JBoss ESB 4.2 Milestone Release 2

Administration Guide

JBESB-AG-5/8/07

Legal Notices

The information contained in this documentation is subject to change without notice.

JBoss Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. JBoss Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Java™ and J2EE is a U.S. trademark of Sun Microsystems, Inc. Microsoft® and Windows NT® are registered trademarks of Microsoft Corporation. Oracle® is a registered U.S. trademark and Oracle9™, Oracle9 Server™ Oracle9 Enterprise Edition™ are trademarks of Oracle Corporation. Unix is used here as a generic term covering all versions of the UNIX® operating system. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Copyright

Software Version

JBoss ESB 4.2 Milestone Release 2

Restricted Rights Legend

Use, duplication, or disclosure is subject to restrictions as set forth in contract subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause 52.227-FAR14.

Contents

Table of Contents

About This Guide

What This Guide Contains

The Administration Guide contains important information on how to configure and manage installations of JBoss ESB 4.2 Milestone Release 2.

Audience

This guide is most relevant to system administrators who are responsible for managing and deploying JBoss ESB 4.2 Milestone Release 2 installations.

Prerequisites

None.

Organization

This guide contains the following chapters:

- **Chapter 1, Configuration:** How to configure JBossESB and the services it supports.

Documentation Conventions

The following conventions are used in this guide:

Convention	Description
Italic	In paragraph text, italic identifies the titles of documents that are being referenced. When used in conjunction with the Code text described below, italics identify a variable that should be replaced by the user with an actual value.
Bold	Emphasizes items of particular importance.
`Code`	Text that represents programming code.
Function | Function	A path to a function or dialog box within an interface. For example, "Select File | Open." indicates that you should select the Open function from the File menu.
( ) and |	Parentheses enclose optional items in command syntax. The vertical bar separates syntax items in a list of choices. For example, any of the following three items can be entered in this syntax: `persistPolicy (Never | OnTimer | OnUpdate | NoMoreOftenThan)`
Note: **Caution**:	A note highlights important supplemental information. A caution highlights procedures or information that is necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

Table 1 Formatting Conventions

Additional Documentation

In addition to this guide, the following documents are available in the JBoss ESB 4.2 Milestone Release 2 documentation set:

1. **JBoss ESB 4.2 Milestone Release 2** *Trailblazer Guide*: Provides guidance for using the trailblazer example.

2. **JBoss ESB 4.2 Milestone Release 2** *Getting Started Guide*: Provides a quick start reference to configuring and using the ESB.

3. **JBoss ESB 4.2 Milestone Release 2** *Programmers Guide*: How to use JBossESB.

4. **JBoss ESB 4.2 Milestone Release 2** *Release Notes*: Information on the differences between this release and previous releases.

5. **JBoss ESB 4.2 Milestone Release 2** *Services Guides*: Various documents related to the services available with the ESB.

Contacting Us

Questions or comments about JBoss ESB 4.2 Milestone Release 2 should be directed to our support team.

Chapter 1

Configuration

JBossESB JMS Providers

The JBossESB supports a number of JMS providers. Currently we have successfully test JBossMQ, ActiveMQ and Websphere MQ Series (version 5.3 and 6.0). There is no reason that any other JMS provider would not work. We simply did not have time to validate more then these three for now.

How can I configure them?

JMSListeners and JMSGateways can be configured to listen to a Queue or Topic. For this you can use the following parameters in their configuration (jbossesb-listener.xml and jbossesb-gateway.xml): `jndi-URL`, `jndi-context-factory`, `jndi-pkg-prefix`, `connection-factory`, `destination-type` and `destination-name`. Furthermore you will need to add the client jms jars of the JMS-provider you want to use to the classpath.

In the following sections we will assume that your JMS provider runs on 'localhost', that the connection-factory is 'ConnectionFactory', that we are listenening to a destination-type 'queue' and that it's name is 'queue/A'.

> Note: Each JMSListener and JMSGateway can be configured to use it's own JMS provider, so you can use more then one provider in your deployment.

