



JBoss Enterprise SOA Platform 5

ESB Tools Reference Guide

This guide is for developers

Edition 5.3.1

Last Updated: 2017-10-27

JBoss Enterprise SOA Platform 5 ESB Tools Reference Guide

This guide is for developers
Edition 5.3.1

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

B Long
Red Hat Engineering Content Services
belong@redhat.com

Legal Notice

Copyright © 2013 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 developers how to use JBDS' ESB Tools plug-in.

Table of Contents

PREFACE	2
CHAPTER 1. PREFACE	3
1.1. BUSINESS INTEGRATION	3
1.2. WHAT IS A SERVICE-ORIENTED ARCHITECTURE?	3
1.3. KEY POINTS OF A SERVICE-ORIENTED ARCHITECTURE	3
1.4. WHAT IS THE JBOSS ENTERPRISE SOA PLATFORM?	4
1.5. THE SERVICE-ORIENTED ARCHITECTURE PARADIGM	4
1.6. CORE AND COMPONENTS	4
1.7. COMPONENTS OF THE JBOSS ENTERPRISE SOA PLATFORM	5
1.8. JBOSS ENTERPRISE SOA PLATFORM FEATURES	5
1.9. FEATURES OF THE JBOSS ENTERPRISE SOA PLATFORM'S JBOSS-ESB COMPONENT	5
1.10. TASK MANAGEMENT	6
1.11. INTEGRATION USE CASE	6
1.12. UTILISING THE JBOSS ENTERPRISE SOA PLATFORM IN A BUSINESS ENVIRONMENT	7
CHAPTER 2. INTRODUCTION	8
2.1. INTENDED AUDIENCE OF THE ESB TOOLS REFERENCE GUIDE	8
2.2. AIM OF THE ESB TOOLS REFERENCE GUIDE	8
2.3. BACK UP YOUR DATA	8
CHAPTER 3. TASKS	9
3.1. USING AND CONFIGURING THE SOA PLATFORM	9
3.2. INSTALLING JBOSS SOA TOOLING PLUG-IN	9
3.3. CONFIGURING A NEW ESB RUNTIME	9
3.4. CREATING AN ESB PROJECT	11
3.5. USING ESB PROJECT EXAMPLES	11
3.6. ADDING ESB PROJECTS TO THE JBOSS SERVER MODULE LIST	13
3.7. DEPLOYING AN ESB PROJECT	14
3.8. CREATING AN ESB FILE	14
3.9. CREATING AN ESB ACTION	15
CHAPTER 4. REFERENCE	16
4.1. THE JBOSS ESB EDITOR	16
4.2. ADDING A NEW CUSTOM ACTION IN JBOSS ESB EDITOR	17
4.3. CONTENT ASSIST FOR ESB XML FILES	18
4.4. OPENON FOR ESB XML	18
CHAPTER 5. SUMMARY	20
5.1. SUMMARY OF ESB TOOLS AND ADDITIONAL RESOURCES	20
APPENDIX A. REVISION HISTORY	21

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 that 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 JBOSS ESB 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 occurring.

[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 OF THE ESB TOOLS REFERENCE GUIDE

Introduction

This book is aimed at developers who wish to learn about performing tasks and using features in the JBoss Enterprise Service Bus.

[Report a bug](#)

2.2. AIM OF THE ESB TOOLS REFERENCE GUIDE

This book aims to teach users about the functionality that JBoss ESB plugin provides for work with JBoss ESB. Readers will learn how to perform basic tasks such as creating ESB projects, using the ESB editor and how to utilize features to make for a more efficient workflow.

[Report a bug](#)

2.3. 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](#)

CHAPTER 3. TASKS

3.1. USING AND CONFIGURING THE SOA PLATFORM

1. To configure the JBoss Enterprise SOA Platform, select **Window** → **Preferences** → **Server** → **Runtime Environments**. This will open the Server Runtime Environments page where you can add, remove and edit a server runtime environment.
2. Select **Add** and choose **JBoss Enterprise Middleware** → **JBoss Enterprise Application Platform 5.x Runtime** as a type of runtime environment, then check the **Create a new local server** box and click **Next**.
3. On the next step you can specify a name of the server runtime environment and browse to its location. Click **Finish** to add the server runtime environment.
4. You have now configured the SOA Platform. Check the configuration by creating and running an ESB project.

