JBoss Enterprise SOA Platform 4.3

Administration Guide

Your guide to administering the JBoss Enterprise SOA Platform 4.3 CP05

JBoss Enterprise SOA Platform 4.3 Administration Guide
Your guide to administering the JBoss Enterprise SOA Platform 4.3 CP05
Edition 4.3.5

The Administration Guide contains guidance on how to configure and manage installations of JBoss SOA Platform.

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;
```

```
public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *http://bugzilla.redhat.com/bugzilla/* against the product **JBoss Enterprise SOA Platform.**

When submitting a bug report, be sure to mention the manual's identifier: *SOA_ESB_Administration_Guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Configuration

This book teaches system administrators how to configure the **JBoss SOA Platform Enterprise Service Bus** for use in financial institutions, banks, telecommunications companies and other large corporations. This chapter explains how to configure the software after the reader has followed the initial installation steps in the *Getting Started Guide*.

1.1. Introduction to Basic Concepts

A *Service Oriented Architecture* (SOA) is not a single program or technology. Rather, one should think of it as a software architectural or design paradigm. A hardware *bus* is a physical connector that ties together multiple systems and subsystems. Instead of having a large number of point-to-point connectors between pairs of systems, one can simply connect each system to the bus just once. An *enterprise service bus* (ESB) does the same thing in software.

Instead of passing data over the bus via the connections (known as *end points*), the ESB sits, logically, in the architectural layer above a messaging system. This messaging system allows asynchronous communications to occur between services over the ESB. In fact, when one is using an ESB, everything is, conceptually, either a *service* (which in this context is one's application software) or a *message* being sent between services.

It is important to note that a “service” may not necessarily be a web service. Other types of application, using various transports such as File Transfer Protocol and Java Message Service, can also be "services."

Note

At this point, one may be wondering if an Enterprise Service Bus is the same thing as a Service Oriented Architecture. The answer is, "Not exactly." This is because an ESB does not actually 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 SOA as being more than just software: it is a series of principles, patterns and best practices.

The **JBoss Enterprise Service Bus** is an open source ESB, provided by Red Hat. It supports:

- multiple transports and protocols
- a *listener-action model* (used to loosely-couple services)
- *content-based routing* (through **JBoss Rules**)
- integration with the **JBoss Business Process Manager** (to facilitate *service orchestration*.)

1.2. Stand-Alone Server

To learn how to run the **JBoss Enterprise SOA Platform** server on the same machine as another JBoss Application Server, study the information on this website: *http://www.jboss.org/community/wiki/ConfiguringMultipleJBossInstancesOnOnemachine*.

1.3. Clustered ESB Service

To set up a clustered Enterprise Service Bus environment, configure the nodes in one of the following two ways:

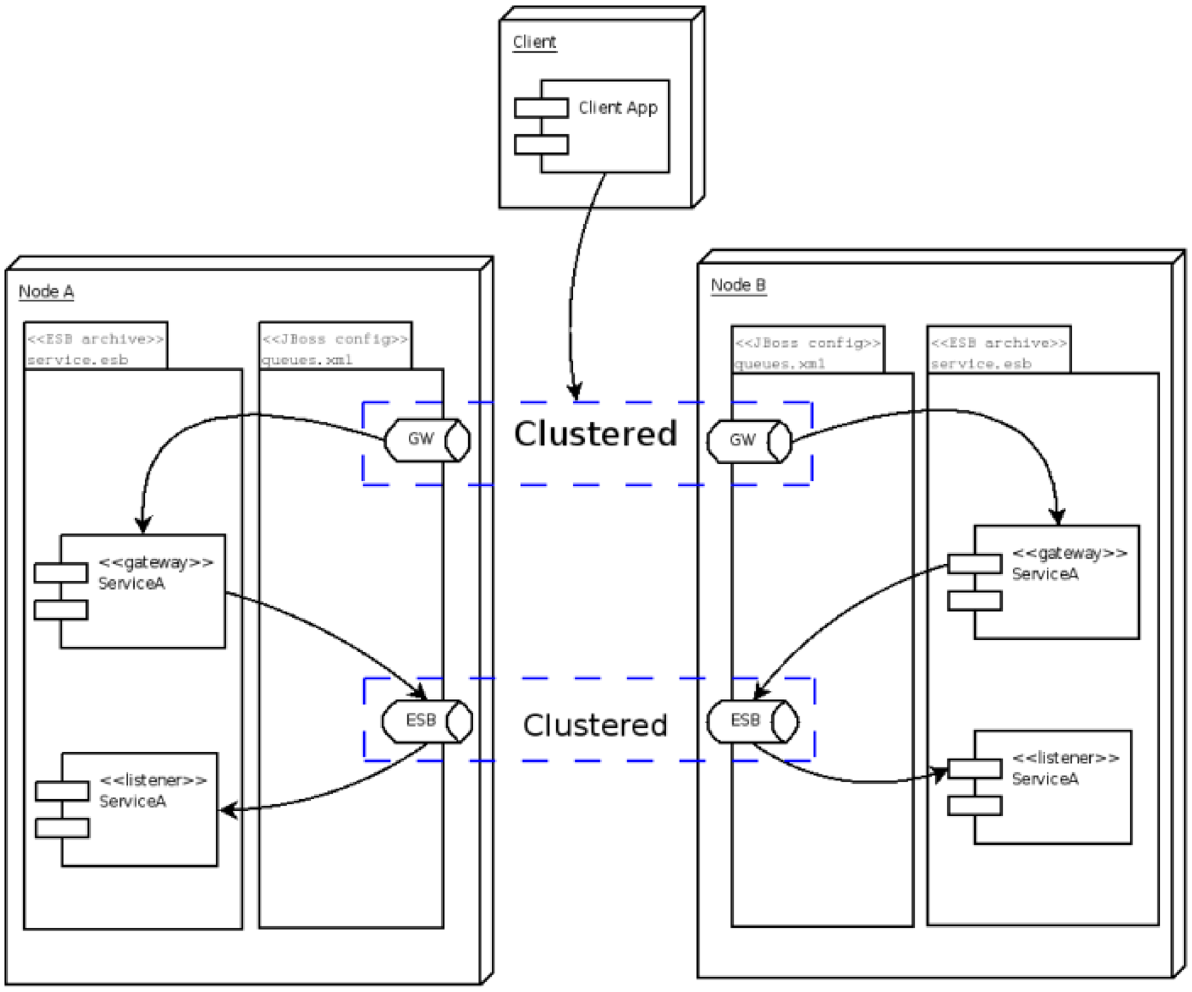


Figure 1.1. Clustering Scenario One

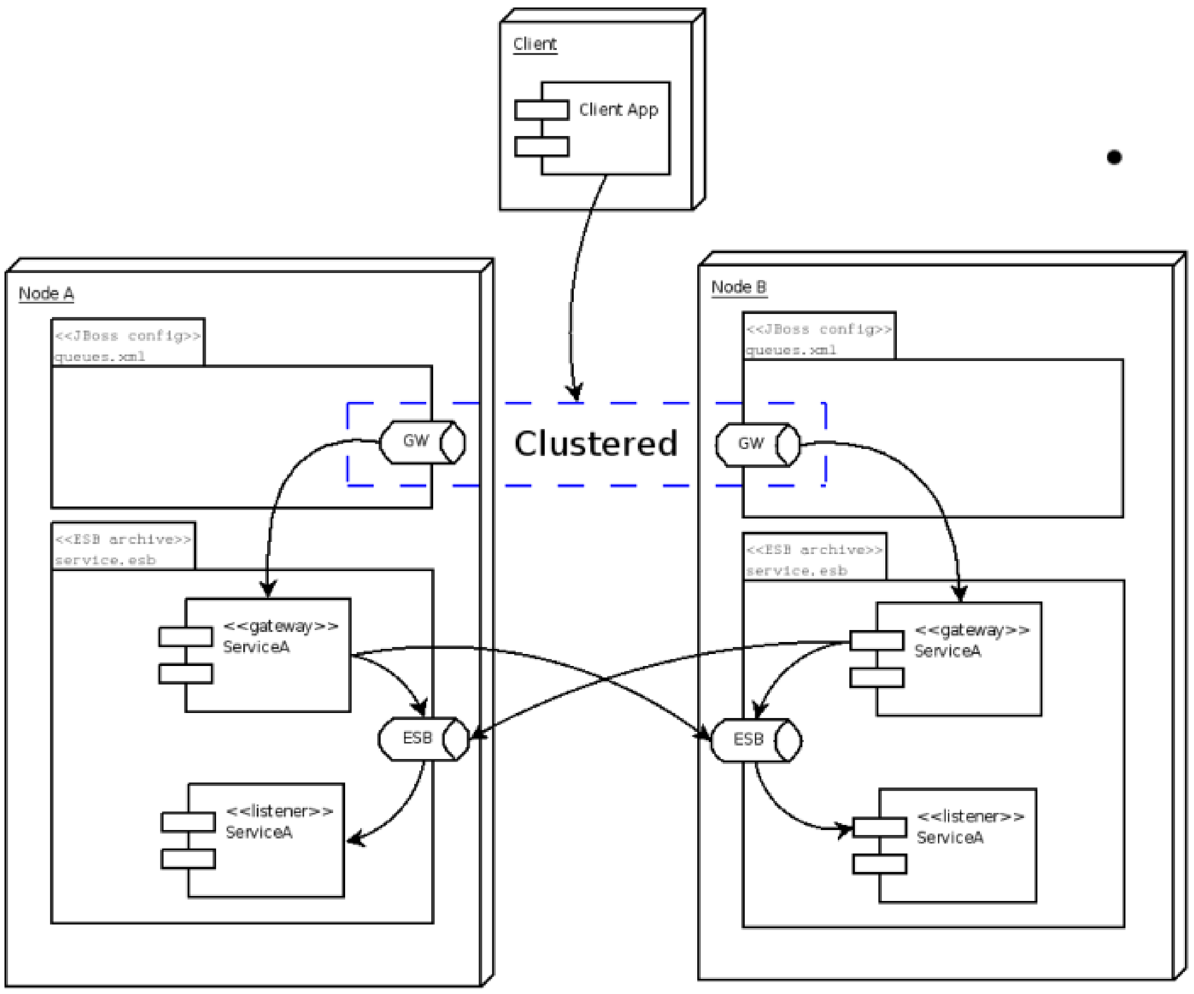


Figure 1.2. Clustering Scenario Two

The difference between these methods is that the Enterprise Service Bus-aware gateways are *clustered* in the first case and *duplicated* in the other.

Two other points to bear in mind are that:

- every clustered queue must be deployed on each and every cluster node at all times.
- the client must list all of the nodes in the **`jndi.properties`** file. This example code shows how:

```
java.naming.provider.url=jnp://jawa01:1099
java.naming.provider.url=jnp://jawa02:1099
```

1.4. Java Message Service Providers

The JBoss Enterprise SOA Platform currently supports the following *Java Message Service* (JMS) providers:

- **JBoss Messaging**
- **IBM Websphere MQ Series 6.0**
- **Tibco EMS**

Important

Red Hat recommends using **JBoss Messaging**, which is included with the default configuration.

Any *JSR-914* (*http://jcp.org/en/jsr/detail?id=914*)-compliant Java Message Service implementation (such as **Apache ActiveMQ** or **OracleAQ**) should also work. However, they have not been fully tested and, hence, are not supported at this time. If intending to try another vendor's product, please consult their documentation.

Warning

This section is not intended to be a replacement for the configuration documentation that comes with the supported Java Message Service implementations. Consult that documentation to learn about advanced capabilities, such as *clustering* and *management*.

How Are They Configured?

Configure `JMS Listeners` and `JMS Gateways` to listen to *Queues* and *Topics* by specifying the following parameters in their configuration files (namely **`jbossesb-listener.xml`** and **`jbossesb-gateway.xml`**):

- jndi-URL
- jndi-context-factory
- jndi-pkg-prefix
- connection-factory
- destination-type
- destination-name

Important

Ensure that the client **`JAR`** files for the chosen JMS provider are included in the classpath.

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 that of 'queue'
- the destination-name is that of 'queue/A'

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

Note

Every `JMSListener` and `JMSGateway` can be configured to use its own JMS provider. Hence, one can use multiple providers within a single deployment.

