

JBoss ESB 4.2 GA

Administration Guide

JBESB-AG-8/29/07



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

JBoss, Home of Professional Open Source Copyright 2006, JBoss Inc., and individual contributors as indicated by the @authors tag. All rights reserved.

See the copyright.txt in the distribution for a full listing of individual contributors. This copyrighted material is made available to anyone wishing to use, modify, copy, or redistribute it subject to the terms and conditions of the GNU General Public License, v. 2.0. This program is distributed in the hope that it will be useful, but WITHOUT A WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the GNU General Public License for more details. You should have received a copy of the GNU General Public License, v. 2.0 along with this distribution; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Software Version

JBoss ESB 4.2 GA

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.

© Copyright 2007 JBoss Inc.

Contents

Table of Contents

Contents	iii	Configuring Webservice Integration	17
About This Guide	4	Default ReplyTo EPR	18
What This Guide Contains.....	4	ServiceBinding Manager	19
Audience.....	4	Monitoring and Management	20
Prerequisites.....	4	Hot Deployment	21
Organization.....	4	Server mode.....	21
Documentation Conventions.....	4	1. sar files.....	21
Additional Documentation.....	6	2. esb files.....	21
Contacting Us.....	6	3. rule files.....	21
Configuration	8	4. transformation files.....	21
Standalone server.....	8	5. Business Process Definitions.....	22
JBossESB JMS Providers.....	8	Standalone (bootstrap) mode.....	22
How can I configure them?.....	8	Contract Publishing	23
JBossMQ or JBossMessaging.....	8	Overview.....	23
JBoss Messaging Clustering configuration.....	9	“Contract” Application.....	23
ActiveMQ.....	10	Publishing a Contract from an Action.....	24
Websphere MQ Series.....	11	Index	25
Oracle AQ.....	11		
Database Configuration.....	12		
Using a JSR-170 Message Store.....	14		
Registry	16		

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 GA.

Audience

This guide is most relevant to system administrators who are responsible for managing and deploying JBoss ESB 4.2 GA 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
<i>Italic</i>	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	<p>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:</p> <pre>persistPolicy (Never OnTimer OnUpdate NoMoreOftenThan)</pre>
Note:	A note highlights important supplemental information.
Caution:	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 GA documentation set:

1. **JBoss ESB 4.2 GA Trailblazer Guide:** Provides guidance for using the trailblazer example.
2. **JBoss ESB 4.2 GA Getting Started Guide:** Provides a quick start reference to configuring and using the ESB.
3. **JBoss ESB 4.2 GA Programmers Guide:** How to use JBossESB.
4. **JBoss ESB 4.2 GA Release Notes:** Information on the differences between this release and previous releases.
5. **JBoss ESB 4.2 GA Services Guides:** Various documents related to the services available with the ESB.

Contacting Us

Questions or comments about JBoss ESB 4.2 GA should be directed to our support team.

Configuration

Standalone server

If you wish to run the JBossESB server on the same machine as JBossAS, then you should look at <http://wiki.jboss.org/wiki/Wiki.jsp?page=ConfiguringMultipleJBossInstancesOnOneMachine>

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 listening 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.

When using JMS, JBossESB utilizes a connection pool to improve performance. By default the size of this pool is set to 20, but can be over-riden by setting the `org.jboss.soa.esb.jms.connectionPool` property in the transports section of the JBossESB configuration file. Likewise, if a session cannot be obtained initially, JBossESB will keep retrying for up to 30 seconds before giving up. This time can be configured using the `org.jboss.soa.esb.jms.sessionSleep` property.

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>

JBoss Messaging Clustering configuration

Configuring JBoss Messaging in a clustered setup gives you loadbalancing and failover for JMS.

The file `deploy/jboss-messaging.sar/messaging-service.xml` contains the configuration for a `ServerPeer`. JBoss Messaging is serverless which means that the system does not rely on a central server. Every `ServerPeer` must have a unique id:

```
<mbean code="org.jboss.jms.server.ServerPeer"
  name="jboss.messaging:service=ServerPeer"
  xmbbean-dd="xmdesc/ServerPeer-xmbean.xml">

  <constructor>
    <!-- ServerPeerID -->
    <arg type="int" value="1"/>
  </constructor>
</mbean>
```

Specify a unique integer for each `ServerPeer` in the cluster.

