JBoss Enterprise SOA Platform 5

Installation and Configuration Guide

This guide is for installation teams.

Edition 5.3.1

Last Updated: 2017-10-27

JBoss Enterprise SOA Platform 5 Installation and Configuration Guide

This guide is for installation teams.
Edition 5.3.1

David Le Sage
Red Hat Engineering Content Services

Suzanne Dorfield
Red Hat Engineering Content Services

Legal Notice

Abstract

This document will guide the user through all of the installation and configuration options for the JBoss Enterprise SOA Platform product.

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?

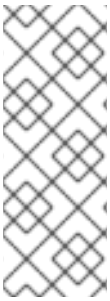
Introduction

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

PART I. INTRODUCTION

CHAPTER 2. INTRODUCING THE JBOSS ENTERPRISE SOA PLATFORM

2.1. INTENDED AUDIENCE

This book has been written for installation project teams needing a comprehensive guide to all the options available when installing the JBoss Enterprise SOA Platform product.

Report a bug

2.2. AIM OF THIS BOOK

Read this book in order to learn how to install and configure the JBoss Enterprise SOA Platform on a Red Hat Enterprise Linux or Microsoft Windows computer. It guides you through all of the installation options available to you. It also teaches basic operation of the product for both testing and production environments.

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

2.4. 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

2.5. OUT-OF-THE-BOX ACTIONS

Out-of-the-box actions are generic pieces of code for actions that come prepackaged with the JBoss Enterprise SOA Platform product. You can use them immediately in your services or customize them to suit your needs.

Report a bug

2.6. JBOSS ENTERPRISE SOA PLATFORM OUT-OF-THE-BOX ACTIONS

The out-of-the-box actions implemented in the SOA Platform are divided into the following functional groups:

Transformers and Converters

Use transformer and converter actions to change message data from one form to another.

Business Process Management

Use the business process management actions when integrating your software with the jBPM.

Scripting

Use scripting actions to automate tasks written in the supported scripting languages.

Services

Use service actions when integrating your code with Enterprise Java Beans.

Routing

Use routing actions when moving message data to destination services.

Notifier

Use notifier actions when sending data to ESB-unaware destinations.

Web Services/SOAP

Use web service actions when you need to support web services.

Report a bug

CHAPTER 3. PREREQUISITES

3.1. PREREQUISITES FOR INSTALLING THE JBOSS ENTERPRISE SOA PLATFORM

In order to install and run the JBoss Enterprise SOA Platform, you must already have the following items on your computer:

- A supported Java Virtual Machine
- A supported Java Development Kit (for running the quickstarts)
- A supported database server (needed to run the JBoss Server)
- Apache Ant 1.7 or later (needed to run the Database Schema Configuration Tool and deploy the JBoss ESB quick start examples)
- An archiving tool (such as FileRoller, ark or tar). (You need this to extract the contents of compressed files)
- JBoss Developer Studio 5.0. (Obtain it from the Red Hat Customer Portal at https://access.redhat.com/home)

Red Hat tests and certifies the JBoss Enterprise SOA Platform against several different hardware platforms, Java Virtual Machines, operating systems and databases. This is an ongoing process and the list of supported environments is always growing. Find the list of currently-supported environments at http://www.jboss.com/products/platforms/soa/testedconfigurations/.

Report a bug

3.2. INSTALL OPEN JDK ON RED HAT ENTERPRISE LINUX

Procedure 3.1. Install Open JDK on Red Hat Enterprise Linux

1. **Subscribe to the Base Channel**
   Obtain the OpenJDK from the RHN base channel. (Your installation of Red Hat Enterprise Linux is subscribed to this channel by default.)

2. **Install the Package**
   Use the yum utility to install OpenJDK: **`yum install java-1.7.0-openjdk-devel`**

3. **Verify that OpenJDK is Now Your System Default**
   To ensure that the correct JDK is set as the system default, login as root and run the alternatives command: **`/usr/sbin/alternatives --config java`**

   Select /usr/lib/jvm/jre-1.6.0-openjdk/bin/java

   Set javac **`/usr/sbin/alternatives --config javac`**

   Select /usr/lib/jvm/java-1.6.0-openjdk/bin/java

Report a bug

3.3. APACHE ANT

Apache Ant ("Another Neat Tool") is a Java-based build tool. It is designed to automate the process of software compilation. It requires the developer to provide it with an XML-based build file from which it receives custom build instructions. Refer to http://ant.apache.org/ for more information.

Report a bug

3.4. INSTALL APACHE ANT

Procedure 3.2. Installing Apache Ant on Red Hat Enterprise Linux

1. **Download Apache Ant**
   Open a terminal and input this command: **`sudo yum install ant-trax`**

2. **Install Apache Ant**
   Enter **`Y`** when prompted to do so by the installer.

3. **Add the ANT_HOME Environment Variable**

   a. **`vi ~/.bash_profile`**.

   b. Add the following line:

   ```
   export ANT_HOME=/FILEPATH/ant
   ```

   where filepath is the directory in which Apache Ant was installed. Save and exit vi.

   Example 3.1.

   ```
   export ANT_HOME=/opt/apache-ant-1.8.2
   ```

4. **Append the Ant installation's `bin` directory to the Path environmental variable.**

   a. **`vi ~/.bash_profile`**.

   b. Add the following line and then save and exit vi:

   ```
   export PATH=$PATH:$ANT_HOME/bin
   ```

5. **Test the installation**
   Go back to your terminal and run **`ant -version`**. The output should resemble the following:

   ```
   [localhost]$ ant -version
   Apache Ant(TM) version 1.8.2 compiled on July 6 2011
   ```

Procedure 3.3. Installing Apache Ant on Microsoft Windows

1. **Download Apache Ant**
   Download the latest Apache Ant binary release from http://ant.apache.org/.

2. **Extract Apache Ant**
   Extract the files to a preferred installation location such as:
   - **`c:\Program Files\Apache\Ant\`**
3. **Add the ANT_HOME Environment Variable**
   a. Click on the **Start Menu**.
   b. Open the **Control Panel**.
   c. Select **System** → **Advanced** → **Environment Variables**
   d. Create a new variable called **`ANT_HOME`**.
   e. Configure the ANT_HOME variable so that it points to the Apache Ant directory.
4. **Append the bin directory of the Ant installation to the Path environmental variable.**
   a. Click on the **Start Menu**.
   b. Open the **Control Panel**.
   c. Select **System** → **Advanced** → **Environment Variables** → **System Variables**
   d. Edit the **PATH** variable and append the text:

```
;%ANT_HOME%\bin
```

5. **Test the installation**
   Run **`ant -version`** in a command line terminal. The version number should appear.

Report a bug

PART II. BASIC INSTALLATION AND OPERATION

CHAPTER 4. DOWNLOAD THE PRODUCT

4.1. RED HAT CUSTOMER PORTAL

The Red Hat Customer Portal is a website, located at https://access.redhat.com/home. It provides a central point from where you can manage and maintain your subscription, access the Red Hat Knowledgebase and engage with Red Hat and our partners.

Report a bug

4.2. PACKAGES AVAILABLE FOR DOWNLOAD

Table 4.1. Packages Available for Download

Package	Description
JBoss Enterprise SOA Platform Package	The SOA Platform package is a complete JBoss application deployment environment. This single installation provides a complete environment for deploying SOA applications. It includes Seam, Hibernate, clustering, and transaction services.
JBoss Enterprise SOA Platform Standalone Edition Package	The SOA Standalone package provides a light-weight solution for deployments where only the core SOA functionality is needed. It does not support clustering.
JBoss Enterprise SOA Platform Source Code Package	The source code package contains the complete source code for the JBoss Enterprise SOA Platform product.
SOA Platform JavaDocs	The JavaDocs package contains the complete JavaDocs for the JBoss Enterprise SOA Platform's APIs.

Report a bug

4.3. DIFFERENCES BETWEEN VERSIONS OF THE JBOSS ENTERPRISE SOA PLATFORM

Table 4.2. Differences Between Versions of the JBoss Enterprise SOA Platform

	SOA Platform Package	SOA Standalone Package
JBoss ESB	YES	YES
JBoss Rules	YES	YES

	SOA Platform Package	SOA Standalone Package
JBoss JBPM	YES	YES
JBoss EAP	YES	YES
BPEL Engine	YES	YES
EJB3	YES	NO
JBoss RestEasy	YES	NO
JBoss Seam	YES	NO
Support for JBoss Enterprise Data Services Deployment	YES	NO

Report a bug

4.4. JAVADOCS

JavaDocs are automatically-generated documentation for Java APIs. They are created from the comments developers add to source code as they write it. JavaDocs have become the de facto standard way of documenting Java APIs.

Report a bug

4.5. DOWNLOAD FILES FROM THE RED HAT CUSTOMER PORTAL

Task Prerequisites

Before you begin this task, you need a Customer Portal account. Browse to https://access.redhat.com and click the **Register** link in the upper right corner to create an account.

Procedure 4.1. Task:

1. Browse to https://access.redhat.com and click the **Log in** link in the top right corner. Enter your credentials and click **Log In**.

   Result:

   You are logged into RHN and you are returned to the main web page at https://access.redhat.com.

2. **Navigate to the Downloads page.**
   Use one of the following options to navigate to the **Downloads** page.

   - Click the **Downloads** link in the top navigation bar.

- Navigate directly to https://access.redhat.com/downloads/.

3. **Select the product and version to download.**
   Use one of the following ways to choose the correct product and version to download.
   - Step through the navigation one level at a time.
   - Search for your product using the search area at the top right-hand side of the screen.
4. **Download the appropriate file for your operating system and installation method of choice.**
   Depending on the product you choose, you may have the choice of a Zip archive, RPM, or native installer for a specific operating system and architecture. Click either the file name or the **Download** link to the right of the file you want to download.

Result:

The file is downloaded to your computer.

Report a bug

4.6. CHECKSUM VALIDATION

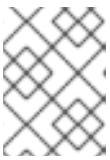
Checksum validation is used to ensure a downloaded file has not been corrupted. Checksum validation employs algorithms that compute a fixed-size datum (or checksum) from an arbitrary block of digital data. If two parties compute a checksum of a particular file using the same algorithm, the results will be identical. Therefore, when computing the checksum of a downloaded file using the same algorithm as the supplier, if the checksums match, the integrity of the file is confirmed. If there is a discrepancy, the file has been corrupted in the download process.

Report a bug

4.7. VERIFY THE DOWNLOADED FILE

Procedure 4.2. Verify the Downloaded File

1. To verify that a file downloaded from the Red Hat Customer Portal is error-free, whilst still on the portal site, go to that package's **Software Details** page. Here you will find **MD5** and **SHA256**"checksum" values that you will use to check the integrity of the file.
2. Open a terminal window and run either either the **md5sum** or **sha256sum** command, supplying the filename of the downloaded **ZIP** as an argument. The program will output the checksum value for the file.
3. Compare the checksum value returned by the command to the corresponding value displayed on the **Software Details** page for the file.



NOTE

Microsoft Windows does not come equipped with a checksum tool. Users of that operating system will have to download a third-party product instead.

Result

If the two checksum values are identical then the file has not been altered or corrupted and is, therefore, safe to use.

If the two checksum values are not identical, then download the file again. A difference between the checksum values means that the file has either been corrupted during download or has been modified since it was uploaded to the server. If, after several downloads, the checksum will still not successfully validate, please contact Red Hat Support for assistance.

Report a bug

4.8. RED HAT DOCUMENTATION SITE

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

Report a bug

CHAPTER 5. INSTALLATION

5.1. VARIABLE NAME: SOA_ROOT DIRECTORY

SOA Root (often written as SOA_ROOT) is the term given to the directory that contains the application server files. In the standard version of the JBoss Enterprise SOA Platform package, SOA root is the **`jboss-soa-p-5`** directory. In the Standalone edition, though, it is the **`jboss-soa-p-standalone-5`** directory.

Throughout the documentation, this directory is frequently referred to as **`SOA_ROOT`**. Substitute either **`jboss-soa-p-5`** or **`jboss-soa-p-standalone-5`** as appropriate whenever you see this name.

Report a bug

5.2. VARIABLE NAME: PROFILE

PROFILE can be any one of the server profiles that come with the JBoss Enterprise SOA Platform product: default, production, all, minimal, standard or web. Substitute one of these that you are using whenever you see "PROFILE" in a file path in this documentation.

Report a bug

5.3. INSTALL THE JBOSS ENTERPRISE SOA PLATFORM ON RED HAT ENTERPRISE LINUX

Prerequisites

- a Java Development Kit (Red Hat recommends OpenJDK)
- a database server
- Apache Ant 1.7 or later
- an archiving tool that can open ZIP files

WARNING

These instructions explain how to install and configure the product on a test system only. These defaults will not result in the level of security needed for a production system. Please read the full Installation Guide for information on installing and configuring this product for production use.

Follow these steps after you have downloaded either the Standalone or full SOA package from the Red Hat Customer Portal and verified its integrity.

Procedure 5.1. Installation

Procedure 5.1. Installation

1. Extract the installation directory by running **`unzip soa-p-VERSION.zip`**.
2. Open the user account settings file in your text editor: **`vi SOA_ROOT/jboss-as/server/default/conf/props/soa-users.properties`**.

   The contents of this file using the following syntax: **`username=password`**.

   ```
   #admin=admin
   ```

   Confirm that the security roles for your admin account are enabled by removing a leading hash character, if it is present.

   WARNING

   Because this account is not secure and the password can be easily guessed, use admin=admin for test purposes only. Do not use **`admin`** as a password on production systems as this may compromise security.

3. Save the file and exit vi.
4. Open the security permissions file in your text editor: **`vi SOA_ROOT/jboss-as/server/default/conf/props/soa-roles.properties`**.

   ```
   #admin=JBossAdmin,HttpInvoker,user,admin
   ```

   Confirm that the security roles for your admin account are enabled by removing a leading hash character, if it is present.
5. Save the changes to the file and exit vi.

Result

The JBoss Enterprise SOA Platform is now installed and configured for basic use.

Report a bug

5.4. INSTALL THE JBOSS ENTERPRISE SOA PLATFORM ON MICROSOFT WINDOWS

Prerequisites

- a Java Development Kit
- a database server
- Apache Ant 1.7 or later

- an archiving tool that can open ZIP files

WARNING

These instructions explain how to install and configure the product on a test system only. These defaults will not result in the level of security needed for a production system. Please read the full Installation Guide for information on installing and configuring this product for production use.

Follow these steps after you have downloaded either the Standalone or full SOA package from the Red Hat Customer Portal and verified its integrity.

Procedure 5.2. Installation

1. Extract the **`soa-p-VERSION.zip`** using the ZIP tool of your choice.
2. Confirm that the security roles for your admin account are enabled by removing a leading hash character, if it is present.

WARNING

Because this account is not secure and the password can be easily guessed, use admin=admin for test purposes only. Do not use **`admin`** as a password on production systems as this may compromise security.

3. Save the file and exit Notepad.
4. Open **`SOA_ROOT\jboss-as\server\default\conf\props\soa-roles.properties`** in Notepad.

```
#admin=JBossAdmin,HttpInvoker,user,admin
```

Remove the hash to enable the security permissions for your admin account.

5. Save the file and exit Notepad.

Result

The JBoss Enterprise SOA Platform is now installed and configured for basic use.

Report a bug

CHAPTER 6. BASIC OPERATION TUTORIAL

6.1. RUNNING THE JBOSS ENTERPRISE SOA PLATFORM FOR THE FIRST TIME

Introduction

In the following section you will learn how to launch and run the JBoss Enterprise SOA Platform for the first time. The simplest thing you can do is run the demonstration code found in the "Hello World" quick start.

As you run this quick start, you will be taught how it works.

Report a bug

6.2. START THE JBOSS ENTERPRISE SOA PLATFORM

Prerequisites

The following software must be installed:

- JBoss Enterprise SOA Platform

Procedure 6.1. Start the JBoss Enterprise SOA Platform

- **Start the SOA server in a *server window***
  - **Red Hat Enterprise Linux**
    a. Open a terminal and navigate to the **bin** directory by entering the command **cd *SOA_ROOT*/jboss-as/bin**.
    b. Enter **./run.sh** to start the SOA server. (Because you are not specifying a server profile, "default" will be used.)
  - **Microsoft Windows**
    a. Open a terminal and navigate to the **bin** directory by entering the command **chdir *SOA_ROOT*\jboss-as\bin**.
    b. Enter **run.bat** to start the SOA server. (Because you are not specifying a server profile, "default" will be used.)

Result

The server starts. Note that this will take approximately two minutes, depending on the speed of your hardware.

NOTE

To verify that there have been no errors, check the server log: **less *SOA_ROOT*/jboss-as/server/*PROFILE*/log/server.log**. As another check, open a web browser and go to http://localhost:8080. Make sure you can login to the admin console with the user name and password you have set.

Report a bug

6.3. TROUBLESHOOTING THE BOOT PROCESS

Errors in the server.log are indicated by the keyword "ERROR". If you see an error in the log, look through this list to find the cause:

1. "Address already in use" - There is already a server running on port 8080.
2. "Java not found" - The Java JRE may not be installed, or if it is, your PATH environment variable is not set to locate the java runtime.
3. "Class not found" - The CLASSPATH environment variable is not set properly. You really don't need to set this variable as the server startup script sets it for you.
4. If you see any of these errors, examine the server.log messages that come before and after the error message for additional information regarding the root cause of the error.

Report a bug

6.4. RUNNING THE "HELLO WORLD" QUICKSTART

6.4.1. Quickstart

The quickstarts are sample projects. Each one demonstrates how to use a specific piece of functionality in order to aid you in building services. There are several dozen quickstarts included in the **`SOA_ROOT/jboss-as/samples/quickstarts/`** directory. Build and deploy every quickstart by using **Apache Ant**.

Report a bug

6.4.2. Important Notes About Quickstarts

When intending to run a quickstart, remember the following points:

1. Each quickstart needs to be built and deployed using Apache Ant.
2. Each quickstart uses the **`samples/quickstarts/conf/quickstarts.properties`** file to store environment-specific configuration options such as the directory where the server was installed. You must create a **`quickstarts.properties`** file that matches your server installation. An example properties file (**`quickstarts.properties-example`**) is included.
3. Each quickstart has different requirements. These are documented in their individual **`readme.txt`** files.
4. Not every quickstart can run under every server profile.
5. The jBPM quickstarts require a valid jBPM Console user name and password. Supply these by adding them as properties in the **`SOA_ROOT/jboss-as/samples/quickstarts/conf/quickstarts.properties`** file:

```
# jBPM console security credentials
```

```
jbpm.console.username=admin
jbpm.console.password=adminpassword
```

The quickstarts that are affected by this requirement are **bpm_orchestration1**, **bpm_orchestration2**, **bpm_orchestration3** and **bpm_orchestration4**.

6. You can only execute some of the quickstarts (such as **groovy_gateway**) if the server is not running in *headless* mode. (The JBoss Enterprise SOA Platform is configured to launch in headless mode by default.)

IMPORTANT

Red Hat recommends that you run production servers in headless mode only.

Report a bug

6.4.3. Deploy the "Hello World" Quickstart on Your Test Server

Prerequisites

- Check that the setting in **SOA_ROOT/jboss-as/samples/quickstarts/conf/quickstarts.properties-example** matches the server configuration (**default** in a testing environment).

Procedure 6.2. Deploy the "Hello World" Quickstart

1. Check that the server has fully launched.
2. Open a second terminal window and navigate to the directory containing the quick start: **cd SOA_ROOT/jboss-as/samples/quickstarts/helloworld** (or **chdir SOA_ROOT\jboss-as\samples\quickstarts\helloworld** in Microsoft Windows).
3. Run **ant deploy** to deploy the quickstart. Look for messages such as this to confirm if the deployment was successful:

```
deploy-esb:
     [copy] Copying 1 file to
/jboss/local/53_DEV2/jboss-soa-p-5/jboss-as/server/default/deploy

deploy-exploded-esb:

quickstart-specific-deploys:
     [echo] No Quickstart specific deployments being made.

display-instructions:
     [echo]
     [echo] ******************
     [echo] Quickstart deployed to target JBoss ESB/App Server at
'/jboss/local/53_DEV2/jboss-soa-p-5/jboss-as/server/default/deploy'.
     [echo] 1.  Check your ESB Server console to make sure the
deployment was
executed without errors.
     [echo] 2.  Run 'ant runtest' to run the Quickstart.
```

```
     [echo] 3.  Check your ESB Server console again.  The Quickstart
should
have produced some output.
     [echo] ******************

deploy:

BUILD SUCCESSFUL
```

Also, check for this in the **SOA_ROOT/jboss-as/server/default/log/server.log**:

```
10:58:52,998 INFO  [NamingHelper] JNDI InitialContext properties:{}
11:00:58,154 INFO  [QueueService]
Queue[/queue/quickstart_helloworld_Request_esb] started,
fullSize=200000,
pageSize=2000, downCacheSize=2000
11:00:58,186 INFO  [QueueService]
Queue[/queue/quickstart_helloworld_Request_gw] started,
fullSize=200000,
pageSize=2000, downCacheSize=2000
11:00:58,427 INFO  [EsbDeployment] Starting ESB Deployment
'Quickstart_helloworld.esb'
```

4. Run the quickstart by issuing this command: **ant runtest**. When the quickstart is run, messages such as this are written to the **SOA_ROOT/jboss-as/server/default/log/server.log**:

```
11:03:02,190 INFO  [STDOUT]
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
11:03:02,191 INFO  [STDOUT] Body: Hello World
11:03:02,191 INFO  [STDOUT]
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
11:03:02,192 INFO  [STDOUT] Message structure:
11:03:02,192 INFO  [STDOUT] [Hello World].
```

Result

The words "Hello World" will appear on the server terminal. This message will also be appended to the **SOA_ROOT/jboss-as/server/default/log/server.log** file.

Report a bug

6.4.4. ant deploy

ant deploy compiles a quickstart's source code in the **build** directory, then generates an .ESB file (such as **Quickstart_helloworld.esb**) in the server profile's **deploy** directory. (Note that it generates .JAR files for BPEL quickstarts.) The server detects the presence of the new .esb archive and deploys it. In the .ESB archive is a deployment.xml file that **ant deploy** uses to configure a queue.

ant deploy also uses an XSL template to transform generic JMS queue names into the specific JMS queues needed by the target server's messaging provider. Ant selects the correct messaging provider by examining the server for a messaging provider deployment. Only JBoss Messaging, JBoss MQ, and

HornetQ are detected by the build script. Other messaging providers are not supported by the quick start. Ant then puts the **deployment.xml** file into the **build/META-INF** directory before including it in the same .ESB archive as the rest of the quickstart.

Report a bug

6.4.5. ant runtest

ant runtest sends an ESB-unaware "Hello World" message (which is a plain String object) to the JMS Queue (**queue/quickstart_helloworld_Request_gw**). It instructs Java to run the sender class (in the case of the "Hello World" quick start, this is called **org.jboss.soa.esb.samples.quickstart.helloworld.test.sendJMSMessage**). By doing so, it sends a message directly to the deployed process.

Report a bug

6.4.6. ant sendesb

The **ant sendesb** command sends an ESB message to the SOA Server. It sends the ESB-aware message directly to the ESB listener, meaning that it does not have to utilize a gateway.

Report a bug

6.4.7. Undeploy the "Hello World" Quickstart

Procedure 6.3. Task

1. Navigate to the quickstart's directory: **cd SOA_ROOT/jboss-as/samples/quickstarts/helloworld** (or **chdir SOA_ROOT\jboss-as\samples\quickstarts\helloworld** if you are running Microsoft Windows).
2. Run the **ant undeploy** command. You should see messages such as this displayed:

```
undeploy:
[delete] Deleting:
/jboss/local/53_DEV2/jboss-soa-p-5/jboss-
as/server/default/deploy/Quickstart_helloworld.esb

BUILD SUCCESSFUL
```

   And messages such as this written to the server.log:

```
11:10:08,205 INFO  [EsbDeployment] Stopping
'Quickstart_helloworld.esb'
11:10:08,577 INFO  [EsbDeployment] Destroying
'Quickstart_helloworld.esb'
```

Report a bug

6.5. STOP THE JBOSS ENTERPRISE SOA PLATFORM SERVER

Procedure 6.4. Stop the JBoss Enterprise SOA Platform Server

- **Stop the SOA server**
  Press **ctrl-c** in the *server window* (the terminal window where the SOA server was started).