The SOA Platform utilizes a *connection pool* to improve performance when the Java Message Service is used. The default size of this pool is **`20`**. To change this value, set the org.jboss.soa.esb.jms.connectionPool property in the **`transports`** section of the ESB configuration file.

The service will keep re-trying for up to thirty seconds if an initial session cannot be obtained. (This time-out period can be configured by using the org.jboss.soa.esb.jms.sessionSleep property.)

1.4.1. Maximum Sessions per Connection

As its name implies, the **`JmsConnectionPool`** pools the JMS Sessions that are to be used by all Java Message Service components including JMS Listeners, Couriers and Routers.

Be aware that some JMS providers limit the number of sessions per connection. In this case, for each **`JmsConnectionPool`** instance, specify the maximum number of sessions created by each JMS Connection. Do this by specifying one or both of the following properties in the applicable JMS component's JNDI configuration file:

`org.jboss.esb.jms.max.sessions.per.connection`
: This is the maximum total number of Sessions allowed per connection (including both XA and non-XA Session instances.) The default is the maximum number of JMS Sessions allowed for a **`JmsConnectionPool`**, normally **`20`** (as configured in the **`jbossesb-properties.xml`** file.)

`org.jboss.esb.jms.max.xa.sessions.per.connection`
: This is the maximum number of XA Sessions allowed per connection. This value defaults to that of **`org.jboss.esb.jms.max.sessions.per.connection`**.

If neither of the parameters above are configured, the **`JmsConnectionPool`** will create a single JMS Connection and use it to create all of the JMS Sessions.

Make these configuration changes via the generic properties on the JMS Provider configuration, as per the following example:

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

1.4.2. JBoss Messaging

JBoss Messaging is the default JMS provider for the **JBoss SOA Platform**.

Set its parameters so that they reflect the following configuration:

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

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

Always include the **`jboss-messaging-client.jar`** file in the class path.

Note

Configuring **JBoss Messaging** for use in a clustered environment gives one load balancing and fail-over facilities for JMS. Since this capability has changed between different versions of **JBoss Messaging** and may continue to do so, always consult the documentation for the version to be used.

1.4.3. Apache ActiveMQ

Warning

Apache ActiveMQ has not been fully tested and is, therefore, not a supported Java Message Service implementation.

To use, set the parameters to the following:

```
jndi-URL="tcp://localhost:61616"
jndi-context-factory="org.apache.activemq.jndi.ActiveMQInitialContextFactory"
connection-factory="ConnectionFactory"
destination-type="queue"
destination-name="queue/A"
```

Next, ensure that the class-path contains:

- **`activemq-core-4.x`**
- **`backport-util-concurrent-2.1.jar`**

Both of these files are located in the **`lib/ext/jms/activemq`** sub-directory.

1.4.4. IBM Websphere MQ Series 6.0

Use the following JNDI parameters to configure the **IBM Websphere MQ** series (WMQ) on any of the ESB's Java Message Service components:

```
jndi-URL="localhost:1414/SYSTEM.DEF.SVRCONN"
jndi-context-factory="com.ibm.mq.jms.context.WMQInitialContextFactory"
connection-factory="WMQQueueManager"
destination-type="queue"
destination-name="QUEUEA"
```

Extra WMQ JAR files

Note that the connection-factory setting should reference the name of the WMQ Queue Manager. (This must be on the WMQ Server on which the Java Message Service destination is configured.)

In order to connect to a WMQ Provider from the **JBoss Enterprise Service Bus**, add some additional JAR files to the ***`$SOA_ROOT`*`/server/`*`$PROFILE`*`/lib/`** directory:

The following should be on the class-path:

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

Note

These JAR files are to be found in the WMQ installation's **`Java/lib`** directory. (The client JAR files differ between MQ 5.3 and MQ 6.0 but the 6.0 JARs should be backward compatible.) Note that these JARs are not open source and are not provided by Red Hat. You will have to obtain them from your Websphere and MQ installations.

1.4.4.1. XA Connections

To manage any *XA connection resources* on the **JBoss Enterprise Service Bus** using Websphere MQ, install the **`WMQ Extended Client`** JAR file in the ***`$SOA_ROOT`*`/server/`*`$PROFILE`*`/lib/`** directory. This JAR file is normally named **`com.ibm.mqetclient.jar`** and must be acquired from one's IBM partner or agent.

Once this library is installed, Websphere MQ will restrict the number of Java Message Service Sessions per JMS Connection to one. This restriction applies to both XA and non-XA connections.

Next, configure the **`JmsConnectionPool`**'s appropriate org.jboss.esb.jms.max.sessions.per.connection property. Simply set the value of this property to **`1`**. (See the "Max Sessions Per Connection" section of this document for more information.)

Note

The implication of setting this property is that the **`JmsConnectionPool`** will consume more Java Message Service connections on the WMQ Provider.

Note that this setting does not need to be adjusted for any Websphere MQ-based *J2EE Connector Architecture* (JCA) provider configuration.

1.4.4.2. JCA Adapter

Note

The default installation of **IBM Websphere MQ** does not include the JCA Adapter.

Procedure 1.1. Steps to Configure the JCA Adapter

1. **Update IBM Websphere MQ**

   Firstly, update the **IBM Websphere MQ** version to at least Version 6.0.2.1 in order to obtain the J2EE Connector Architecture Adapter.

2. **Deploy the Adapter on the JBoss Enterprise SOA Platform Server**
   The WMQ JCA Adapter, **`wmq.jmsra.rar`**, is found in the WebSphereMQ installation's **`Java/lib/jca/`** directory. To deploy the adapter, copy this file to the SOA Platform server's **`${SOA_ROOT}/server/${CONFIG}/deploy/`** directory.

3. **Create a JCA Connection Factory Configuration**
   Configure the SOA Platform Server to use the WMQ JCA Adapter by creating a **`JCA Connection Factory`** configuration file. (The filename does not matter but should be descriptive.) Copy this file into the SOA Platform Server's **`${SOA_ROOT}/server/{CONFIG}/deploy/`** directory.

   Example 1.1. JCA Connection Factory Configuration

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

<mbean code="org.jboss.jms.jndi.JMSProviderLoader"
    name=":service=JMSProviderLoader,name=WSMQJmsProvider">
    <attribute name="ProviderName">WSMQProvider</attribute>
    <attribute name="ProviderAdapterClass">
        org.jboss.jms.jndi.JNDIProviderAdapter
    </attribute>
    <attribute name="QueueFactoryRef">ConnectionFactory</attribute>
    <attribute name="TopicFactoryRef">ConnectionFactory</attribute>
    <attribute name="FactoryRef">ConnectionFactory</attribute>
    <attribute name="Properties">
java.naming.factory.initial=com.ibm.mq.jms.context.WMQInitialContextFactory
java.naming.provider.url=mqserver.domain.com:1414/SYSTEM.DEF.SVRCONN
    </attribute>
</mbean>

<tx-connection-factory>
    <jndi-name>WSMQJmsXA</jndi-name>
    <xa-transaction/>
    <rar-name>jms-ra.rar</rar-name>
    <connection-definition>
        org.jboss.resource.adapter.jms.JmsConnectionFactory
    </connection-definition>
    <config-property name="SessionDefaultType" type="java.lang.String">
        javax.jms.Queue
    </config-property>
    <config-property name="JmsProviderAdapterJNDI" type="java.lang.String">
        java:/WSMQProvider
    </config-property>
    <max-pool-size>20</max-pool-size>
    <security-domain-and-application>
        JmsXARealm
    </security-domain-and-application>
</tx-connection-factory>

</connection-factories>
```

   The J2EE Connector Architecture Adapter is now available. Access it via the configured connection factory.

The connection factory is now ready for use. Do so through the JCA Provider configuration to provide transactional context. (The following example does this in **`${SOA_ROOT}/server/${CONFIG}/deploy/jbpm.esb/META-INF/jboss-esb.xml`**.)

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

```
<jbossesb xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/
etc/schemas/xml/jbossesb-1.0.1.xsd" parameterReloadSecs="5">

<providers>
  <jms-jca-provider connection-factory="ConnectionFactory"
    jndi-URL="mqserver.domain.com:1414/SYSTEM.DEF.SVRCONN"
    jndi-context-factory="com.ibm.mq.jms.context.WMQInitialContextFactory"
    name="CallbackQueue-JMS-Provider"
    providerAdapterJNDI="java:/WSMQProvider">
    <jms-bus busid="jBPMCallbackBus">
      <jms-message-filter dest-name="queue/CallbackQueue"
        dest-type="QUEUE"/>
    </jms-bus>
  </jms-jca-provider>
</providers>

<services>
  <service category="JBossESB-Internal"
    description="Service which makes Callbacks into jBPM"
    name="JBpmCallbackService">
    <listeners>
      <jms-listener busidref="jBPMCallbackBus" maxThreads="1"
        name="JMS-DCQListener"/>
    </listeners>
    <actions mep="OneWay">
      <action
        class="org.jboss.soa.esb.services.jbpm.actions.JBpmCallback"
        name="action"/>
    </actions>
  </service>
</services>

</jbossesb>
```

1.4.4.3. Authentication

Warning

The following exception message may appear if **Websphere MQ** is being used. Fix the problem by adding the name of the user who runs the **JBoss Enterprise SOA Platform** to the `mqm` group.

```
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
```

The user accessing the MQ must be a member of the `mqm` group.

1.4.5. Oracle Advanced Queuing

Warning

Oracle Advanced Queuing has not been fully tested and is not a supported Java Message Service implementation.

To use Oracle Advanced Queuing, set the parameters to:

```
connection-factory=”QueueConnectionFactory”
```

Use these properties:

```
<property name="java.naming.factory.initial"
    value="org.jboss.soa.esb.oracle.aq.AQInitialContextFactory"/>
<property name="java.naming.oracle.aq.user" value="<user>"/>
<property name="java.naming.oracle.aq.password" value="<pw>"/>
<property name="java.naming.oracle.aq.server" value="<server>"/>
<property name="java.naming.oracle.aq.instance" value="<instance>"/>
<property name="java.naming.oracle.aq.schema" value="<schema>"/>
<property name="java.naming.oracle.aq.port" value="1521"/>
<property name="java.naming.oracle.aq.driver" value="thin"/>
```

Note

One may have noticed the reference to the `InitialContext` factory. This is only needed if one wishes to avoid having Oracle Advanced Queuing registering with an Lightweight Directory Access Protocol (LDAP) server.

The **`AqinitialContextFactory`** refers to code in the **`org.jboss.soa.esb.oracle.aq-4.2.jar`** file, which is located in the **`plugins/org.jboss.soa.esb.oracle.aq`** directory. To use it, deploy it to the **`jbossesb.sar/lib`** directory.

When creating a queue, always select a payload of the type **`SYS AQ$_JMS_MESSAGE`**.

Note

An example **`jboss-esb.xml`** configuration file can be found in the ***`${SOA_ROOT}`*`/samples/quickstarts/helloworld_action/oracle-aq/`** directory.

1.4.6. Tibco Enterprise Message Service

To use Tibco Enterprise Message Service, set the following parameters:

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

Next, check that the client JARs that ship with the **Tibco Enterprise Message Service** can be found in the class-path. (These files 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`**

Note

TibcoEMS versions 4.4.1 and 5.0 have been tested with JBoss Enterprise SOA Platform.

1.4.7. Extension Properties

By default, the JNDI configuration (used to retrieve the Java Message Service resources) is set to inherit all of those properties whose names are prefixed with "java.naming".

Note

Some Java Message Service providers specify different naming prefixes. In order to support these different schemes, Red Hat provides functionality that allows one to specify individual property prefixes for each provider.

To use this functionality, define the "jndi-prefixes" property for the relevant jms-provider element by adding a comma-separated list of the additional prefixes. (The property for extensions is also configured in this same location.)