To make destinations distributable one has to set the 'Clustered' attribute to true in the destination definition xml file. Example of defining a clustered destination:

```
<mbean
  code="org.jboss.jms.server.destination.QueueService"
  name="esb.destination:service=Queue,name=quickstart_Request_esb"
  xmbbean-dd="xmdesc/Queue-xmbean.xml">
<depends optional-attribute-name="ServerPeer">
  jboss.messaging:service=ServerPeer
</depends>
<depends>jboss.messaging:service=PostOffice</depends>
<attribute name="Clustered">true</attribute>
</mbean>
```


The jboss-esb.xml configuration must be updated to use a 'ClusteredConnectionFactory':

```
<jms-provider name="JBossMessaging"
  connection-factory="/ClusteredConnectionFactory"
  jndi-context-factory="org.jnp.interfaces.NamingContextFactory"
  jndi-URL="localhost">
```

JBoss Messaging has the ability for queues on one node to pull messages from other nodes when they have exhausted their local messages.

This prevents messages from getting stranded on nodes with slow or no consumers, and balances out message processing across the cluster.

How, if and when messages are pulled from one node to another is determined by the MessagePullPolicy. The default policy never pulls messages from one node to another. Whether you need message redistribution or not depends on your application topology.

To specify that you do want messages to be balanced out across the cluster, update the jboss-messaging.sar/clustered-xxx-persistence-service.xml to uses a 'MessagePullPolicy':

```
<attribute name="MessagePullPolicy">
org.jboss.messaging.core.plugin.postoffice.cluster.DefaultMessagePullPolicy
</attribute>
```

During development it might be convenient to set the attribute 'GroupName' to a unique name for every developer's local machines to avoid creating a cluster containing all developer instances. Note that changing the multicast address and port is also recommended to further separate the clusters. More information about the reason for this can be found here:

<http://wiki.jboss.org/wiki/Wiki.jsp?page=PromiscuousTraffic>

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="QUEUEEA"
```

Note: WebSphere likes all CAPS queue names and no slashes (QUEUEEA), 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 therefore 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.

Oracle AQ

For Oracle AQ you should set the parameters to:

```
connection-factory="QueueConnectionFactory"
```

and use the following 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"/>
```

You may notice the reference to the InitialContext factory. You only need this is if you want to avoid OracelAQ to register its queues with an LDAP. The AqinitialContextFactory references code in a plugin jar that you can find in the plugins/org.jboss.soa.esb.oracle.aq directory. The jar is called org.jboss.soa.esb.oracle.aq-4.2.jar and you will have to deploy it to the jbossesb.sar/lib directory.

Note that when creating a Queue in Oracle AQ make sure to select a payload type of SYS AQ\$_JMS_MESSAGE.

For a sample you can check the samples/quickstarts/helloworld_action/oracle-aq directory for an example jboss-esb.xml configuration file.

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/services/jbossesb/src/main/resources/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 (if you do, please contribute it back to us).

For the Message-Store you will need to also update the data-source setting properties in the main ESB config file jbossesb-properties.xml. The following are settings you will need to change, based on the 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. By default JBossESB is configured to use a JEE DataSource.

```
<properties name="dbstore">
  <property name="org.jboss.soa.esb.persistence.db.conn.manager"
value="org.jboss.soa.esb.persistence.manager.J2eeConnectionManager"/
>

  <!-- this property is only used if using the j2ee connection
manager -->
  <property
name="org.jboss.soa.esb.persistence.db.datasource.name"
value="java:/JBossESBDS"/>
</properties>
```

When running from the standalone bootstrapper use:

```
<properties name="dbstore">
  <!-- connection manager type -->
```

```

<property name="org.jboss.soa.esb.persistence.db.conn.manager"
value="org.jboss.soa.esb.persistence.manager.StandaloneConnectionMan
ager"/>

<property name="org.jboss.soa.esb.persistence.db.conn.manager"

<property name="org.jboss.soa.esb.persistence.db.connection.url"
value="jdbc:hsqldb:hsqldb://localhost:9001/jbossesb"/>

<property name="org.jboss.soa.esb.persistence.db.jdbc.driver"
value="org.hsqldb.jdbcDriver"/>

<property name="org.jboss.soa.esb.persistence.db.user"
value="sa"/>

<property name="org.jboss.soa.esb.persistence.db.pwd"
value=""/>

<property
name="org.jboss.soa.esb.persistence.db.pool.initial.size"
value="2"/>

<property name="org.jboss.soa.esb.persistence.db.pool.min.size"
value="2"/>

<property name="org.jboss.soa.esb.persistence.db.pool.max.size"
value="5"/>

<property name="org.jboss.soa.esb.persistence.db.pool.test.table"
value="pooltest"/>

<property
name="org.jboss.soa.esb.persistence.db.pool.timeout.millis"
value="5000"/>
</properties>

