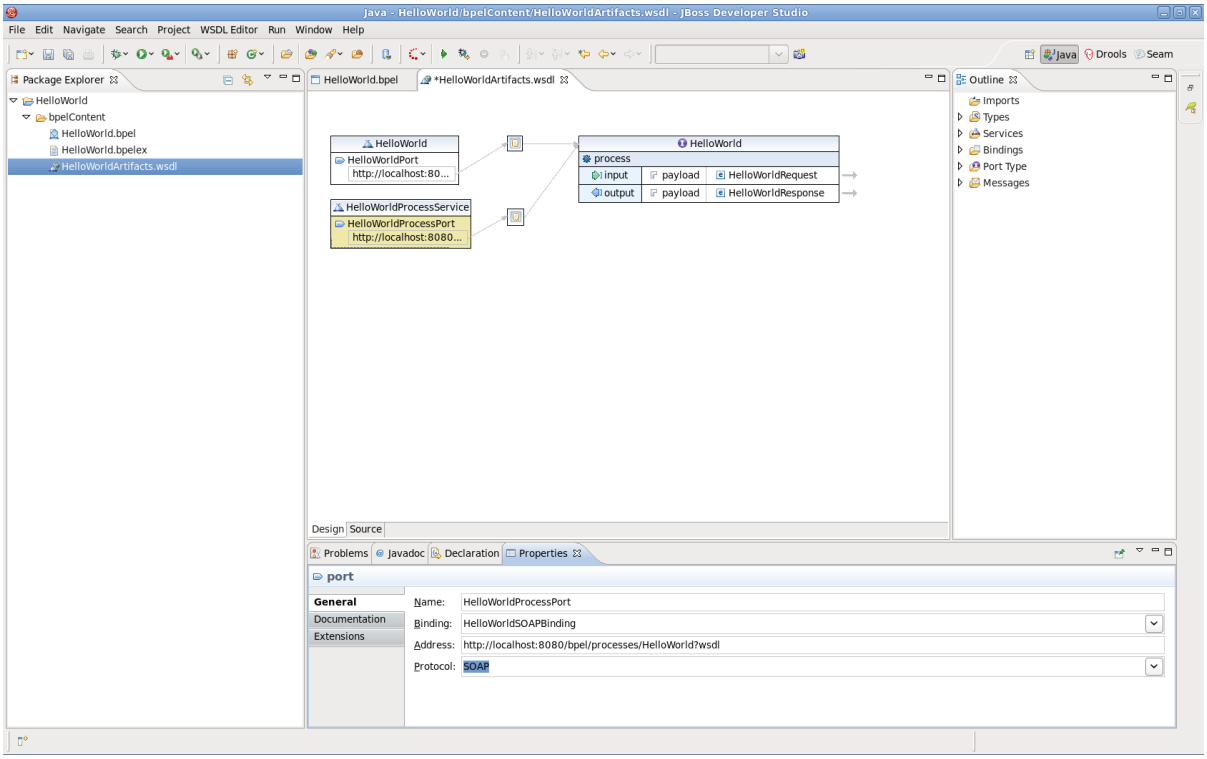


Figure 3.10. Diagram 3

Report a bug

3.8. CREATING A DEPLOY.XML FILE

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.