Result

The server will shut down. Note that this process will take a few minutes. Look for this line in the **server.log** file to confirm that the server has shut down successfully:

```
        12:17:02,786 INFO  [ServerImpl] Shutdown complete
```

Report a bug

6.6. EXAMINING THE "HELLO WORLD" QUICKSTART

6.6.1. Overview of How the "Hello World" Quickstart Works

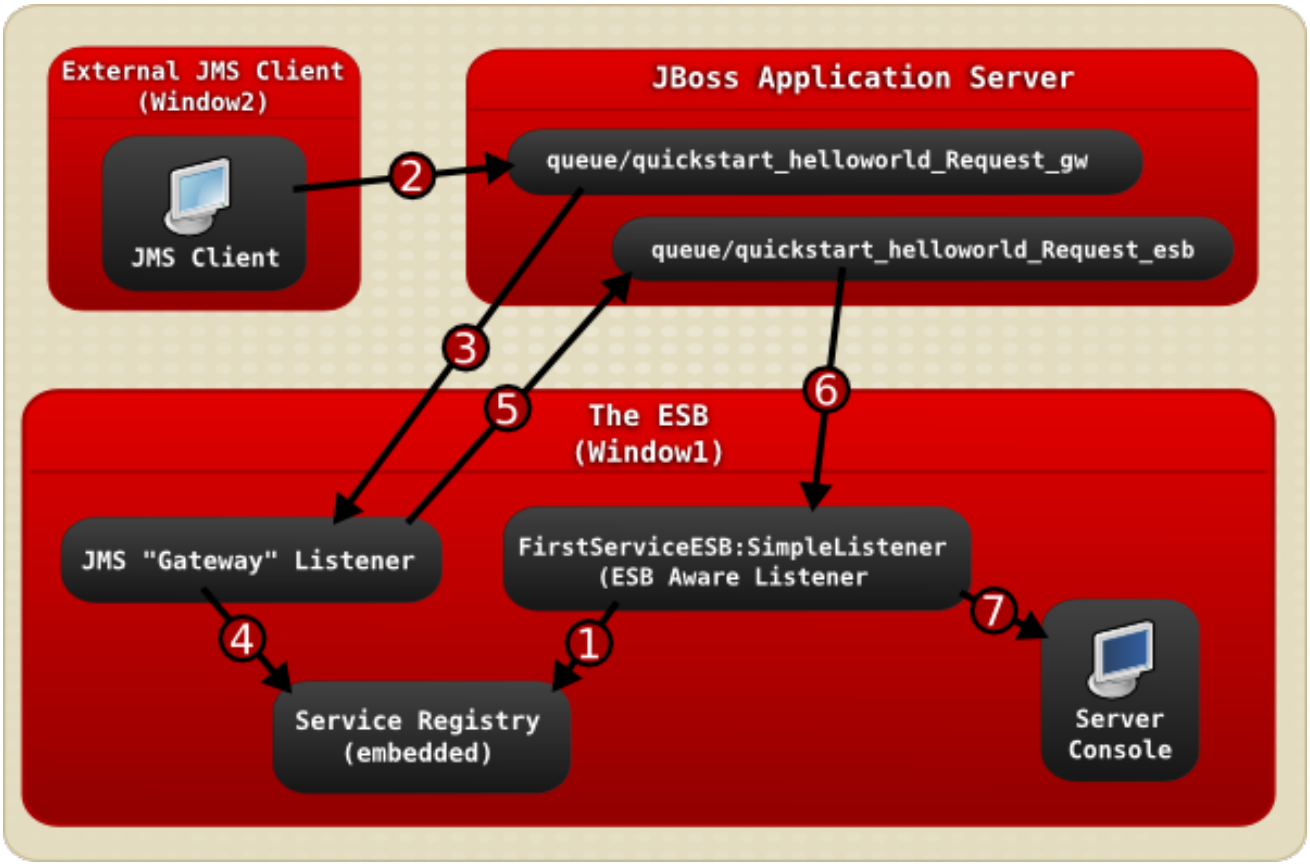


Figure 6.1. Image

1. The **JBoss Enterprise SOA Platform** server is launched in **Window1** and then the **FirstServiceESB:SimpleListener** service is added to the Service Registry service when the helloworld quickstart is deployed.

2. A JMS client sends an ESB-unaware "Hello World" message, (it is a plain **String** object), to the JMS Queue (**queue/quickstart_helloworld_Request_gw**).

3. The JMS Gateway Listener receives the ESB-unaware message and creates from it an ESB-aware message for use by ESB-aware end-points.

4. The **`JMS Gateway Listener`** uses the **`service registry`** to find the **`FirstServiceESB:SimpleListener`** service's *end-point reference* (EPR). In this case, the EPR is the **`queue/quickstart_helloworld_Request_esb`** JMS queue.

5. The **`JMS Gateway Listener`** takes the new ESB-aware message and sends it to the **`queue/quickstart_helloworld_Request_esb`** JMS queue.

6. The **`FirstServiceESB:SimpleListener`** service receives the message.

7. The **`FirstServiceESB:SimpleListener`** service extracts the payload from the message and outputs it to the console.

Report a bug

6.6.2. ESB Message

An ESB message is a message that takes the form defined by the **`org.jboss.soa.esb.message`** interface. This standardized format consists of a header, body (payload) and attachments. All ESB-aware clients and services communicate with one another using messages.

Report a bug

6.6.3. Components of an ESB Message

An ESB message is made up of the following components:

Header

The header contains such information as the destination end-point reference, the sender end-point reference, and where the reply goes. This is all general message-level functional information.

Context

This is additional information that further explains the message; for example, transaction or security data, the identity of the ultimate receiver or HTTP-cookie information.

Body

The actual contents of the message.

Fault

Any error information associated with the message.

Attachment

Any attachments (additional files) associated with the message.

Properties

Any message-specific properties.(For example, the jbossesb.message.id property specifies a unique value for each message).

Here is a code representation:

```
    <xs:complexType name="Envelope">
 <xs:attribute ref="Header" use="required"/>
 <xs:attribute ref="Context" use="required"/>
 <xs:attribute ref="Body" use="required"/>
 <xs:attribute ref="Attachment" use="optional"/>
 <xs:attribute ref="Properties" use="optional"/>
 <xs:attribute ref="Fault" use="optional"/>
</xs:complexType>
```

Report a bug

6.6.4. How Message Objects are Sent to the Queue

Overview

The JBoss Enterprise SOA Platform product uses a properties object that is populated with parameters to identify the presence of JNDI on the local server. It is then used as the parameter for a call to create a new Naming Context which is used to obtain the ConnectionFactory. The Connection Factory, in turn, creates the QueueConnection, which creates the QueueSession. This QueueSession creates a Sender object for the Queue. The Sender object is used to create an ObjectMessage for the sender and to then send it to the Queue.

Report a bug

6.6.5. Properties Object

The Properties object is a container for a generic list of keys and values. You should populate the Properties object with the parameters needed to identify JBoss JNDI on the local server and then use it as the parameter for a call to create a new Naming Context.

Report a bug

6.6.6. Naming Context

A Naming Context is an object that connects to a directory and allows the user to obtain objects by name reference.

Report a bug

6.6.7. ConnectionFactory

The ConnectionFactory is an object (**`org.jboss.jms.client.JBossConnectionFactory`**) that creates a QueueConnection. The Naming Contect obtains the ConnectionFactory from JNDI.

Report a bug

6.6.8. QueueConnection

The QueueConnection is an object created by the ConnectionFactory. It is used to create and start a QueueSession.

Report a bug

6.6.9. QueueSession

The QueueSession is created by the QueueConnection. It creates a Sender object for the queue and an ObjectMessage containing a string.

Report a bug

6.6.10. SOA_ROOT/jboss-as/samples/quickstarts/helloworld/build.xml

The **`build.xml`** file contains the instructions used by **`ant deploy`** to compile the quickstart's source code in the **`build`** directory. You can edit this file to add your own custom instructions.

Report a bug

6.6.11. SOA_ROOT/jboss-as/samples/quickstarts/helloworld/deployment.xml

The **`deployment.xml`** file is used by **`ant runtest`** to create and configure a messaging queue.

`ant deploy` generates the **`deployment.xml`** file in the **`build/META-INF`** directory during the compilation process. It then populates it when it determines which of the hard-coded JMS queues should be used. Once populated, the file is packaged as part of the .ESB archive. (Ant uses an XSL template to transform generic JMS queue names into the specific JMS queues required by the target server's messaging provider. It is from this template that the **`deployment.xml`** file is created.)

Report a bug

6.6.12. Messaging Queues

A message queue is a queue that is generated when an application is deployed. Messages are sent to these queues where they await the message listener.

Report a bug

6.6.13. Message Listeners

Message listeners encapsulate the communications end-points needed to receive SB-aware messages. Listeners are defined by services and their role is to monitor queues. They receive any messages as they land in those queues. When a listener receives a message, the ESB server calls the appropriate action class defined in the action definition. The methods in this class process the message. In other words, listeners act as inbound routers, directing messages to the action pipeline. When the message has been modified by the actions on the pipeline, the listener sends the result to the replyTo end-point.

You can configure various parameters for listeners. For instance, you can set the number of active worker threads.

There are two types of listeners: ESB-aware listeners and gateway listeners. Gateway listeners are different from ESB-aware listeners in that they accept data in different formats (such as objects in files, SQL tables and JMS messages). They then convert them from these formats to the ESB messaging format. By contrast, ESB-aware listeners can only accept messages that are in the **`org.jboss.soa.esb.message.Message`** format. Each gateway listener must have a corresponding ESB listener defined.

With ESB-aware listeners, RuntimeExceptions can trigger rollbacks. By contrast, with a gateway listener, the transaction simply sends the message to the JBoss ESB. The message is then processed asynchronously. In this way, message failures are separated from message receipts.

Report a bug

6.6.14. ESB-Awareness

If application clients and services are referred to as being ESB-aware, this means that they can understand the message format and transport protocols used by the SOA Platform's enterprise service bus.

Report a bug

6.6.15. Gateway Listener

A gateway listener is used to bridge the ESB-aware and ESB-unaware worlds. It is a specialized listener process that is designed to listen to a queue for ESB-unaware messages that have arrived through an external (ESB-unaware) end-point. The gateway listener receives the messages as they land in the queue. When a gateway listener "hears" incoming data arriving, it converts that data (the non-ESB messages) into the **`org.jboss.soa.esb.message.Message`** format. This conversion happens in a variety of different ways, depending on the gateway type. Once the conversion has occurred, the gateway listener routes the data to its correct destination.

Report a bug

6.6.16. Senders

Senders are created by QueueSessions. There is a sender for each queue. The Sender's **`send`** method is called by its QueueSession's ObjectMessage when **`ant runtest`** is executed. When this happens, the client sends a message to the queue.

Report a bug

6.6.17. Learn More About a Quickstart

To learn more about a particular quickstart:

Procedure 6.5. Task

1. Study the quickstart's **`readme.txt`** file.
2. Run the **`ant help`** command in the quickstart's directory.

Report a bug

6.7. THE "HELLO WORLD" QUICKSTART'S SOURCE CODE

6.7.1. SOA_ROOT/jboss-as/samples/quickstarts/helloworld/src

The "Hello World" quickstart's **`src`** directory contains the uncompiled programming instructions. The classes are in files nested in subdirectories. ant deploy compiles this source code.

Report a bug

6.7.2. SOA_ROOT/jboss-as/samples/quickstarts/helloworld/lib

The **`lib`** directory contains the classes needed by **`ant deploy`**, (in addition to the source code), to compile the quick start.

Report a bug

6.7.3. SOA_ROOT/jboss-as/server/SERVER_PROFILE/deploy

`ant deploy` moves the compiled version of a quick start (in the form of an .ESB archive file) from the **`build`** directory to the **`/jboss-as/server/default/deploy/`** directory. The JBoss Enterprise SOA Server is polling this directory and, when it detects the presence of a new .ESB file, it deploys it.

Report a bug

PART III. ACCOUNT MANAGEMENT

CHAPTER 7. CONFIGURING USER ACCOUNTS

7.1. USER ACCOUNTS

A user needs an account to be able to log in and use the JBoss Enterprise SOA Platform's various web-based consoles. The default security system reads plain text files (namely **soa-users.properties** and **soa-roles.properties**) to check a user's password and determine their level of access. SOA uses the Java Authentication and Authorization Service (JAAS) to authenticate user accounts.

WARNING

Red Hat does not recommend that you run production servers configured with user passwords in clear text files as it compromises security.

Report a bug

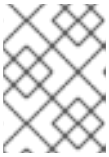
7.2. CREATE USER ACCOUNTS

Procedure 7.1. Add a New User

1. Open the **soa-users.properties** file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/conf/props/soa-users.properties**. Add the user's name and password on a new line, using this syntax: **username=password**.

   Here is an example for a user with the login name "Harold":

   ```
   harold=@dm1nU53r
   ```

   NOTE

   Any line in this file that begins with a hash (#) is ignored. (You can use this convention to temporarily disable a user account.)

2. Save the changes to the file and exit the text editor.

3. Open the **soa-roles.properties** file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/conf/props/soa-roles.properties**. Add the user and the roles you wish to assign to them on a new line, using this syntax: **username=role1,role2,role3**.

   ```
   harold=JBossAdmin,HttpInvoker,user,admin
   ```

NOTE

You can assign any number of roles. Note that a user must be assigned the **`JBossAdmin`**, **`HttpInvoker`**, **`user`** and **`admin`** roles in order to be able to log into the server consoles.

Any line in this file that begins with a hash (#) is ignored. You can use this convention to temporarily disable user roles.

4. Save the changes to the file and exit the text editor.

Result

The user will now be able to log in to the server console at http://localhost:8080. You do not have to restart the server.

Report a bug

7.3. SECURITY ROLES

Table 7.1. List of Security Permissions for JBoss Enterprise SOA Platform Console Users

Role	Description
JBossAdmin	The JBossAdmin role is required to log into the various management components of SOA. It is the primary role so all system administrators should be assigned this role.
HttpInvoker	The HttpInvoker role is used by the Http Invoker to access JNDIs and EJBs from remote locations.
user	This is used to grant user access to services deployed in SOA if they are configured to utilize the JAAS security domains. The jBPM Console relies on this one role only.
admin	This is used to grant administrative access to services deployed in SOA if they are configured to utilize the JAAS security domains.

Report a bug

7.4. JAVA AUTHENTICATION AND AUTHORIZATION SERVICE (JAAS)

The JAAS 1.0 API consists of a set of Java packages designed for user authentication and authorization. The API implements a Java version of the standard Pluggable Authentication Modules (PAM) framework and extends the Java 2 Platform access control architecture to support user-based authorization.

JAAS was first released as an extension package for JDK 1.3 and is bundled with JDK 1.6.

Report a bug

7.5. DISABLE A USER'S ACCOUNT

Procedure 7.2. Disable a User's Account

1. Open the **soa-users.properties** file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/conf/props/soa-users.properties**. Either delete the entire line containing the user's name and password or simply put a hash (#) in front of it to "comment it out."

   Here is an example for a user with the login name "Harold":

   ```
   #harold=@dm1nU53r
   ```

2. Save the changes to the file and exit the text editor.

Result

The user will no longer be able to log in to the server console. You do not have to restart the server.

Report a bug

PART IV. ADVANCED CONFIGURATION OPTIONS

CHAPTER 8. CONFIGURING THE DEFAULT DATABASE

8.1. HYPERSONIC DATABASE

Hypersonic is a database that comes included with the JBoss Enterprise SOA Platform product and works out-of-the-box. It is a demonstration database only and is not suitable for use on production systems. Use it for development and evaluation purposes only. Red Hat does not support the use of Hypersonic on a production system.

Report a bug

8.2. HYPERSONIC IS NOT SUPPORTED

The Hypersonic database is included for testing purposes only and is not supported.

Known issues with the Hypersonic Database include:

1. No transaction isolation thread and socket leaks (connection.close() does not tidy up resources)
2. Persistence quality (logs commonly become corrupted after a failure, preventing automatic recovery)
3. Database corruption stability under load (database processes cease when dealing with too much data)
4. Not viable in clustered environments

Report a bug

8.3. H2 DATABASE

H2 is an in-memory reference database that comes included with the JBoss Enterprise SOA Platform product and works out-of-the-box. Use it in testing environments only. It is not suitable for production systems and Red Hat does not support its use in such environments.

Report a bug

8.4. DATABASE CONFIGURATION TOOL

The Database Configuration Tool (found in the **`SOA_ROOT/jboss-as/tools/schema/`** directory) is an Apache Ant script. It sets the database to be used by the JBoss Enterprise SOA Platform. A list of the supported databases can be found at http://www.jboss.com/products/platforms/soa/supportedconfigurations/.

Report a bug

8.5. CONFIGURE A DATABASE FOR PRODUCTION SYSTEM USE

Prerequisites

- Apache Ant
- the database that you wish to use must already exist
- a user with permission to make changes to that database must already exist.
- the JDBC driver JAR file for the database must be in the server configuration's **`lib/`** directory.

The JBoss Enterprise SOA Platform uses a database to persist registry services and the Message Store. The default database is Hypersonic but this is not suitable for production systems and is not supported. You must switch to a supported database before running the JBoss Enterprise SOA Platform in a production environment.

WARNING

You can only use the Database Configuration Tool to change the database configuration once. Also, it must be run before any other changes are made. If you try to run the script on an installation that has already been configured, it may not work as intended.

1. **Back Up Your Server Profile**
   Make a copy of the server profile for which you plan to configure your database as the Database Configuration Tool modifies the configuration settings. **`cp -R SOA_ROOT/jboss-as/server/Profile /path/to/backup/folder`**.

2. **Running the Database Configuration Tool**
   Change to the directory containing the Database Configuration script: **`cd SOA_ROOT/jboss-as/tools/schema`**

3. **Run Ant**
   Run the **`ant`** command to launch the script.

4. **Enter Data**
   Following the prompts, enter the following information as it is requested:

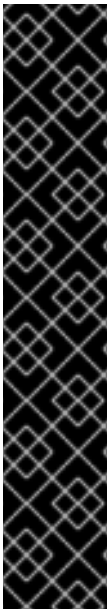
   - the type of database being used,
   - the name of the database,
   - the host name or IP Address of the database,
   - the TCP port being used for the database,
   - the user name needed to access the database, and
   - the password for this user account.

NOTE

You could also add these values directly to the **`build.properties`** file (found in the same directory) before running the script. The Database Schema Tool will not prompt you for these properties if it finds you have already added them to the file.

Result

The Tool updates the relevant configuration files and exits.



IMPORTANT

Users may experience deadlocks in jBPM 5 when running this process. To avoid this, add the following code to **`to jbpm.esb/jbpm-ds.xml`**:

```
<datasources>
    <local-tx-datasource>
        <jndi-name>BPELDB</jndi-name>
        ...
        <transaction-
isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
        <new-connection-sql>SET TRANSACTION ISOLATION LEVEL
SNAPSHOT;</new-connection-sql>
        ...
    </local-tx-datasource>
</datasources>
```

Report a bug

8.6. SOA_ROOT/JBOSS-AS/TOOLS/SCHEMA/BUILD.XML

The **`SOA_ROOT/jboss-as/tools/schema/build.xml`** contains the settings for the Database Schema Tool. It is used by the **`ant`** command to run through the database setting options.

Report a bug

8.7. SOA_ROOT/JBOSS-AS/TOOLS/SCHEMA/BUILD.PROPERTIES

This is the file in which the Database Schema Tool saves settings. You can also edit it by hand using a text editor.

Report a bug

CHAPTER 9. CONFIGURE THE SERVICE REGISTRY

9.1. SERVICE REGISTRY

A service registry is a central database that stores information about services, notably their end-point references. The default service registry for the JBoss Enterprise SOA Platform is jUDDI (Java Universal Description, Discovery and Integration). Most service registries are designed to adhere to the Universal Description, Discovery and Integration (UDDI) specifications.

From a business analyst's perspective, the registry is similar to an Internet search engine, albeit one designed to find web services instead of web pages. From a developer's perspective, the registry is used to discover and publish services that match various criteria.

In many ways, the Registry Service can be considered to be the "heart" of the JBoss Enterprise SOA Platform. Services can "self-publish" their end-point references to the Registry when they are activated and then remove them when they are taken out of service. Consumers can consult the registry in order to determine which end-point reference is needed for the current service task.

Report a bug

9.2. JUDDI REGISTRY

The jUDDI (Java Universal Description, Discovery and Integration) Registry is a core component of the JBoss Enterprise SOA Platform. It is the product's default service registry and comes included as part of the product. In it are stored the addresses (end-point references) of all the services connected to the Enterprise Service Bus. It was implemented in JAXR and conforms to the UDDI specifications.

Report a bug

9.3. JUDDI AND THE JBOSS ENTERPRISE SOA PLATFORM

jUDDI and the JBoss Enterprise SOA Platform

The JBoss Enterprise SOA Platform product includes a pre-configured installation of a jUDDI registry. You can use a specific API to access this registry through your custom client. However, any custom client that you build will not covered by your SOA Platform support agreement. You can access the full set of jUDDI examples, documentation and APIs from: http://juddi.apache.org/.

Report a bug

9.4. OTHER SUPPORTED SERVICE REGISTRIES

The JBoss Enterprise SOA Platform also supports these other UDDI registries:

- SOA Software SMS
- HP Systinet

Report a bug

9.5. UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI) REGISTRY

The Universal Description, Discovery and Integration Registry (UDDI) is a directory for web services. Use it to locate services by running queries through it at design- or run-time. Within an UDDI Registry, information is categorized in Pages. UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts.

The UDDI also allows providers to publish descriptions of their services. The typical UDDI Registry will contain a uniform resource locator (URL) that points to both the WSDL document for the web services and the contact information for the service provider.

A business publishes services to the UDDI registry. A client looks up the service in the registry and receives service binding information. The client then uses the binding information to invoke the service. The UDDI APIs are SOAP-based for interoperability reasons.

Report a bug

9.6. UDDI PAGE TYPES

Green Pages

Green Pages provide information that enables you to bind a client to the service being provided.

Yellow Pages

Yellow Pages are used to categorize businesses based upon the industries to which they belong.

White Pages

White Pages contain general information, such as the name, address and other contact details for the company providing the service.

Report a bug

9.7. THE SERVICE REGISTRY AND THE JBOSS ENTERPRISE SOA PLATFORM

The Service Registry is a key part of the JBoss Enterprise SOA Platform. When you deploy services to SOA Platform's ESB, their end-point references are stored in it.

Report a bug

9.8. JUDDI AND THE ESB

The JBoss Enterprise Service Bus directs all interaction with the Registry through the registry interface, the default version of which uses Apache Scout.

Report a bug

9.9. HOW THE REGISTRY WORKS

1. The JBoss Enterprise Service Bus funnels all interaction with the Registry through the registry interface.
2. It then calls a JAXR implementation of this interface.
3. The JAXR API needs to utilize a JAXR implementation. (By default, this is Apache Scout.)
4. Apache Scout, in turn, calls the Registry.

Report a bug

9.10. APACHE SCOUT

Apache Scout is an open source implementation of JAXR, created by the Apache Project.

There are currently four implementations of the **`org.jboss.soa.esb.scout.proxy.transportClass`** class, one each for SOAP, SAAJ, RMI and Embedded Java (Local).

Report a bug

9.11. JAVA API FOR XML REGISTRIES (JAXR)

Java API for XML Registries (JAXR) is an API that provides a standard way to develop for service registries.

Report a bug

9.12. REGISTRY INTERFACE

The Registry Interface is the means by which the ESB communicates with the Service Registry.

Report a bug

9.13. CONFIGURING THE REGISTRY

Introduction

Normally, you will automatically configure the jUDDI Registry when you run the Database Configuration Tool.

Read this section for advanced options on manually configuring it.

Report a bug

9.14. CONFIGURE THE REGISTRY MANUALLY

Procedure 9.1. Task

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.

2. Scroll down to the "registry" section and alter the settings as you wish.

```
<properties name="registry">
  <property name="org.jboss.soa.esb.registry.implementationClass"

value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImpl
"/>

  <property name="org.jboss.soa.esb.registry.factoryClass"
    value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>

  <property name="org.jboss.soa.esb.registry.queryManagerURI"

value="org.apache.juddi.v3.client.transport.wrapper.UDDIInquiryServi
ce#inquire"/>

  <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"

value="org.apache.juddi.v3.client.transport.wrapper.UDDIPublicationS
ervice#publish"/>
  <property name="org.jboss.soa.esb.registry.securityManagerURI"

value="org.apache.juddi.v3.client.transport.wrapper.UDDISecurityServ
ice#secure"/>

  <property name="org.jboss.soa.esb.registry.user" value="root"/>
  <property name="org.jboss.soa.esb.registry.password"
value="root"/>

  <property name="org.jboss.soa.esb.scout.proxy.uddiVersion"
value="3.0"/>
  <property name="org.jboss.soa.esb.scout.proxy.uddiNameSpace"
value="urn:uddi-org:api_v3"/>

  <property name="org.jboss.soa.esb.scout.proxy.transportClass"
    value="org.apache.ws.scout.transport.LocalTransport"/>
  <!-- specify the interceptors, in order -->
  <property name="org.jboss.soa.esb.registry.interceptors"

value="org.jboss.internal.soa.esb.services.registry.InVMRegistryInte
rceptor,
org.jboss.internal.soa.esb.services.registry.CachingRegistryIntercep
tor"/>
  <!-- The following properties modify the cache interceptor
behaviour -->
  <property name="org.jboss.soa.esb.registry.cache.maxSize"
value="100"/>
  <property name="org.jboss.soa.esb.registry.cache.validityPeriod"
```

```
value="600000"/>

  <!-- Organization Category to be used by this deployment. -->
  <property name="org.jboss.soa.esb.registry.orgCategory"
  value="org.jboss.soa.esb.:category"/>
</properties>
```

3. Save the file and exit.

Report a bug

9.15. TABLE OF REGISTRY CONFIGURATION OPTIONS

Table 9.1. Registry Properties

Property	Description
org.jboss.soa.esb.registry.implementationClass	This is a class that implements the JBoss ESB Registry interface. One implementation, **JAXRRegistryImpl**, is included. It uses the **JAXRRegistry** interface.
org.jboss.soa.esb.registry.factoryClass	This is the class name of the JAXR ConnectionFactory implementation.
org.jboss.soa.esb.registry.queryManagerURI	This is the URI used by JAXR to query services.
org.jboss.soa.esb.registry.lifeCycleManagerURI	This the URI that JAXR uses for editing.
org.jboss.soa.esb.registry.user	This is the username utilized for editing.
org.jboss.soa.esb.registry.password	This is the password for the specified user.
org.jboss.soa.esb.scout.proxy.uddiVersion	This is the UDDI version of the query.
org.jboss.soa.esb.scout.proxy.uddiNameSpace	This is the UDDI namespace.
org.jboss.soa.esb.scout.proxy.transportClass	This is the class used by **Apache Scout** to send items to the UDDI Registry.