JBossMQ or JBossMessaging

The settings for JBossMQ and JBossMessaging are identical and you should set the parameters to:

```
jndi-URL="localhost”
jndi-context-factory="org.jnp.interfaces.NamingContextFactory"
connection-factory="ConnectionFactory"
destination-type="queue"
destination-name="queue/A"
```

For JBossMQ you should have

jbossmq-client.jar,

In your classpath. Not that this jar is included in jbossall-client.jar, which can be found in lib/ext. For JbossMessaging it should be

jboss-messaging-client.jar

While -for now- the JBossMQ is the default JMS provider in JBossAS, you can also use JbossMessaging. To setup JbossMessaging see the screenshot document on our wiki:

http://wiki.jboss.org/wiki/attach?page=JBossESBDocumentation%2FJBESB-JBM.pdf

ActiveMQ

For ActiveMQ you should set the parameters to:

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

In your classpath you should have

activemq-core-4.x

backport-util-concurrent-2.1.jar

Both jars can be found in lib/ext/jms/activemq. We tested with version 4.1.0-incubator.

Websphere MQ Series

For Websphere MQ Series you should set the parameters to:

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

> Note: Websphere likes all CAPS queue names and no slashes (QUEUEA), and the name of the Queue Manager in MQ should match what the value of 'connection-factory' is (or bind this name to JNDI). In our case we created a Queue Manager named “ConnectionFactory”.

On your classpath you should have

com.ibm.mq.pcf.jar

mqcontext.jar

and the client jars:

com.ibm.mq.jar

com.ibm.mqjms.jar

Please note that the *client* jars differ between MQ 5.3 and MQ 6.0. However the 6.0 jars should be backward compatible. The jars are not open source, and are therefor not provided by us. You will have to obtain them from your WAS and MQ installs.

Also note that you may get the following exception when running MQ 6.0, which can be fixed by adding the user that runs the jbossesb to the mqm group:

Note that for 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

Explanation: There is a problem with user permissions or access.

Tip: Make sure the user accessing MQ Queue Manager is part of the mqm group.

Database Configuration

The ESB uses database for persisting Registry services, and the Message-Store.

Database scripts for each of these can be found under:

Service Registry: ESB_ROOT/install/juddi-registry/sql

Message-Store: ESB_ROOT/install/message-store/sql

A few database types and their scripts are provided, and you should be able to easily create one for your particular database.

For the Message-Store you will need to also update the properties in the main ESB config file jbossesb-properties.xml. The following are settings you will need to change, based on the database and connection information appropriate to your environment – these settings are found in the DBSTORE section of the file.

As long as there is script for your database the ESB will auto-create the schema's on startup.

Property	Setting
org.jboss.soa.esb.persistence.db.connection.url this is the url for your database.	
org.jboss.soa.esb.persistence.db.jdbc.driver	JDBC Driver
org.jboss.soa.esb.persistence.db.user	db user
org.jboss.soa.esb.persistence.db.pwd	db password
org.jboss.soa.esb.persistence.db.pool.initial.size	initial size of db connection pool
org.jboss.soa.esb.persistence.db.pool.min.size	minimum size of db connection pool
org.jboss.soa.esb.persistence.db.pool.max.size	maximum size of db connection pool
org.jboss.soa.esb.persistence.db.pool.test.table	A table name (created dynamically by pool manager) to test for valid connections in the pool
org.jboss.soa.esb.persistence.db.pool.timeout.millis	timeout period to wait for connection requests from pool

The Service Registry database information is contained in the juudi.properties file. You should consult the Service Registry section of this document for more detailed information on what settings and their values and how they effect the behaviour of the ESB.

Addition of ESB tables to JBoss Hypersonic Database

JBoss server comes with a preinstalled hypersonic database (HSQLDB). The database can only be accessed in the same JVM.

The process to create ESB tables in the HSQLDB for simpler installation and use of ESB as a product is as follows.

Note: Use of HSQLDB for production is not recommended.

Steps

Uncomment the following section in hsqldb-ds.xml which is in the JBoss domain.

```
    <mbean code="org.jboss.jdbc.HypersonicDatabase"
      name="jboss:service=Hypersonic">
      <attribute name="Port">1701</attribute>
      <attribute name="BindAddress">${jboss.bind.address}</attribute>
      <attribute name="Silent">true</attribute>
      <attribute name="Database">default</attribute>
      <attribute name="Trace">false</attribute>
      <attribute name="No_system_exit">true</attribute>
    </mbean>
```

This step will make sure that TCP connections can be made to Hypersonic. JBoss uses MBeans to connect to Hypersonic DB.