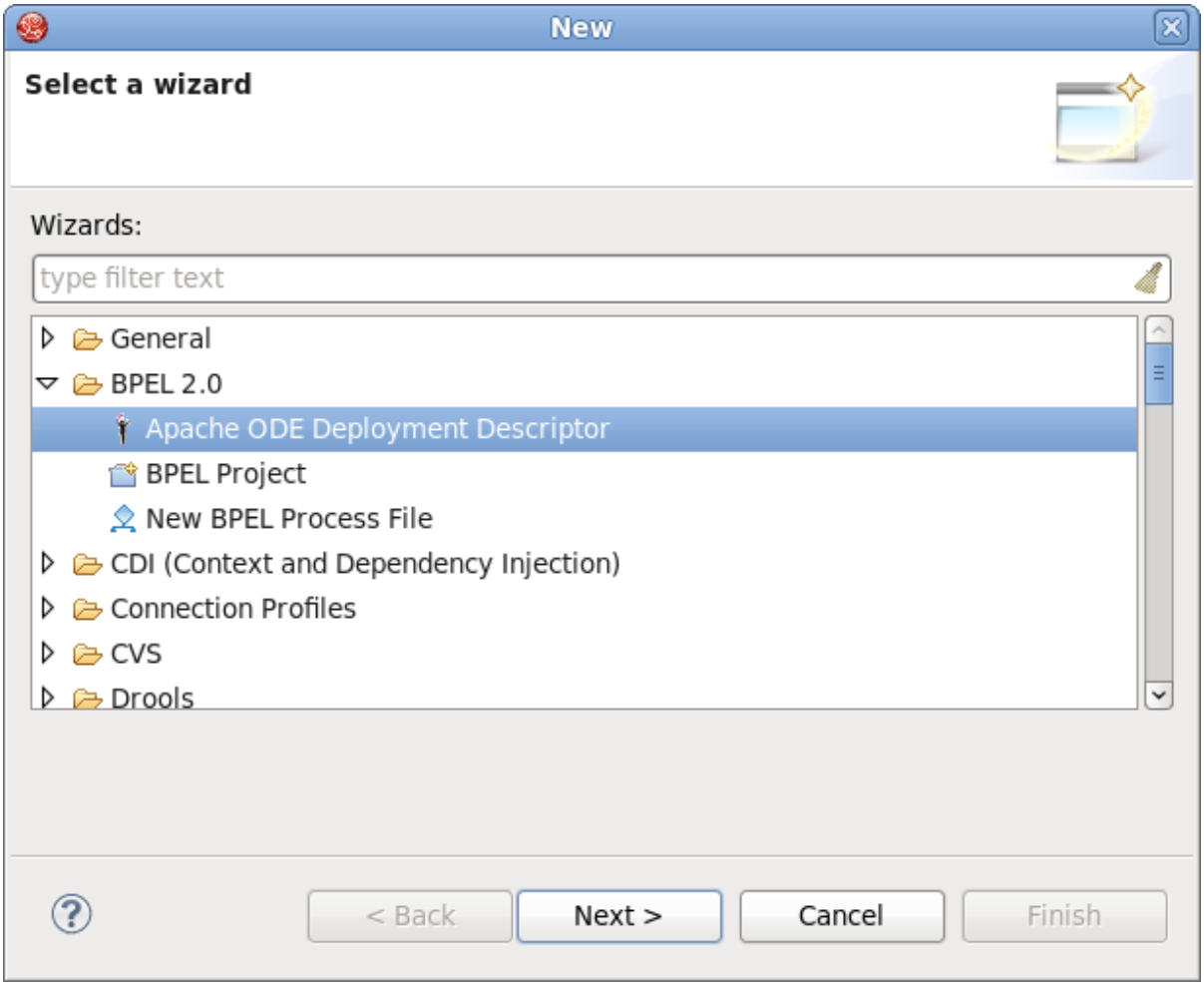


Figure 3.11. Diagram 1

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 will be created.