```
<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>
```

1.5. Database Configuration

The JBoss Enterprise SOA Platform uses a database to persist the ESB `service registry` and `message store`.

Database configuration is handled using the **Database Configuration Script**. Refer to the *SOA Getting Started Guide* for more information. The **Database Configuration Script** is the supported mechanism for database configuration. The remaining content in this section is for reference only.

Note

The SQL scripts used for message store database configuration can be found in the **`PROFILE/deploy/jbossesb.esb/message-store-sql`** directory. The service registry , **jUDDI v3**, uses Hibernate to persist its data and so there are no SQL scripts for jUDDI v3.

1.5.1. Switching Databases Manually

This section explains how to migrate from the default database (**Hypersonic**) to **PostgreSQL**. These steps will be almost identical for any other database.

Warning

This section is for illustrative purposes only. The database configuration should not be done manually. These instructions have been included in this book to show how the the **Database Configuration Script** works and should only be used as a reference. Manual configuration may prevent the **Database Configuration Script** from working later. Contact Red Hat Support for assistance if manually configuring the database.

1. Remove the **`deploy/hsqldb-ds.xml`** file and add the following code to a new file named **`deploy/postgres-ds.xml`**:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <datasources>
     <local-tx-datasource>
       <jndi-name>DefaultDS</jndi-name>
       <connection-url>jdbc:postgresql://host:port/database</connection-url>
       <driver-class>org.postgresql.Driver</driver-class>
       <user-name>username</user-name>
       <password>password</password>
         <metadata>
            <type-mapping>PostgreSQL 8.3</type-mapping>
         </metadata>
       <check-valid-connection-sql>select count(*) from jbm_user</check-valid-connection-
   sql>
     </local-tx-datasource>
   </datasources>
   ```

   This can be modified to suit one's needs with respect to connection parameters and the like. However, the name of the DS must always be **`DefaultDS`**.

2. Replace the contents of **`deploy/jbossesb-registry.sar/juddi-ds.xml`** with the same configuration as that created in the previous step (change the database name if need be.) Ensure that jndi-name (**`juddiDB`**) is kept.

3. Replace the contents of **`deploy/jbossesb.esb/message-store-ds.xml`** with the same configuration that was created in step one (change the database name if need be.) Again, ensure that jndi-name (**`JBossESBDS`**) is kept.

4. Replace the database name in the **`deploy/jbossesb.esb/jbossesb-service.xml`** file's **`message-store-sql`** element with the following:

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

   <server>
      <mbean code="org.jboss.internal.soa.esb.dependencies.DatabaseInitializer"
          name="jboss.esb:service=MessageStoreDatabaseInitializer">
         <attribute name="Datasource">java:/JBossESBDS</attribute>
         <attribute name="ExistsSql">select * from message</attribute>
         <attribute name="SqlFiles">
            message-store-sql/postgresql/create_database.sql
         </attribute>
         <depends>jboss.jca:service=DataSourceBinding,name=JBossESBDS</depends>
      </mbean>
   ```

```
</server>
```

5. Edit the **`jbossesb-registry.sar/META-INF/persistence.xml`** file. The hibernate.dialect property must be set to the type of database that is going to be used as the data source. (It may, for example, be **`org.hibernate.dialect.PostgreSQLDialect`**.)

6. Replace **`deploy/jboss-messaging/hsqldb-persistence-service.xml`** with the correct **`postgres-persistence-service.xml`** file for the version of JBM being used.

   Note that this needs to match the version. It will not work if there is a mismatch. (These files are found in the the JBM distribution's **`src/etc/server/default/deploy`** directory.)

7. Copy the database driver to the server's **`lib`** directory.

8. Start the server.

1.6. Using a JSR-170 Message Store

Multiple message stores can be implemented via the JBoss SOA Platform's plug-in architecture. One option is to replace the default database with a JSR-170 compliant *Java content repository* (JCR). The implementation included is known as **Apache Jackrabbit**

To enable it, add the following property to the "core" section of ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/deploy/jbossesb.sar/jbossesb-properties.xml`**:

```
<property name="org.jboss.soa.esb.persistence.base.plugin.jcr" value=
"org.jboss.internal.soa.esb.persistence.format.jcr.JCRMessageStorePlugin"
/>
```

This adds the JCR plug-in to the list of available message stores. The JCR message store can use an existing repository via JNDI or can create a standalone instance locally on the application server. The following list of properties should be added to the "dbstore" section of **`jbossesb-properties.xml`** to configure repository access:

```
<property name="org.jboss.soa.esb.persistence.jcr.jndi.path" value="jcr"/>
<property name="org.jboss.soa.esb.persistence.jcr.username"
  value="username"/>
<property name="org.jboss.soa.esb.persistence.jcr.password"
  value="password"/>
<property name="org.jboss.soa.esb.persistence.jcr.root.node.path"
  value="JBossESB/MessageStore"/>
```

- jcr.jndi.path - optional path in JNDI where the repository is found. If this property is not specified, a new repository will be created based on ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/deploy/jbossesb.esb/repository.xml`** , and the repository data will be stored in ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/data/repository/`**

- jcr.username - username for getting a repository session

- jcr.password - password for getting a repository session

- jcr.root.node.path - the path relative to the root of the repository where messages will be stored.

To test that the JCR Message Store has been configured properly, add the **`org.jboss.soa.esb.actions.persistence.StoreJCRMessage`** action to an existing service. The action will attempt to store the current message in the JCR store.

1.7. Message Tracing

It is possible to trace any and all Messages sent through the JBoss SOA Platform. This is useful for a number of reasons, including auditing and debugging. To trace messages, they must each have a unique number in their MessageID field, located in the header. (This is referred to in the *Programmers' Guide*.)

All interactions between JBoss SOA components and messages are logged. The log reports contain the messages' header information, enabling them to be correlated across multiple SOA Platform instances. Identify them by looking for the following amongst the output:

```
header: [ To: EPR: PortReference < <wsa:Address ftp://foo.bar/> >,
From: null, ReplyTo: EPR: PortReference < <wsa:Address http://bar.
foo/> >, FaultTo: null, Action: urn:dowork, MessageID: urn:foo/bar
/1234, RelatesTo: null ]
```

One can also enable a *meta-data log filter*, the role of which iss to issue log reports whenever a message is either input to, or output from, a SOA Platform component. This filter, **`org.jboss.internal.soa.esb.message.filter.TraceFilter`**, can be added to the Filter section of the **`JBossESB configuration`** file, in conjunction with any other filters, as it has no effect on the input or output messages.

Whenever a message passes through this filter, one will see the following log report at the `Information Level`:

```
TraceFilter.onOutput ( header: [ To: EPR: PortReference < <wsa:Add
ress ftp://foo.bar/> >, From: null, ReplyTo: EPR: PortReference <
<wsa:Address http://bar.foo/> >, FaultTo: null, Action: urn:dowork
, MessageID: urn:foo/bar/1234, RelatesTo: null ] )
```

```
TraceFilter.onInput ( header: [ To: EPR: PortReference < <wsa:Addr
ess ftp://foo.bar/> >, From: null, ReplyTo: EPR: PortReference < <
wsa:Address http://bar.foo/> >, FaultTo: null, Action: urn:dowork,
MessageID: urn:foo/bar/1234, RelatesTo: null ] )
```

TraceFilter will only log messages if the org.jboss.soa.esb.messagetrace property is set to **`on/ON`**. The default setting is **`off/OFF`**. If enabled it will log every message that passes through it. However, there are facilities for more finely-tuned control over this functionality. To undertake such configuration work, make sure that the property is set to **`on/ON`**. Those Messages for which org.jboss.soa.esb.message.unloggable is set to **`yes/YES`** will now be ignored by the filter.

1.8. Clustering and Fail-Over Support

The **JBoss Service-Oriented Architecture Platform** supports the *fail-over* of *stateless* services. Consult the *Programmers' Guide* for detailed information on this topic but the main points to note are these:

- the `ServiceInvoker` hides much of the fail-over complexity from users but it only works with native ESB Messages and, furthermore, not all gateways have been modified to use take advantage of it. Non-ESB Aware Messages sent to those gateway implementations may not be able to take advantage of service fail-over.

- when the `ServiceInvoker` tries to deliver a message to a service it may potentially be given a choice of multiple *end-point references*. In order to help it determine which one to select, one can configure a *policy*. To do so, set the org.jboss.soa.esb.loadbalancer.policy property in the **`jbossesb-properties.xml`** file. Three policies are provided but custom ones can also be created. The three pre-packaged ones are:

1. first available: if a healthy `service binding` is found it will be used until it dies. The next end-point reference in the list will then be used.

   There is no load balancing between the two service instances with this policy.

2. round robin: a typical load balancing policy whereby each end-point reference is "hit" in list order.

3. random robin: this is like the round robin, but the selection is randomized.

- The end-point reference list used by the the policy may become smaller over time as "dead" EPRs are removed. When the list is exhausted or the time-to-live of the list cache is exceeded, the ServiceInvoker will obtain a fresh list of EPRs from the Registry. The org.jboss.soa.esb.registry.cache.life property defaults to 60000 milliseconds but can be set in the **`jbossesb-properties`** file.

- If none of the end-point references work then use the Message Re-delivery Service.

- To run the same service on more than one node in a cluster, wait until the service registry cache re-validates first. (Configure the cache re-validation time-out in the ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/deploy/jbossesb.sar/jbossesb-properties.xml`** file.)

```
<properties name="core">
  <!-- 60 seconds is the default -->
  <property name="org.jboss.soa.esb.registry.cache.life" value="60000"/>
</properties>
```

- Setting the org.jboss.soa.esb.failure.detect.removeDeadEPR property to **`true`**, means that whenever the `Service Invoker` suspects an end-point reference has failed, it will remove it from the registry. (The default setting is **`false`** because this should be used with extreme care. A service that is simply overloaded and slow to respond could, potentially, have its end-point reference removed from the registry by mistake.) These "orphaned" services will not be subject to any further interactions and may have to be restarted.

1.9. Using OpenSSO

The **JBoss Service-Orientated Architecture Platform** includes the *Open Web Single Sign-On* service software (**OpenSSO**.) Use this to simplify the implementation of a transparent service.

Note

To learn more about **OpenSSO**, please visit the project's website at *http://opensso.dev.java.net*.

1.9.1. Installing and Configuring the OpenSSO in Tomcat

There is an known issue with deploying **OpenSSO** on the JBoss Enterprise SOA Platform but it can be deployed to other web-containers for use with the SOA Platform.

Note

Details of the deployment issue can be found at *https://jira.jboss.org/jira/browse/SOA-731*.

The following instructions teach the reader how to deploy the **OpenSSO** onto **Tomcat**. Information about using the OpenSSO with other web-containers can be found at *https://opensso.dev.java.net/public/use/docs/fampdf/index.html*.

Procedure 1.2. Deploying OpenSSO to Tomcat

1. Download the **Tomcat** software from the Apache Project's website at: *http://tomcat.apache.org*.

   Note

   Red Hat customers can obtain and install **Tomcat** from the software repository, without needing to download it from the Apache Project's website.

2. Extract the archived files into a directory. (The following examples assume that this directory is called **`/opt/tomcat`**.)

3. Edit **`/opt/tomcat/bin/catalina.sh`** (**`catalina.bat`** for Windows deployments) and add *`-Xmx1G`* to the JAVA_OPTS property. This specifies the maximum heap size of the JVM instance as one gigabyte.

   Example 1.2. Adding max size to JAVA_OPTS

   ```
   JAVA_OPTS="$JAVA_OPTS -Xmx1G -
   Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
   ```

4. Now download **`opensso.zip`** (build 4.5) from the **OpenSSO** website: *https://opensso.dev.java.net/public/use/index.html*.

5. Extract the contents of **`opensso.zip`** and copy **`opensso.war`** from **`deployable-war/`** to **`/opt/tomcat/webapps/`**.

6. To deploy **JBoss Enterprise Service-Oriented Architecture Platform** and **Tomcat** on the same machine, alter the **Tomcat** port specified in the **`$tomcat/server.xml`** as per this example:

   Example 1.3. Updating Tomcat port

   ```
   <Connector port="8090" protocol="HTTP/1.1">
   <Connector port="8099" protocol="AJP/1.3" redirectPort="8443"/>
   ```

7. Start **Tomcat** by running the **`/opt/tomcat/bin/startup.sh`** shell script (or the **`startup.bat`** batch file, if one is using a Microsoft Windows deployment.)

8. Open *http://localhost:8090/opensso* in a web browser client.

9. Click on **Create Default Configuration**. This causes OpenSSO to configure itself with its default values.

10. Enter passwords for the default user and agent accounts. The two accounts cannot have the same password.

11. Now you can login to *http://localhost:8090/opensso* with the credentials you supplied in the previous step.

1.9.2. Configuring OpenSSO for the JBoss SOA Platform

The **`AuthContext`** class found in **`openssoclientsdk.jar`** performs the authentication. The following steps describe the configuration required to enable this integration.

Procedure 1.3. Configuring OpenSSO integration

1. Edit ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/conf/login-config.xml`**

   You need to have the following configuration in **`login-config.xml`** to integrate with OpenSSO. The orgName and the moduleName properties must be the same values as are configured in the OpenSSO system. The last property shows where the **`AMConfig.properties`** file is located.

   Example 1.4. Editing **`login-config.xml`** for OpenSSO

   ```
   <application-policy name="OpenSSOLogin">
    <authentication>
      <login-module
          code="org.jboss.soa.security.opensso.OpenSSOLoginModule"
          flag="required">
        <module-option name="orgName">opensso</module-option>
        <module-option name="moduleName">DataStore</module-option>
        <module-option name="amPropertiesFile">
          props/AMConfig.properties
        </module-option>
      </login-module>
    </authentication>
   </application-policy>
   ```

2. Configure ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/conf/props/AMConfig.properties`**

   By default this file contains configuration information for the hostname of **`localhost`**, **`port 8080`** and the context path of **`opensso`**. You can edit the configuration by hand, but we recommended using the supplied setup script, **`setup.sh`** (**`setup.bat`** for Windows deployments).

   The setup script is found in the directory ***`${SOA_ROOT}`*****`/samples/quickstarts/opensso/opensso-sdk/scripts`** . You must run this script from the **`opensso-sdk`** directory to ensure that the classpath will be correct.

   ```
   [opensso-sdk]$ ./scripts/setup.sh
   Debug directory (make sure this directory exists): /var/tmp
   Password of the server application: password
   Protocol of the server: http
   Host name of the server: gatekeeper.company.com
   Port of the server: 8080
   Server's deployment URI: opensso
   Naming URL (hit enter to accept default value, http://gatekeeper.company.com:8080/
   opensso/namingservice):
   [opensso-sdk]$
   ```

   The script will prompt you for the values it needs and create your new **`AMConfig.properties`** file in the **`resources`** directory. You then need to copy this new file to ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/conf/props/`** and overwrite the existing one.

Having successfully undertaken the two steps above, one is now able to use the `OpenSSOLogin` module as a JAAS plug-in provider.

It can also be used as an identity provider, allowing one to secure the Service-Oriented Architecture Platform:

Example 1.5. Using the OpenSSOLogin module as a identity provider