Property	Description
`org.jboss.soa.esb.registry.interceptors`	This is the list of *interceptors* that are applied to the Registry. The ESB provides two interceptors, one for handling InVM registrations and one that supplies the Registry with a cache. The default interceptor list contains only one entry, that for the InVM interceptor.
`org.jboss.soa.esb.registry.cache.maxSize`	This is the maximum number of server entries allowed in the cache. If this value is exceeded, entries will be removed on a "Least Recently Used" basis. The default value is `100`.
`org.jboss.soa.esb.registry.cache.validityPeriod`	This is the period of validity that has been set for the caching interceptor. The value is specified in milliseconds and defaults to `600000` (ten minutes). Set this value to `0` to if you do not wish the cache to expire.
`org.jboss.soa.esb.registry.orgCategory`	This is the name of the ESB's *organization category*. The default is `org.jboss.soa.esb.:category`.

Report a bug

9.16. REGISTRY CONFIGURATION USE CASES

Here are some cases where a user would want to configure the registry manually:

- To change the queryManagerURI/lifeCycleManagerURI/securityManagerURI to match the locations of a UDDI registry.
- To change the registry user/password.
- To provide a custom org.jboss.soa.esb.registry.orgCategory
- To specify custom registry interceptors

Report a bug

9.17. EMBEDDING THE REGISTRY

If you want the server components to share a single jUDDI Registry, you should embed it. This method even allows multiple instances of the JBoss Enterprise SOA Platform itself to share the same registry.

Report a bug

9.18. EMBED THE REGISTRY

Procedure 9.2. Task

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.
2. Scroll down to the "registry" section and alter the settings like this:

```
<properties name="registry">

     <property name="org.jboss.soa.esb.registry.implementationClass"

value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImpl
"/>

     <property name="org.jboss.soa.esb.registry.factoryClass"
     value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>

     <property name="org.jboss.soa.esb.registry.queryManagerURI"

value="org.apache.juddi.registry.local.InquiryService#inquire"/>

     <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"

value="org.apache.juddi.registry.local.PublishService#publish"/>

     <property name="org.jboss.soa.esb.registry.securityManagerURI"

value="org.apache.juddi.registry.local.SecurityService#secure"/>

     <property name="org.jboss.soa.esb.registry.user"
value="jbossesb"/>
     <property name="org.jboss.soa.esb.registry.password"
value="password"/>

     <property name="org.jboss.soa.esb.scout.proxy.transportClass"
     value="org.apache.ws.scout.transport.LocalTransport"/>

</properties>
```

3. Save the file and exit.

Report a bug

9.19. CONFIGURING THE REGISTRY TO USE REMOTE METHOD INVOCATION

The Registry can be configured to utilize RMI. (The JBoss Enterprise SOA Platform deploys the remote method invocation service by default: the ESB starts the Registry within the **jbossesb.sar** archive and it is this same archive that also automatically registers a remote method invocation service.)

Report a bug

9.20. CONFIGURE THE REGISTRY TO USE REMOTE METHOD INVOCATION

Procedure 9.3. Task

1. Open the global configuration file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml**.

2. Scroll down to the "registry" section and alter the settings like this:

```
<properties name="registry">
  <property name="org.jboss.soa.esb.registry.implementationClass"

value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImpl
"/>

  <property name="org.jboss.soa.esb.registry.factoryClass"
    value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>

  <property name="org.jboss.soa.esb.registry.queryManagerURI"
    value="jnp://localhost:1099/InquiryService?
org.apache.juddi.registry.rmi.Inquiry#inquire"/>

  <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
    value="jnp://localhost:1099/PublishService?
org.apache.juddi.registry.rmi.Publish#publish"/>
  <property name="org.jboss.soa.esb.registry.securityManagerURI"
    value="jnp://localhost:1099/PublishService?
org.apache.juddi.registry.rmi.Publish#publish"/>

  <property name="org.jboss.soa.esb.registry.user"
value="jbossesb"/>
  <property name="org.jboss.soa.esb.registry.password"
value="password"/>

  <property name="org.jboss.soa.esb.scout.proxy.transportClass"
    value="org.apache.ws.scout.transport.RMITransport"/>
</properties>
```

3. Save the file and exit.

4. Open the **web.xml** file in your text editor.

5. Configure it like this:

```
<!--  uncomment if you want to enable making calls in juddi with rmi
-->
```

```
  <servlet>
   <servlet-name>RegisterServicesWithJNDI</servlet-name>
   <servlet-
class>org.apache.juddi.registry.rmi.RegistrationService</servlet-
class>
   <load-on-startup>1</load-on-startup>
  </servlet>
```

6. Save the file and exit.
7. Open the jUDDI configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/standard/deploy/juddi-service.sar/juddi.war/WEB-INF/juddi.properties`**.
8. Configure it like this:

```
# JNDI settings (used by RMITransport)
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.provider.url=jnp://localhost:1099
java.naming.factory.url.pkgs=org.jboss.naming
```

9. Save the file and exit.
10. Add **`scout-client.jar`** to the RMI client's class-path.

Report a bug

9.21. REMOTE METHOD INVOCATION USING A CUSTOM JNDI REGISTRATION OF THE RMI SERVICE

If, for some reason, you do not want to deploy the **`juddi.war`** archive, you can configure one of the Enterprise Service Bus components running in the same Java Virtual Machine as the jUDDI Registry to register the RMI service.

Report a bug

9.22. CONFIGURE RMI USING YOUR OWN JNDI REGISTRATION

1. For a local application, open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.
2. Scroll down to the "registry" section and alter the local settings like this:

```
<properties name="registry">
 <property name="org.jboss.soa.esb.registry.implementationClass"
value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImpl
"/>

 <property name="org.jboss.soa.esb.registry.factoryClass"
value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>
```

```
     <property name="org.jboss.soa.esb.registry.queryManagerURI"
value="org.apache.juddi.registry.local.InquiryService#inquire"/>

     <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
value="org.apache.juddi.registry.local.PublishService#publish"/>

     <property name="org.jboss.soa.esb.registry.securityManagerURI"
value="org.apache.juddi.registry.local.SecurityService#secure"/>

     <property name="org.jboss.soa.esb.registry.user"
value="jbossesb"/>
     <property name="org.jboss.soa.esb.registry.password"
value="password"/>

     <property name="org.jboss.soa.esb.scout.proxy.transportClass"
value="org.apache.ws.scout.transport.LocalTransport"/>
</properties>
```

3. Save the file and exit.

4. For a remote application, open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.

5. Scroll down to the "registry" section and alter the remote method invocation settings like this:

```
<properties name="registry">
 <property name="org.jboss.soa.esb.registry.implementationClass"
value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImpl
"/>

 <property name="org.jboss.soa.esb.registry.factoryClass"
value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>

     <property name="org.jboss.soa.esb.registry.queryManagerURI"
value="jnp://localhost:1099/InquiryService?
org.apache.juddi.registry.rmi.Inquiry#inquire"/>

     <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
value="jnp://localhost:1099/PublishService?
org.apache.juddi.registry.rmi.Publish#publish"/>

     <property name="org.jboss.soa.esb.registry.user"
value="jbossesb"/>
     <property name="org.jboss.soa.esb.registry.password"
value="password"/>

     <property name="org.jboss.soa.esb.scout.proxy.transportClass"
value="org.apache.ws.scout.transport.RMITransport"/>
</properties>
```

6. Save the file and exit.

7. Point the host names of the **queryManagerURI** and **lifeCycleManagerURI** classes to the host on which the jUDDI Registry is running (this is also where the local is running). Note that the local application needs to have access to a naming service.

8. Use these settings to register the application:

```
//Getting the JNDI setting from the config
Properties env = new Properties();
env.setProperty(RegistryEngine.PROPNAME_JAVA_NAMING_FACTORY_INITIAL,
factoryInitial);
env.setProperty(RegistryEngine.PROPNAME_JAVA_NAMING_PROVIDER_URL,
providerURL);
env.setProperty(RegistryEngine.PROPNAME_JAVA_NAMING_FACTORY_URL_PKGS
,
  factoryURLPkgs);

InitialContext context = new InitialContext(env);
Inquiry inquiry = new InquiryService();
log.info("Setting " + INQUIRY_SERVICE + ", " +
inquiry.getClass().getName());
mInquery = inquiry;
context.bind(INQUIRY_SERVICE, inquiry);
Publish publish = new PublishService();
log.info("Setting " + PUBLISH_SERVICE + ", " +
publish.getClass().getName());
mPublish = publish;
context.bind(PUBLISH_SERVICE, publish);
```

9. Include the **scout-client.jar** file on the RMI client's class-path.

Report a bug

9.23. COMMUNICATE WITH THE JUDDI REGISTRY VIA SOAP

You can make the Enterprise Service Bus communicate with the jUDDI Registry by using SOAP (via Apache Scout).

Report a bug

9.24. SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Simple Object Access Protocol (SOAP) is a lightweight protocol that enables the user to define the content of a message and to provide hints as to how recipients should process that message. SOAP is an XML-based communication protocol.

Report a bug

9.25. CONFIGURE APACHE SCOUT TO USE SOAP

Procedure 9.4. Task

1. Shut down the RMI service by "commenting out" the **web.xml** file's RegisterServicesWithJNDI servlet.
2. Deploy the **juddi.war** archive.
3. Configure the data-source.

   Here are some sample properties:

```
<properties name="registry">
 <property name="org.jboss.soa.esb.registry.implementationClass"
 value="org.jboss.internal.soa.esb.services.registry.JAXRRegistryImp
l"/>

 <property name="org.jboss.soa.esb.registry.factoryClass"
 value="org.apache.ws.scout.registry.ConnectionFactoryImpl"/>

    <property name="org.jboss.soa.esb.registry.queryManagerURI"
    value="http://localhost:8080/juddi/inquiry"/>

    <property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
    value="http://localhost:8080/juddi/publish"/>

    <property name="org.jboss.soa.esb.registry.user"
value="jbossesb"/>
    <property name="org.jboss.soa.esb.registry.password"
value="password"/>

    <property name="org.jboss.soa.esb.scout.proxy.transportClass"
    value="org.apache.ws.scout.transport.AxisTransport"/>
</properties>
```

Report a bug

9.26. JUDDI CONSOLE

The jUDDI Console is a web-based graphical interface that you must use in order to configure the jUDDI Registry. It is accessible at http://localhost:8080/uddi-console/.

Report a bug

9.27. GRANT ACCESS TO THE JUDDI CONSOLE

Prerequisites

- A user with the name "root" who has been assigned the security roles of "user" and "admin".

You must log in as a jUDDI Publisher named root to give anyone administration rights. Once a user has these administration rights, they can grant them to other users.

Procedure 9.5. Task

1. Open a web browser session and go to the jUDDI Console at http://localhost:8080/uddi-console/. Log in as root.
2. Click "Publisher".
3. From the Publisher ID list, click on the username.
4. Select the "Is Admin" checkbox.

Result

The user you selected now has administrative rights.

Report a bug

9.28. INSTALL JBOSS DEVELOPER STUDIO PLUG-INS FOR SOA

Prerequisites

- JBoss Developer Studio
- JBDS JBoss Tools Component

Procedure 9.6. Task

1. Launch JBoss Developer Studio
2. Go to JBoss Central screen
3. Download the plug-ins you require to do your SOA development work.

Report a bug

CHAPTER 10. ADVANCED SERVICE REGISTRY CONFIGURATION OPTIONS

10.1. CONFIGURE AN ALTERNATIVE JAXR IMPLEMENTATION

Procedure 10.1. Task

1. Choose a specific JAXR implementation.
2. Set the property to configure the class. (Because the **JBoss Enterprise SOA Platform** uses **Apache Scout** by default, this property is set to the Scout factory class, namely **`org.apache.ws.scout.registry.ConnectionFactoryImpl`**.)
3. Configure the JAXR implementation by providing it with the Registry's location. Do so by editing **`org.jboss.soa.esb.registry.queryManagerURI`**, **`org.jboss.soa.esb.registry.lifeCycleManagerURI`** and **`org.jboss.soa.esb.registry.securityManagerURI`**.

Report a bug

10.2. CONFIGURE AN ALTERNATIVE API TO JAXR

Procedure 10.2. Task

- Write a new **`SystinetRegistryImplemtation`** class and provide a reference to it from within this property.

Report a bug

10.3. USING TRANSPORTS

When using Apache Scout, you can set a special optional parameter: **`org.jboss.soa.esb.scout.proxy.transportClass`**. This is the transport class that facilitates communications between **Scout** and the jUDDI Registry.

If you are using Scout to communicate with jUDDI, leave the transport class as **`LocalTransport`** and configure the file to make use of the jUDDI registry's transports (InVM, RMI and WS). This file defines the Registry's nodes.

Report a bug

10.4. NODE

A node is a location in the service registry.

Report a bug

10.5. SELECT A TRANSPORT

Procedure 10.3. Task

1. Edit the node definition file: **`vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.sar/esb.juddi.client.xml`**

2. Use the node settings to select which transport to use:

```
<node>
     <!-- required 'default' node -->
     <name>default</name>
     <description>Main jUDDI node</description>
     <properties>
 <property name="serverName" value="localhost" />
  <property name="serverPort" value="8880" />
     </properties>
     <!-- JAX-WS Transport

<proxyTransport>org.apache.juddi.v3.client.transport.JAXWSTransport<
/proxyTransport>

<custodyTransferUrl>http://${serverName}:${serverPort}/juddiv3/servi
ces/custody-transfer?wsdl</custodyTransferUrl>

<inquiryUrl>http://${serverName}:${serverPort}/juddiv3/services/inqu
iry?wsdl</inquiryUrl>

<publishUrl>http://${serverName}:${serverPort}/juddiv3/services/publ
ish?wsdl</publishUrl>

<securityUrl>http://${serverName}:${serverPort}/juddiv3/services/sec
urity?wsdl</securityUrl>

<subscriptionUrl>http://${serverName}:${serverPort}/juddiv3/services
/subscription?wsdl</subscriptionUrl>

<subscriptionListenerUrl>http://${serverName}:${serverPort}/juddiv3/
services/subscription-listener?wsdl</subscriptionListenerUrl>

<juddiApiUrl>http://${serverName}:${serverPort}/juddiv3/services/jud
di-api?wsdl</juddiApiUrl>
     -->
    <!-- In VM Transport Settings

<proxyTransport>org.jboss.internal.soa.esb.registry.client.JuddiInVM
Transport</proxyTransport>

<custodyTransferUrl>org.apache.juddi.api.impl.UDDICustodyTransferImp
l</custodyTransferUrl>

<inquiryUrl>org.apache.juddi.api.impl.UDDIInquiryImpl</inquiryUrl>

<publishUrl>org.apache.juddi.api.impl.UDDIPublicationImpl</publishUr
l>
```

```
<securityUrl>org.apache.juddi.api.impl.UDDISecurityImpl</securityUrl
>

<subscriptionUrl>org.apache.juddi.api.impl.UDDISubscriptionImpl</sub
scriptionUrl>

<subscriptionListenerUrl>org.apache.juddi.api.impl.UDDISubscriptionL
istenerImpl</subscriptionListenerUrl>

<juddiApiUrl>org.apache.juddi.api.impl.JUDDIApiImpl</juddiApiUrl>
  -->
 <!-- RMI Transport Settings -->
<proxyTransport>org.apache.juddi.v3.client.transport.RMITransport</p
roxyTransport>
 <custodyTransferUrl>/juddiv3/UDDICustodyTransferService</custodyTra
nsferUrl>
 <inquiryUrl>/juddiv3/UDDIInquiryService</inquiryUrl>
 <publishUrl>/juddiv3/UDDIPublicationService</publishUrl>
<securityUrl>/juddiv3/UDDISecurityService</securityUrl>
<subscriptionUrl>/juddiv3/UDDISubscriptionService</subscriptionUrl>
<subscriptionListenerUrl>/juddiv3/UDDISubscriptionListenerService</s
ubscriptionListenerUrl>
<juddiApiUrl>/juddiv3/JUDDIApiService</juddiApiUrl>
<javaNamingFactoryInitial>org.jnp.interfaces.NamingContextFactory</j
avaNamingFactoryInitial>
 <javaNamingFactoryUrlPkgs>org.jboss.naming</javaNamingFactoryUrlPkg
s>
 <javaNamingProviderUrl>jnp://localhost:1099</javaNamingProviderUrl>
 </node>
```

3. By default, the Remote Method Invocation (RMI) settings are enabled. To switch transports, comment those ones out and enable the ones you want to use.

4. Save the file and exit.

Report a bug

10.6. REMOTE INVOCATION CLASS

As its name implies, a remote invocation class is a class that can be called from a remote machine. This can be useful for developers but can also lead to potential security risks.

Report a bug

10.7. TRANSPORT SETTINGS

When you configure a transport, you must specify the following items:

- a **`proxyTransport`**

- a URL for all of the supported UDDI application programming interfaces (**inquiry**, **publish**, **security**, **subscription**, **subscription-listener** and **custodytransfer**)
- a jUDDI application programming interface URL.
- the RMI transport (which also includes *JNDI* settings)

Report a bug

10.8. CONFIGURE APACHE SCOUT

Procedure 10.4. Task

1. Check that the **transportClass** is set to LocalTransport in the **uddi.xml** file.
2. Configure the **uddi.xml** file to utilize the Registry's transports (these being InVM, RMI and WS, respectively). The file is located in **SOA_ROOT/jboss-as/server/all/deploy/jbossesb.sar/esb.juddi.client.xml**
3. Create a schema in the jUDDI registry by adding the **jbossesb publisher** to it.

NOTE

The **product/install/jUDDI-registry** directory contains **database-create** scripts for most common databases. These files are located in **SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb-registry.sar/juddi-sql**.

NOTE

The system can generate the database automatically if the user has been granted permission to create tables. (The Registry can create a database of any type for which there exists an associated *Hibernate* dialect.)

4. Check that the **SOA_ROOT/jboss-as/server/default/deploy/jbossesb-registry.sar/esb.juddi.xml** and **SOA_ROOT/jboss-as/server/all/deploy/jbossesb.sar/esb.juddi.client.xml** exist. These files contain the Registry's configuration settings.

NOTE

If you want the software to communicate with another UDDI registry, use Apache Scout's JAXR transports. There are four implementations of this class and they are also based on SOAP, SAAJ, RMI and Embedded Java.

5. If you decide to change the transport, always change the query and life-cycle uniform resource indicators as well. Here is sample code that demonstrates how to do so:

```
SOAP
```

```
queryManagerURI http://localhost:8080/juddi/inquiry
lifeCycleManagerURI http://localhost:8080/juddi/publish
transportClass  org.apache.ws.scout.transport.AxisTransport

RMI
queryManagerURI  jnp://localhost:1099/InquiryService?
org.apache.juddi.registry.rmi.Inquiry#inquire
lifeCycleManagerURI  jnp://localhost:1099/PublishService?
org.apache.juddi.registry.rmi.Publish#publish
transportClass  org.apache.ws.scout.transport.RMITransport

Local
queryManagerURI
org.apache.juddi.registry.local.InquiryService#inquire
lifeCycleManagerURI
org.apache.juddi.registry.local.PublishService#publish
transportClass  org.apache.ws.scout.transport.LocalTransport
```

Report a bug

10.9. INTERCEPTOR

Interceptors are used by the Service Registry to intercept requests. They are stored in an interceptor stack. Each interceptor in a stack can do the following things:

- service the request
- provide direct responses to the request
- augment the responses received from a lower interceptor or registry implementation

There are two interceptors provided in the current implementation.

Report a bug

10.10. THE LOCALREGISTRYINTERCEPTOR

The **`LocalRegistryInterceptor`** is the class that is responsible for processing local services.

Report a bug

10.11. CONFIGURE THE INTERCEPTOR STACK

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.
2. Modify the org.jboss.soa.esb.registry.interceptors properties.
3. Save the file and exit.

Report a bug

10.12. INTERCEPTOR SETTINGS

Table 10.1. Interceptor Properties

Property	Description
org.jboss.internal.soa.esb.services.registry.InVMRegistryInterceptor	The InVM registry interceptor is responsible for handling any InVM end-point references. These are registered by any of the services executing within the same server instance. The information about the InVM EPR and its associated service will be cached within the interceptor, will not be propagated to subsequent interceptors and will be returned to the caller by augmenting results from subsequent interceptors or registry queries.
org.jboss.internal.soa.esb.services.registry.CachingRegistryInterceptor	The caching registry interceptor retains a cache of end-point references and their associated services, evicting information from the cache on a LRU basis or after the information has expired. The interceptor can be configured through the org.jboss.soa.esb.registry.cache.maxSize and org.jboss.soa.esb.registry.cache.validityPeriod properties within jbossesb-properties.xml

Report a bug

CHAPTER 11. SERVICE REGISTRY INTEGRATION WITH THE BPEL ENGINE

11.1. BPEL ENGINE

A BPEL engine executes BPEL business process instructions. The BPEL engine included as part of the JBoss Enterprise SOA Platform product is based on Apache ODE.

NOTE

It is recommended that you only open one BPEL console window in your browser. Failing to do so can result in seeing a blank window upon login or being unable to login from your second window. For details, see RIFTSAW-400.

Report a bug

11.2. BUSINESS PROCESS EXECUTION LANGUAGE (BPEL)

Business Process Execution Language (BPEL) is an OASIS-standard language for business rules orchestration. Refer to http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html for more information.

Report a bug

11.3. BPEL AND THE SERVICE REGISTRY

Because BPEL is integrated with the Service Registry, services can register themselves automatically as they are deployed.

This registration process utilizes the jUDDI client libraries. When a service is deployed, both it and its BindingTemplate (end-point reference) are registered and a partnerLinkChannel is created for each partnerLink. At the same time, the WSDL end-point is obtained from the UDDI.

Upon undeployment, the BindingTemplate is removed from the **UDDI Registry**.

Report a bug

11.4. ACTIVATE BPEL-SERVICE REGISTRY INTEGRATION

Procedure 11.1. Task

- Integration is turned on by default. To confirm this, open **`vi SOA_ROOT/jboss-as/server/PROFILE/deploy/riftsaw.sar/bpel.properties.xml`** and ensure that is set as follows: **`bpel.uddi.registration=true`**.

Report a bug

11.5. PARTNER LINK

A partner link is a link which establishes a relationship between a BPEL process and its client.

Report a bug

11.6. PARTNER LINK CHANNEL

A partner link channel is a communications channel that is used to interact with a client and the services integrated in a BPEL process.

Report a bug

11.7. ESB.JUDDI.CLIENT.XML

The **`SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.sar/esb.juddi.client.xml`** file is the client configuration file for the jUDDI Service Registry.

Report a bug

11.8. BPEL.PROPERTIES CONFIGURATION SETTINGS

Table 11.1. The UDDI-related properties in the bpel.properties file

attribute	type (default)	description
bpel.uddi.registration	boolean (true)	If set to 'false', the UDDI integration is turned off. The RiftSaw installation process sets this value to 'true' only if the jbossesb-registry.sar is detected containing a jUDDI v3 registry. In every other case it is automatically set to false.
bpel.webservice.secure	boolean (false)	The UDDI Registration process registers an WSDL AccessPoint in the BindingTemplate for the BPEL Service it is registering. The BPEL server exposes the service WSDL Endpoint on the WS stack (Currently Red Hat supports JBossWS and CXF). If your webservice stack is configured to use a secure protocol (such as https), you need to switch this setting to 'true'. (Note that this setting is used during the registration process only.)

attribute	type (default)	description
bpel.uddi.client.impl	String (org.jboss.soa.bpel.uddi.UDDIRegistrationImpl)	This is the name of the class that implements the org.jboss.soa.bpel.runtime.engine.ode.UDDIRegistration interface.
bpel.uddi.clerk.config	String (not used by default)	This defines the path to the **`bpel.uddi.client.xml`** configuration file. This can be left "commented out" if you want to use the **`riftsaw.sar/META-INF/riftsaw.uddi.xml`**. In this case, a **`bpel.uddi.clerk.manager`** must be defined.
bpel.uddi.clerk.manager	String (riftsaw-manager)	This defines the name of the ClerkManager that will be used if the riftsaw.uddi.xml is left commented out. This value should correspond to the name of the manager in the **`esb.juddi.client.xml`**. Note that if the bpel.uddi.clerk.config is defined, the bpel.uddi.clerk.manager setting is ignored.
bpel.uddi.clerk	String (BPEL_clerk)	This defines the name of the clerk that will be used. This value should correspond to the name of the clerk in the **`riftsaw.uddi.xml`**. (By default this is set to BPEL_clerk.)

attribute	type (default)	description
bpel.uddi.lookup	boolean (true)	If this is set to true, the creating process of the partner channel will do a lookup by serviceName in the UDDI, and a WSDL Endpoint is retrieved. This process makes it easier to move process deployment around within your server farm, without having to update the partnerlink WSDL files in your BPEL process deployments. If more than one end-point (BindingTemplate) is found, the default policy used by the ServiceLocator is 'PolicyLocalFirst'. Note that it is still a requirement to deploy the initial partnerlink WSDL file for each partnerLink.

NOTE

The names of both the ClerkManager and the Clerk itself are specified in the **`bpel.properties`** file.

Report a bug

11.9. CLERK

The clerk (**`org.apache.juddi.v3.client.config.UDDIClerk`**) is responsible for registering service end-points in the Service Registry.