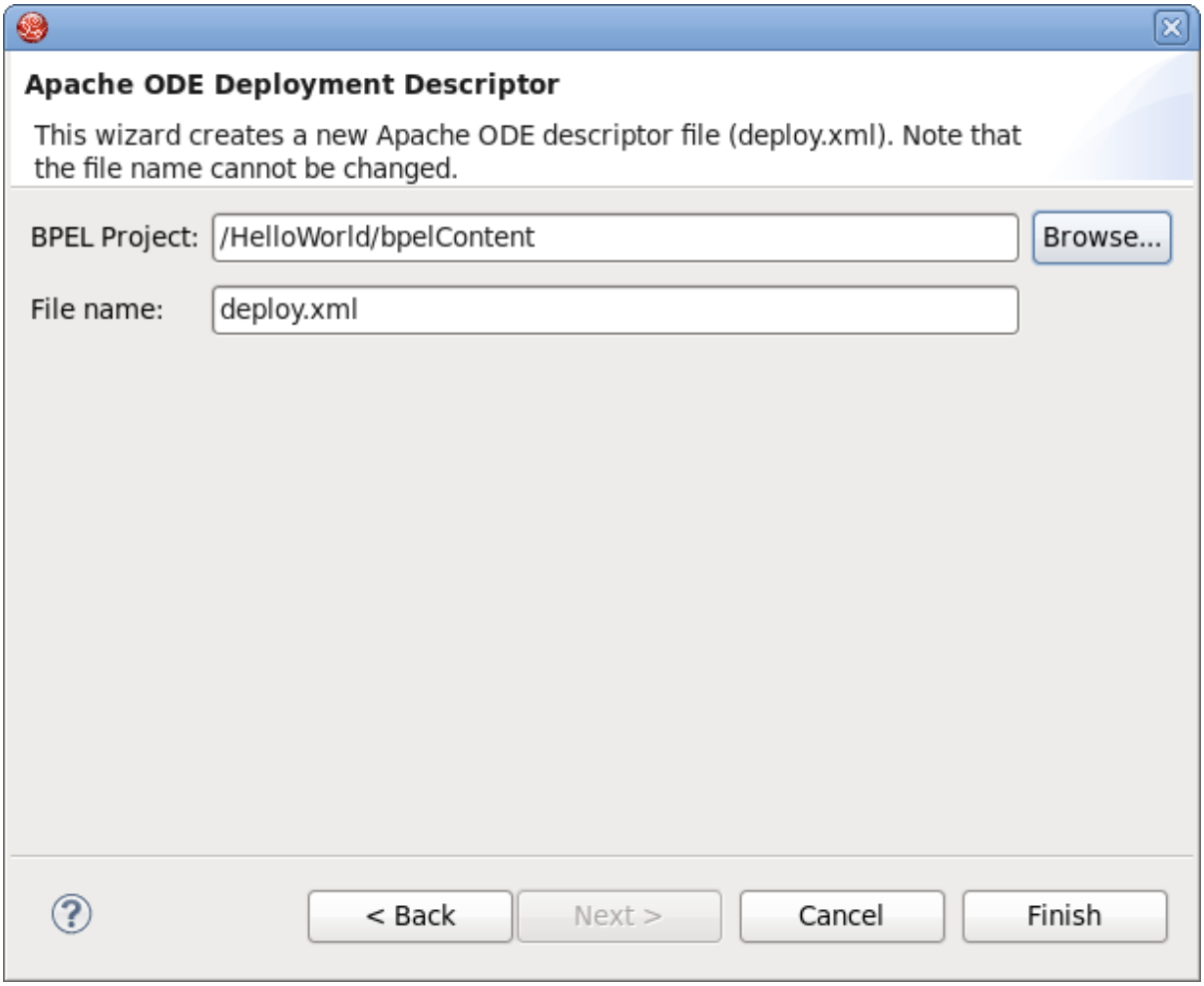


Figure 3.12. Diagram 2

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.

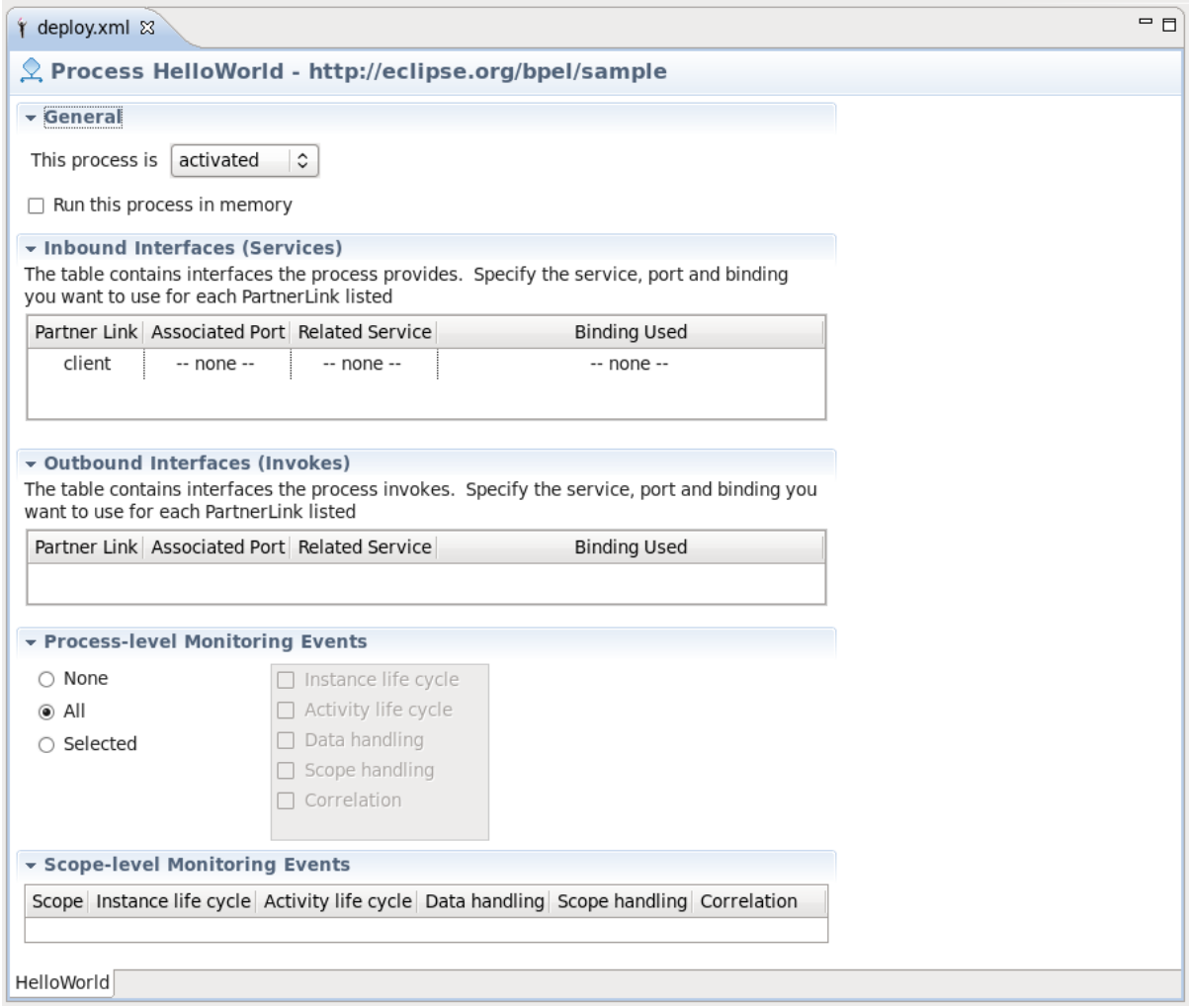


Figure 3.13. Diagram 3

Report a bug

3.9. CREATING JBOSS BPEL SERVER

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.