[Report a bug](#)

3.2. INSTALLING JBOSS SOA TOOLING PLUG-IN

Procedure 3.1. Task

1. Start the JBoss Developer studio.
2. Select *Get started with JBoss Central* from the main menu.
3. Select *Software/Update* from the menu.
4. Select *SOA Tooling Plug-in* and click *Install*
5. Confirm that the installation details are correct and click *Next* and then *Next* again.
6. Read and accept the liscence agreement and click *Finished*.
7. Restart the JBoss Developer Studio to complete the installation.

[Report a bug](#)

3.3. CONFIGURING A NEW ESB RUNTIME

1. Click **Window** → **Preferences** → **JBoss Tools** → **JBoss ESB Runtimes** to open the JBoss ESB Runtimes page where you can add, remove and edit a JBoss ESB runtime.

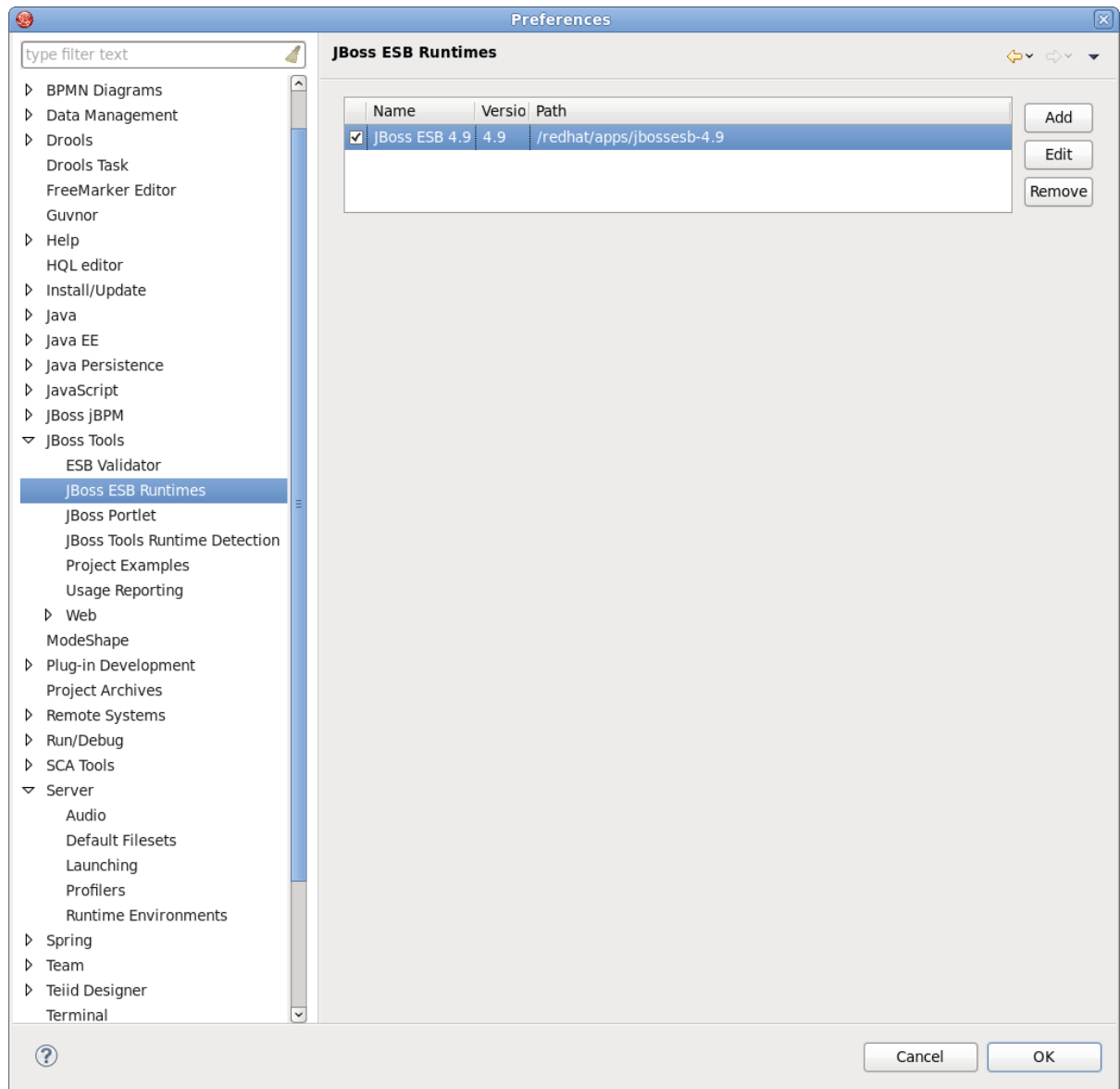


Figure 3.1. Runtimes

2. Select **Add** to open a dialog box where you can specify the JBoss ESB runtime location, name and version number. You can define the configuration if you specify the home directory as JBoss AS or SOA-P. If you select a standalone ESB runtime location, the configuration will be empty and can be ignored. You can also customize the libraries of the runtime by checking the **Customize JBoss ESB Runtime jars** box.
3. The new JBoss ESB Runtime will be configured. Click **Finish** to finish and save the preferences. You can use the configuration when creating a JBoss ESB project.
4. When an ESB runtime is configured for your project, you are able to modify it by using the Classpath Container page for ESB runtime. To do so, go to the Package Explorer view and right-click on "JBoss ESB Runtime" library.
5. Select **Properties** and a table listing all available JBoss ESB runtimes will appear.
6. Choose one of the runtimes to set to the ESB project and click **OK**.
7. The ESB container allows Source and JavaDoc locations to be set via the Properties dialog on each contained JAR. Right-click on any JAR file and select **Properties**.

8. Choose **Java Source Attachment** and select the location (folder, JAR or zip) containing new source for the chosen JAR file. You can do this using one of the suggested options (workspace, external folder or file). You can also enter the path manually.
9. Click on **Apply** and then on **OK**.
10. To change the Javadoc location, choose **Javadoc Location** and specify URL to the documentation generated by Javadoc. The Javadoc location will contain a file called **package-list**.
11. Click on **Apply** and then on **OK**.

[Report a bug](#)

3.4. CREATING AN ESB PROJECT

1. Open the JBoss ESB Project Creation wizard by selecting **File** → **New** → **Project...** in the main menu bar. (Ensure that you have defined the server runtime and ESB runtime in JBDS.)
2. Select **ESB** → **ESB Project** in the dialog box.
3. Clicking **Next** brings you to the JBoss ESB Project wizard page where a project name, ESB version and target JBoss Runtime are to be specified. Specify, for example, helloworld as a project name and accept the default JBoss ESB version.
4. Click the **Next** button to move to the next step in the wizard, where you can select the source and output folders.