```
<service category="OpenSSO"
    name="SimpleListenerSecured" description="Hello World">
  <security moduleName="OpenSSOLogin" runAs="adminRole"/>
  <listeners>
    <jms-listener name="JMS-Gateway" busidref="quickstartGwChannel"
      maxThreads="1" is-gateway="true"/>
  </listeners>

  <actions mep="OneWay">
    <action name="debug" class="org.jboss.soa.esb.actions.SystemPrintln">
      <property name="printfull" value="false"/>
      <property name="message" value="In Service1"/>
    </action>
  </actions>
</service>
```

Having studied this chapter, you should now be confident to configure the SOA Platform company's needs. The rest of the book explains day-to-day administration.

The Registry

At the heart of all **JBoss Service-Oriented Architecture Platform** deployments lies a *Registry*. This is fully described in the *JBoss SOA Platform Services Guide* (in which ways in which it can be configured are also discussed.) However, it is worth briefly making note of the following:

- When services run, they usually place the *end-point reference* (through which they can be contacted) into the registry. If services have been correctly developed, they should automatically remove these end-point references from the registry when they terminate. However, in certain circumstance, entries will be left in the registry. Some causes of these situations include machine crashes and incorrect programming. These "stale" entries prevent the correct execution of subsequent deployments. If this occurs, these entries can be removed manually but always ensure that the system is in an inactive state before doing so.

- There is an optional feature that makes the Enterprise Service Bus remove all existing service entries from the Registry prior to adding a new instance. To use it, simply set the end-point reference's **`remove-old-service`** tag name to **`true`**.

  Warning

  Note that this option should be used with care, because the entire service will be removed, including all end-point references.

Configuring Web Service Integration

The **JBoss Service-Oriented Architecture Platform** exposes *web service end points* through the *SOAPProcessor* action. This action integrates the **JBoss Webservices** container with the Enterprise Service Bus, allowing one to invoke *JBossWS* end points over any channel supported by the SOA Platform.

Important

JBossWS 2.0.1.SP2 (native) or higher must be installed on one's **JBoss Service-Oriented Architecture Platform** server in order to be able to use the **`SOAPProcessor`** action.

Note

Refer to the *Programmers' Guide* for more information.

Default "ReplyTo" End-Point References

The **JBoss Enterprise SOA Platform** employs end-point references to address messages going to and from services. As described in the *Programmers' Guide*, messages have headers that contain recipient addresses and sequential numbers (the latter being used for the purpose of correlation.) Message headers may also contain further optional addresses for replies, faults and so forth. Because the recommended interaction pattern within the **JBoss Service-Oriented Architecture Platform** is based on a one-way message exchange, messages may not necessarily receive responses automatically: it is dependent on the individual application as to whether or not a sender expects a response.

A reply address is an optional part of the header's routing information, which an application can set if necessary. When a response is required and the `ReplyTo` end-point reference has not been set, the **JBoss Enterprise SOA Platform** will use a default value, specific to each type of transport. Note that some of these `ReplyTo` defaults will only work correctly if the system administrator performs additional configuration work:

- if the Java Messaging Service is being used, the software assumes it to be in the form of a queue with the same name as that which was used to deliver the original request, (which was prefixed with '_reply.')
- if it is the JDBC that is being employed, the software assumes that it is using a table in the same database with the same name as that which was used to deliver the original request, (which was prefixed with '_reply_table.') Also, please bear in mind that the new table needs the same columns as the request table.
- if local and remote files are being used, then no administrative changes are required. Responses are written into the same directory as the request, albeit with a unique suffix in order to ensure that only the original sender will collect the response.

The ServiceBinding Manager

In order to run multiple **JBoss Enterprise SOA Platform** servers on the same machine, use the the *ServiceBinding Manager* to centralize the port configuration for all of the instances. (The Platform includes a sample "bindings" file, **`docs/examples/binding-manager/sample-bindings.xml`**.)

Note

The JBoss Enterprise Application Platform Server documentation contains detailed instructions, that teaches one how to configure the `ServiceBinding Manager`.

Note

If `jboss-messaging` is being used as a Java Messaging Service provider, ensure that the `ServiceBinding Manager` configuration for it matches with the contents of the **`remoting-service.xml`** file.

Monitoring and Management

The JBoss Enterprise SOA Platform provides several ways to monitor and manage the server. Read this chapter to learn about them.

6.1. JMX MBeans

In the `jboss.esb` domain, one will see the following types of `M-Bean`:

deployment=<ESB package name>
: The `Deployments M-Bean` shows the statuses of all deployed ESB packages. It also provides information about their XML configurations.

listener-name=<Listener name>
: This `M-Bean` displays all of the deployed *listeners*. It shows information about their XML configurations, start times, `maxThreads` and states. The system administrator has the option of initializing, starting, stopping or destroying each listener.

category=MessageCounter
: The message counters display all of the services deployed for a `listener`, each service's separate actions and counts of how many messages were processed, as well as the time taken to process each message.

service-name=<Service name>
: This M-Bean displays a variety of statistics for each service, including message counts, state, average size and processing time. The message counts may be reset and services may be started and stopped.

Note

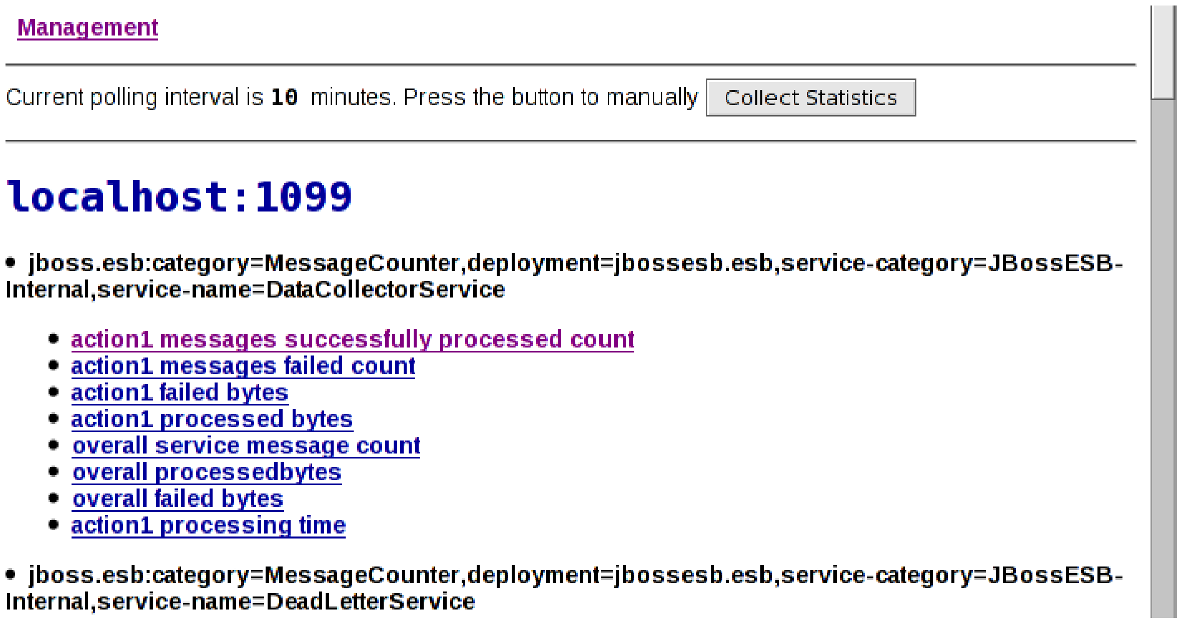
In addition to the M-Beans listed above, the Java Message Service domain provide some that show statistics for message queues. This information can be useful when debugging or analysing performance.

6.2. The Monitoring and Management Console

The JBoss Enterprise SOA Platform includes a **Monitoring and Management Console**. It requests and displays M-Bean information from each node within the Enterprise Service Bus registry. To access the console, load the following address in a web browser: *http://localhost:8080/jbossesb*.

The **Monitoring and Management Console** gathers information on the performance of different deployed ESB services. It records a history of these.

The **Monitoring and Management Console** lets system administrators see message counts by service, action and node. It also displays other information such as processing times, numbers of failed messages, bytes transferred and time-stamps for the last successfully-processed and failed messages.


Management

Current polling interval is 10 minutes. Press the button to manually Collect Statistics

localhost:1099

- jboss.esb:category=MessageCounter,deployment=jbossesb.esb,service-category=JBossESB-Internal,service-name=DataCollectorService
  - action1 messages successfully processed count
  - action1 messages failed count
  - action1 failed bytes
  - action1 processed bytes
  - overall service message count
  - overall processedbytes
  - overall failed bytes
  - action1 processing time
- jboss.esb:category=MessageCounter,deployment=jbossesb.esb,service-category=JBossESB-Internal,service-name=DeadLetterService

Figure 6.1. JBoss Enterprise Service Bus Monitoring and Management Console

The **Monitoring and Management Console** is installed by default in the `All` and `Production` server configurations. It is also easy to install in other configurations if required.

6.2.1. Installing the Enterprise Service Bus Console

Install the **ESB Console** running the command **`ant deploy`** in the ***`${SOA_ROOT}`*****`/tools/console/management-esb/`** directory. The script will determine automatically whether to deploy to the **`production`** or **`default`** configuration by checking for the existence of **`jbossesb.sar`** in each of them. (Preference is given to the **`production`** configuration. If neither is found then deployment will stop with an error message.

To deploy to a different server configuration, the **`build.xml`** must first be edited.

Procedure 6.1. Manually Installing the Console

1. To override the default choice of server configuration, edit the file ***`${SOA_ROOT}`*****`/tools/console/management-esb/build.xml`** by adding the following line:

   ```
   <property name="org.jboss.esb.server.config" value="my_config"/>
   ```

   In the above, **`value`** is the name of the server configuration to which you wish to deploy, it being **`my_config`** in this example. Add this line to the beginning of the file with the other property settings.

2. If **Hypersonic** is not being used as the console's back-end database, one will need to edit the ***`${SOA_ROOT}`*****`/tools/console/management-esb/db.properties`** file to define the correct database settings. (Read *Section 6.2.2, “ Using an Alternative Database ”* to learn more about this.)

3. Go to the ***`${SOA_ROOT}`*****`/tools/console/management-esb`** directory and run **`ant deploy`**

Example 6.1. Deploying the Enterprise Service Bus Console on Linux/Unix