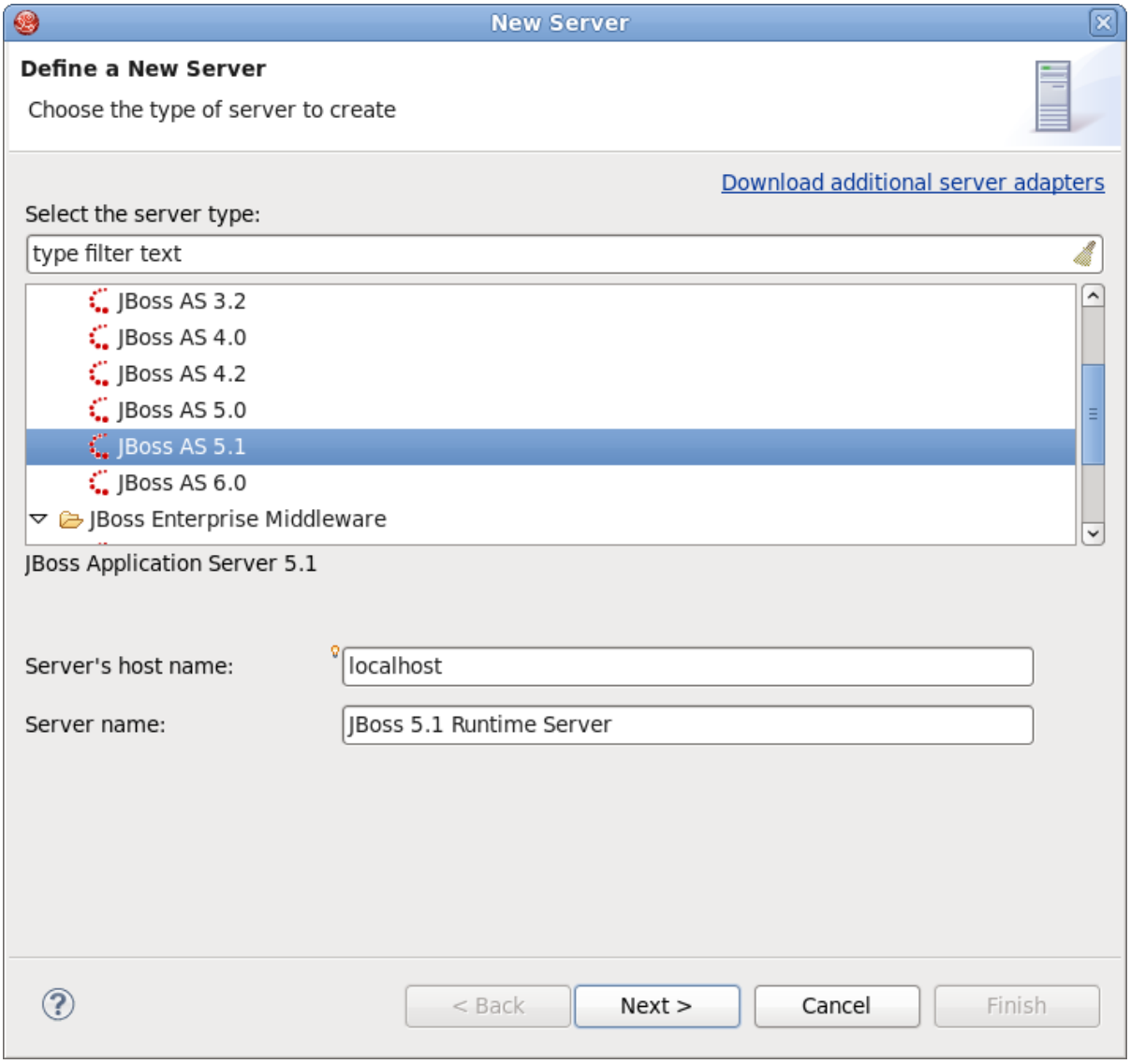


Figure 3.14. Diagram 1

3. Select **JBoss EAP 5.x** as a server type.

4. Click the **Next** button. On this page, input your JBoss EAP location. Then click **Next**.

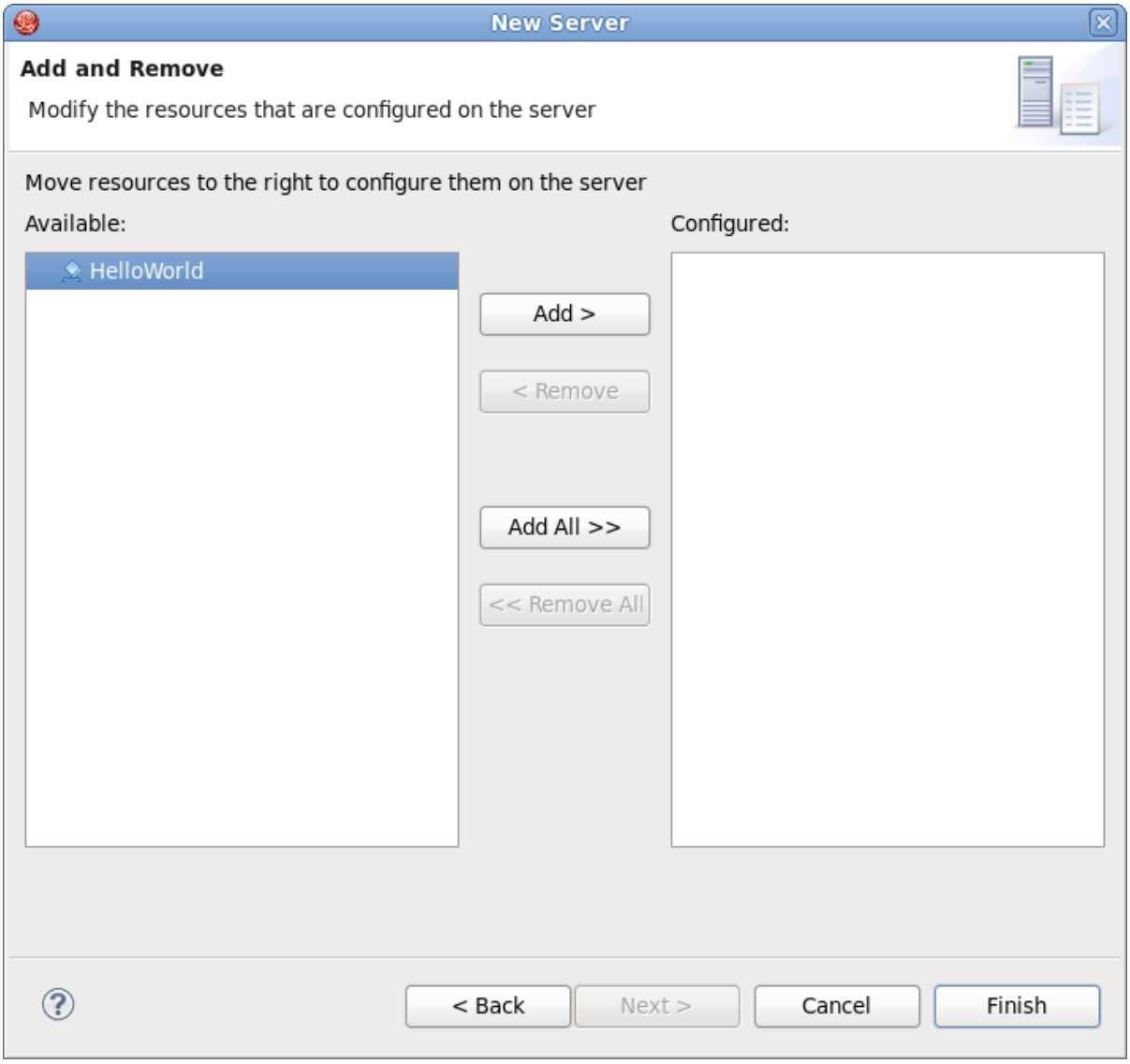


Figure 3.15. Diagram 2

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.

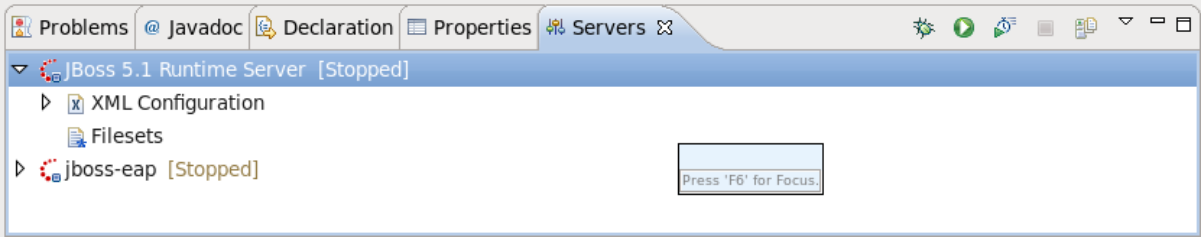


Figure 3.16. Diagram 3

7. Enter the link http://localhost:8080/bpel-console/app.html in your web browser to access the deployed processes.

Report a bug

3.10. CREATING CORRELATION SETS

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. This will display the **Select a Property** dialog.
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. This will display the **Type Selection** dialog.
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

CHAPTER 4. REFERENCE

4.1. WIZARDS

Table 4.1. 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 **`bpelContent`** folder contains all the files necessary for your project.
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 will be 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 will open once this wizard completes.