5. Click the **Next** button to view the Install ESB Facet wizard page where you can view the ESB Content Directory. This folder contains most the of artifacts that an ESB archive needs.
6. You can configure ESB libraries to the project by using the **Server Supplied ESB Runtime**. Make sure the project has the target runtime set and this runtime has an ESB runtime installed.



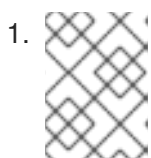
NOTE

If you are using JBoss SOA Platform 5.3 or higher, select ESB config version 1.3.1.

7. Click the **Finish** button and an ESB project with the default **jboss-esb.xml** file will be created.

[Report a bug](#)

3.5. USING ESB PROJECT EXAMPLES



NOTE

Creating ESB project examples is useful for observing how your finished project will behave.

Before creating an ESB project example, create a JBoss SOA-P runtime.

2. Select **File** → **New** → **Other...** in the main menu bar and then click **Examples** → **JBoss Tools** → **Project Examples** in the new dialog.
3. Clicking the **Next** button brings you to the wizard page where you can select an ESB project example from the list.
4. Every ESB example has two projects. One is an ESB project and the other is a Java project used to test the ESB project. Below is a list of ready-to-go examples. Each example includes an ESB application that is deployed to the SOA-P server, and a test client that is run as a Java application and interacts with the deployed ESB application:
5.
 - **JBoss ESB HelloWorld Example** - demonstrates the minimal files necessary to make a basic ESB component execute as well as to prove that the ESB runtime is properly configured.
 - **JBoss ESB HelloWorld Action Example** - demonstrates the use of multiple action invocations from a single configuration. You can use a single Action class and make multiple method calls or use multiple Action classes.
 - **JBoss ESB HelloWorld File Action Example** - demonstrates using the File gateway feature of the JBoss ESB. Files that are found in a particular directory with a particular extension are sent to a JMS queue with actions for processing.
 - **JBoss ESB Web Service consumer1 Example** - demonstrates how to consume a JSR181 Web Service in an ESB action.
 - **JBoss ESB Web Service producer Example** - demonstrates how to deploy a JSR181 Webservice endpoint on JBossESB using the SOAPProcessor action.
 - **JBoss ESB Smooks CSV -> XML Example** - demonstrates how to transform a comma separated value (CSV) file to an XML.
 - **JBoss ESB Smooks XML -> POJO Example** - demonstrates the use of Smooks performing a simple transformation by converting an XML file into Java POJOs.
 - **JBoss ESB Smooks XML -> XML date-manipulation Example** - demonstrates how to manually define and apply a Message Transformation within JBoss ESB.
 - **JBoss ESB Smooks XML -> XML Example** - a very basic example of how to manually define and apply a Message Transformation within JBoss ESB. It applies a very simple XSLT to a SampleOrder.xml message and prints the before and after XML to the console.
6. Select the relevant ESB project example and then click the **Finish** button. The ESB application project and the test client project will be created.