Report a bug

11.10. SET THE PROPERTIES TO BE USED BY THE CLERK WHEN REGISTERING SERVICES

Procedure 11.2. Task

1. Open the **`esb.juddi.client.xml`** file in your text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.sar/esb.juddi.client.xml`**
2. Configure the settings. For example:

```
</nodes>
   <clerks registerOnStartup="false">
      <clerk name="SOAExample" node="default" publisher="root"
password="root"/>
```

```
    </clerks>
    </manager>
</uddi>
```

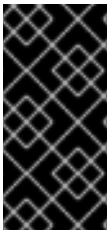
3. Save the file and exit.
4. Put another copy of the file in here (the files must always correspond: **SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb-registry.sar/juddi_custom_install_data/**
5. Save the file and exit.

Report a bug

11.11. DEFAULT SETTINGS FOR THE SERVICE REGISTRY CLERK

Table 11.2. Default Settings

Property	Value
keyDomain	esb.jboss.org
businessKey	redhat-jboss
serviceDescription	BPEL Service deployed by Riftsaw
bindingDescription	BPEL Endpoint deployed by Riftsaw



IMPORTANT

The **SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb-registry.sar/esb.juddi.xml** file contains a property called juddi.seed.always which is set to false. This means that it is always trying to load the root seed data when the server starts.

Report a bug

11.12. UDDI REGISTRATION

Upon deployment of a BPEL process, the process information is registered to the UDDI registry according to the BPEL4WS OASIS technote (http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040725.htm).

Report a bug

11.13. UDDI END-POINT LOOK-UP

If a BPEL service invokes another BPEL service (or a web service end-point in general), the BPEL Engine performs a lookup by serviceQName and portName (obtained from the WSDL). The result is stored in a client-side service cache, resulting in increased performance. To prevent the client-side cache from returning "stale" information, the cache is automatically invalidated by the UDDI registry using the Subscription API whenever changes occur in the registry.

Report a bug

CHAPTER 12. CONFIGURING A JAVA MESSAGE SERVICE PROVIDER

12.1. JAVA MESSAGE SERVICE

A Java Message Service (JMS) is a Java API for sending messages between two clients. It allows the different components of a distributed application to communicate with each other and thereby allows them to be loosely coupled and asynchronous. There are many different Java Message Service providers available. Red Hat recommends using HornetQ.

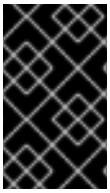
Report a bug

12.2. CONFIGURING A JAVA MESSAGE SERVICE PROVIDER

Introduction

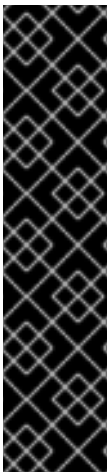
You can choose from a number of Java Message Service applications to use in conjunction with the JBoss Enterprise SOA Platform.

Instructions on configuring each of the supported JMS applications is provided in the following sections.



IMPORTANT

Note that the material in this book is not intended to replace your vendor-supplied Java Message Service documentation. Consult their books to learn about advanced capabilities, such as clustering.



IMPORTANT

In the following sections it is assumed that:

- the JMS provider is running on localhost
- the connection-factory is **`ConnectionFactory`**
- the destination-type is **queue**
- the destination-name is **`queue/A`**

Please bear this in mind when reading through the rest of this material.

Report a bug

12.3. SUPPORTED JAVA MESSAGE SERVICES

Refer to the supported configurations page for the complete list of supported JMS providers: http://www.jboss.com/products/platforms/soa/supportedconfigurations.

Any other JSR-914-compliant Java Message Service implementation (http://jcp.org/en/jsr/detail?id=914) such as Apache ActiveMQ or OracleAQ should also work. However only those JMS providers that are listed on the website above have been fully tested.

Report a bug

12.4. HORNETQ

HornetQ is a multi-protocol, asynchronous messaging system developed by Red Hat. HornetQ provides high availability (HA) with automatic client failover to guarantee message reliability in the event of a server failure. HornetQ also supports flexible clustering solutions with load-balanced messages.

Report a bug

12.5. CONFIGURE HORNETQ FOR USE AS THE JAVA MESSAGE SERVICE PROVIDER

WARNING

The HornetQ installer is designed to run only on a fresh installation and is not reversible, nor should it be run multiple times.

Procedure 12.1. Task

1. Go to the HornetQ Directory: **`cd SOA_ROOT/jboss-as/extras/hornetq`**.
2. **Deploy It**
   Run **`ant`**.

Report a bug

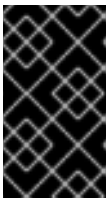
12.6. CONFIGURE JBOSS MESSAGING FOR USE AS THE JAVA MESSAGE SERVICE PROVIDER

Procedure 12.2. Task

1. **Configure the JMS Provider**
   To configure JBoss Messaging, open the configuration file in a text editor: **`vi jboss-esb.xml`**.
2. Modify these settings:

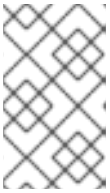
```
jndi-URL="localhost”
 jndi-context-factory="org.jnp.interfaces.NamingContextFactory"
 connection-factory="ConnectionFactory"
```

```
destination-type="queue"
destination-name="queue/myqueue"
```



IMPORTANT

Always include the **jboss-messaging-client.jar** file on the class-path. (This file is found in the **SOA_ROOT/jboss-as/client/jboss-messaging-client.jar** archive.)



NOTE

If you want to use load balancing and fail-over capabilities, you must configure JBoss Messaging for clustering. This functionality is constantly evolving. To learn how to configure it, please consult the JBoss Messaging documentation.

3. Check that JBoss Messaging's Service Binding Manager configuration matches that found in the **SOA_ROOT/seam/bootstrap/deploy/remoting-service.xml** file.

Report a bug

12.7. ACTIVEMQ JAVA MESSAGE SERVICE PROVIDER

IMPORTANT

ActiveMQ is not a certified or supported messaging provider for the JBoss Enterprise SOA Platform 5.

Report a bug

12.8. IBM WEBSPHERE MQ INSTALLATION OPTIONS

There are two ways in which to configure a Websphere MQ JMS provider:

1. As a JCA Provider by using the Websphere MQ JCA Adapter through the <jms-jca-provider> configuration.

2. As a standard JMS provider through the jms-provider configuration.

Report a bug

12.9. CONFIGURE IBM WEBSPHERE MQ FOR USE AS THE JAVA MESSAGE SERVICE PROVIDER

Procedure 12.3. Task

1. Copy the IBM Websphere wmq.jmsra.rar file to the JBoss Enterprise SOA Platform: **cp wmq.jmsra.rar SOA_ROOT/jboss-as/server/PROFILE/deploy/**

2. Copy the IBM Websphere com.ibm.mqetclient.jar to the JBoss Enterprise SOA Platform: **cp com.ibm.mqetclient.jar SOA_ROOT/jboss-as/lib/**.

3. Open run.conf in your text editor: **vi SOA_ROOT/jboss-as/bin/run.conf**

4. Add this setting by modifying the JAVA_OPTS in **run.conf**:

```
JAVA_OPTS="-DtraceEnabled=true -DtraceDestination=wmq_jca.trc
    -DtraceLevel=10 -DlogWriterEnabled=false"
```

5. Save the file and exit.

6. Open the jboss-esb.xml file in your text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.esb/META-INF/jboss-esb.xml**

7. Add these settings:

```
<jms-jca-provider name="WMQ-JCA" connection-
factory="MyAppXAConnectionFactory"
adapter="wmq.jmsra.rar"
       jndi-URL="10.12.10.110:1414/CH1"
       jndi-context-
factory="com.ibm.mq.jms.context.WMQInitialContextFactory">
    <property name="max-xa-sessions-per-connection" value="1" />
<jms-bus busid="quickstartEsbChannel">
  <jms-message-filter dest-type="QUEUE" dest-name="Q1"
transacted="true"/>
</jms-bus>
<activation-config>
     <property name="queueManager" value="QM1" />
     <property name="channel" value="CH1" />
     <property name="hostName" value="10.12.10.110" />
     <property name="port" value="1515" />
     <property name="transportType" value="CLIENT" />
     <property name="useJNDI" value="false" />
</activation-config>
</jms-jca-provider>
```

8. Save the file and exit.

9. Run **crtmqm -q QM1**

10. Run **strmqm QM1**

```
define channel (CH.1) chltype (RCVR) trptype (TCP)
start channel (CH.1)
define qlocal (Q1)
define listener(QM1.LISTENER) trptype(TCP) port(30001)
ipaddr(10.12.58.110)
start listener (QM1.LISTENER)
```

Report a bug

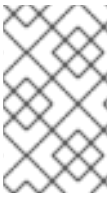
12.10. IBM WEBSPHERE MQ CONFIGURATION CHECKLIST

Check that the following items are present on the class-path:

- **com.ibm.mq.pcf.jar**
- **mqcontext.jar**
- **dhbcore.jar**
- **com.ibm.mq.jar** (client **JAR**)
- **com.ibm.mqjms.jar** (client **JAR**)

If you are using the **Websphere MQ v7.0 Client JAR** files, add the following to the items to the class-path:

- **com.ibm.mq.commonservices.jar**
- **com.ibm.mq.headers.jar**
- **com.ibm.mq.jmqi.jar**



NOTE

The client **JAR** files differ between MQ 5.3 and MQ 6.0. However the 6.0 **JAR** files should be backward-compatible. They are not open source and therefore Red Hat does not provide them. You will have to obtain them from your WAS and MQ installs.

Report a bug

12.11. JAVA MESSAGE SERVICE AND JNDI

The standard way for Java to use JMS is for it to look up the connection factory and the destination in JNDI. The JNDI context factory used for this is com.ibm.mq.jms.context.WMQInitialContextFactory. The default connection factory has the same name as your queue manager. Plain JMS connections require a connection factory to be set in JNDI.

Report a bug

12.12. CONFIGURE IBM WEBSPHERE MQ TO USE JNDI

Prerequisites

- IBM Websphere MQ must be installed on your system.
- Ensure Java is on your classpath.

Procedure 12.4. Task

1. Run this command: **/opt/mqm/java/bin/JMSAdmin -v**

```
java -
Dcom.ibm.msg.client.commonservices.log.outputName=$MQ_JAVA_DATA_PATH
/log -
Dcom.ibm.msg.client.commonservices.trace.outputName=$MQ_JAVA_DATA_PA
TH/trace -DMQJMS_INSTALL_PATH=$MQ_JAVA_INSTALL_PATH
com.ibm.mq.jms.admin.JMSAdmin $*
```

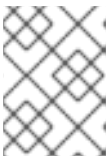
2. Open JMSAdmin.config in your text editor: **`vi JMSAdmin.config`**
3. Edit the file as per this sample code:

```
INITIAL_CONTEXT_FACTORY=com.ibm.mq.jms.context.WMQInitialContextFact
ory

PROVIDER_URL=localhost:30002/SYSTEM.DEF.SVRCONN

SECURITY_AUTHENTICATION=none
```

NOTE

If you have a listener on a different port, use it instead. (The default is 1414).

4. Save the file and exit.
5. Define objects with these commands: **`InitCtx> def cf(ConnectionFactory) qmgr(QM1) tran(CLIENT) host(10.12.58.105) BROKERQMGR(QM1)`** and **`InitCtx> def qcf(QueueConnectionFactory) qmgr(QM1) tran(CLIENT) host(10.12.58.105)`**
6. If you are having problems with JMSAdmin, troubleshoot the logs: **`less /var/mqm/errors/AMQERR01.LOG`** and **`less /var/mqm/qmgrs/QUEUE MANAGER NAME/errors/AMQERR01.LOG`**



NOTE

For more information, refer to http://www.ibm.com/developerworks/websphere/techjournal/0502_woolf/0502_woolf

Report a bug

12.13. JNDI OBJECTS VIEWABLE WITH JMSADMIN

`InitCtx> display qcf(QM1)`

- ASYNCEXCEPTION(-1)
- CCSID(819)
- CHANNEL(SYSTEM.DEF.SVRCONN)

- CLIENTRECONNECTTIMEOUT(1800)
- COMPHDR(NONE)
- COMPMSG(NONE)
- CONNECTIONNAMELIST(10.12.58.105(1414))
- CONNOPT(STANDARD)
- FAILIFQUIESCE(YES)
- HOSTNAME(10.12.58.105)
- LOCALADDRESS()
- MAPNAMESTYLE(STANDARD)
- MSGBATCHSZ(10)
- MSGRETENTION(YES)
- POLLINGINT(5000)
- PORT(1414)
- PROVIDERVERSION(UNSPECIFIED)
- QMANAGER(QM1)
- RESCANINT(5000)
- SENDCHECKCOUNT(0)
- SHARECONVALLOWED(YES)
- SSLFIPSREQUIRED(NO)
- SSLRESETCOUNT(0)
- SYNCPOINTALLGETS(NO)
- TARGCLIENTMATCHING(YES)
- TEMPMODEL(SYSTEM.DEFAULT.MODEL.QUEUE)
- TEMPQPREFIX()
- TRANSPORT(CLIENT)
- USECONNPOOLING(YES)
- VERSION(7)
- WILDCARDFORMAT(TOPIC_ONLY)

```
InitCtx> display q(Q1)
```

- CCSID(1208)
- ENCODING(NATIVE)
- EXPIRY(APP)
- FAILIFQUIESCE(YES)
- MDMSGCTX(DEFAULT)
- MDREAD(NO)
- MDWRITE(NO)
- MSGBODY(UNSPECIFIED)
- PERSISTENCE(APP)
- PRIORITY(APP)
- PUTASYNCALLOWED(AS_DEST)
- QMANAGER(QM1)
- QUEUE(Q1)
- READAHEADALLOWED(AS_DEST)
- READAHEADCLOSEPOLICY(DELIVER_ALL)
- REPLYTOSTYLE(DEFAULT)
- TARGCLIENT(JMS)
- VERSION(7)

```
InitCtx> display cf(ConnectionFactory)
```

- ASYNCEXCEPTION(-1)
- BROKERCCSUBQ(SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE)
- BROKERCONQ(SYSTEM.BROKER.CONTROL.QUEUE)
- BROKERPUBQ(SYSTEM.BROKER.DEFAULT.STREAM)
- BROKERQMGR(QM1)
- BROKERSUBQ(SYSTEM.JMS.ND.SUBSCRIBER.QUEUE)
- BROKERVER(UNSPECIFIED)
- CCSID(819)
- CHANNEL(SYSTEM.DEF.SVRCONN)
- CLEANUP(SAFE)

- CLEANUPINT(3600000)
- CLIENTRECONNECTTIMEOUT(1800)
- CLONESUPP(DISABLED)
- COMPHDR(NONE)
- COMPMSG(NONE)
- CONNECTIONNAMELIST(10.12.58.105(1414))
- CONNOPT(STANDARD)
- FAILIFQUIESCE(YES)
- HOSTNAME(10.12.58.105)
- LOCALADDRESS()
- MAPNAMESTYLE(STANDARD)
- MSGBATCHSZ(10)
- MSGRETENTION(YES)
- MSGSELECTION(CLIENT)
- OPTIMISTICPUBLICATION(NO)
- OUTCOMENOTIFICATION(YES)
- POLLINGINT(5000)
- PORT(1414)
- PROCESSDURATION(UNKNOWN)
- PROVIDERVERSION(UNSPECIFIED)
- PUBACKINT(25)
- QMANAGER(QM1)
- RECEIVEISOLATION(COMMITTED)
- RESCANINT(5000)
- SENDCHECKCOUNT(0)
- SHARECONVALLOWED(YES)
- SPARSESUBS(NO)
- SSLFIPSREQUIRED(NO)
- SSLRESETCOUNT(0)

- STATREFRESHINT(60000)
- SUBSTORE(BROKER)
- SYNCPOINTALLGETS(NO)
- TARGCLIENTMATCHING(YES)
- TEMPMODEL(SYSTEM.DEFAULT.MODEL.QUEUE)
- TEMPQPREFIX()
- TEMPTOPICPREFIX()
- TRANSPORT(CLIENT)
- USECONNPOOLING(YES)
- VERSION(7)
- WILDCARDFORMAT(TOPIC_ONLY)

Report a bug

12.14. CONFIGURE IBM WEBSPHERE MQ USING THE JMS-JCA-PROVIDER

Procedure 12.5. Task

1. **Configure the Adapter**
   The adapter name is **`wmq.jmsra.rar`**. Copy this file into the JBoss Enterprise SOA Platform's **`SOA_ROOT/jboss-as/server/PROFILE/deploy`** directory.

2. **Check the Context Factory**
   Ensure that the initial context factory is **`com.ibm.mq.jms.context.WMQInitialContextFactory`**.

3. **Configure the QueueManager**
   Ensure that the queueManager configuration is in the <activation-config>. You will need to configure a queue manager in your own Websphere MQ installation. (Note that all of your destination queues are configured in your queue manager.)

4. **Configure the Channel**
   Ensure that the channel configuration is in the <activation-config> property.

NOTE

You can configure a queue manager to use a number of different types of channel. Clients connect through server connection channels. A default server connection channel (SYSTEM.DEF.SVRCONN), is added as part of the Websphere MQ installation but it is a good idea to also create your own dedicated channels, especially if you are using extended (XA) transactions.

5. **Configure the transportType**
   Edit the transportType. The configuration for this is in the <activation-config> property. Websphere MQ supports two transport types (Client and Binding).

6. **Edit the Settings**
   Open the configuration file in a text editor and edit it.

   Here is an example of a <jms-jca-provider> configuration. It shows how to set a gateway provider configuration:

```
<jms-jca-provider adapter="wmq.jmsra.rar" connection-
factory="MyAppXAConnectionFactory" jndi-context-
factory="com.ibm.mq.jms.context.WMQInitialContextFactory" name="WMQ-
JCA">
 <property name="max-xa-sessions-per-connection" value="1"/>
  <jms-bus busid="quickstartGwChannel">
  <jms-message-filter dest-name="QUEUE1_JMS" dest-type="QUEUE"
transacted="true"/>
  </jms-bus>
   <activation-config>
    <property name="queueManager" value="TQM"/>
    <property name="channel" value="Q1CONN"/>
    <property name="hostName" value="localhost"/>
    <property name="port" value="1414"/>
    <property name="transportType" value="CLIENT"/>
   </activation-config>
</jms-jca-provider>
```

NOTE

It is actually the <jms-listener> configuration that determines the gateway/message-aware characteristics of a listener. However, as far as Websphere MQ is concerned, this is only of use as a gateway listener provider. In other words, this configuration would not work as a message-aware listener for Websphere MQ. This is because it does not specify an appropriate JNDI provider URL (as used by the ServiceInvoker) for routing ESB messages to the destinations (buses) defined on the provider.

7. **Configure the JCA Adapter**
   The <activation-config> configures the JCA adapter so that it obtains messages from the destinations. It does not determine how messages are to be delivered to the destinations. Here is an example of a configuration that could be used as a message-aware listener's provider:

```
<jms-jca-provider adapter="wmq.jmsra.rar" connection-
factory="MyAppXAConnectionFactory" jndi-URL="localhost:1414/CHANX"
jndi-context-
factory="com.ibm.mq.jms.context.WMQInitialContextFactory" name="WMQ-
JCA">
 <property name="max-xa-sessions-per-connection" value="1"/>
  <jms-bus busid="quickstartEsbChannel">
   <jms-message-filter dest-name="QUEUE2_JMS" dest-type="QUEUE"
transacted="true"/>
  </jms-bus>
 <activation-config>
```

```
  <property name="queueManager" value="TQM"/>
  <property name="channel" value="Q2CONN"/>
  <property name="hostName" value="localhost"/>
  <property name="port" value="1414"/>
  <property name="transportType" value="CLIENT"/>
 </activation-config>
</jms-jca-provider>
```

8. **Creating a Standard JMS Provider**
   The settings for creating a standard JMS provider configuration are similar:

```
<jms-provider connection-factory="MyAppConnectionFactory" jndi-
URL="localhost:1414/CHAN1" jndi-context-
factory="com.ibm.mq.jms.context.WMQInitialContextFactory"
name="JMS">
 <jms-bus busid="quickstartGwChannel2">
  <jms-message-filter dest-name="QUEUE3_JMS" dest-type="QUEUE"/>
 </jms-bus>
</jms-provider>
```

NOTE

With the standard provider, there is neither an adapter nor any <activation-config> configurations. Listeners receiving messages from the destinations defined in a standard provider do not use a JCA Adapter Inflow to obtain the messages. Instead, they need to use JNDI to find the destination and obtain the messages. This means that, for Websphere MQ, the jndi-URL must always be specified (remember that for JCA it is only required for destinations that service a Message-Aware listener).

9. **Save**
   Save the file and exit.

WARNING

You may encounter this exception if you are running **Websphere MQ 6.0**:

```
Message: Unable to get a MQ series Queue Manager or Queue
Connection. Reason: failed to create connection javax.jms.
JMSSecurityException: MQJMS2013: invalid security
authentication supplied for MQQueueManager
```

This is caused by a permissions issue. To fix this problem, add the user responsible for running the JBoss Enterprise Service Bus to the mqm group.

Report a bug

12.15. CONFIGURE IBM WEBSPHERE MQ TO USE THE JMS ROUTER

Procedure 12.6. Task

1. **Edit the JMSRouter Configuration**
   Open the JMSRouter's configuration file in a text editor and modify the settings in your **`jboss-esb.xml`**. Here is an example of a JMSRouter configuration for routing messages to Websphere MQ from inside an ESB action pipeline:

```
<action class="org.jboss.soa.esb.actions.routing.JMSRouter"
name="routeToORDERSQueue">
 <property name="jndi-context-factory"
value="com.ibm.mq.jms.context.WMQInitialContextFactory"/>
 <property name="jndi-URL" value="wmqserver:1414/CHANX"/>
 <property name="connection-factory" value="WMQConnectionFactory"/>
 <property name="jndiName" value="ORDERS"/>
</action>
```

2. **Save**
   Save the file and exit.

Report a bug

12.16. CONFIGURE IBM WEBSPHERE MQ TO USE AN EXTENDED TRANSACTION CLIENT

Prerequisites

- The Extended Transactional Client bundle from IBM (not part of the default Websphere MQ installation) must be installed on the class-paths of your application server and any external client applications
- The XA Connection Factories must be configured for your WMQ JNDI namespace. See http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/umj_pjcfm.html. (Once you have configured the XA Connection Factories, you can reference them using their JNDI name in the connection-factory property of the JNDI connection property.)

Procedure 12.7. Task

1. **Edit the jms-jca-provider Configuration**
   Edit the **`jboss-esb.xml`** file in a text editor. Here is how the jms-jca-provider configuration would look for an Extended Transaction Client configuration. (In this example, the XA Connection Factory that has been configured in the WMQ JNDI namespace is called WMQXAConnectionFactory.)

NOTE

Inflow related configurations (adapter, <activation-config> etc) were intentionally omitted as they are not relevant to the Extended Client configuration:

```
<jms-jca-provider name="WMQ" connection-
factory="WMQXAConnectionFactory"
 jndi-URL="wmqserver:1414/CHANXA_SEND"
 jndi-context-
factory="com.ibm.mq.jms.context.WMQInitialContextFactory">
  <property name="max-xa-sessions-per-connection" value="1" />
   <jms-bus busid="ordersGwChannel">
   <jms-message-filter dest-type="QUEUE" dest-name="ORDERS"
transacted="true"/>
   </jms-bus>
    <activation-config>
     <!--
     Used by inflow... not relevant to client
     See section on JMS and JCA.
     -->
  </activation-config>
</jms-jca-provider>
```

2. **Configuring for Use with the JMSRouter**
   This example shows how to use the same XA Connection Factory on the JMSRouter:

```
<action name="routeToORDERSQueue"
class="org.jboss.soa.esb.actions.routing.JMSRouter">
<property name="jndi-context-factory"
value="com.ibm.mq.jms.context.WMQInitialContextFactory"/>
<property name="jndi-URL" value="wmqserver:1414/CHANXA_SEND"/>
<property name="connection-factory" value="WMQXAConnectionFactory"/>
<property name="jndiName" value="ORDERS"/>
<property name="max-xa-sessions-per-connection" value="1"/>
<!-- etc... -->
</action>
```

IMPORTANT

An important point to note about Websphere MQ and XA is that it does not support both getting and putting messages concurrently on the same Queue Manager Channel. For this reason, it is a good idea to configure a dedicated channel for each component that gets or puts messages into a Websphere MQ destination in the context of XA transactions.

3. Set the max-xa-sessions-per-connection property in **`jboss-esb.xml`** to **`1`**.

4. **Save**
   Save the file and exit.

5. Configure a dedicated queue manager channel for each component that obtains or sends messages to a Websphere MQ destination.

6. Make sure the GET and PUT properties on each of the Websphere MQ destinations are not set to **`Inhibit`**. (This can be happen when the destination is created and it results in the ESB being unable to obtain or send messages.)

Report a bug

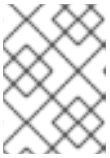
12.17. CONFIGURE IBM WEBSPHERE MQ TO USE PLAIN JMS INTERACTIONS

Prerequisites

- IBM Websphere MQ pre-configured as the JBoss Enterprise SOA Platform's JMS provider

Procedure 12.8. Task

1. Navigate to the **bin** directory: **cd /opt/mqm/java/bin**.
2. Launch **JMSAdmin**.
3. Use this program to administer your JNDI settings and obtain a ConnectionFactory for Plain JMS interactions.



NOTE

For more information about using JMSAdmin, refer to http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp.