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

4.2. VIEWS

Table 4.2. Views

View	Description

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.
Palette	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. • 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 *pinned* 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	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.

Report a bug

4.3. PROPERTY SECTION TABS

Table 4.3. Property Sections

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 will prompt 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

4.4. PROCESS PROPERTY SHEET TABS

Table 4.4. 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 will cause 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 will handle join failures. When set to **Yes**, any **JoinFailure** fault will be 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

4.5. DETAILS TAB OPTIONS

Table 4.5. 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 will open 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 an Input Variable. For request-response invocations you must also specify an Output Variable. The checkbox 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`** checkbox can be set to store the incoming message in an anonymous WSDL message variable. The **`Create a new Process Instance`** checkbox, when enabled, will cause the BPEL engine to start a new process. This will start 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** checkbox 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 will be 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 will appear below the **From** and **To** combo boxes, depending on the source and target item categories selected in the combo boxes. Initially these will be 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 will cause 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 will create a new process instance.
OnMessage	The OnMessage activity is used inPick 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 will contain the received message data.

Name	Description
OnAlarm	The OnAlarm activity is used in either aPick 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 will contain the received message data. **`Repeat`** conditions are only allowed for an OnAlarm 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 will wait before each repetition.
ForEach	ForEach allows you to specify a counter variable to be used for keeping track of the loop iterations. The **`Parallel execution`** checkbox will execute 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.
Wait	The details tab of the Wait activity allows you set a delay (Duration) or specify a date and time to continue process execution.
Scope	The details tab for the Scope activity allows you to define whether the Scope is **`isolated`**.
Throw	The Throw activity will invoke 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.
CompensateScope	The CompensateScope activity will invoke a compensation handler in the Scope or the Invoke activity given by the name of the **`Target Activity`**.

Report a bug

4.6. BPEL DESIGNER FEATURES

Table 4.6. BPEL Designer Features

Name	Description
Drawing Canvas	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.
Palette	The primary editing, creation and viewing tools of the BPEL Designer are accessed from this tool.
Dashboard	Provides an overview of the BPEL process.
Property Sheet	Displays the properties of an activity when it is selected in the drawing canvas.
Outline	This panel provides a structural view of the BPEL process.

Report a bug

4.7. BPEL DESIGNER CONCEPTS

Table 4.7. BPEL Designer Concepts

Name	Description
Assign error	Hovering your mouse over this icon will display 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 will wait 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 invalidVariables 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 will delay 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

4.8. STRUCTURED ACTIVITIES

Structured activities are containers that can hold one or more activities. The **`Controls`** section of the **`Palette`** contains all of the *structured activities*. When you drag and drop one of these onto the drawing canvas, the BPEL Designer will create a basic skeleton of the activity and assign default properties.

All structured activities will require some additional configuration before they are considered valid. For example, BPEL does not allow an empty Sequence activity. Invalid structured activities will be decorated with an error icon similar to basic activities.

Structured activities can be expanded and collapsed on the drawing canvas by clicking the plus and minus buttons at the bottom of the figure. See the list of activities below.

Table 4.8. Structured Activities

Name	Description
If, ElseIf and Else	The If activity allows conditional execution of one or more sequences of activities. It consists of a sequence of one or more conditional branches defined by If and optional ElseIf elements. The elements are evaluated in left-to-right order (or top-to-bottom if you have selected horizontal layout). An optional Else branch will be executed if none of the other conditions are true. An If activity must define a condition (expressed as an XPath) and an activity which is executed if the condition evaluates true. To insert additional ElseIf and Else elements, right-click the If figure and select the desired element from the context menu. The figure above shows a complete If activity with optional ElseIf and Else elements.
Pick	The Pick activity will cause the process to wait for one of any number of messages to be received. An optional timer can be set to limit the time to wait for receipt of these messages. Activities to handle receipt of messages and timer expiration are defined in the Pick. Message receipts are handled by OnMessage activities and timer expiration is handled by the OnAlarm activity.
While	The While activity repeatedly executes the contained activity as long as a condition evaluates true at the beginning of each iteration. A While activity must define a condition and must contain an activity.
ForEach	ForEach is a looping activity that executes the activities contained in its Scope a specified number of times. A counter variable, defined in the ForEach property detail tab, is used to keep track of the iterations. The ForEach properties must be configured with starting and ending value expressions for this counter variable. The counter is initially set to the starting value and activities in the Scope are executed until the counter exceeds the ending value. This activity can also be configured to execute all iterations in parallel, meaning the enclosed Scope activity behaves as if multiple Scopes are enclosed in a Flow activity. An optional early termination value can be defined, which will cause the loop to complete before the counter has reached its ending value. The ForEach will complete when the counter is equal to this early termination value for the sequential execution case. For the parallel execution case, the early termination value is the number of completed iterations. For example, the ForEach completes when at least *some number* of *some action* have finished.