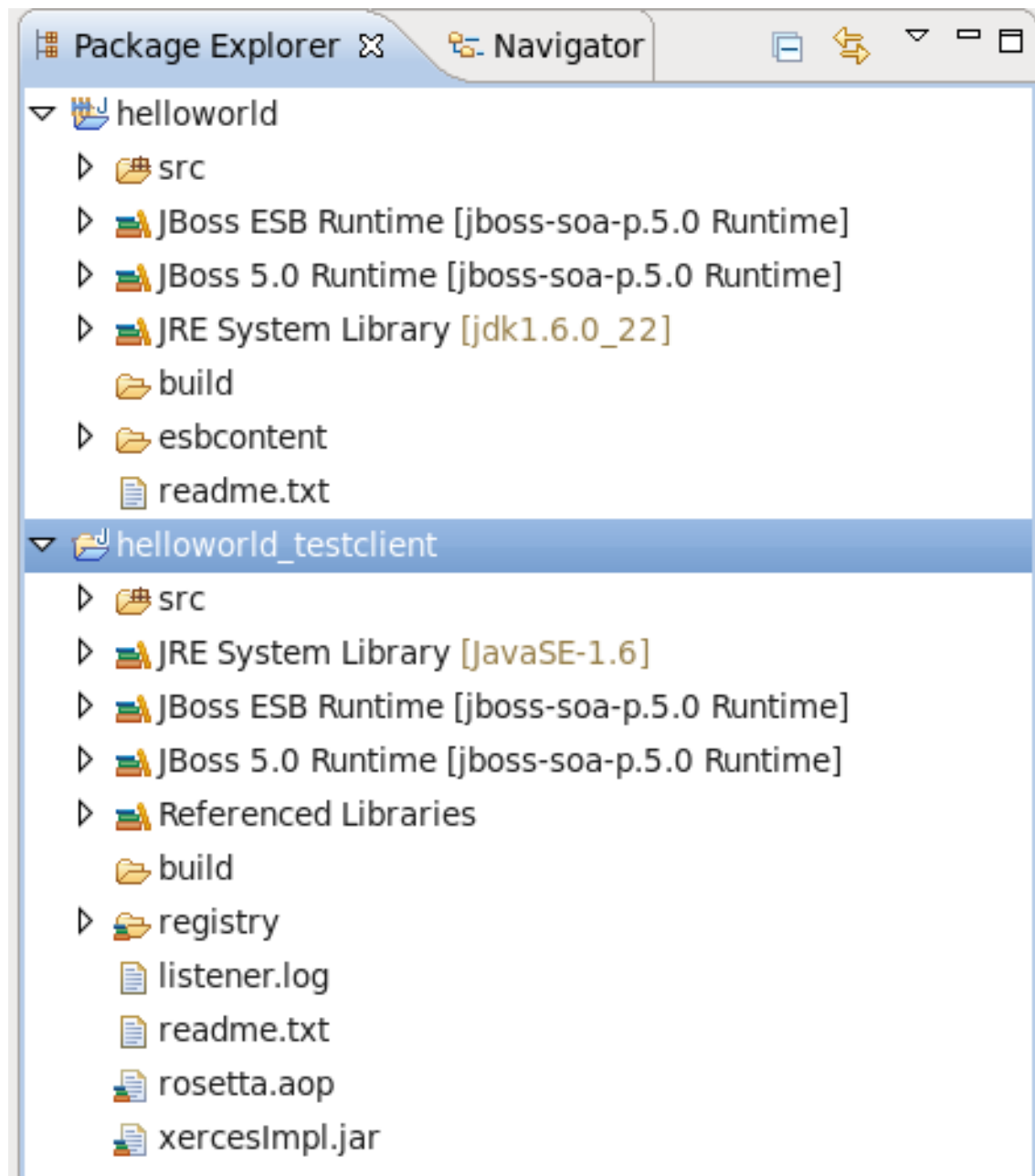


Figure 3.2. HelloWorld example

7. Deploy your ESB project and run a test class as a local Java application to see the test result in the Console view.

[Report a bug](#)

3.6. ADDING ESB PROJECTS TO THE JBOSS SERVER MODULE LIST

1. Before deploying an ESB project, open the Servers view by selecting **Window** → **Show View** → **Other** → **Server** → **Servers**.
2. Create a JBoss server with an ESB runtime installed in the Servers view and start it.
3. Right-click on the new JBoss server, select **Add and Remove Projects** and add the ESB projects you want to deploy in the open dialog.