```

Property	Setting
org.jboss.soa.esb.persistence.db.conn.manager	the db connection manager.
org.jboss.soa.esb.persistence.db.datasource.name	The datasource name (used for JNDI lookup)
org.jboss.soa.esb.persistence.db.connection.url	this is the db connection 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

<code>org.jboss.soa.esb.persistence.db.pool.max.size</code>	maximum size of db connection pool
<code>org.jboss.soa.esb.persistence.db.pool.test.table</code>	A table name (created dynamically by pool manager) to test for valid connections in the pool
<code>org.jboss.soa.esb.persistence.db.pool.timeout.millis</code>	timeout period to wait for connection requests from pool

The Service Registry database information is contained in the `juddi.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 behavior of the ESB.

JBoss server comes with a pre-installed hypersonic database (HSQLDB). The database can only be accessed in the same JVM. The data-source definition can be found in the `jbossesb.sar/message-store-ds.xml`.

| Note: Use of HSQLDB for production is not recommended.

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 getting 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.

Registry

At the heart of all JBossESB deployments is the registry. This is fully described elsewhere in the Registry Guide, where configuration information is also discussed. However, it is worth noting the following:

- When services run they typically place the EPR through which they can be contacted within the registry. If they are correctly developed, then services should remove EPRs from the registry when they terminate. However, machine crashes, or incorrectly developed services, may leave stale entries within the registry that prevent the correct execution of subsequent deployments. In that case it is necessary for these entries to be removed manually. However, it is obviously important that you ensure the system is in a quiescent state before doing so.

Configuring Webservice Integration

JBoss ESB 4.2 GA exposes Webservice Endpoints for through the SOAPProcessor action. This action integrates the JBoss Webservices v2.x container into JBossESB, allowing you to invoke JBossWS Endpoints over any channel supported by JBossESB. See the Programmers Guide for more details.

The SOAPProcessor action requires JBossWS 2.0.0GA (native) to be properly installed on your JBoss Application Server (v4.2.0GA).

To install and configure JBossWS 2.0.0, please follow the following steps:

1. Download jbossws-native-2.0.0.GA.zip and unpack in an appropriate location.
2. Edit the ant.properties file and modify jboss42.home to reflect the location of the application server installation.
3. Stop your JBoss Application Server (if running).
4. Execute 'ant deploy-jboss42'.

An additional requirement is to install [JAXB Introductions](#) into your JBossWS 2.0.0GA container. To do this, please follow the following steps:

1. Goto the JBossESB distribution “install” directory.
2. Execute 'ant patch-jbossws'.
3. Restart your JBoss Application Server.

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 application 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*.

Monitoring and Management

There are a number of options for monitoring and managing your ESB server. Shipping with the ESB are a number of useful JMX MBeans that help administrators monitor the performance of their server.

Under the `jboss.esb` domain, you should see the following MBean types :

- **deployment=<ESB package name>** – Deployments show the state of all of the esb packages that have been deployed and give information about their XML configuration and their current state.
- **listener-name=<Listener name>** – All deployed listeners are displayed, with information on their XML configuration, the start time, `maxThreads`, state, etc. The administrator has the option of initialising/starting/stopping/destroying a listener.
- **category=MessageCounter** – Message counters break all of the services deployed for a listener down into their separate actions and give counts of how many messages were processed, as well as the processing time of each message.
- **service=<Service-name>** - Displays statistics per-service (message counts, state, average size of message, processing time, etc). The message counts may be reset and services may be stopped and started.

Additionally, `jms` domain MBeans show statistics for message queues, which is useful information when debugging or determining performance.

Hot Deployment

Server mode

JBossAS as well as the JBossESB-Server are always checking the 'deploy' directory for new files to deploy. So we're really talking about hot redeployment. So here is what you have to do to make it redeploy an existing deployment for the different components.

1. sar files

The jbossesb.sar is hot deployable. It will redeploy when

- the timestamp of the archive changes, if the sar is compressed archive.
- the timestamp of the META-INF/jboss-service.xml changes, if the sar is in exploded form.

2. esb files

Any *.esb archive will redeploy when

- the timestamp of the archive changes, if the sar is compressed archive.
- the timestamp of the META-INF/jboss-esb.xml changes, if the esb is in exploded form.

Our actions have lifecycle support, so upon hot deployment it goes down gracefully, finishes active requests, and does not accept any more incoming messages until it is back up. All of this can be done by simply redeploying the .esb archive. If you want to update just one action, you can use groovy scripting to modify an action at runtime (see the [groovy QuickStart](#)).