Name	Description
RepeatUntil	The RepeatUntil activity repeatedly executes the contained activity as long as a condition evaluates true at the end of each iteration. A condition must be defined for a RepeatUntil, and it must contain an activity.
Sequence	A Sequence is a container for one or more other activities. They are executed in sequential order and (unlike Scope and Flow activities), have no other special characteristics. Because the conditional activities (If, While, RepeatUntil and ForEach) can have only one activity as the target of their execution, a Sequence is typically used to execute multiple activities. The BPEL Designer will automatically create a Sequence if you drag-drop a second activity into any of the conditional activities.
Scope	A Scope provides a context for its enclosed activity. It can be thought of as a compartmentalized sub-process. If the Scope is declared as being *isolated*, then the variables and partner links shared with the process are locked to prevent other concurrent Scopes from altering them while a Scope is executing. Scope may also be nested to any depth and all variables, partner links and others defined in a Scope, are inherited by its children. To be valid, a Scope must have a single activity. It is usually used to invoke a service and wait for a response message or timeout.
Flow	The Flow activity allows multiple activities to be executed in parallel. All activities or Sequences contained in a Flow are executed at the same time by the BPEL engine. A Flow completes when all of its enclosed activities have completed.
Link	Links are used for synchronization. To create a Link, right-click on a completed activity and select **Add Link**. Next, move the mouse to the activity in the Flow that depends on this one (the *target*) and click the left mouse button to create the link. A Link is identified by a name that must be unique within the Flow. The BPEL Designer generates a default name, but you can change this in its properties. You can also add a test to the Link that defines the conditions for considering an activity to be complete.

Name	Description
Faults	Fault activities cause the normal process execution flow to jump to a specialized handler, similar to exceptions in modern programming languages. The Exit fault causes the process to immediately terminate. The Throw fault propagates a specified fault to its ancestor Scope, or the process itself. The Rethrow fault is used inside a fault handler and propagates the fault using the original fault data. The Compensate fault is used to invoke a compensation handler. Finally, the CompensateScope fault is used to invoke a compensation handler in the enclosing Scope.

Report a bug

4.9. FAULT, COMPENSATION, TERMINATION AND EVENT HANDLERS

Table 4.9. Fault, Compensation, Termination and Event Handlers

Name	Description
Fault handler	Executed when a fault is thrown by an activity.
Compensation handler	Executed when the BPEL process encounters a Compensate or CompensateScope activity.
Termination handler	Executed if a Scope is forced to terminate early.
Event handler	Executed for events include the receipt of a message and a timer expiration.
Scope-level handlers	A Scope may have any handler. Since Scopes can be nested, each level can define its own set of handlers. Events that are not caught and processed by a handler in an inner Scope, will be propagated to its ancestors.
Activity-level handlers	Only the Invoke activity can define handlers. The handlers available to it are the fault and compensation handlers.

Report a bug

4.10. BPEL DEPLOYMENT DESCRIPTOR EDITOR PROPERTIES

Table 4.10. 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 *active*, *inactive* or *retired* state.
Inbound interfaces selection	Select the WSDL port type that clients will use to invoke this service.
Output interfaces selection:	Each invoked service (if any) will require 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

4.11. DIALOGS

Table 4.11. 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	This dialog is displayed whenever the BPEL Designer requires you to select a message, message part, XML Schema type or XML element. 1. **`Type Name`**: Used to limit the items displayed in the **`Matches`** (4) list. Only items that begin with the text in this filter will be displayed. 2. **`Show XSD Types`**: Can be used to limit where the editor will search 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 will update 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

4.12. TYPE SELECTION

Table 4.12. Type Selection

Name	Description
`Type Name`	Used to limit the items displayed in the **`Matches`** (4) list. Only items that begin with the text in this filter will be displayed.
`Show XSD Types`	Can be used to limit where the editor will search for XSD files.
`Filter`	Further reduces the number of matches according to types.

Name	Description
Matches	Displays the items matching the selected filters. Selecting an item in this list will update the **Type Structure** (5) tree view.
Type Structure	Displays the structure of the item selected in the **Matches** list. You may need to select an item from this tree control. The **OK** button will be enabled if this is required.
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

4.13. SELECT WSDL PROPERTY

This dialog allows you to select an existing WSDL property or create a new one.