4. Click the **Finish** button to add the project to the server. You also can drag the ESB project from the Project Explorer view to the server.

[Report a bug](#)

3.7. DEPLOYING AN ESB PROJECT

1. After adding the ESB project to the JBoss server module list, right-click the JBoss Server and select **Publish** to publish the project on the server. You can check the deploying result in the Console view.
2. The **Run** and **Debug** options work on ESB projects causing a (re)deploy for a user designated server.

You can also click the "Finger touch" button for a quick restart of the project without restarting the server.

3. Exporting

Additionally, you can deploy your ESB project as an .esb archive. Right-click on the project and choose **Export**.

Choose **ESB** → **ESB File** and click **Next**.

Export the ESB project to the file system. Choose the destination and target runtime, then make the appropriate settings for the archive. Finally, click **Finish**

4. Your project is now deployed as an .esb archive.

[Report a bug](#)

3.8. CREATING AN ESB FILE

1. Open a project that has the ESB facet enabled. Select **File** → **New** → **Other...** in the main menu bar or context menu for your selected project, then click **ESB** → **ESB File** in the new dialog
2. Clicking the **Next** button brings you to the wizard page where a folder, name and version should be specified. For example, choose *jboss-esb.xml* as the name and accept the selected projects folder and the default version.



NOTE

For SOA-P version 5.3 your ESB version is 4.11 and the ESB file schema version is 1.3.1. Older ESB file schema versions are also supported if you are using an older SOA-P server release.

3. Your file will be created in the selected project's folder by default. If you want to change the folder it is saved in, click **Browse** and select another destination.
4. Click **Finish** to generate the file.

[Report a bug](#)

3.9. CREATING AN ESB ACTION

1. Open an ESB or Java project.
2. Select **File** → **New** → **Other** in the main menu or from the context project menu. Then click **ESB** → **ESB Action** in the new dialog.
3. Click **Next** and you will be brought to the *New ESB Action* wizard where the class name should be specified. You can also set a package or add an interface.
4. JBoss ESB 4.9 introduced a new set of Action Annotations. The new set of Action Annotations help make it easier to create clean ESB Action implementations by hiding the complexities of implementing interfaces and abstract classes, as well as dealing with the ConfigTree type. If you wish use these annotations, check the type **As annotated POJO**. Otherwise check the type **As AbstractActionPipelineProcessor implementation**.

The Annotations are:

- @Process
 - @ConfigProperty
 - @Initialize
 - @Destroy
 - @BodyParam
 - @PropertyParams
 - @AttachmentParam
 - @OnSucess
 - @OnException
5. Click on **Finish** to generate the ESB Action class. This class will become available in ESB Editor wizards.

[Report a bug](#)

CHAPTER 4. REFERENCE

4.1. THE JBOSS ESB EDITOR

1. The JBoss ESB editor has two views: Tree and Source. You can easily switch from the Tree view to the Source view by selecting the **Source** tab at the bottom of the editor and work in Source view, and vice-versa.