Report a bug

12.18. VALIDATE YOUR IBM WEBSPHERE MQ INSTALLATION

Prerequisites

- IBM Websphere MQ pre-configured as the JBoss Enterprise SOA Platform's JMS provider

Procedure 12.9. Task

1. Make a queue manager called QM1: **crtmqm -q QM1**
2. Start runmqsc: **runmqsc QM1**
3. Input the following settings when prompted:

```
define channel (CH.1) chltype (RCVR) trptype (TCP)
start channel (CH.1)
define qlocal (Q1)
define listener(QM1.LISTENER) trptype(TCP) port(30001)
ipaddr(10.12.58.110)
start listener (QM1.LISTENER)
(end)
```

4. Exit the program.
5. Copy **com.ibm.mqetclient.jar** from your Websphere MQ installation to your server: **cp com.ibm.mqetclient.jar SOA_ROOT/jboss-as/PROFILE/lib/**

6. Copy **wmq.jmsra.ivt.ear** from your Websphere MQ installation to your server: **cp wmq.jmsra.ivt.ear SOA_ROOT/jboss-as/server/PROFILE/deploy**

7. Copy **wmq.jmsra.rar** from your Websphere MQ installation to your server: **cp wmq.jmsra.rar SOA_ROOT/jboss-as/server/PROFILE/deploy**

8. Create a data source file in a text editor (be sure to change the IP address to that of your WSMQ server): **vi SOA_ROOT/jboss-as/server/PROFILE/deploy/wsmq-ds.xml**

   Here is a sample configuration:

```
<?xml version="1.0" encoding="UTF-8"?>



<connection-factories>
  <!-- connection factory definition -->
  <tx-connection-factory>



    <jndi-name>IVTCF</jndi-name>
    <xa-transaction />
    <rar-name>wmq.jmsra.rar</rar-name>



    <connection-definition>
      javax.jms.ConnectionFactory
    </connection-definition>



    <config-property name="channel"
type="java.lang.String">SYSTEM.DEF.SVRCONN</config-property>
    <config-property name="hostName" type="java.lang.String">
      10.12.58.110
    </config-property>
    <config-property name="username"
type="java.lang.String">mqm</config-property>
    <config-property name="password"
type="java.lang.String">mqm</config-property>
    <config-property name="port"
type="java.lang.String">30001</config-property>
    <config-property name="queueManager"
type="java.lang.String">QM1</config-property>
    <config-property name="transportType"
type="java.lang.String">CLIENT</config-property>



    <security-domain-and-application>JmsXARealm</security-domain-
and-application>
  </tx-connection-factory>
```

```
  <!-- admin object definition -->
  <mbean code="org.jboss.resource.deployment.AdminObject"
      name="jca.wmq:name=ivtqueue">



    <attribute name="JNDIName">
      IVTQueue
    </attribute>
    <depends optional-attribute-name="RARName">
      jboss.jca:service=RARDeployment,name='wmq.jmsra.rar'
    </depends>
    <attribute name="Type">javax.jms.Queue</attribute>



    <attribute name="Properties">
      baseQueueManagerName=QM1
      baseQueueName=Q1
    </attribute>
  </mbean>
</connection-factories>
```

9. Save the file as and exit your text editor.
10. Launch a web browser and go to http://localhost:8080/WMQ_IVT/.

    This brings up the IVT ("Install Verification Test").
11. Run the normal test and the transactional test.

NOTE

For more information, refer to
http://www.ibm.com/developerworks/websphere/library/techarticles/0710_ritchie/071

Report a bug

12.19. IBM WEBSPHERE MQ JAVA MESSAGE SERVICE PROVIDER DIAGNOSTIC TRACING FUNCTIONALITY

The IBM WebSphere MQ Java Message Service component has a message tracing capability. This can be useful when you are trying to diagnose problems.

Report a bug

12.20. ENABLE DIAGNOSTIC TRACING FOR THE IBM WEBSPHERE MQ JCA ADAPTER

You can set the diagnostic trace as a property on the resource adapter, or in the Java Virtual Machine's system properties. With the JBoss ESB/Application, Red Hat recommends you use the JVM system properties approach.

However, on systems that are started via use of the **`./run.sh`** shell script, you should use the following approach:

Procedure 12.10. Task

1. **Open the run.conf File**
   Open the file in a text editor: **`vi SOA_ROOT/jboss-as/bin/run.conf`**.

2. **Edit the run.conf File**
   Appending the following lines onto the end of the file:

   ```
   # Settings to enable WebSphere MQ resource adapter trace
   JAVA_OPTS="$JAVA_OPTS -DtraceEnabled=true -
   DtraceDestination=wmq_jca.trc
   -DtraceLevel=10 -DlogWriterEnabled=false"
   ```

3. **Enable Client Logging**
   Still in the text editor, set the MQJMS_TRACE_LEVEL property:

   ```
   # Settings to enable WebSphere MQ resource adapter and client trace
   JAVA_OPTS="$JAVA_OPTS -DtraceEnabled=true -
   DtraceDestination=wmq_jca.trc
   -DtraceLevel=10 -DlogWriterEnabled=false -DMQJMS_TRACE_LEVEL=base"
   ```

4. **Save**
   Save the file and exit.

Report a bug

12.21. ENABLE DIAGNOSTIC TRACING FOR THE IBM WEBSPHERE MQ JAVA CLIENT

Procedure 12.11. Task

- **Call the enableTrace Static Method**
  Call the **`com.ibm.mq.MQEnvironment`**'s enableTrace static method.

Report a bug

12.22. CONFIGURE RED HAT ENTERPRISE (MRG) MESSAGING FOR USE AS THE JAVA MESSAGE SERVICE PROVIDER

Procedure 12.12. Task

1. **Edit the Configuration**

Open the configuration file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.esb/META-INF/jboss-esb.xml**.

2. Add these parameters:

```
<property name="jndi-prefixes" value="connectionFactory. ,
destination"/>
<property name="jndi-connection-factory"
value="org.apache.qpid.jndi.PropertiesFileInitialContextFactory"/>
<property name="connectionFactory.qpidConnectionFactory"
value="amqp:// guest:guest@clientid/virtualHost?
brokerlist='tcp://localhost:5672'"/>
<property name="destination.[queueName]Queue"
value="direct://amq.direct//[queueName]? routingkey=
[routingkeyname]"/>
```

3. **Save**
Save the file and exit.

4. **Add the Requisite Files to the Class-Path**
Add these .JAR files to the classpath:

- **qpid-common-0.6.jar**
- **qpid-client-0.6.jar**

Report a bug

12.23. CONFIGURE TIBCO ENTERPRISE MESSAGE SERVICE FOR USE AS THE JAVA MESSAGE SERVICE PROVIDER

Procedure 12.13. Task

1. **Edit the Configuration**
Open the configuration file in a text editor (**vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.esb/META-INF/jboss-esb.xml**) and set these parameters:

```
jndi-URL="tcp://localhost:7222”
jndi-context-
factory=”com.tibco.tibjms.naming.TibjmsInitialContextFactory"
connection-factory="QueueConnectionFactory"
destination-type="queue"
destination-name="myqueue"
```

2. **Save**
Save the file and exit.

3. **Add the Requisite Files to the Class-Path**
Add these .JAR files to the classpath (they are located in the **tibco/ems/clients/java** directory):

- **jaxp.jar**
- **jndi.jar**
- **tibcrypt.jar**
- **tibjmsapps.jar**
- **tibrvjms.jar**
- **jms.jar**
- **jta-spec1_0_1.jar**
- **tibjmsadmin.jar**
- **tibjms.jar**

Report a bug

12.24. CONFIGURING JBOSS TO USE THE SOA SERVICE MANAGER REGISTRY

1. To configure the settings, open the properties file: **vi SOA_ROOT/jboss-as/server/PROFILE/deploy/jbossesb.sar/jbossesb-properties.xml**
2. Locate the following lines:

```
<property name="org.jboss.soa.esb.registry.queryManagerURI"
value="org.apache.juddi.registry.local.InquiryService#inquire"/>
<property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
value="org.apache.juddi.registry.local.PublishService#publish"/>
```

   Change them to this:

```
<property name="org.jboss.soa.esb.registry.queryManagerURI"
value="http://<server-name>:9901/uddi/inquiry_v2"/>
<property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
value="http://<server-name>:9901/uddi/publish_v2"/>
<property name="org.jboss.soa.esb.registry.uddi.maxRows"
value="100"/>
```

3. Edit the queryManagerURI and the lifeCycleManagerURI to match the URLs from your installation of SOA Workbench:

```
<property name="org.jboss.soa.esb.registry.uddi.maxRows"
value="100"/>
<property name="org.jboss.soa.esb.registry.queryManagerURI"
value="http://<server-name>:9901/uddi/inquiry_v2"/>
<property name="org.jboss.soa.esb.registry.lifeCycleManagerURI"
value="http://<server-name>:9901/uddi/publish_v2"/>
```

4. Locate this line:

```
<property name="org.jboss.soa.esb.scout.proxy.transportClass"
value="org.apache.ws.scout.transport.LocalTransport"/>
```

Change it to this:

```
<property name="org.jboss.soa.esb.scout.proxy.transportClass"
value="org.apache.ws.scout.transport.AxisTransport"/>
```

5. Within jbossesb-properties.xml, change this to match the SOA Workbench username and password that you will be using:

```
<property name="org.jboss.soa.esb.registry.user"
value="administrator"/>
<property name="org.jboss.soa.esb.registry.password"
value="password"/>
```

6. Save the file and exit.

Report a bug

12.25. ADDING AXIS AND COMMON-DISCOVERY FILES

1. Download axis-bin-1_4.tar.gz from http://ws.apache.org/axis/ and extract axis-1.4.jar and commons-discovery-0.2.jar (or later).

2. Copy these two files into the lib directory: **`cp *jar SOA_ROOT/jboss-as/server/PROFILE/lib`**.

Report a bug

12.26. CONFIGURING THE SOA WORKBENCH

1. To apply the guest role to a guest user to grant anonymous access, go into the Organization Tree and click on the top-level registry node in the SOA Workbench hierarchy.

NOTE

SOA Workbench is a section of the SOA Service Manager Registry that acts as a third party UDDI registry provider.

2. Click on the second-level “Security” tab (within the Workbench hierarchy).

3. Click “Manage Role” for “Guest” under “Role Memberships.

4. Search for the user “guest”.

5. Select the checkbox next to “guest” and click “Apply".

Report a bug

12.27. CONFIGURING THE WORKFLOW

1. To change the workflow so that services are published upon creation, click on the “Configure” tab and then select “Workflow”.
2. View the Workflow Definition, save it to your local disk, and then change the function type line under @create to <function type="publish" /> and the step attribute to “100”.
3. Save this.
4. Update the workflow to the local copy you have just saved.

Report a bug

12.28. MANAGING SERVICES

- Use the "Search" option to find services. This feature also displays information about each service, including a description, its current state, its version number and the number of alerts it has generated. Services are registered in the SOA Service Workbench so manage them directly through its graphical user interface.

Report a bug

12.29. DIAGNOSTICS

- If you encounter an error when configuring and updating the **`Workflow`**, you can verify it has been modified correctly by checking that its configuration has changed from the original configuration (which looked like this):

```
<action id="1" name="@create">
 <results>
  <unconditional-result old-status="Created" status="Draft"
step="100" owner="${caller}"/>
 </results>
 <post-functions>
  <function type="setLifecycleStage">
   <arg name="stage">Design</arg>
   </function>
 </post-functions>
</action>
```

Rather, it should now look like this:

```
<action id="1" name="@create">
 <results>
  <unconditional-result old-status="Created" status="Draft"
step="100" owner="${caller}"/>
  </results>
  <post-functions>
   <function type="publish">
```

```
    </function>
   </post-functions>
  </action>
```

Report a bug

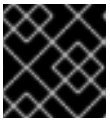
CHAPTER 13. OTHER JAVA MESSAGE SERVICE PROVIDER CONFIGURATION OPTIONS

13.1. CONFIGURE JAVA MESSAGE SERVICE LISTENERS AND GATEWAYS

Follow these steps to make your JMS listeners and gateways listen to queues and topics. **`jboss-esb.xml`** configuration file:

Procedure 13.1. Task

1. **Open the Configuration File**
   `vi jboss-esb.xml`
2. **Edit the File**
   Specify the following parameters:
   - jndi-URL
   - jndi-context-factory
   - jndi-pkg-prefix
   - connection-factory
   - destination-type
   - destination-name

   IMPORTANT

   Make sure you include your JMS provider's client's JAR files in the class-path.
3. **Save**
   Save the file and exit the text editor.

Report a bug

13.2. JMSCONNECTIONPOOL

As its name implies, the JmsConnectionPool pools Java Message Service sessions. It is used by every JMS-based component, including the listeners, couriers and routers.

Some Java Message Service providers limit the number of JMS sessions allowed per connection. As a result, the JBoss Enterprise Service Bus' JMS components utilize a mechanism that controls the maximum number of sessions created from each JMS connection that is managed by a single JmsConnectionPool instance.

Report a bug

13.3. SET THE MAXIMUM NUMBER OF SESSIONS PER CONNECTION

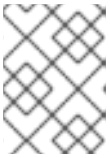
Procedure 13.2. Task

1. **Open the Relevant Configuration File**
   Open the component in question's JNDI configuration file in a text editor.

2. **Set the relevant Properties**
   You can edit these two properties: max-sessions-per-connection and max-xa-sessions-per-connection

   You must set the parameters as generic properties in the JMS provider configuration files. This example code demonstrates how to do so:

```
<jms-provider ...>
 <property name="max-sessions-per-connection" value="5" />
 <property name="max-xa-sessions-per-connection" value="1" />
 <!-- And add providers.... -->
</jms-provider>
```



NOTE

If you do not configure either of these parameters, then the JmsConnectionPool creates a single JMS connection and derives every session from it.

Report a bug

13.4. JNDI CONFIGURATION FILE PROPERTIES

Table 13.1. JNDI Configuration File Properties

Property	Description
max-sessions-per-connection	This is the maximum total number of sessions allowed per connection (for both extended (XA) and non-XA session instances.) The default value is the same as the maximum number of JMS sessions allowed for the JmsConnectionPool as a whole, (which you can configure by changing the setting in the **`jbossesb-properties.xml`** file. The default is twenty.)
max-xa-sessions-per-connection	This is the maximum number of XA sessions allowed per connection. The default value is the same as that which has been set for max-sessions-per-connection.

Report a bug

13.5. JMS-JCA-PROVIDER

The jms-jca-provider is a dedicated configuration for the JMS Java Connector Architecture (JCA). The jms-jca-provider facilitates JCA message inflow by providing two ways of connecting to the JMS provider, these being a gateway and an ESB-aware JMS message listener.

Report a bug

13.6. CONFIGURE THE JMS-JCA-PROVIDER

Procedure 13.3. Task

1. **Edit the Configuration File**
   Open the configuration file in a text editor and modify the settings. Here is a sample:

```
<jms-jca-provider connection-factory="XAConnectionFactory"
name="JBossMessaging">
 <jms-bus busid="ordersGwChannel">
  <jms-message-filter dest-name="queue/orders" dest-type="QUEUE"
transacted="true"/>
 </jms-bus>
  <activation-config>
   <property name="dLQMaxResent" value="5"/>
  </activation-config>
</jms-jca-provider>
```

2. **Save**
   Save the file and exit.

Report a bug

13.7. JMS-JCA-PROVIDER CONFIGURATION OPTIONS

The jms-jca-provider settings configure two different parts of the system, these being as follows:

1. The JCA inflows (which are on the server side).

   You can configure these items for them:

   - The JCA Adapter Name. This is configured as an attribute on the <jms-jca-provider>element.
   - The JCA Provider Adapter JNDI. This is configured as an attribute on the <jms-jca-provider> element.
   - The JCA Endpoint Class. This is configured as an attribute on the <jms-jca-provider> element.
   - The transacted flag. This is configured as an attribute on the <jms-jca-provider> element.
   - The message type. This is configured as an attribute on the <jms-jca-provider> element.
   - The JCA bridge. This is configured as an attribute on the <jms-jca-provider> element.

- The JCA adapter activation configuration. This is configured in the <activation-config> element inside the <jms-jca-provider>.

NOTE

The JCA activation configuration also extracts some configuration properties for the JCA Inflow from the <jms-bus> and <jms-message-filter> that are nested inside the <jms-jca-provider> element. These include the destination type, destination name and message selector.

2. The JMS connection details (which are on the client side), used to deliver messages to the JMS JCA inflows (which are either gateways or ESB-aware listeners.)

   These settings are used to generate the end-point reference for clients connecting to the JMS JCA inflow. They include the following:

   - The JMS connection factory to be used by the client. This is configured as an attribute on the <jms-jca-provider> element.
   - The JMS JNDI properties to be used by the client for connecting to the JMS provider. These are configured as an attributes on the <jms-jca-provider> element (**`jndi-*`**).
   - The JMS bus endpoint destination configurations. These are configured on the <jms-message-filter> elements inside the <jms-bus> within the <jms-jca-provider> element.

Make sure that the JMS client connection configurations outlined above are not configured to connect through JCA managed resources. The JMS client connection configurations should connect directly to the JMS Provider and not through JCA managed resources. This is important because this information is built into the end-point references used by the JBoss Enterprise SOA Platform's **`ServiceInvoker`** class, as well as other components that use the system's connection pooling functionality. (It is important to avoid using this to pool JCA-managed connections.)

These are the components that utilize the connection pooling feature:

1. Routing actions that use the **`ServiceInvoker`** class such as the static- and content-based routers.
2. Any gateway listeners that deliver messages to an action pipeline that is serviced by an ESB-aware JMS listener that, in turn, uses a JCA JMS provider. (Most of these gateway listeners deliver messages using the **`ServiceInvoker`** class.)
3. The JMS Router.

Make sure you connect these components directly to the JMS provider and not through local JCA-managed resources.

Report a bug

13.8. JNDI EXTENSION PROPERTIES

By default, the JNDI Java Message Service's resource retrieval configuration inherits every property that is prefixed with **`java.naming`**.

Some Java Message Service providers specify different naming prefixes. In order to support these, the JBoss Enterprise SOA Platform allows you to specify custom property prefixes for each provider.

Report a bug

13.9. CONFIGURE JNDI EXTENSION PROPERTIES

Procedure 13.4. Task

1. **Add the jndi-prefixes for the Appropriate JMS Provider Element**
   Open the configuration file in a text editor and add the jndi-prefixes for the appropriate JMS provider element with a comma-separated list of the additional prefixes you want to use:

   ```
   <jms-provider name="JMS" connection-factory="ConnectionFactory">
   <property name="jndi-prefixes” value="test.prefix." />
   <property name="test.prefix.extension1" value="extension1" />
   <property name="test.prefix.extension2" value="extension2" />
   </jms-provider>
   ```

   NOTE

   You can also configure the extensions property in this same location.

2. **Save**
   Save the file and exit.

Report a bug

CHAPTER 14. CONFIGURING JBPM

14.1. JBPM

The JBoss Business Process Manager (jBPM) is a workflow management tool that provides the user with control over business processes and languages. jBPM 3 is used as default.

Report a bug

14.2. JBPM 3 AND JBOSS ENTERPRISE SOA PLATFORM INTEGRATION

The default way in which the JBoss Enterprise SOA Platform integrates with jBPM 3 is via the JMS/JCA-Inflow:

```
<service name="message" factory="org.jbpm.msg.db.DbMessageServiceFactory"
/>
<service name="scheduler"
factory="org.jbpm.scheduler.db.DbSchedulerServiceFactory" />

<bean name="jbpm.job.executor" class="org.jbpm.job.executor.JobExecutor">
    ...
</bean>
```

JMS/JCA inflow is the only supported jBPM/ESB integration configuration for the JBoss Enterprise SOA Platform. The **jbpm.esb** only provides support for the JBoss Messaging JMS provider.

IMPORTANT

The jBPM enterprise module is not supported. (This consists of message and scheduler services based on JMS and EJB timers respectively.)

The simulation module is also unsupported. (This is a process analysis and optimization tool.)

Report a bug

14.3. CHANGE THE JBPM INTEGRATION SERVICE

Procedure 14.1. Task

1. Go into the **jbpm.esb** archive's **config** directory.
2. Locate the configuration file for the service you want to use.
3. To replace the active configuration, remove the one found in the **jbpm.esb** directory and replace it with the configuration file you wish to use (having first removed the **.config** suffix from its name).

Report a bug

14.4. JBPM JOB EXECUTOR

The jBPM Job Executor is a web listener. It is only supported if the Enterprise Service Bus is not deployed on the execution.

Report a bug

14.5. CONFIGURE THE JBPM JOB EXECUTOR

Procedure 14.2. Task

1. To use the jBPM Job Executor in a web application, open the application's **web.xml** file in a text editor and add the following code:

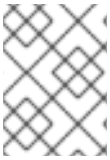
```
<!-- Job executor launcher -->
 <listener>
  <description>
   Starts the job executor on initialization and stops it on
destruction.
  </description>
   <listener-class>org.jbpm.web.JobExecutorLauncher</listener-class>
 </listener>
<!-- Job executor launcher -->
```

2. Save the file and exit.

Report a bug

14.6. USE A REMOTE JAVA MESSAGE SERVICE PROVIDER

If you want to make the JMS-based message and scheduling services reference remote JMS providers, you must add a provider adapter configuration for the local JNDI.

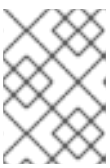


NOTE

The provider adapter is an instance of **JMSProviderAdapter**, as used within the standard application server's JCA Inflow configuration.

Procedure 14.3. Task

1. Open a **service.xml** file in a text editor and add an MBean to it

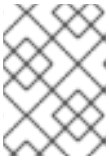


NOTE

You can either make this specific to the jBPM (adding it to the **jbpm-service.xml** file) or it can be shared with other **JCA inflow** configurations.

The following example code creates a provider adapter which refers to a remote JMS provider executing at this address: 192.168.1.1:1099.

```
<mbean code="org.jboss.jms.jndi.JMSProviderLoader"
name="jboss.messaging:service=JMSProviderLoader,name=RemoteProviderL
oader">
   <attribute name="ProviderName">RemoteProviderAdapter</attribute>
   <attribute
name="ProviderAdapterClass">org.jboss.jms.jndi.JNDIProviderAdapter</
attribute>
   <attribute
name="QueueFactoryRef">XAQueueConnectionFactory</attribute>
   <attribute
name="TopicFactoryRef">XATopicConnectionFactory</attribute>
   <attribute name="Properties">
      java.naming.provider.url=192.168.1.1:1099
   </attribute>
</mbean>
```



NOTE

Use the value set for the Properties attribute to create the **`InitialContext`** for the JNDI.

2. Provide references to the provider adapter within the **`JMSMessageServiceFactory`** file. Open it in a text editor: **`vi JMSMessageServiceFactory`**

3. Add the location of the JNDI in the providerAdapterJNDI field:

```
<service name="message">
  <factory>
    <bean
class="org.jboss.soa.esb.services.jbpm.integration.msg.JmsMessageSer
viceFactory">
      <field name="providerAdapterJNDI"><string
value="RemoteProviderAdapter"/></field>
    </bean>
  </factory>
</service>
```

4. Save the file and exit.

5. Provide references to the provider adapter within the **`JMSSchedulerServiceFactory`** file in a text editor: **`vi JMSSchedulerServiceFactory`**

6. Add the location of the JNDI in the providerAdapterJNDI field.

7. Save the file and exit.

Report a bug

CHAPTER 15. ADVANCED INSTALLATION OPTIONS

15.1. JBOSS_HOME ENVIRONMENTAL VARIABLE

The JBOSS_HOME environmental variable is an optional **.bash_profile** setting. You do not normally need to configure this setting but some scripts and third-party software require it.

IMPORTANT

If you have more than one JBoss server installed on the one machine, Red Hat recommends that you avoid using this setting if you possibly can or only setting it within those scripts that require it.

Report a bug

15.2. CONFIGURE THE JBOSS_HOME ENVIRONMENTAL VARIABLE

Procedure 15.1. Task

1. **Add the Variable on a Red Hat Enterprise Linux System**
   Open the file with the **vi ~./bash_profile** command and add the following line:

   ```
   export JBOSS_HOME=path
   ```

   (export **JBOSS_HOME=/path/to/SOA_ROOT**.)

2. Save and exit.

3. **Add the Variable on a Microsoft Windows System**
   Click on **Control Panel** → **System** → **Advanced** → **Environment Variables** and then **New**. Set the variable name to JBOSS_HOME and the variable value to the SOA_ROOT directory.

Report a bug

15.3. NATIVE COMPONENTS PACKAGE

The Native Components package is an optional component for the JBoss Enterprise Application Platform that incorporates native operating system components and connectors for web servers.

Report a bug

15.4. INSTALL NATIVE JBOSS COMPONENTS

Procedure 15.2. Task

- Refer to the JBoss Enterprise Application Platform's Installation Guide for instructions.

NOTE

That Guide instructs you to install the files into the **`jboss-eap-5.1`** directory. Substitute your SOA_ROOT every time you see mention of that directory.

Report a bug

15.5. MRG-M

MRG-M is the messaging component of Red Hat's Messaging/Realtime/Grid offering.

Report a bug

15.6. INSTALL MRG-M

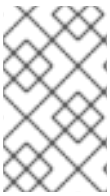
Procedure 15.3. Task

1. **Download the Package**
   Download the MRG JCA Adapter from the Red Hat Customer Portal. Save it in the **`SOA_ROOT/jboss-as/server/PROFILE/deploy`** directory.

2. **Create Queues**
   To create queues, use the **`qpid-config add queue`** command.