```
$ cd tools/console/management-esb
$ ant deploy
```

6.2.2. Using an Alternative Database

The **Hypersonic Database** is the default "back-end" for the **Management and Monitoring Console**. However, Red Hat neither recommends nor supports the use of **Hypersonic** in production environments. When deploying to live, always change the database to a supported configuration. The process to do so is described in this section.

The ***${SOA_ROOT}*/tools/console/management-esb/db.properties** file defines the database settings for the console. By default this file contains the following line: **db=hsqldb**. This line sets **Hypersonic** as the database to use. To change the database, simply edit this line. The following values are allowed:

- **hsqldb** - **Hypersonic**
- **mysql** - **MySQL**
- **oracle9i** - **Oracle 9i**
- **oracle10g** - **Oracle 10g**

Once the file has been edited, deploy the console as described in *Section 6.2.1, “ Installing the Enterprise Service Bus Console ”*.

Next, add the JDBC driver JAR file into the server's ***${SOA_ROOT}*/server/*${CONFIG}*/lib** directory. (JBoss ships with **hsqldb.jar** in this directory by default.)

For **MySQL** users, there is an addition step: it may also be necessary to create the database statistics before deploying. Please look over the **management-ds.xml** file to see if the database has been listed in the **management-esb/src/main/resources/*${DB}*/** directory.

6.2.3. Collection Periods

The default period of time between data collections, (known as the *polling period*), is ten minutes. This is specified at build time. The property is pollMinuteFrequency and is found in the **management-esb/db.properties** file.

The *current* polling period can be changed at run-time. To do so, use either the **Monitoring and Management Console** or the **jmx-console** to edit the pollMinuteFrequency property, found in the **jboss.esb:service=DataFilerScheduler** M-Bean.

Note

The **Collect Statistics** button at the top of the Console's page allows the system administrator to force an immediate collection of statistics.

6.2.4. Services

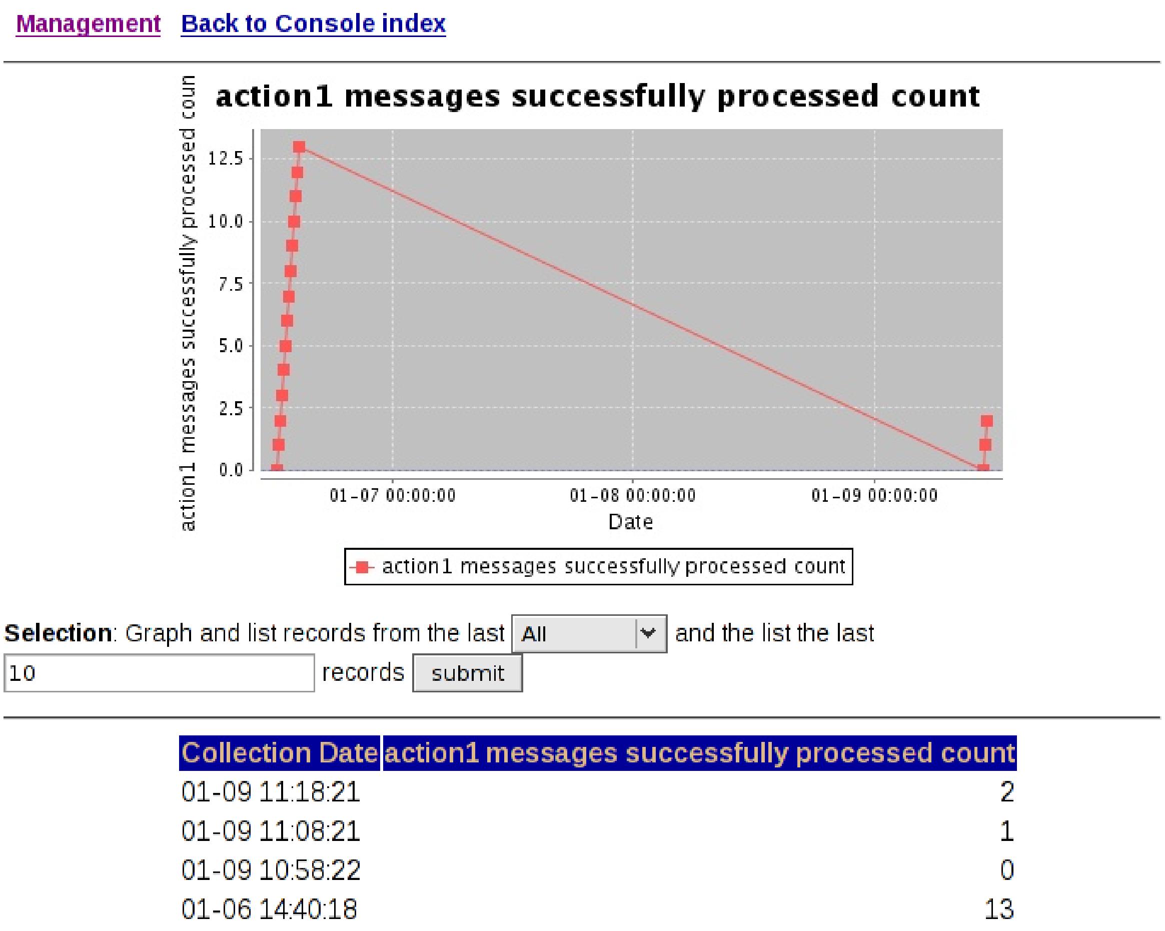
The following information is displayed for each ESB service:

- the processing time per action
- processed count per action
- failed count per action
- overall message count per service

Click on any one of these statistics to be presented with a page charts its history.

By default, the last ten records will be shown. To display more, change the value in the **Display Records** text box or alter the charting period. The available chart periods are:

- the last five minutes
- hour
- day
- week
- month
- all records



Management Back to Console index

action1 messages successfully processed count
action1 messages successfully processed coun
12.5
10.0
7.5
5.0
2.5
0.0
01-07 00:00:00
01-08 00:00:00
01-09 00:00:00
Date
action1 messages successfully processed count

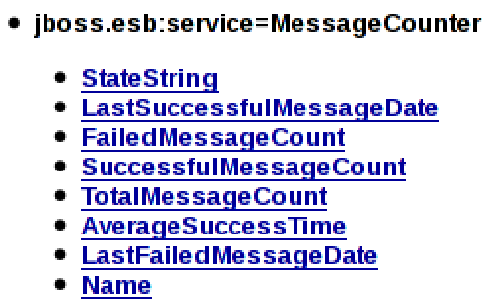
Selection: Graph and list records from the last All and the list the last 10 records submit

Collection Date	action1 messages successfully processed count
01-09 11:18:21	2
01-09 11:08:21	1
01-09 10:58:22	0
01-06 14:40:18	13

Figure 6.2. Monitoring and Management Console Statistical Charting Feature

6.2.5. Message Counter

The **Monitoring and Management Console** also provides an overall count of the messages that pass through the Enterprise Service Bus. The *Message Counter* keeps track of the number of both successful and failed messages, along with their processed bytes and the dates.

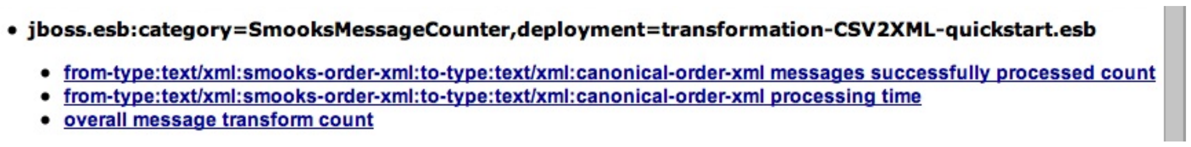


- jboss.esb:service=MessageCounter
  - StateString
  - LastSuccessfulMessageDate
  - FailedMessageCount
  - SuccessfulMessageCount
  - TotalMessageCount
  - AverageSuccessTime
  - LastFailedMessageDate
  - Name

Figure 6.3. Monitoring and Management Console Message Counter

6.2.6. Smooks Transformations

There is an M-Bean that keeps track of the count of "processed" *Smooks Transformations* and the time taken to process each. It also keeps track of the overall count of the *Transformation Chain*.



- jboss.esb:category=SmooksMessageCounter,deployment=transformation-CSV2XML-quickstart.esb
  - from-type:text/xml:smooks-order-xml:to-type:text/xml:canonical-order-xml messages successfully processed count
  - from-type:text/xml:smooks-order-xml:to-type:text/xml:canonical-order-xml processing time
  - overall message transform count

Figure 6.4. Monitoring and Management Console Transformations Functionality

6.2.7. Dead Letter Service

As has been mentioned in the *Programmers' Guide*, the *DeadLetterService* (DLQ) can be used to store those messages that cannot be delivered. This is a JBoss ESB service and can, therefore, be both monitored and inspected. Note, however, that the **`DeadLetterService`** is not used if the underlying transport has native support, as, for example, is the case with the Java Messaging Service. In that case, one must inspect the Dead Letter Service and any transport-specific equivalent.

6.3. Message Alerts

The **JBoss Web Console** is a utility that is available from within both the JBoss Application Server and the JBoss ESB Server. It is capable of monitoring and sending alerts based on `JMX M-Bean` properties. Use this functionality to receive alerts for Enterprise Service Bus-related events, such as when the `Dead Letter Service Counter` has reached a certain threshold.

These are the steps one needs to undertake to configure it:

1. Add your SMTP settings to **`./deploy/mail-service.xml`**.

2. Open **`./deploy/monitoring-service.xml`** and "un-comment" the EmailAlertListener section and add appropriate header-related information.

3. Create a **`./deploy`** file to serve as the `Monitor M-Bean`.

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
 <mbean code="org.jboss.monitor.ThresholdMonitor"
  name="jboss.monitor:service=ESBDLQMonitor">
  <attribute name="MonitorName">
    ESB DeadLetterQueue Monitor
  </attribute>

  <attribute name="ObservedObject">
```

```
   jboss.esb:category=MessageCounter,
deployment=jbossesb.esb,service-name=DeadLetterService
  </attribute>

  <attribute name="ObservedAttribute">
    overall service message count
  </attribute>
  <attribute name="Threshold">4</attribute>
  <attribute name="CompareTo">-1</attribute>
  <attribute name="Period">1000</attribute>
  <attribute name="Enabled">true</attribute>
  <depends-list optional-attribute-name="AlertListeners">
    <depends-list-element>
      jboss.alerts:service=ConsoleAlertListener
    </depends-list-element>
    <depends-list-element>
      jboss.alerts:service=EmailAlertListener
    </depends-list-element>
  </depends-list>
  <depends>jboss.esb:deployment=jbossesb.esb</depends>
 </mbean>
</server>
```

This M-Bean will serve as a monitor and, once the DeadLetterService counter reaches **5**, it will send an e. mail to the address(es) specified in the **`monitoring-service.xml`** file. Note that the alert is only sent once, this being when the threshold has been reached. To be alerted again once the counter has been reset, edit the alerted flag on the Monitoring Service M-Bean (in this case, **`jboss.monitor:service=ESBDLQMonitor`**.)

Note

To learn more about how to use the JBoss Web Console Monitoring functionality, please study *http://www.jboss.org/community/docs/DOC-12659*.

6.4. JON for SOA

The **JBoss Operations Network** (JON) program is an additional tool that system administrators can use to monitor and administer the JBoss SOA Platform servers.

This software allows one to undertake inventorying, administration, monitoring, deployment and updating tasks. It performs these jobs by applying a centrally-managed model equipped with a customizable web-portal interface.

Note

To learn more about the **JBoss Operations Network** project, visit its official website at *http://www.jboss.com/products/jbosson*.