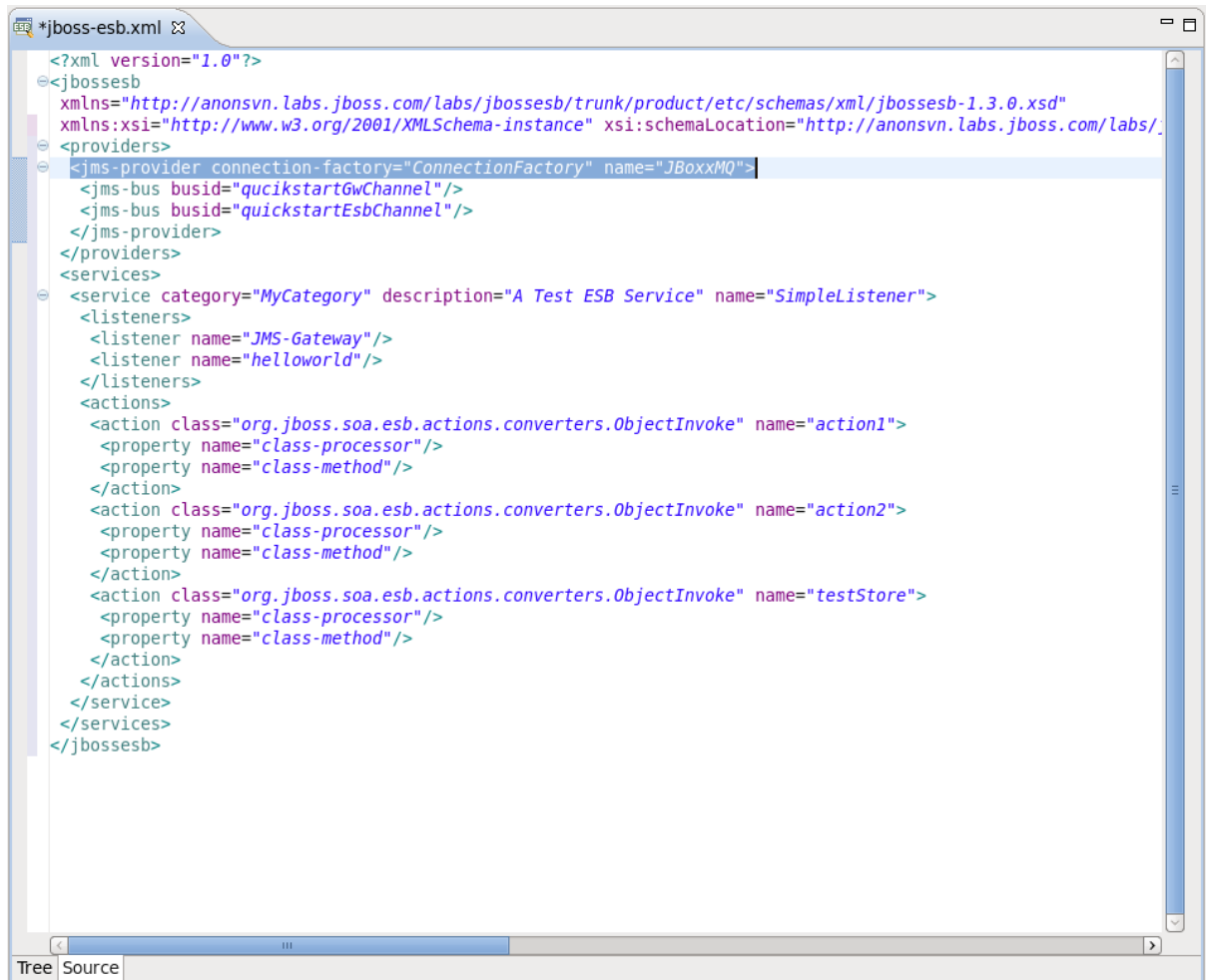


Figure 4.1. Source view

2. Both views have full integration with the Outline view. The Outline view shows a tree structure of the ESB file. Select any element in the Outline view and it will jump to the same place in the Tree/Source editor, so you can navigate through the source code with Outline view.
3. Both views are synchronized with each other, so any changes made in one of the views will immediately appear in the other.
4. The Tree view for the editor displays all ESB artifacts in a tree format. By selecting any node you can view and edit its properties which will appear on the right. Some properties are represented as links to the associated editors.