3. **Edit the Configuration File**
   Open the configuration file: **`vi qpid-jca-ds.xml`**. In here, there are three JCA administration objects. You must set the destination addresses for the inbound (MRG-ESB_GW) and outbound (MRG-ESB_RESP) queues and the routingKey for the outbound queue.

4. **Save**
   Save the file and exit.



NOTE

The QpidConnectionFactory is the connection factory used to send out messages and is utilized in the JMSRouter action. Do not try to use the other two CF administration objects as these do not work with the ESB.

Report a bug

15.7. FILE TRANSFER PROTOCOL

The File Transfer Protocol (FTP) is the standard protocol for sending files across networks. It can transmit text and binary files.

Report a bug

15.8. FILE TRANSFER PROTOCOL AND THE JBOSS ENTERPRISE SOA PLATFORM

The JBoss Enterprise SOA Platform renames all files so that they have the same filename prior to their transmission via FTP. (This prevents the individual files from being processed until the software renames them at the other end.) Unfortunately, some FTP servers retain a lock on the file after it has been written, which prevents the renaming process from taking place. If this problem occurs, the software will make a pre-defined number of attempts (the default being ten) to rename the files. If it still cannot do so after the maximum number of attempts, it generates an error message.

Report a bug

15.9. CONFIGURE FILE TRANSFER PROTOCOL SETTINGS

Procedure 15.4. Task

1. **Edit the Properties**
   Open the global settings file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**. Search for org.jboss.soa.esb.ftp.renameretry and alter the value for this setting. It will look like this:

   ```
   <property name="org.jboss.soa.esb.ftp.renameretry" value="10"/>
   ```

2. **Save**
   Save the file and exit.

Result

This changes the number of retry attempts.

NOTE

Changing the settings in this file has a global effect. To do it on a per-instance basis, change the setting in the specific end-point reference instead.

Report a bug

PART V. SECURITY

CHAPTER 16. SECURING YOUR SYSTEM

16.1. SECURITY ASSERTION MARKUP LANGUAGE (SAML)

Security Assertion Markup Language (SAML) is a framework compiled in XML. It is used to sercurely pass information between services. It is most commonly used for authentication and identification. With SAML, users can login once and have their identity verified by SAML, thus negating the need to re-enter their credentials repeatedly.

The JBoss ESB uses the SAML provided by PicketLink Project via JAAS Login Modules. It provides users with the ability to assign and validate SAML security tokens.

Report a bug

16.2. ISSUING A SAML SECURITY TOKEN

Procedure 16.1. Task

1. Obtain the Login Module (LM) located in org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLoginModule
2. Open the LM's configuration file.
3. Enter the following code, inserting the names of the services you wish to use:

```
<application-policy name="saml-issue-token">
     <authentication>
         <login-module
code="org.picketlink.identity.federation.core.wstrust.auth.STSIssuin
gLoginModule" flag="required">
             <module-option name="configFile">picketlink-sts-
client.properties</module-option>
             <module-option
name="endpointURI">http://security_saml/goodbyeworld</module-option>
         </login-module>
         <login-module
code="org.picketlink.identity.federation.core.wstrust.auth.STSValida
tingLoginModule" flag="required">
             <module-option name="configFile">picketlink-sts-
client.properties</module-option>
         </login-module>
     </authentication>
 </application-policy>
```

This configuration uses a stacked LM. The security token from the first LM is later used by the second LM which will validate the security token. Having two separate LMs for this can be useful as there can be situations where you only need to validate a security token.

4. Specify the *picketlink-sts-client* properties:

```
serviceName=PicketLinkSTS
portName=PicketLinkSTSPort
endpointAddress=http://localhost:8080/picketlink-sts/PicketLinkSTS
```

```
username=admin
password=admin
```

NOTE

The username and password in this file are only used by the *STSValidatingLoginModule*. The username and password may also be stacked or provided by a callback.

5. To use this LM in JBossESB you need to update your server's *login-config.xml* with the above application-policy. You must also point the ESB service to where you want this LM to be used.

   For example, this is how you could configure it in *jboss-esb.xml*:

```
<service category="SamlSecurityQuickstart" name="issueTokenService"
invmScope="GLOBAL"
    description="This service demonstrates how a service can be
configured to issue and validate a security token">



    <security moduleName="saml-issue-token"
callbackHandler="org.jboss.soa.esb.services.security.auth.login.JBos
sSTSIssueCallbackHandler">
        <!-- disable the security context timeout so that our
security context is re-evaluated -->
        <property
name="org.jboss.soa.esb.services.security.contextTimeout"
value="0"/>
    </security>
    ...
</service>
```

   The callbackHandler that is is specified is specific to the ESB. This is because it requires access to the authentication request in the ESB for retreiving the username and password of the user for whom a security token should be issued.

Report a bug

16.3. VALIDATING A SAML SECURITY TOKEN

Procedure 16.2. Task

1. Open the Login Module (LM) from *org.picketlink.identity.federation.core.wstrust.auth.STSIssuingLoginModule*.

2. Configure the properties file as shown in the example below:

```
        <application-policy name="saml-validate-token">
    <authentication>
        <login-module
code="org.picketlink.identity.federation.core.wstrust.auth.STSValida
```

```
tingLoginModule" flag="required">
            <module-option name="configFile">picketlink-sts-
client.properties</module-option>
        </login-module>
    </authentication>
</application-policy>



And in jboss-esb.xml:
<service category="SamlSecurityQuickstart" name="securedSamlService"
invmScope="GLOBAL"
    description="This service demonstrates that an ESB service can
be configured to only validate a security token.">

    <security moduleName="saml-validate-token"
callbackHandler="org.jboss.soa.esb.services.security.auth.login.JBos
sSTSTokenCallbackHandler">
        <!-- disable the security context timeout so that our
security context is re-evaluated -->
        <property
name="org.jboss.soa.esb.services.security.contextTimeout"
value="0"/>
    </security>
    ...
</service>
```

NOTE

The callbackHandler that is specified is specific to the ESB. This is because it requires access to the authentication request in the ESB for retreiving the SAML Token which is to be validated.

NOTE

An example of SAML support in JBossESB can be found in the security_saml quickstart. More information about the Login Modules provied by PicketLink can be found at http://www.jboss.org/community/wiki/STSLoginModules

Report a bug

16.4. PICKETLINK

The *PicketLink* IDM is a framework that links security systems together. It does so by using identity management. It contains information pertaining to user identification, groups, and permissions.

Report a bug

16.5. INTEGRATION BETWEEN SAML AND PICKETLINK

- The client must first obtain the SAML assertion from PicketLink STS by sending a WS-Trust request to the token service. This process usually involves authentication of the client.
- After obtaining the SAML assertion from the STS, the client includes the assertion in the security context of the EJB request before invoking an operation on the bean.
- Upon receiving the invocation, the EJB container extracts the assertion and validates it by sending a WS-Trust message to the STS. If the assertion is deemed valid by the STS (and the proof of possession token has been verified if needed), the client is authenticated.
- In JBoss, the SAML assertion validation process is handled by the SAML2STSLoginModule. It reads properties from a configurable file (specified by the configFile option) and establishes communication with the STS based on these properties.
- If the assertion is valid, a Principal is created using the assertion subject name. If the assertion contains roles, these roles are also extracted and associated with the caller's Subject.

Report a bug

16.6. SECURING YOUR JBOSS ENTERPRISE SOA PLATFORM INSTALLATION

Introduction

The JBoss Enterprise SOA Platform can be made secure, in the sense that you can configure the product so that services will only be executed if caller authentication succeeds and said caller possesses the correct permissions. The default security implementation is based on JAAS.

There are two ways in which to invoke a service:

1. through a gateway
2. directly via the ServiceInvoker.

When you use the gateway option, it is made responsible for obtaining the security information needed to authenticate the caller. It does this by extracting the needed information from the transport. Once it has done so, it creates an authentication request that is encrypted and passed to the Enterprise Service Bus.

If you use the ServiceInvoker instead, it becomes the client application's responsibility to make the authentication request prior to invoking the service. This entails extracting either the **`UsernameToken`** or the **`BinarySecurityToken`** from the SOAP header's security element.

Report a bug

16.7. JAVA AUTHENTICATION AND AUTHORIZATION SERVICE (JAAS)

The JAAS 1.0 API consists of a set of Java packages designed for user authentication and authorization. The API implements a Java version of the standard Pluggable Authentication Modules (PAM) framework and extends the Java 2 Platform access control architecture to support user-based authorization.

JAAS was first released as an extension package for JDK 1.3 and is bundled with JDK 1.6.

Report a bug

16.8. JAASSECURITYSERVICE

JaasSecurityService is the default implementation of JAAS used in the JBoss Enterprise SOA Platform.

Report a bug

16.9. SECURE YOUR SYSTEM

Procedure 16.3. Task

Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.

1. Scroll down to the section that contains properties name="security" and edit the settings to suit your system:

```
<properties name="security">
<property
name="org.jboss.soa.esb.services.security.implementationClass"
value="org.jboss.internal.soa.esb.services.security.JaasSecurityServ
ice"/>

<property name="org.jboss.soa.esb.services.security.callbackHandler"
value=
"org.jboss.internal.soa.esb.services.security.UserPassCallbackHandle
r"/>

<property name="org.jboss.soa.esb.services.security.sealAlgorithm"
value="TripleDES"/>

<property name="org.jboss.soa.esb.services.security.sealKeySize"
value="168"/>

<property name="org.jboss.soa.esb.services.security.contextTimeout"
value="30000"/>

<property name=
"org.jboss.soa.esb.services.security.contextPropagatorImplemtationCl
ass"
value=
"org.jboss.internal.soa.esb.services.security.JBossASContextPropagat
or"/>

<property name="org.jboss.soa.esb.services.security.publicKeystore"
value="/publicKeyStore"/>


<property
name="org.jboss.soa.esb.services.security.publicKeystorePassword"
value="testKeystorePassword"/>
```

```
<property name="org.jboss.soa.esb.services.security.publicKeyAlias"
value="testAlias"/>

<property
name="org.jboss.soa.esb.services.security.publicKeyPassword"
value="testPassword"/>

<property
name="org.jboss.soa.esb.services.security.publicKeyTransformation"
value="RSA/ECB/PKCS1Padding"/>

</properties>
```

2. Save the file and exit.
3. Open the log-in configuration file in your text editor: **`vi SOA_ROOT/server/PROFILE/conf/login-config.xml`**
4. Configure the JAAS log-in modules by editing the settings in this file. (You can use either a pre-configured option or create your own custom solution.)
5. Save the file and exit.

Report a bug

16.10. CREATE AN ENCRYPTED PASSWORD FILE

Procedure 16.4. Task

1. Go to the **`conf`** directory: **`cd SOA_ROOT/jboss-as/server/PROFILE/conf`**
2. Execute this command: **`java -cp ../../../lib/jbosssx.jar org.jboss.security.plugins.FilePassword welcometojboss 13 testpass esb.password`**

Result

An encrypted password file is created.

Report a bug

16.11. ENCRYPTION OPTIONS

Table 16.1. Encryption Options

Option	Description
Salt	This is the "salt" used to encrypt the password file. (In the example above, it is the **`welcometojboss`** string .)

Option	Description
Iteration	This is the number of iterations. (In the example above, it is the number **13**.)
Password File Name	This is the name of the file where the encrypted password will be saved. In the example above, it is the **esb.password** string.
testpass	This is the test password.

Report a bug

16.12. CLEAR-TEXT PASSWORD

A clear-text password is the plain text version of a password. It has either not been encrypted or has just been decrypted. Clear text passwords are unsecure.

Report a bug

16.13. PASSWORD MASK

A password mask is a template which determines what characters are allowed to be used in a password. For example, some password masks dictate that a password can only be alphanumeric while others allow special characters like ! and $ signs. Passwords which contain special characters are generally viewed as being more secure.

Report a bug

16.14. MASKING PASSWORDS

Introduction

Passwords are secret authentication tokens that are used to limit access to resources to authorized parties only. For a JBoss services to access password-protected resources, the password must obviously be made available to it.

This can be done by means of command line arguments passed to the JBoss Enterprise SOA Platform on launch, however this is not practical in a production environment. Instead, passwords are normally made available to JBoss services through their inclusion in configuration files.

All JBoss Enterprise SOA Platform configuration files should be stored on secure file systems, and be made readable by the process owner only.

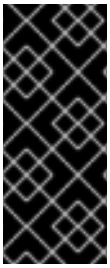
For an added level of security, you can also mask the password in the configuration file. This section will tell you how to do so.

IMPORTANT

There is no such thing as impenetrable security. Masking passwords is no exception - it is not impenetrable, but it does defeat casual inspection of configuration files, and increases the amount of effort that will be required to extract the password.

Report a bug

16.15. MASK A CLEAR-TEXT PASSWORD



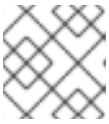
IMPORTANT

You should only perform this key store password encryption procedure once. If you make a mistake entering the keystore password, or you change the key store at a later date, you should delete the **`jboss-keystore_pass.dat`** file and repeat the procedure. Be aware that if you change the key store any masked passwords that were previously generated will no longer function.

Procedure 16.5. Task

1. Generate a key pair using this command: **`keytool -genkey -alias jboss -keyalg RSA -keysize 1024 -keystore password.keystore`** and follow the prompts:

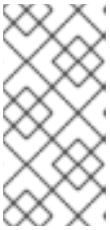
```
        keytool -genkey -alias jboss -keyalg RSA -keysize 1024 -
keystore password.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Bob Bobson
What is the name of your organizational unit?
  [Unknown]:  Corporate_IT
What is the name of your organization?
  [Unknown]:  XYZ
What is the name of your City or Locality?
  [Unknown]:  BRISBANE
What is the name of your State or Province?
  [Unknown]:  QLD
What is the two-letter country code for this unit?
  [Unknown]:  AU
Is CN=Bob Bobson, OU=Corporate_IT, O=XYZ, L=BRISBANE, ST=QLD, C=AU
correct?
  [no]:  yes

Enter key password for jboss
 (RETURN if same as keystore password):
```



NOTE

You must specify the same password for the key store and key pair.

2. Run **chown** to change ownership to the JBoss Application Server process owner, and **chmod 600 password.keystore** to make sure only the file's owner can read it.



NOTE

The process owner should not have console log-in access. In that case you will be performing these operations as another user. Creating masked passwords requires read access to the key store, so you may wish to complete configuration of masked passwords before restricting the key store file permissions.

3. Navigate to the **jboss-as/bin** directory: **cd SOA_ROOT/jboss-as/bin**
4. Run the password tool, using the command **./password_tool.sh** on Red Hat Enterprise Linux systems, (or **password_tool.bat** on Microsoft Windows-based systems.)
5. Select **0: Encrypt Keystore Password** by pressing 0, then Enter.
6. Enter the key store password you specified above.
7. Enter a random string of characters to aid with encryption strength. This is the salt.
8. Enter a whole number for the iterator count to aid with encryption strength.
9. Select **5: Exit** to exit.

NOTE

The password tool will exit with the message: **Keystore is null. Cannot store.** This is normal.

10. Use the **chown** command to change ownership of the **password/jboss_keystore_pass.dat** file to the process owner, and **chmod 600 jboss-keystore_pass.dat** to ensure that only that owner can read the file.
11. Navigate to the **jboss-as/bin** directory: **cd SOA_ROOT/jboss-as/bin**
12. Run the password tool, using the command **./password_tool.sh** on Red Hat Enterprise Linux systems (or **password_tool.bat** on Microsoft Windows systems).
13. Select **1: Specify KeyStore** by pressing 1 then Enter.
14. Enter the path to the key store you created above. (You can specify an absolute path, or the path relative to **SOA_ROOT/jboss-as/bin**. This should be **SOA_ROOT/jboss-as/bin/password.keystore**, unless you have changed the defaults.)
15. Enter the key alias. This should be "jboss" (unless you have performed an advanced installation and changed the defaults).
16. Select **2: Create Password** by pressing 2, then Enter. You will be prompted for the security domain. Follow the prompts on screen.

```
        /password_tool.sh
*********************************
****  JBoss Password Tool********
```

```
*********************************
Error while trying to load data:Encrypted password file not located
Maybe it does not exist and need to be created.
0: Encrypt Keystore Password 1:Specify KeyStore 2:Create Password
3: Remove a domain 4:Enquire Domain 5:Exit
1
Enter Keystore location including the file name
password.keystore
Enter Keystore alias
jboss
0: Encrypt Keystore Password 1:Specify KeyStore 2:Create Password
3: Remove a domain 4:Enquire Domain 5:Exit
2
Enter security domain:

default
Enter passwd:
passwordmask
Password created for domain:default
0: Encrypt Keystore Password 1:Specify KeyStore 2:Create Password
3: Remove a domain 4:Enquire Domain 5:Exit
```

17. Enter a name for the password mask. This is an arbitrary unique name that you will use to identify the password mask in configuration files.
18. Enter the password that you wish to mask.
19. Repeat the password mask creation process to create masks for all passwords you wish to mask.
20. Exit the program by choosing **`5: Exit`**
21. Navigate to the **`password`** directory: **`cd SOA_ROOT/jboss-as/bin/password`**

Report a bug

16.16. REPLACE A CLEAR TEXT PASSWORD WITH ITS PASSWORD MASK

Prerequisites

- Pre-existing password masks

Procedure 16.6. Task

- Launch a text editor and replace each occurrence of a clear text password in the configuration files with an annotation referencing its mask.

  This is the general form of the annotation:

```
<annotation>
@org.jboss.security.integration.password.Password(securityDomain=MAS
K_NAME, methodName=setPROPERTY_NAME)
```

```
</annotation>
```

As a concrete example, the JBoss Messaging password is stored in the server profile's **deploy/messaging/messaging-jboss-beans.xml** file. If you create a password mask named "messaging", then the before and after snippet of the configuration file will looks like this:

```
<property name="suckerPassword">
CHANGE ME!!
</property>
```

```
<annotation>
@org.jboss.security.integration.password.Password(securityDomain=mes
saging,
methodName=setSuckerPassword)
</annotation>
```

Report a bug

16.17. CHANGE THE DEFAULT PASSWORD MASK SETTINGS

By default the server profiles are configured to use the keystore **SOA_ROOT/jboss-as/bin/password/password.keystore**, and the key alias "jboss". If you store the key pair used for password masking elsewhere, or under a different alias, you will need to update the server profiles with the new location or key alias.

Procedure 16.7. Task

1. Open the security configuration file in a text editor: **vi SOA_ROOT/jboss-as/server/PROFILE/deploy/security/security-jboss-beans.xml**.
2. Edit the key store location and key alias. Here are some example settings:

```
<!-- Password Mask Management Bean-->
   <bean name="JBossSecurityPasswordMaskManagement"

class="org.jboss.security.integration.password.PasswordMaskManagemen
t" >
         <property
name="keyStoreLocation">password/password.keystore</property>
         <property name="keyStoreAlias">jboss</property>
         <property
name="passwordEncryptedFileName">password/jboss_password_enc.dat</pr
operty>
         <property
name="keyStorePasswordEncryptedFileName">password/jboss_keystore_pas
s.dat</property>
   </bean>
```

3. Save the file and exit.

Report a bug

16.18. GLOBAL CONFIGURATION FILE SECURITY SETTINGS

Table 16.2. `jbossesb-properties.xml` Security Settings

Property	Description	Required?
`org.jboss.soa.esb.services.security.implementationClass`	This is the "concrete"*SecurityService* implementation that should be used. The default setting is **`JaasSecurityService`**.	Yes
`org.jboss.soa.esb.services.security.callbackHandler`	This is a default **`CallbackHandler`** implementation, utilized when a JAAS-based **`SecurityService`** is employed. See “Customizing Security” for more information about the CallbackHandler property.	No
`org.jboss.soa.esb.services.security.sealAlgorithm`	This is the algorithm to use when "sealing" the **`SecurityContext`**.	No
`org.jboss.soa.esb.services.security.sealKeySize`	This is the size of the secret/symmetric key used to encrypt/decrypt the **`SecurityContext`**.	No
`org.jboss.soa.esb.services.security.contextTimeout`	This is the amount of time (in milliseconds) for which a security context is valid. A global setting, this may be over-ridden on a per-service basis. To do so, specify the property of the same name that exists on the security element in the **`jboss-esb.xml`** file.	No
`org.jboss.soa.esb.services.security.contextPropagatorImplementationClass`	Use this to configure a global **`SecurityContextPropagator`**. (For more details on the **`SecurityContextPropagator`**, please refer to the section on “Advanced Security Options”.)	No
`org.jboss.soa.esb.services.security.publicKeystore`	This is the *Keystore* which holds the keys used to encrypt and decrypt that data which is external to the Enterprise Service Bus. The Keystore is used to encrypt the **`AuthenticationRequest`**.	No
`org.jboss.soa.esb.services.security.publicKeystorePassword`	This is the password for the public keystore.	No

Property	Description	Required?
`org.jboss.soa.esb.services.security.publicKeyAlias`	This is the alias to use for the public key.	No
`org.jboss.soa.esb.services.security.publicKeyPassword`	This is the password for the alias if one was specified upon creation.	No
`org.jboss.soa.esb.services.security.publicKeyPassword`	This is a cipher transformation. It is in this format: **`algorithm/mode/padding`**. If this is not specified, the "keys" algorithm will be used by default.	No

Report a bug

16.19. KEY PAIR

A key pair is a set of security tools consisting of a public key and a private key. The public key is used for encryption and the private key is used for decryption.

Report a bug

16.20. KEYSTORE

A keystore is a security mechanism. It contains a number of security certificates and their assigned "keys". It is used when client authentication is required.

The JBoss Enterprise SOA Platform ships with an example key-store, found in **`SOA_ROOT/jboss-as/samples/quickstarts/security_cert/keystore`**. Do not use this in a production environment. It is provided as an example only.

Report a bug

16.21. JBOSS RULES

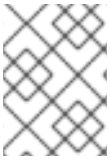
JBoss Rules is the name of the business rule engine provided as part of the JBoss Enterprise SOA Platform product.

Report a bug

16.22. CONTENT BASED ROUTING USING THE JBOSS RULES ENGINE

JBoss Rules is the rule provider "engine" for the content-based router. The Enterprise Service Bus integrates with this engine through three different routing **`action classes`**, these being:

- a routing rule set, written in the **`JBoss Rules`** engine's DRL language (alternatively, you can use the DSL language if you prefer it);
- the message content. This is the data that goes into the JBoss Rules engine (it comes in either XML format or as objects embedded in the message);
- the destination (which is derived from the resultant information coming out of the engine).



NOTE

When a message is sent to the **`content-based router`**, a rule-set will evaluate its content and return a set of service destinations.

- **org.jboss.soa.esb.actions.ContentBasedRouter**: This action class implements the *content-based routing* pattern. It routes a message to one or more destination services, based on the message content and the rule set against which it is evaluating that content. The content-based router throws an exception when no destinations are matched for a given rule set or message combination. This action will terminate any further pipeline processing, so always position it last in your pipeline.
- **org.jboss.soa.esb.actions.ContentBasedWiretap**: This implements the *WireTap* pattern. The **`WireTap`** is an enterprise integration pattern that sends a copy of the message to a control channel. The **`WireTap`** is identical in functionality to the standard content-based router, however it does not terminate the pipeline. It is this latter characteristic which makes it suitable to be used as a wire-tap, hence its name. For more information, see http://www.eaipatterns.com/WireTap.html.
- **org.jboss.soa.esb.actions.MessageFilter**: This implements the *message filter* pattern. The message filter pattern is used in cases where messages can simply be dropped if certain content requirements are not met. In other words, it functions identically to the content-based router except that it does not throw an exception if the rule set does not match any destinations, it simply filters the message out. For more information, see http://www.eaipatterns.com/Filter.html.

Report a bug

16.23. RULE BASE

Rule bases are collections of rules. They are used in processing events. The rules help determine how information is stored and processed, which actions are allowed and what action to take when a message is being sent.

Report a bug

16.24. SERIALIZE

To serialize an object is to convert it to a data object.

Report a bug

16.25. DESERIALIZE

To deserialize a file is to transform it back into an object. It is the opposite of serialization.

Report a bug

16.26. JBOSS RULES AND SECURITY

By default, the JBoss Rules component does not deserialize rules packages or unsigned rule bases.

IMPORTANT

You must activate this serialization security feature in order for your configuration to be supported by Red Hat. You need to configure system properties for both the application that serializes the packages and its rule bases (hereafter referred to as the server), as well as the application that deserializes the packages its rule bases (hereafter referred to as the client).

Report a bug

16.27. ENABLE SERIALIZATION ON THE SERVER

Procedure 16.8. Task

1. Navigate to the SOA_ROOT directory: **`cd SOA_ROOT`**.
2. Run the **`keytool`** command and follow the prompts on screen:

```
keytool -genkey -alias droolsKey -keyalg RSA -keystore
MyDroolsPrivateKeyStore.keystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Test User
What is the name of your organizational unit?
  [Unknown]:  HR
What is the name of your organization?
  [Unknown]:  Test Org
What is the name of your City or Locality?
  [Unknown]:  Brisbane
What is the name of your State or Province?
  [Unknown]:  QLD
What is the two-letter country code for this unit?
  [Unknown]:  AU
Is CN=Test User, OU=HR, O=Test Org, L=Brisbane, ST=QLD, C=AU
correct?
  [no]:  yes
Enter key password for droolsKey
     (RETURN if same as keystore password):
Re-enter new password:
```

After answering all of the questions, a password-protected file named **MyDroolsPrivateKeyStore.keystore** is created. This keystore file has a private key called droolsKey with the password "drools". Store this file in a safe location in your environment, which will hereafter be referred to as the **keystoredir**.