JON for SOA is a stand-alone release of the **JBoss Operations Network** software. It includes additional functionality specifically designed for the JBoss SOA Platform. This section provides an overview of that functionality and assumes that the readership already has a basic knowledge of the JBoss Operations Network product.

Important

In contrast to the various embedded JBoss SOA Platform consoles, access to the *JON Console* is not restricted to the local server. This grants the user with greater flexibility, but also means that the inherent restrictions of the latter cannot be relied upon to ensure the security of the JON Console.

6.4.1. Adding a JBoss SOA Platform Server to the JON Inventory

The JBoss SOA Platform Server will appear in JON as a resource of the type JBossAS Server. The description against it will be "JBoss Enterprise SOA Platform".

Important

An error message will be seen when JON is used to access the SOA Platform Server for the first time. This is simple because it has not yet been provided with the authentication details for a valid SOA Platform user that it needs.

Configure the SOA user information by changing the settings in the ***`${SOA_ROOT}`*****`/server/`*****`${CONFIG}`*****`/conf/props/soa-users.properties`** file. This information is entered as the Principal and Credentials (user-name and password) in the server's **Connection Properties**. (These details are accessed by selecting the server and then the **INVENTORY** tab. The error message described above also contains a shortcut link to the connection properties page.)

6.4.2. JBoss SOA-P Enterprise Service Bus Statistics

Having correctly configured the server to use the **JBoss Operations Network** tool, one should now find the item **JBoss ESB Statistics** available above the **Resources** menu entry.

Click on the **JBoss ESB Statistics** link to display an overview of the Enterprise Service Bus instance. To view all deployed ESB packages, click on the **JBoss ESB Deployment** link.

After that, one can drill-down into each ESB package to view detailed statistics about the components of which the packages consist. The metrics collected and displayed will vary depending on the component being viewed.

JBoss ESB Statistics

Message Counts (Failed)	Last Failed Message Date
Last Successful Message Date	Processed Bytes
Message Count (Successful)	Message Count (Total)

ESB Deployment

Deployment Type

Service

Overall Bytes Failed	Overall Bytes Processed
Overall Bytes	Message Count
Message Count (avg)	Message Count (avg) per Minute

Listener Configuration

Start Date	Maximum Number of Threads

MEP	Service Category
Service Description	Service Name

Actions

Messages Successfully Processed (avg)	Message Count (avg)
Messages Failed	Messages Failed (avg)
Processing Time	Overall Bytes Processed
Message Count	Messages Successfully Processed
Overall Bytes	Overall Bytes Failed
Message Count (avg) per Minute	Messages Failed (avg) per Minute
Messages Successfully Processed (avg) per Minute	

Note

All of the metrics listed above are self-explanatory with two exceptions:

1. **`MEP`** is a string that indicates the Message Exchange Protocol being used, an example being "OneWay."

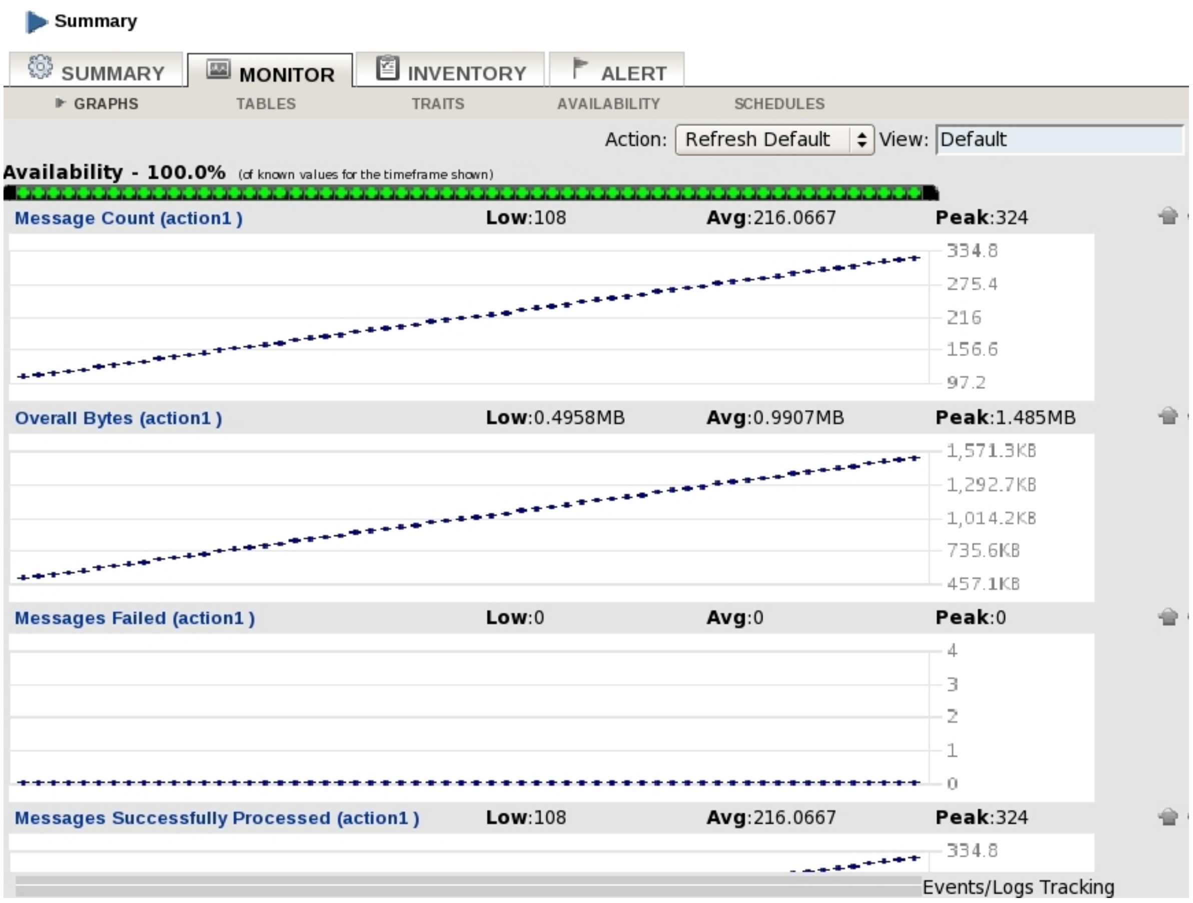
2. **`Deployment Type`** is a string that indicates the the deployed ESB's class, an example being "JBoss4ESBDeployment."

Important

The values displayed in the Enterprise Console's user interface for message count trend upwards over time, irregardless of the data collection interval. For example, if, at 12:00 the user starts the JON server and agent, deploys an ESB archive and, before 13:00, sends 1000 messages through the a service deployed in the ESB archive, then later specifies a data "Metric Collection Schedule" of from 14:00 to 15:00, (and no messages are processed during this time period), the message count metric displayed will still be 1000.

The only way in which the user can make the message counter display a zero value is to reset it.

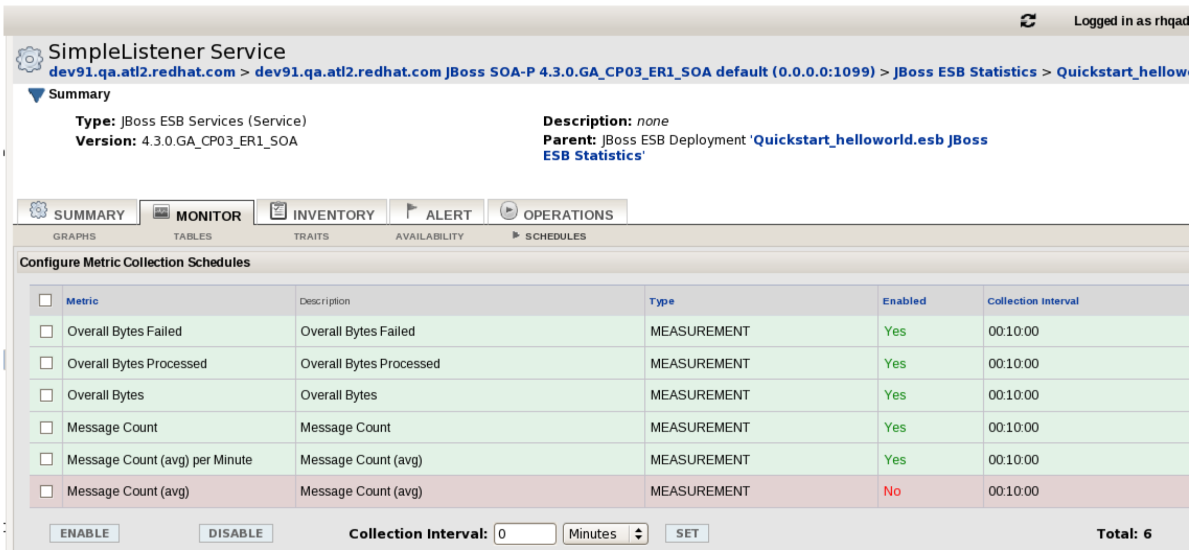
All of the standard **JBoss Operations Network** functionality such as alerts can be configured for any of an Enterprise Service Bus deployment, service or action, by being based on these statistics.



Availability - 100.0% (of known values for the timeframe shown)

	Low	Avg	Peak
Message Count (action1)	108	216.0667	324
Overall Bytes (action1)	0.4958MB	0.9907MB	1.485MB
Messages Failed (action1)	0	0	0
Messages Successfully Processed (action1)	108	216.0667	324

Figure 6.5. Displayed Metrics



SimpleListener Service
dev91.qa.atl2.redhat.com > dev91.qa.atl2.redhat.com JBoss SOA-P 4.3.0.GA_CP03_ER1_SOA default (0.0.0.0:1099) > JBoss ESB Statistics > Quickstart_hellow

Summary

Type: JBoss ESB Services (Service)
Version: 4.3.0.GA_CP03_ER1_SOA

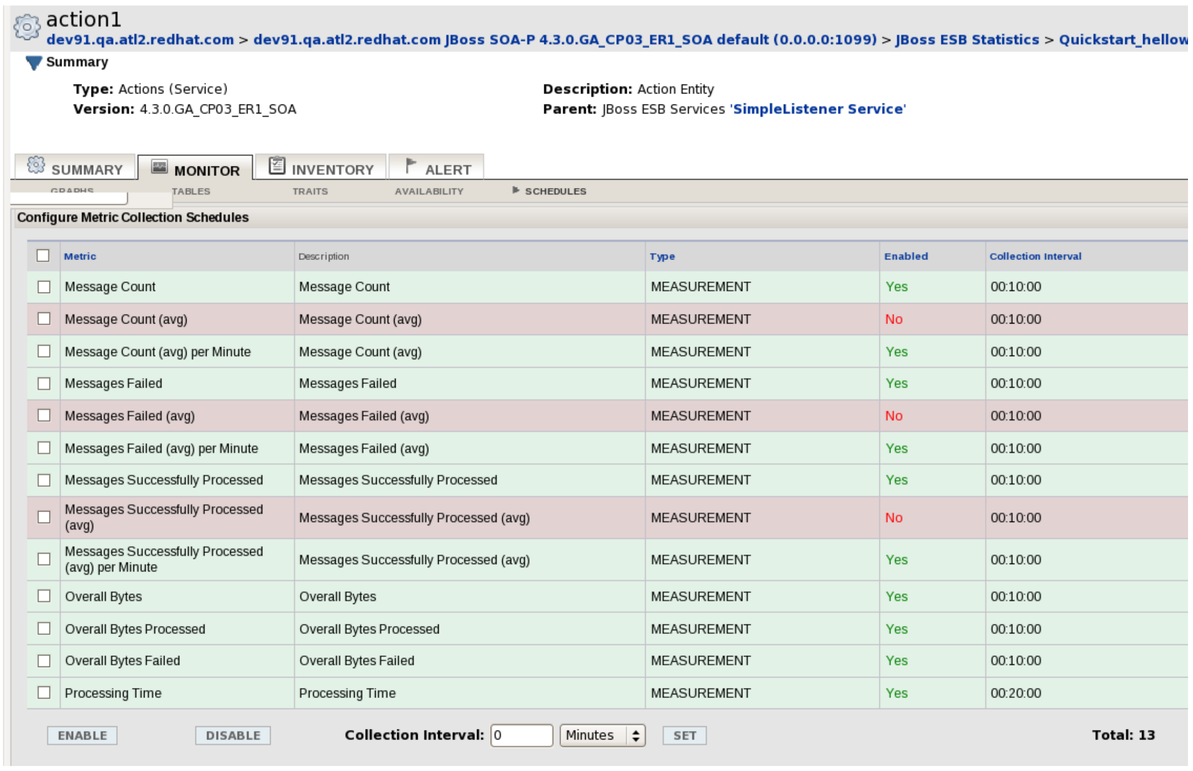
Description: *none*
Parent: JBoss ESB Deployment 'Quickstart_helloworld.esb JBoss ESB Statistics'