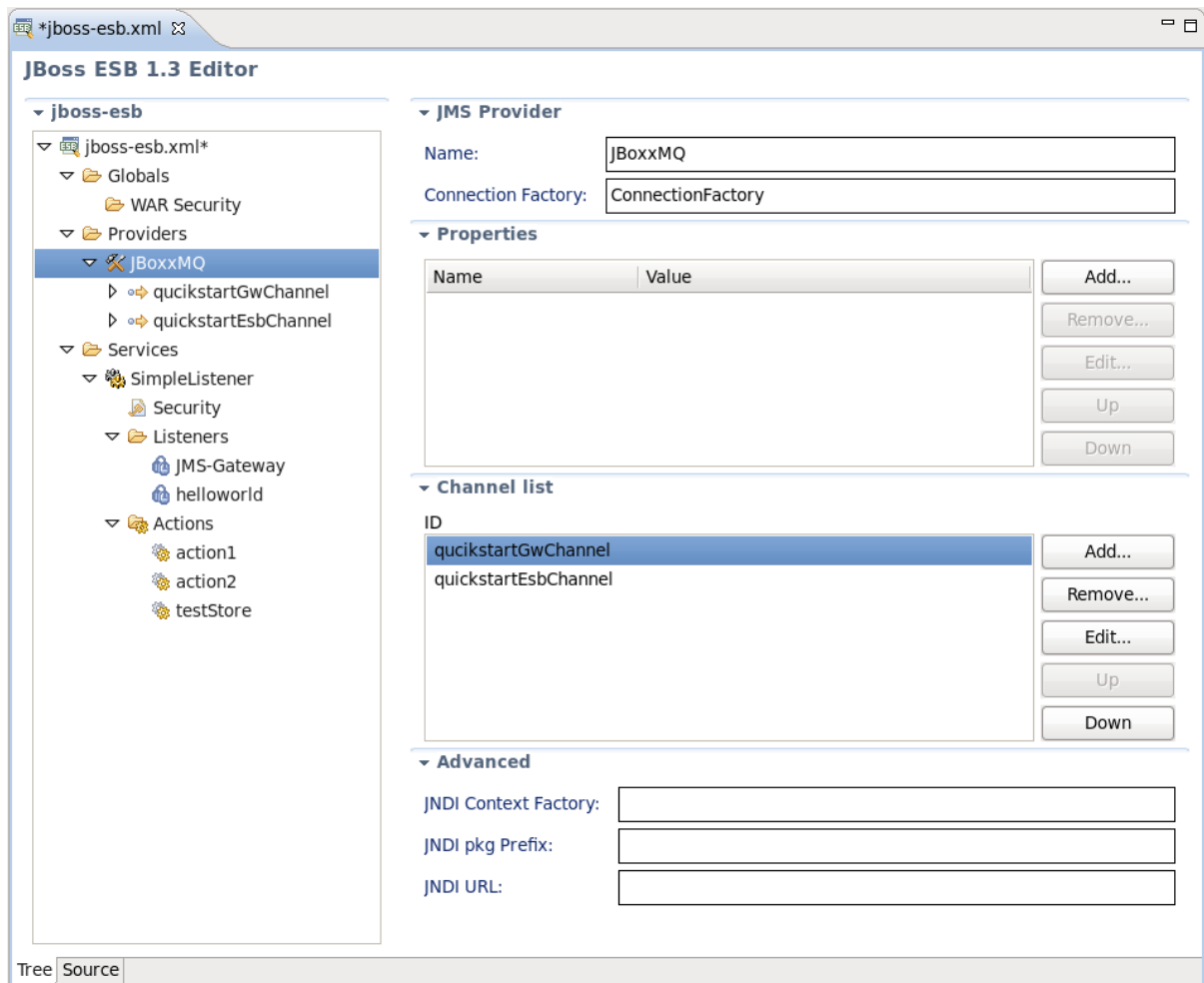


Figure 4.2. Tree view

5. When editing ESB actions which refer to other files (Drools, Groovy, Smooks, etc.), the label for the field turns into a link to launch the appropriate editor for that file.
6. Adding, editing or deleting of some artifacts operations are available right in the Tree view. Right-click any node and select one of the available actions in the context menu.
7. You can add Channels and Properties for the Providers the same way or using the forms with Add, Edit and Remove buttons to the right.
8. Additionally, you can add a Service and a new listener for it. Filters can also be added this way.

[Report a bug](#)

4.2. ADDING A NEW CUSTOM ACTION IN JBOSS ESB EDITOR

1. To add a new custom Action to your ESB XML file, go to the Tree view in the JBoss ESB Editor.
2. Select the **Actions** node under the **Services** node, then right-click and choose **New** → **Custom Action**.

Alternatively, click the **Add...** button in the editor on the left.

3. You will see the ESB Action wizard. Specify a name and a Java class. Optionally, you can specify a process for the Action.

4. After creating the Action you can see it in the Tree view under the **Actions** node. You can preview and edit its settings in the Form editor on the left.

[Report a bug](#)

4.3. CONTENT ASSIST FOR ESB XML FILES

When you work with any ESB XML file in Source mode, Content Assist is available. It provides pop-up tips to help you complete your code statements. Type **Ctrl-Space** to see what tips are available.