3. rule files

There are two options to refresh rule files (drl or dsl)

- redeploy the jbrules.esb (see 2)
- turn on the 'ruleReload' in the action config (see [JBossESBContentBasedRouting](#)). Now if a rule file *changes* it will be reloaded.

4. transformation files

There are two options to refresh transformation files

- redeploy the esb archive in which the transformation file resides.
- send out a notification message over JMS(topic) using the esb-console. The Smooks processors will receive this event and reload.

5. Business Process Definitions

When using jBPM new Business Process Definitions can be deployed. From within the jBBPM eclipse plugin you can deploy a new definition to the jbpm database. New process instances will get the new version, in flight processes will finish their life cycle on the previous definitions. For details please see the documentation on jBPM.

Standalone (bootstrap) mode.

The bootstrapper does not deploy esb archives. You can only have one jboss-esb.xml configuration file per node. I will monitor the timestamp on this file and it will reread the configuration if a change occurs. To update rules you will have to use the 'ruleReload', to update transformation you need to send out a Smooks JMS notification using the esb-console. And finally to update BPDs you can follow the same process mentioned above.

Contract Publishing

Overview

Integrating to certain ESB endpoints may require information about that endpoint and the operations it supports. This is particularly the case for Webservice endpoints exposed via the SOAPProcessor action (see MessageActionGuide).

“Contract” Application

For this purpose, we bundle the “Contract” application with the ESB¹. This application is installed by default with the ESB (after running “ant deploy” from the install directory of the distro)².

It can be accessed through the following URL:

<http://localhost:8080/contract/>

The following is a screenshot of this application.

JBoss ESB Service Deployments

<p>JBossESB-Internal:DataCollectorService Service which sends a CommandMessage with statistics</p> <p>JMS</p> <ul style="list-style-type: none"> Endpoint: jms://localhost/queue/DataCollectorQueue Contract: Unavailable
<p>JBossESB-Internal:DeadLetterService Dead Messages can be send to this service, which is configured to store and/or notify</p> <p>JMS</p> <ul style="list-style-type: none"> Endpoint: jms://localhost/queue/DeadMessageQueue Contract: Unavailable
<p>JBossESB-Internal:RedeliverService Scheduled Service to Redeliver Messages</p>
<p>ABI_OrderManager:ABI_OrderManager ABI OrderManager Service</p> <p>SOCKET</p> <ul style="list-style-type: none"> Endpoint: socket://localhost:8988 Contract: http://localhost:8080/contract/contract.jsp?serviceCat=ABI_OrderManager&serviceName=ABI_OrderManager&protocol=socket <p>HTTP</p> <ul style="list-style-type: none"> Endpoint: http://localhost:8865 Contract: http://localhost:8080/contract/contract.jsp?serviceCat=ABI_OrderManager&serviceName=ABI_OrderManager&protocol=http <p>JMS</p> <ul style="list-style-type: none"> Endpoint: jms://localhost/queue/quickstart_webservice_bpel_esb Contract: http://localhost:8080/contract/contract.jsp?serviceCat=ABI_OrderManager&serviceName=ABI_OrderManager&protocol=jms

As you can see, it groups the endpoint according to Service with which they are associated (servicing). Another thing you'll notice is how some of them have an

¹**NOTE:** This application is only being offered as a Technical Preview. It will be superseded in a later release.

²Note that the Contract application is also bundled inside the JBossESB Console. If you are deploying the console, you will first need to undeploy the default Contract application. Just remove contract.war from the default/deploy folder of your ESB/App Server.

active “Contract” hyperlink. The ones visible here are for Webservice endpoints exposed via the SOAPProcessor. This hyperlink links off to the WSDL.

Publishing a Contract from an Action

JBossESB discovers endpoint contracts based on the action pipeline that's configured on a Service. It looks for the first action in the pipeline that publishes contract information. If none of the actions publish contract information, then the Contract application just displays “Unavailable” on Contract for that endpoint.

An Action publishes contract information by being annotated with the `org.jboss.internal.soa.esb.publish.Publish` annotation as follows (using the SOAPProcessor as an example):

```
@Publish(WebServiceContractPublisher.class)
public class SOAPProcessor extends AbstractActionPipelineProcessor {
    ...
}
```

[See the SOAPProcessor code as an example.](#)

You then need to implement a “ContractPublisher” (*org.jboss.soa.esb.actions.soap.ContractPublisher*), which just requires implementation of a single method:

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

[See the WebserviceContractPublisher code as an example.](#)



Index

Configuring Databases	
Background	10
Configuring JMS	
ActiveMQ	8, 9
JBossMQ	7
Legacy	7
WebSphere MQ	10