Configure Metric Collection Schedules

	Metric	Description	Type	Enabled	Collection Interval
☐	Overall Bytes Failed	Overall Bytes Failed	MEASUREMENT	Yes	00:10:00
☐	Overall Bytes Processed	Overall Bytes Processed	MEASUREMENT	Yes	00:10:00
☐	Overall Bytes	Overall Bytes	MEASUREMENT	Yes	00:10:00
☐	Message Count	Message Count	MEASUREMENT	Yes	00:10:00
☐	Message Count (avg) per Minute	Message Count (avg)	MEASUREMENT	Yes	00:10:00
☐	Message Count (avg)	Message Count (avg)	MEASUREMENT	No	00:10:00

Total: 6

Figure 6.6. Service Metrics Disabled by Default


action1
dev91.qa.atl2.redhat.com > dev91.qa.atl2.redhat.com JBoss SOA-P 4.3.0.GA_CP03_ER1_SOA default (0.0.0.0:1099) > JBoss ESB Statistics > Quickstart_hellow
Summary
SUMMARY
MONITOR
INVENTORY
ALERT
GRAPHS
TABLES
TRAITS
AVAILABILITY
SCHEDULES
ENABLE
DISABLE
Minutes
SET

Type: Actions (Service)
Version: 4.3.0.GA_CP03_ER1_SOA

Description: Action Entity
Parent: JBoss ESB Services 'SimpleListener Service'

Configure Metric Collection Schedules

☐	Metric	Description	Type	Enabled	Collection Interval
☐	Message Count	Message Count	MEASUREMENT	Yes	00:10:00
☐	Message Count (avg)	Message Count (avg)	MEASUREMENT	No	00:10:00
☐	Message Count (avg) per Minute	Message Count (avg)	MEASUREMENT	Yes	00:10:00
☐	Messages Failed	Messages Failed	MEASUREMENT	Yes	00:10:00
☐	Messages Failed (avg)	Messages Failed (avg)	MEASUREMENT	No	00:10:00
☐	Messages Failed (avg) per Minute	Messages Failed (avg)	MEASUREMENT	Yes	00:10:00
☐	Messages Successfully Processed	Messages Successfully Processed	MEASUREMENT	Yes	00:10:00
☐	Messages Successfully Processed (avg)	Messages Successfully Processed (avg)	MEASUREMENT	No	00:10:00
☐	Messages Successfully Processed (avg) per Minute	Messages Successfully Processed (avg)	MEASUREMENT	Yes	00:10:00
☐	Overall Bytes	Overall Bytes	MEASUREMENT	Yes	00:10:00
☐	Overall Bytes Processed	Overall Bytes Processed	MEASUREMENT	Yes	00:10:00
☐	Overall Bytes Failed	Overall Bytes Failed	MEASUREMENT	Yes	00:10:00
☐	Processing Time	Processing Time	MEASUREMENT	Yes	00:20:00

Collection Interval: 0

Total: 13

Figure 6.7. Action Metrics Disabled by Default

Important

Any values displayed in red are disabled by default.

6.4.3. Managing Deployed Enterprise Service Bus Archives

The **JON for SOA** software also allows one to deploy or delete Enterprise Service Bus archives. To access this functionality, go to the **JBoss ESB Statistics** screen, and click on the **INVENTORY** tab. It will then be visible under **Child Resources**.

A new Enterprise Service Bus archive can be deployed by selecting **JBoss ESB Deployment** from the **Create New** menu. Then, from the **Create New Resource** page, specify which archive to deploy and to where it should be sent (which, under normal circumstance, would be one's **`deploy`** directory.) Only compressed files can be uploaded; the **Deploy Zipped** option tells whether it should be deployed as a compressed or an exploded archive.

Existing archives can be deleted. To do so, simply go to the **Child Resources** list, tick their entries and then click **DELETE**.

Note

Historical deploy and delete requests can be viewed in this section as well.

6.4.4. Automatic Service Discovery

The *JBoss Operations Network Agent* will automatically detect Enterprise Service Bus archives that have been deployed or deleted independently of the JON interface. Newly-deployed ESB archives are added to the server inventory automatically but deleted archives are not removed in the same way.

The default agent is configured to only perform this "service discovery" once every twenty-four hours. There are two ways in which to change this time period:

1. by editing the **`conf/agent-configuration.xml`** file. (Restart the agent for the change to take effect.)

   ```
   <entry key="rhq.agent.plugins.service-discovery.period-secs" value="86400"/>
   ```

   Figure 6.8. Service discovery period setting in **`conf/agent-configuration.xml`**

2. by using the **JBoss Operations Network Console** to edit the configuration.

   JBoss Operations Network Agents can be added to the inventory of server resources. Once added, their configurations can be edited like those of any other inventoried resource. Do so by changing the **Service Discovery Period** value under the **CONFIGURE** tab of the applicable resource. (There is no need to restart the Agent in order for the change to take effect.)

In contrast to the SOA Platform's embedded consoles, there is no way to force the **JBoss Operations Network Console** to perform an immediate collection of new data. Buttons such as **Get Current Values** in the **Metric Data** tab only update the display to reflect the most recently collected data. To obtain an immediate update, one can, however, set the collection period to a very low value such as thirty seconds (and then set the interval back to the previous figure afterwards.)

Important

For performance reasons, Red Hat does not recommend lowering the collection period by a significant amount.

Hot Deployment

7.1. Server Mode

The **JBoss Service-Oriented Architecture Platform** supports *hot deployment*. It achieves this by regularly checking the **`deploy`** directory for new files. In addition, it also checks those files that have already been deployed for specific changes. When these changes are detected, the files are removed from deployment by the server and the new files are substituted in their place. This latter case os referred to as *hot re-deployment*.

The specific changes monitored vary by package type.

1. **SAR** files

   The **`jbossesb.sar`** is hot deploy-able. It will re-deploy when:

   - the archive's time-stamp changes, (if the **SAR** file is compressed.)
   - the timestamp of the **`META-INF/jboss-service.xml`** file changes, (if the **SAR** file is in exploded form.)

2. **ESB** files

   Any **`*.esb`** archive will re-deploy when

   - the time-stamp of the archive changes, (if the **ESB** file is compressed.)
   - The time-stamp of the **`META-INF/jboss-esb.xml`** changes, (if the **ESB** is in exploded form.)

   The actions have *life-cycle support*. This means that, upon hot re-deployment, they terminate "gracefully," by finishing active requests. They will not accept any more incoming messages until they have re-started. (All of this occurs automatically.) In order to update just one action, use **Groovy** scripting to modify an action at run-time (see the **Groovy** Quick Start at *http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossESBQuickStart*.)

3. Rule Files

   There are two ways in which to refresh rule files (**DRL** or **DSL** files):

   - by re-deploying the **`jbrules.esb`** archive.
   - by turning on the **`ruleReload`** feature in the *Action Configuration* (see *JBossESBContentBasedRouting* at *http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossESBContentBasedRouting*.)

   After doing so, if a change to a rule file is detected, it will be re-loaded.

4. Transformation Files

   There are also two ways of refreshing *transformation files*:

   - by re-deploying the **ESB** archive in which the transformation file resides.
   - by sending out a notification message over the Java Message Service (`topic`) using the **Enterprise Service Bus Console**. The **Smooks** processors will receive this event and be prompted to re-load.

5. Business Process Definitions

   New versions of the **JBoss Business Process Management Tool**'s *Business Process Definitions* can be deployed to the jBPM database via the **Eclipse** plug-in. Be aware that a new version will only be used by fresh process instances. Existing processes will finish their life-cycles still using the previous definition. For more details, please refer to the *jBPM User Guide*.

7.2. Stand-Alone ("Bootstrap") Mode

When the *Boot-Strapper* mode is run, Enterprise Service Bus archives will not be deployed. There can be only one **`jboss-esb.xml`** configuration file per node. The node monitors the time-stamp on this file and re-reads its contents if a change occurs. In order to updates rules, use the `ruleReload` functionality.

Finally, to update Business Process Definitions, use the same process that is outlined above.

Contract Publishing

If trying to integrate an end-point, one might sometimes be required to provide information about both it and the operations it supports. This commonly occurs when those web service end-points exposed by the `SOAPProcessor` action are utilized (this is described in the *Programmers' Guide*.)

8.1. The Contract Application

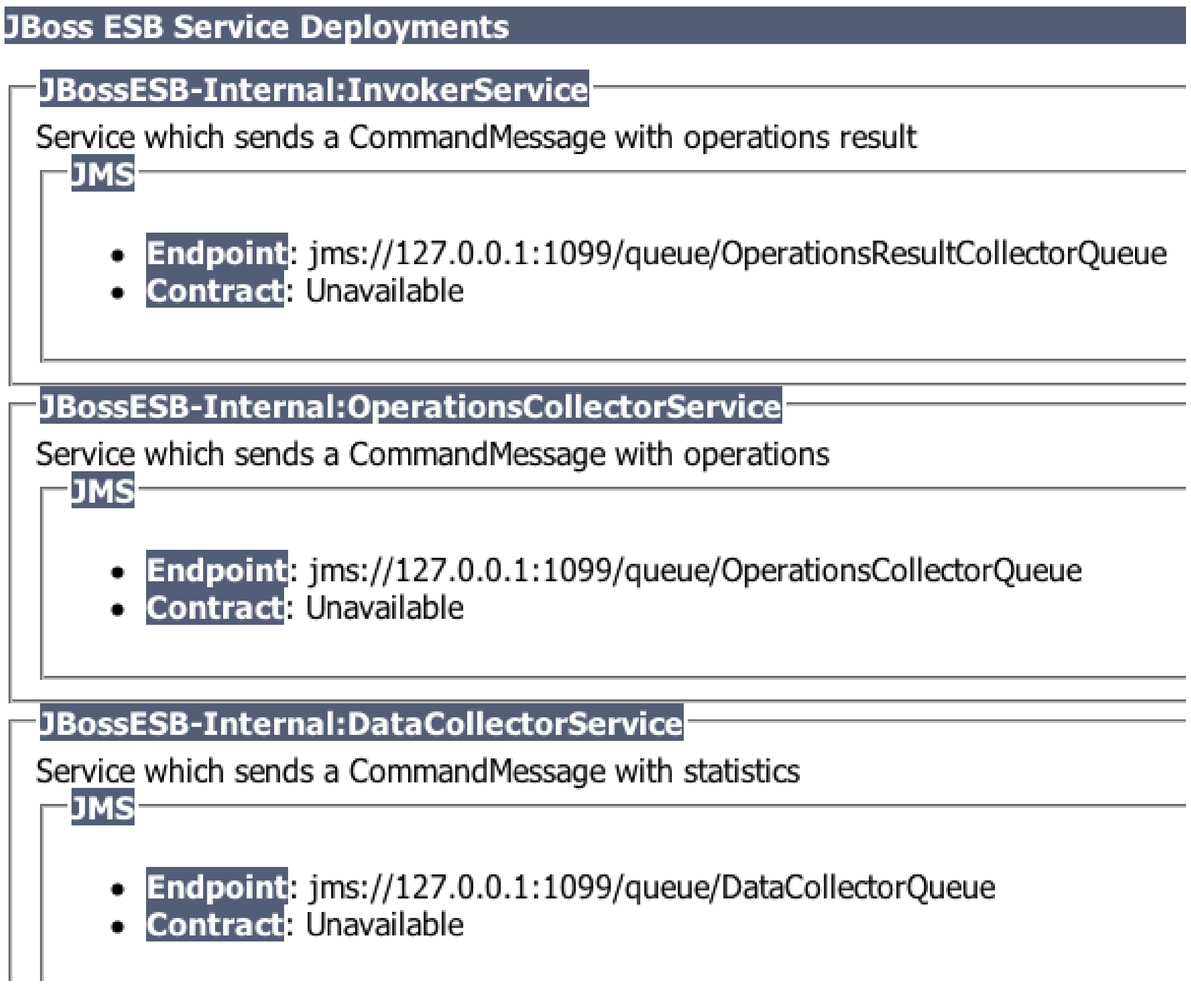
The **Contract Application** is supplied for this reason. Access it via *http://127.0.0.1:8080/contract/*.