Table 4.13. Select WSDL Property

Name	Description
Property Name	Used to limit the items displayed in the **Matches** list. Only items beginning with the text in this filter will be displayed.
New	Click this button to create a new WSDL property.
Matches	Displays the items that match the **Property Name**, or all items if the filter is blank.
Property Type	Displays the type of an item selected in the **Matches** list.

Report a bug

4.14. CREATE WSDL PROPERTY

This dialog is used to create a new WSDL property and is displayed when you click the **New** button in the Select WSDL propertydialog.

Table 4.14. Create WSDL Property

Name	Description
Name	Enter the name for the new property.
Defined As	Select how the property type will be defined.
Browse	Click this button to select the property type.
New	Click this button to create a new property alias.
Aliases	This list displays all property aliases defined for the property.

Report a bug

4.15. CREATE WSDL PROPERTY ALIAS

This dialog allows you to create a WSDL property alias for a selected property.

Table 4.15. Create WSDL Property Alias

Name	Description
Defined As	Select how the property alias type will be defined. **
Browse	Click this button to select the property alias type.
Query	This editor allows you to use the XPath Expression Editor to define a query for the property alias.

Report a bug

4.16. CHEAT SHEETS

Procedure 4.1. Task

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

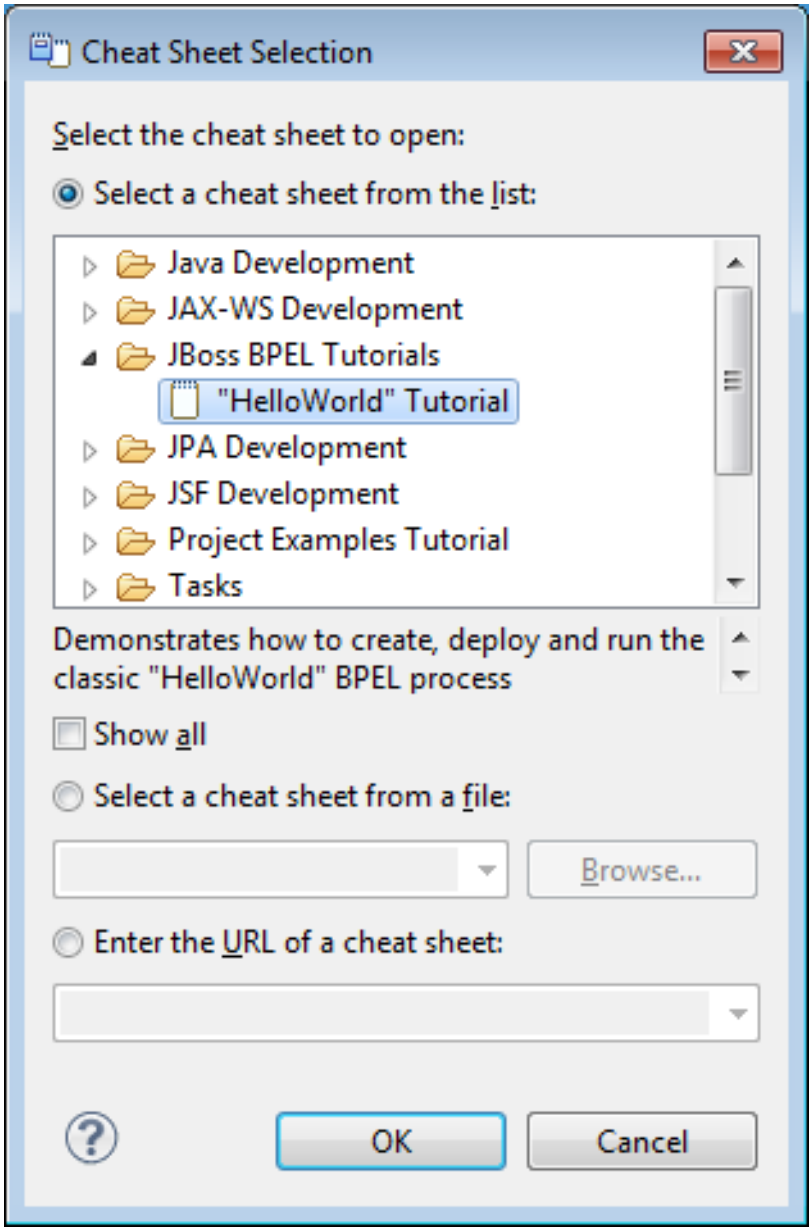


Figure 4.1. Diagram 1

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

Report a bug

4.17. CONTEXT MENU

Access the context menu by right-clicking over an activity figure on the drawing canvas. A sub-menu will appear. 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

APPENDIX A. REVISION HISTORY

Revision 5.3.1-31.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revision 5.3.1-31 **Wed Feb 20 2013** **CS Builder Robot**

Built from Content Specification: 7172, Revision: 375275