IMPORTANT

The passwords above are examples only and should not be used in production.

3. Open the configuration file: **vi jboss-as/server/default/deploy/properties-service.xml**

4. Configure the JBoss Enterprise SOA Platform to use the JBoss Rules serialization feature by adding this snippet to **properties-service.xml**:

```
<mbean code="org.jboss.varia.property.SystemPropertiesService"
name="jboss:type=Service,name=SystemProperties">
    <attribute name="Properties">
      # Drools Security Serialization specific properties
      drools.serialization.sign=true

drools.serialization.private.keyStoreURL=file://$keystoredir/MyDrool
sPrivateKeyStore.keystore
      drools.serialization.private.keyStorePwd=drools
      drools.serialization.private.keyAlias=droolsKey
      drools.serialization.private.keyPwd=drools
    </attribute>
</mbean>
```

5. Set the drools.serialization.sign property to "true":

```
drools.serialization.sign=true
```

   - drools.serialization.private.keyStoreURL=<RL> is the URL of the private keystore location.
   - In the example above, replace **keystoredir** and **MyDroolsKeyStore.keystore** with your keystore directory and the name of the keystore you created with the keytool
   - drools.serialization.private.keyStorePwd=<password> is the password to access the private keystore.
   - drools.serialization.private.keyAlias=<key> is the key alias (identifier) of the private key.
   - drools.serialization.private.keyPwd=<password> is the private key password.

6. Save the file and exit.

7. Restart the server instance.

WARNING

If the system properties were not configured properly, you will see this error when you try to build a rules package:

```
An error occurred building the package.

Error
signing object store: Key store with private key not
configured. Please
configure it properly before using signed
serialization
```

Report a bug

16.28. ENABLE SERIALIZATION ON THE CLIENT

Prerequisites

- Server serialization must already be enabled.

Procedure 16.9. Task

1. Create a public key certificate from the private keystore. (You can access the keytool by running **`keytool -genkey -alias droolsKey -keyalg RSA -keystore`**.):

```
keytool -export -alias droolsKey -file droolsKey.crt -keystore
```

```
MyDroolsPrivateKeyStore.keystore
Enter keystore password:
Certificate stored in file <droolsKey.crtU>
```

2. Import the public key certificate into a public keystore. (This is where it will be used by your client applications):

```
keytool -import -alias droolsKey -file droolsKey.crt -keystore
```

```
MyPublicDroolsKeyStore.keystore
Enter keystore password:
Re-enter new password:
Owner: CN=Test User, OU=Dev, O=XYZ Corporation, L=Brisbane, ST=QLD,
C=AU
Issuer: CN=Test User, OU=Dev, O=XYZ Corporation, L=Brisbane, ST=QLD,
C=AU
Serial number: 4ca0021b
Valid from: Sun Sep 26 22:31:55 EDT 2010 until: Sat Dec 25 21:31:55
```

```
EST 2010
Certificate fingerprints:
     MD5:  31:1D:1B:98:59:CC:0E:3C:3F:57:01:C2:FE:F2:6D:C9
     SHA1:
4C:26:52:CA:0A:92:CC:7A:86:04:50:53:80:94:2A:4F:82:6F:53:AD
     Signature algorithm name: SHA1withRSA
     Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

3. Open the server configuration file: **`vi grep drools jboss-as/server/default/deploy/properties-service.xml`**

4. Replace keystoredir and MyPublicDroolsKeyStore.keystore with your keystore directory, and the name of the public keystore you created previously:

```
# Drools Client Properties for Security Serialization
drools.serialization.public.keyStoreURL=file://$keystoredir/MyPublic
DroolsKeyStore.keystore
drools.serialization.public.keyStorePwd=drools
```

5. Save the file and exit.

6. Restart the JBoss Enterprise SOA Platform server.

7. For Java client applications, set the system properties in your code like this:

```
// Set the client properties to deserialize the signed packages
URL clientKeyStoreURL = getClass().getResource(
"MyPublicDroolsKeyStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_SIGN,
                            "true" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
                            clientKeyStoreURL.toExternalForm() );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_PWD,
                            "drools" );
...
```

Alternatively, open the **`run.sh`** shell script (**`vi SOA_ROOT/jboss-as/bin/run.sh`**) script and edit the JAVA_OPTS section:

```
# Serialization Security Settings
JAVA_OPTS="-Ddrools.serialization.sign=true $JAVA_OPTS"
JAVA_OPTS="-
Ddrools.serialization.private.keyStoreURL=file://$keystoredir/MyDroo
lsKeyStore.keystore $JAVA_OPTS"
JAVA_OPTS="-Ddrools.serialization.private.keyStorePwd=drools
$JAVA_OPTS"
JAVA_OPTS="-Ddrools.serialization.private.keyAlias=droolsKey
$JAVA_OPTS"
JAVA_OPTS="-Ddrools.serialization.private.keyPwd=drools $JAVA_OPTS"
JAVA_OPTS="-
Ddrools.serialization.public.keyStoreURL=file://$keystoredir/MyPubli
```

```
cDroolsKeyStore.keystore $JAVA_OPTS"
JAVA_OPTS="-Ddrools.serialization.public.keyStorePwd=drools
$JAVA_OPTS"
```

Replace the values shown above with ones specific to your environment, and then restart the server instance.

Report a bug

16.29. DISABLE SERIALIZATION SIGNING

1. Open the configuration file: **`vi SOA_ROOT/jboss-as/server/PROFILE/deploy/properties-service.xml`**.
2. Remove the drools.serialization.sign property's value.
3. Save the file and exit.

   An alternative way to do this task is to open the **`run.sh`** shell script (**`vi SOA_ROOT/jboss-as/bin/run.sh`**) and edit it as follows:

   ```
   JAVA_OPTS="-Ddrools.serialization.sign=false $JAVA_OPTS"
   ```

4. Restart the server instance.
5. To turn signing off for Java client applications, remove the drools.serialization.sign property or add the following snippet to each application's code:

   ```
   System.setProperty( KeyStoreHelper.PROP_SIGN, "false" );
   ```

Report a bug

16.30. CONFIGURE SECURITY ON A PER-SERVICE BASIS

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jboss-esb.xml`**.
2. Scroll down to the service you want to configure.
3. Add a security element. This setting shows you how to do so:

   ```
   <service category="Security" name="SimpleListenerSecured">
      <security moduleName="messaging" runAs="adminRole"
       rolesAllowed="adminRole, normalUsers"

   callbackHandler="org.jboss.internal.soa.esb.services.security.UserPa
   ssCallbackHandler">
        <property name="property1" value="value1"/>
        <property name="property2" value="value2"/>
   ```

```
    </security>
    ...
</service>
```

4. Save the file and exit.

Report a bug

16.31. PER-SERVICE SECURITY PROPERTIES

Table 16.3. Security Properties

Property	Description	Required?
moduleName	This is a module that exists in the **`SOA_ROOT/jboss-as/server/PROFILE/conf/login-config.xml`** file.	No
runAs	This is the runAs role.	No
rolesAllowed	This is an comma-separated list of those roles that have been granted the ability to execute the service. This is used as a check that is performed after a caller has been authenticated, in order to verify that they are indeed belonging to one of the roles specified. The roles will have been assigned after a successful authentication by the underlying security mechanism.	No
callbackHandler	This is the **`CallbackHandler`** that will override that which was defined in the **`jbossesb-properties.xml`** file.	No
property	These are optional properties that, once defined, will be made available to the **`CallbackHandler`** implementation.	No

Report a bug

16.32. OVERRIDE GLOBAL SECURITY SETTINGS

Procedure 16.10. Task

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.

2. Configure the setting in question. Here is an example:

```
<security moduleName="messaging"
  runAs="adminRole" rolesAllowed="adminRole">

<property
    name="org.jboss.soa.esb.services.security.contextTimeout"
    value="50000"/>

<property name=
"org.jboss.soa.esb.services.security.contextPropagatorImplementation
Class"
    value="org.xyz.CustomSecurityContextPropagator" />

</security>
```

3. Save the file and exit.

Report a bug

16.33. SECURITY PROPERTY OVERRIDES

Table 16.4. Security Property Overrides:

Property	Description	Required?
org.jboss.soa.esb.services.security.contextTimeout	This property lets the service override the global security context time-out (milliseconds) that is specified in the **`jbossesb-properties.xml`** file.	No
org.jboss.soa.esb.services.security.contextPropagatorImplementationClass	This property lets the service to override the "global security context propagator" class implementation, that is specified in the **`jbossesb-properties.xml`** file.	No

Report a bug

16.34. SECURITY CONTEXT

The SecurityContext is an object which is created after a security certificate is confirmed. After creation, it will be configured so that you do not have to re-authenticate the certificate every time you perform an action related to it. If the ESB detects that a message has a SecurityContext, it will check that it is still

valid and, if so, it does not try to re-authenticate it. (Note that the SecurityContext is only valid for a single Enterprise Service Bus node. If the message is routed to a different ESB node, it will have to be re-authenticated.)

Report a bug

16.35. AUTHENTICATION REQUEST

An AuthenticationRequest carries the security information needed for authentication between either a gateway and a service or between two services. You must set an instance of this class on the message object prior to the authenticating service being called. The class must contain the principle and the credentials needed to authenticate a caller. This class is made available to the Callback Handler.

Report a bug

16.36. SECURITYCONFIG

The **`SecurityConfig`** class grants access to the security configuration specified in the **`jboss-esb.xml`** file. This class is made available to the Callback Handler.

Report a bug

16.37. ADD AN AUTHENTICATION CLASS TO A MESSAGE OBJECT

Procedure 16.11. Task

- Execute this code:

```
byte[] encrypted = PublicCryptoUtil.INSTANCE.encrypt((Serializable)
 authRequest);
message.getContext.setContext(SecurityService.AUTH_REQUEST,
encrypted);
```

Result

The authentication context is encrypted and then set within the message context. (It is later decrypted by the Enterprise Service Bus so that it can authenticate the request.)

Report a bug

16.38. SECURITY_BASIC QUICK START

The **`SOA_ROOT/jboss-as/samples/quickstarts/security_basic`** quick start demonstrates how to prepare the security on a message before you use the SecurityInvoker. The quick start also demonstrates how to configure the **`jbossesb-properties.xml`** global configuration file for use by client services.

Report a bug

16.39. SET A TIME LIMIT FOR THE SECURITY CONTEXT GLOBALLY

Procedure 16.12. Task

1. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.
2. Scroll down to the section that contains security.contextTimeout. Set the time-out value (in milliseconds).
3. Save the file and exit.

Report a bug

16.40. SET A TIME LIMIT FOR THE SECURITY CONTEXT ON A PER-SERVICE BASIS

Procedure 16.13. Task

1. Open the service's configuration file in a text editor: **`vi jboss-esb.xml`**.
2. Scroll down to the section that contains Security Context. Set the time-out value (in milliseconds).
3. Save the file and exit.

Report a bug

CHAPTER 17. ADVANCED SECURITY OPTIONS

17.1. SECURITY PROPAGATION

The term "security propagation" refers to the process of passing security information to an external system. For example, you might want to use the same credentials to call both the Enterprise Service Bus and an Enterprise Java Beans method.

Report a bug

17.2. SECURITYCONTEXTPROPAGATOR

The SecurityContextPropagator class passes the security context to the destination environment.

Report a bug

17.3. SECURITYCONTEXTPROPAGATOR IMPLEMENTATIONS

Table 17.1. Implementations of SecurityContextPropagator

Class	Description
Package: org.jboss.internal.soa.esb.services.security Class: JBossASContextPropagator	This propagator will send security credentials to the ESB. If you need to write your own implementation you only have to write a class that implements **`org.jboss.internal.soa.esb.services.security.SecurityContextPropagator`** and then either specify that implementation in **`jbossesb-properties.xml`** or **`jboss-esb.xml`**.

Report a bug

17.4. ADD A CUSTOM LOG-IN MODULE

Procedure 17.1. Task

1. Open the log-in configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/conf/login-config.xml`**
2. Add the details of your custom log-in module.
3. Save the file and exit.
4. Since different log-in modules require different information, you must specify the CallbackHandler attribute to be used. Open the specific security configuration for that service.
5. Make sure that the **`CallbackHandler`** specifies a *fully-qualified classname* for the class which implements the **`EsbCallbackHandler`** interface. This code shows you how to do so:

```
public interface EsbCallbackHandler extends CallbackHandler
{
  void setAuthenticationRequest(final AuthenticationRequest
authRequest);
  void setSecurityConfig(final SecurityConfig config);
}
```

6. Add both the "principle" and the credentials needed to authenticate a caller to the **`AuthenticationRequest`** class.

Result

JaasSecurityService is replaced with your custom security implementation.

Report a bug

17.5. CERTIFICATE LOG-IN MODULE

The Certificate Log-in Module performs authentication by verifying the certificate that is passed with the call to the Enterprise Service Bus against a certificate held in a local key-store. The certificate's common name creates a "principle".

Report a bug

17.6. CERTIFICATE LOG-IN MODULE PROPERTIES

```
<security moduleName="CertLogin" rolesAllowed="worker"
  callbackHandler="org.jboss.soa.esb.services.security.auth.loginUserPass
CallbackHandler">
  <property name="alias" value="certtest"/>
</security>
```

Table 17.2. Properties

Property	Description
moduleName	This identifies the JAAS Login module to use. This module will be specified in JBossAS login-config.xml.
rolesAllow	This is a comma-separated list of the roles that are allowed to execute this service.
alias	This is the alias which is used to look up the local key-store and which will be used to verify the caller's certificate.

Report a bug

17.7. CERTIFICATE LOG-IN MODULE CONFIGURATION FILE PROPERTIES

```
<application-policy name="CertLogin">
<authentication>
  <login-module
code="org.jboss.soa.esb.services.security.auth.login.CertificateLoginModul
e"
flag = "required" >
  <module-option name="keyStoreURL">
    file://pathToKeyStore
  </module-option>
  <module-option name="keyStorePassword">storepassword</module-option>
  <module-option name="rolesPropertiesFile">
    file://pathToRolesFile
  </module-option>
  </login-module>
</authentication>
</application-policy>
```

Table 17.3. Certificate Log-In Module Configuration File Properties

Property	Description
keyStoreURL	This is the path to the key-store used to verify the certificates. It can be a file on the local file system or on the class-path.
keyStorePassword	This is the password for the key-store above.
rolesPropertiesFile	This is optional. It is the path to a file containing role mappings. Refer to the “Role Mapping” section of the Getting Started Guide for more details about this.

Report a bug

17.8. CALLBACK HANDLER

A callback handler is a type of library used in back-end operations. It allows applications to "talk" to each other through security services and can be used to confirm authentication data.

Report a bug

17.9. ROLE MAPPING

Role mapping is a way of sharing data between secure hosts. A file containing a list of trusted hosts is created, with each host assigned several role mappings. Mapping occurs when you accesses data from one of the hosts. The sender's roles are mapped onto the receiver's roles to allow for authentication and data sharing. Types of roles include user roles, application roles and so forth. This is an optional feature and is not enabled by default.

Report a bug

17.10. ENABLE ROLE MAPPING

Procedure 17.2. Task

1. Open the log-in configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/conf/login-config.xml`**
2. Set the rolesPropertiesFile property. (This property can point to a file located on either the local file system or the class-path).
3. Map users to roles. This example code shows how to do so:

```
# user=role1,role2,...
guest=guest
esbuser=esbrole
# The current implementation will use the Common Name(CN) specified
# for the certificate as the user name.
# The unicode escape is needed only if your CN contains a space
Andy\u0020Anderson=esbrole,worker
```

4. Save the file and exit.

Report a bug

17.11. SECURITY_CERT QUICKSTART

The security_cert quickstart demonstrates the JBoss Enterprise SOA Platform's role-mapping functionality.

Report a bug

17.12. SECURITY SERVICE

The **`SecurityService`** interface is the Enterprise Service Bus' central security component.

Report a bug

17.13. CUSTOMIZE THE SECURITY SERVICE INTERFACE

Procedure 17.3. Task

1. Implement the **`SecurityService`** interface:

```
public interface SecurityService
{
    void configure() throws ConfigurationException;
```

```
    void authenticate(
        final SecurityConfig securityConfig,
        final SecurityContext securityContext,
        final AuthenticationRequest authRequest)
        throws SecurityServiceException;

    boolean checkRolesAllowed(
        final List<String> rolesAllowed,
        final SecurityContext securityContext);

    boolean isCallerInRole(
        final Subject subject,
        final Principle role);

    void logout(final SecurityConfig securityConfig);

    void refreshSecurityConfig();
}
```

2. Open the global configuration file in a text editor: **`vi SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**.

3. Configure the file to use the customized **`SecurityService`**

4. Save the file and exit.

Report a bug

17.14. REMOTE INVOCATION CLASS

As its name implies, a remote invocation class is a class that can be called from a remote machine. This can be useful for developers but can also lead to potential security risks.

Report a bug

17.15. SECURE NON-REMOTE METHOD INVOCATION CLASSES ON PORT 8083

Client applications can, by default, utilize Remote Method Invocation to download Enterprise Java Bean classes through **`port 8083`**. However, you can also configure the system's Remote Method Invocation settings to allow client applications to download any deployed resources you desire.

Procedure 17.4. Task

1. **Edit the Settings in the jboss-service.xml File**
   Open the file in a text editor: **`vi SOA_ROOT/server/PROFILE/conf/jboss-service.xml`**

2. **Configure the Settings in the File**
   Here is an example:

```
<attribute name="DownloadServerClasses">false</attribute>
```

Set this value to false to ensure that client applications can only download Enterprise Java Bean classes.

IMPORTANT

By default, this value is set to false in the SOA Platform's 'production' profile. The value is set to true in all other cases, including the SOA Standalone version's default profile. Note that this is not a secure configuration and should only be used in development environments.

Report a bug

CHAPTER 18. SECURING THE SERVICE REGISTRY

18.1. SERVICE REGISTRY AUTHENTICATION

Introduction

Here is a theoretical understanding of how the authentication process works.

Authentication is a two-phase process. These are known as the *authenticate phase* and the *identify phase*. Both of these phases are represented by a method in the **`Authenticator`** interface.

The authenticate phase occurs when the **`GetAuthToken`** request is made. The goal of this phase is to turn a user id and credentials into a valid publisher id. The publisher id (referred to as the *authorized name* in UDDI terminology) is the value that assigns ownership within UDDI. Whenever a new entity is created, it must be tagged with ownership by the authorized name of the publisher.

The value of the publisher id is irrelevant to the jUDDI Registry: the only requirement is that one exists to assign to new entities so it must be non-null. Upon completion of the **`GetAuthToken`** request, an **`authentication token`** is issued to the caller.

When you make subsequent calls to the UDDI API that require authentication, you must provide the token issued in response to the **`GetAuthToken`** request. This leads to the identify phase.

The identify phase is responsible for turning the authentication token (or the publisher id associated with that token) into a valid **`UddiEntityPublisher`** object. This object contains all the properties necessary to handle ownership of UDDI entities. Thus, the token (or publisher id) is used to identify the publisher.

The two phases provide compliance with the UDDI authentication structure and grant flexibility if you wish to provide your own authentication mechanism.

Handling of credentials and publisher properties could be done entirely outside of the jUDDI Registry. However, by default, the Registry provides the **`Publisher`** entity, which is a sub-class of **`UddiEntityPublisher`**. This sub-class makes publisher properties persist within the jUDDI Registry.

Report a bug

18.2. AUTHTOKEN

An authToken is a security container holding password credentials.

Report a bug

18.3. AUTHTOKEN AND THE SERVICE REGISTRY

In order to enforce proper write access to the Service Registry, each request made to it needs a valid **`authToken`**.

IMPORTANT

Note that read access is not restricted at all.

Report a bug

18.4. OBTAIN AN AUTHTOKEN

Procedure 18.1. Task

1. Make a **`GetAuthToken()`** request.
2. A **`GetAuthToken`** object is returned. Set a userid and credential (password) on this object:

```
org.uddi.api_v3.GetAuthToken ga = new
org.uddi.api_v3.GetAuthToken();
ga.setUserID(pubId);
ga.setCred("");

org.uddi.api_v3.AuthToken token = securityService.getAuthToken(ga);
```

3. Locate the **`juddi.properties`** configuration file in **`SOA_ROOT/jboss-as/server/PROFILE/deploy/juddi-service.sar/juddi.war/WEB-INF`**. Open it in a text editor.
4. Configure the juddi.authenticator property to how the Service Registry will check the credentials passed to it by the **`GetAuthToken`** request. (By default it uses the **`jUDDIAuthenticator`** implementation.)
5. Save the file and exit.

Report a bug

18.5. SECURITY AUTHENTICATION IMPLEMENTATIONS AVAILABLE FOR THE SERVICE REGISTRY

jUDDI Authentication

WARNING

Do not use this authentication method in a production environment. It accepts any credentials provided, and effectively removes the need for clients to authenticate when accessing the registry.

The default authentication mechanism provided by the Service Registry is the **`jUDDIAuthenticator`**. **`jUDDIAuthenticator`**'s authenticate phase checks to see if the, user ID submitted matches against a record in the **`Publisher`** table. No credentials checks are made. If, during the authentication process, the Publisher record is found to be non-existent, it is added "on-the-fly".

In the identify phase, the publisher ID is used to retrieve the Publisher record and return it. The Publisher inherits every property it needs from **`UddiEntityPublisher`**:

```
juddi.authenticator = org.apache.juddi.auth.JUDDIAuthentication
```

XMLDocAuthentication

The authenticate phase checks that the user id and password match a value in the XML file. The identify phase uses the user ID to populate a new **`UddiEntityPublisher`**.

CryptedXMLDocAuthentication

The **`CryptedXMLDocAuthentication`** implementation is similar to the **`XMLDocAuthentication`** implementation, but the passwords are encrypted:

```
juddi.authenticator = org.apache.juddi.auth.CryptedXMLDocAuthentication
juddi.usersfile = juddi-users-encrypted.xml
juddi.cryptor = org.apache.juddi.cryptor.DefaultCryptor
```

Here, the user credential file is **`juddi-users-encrypted.xml`**, and the content of the file will be similar to this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<juddi-users>
<user userid="anou_mana" password="+j/kXkZJftwTFTBH6Cf6IQ=="/>
<user userid="bozo" password="Na2Ait+2aW0="/>
<user userid="sviens" password="+j/kXkZJftwTFTBH6Cf6IQ=="/>
</juddi-users>
```

The **`DefaultCryptor`** implementation uses **`BEWithMD5AndDES`** and **`Base64`** to encrypt the passwords.

NOTE

You can use the code in the **`AuthenticatorTest`** to learn more about how to use this Authenticator implementation. You can plug in your own encryption algorithm by implementing the **`org.apache.juddi.cryptor.Cryptor`** interface and referencing your implementation class in the juddi.cryptor property.

The authenticate phase checks that the user ID and password match values in the XML file. The identify phase uses the user ID to populate a new **`UddiEntityPublisher`**.

LDAP Authentication

Use **`LdapSimpleAuthenticator`** to authenticate users via LDAP's simple authentication functionality. This class allows you to authenticate a user based on an *LDAP principle*, provided that the principle and the jUDDI publisher ID are identical.

JBoss Authentication

A final alternative is to interface with third-party credential stores. You can link it to the JBoss Application Server's authentication component.

You will find the **JBossAuthenticator** class provided in the **docs/examples/auth** directory. This class enables jUDDI deployments on JBoss to use a server security domain to authenticate users.