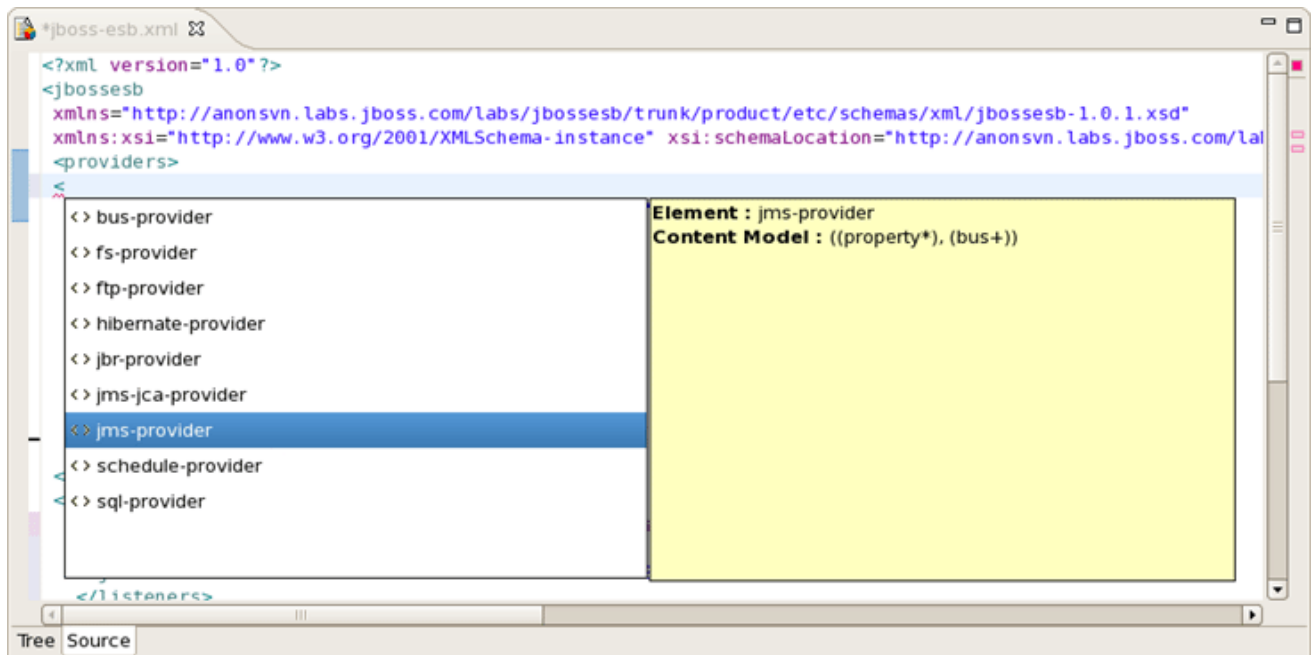


Figure 4.3. Content Assist for ESB XML file

[Report a bug](#)

4.4. OPENON FOR ESB XML

ESB files come with the OpenOn feature that allows you to utilize multiple references in the file. Click and hold the **Ctrl** key down to view it.

```

    <jms-message-filter
      dest-name="queue/quickstart_helloworld_Request_gw" dest-type="QUEUE"/>
  </jms-bus>
  <jms-bus busid="quickstartEsbChannel">
    <jms-message-filter
      dest-name="queue/quickstart_helloworld_Request_esb" dest-type="QUEUE"/>
  </jms-bus>
</jms-provider>
</providers>
<services>
<service category="FirstServiceESB" description="Hello World" name="SimpleListener"
  <listeners>
    <jms-listener busidref="quickstartGwChannel" is-gateway="true" name="JMS-Gateway"
    <jms-listener busidref="quickstartEsbChannel" name="helloWorld"/>
  </listeners>
  <actions mep="OneWay">
    <action
      class="org.jboss.soa.esb.samples.quickstart.helloworld.MyJMSListenerAction"
      name="action1" process="displayMessage"/>
    <action class="org.jboss.soa.esb.actions.SystemPrintln" name="action2">
      <property name="message"/>
      <property name="printfull" value="true"/>
    </action>
    <!-- The next action is for Continuous Integration testing -->
    <action class="org.jboss.soa.esb.actions.TestMessageStore" name="testStore"/>
    <action class="org.jboss.soa.esb.smooks.SmooksAction" name="smooks">
      <property name="smooksConfig" value="META-INF/smooks-config.xml"/>
    </action>
  </actions>
</service>

```

Figure 4.4. OpenOn

The OpenOn is implemented for different types of files with extensions such as `.xsd`, `.xml`, and so on.

[Report a bug](#)

CHAPTER 5. SUMMARY

5.1. SUMMARY OF ESB TOOLS AND ADDITIONAL RESOURCES

This book has supplied you with all necessary information on the functionality that the JBoss ESB plugin provides for work with JBoss ESB.

You can find additional information by visiting the links below:

- [JBoss ESB](#)
- [JBoss Wiki](#)
- [JBoss ESB Documentation Library](#)

The latest JBoss Tools/JBoss Developer Studio documentation builds are available on the [JBoss Tools nightly documentation page](#) .

[Report a bug](#)

APPENDIX A. REVISION HISTORY

Revision 5.3.1-111.400

2013-10-31

Rüdiger Landmann

Rebuild with publican 4.0.0

Revision 5.3.1-111

Wed Feb 20 2013

CS Builder Robot

Built from Content Specification: 6854, Revision: 375405