Start the JBoss Server and make sure it is up and running. This step is to start the HSQLDB database which is part of server startup process.

Create a hsqldb.rc file in the directory product\docs\Install\database

The rc file contains the following parameters

```
urlid JBoss-LocalDB
url jdbc:hsqldb:hsql://localhost:1701
username sa
password
```

The hsqldb.rc file will be used by the Hypersonic SQL Tool to connect to the DB and using the parameters defined in the file.

Note:Run the following java command:

```
java -cp $JBOSS_HOME/server/${DOMAIN}/lib/hsqldb.jar org.hsqldb.util.SqlTool --autoCommit
--continueOnErr --rcfile hsqldb.rc JBoss-LocalDB JBossESB-DB-HSQL.sql
```

The command runs a HSQLDB SQLTool in non interactive mode and uses the urlid called JBoss-LocalDB as defined in the hsqldb.rc file to connect to the DB. It takes as input a SQL file with SQL scripts in it. The SQL file mentioned above in the command is JBossESB-DB-HSQL.sql

JBOSS_HOME – Home directory where JBoss is installed. If JBoss is installed in C:\JBoss then JBOSS_HOME refers to C:\JBoss.

DOMAIN - The domain where hsqldb libraries exist. Mostly default directory.

Using a JSR-170 Message Store

JBossESB allows for multiple message store implementations via a plugin-based architecture. As an alternative to the default database message store, a JSR-170 (Java content repository) message store may be used. The JCR implementation included with JBossESB is Apache Jackrabbit. To enable the JCR message store, add the following property to the "core" section of jbossesb-properties.xml in the root of the jboss-esb.sar:

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

This adds the JCR plugin 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 in 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 not specified, a new repository will be created based on the repository.xml located in the root of jbossesb.sar. In this case, repository data is stored in the JBossAS/server/{servername}/data/repository directory.
- *jcr.username* - username for getting a repository session
- *jcr.password* - password for gettging a repository session
- *jcr.root.node.path* - the path relative to the root of the repository where messages will be stored.

An easy test for whether the JCR message store is configured properly is to add the org.jboss.soa.esb.actions.persistence.StoreJCRMessage action onto an existing service. The action will attempt to store the current message to the JCR store.

Default ReplyTo EPR

JBossESB uses Endpoint References (EPRs) to address messages to/from services. As described in the Programmers Guide, messages have headers that contain recipient addresses, sequence numbers (for message correlation) and optional addresses for replies, faults etc. Because the recommended interaction pattern for within JBossESB is based on one-way message exchange, responses to messages are not necessarily automatic: it is application dependent as to whether or not a sender expects a response.

As such, a reply address (EPR) is an optional part of the header routing information and applications should be setting this value if necessary. However, in the case where a response is required and the reply EPR (ReplyTo EPR) has not been set, JBossESB supports default values for each type of transport. Some of these ReplyTo defaults require system administrators to configure JBossESB correctly.

- For JMS, it is assumed to be a queue with a name based on the one used to deliver the original request: <request queue name>_reply
- For JDBC, it is assumed to be a table in the same database with a name based on the one used to deliver the original request: <request table name>_reply_table. The new table needs the same columns as the request table.
- For files (both local and remote), no administration changes are required: responses will be written into the same directory as the request but with a unique suffix to ensure that only the original sender will pick up the response.

ServiceBinding Manager

If you wish to run multiple ESB servers on the same machine, you may want to use JBoss ServiceBinding Manager. The binding manager allows you to centralize port configuration for all of the instances you will be running. The ESB server ships with a sample bindings file in *docs/examples/binding-manager/sample-bindings.xml*. Chapter Ten of the Jboss appplication server documentation contains instructions on how to set up the ServiceBinding manager. Two notes :

- *esb.juddi.properties* – The *java.naming.provider.url* specified in must match what you configured in your ServiceBinding manager xml for your naming port.
- *remoting-service.xml* – If you are using jboss-messaging as your JMS provider, please note that what you specify in your ServiceBinding manager xml for jboss-messaging configuration must match what is in *remoting-service.xml*.

Index