Important

This application is only offered as a Technical Preview. Note that it will be superseded in a later release.

Note

The **Contract Application** is also bundled inside the JBoss SOA Console. If the console is to be deployed, first remove the **`contract.war`** file from the SOA Platform server's **`deploy`** directory.



JBoss ESB Service Deployments

JBossESB-Internal:InvokerService

Service which sends a CommandMessage with operations result

JMS

- Endpoint: jms://127.0.0.1:1099/queue/OperationsResultCollectorQueue
- Contract: Unavailable

JBossESB-Internal:OperationsCollectorService

Service which sends a CommandMessage with operations

JMS

- Endpoint: jms://127.0.0.1:1099/queue/OperationsCollectorQueue
- Contract: Unavailable

JBossESB-Internal:DataCollectorService

Service which sends a CommandMessage with statistics

JMS

- Endpoint: jms://127.0.0.1:1099/queue/DataCollectorQueue
- Contract: Unavailable

Figure 8.1. JBoss ESB "Contract" Application

As can be seen, it groups the end-points according to the `services` with which they are associated. This is known as *servicing*.

Another thing to notice is how some of them have an active "Contract" hyperlink. Those visible here are for webservice end-points exposed via the `SOAPProcessor`. These hyperlinks point to the WSDL.

8.2. Publishing a Contract from an Action

The **JBoss Enterprise SOA Platform** "discovers" end-point contracts by using the `action pipeline` that has been configured on the service. Initially, it looks for the first action in the pipeline that publishes contract information. If none of the actions do so, then the **Contract Application** will display the words

```
Unavailable on Contract
```

.

In order to publish contract information, an action receives the **`org.jboss.internal.soa.esb.publish.Publish`** annotation as follows. (This example uses the `SOAPProcessor` for demonstrative purposes):

```
@Publish(JBossWSWebserviceContractPublisher.class)
public class SOAPProcessor extends AbstractActionPipelineProcessor {
    //TODO: implement
}
```

Note

Some example `SOAPProcessor` source code can be found here: *http://anonsvn.jboss.org/repos/labs/labs/jbossesb/trunk/product/services/soap/src/main/java/org/jboss/soa/esb/actions/soap/SOAPProcessor.java*.

Next, implement `org.jboss.soa.esb.actions.soap.ContractPublisher`. (One only needs to implement a single method):

```
public ContractInfo getContractInfo(EPR epr);
```

Note

An example, depicting `JBossWSWebserviceContractPublisher` code can be viewed here: *http://anonsvn.jboss.org/repos/labs/labs/jbossesb/trunk/product/services/soap/src/main/java/org/jboss/soa/esb/actions/soap/JBossWSWebserviceContractPublisher.java*.

JBoss Business Process Manager

9.1. jBPM Console

The *jBPM Web Console* is deployed by default as part of the **`jbpm.esb`** file. View it by loading this address in a web browser: *http://localhost:8080/jbpm-console/*. Please refer to the *jBPM User Guide* for detailed instructions on its use.

9.2. jBPM Message and Scheduler Services

The Business Process Manager's default configuration uses its own **`JobExecutor`** and the database implementations of the `message` and `scheduler` services.

```
<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>
```

The JBoss Enterprise SOA Platform also allows one to use ITS `message` and `scheduler` services for the jBPM in place of those supplied natively by the latter. These additional services are included:

- a JMS-based Message Service that uses JCA in-flow
- a Scheduling Service based on JBoss Messaging.

To use the JMS-based Message Service and the JBoss Messaging-based Scheduler Service, simply replace the default jBPM configuration files found in ***`${SOA_ROOT}`*`/server/production/deploy/jbpm.esb/`** with those located in ***`${SOA_ROOT}`*`/server/production/deploy/jbpm.esb/config/jmsscheduler/`**.

Example 9.1. Replacing the jBPM configuration Using a **BASH** Terminal on Linux

```
$ cd ${SOA_ROOT}/server/production/deploy/jbpm.esb/
$ cp -fb config/jmsscheduler/jbpm.cfg.xml.config jbpm.cfg.xml
$ cp -fb config/jmsscheduler/jbpm-service.xml.config jbpm-service.xml
$ cp -fb config/jmsscheduler/jbm-queue-service.xml.config jbm-queue-service.xml
```

Important

Note that the names of the replacement configuration files have **`.config`** appended to them and, therefore, must be renamed.

Performance Tuning

10.1. Overview

Read this chapter to learn how to optimize the performance of the **JBoss Enterprise Service Bus** for one's specific environment. Before doing any work however, realize that, as with any system, there will be a compromise between performance and reliability. (The default configuration is designed for maximum reliability and stability, which may have an adverse affect on performance in certain circumstances.)

10.2. InVM Transport

An *InVM Transport* (In Virtual Machine) invokes a service by using the `ServiceInvoker` from within the same virtual machine. It has a minimal impact upon resources because it does not incur any networking or message serialisation overhead.

Important

Due to the volatility of the InVM queue, one may not be able to achieve all of the ACID semantics, particularly when this functionality is used in conjuction with other transactional resources, such as databases.

For additional information about this topic, please refer to the "InVM Transport" section of the *Programmers' Guide*.

This code demonstrates how to configure a service using the InVM Transport functionality:

```
<service category="HelloWorld" name="Service1"
                        description="Service 1" invmScope="GLOBAL">
    <listeners>
        <!-- So we just need to define a Gateway to the service... -->
        <jms-listener name="JMS-Gateway" busidref="quickstartGwChannel"
            is-gateway="true"/>
    </listeners>

    <actions>
        <action name="println"
            class="org.jboss.soa.esb.actions.SystemPrintln">
            <property name="message" value=" - > Service 1"/>
        </action>
        <!-- Route to the "Service 2" -->
        <action name="routeAction"
            class="org.jboss.soa.esb.actions.StaticRouter">
            <property name="destinations">
                <route-to service-category="HelloWorld"
                    service-name="Service2"/>
            </property>
        </action>
    </actions>

</service>
```

10.3. Transport Threads

Almost every transport has the ability to configure the number of threads that serve requests. Increasing the number of threads can bring about significant performance gains.

Here are settings for the InVM transport's maxThreads property:

```
<service category="ServiceCategory" name="ServiceName"
description="..." invmScope="GLOBAL">
    <property name="maxThreads" value="100" />
    ...
</service>
```

Here are settings for the JMS Listener's maxThreads property:

```
<jms-listener name="GW" busidref="gwBus" maxThreads="100" is-gateway="true"/>
```

10.4. Message Filters

Message filters are used to dynamically augment messages. For instance, they can be used to add transaction or security information to a message when it flows through the Enterprise Service Bus. Using these filters may impact upon system performance. (This is dependent upon the particular message filters that have been configured in the Enterprise Service Bus.) For further information on this topic, please refer to the "Meta-Data and Filters" section in the *Programmers' Guide*.

Note

For more information about this overall subject, please refer to the "InVM Transport" section in the *Programmers' Guide*.

10.5. "Passing By Reference"

When using the InVM Transport, one can pass messages by value or by *reference*. Passing by reference is the faster of the two ways. However, it might not be suitable in all cases. Consider the circumstances in which it is to be used before implementing it.

```
<service category="ServiceCategory" name="ServiceName"
description="..." invmScope="GLOBAL">
    <property name="inVMPassByValue" value="false" />
    ...
</service>
```

10.6. HTTP Router

The HTTP Router can retain multiple connections. Set the following configuration options in the **`jboss-esb.xml`** file, (which is found in the HttpRouter action):

```
<http-client-property name="max-total-connections" value="100" />
<http-client-property name="max-connections-per-host" value="50" />
```

Note

Similar configuration settings apply to other actions that use the HTTP Router internally, an example being `SOAPProxy`.

10.7. HTTP Connector

The *HTTP Server* is based on **Apache Tomcat** and it has a similar configuration which can be found in the files located in the **`server/*/deploy/jbossweb.sar/server.xml`** sub-directory. Red Hat recommends configuring more threads for a connector:

```
<Connector protocol="HTTP/1.1" port="8080" address="${jboss.bind.address}"
connectionTimeout="20000" redirectPort="8443" maxThreads="200" />
```

Note

See the **Apache Tomcat** documentation for more information about configuring connectors.

10.8. Logging

10.8.1. HTTP Connector

Be aware that the `log4j` **`INFO`** log records have an influence on the measurement of performance. This section describes the ways in which logging can be reduced for this reason.

Logging from **`org.milyn.util.ClassUtil`** produces the following:

```

        ${date/time} INFO [ClassUtil] Loaded ${number} classes from ${number} URLs through
class list file
        ${file name}. Process took ${number}ms. Turn on debug logging for more info.
```

Reduce this level of logging by adding a *category limit* to the **`${jboss-as-home}/server/${configuration}/conf/jboss-log4j.xml`** file:

```
<category name="org.milyn.util.ClassUtil">
  <priority value="WARN" />
</category>
```

Logging from **`org.milyn.delivery.ContentDeliveryConfigBuilder`** produces the following:

```

        ${date/time} INFO  [ContentDeliveryConfigBuilder] All configured XML
        Element Content Handler resource configurations can be applied using the
        SAX or DOM Stream Filter. Defaulting to DOM Filter.
        Set 'global-parameters:stream.filter.type'.  Turn on debug logging for more info.
```

Reduce this level of logging by adding a *category limit* to the **`${jboss-as-home}/server/${configuration}/conf/jboss-log4j.xml`** file:

```
<category name="org.milyn.delivery.ContentDeliveryConfigBuilder">
  <priority value="WARN" />
</category>
```

Logging from **`org.jboss.soa.esb.client.ServiceInvoker`** produces the following:

```

         ${date/time} INFO  [ServiceInvoker] Badly formed EPR
         [EPR: PortReference wsa:Address ${address}] for Service
         [${service category}:${service name}] and Message [
            ${date/time} INFO  [ServiceInvoker] Invalid EPR for service
            (probably ESB-unaware): ignoring for message: header: [  ]
```

Reduce this level of logging by adding a *category limit* to the **`${jboss-as-home}/server/${configuration}/conf/jboss-log4j.xml`** file:

```
<category name="org.jboss.soa.esb.client.ServiceInvoker">
  <priority value="WARN"/>
</category>
```

Appendix A. Revision History

Revision 1.5	**Mon Mar 21 2011**	**David Le Sage** *`dlesage@redhat.com`*

Updated for 4.3.CP05 Release

Revision 1.4	**Tue Apr 27 2010**	**David Le Sage** *`dlesage@redhat.com`*

Updated for SOA 4.3.CP04
SOA-1943 - JON Message Counter Behaviour. Section 6.4

Revision 1.3	**Tue Apr 20 2010**	**David Le Sage** *`dlesage@redhat.com`*

Updated for SOA 4.3.CP03

Revision 1.2	**Wed Sep 1 2009**	**Darrin Mison** *`dmison@redhat.com`*

Updated for SOA 4.3.CP02
SOA-1279 - Updated database details to refer to Schema Tool. Section 1.4
SOA-1310 - Added Performance Tuning Chapter. Chapter 10
SOA-1032 - Updated jBPM Chapter with alternative configuration details. Section 9.1
SOA-1261 - Updated IBM WebSphere MQ JCA Adapter details. Section 1.3.4.2

Revision 1.1	**Tue Feb 24 2009**	**Darrin Mison** *`dmison@redhat.com`*

Updated for SOA 4.3.CP01
Added Clustered ESB Service Configuration
Updated Database Schema Tool instructions
Updated OpenSSO installation
Updated ESB Console installation instructions

Revision 1.0	**Fri Sep 5 2008**	**Darrin Mison** *`dmison@redhat.com`*

Initial Creation