Report a bug

18.6. CONFIGURE XMLDOCAUTHENTICATION

Procedure 18.2. Task

1. Create a text file called **juddi-users.xml** and save it in **jbossesb-registry.sar**.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<juddi-users>
    <user userid="sscholl" password="password" />
    <user userid="dsheppard" password="password" />
    <user userid="vbrittain" password="password" />
</juddi-users>
```

2. Save the file and exit.
3. Add the file to the class-path.
4. Open the **juddi.properties** file in your text editor (located in **SOA_ROOT/jboss-as/server/PROFILE/deploy/juddi-service.sar/juddi.war/WEB-INF**).
5. Modify the file so that it looks like this:

```
juddi.authenticator = org.apache.juddi.auth.XMLDocAuthentication
juddi.usersfile = juddi-users.xml
```

6. Save the file and exit.

Report a bug

18.7. LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP)

Lightweight Directory Access Protocol (LDAP) is a protocol for accessing distributed directory information over the internet.

Report a bug

18.8. CONFIGURE LDAP AUTHENTICATION

Procedure 18.3. Task

1. Locate the **juddi.properties** file in **SOA_ROOT/jboss-as/server/PROFILE/deploy/juddi-service.sar/juddi.war/WEB-INF**. Open it in your text editor.

2. Add the following configuration settings:

```
juddi.authenticator=org.apache.juddi.auth.LdapSimpleAuthenticator
juddi.authenticator.url=ldap://localhost:389
```

   The juddi.authenticator.url property tells the **LdapSimpleAuthenticator** class where the LDAP server resides.

3. Save the file and exit.

Report a bug

18.9. CONFIGURE JBOSS AUTHENTICATION

Procedure 18.4. Task

1. Locate the **juddi.properties** file in **SOA_ROOT/jboss-as/server/PROFILE/deploy/juddi-service.sar/juddi.war/WEB-INF**. Open it in your text editor.

2. Add the following lines to the file:

```
uddi.auth=org.apache.juddi.auth.JBossAuthenticator
juddi.securityDomain=java:/jaas/other
```

   The juddi.authenticator property connects the **JbossAuthenticator** class to the jUDDI Registry's Authenticator framework. The **juddi.security.domain** tells **JBossAuthenticator** where it can find the Application Server's security domain. It uses this domain to perform the authentications.

   Note that JBoss creates one security domain for each application policy element in the **SOA_ROOT/jboss-as/server/PROFILE/conf/login-config.xml** file. These domains are bound to the server JNDI tree with this name: **java:/jaas/<application-policy-name>**. (If a look-up refers to a non-existent application policy, a policy named **other** will be used by default.)

3. Save the file and exit.

Report a bug

PART VI. OPERATION

CHAPTER 19. RUNNING THE JBOSS ENTERPRISE SOA PLATFORM IN A PRODUCTION ENVIRONMENT

19.1. SERVER PROFILES

Table 19.1. Server Profiles

Profile	Description
default	Use this profile for development and testing. This profile uses less memory than the production profile but clustering is not enabled in this mode. In addition, this profile provides more verbose logging than the "all" and "production" profiles. This verbose logging provides you with additional information, but adversely affects server performance. Unless you explicitly specify a different profile, this profile is used when the server is started.
production	Use this profile on production servers. This profile provides clustering and maximizes performance by using more memory and providing less verbose logging and screen console output than the "all" or "default" profiles. Note that output (such as the message from the "Hello World" quick start) does not appear on the console screen in this mode. It is written to the log only.
minimal	Enables the minimum features needed for a functioning system. No archives are deployed. No ESB or SOA features are enabled. The BPEL Engine is not available.
standard	This provides standard functionality for testing. No web, ESB, or SOA features are enabled. The BPEL Engine is not available.
web	The jbossweb.sar archives are deployed when this profile is run. No ESB, or SOA features are enabled. The BPEL Engine is not available.
all	All of the pre-packaged ESB archives are deployed when this profile is run. This profile offers less performance and scalability than the "production" profile, but requires less memory to run.

Report a bug

19.2. RUN.SH OPTIONAL SWITCHES

Table 19.2. ./run.sh Optional Switches

Switch	Purpose	Example of Use
-c	Make the server use a specific profile. If none is specified, "default" is used.	`./run.sh -c production`
-b	Bind the server to a specific IP address. If none is specified, the default (127.0.0.1) is used.	`./run.sh -b 10.34.5.2`

Report a bug

19.3. START THE JBOSS ENTERPRISE SOA PLATFORM IN A PRODUCTION ENVIRONMENT

Procedure 19.1. Start the JBoss Enterprise SOA Platform in a Production Environment

1. **Navigate to the bin Directory**
   Open a terminal and input this command: **`cd SOA_ROOT/jboss-as/bin`** (or **`chdir SOA_Root\jboss-as\bin`** in Microsoft Windows).

   NOTE

   It is required that you have set up an administration username and password before proceeding.

2. **Launch the JBoss Enterprise SOA Server on Red Hat Enterprise Linux**
   To start the product, run this command: **`./run.sh -c production`**

3. **Launch the JBoss Enterprise SOA Server on Microsoft Windows**
   To start the product, run this command: **`run.bat -c production`**

Result

The server starts. Note that this may up to around two minutes, depending on the speed of your hardware.

NOTE

To verify that there have been no errors, check the server log: **`less SOA_ROOT/jboss-as/server/PROFILE/log/server.log`**. As another check, open a web browser and go to http://localhost:8080. Make sure you can log into the admin console with the username and password you have set.

Report a bug

19.4. SERVER INSTALLATION

A server installation is a way of configuring the JBoss Enterprise SOA Platform on your system. When the software is installed in this way, it can be launched and shutdown with the host operating system. It is set up just like any other service (or daemon in Linux/Unix terminology) on your operating system.

Report a bug

19.5. CONFIGURE THE JBOSS ENTERPRISE SOA PLATFORM TO RUN AS A RED HAT ENTERPRISE LINUX DAEMON

Procedure 19.2. Task

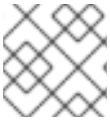
- To make the JBoss Enterprise SOA Platform run as a background daemon (service), you will have to create your own shell script. Red Hat does not supply any scripts to do this.

Report a bug

19.6. START A SERVER INSTALLATION

Prerequisites

- The JBoss Enterprise SOA Platform must be pre-configured to run as a service.

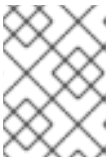


NOTE

This example assumes the service was installed using the name jboss_soa

Procedure 19.3. Task

- To start the JBoss Enterprise SOA Platform as a service, issue this command: **`service jboss_soa start`**



NOTE

If the JBoss user was created as a system account (using the **`-R`** switch) then a warning message is displayed. You can safely ignore this.

Report a bug

19.7. STOP A SERVER INSTALLATION

This example assumes the service was installed using the name jboss_soa

Procedure 19.4. Task

- To stop the JBoss Enterprise SOA Platform when it is running as a service, issue this command: **`service jboss_soa stop`**

Report a bug

PART VII. REMOVAL

CHAPTER 20. REMOVAL

20.1. REMOVE THE JBOSS ENTERPRISE SOA PLATFORM FROM YOUR SYSTEM

Procedure 20.1. Remove the JBoss Enterprise SOA Platform from Your System

1. **Remove the JBoss Enterprise SOA Platform from a Red Hat Enterprise Linux System**
   Having made sure the server is shut down, navigate to the level above the SOA_ROOT directory and issue this command: `rm -Rf SOA_ROOT`.

2. **Remove the JBoss Enterprise SOA Platform from a Microsoft Windows System**
   Having made sure the server is shut down, open Windows Explorer, go to the directory in which the SOA_ROOT is located, select the SOA_ROOT and delete it.

3. Delete the database.

Report a bug

APPENDIX A. CONFIGURING THE JBOSS ENTERPRISE SOA PLATFORM FOR CLOUD COMPUTING

A.1. AMAZON ELASTIC COMPUTE CLOUD (EC2)

Amazon Elastic Compute Cloud (EC2) is a service provided by http://aws.amazon.com/ec2/. You can rent virtual machines on which you run your own software. It is a highly-scalable service and allows you to create virtual machines in realtime to meet specific processing needs.

Report a bug

A.2. CONFIGURE THE JBOSS ENTERPRISE SOA PLATFORM TO BE USED ON AN EC2 CLOUD

Prerequisites

- 

  WARNING

  Because this process requires you to shut down your server's firewall leading to a potential security risk, this configuration task MUST be performed inside a VPC subnet, with a firewall on the external interface. Red Hat will NOT support it otherwise.

Procedure A.1. Task

1. Create the nodes on EC2. They should belong to a security group that does not have restrictions.
2. Within AWS also create an instance that will be used as a database.
3. Create a new S3 bucket on AWS.
4. Download the JBoss Enterprise SOA Platform EC2 patch from the Customer Portal.
5. Unzip the patch and open it from the location you saved it in.
6. Run **`ant`**.

   This creates a new configuration called cluster-ec2, based on the **`production`** configuration.
7. Make any desired changes to the new configuration (such as modifying it to use MySQL instead of the default database.)
8. Shut down the firewall (iptables) on all nodes.
9. Run the newly-created configuration, passing the following parameters: jboss.jgroups.s3_ping.access_key, jboss.jgroups.s3_ping.secret_access_key and jboss.jgroups.s3_ping.bucket. (You can obtain these from the AWS console)

Report a bug

A.3. TROUBLESHOOTING AN EC2 CONFIGURATION

- Ensure that the patch's version number matches that of the JBoss Enterprise SOA Platform distribution.
- If, after starting a large number of nodes, you still see that the cluster has only one member, it is probably a problem related to your firewall.
- Make sure you are not connected to an internal VPN while you are starting the JBoss Enterprise SOA Platform.

Report a bug

APPENDIX B. INSTALLING JBPM 5

B.1. INSTALL JBPM 5

Prerequisites

- A subscription to BRMS 5.3 or greater
- The BRMS deployable package installed on the SOA Server

Procedure B.1. Task

1. Download **`soa-p-VERSION-jbpm.zip`** from the Customer Portal.
2. Open the **`build.properties`** file in a text editor. Make the required configuration changes.
3. Save the **`build.properties`** file and exit the text editor.
4. Run the installation script.

   This script deploys the **`jbpm5.esb`** file from the **`soa-p-VERSION-jbpm.zip`** to the SOA-P server, configures the ESB service repository and copies the jBPM5 console, human task WAR file, and libraries from the BRMS installation to the SOA server.

   Alternatively, if the script does not work, see https://bugzilla.redhat.com/attachment.cgi?id=590710 for the latest instructions.
5. Start the JBoss Enterprise SOA Platform Server.
6. To test the installation, navigate to the directory where you installed the jBPM 5 quickstart: **`cd SOA_ROOT/jboss-as/samples/quickstarts/bpm5processor`**.
7. Deploy the quickstart: **`ant deploy`**
8. Run the quickstart: **`ant runtest`**

Report a bug

APPENDIX C. SOME USEFUL DEFINITIONS

C.1. ENTERPRISE SERVICE BUS

An enterprise service bus is a concrete implementation of an abstract SOA design philosophy. An enterprise service bus (ESB) has two roles: it provides message routing functionality and allows you to register services. The enterprise service bus that lies at the center of the JBoss Enterprise SOA Platform is called JBoss ESB.

An enterprise service bus deals with infrastructure logic, not business logic (which is left to higher levels). Data is passed through the enterprise service bus on its way between two or more systems. Message queuing may or may not be involved. The ESB can also pass the data to a transformation engine before passing it to its destination.

Report a bug

C.2. JAVA VIRTUAL MACHINE

A Java Virtual Machine (JVM) is a piece of software that is capable of running Java bytecode. The JVM creates a standard environment in which the intermediate bytecode is run. By creating the standard environment irrespective of the underlying hardware and operating system combination, it allows programmers to write their Java code once and have confidence that it can be run on any system. Red Hat recommends customers use OpenJDK as it is an open source, supported Java Virtual Machine that runs well on Red Hat Enterprise Linux systems. Windows users should install Oracle JDK 1.6.

Report a bug

C.3. SOA-USERS.PROPERTIES

The **`soa-users.properties`** file is where the user accounts and passwords for accessing the SOA Web consoles are stored. Administrators control access to the system by editing this file. Note that the passwords are saved in clear text so for production systems, password encryption should be used instead.

Report a bug

C.4. SOA-ROLES.PROPERTIES

The **`soa-roles.properties`** file is where user access privileges are defined. This file uses the following syntax: **`username=role1,role2,role3`**. You can assign any number of roles. Note that a user must be assigned the **`JBossAdmin`**, **`HttpInvoker`**, **`user`**, and **`admin`** roles in order to be able to log into the server consoles.

Report a bug

C.5. RUN.SH

`run.sh` is the shell script the user runs to launch the JBoss Enterprise SOA Platform. The Microsoft

Windows equivalent is **run.bat**. The script contains the commands needed to start the server with the profile and port binding which the user has specified in the shell. The script is found in the **SOA_ROOT/jboss-as/bin** directory.

Report a bug

C.6. RUN.CONF

SOA_ROOT/bin/run.conf is the default server configuration file. **run.conf.bat** is the Microsoft Windows equivalent.

Report a bug

C.7. BOOT-STRAPPER MODE

Putting your software into boot-strapper mode tells it what to load and when to do so.

Report a bug

C.8. MESSAGE RE-DELIVERY SERVICE

The Message Re-delivery Service attempts to redeliver messages when none of the end-point references work.

Report a bug

C.9. ACTION PIPELINE

The action pipeline consists of a list of action classes through which messages are processed. Use it to specify which actions are to be undertaken when processing the message. Actions can transform messages and apply business logic to them. Each action passes the message on to the next one in the pipeline or, at the conclusion of the process, directs it to the end-point listener specified in the ReplyTo address.

The action pipeline works in two stages: normal processing followed by outcome processing. In the first stage, the pipeline calls the process method(s) on each action (by default it is called "process") in sequence until the end of the pipeline has been reached or an error occurs. At this point the pipeline reverses (the second stage) and calls the outcome method on each preceding action (by default it is processException or processSuccess). It starts with the current action (the final one on success or the one which raised the exception) and travels backwards until it has reached the start of the pipeline.

Report a bug

C.10. CLASS-PATH

A classpath is a setting that tells the Java Virtual Machine where, on the filesystem, to find user-created classes and packages.

Report a bug

C.11. BUSINESS PROCESS DEFINITION

A business process definition determines the common elements of any runtime instances being used in a process. It is reusable.

Report a bug

C.12. SERVER PROFILES

A server profile is a set of pre-determined settings for running the JBoss Enterprise SOA Platform in different ways. The following profiles come with the product: all, default, minimal, production, standard and web. They are found in the **`SOA_ROOT/jboss-as/server/`** directory. The user specifies which profile to run when launching the software by using the **`-c`** switch. If none is specified, the "Default" profile is used.

Report a bug

C.13. DATASOURCE NAME

A datasource name (DSN) is the title given to a particular piece of data. For example, a DSN could refer to the name of a database.

Report a bug

C.14. DECISION TABLE

A decision table contains a list of actions. These are undertaken by the system when required.

Report a bug

C.15. SERVICE

A service is a list of action classes that process an ESB Message in a sequential manner. Each service element consists of one or more listeners and one or more actions. These are set within the **`jboss-esb.xml`** configuration file.

Report a bug

C.16. STATELESS SERVICE

A stateless service is a self-contained service that independently performs tasks instead of having to receive instructions from the user. Additionally, it does not need to use up vast amounts of data to identify objects.

Report a bug

C.17. SERVICE BINDING

A service binding allows you to transport data between clients and services by linking them.

Report a bug

C.18. ENTERPRISE JAVA BEAN

An Enterprise Java Bean is a Java component architecture designed for enterprise applications. It can be used to create these applications and then deploy them to a server.

Report a bug

C.19. LOOSE COUPLING

Loose coupling is when two components are linked together to perform certain tasks, but remained unlinked the rest of the time.

Report a bug

C.20. PERSISTENCE MECHANISM

A persistence mechanism is a fail-over property. It makes an object persistent, meaning it can automatically start up again after a shutdown and resume the task it was previously performing.

Report a bug

C.21. RESOURCE ADAPTER

A resource adapter allows you to modify an application so that other components can be "plugged" into it. These components are then able to communicate with the rest of the system using the adapter.

Report a bug

C.22. SHELL SCRIPT

A shell script is a text file containing a series commands for UNIX-based operating systems like Red Hat Enterprise Linux. They call on the shell (terminal) when they run. The Microsoft Windows equivalent is a batch file.

Report a bug

C.23. WEB CONTAINER

A web container works with Java servlets. It is responsible for managing their performance as well as making sure they are sending and receiving the correct information. The JBoss Enterprise Application Platform is a type of web container.

Report a bug

C.24. INITIAL CONTEXT FACTORY

An initial context factory is where initial context objects are created. These objects are used to create and view naming and directory properties.

Report a bug

C.25. USERNAMETOKEN

The UsernameToken is used to "propagate" a username (and optionally a password) throughout a system to avoid having to login multiple times in a single session.

Report a bug

C.26. SCHEMA VALIDATION

This is the process of validating or "checking" code to make sure it works. You can make sure there are no errors in your XML code by running it through a schema validation.

Report a bug

C.27. BYTE ARRAY

As the name suggests, this is an array of bytes that make up objects like memory. You can create an array of bytes for use with message sending and processing packets.

Report a bug

C.28. EXTENDED TRANSACTIONAL CLIENT

An extended transactional client allows you to send and receive messages from local queue managers. It also allows you to view and update external queue managers.

Report a bug

C.29. CONNECTION POOLING

Connection pooling is a back-end way of connecting a server to multiple clients. Once a connection pool is created, an application server can draw on it to perform stored actions (for example, requesting a

database to do a certain task). It simplifies common tasks as the actions are ready to go in the connection pool for when the user decides to deploy them.

Report a bug

C.30. POOLED DATABASE MANAGER

As the name implies, this manager works with pooled databases and allows for them to be accessed, managed and configured efficiently.

Report a bug

C.31. CIPHER TRANSFORMATION

Use this transformation to decrypt information.

Report a bug

C.32. CONCURRENCY CONTROL

This method of control allows multiple operations to run concurrently while making sure all their processes are running correctly and efficiently.

Report a bug

C.33. UNIFORM RESOURCE IDENTIFIER

A uniform resource identifier (URI) uses a sequence of alphanumeric characters to identify a resource in the system. A web URL is one type of URI.

Report a bug

C.34. PROVIDER ADAPTER

A provider adapter allows applications to receive information from remote providers.

Report a bug

C.35. IMPLEMENTATION CLASS

An implementation class defines how an object which belongs to certain classes is implemented.

Report a bug

C.36. INTERCEPTOR CLASS

An interceptor class is applied to an object to make it perform additional actions that have been defined in the class.

Report a bug

C.37. TRANSACTED FLAG

You can set your session to have a transacted flag with the value of true or false. It can be set to true on specific endpoints to make them transactional. This means that all the actions of the endpoint can be grouped into one singular action instead of lots of smaller ones.

Report a bug

C.38. JAVA CONNECTOR ARCHITECTURE (JCA) TRANSPORT

The Java Connector Architecture (JCA) Transport is a Java-based piece of architecture that works as a service integrator. It is a connector that links application servers and enterprise information systems.

Report a bug

C.39. JCA BRIDGE

The JCA bridge is a dispatcher which can open and close connections. It identifies connections set by the user and can detect connectors and gateways.

Report a bug

C.40. JCA ADAPTER

The JCA adapter acts as a "go between" that links application servers and enterprise information systems.

Report a bug

C.41. END-POINT CLASS

An end-point class lets you identify resources and services on your network by providing their network address.

Report a bug

APPENDIX D. GLOBAL CONFIGURATION FILE

D.1. JBOSSESB-PROPERTIES.XML

The **`jbossesb-properties.xml`** file is the JBoss Enterprise SOA Platform's global configuration file. Many tasks will require you to edit this file. The location of this file varies depending on how the system has been installed. If you have installed a server deployment, this file will be located at **`SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**, while standalone clients can access it directly through the class-path.

Report a bug

D.2. GLOBAL CONFIGURATION FILE REFERENCE

The global configuration file (**`SOA_ROOT/jboss-as/server/PROFILE/deployers/esb.deployer/jbossesb-properties.xml`**) is split into sections, each concerned with a specific area of configuration. A named property section contains one or more properties which are used to configure the behavior of the ESB, and one property section can "depend" on another section - the dependency specifies which sections are loaded by the PropertyManager first.

core

- org.jboss.soa.esb.jndi.server.context.factory : The JNDI Server initial context factory.
- org.jboss.soa.esb.jndi.server.url : The JNDI Server URL.
- org.jboss.soa.esb.loadbalancer.policy : The ESB load balancer policy.
- org.jboss.soa.esb.mime.text.types : A semicolon-separated list of MIME types that are used to decide whether the payload can be decoded or whether it will remain as a byte array.
- jboss.esb.invm.scope.default : The default InVM scope for an ESB deployment.
- org.jboss.soa.esb.deployment.schema.validation : A true/false flag to enable JBoss ESB schema validation upon deployment.

IMPORTANT

The org.jboss.soa.esb.jndi.server.type and org.jboss.soa.esb.persistence.connection.factory properties within core are now obsolete.

security

- org.jboss.soa.esb.services.security.implementationClass : The concrete SecurityService implementation to be used.
- org.jboss.soa.esb.services.security.callbackHandler : The default callback handler implementation.
- org.jboss.soa.esb.services.security.sealAlgorithm : The algorithm to be used when sealing the SecurityContext.

- org.jboss.soa.esb.services.security.sealKeySize : The size of the key to be used to encrypt/decrypt the SecurityContext.
- org.jboss.soa.esb.services.security.contextTimeout : The amount of time for which SecurityContext is valid.
- org.jboss.soa.esb.services.security.contextPropagatorImplementationClass : Used to configure a global SecurityContextPropagator.
- org.jboss.soa.esb.services.security.publicKeystore : Keystore to encrypt and decrypt data external to the ESB.
- org.jboss.soa.esb.services.security.publicKeystorePassword : The keystore password.
- org.jboss.soa.esb.services.security.publicKeyAlias : The public key alias to use.
- org.jboss.soa.esb.services.security.publicKeyPassword : The public key password to use.
- org.jboss.soa.esb.services.security.publicKeyTransformation : The cipher transformation to use.

registry

- org.jboss.soa.esb.registry.queryManagerURI : The registry query manager URI, which is used to obtain information on services and bindings.
- org.jboss.soa.esb.registry.lifeCycleManagerURI : The registry lifecycle manager URI, which is used to publish information on services and bindings.
- org.jboss.soa.esb.registry.securityManagerURI : The registry security manager URI, which is used to authenticate queries to the registry.
- org.jboss.soa.esb.registry.implementationClass : The JBoss ESB registry implementation class. The JAXR registry implementation is used here.
- org.jboss.soa.esb.registry.factoryClass : The registry factory class, which specifies which JAXR implementation should be used.
- org.jboss.soa.esb.registry.user : The registry user.
- org.jboss.soa.esb.registry.password : The registry password.
- org.jboss.soa.esb.scout.proxy.transportClass The Scout transport class which defines which transport should be used to communicate with the UDDI registry.
- org.jboss.soa.esb.scout.proxy.uddiVersion : The Scout UDDI Version. This is an Apache Scout-specific setting.
- org.jboss.soa.esb.scout.proxy.uddiNameSpace : The Scout UDDI namespace. This is an Apache Scout-specific setting.
- org.jboss.soa.esb.registry.interceptors : The registry interceptor class names.
- org.jboss.soa.esb.registry.cache.maxSize : The maximum cache size for the caching registry.
- org.jboss.soa.esb.registry.cache.validityPeriod : The validity period for the caching registry.

- org.jboss.soa.esb.registry.orgCategory : The UDDI organization value to use - note that this is a UDDI-specific value.

transports

- org.jboss.soa.esb.mail.smtp.host : The host name of the SMTP server.
- org.jboss.soa.esb.mail.smtp.user : The username to use for the SMTP server.
- org.jboss.soa.esb.mail.smtp.password : The password for the user specified on the SMTP server.
- org.jboss.soa.esb.mail.smtp.port : The port number of the SMTP server.
- org.jboss.soa.esb.mail.smtp.auth : Flag which specifies whether to authenticate the user against the SMTP server using the AUTH command.
- org.jboss.soa.esb.ftp.localdir : FTP local directory.
- org.jboss.soa.esb.ftp.remotedir : FTP remote directory.
- org.jboss.soa.esb.ftp.timeout : FTP timeout in milliseconds for opening a socket.
- org.jboss.soa.esb.ftp.timeout.data : FTP timeout in milliseconds for the data connection.
- org.jboss.soa.esb.ftp.timeout.so : FTP timeout in milliseconds used for currently open sockets.
- org.jboss.soa.esb.ftp.timeout.default : FTP timeout in milliseconds which sets the default timeout.
- org.jboss.soa.esb.jms.connectionPool : Size of the ESB JMS connection pool.
- org.jboss.soa.esb.jms.sessionSleep : If a JMS session cannot be obtained, the ESB will keep trying to obtain one. The sessionSleep property decides how long the ESB will try for.
- org.jboss.soa.esb.invm.expiryTime : The expiry time for messages in the InVM temporary transport.
- org.jboss.soa.esb.invm.retry.limit : Maximum number of times to retry redelivery. The default is 5.
- org.jboss.soa.esb.ws.returnStackTrace : True/false flag that determines whether to return stack traces upon fault of SOAP messages.
- org.jboss.soa.esb.ws.timeout : Service invoker timeout for delivering SOAP messages within RequestResponseBaseWebService.
- org.jboss.soa.esb.aggregator.setOnProperties : Aggregate on properties of the message rather than on Context.

jca

- org.jboss.soa.esb.jca.activation.mapper.jms-ra.rar : Specifies the ActivationMapper globally.
- org.jboss.soa.esb.jca.activation.mapper.wmq.jmsra.rar : Specifies the ActivationMapper globally.

dbstore

- org.jboss.soa.esb.persistence.db.conn.manager : Connection Manager implementation class name.
- org.jboss.soa.esb.persistence.db.datasource.name : Datasource name, only used if using the J2EE connection manager.
- org.jboss.soa.esb.persistence.db.connection.url : The JDBC connection URL.
- org.jboss.soa.esb.persistence.db.jdbc.driver : The JDBC driver class.
- org.jboss.soa.esb.persistence.db.user : The database user.
- org.jboss.soa.esb.persistence.db.pwd : The database password.
- org.jboss.soa.esb.persistence.db.pool.initial.size : The initial number of database connections.
- org.jboss.soa.esb.persistence.db.min.size : The minimum number of database connections.
- org.jboss.soa.esb.persistence.db.max.size : The maximum number of database connections.
- org.jboss.soa.esb.persistence.db.pool.test.table : A table name to query for validity of the database connection.
- org.jboss.soa.esb.persistence.db.pool.timeout.millis : Timeout in milliseconds of the database connections.

filters

- org.jboss.soa.esb.filter.1, org.jboss.soa.esb.filter.2, org.jboss.soa.esb.filter.3, etc.

rules

- org.jboss.soa.esb.services.rules.resource.scanner.interval : Defines the polling interval for DRL changes globally across all KnowledgeAgents.
- org.jboss.soa.esb.services.rules.continueState : Setting this property will enable legacy behaviour and not dispose of working memories during stateful rule execution.

IMPORTANT

The message routing properties (org.jboss.soa.esb.routing.cbrClass) are obsolete but may still exist in some files.

Report a bug

APPENDIX E. REVISION HISTORY

Revision 5.3.1-103.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revision 5.3.1-103 **Tue Feb 05 2013** **David Le Sage**

Built from Content Specification: 6483, Revision: 371693 by dlesage