JBoss Enterprise BRMS Platform 5

BRMS Administrator Guide

For JBoss Administrators

Edition 5.3.1

Last Updated: 2017-11-17

JBoss Enterprise BRMS Platform 5 BRMS Administrator Guide

For JBoss Administrators
Edition 5.3.1

Red Hat Content Services

Legal Notice

Abstract

This guide provides the necessary steps for administrators to configure and customize the JBoss Enterprise BRMS Platform.

Table of Contents

PREFACE

CHAPTER 1. INTRODUCTION

1.1. JBOSS ENTERPRISE BUSINESS RULES MANAGEMENT SYSTEM PLATFORM

JBoss Enterprise BRMS Platform is a business rules management system for the management, storage, creation, modification, and deployment of business rules and business processes. Web-based user interfaces and plug-ins for JBoss Developer Studio provide users with different roles the environment suited to their needs. JBoss Enterprise BRMS provides specialized environments for business analysts, rules experts, developers, and rule administrators.

JBoss Enterprise BRMS Platform is supported on a variety of operating systems, Java Virtual Machines (JVMs), and database configurations. A full list of certified and compatible configurations can be found at http://www.redhat.com/resourcelibrary/articles/jboss-enterprise-brms-supported-configurations.

Report a bug

CHAPTER 2. DATABASE CONFIGURATION

2.1. JBOSS ENTERPRISE BRMS AND APACHE JACKRABBIT

The JBoss Enterprise BRMS Platform uses the Content Repository API for Java (JCR) specification to store and track assets. Apache Jackrabbit is the JCR implementation that is included with JBoss BRMS.

The default Apache Jackrabbit repository uses a Derby database that is provided for evaluation purposes only and is not supported for production use. For a full list of supported databases refer to https://access.redhat.com/knowledge/articles/119933.

Report a bug

2.2. SPECIFYING THE REPOSITORY LOCATION

By default, the repository will be created the first time JBoss Enterprise BRMS Platform is run. If a location has not been specified, the repository will be created in the directory the run command is issued from.

The repository should be stored in a secure location that is backed up.

Specify a location for the repository by editing the JBoss Seam **`components.xml`** configuration file.

Procedure 2.1. Specify the Repository's Location

1. Shutdown the Application Server
2. Open the **`components.xml`** file which is located in the **`deploy/jboss-brms.war/WEB-INF/`** directory, and uncomment the repository.root.directory Key-Value attributes.

```
<property name="properties">
   <key>org.drools.repository.configurator</key>

<value>org.drools.repository.jackrabbit.JackrabbitRepositoryConfigur
ator</value>
   <!--  the root directory for the repo storage the directory must
exist. -->
   <!--  <key>repository.root.directory</key>
<value>/opt/yourpath</value>  -->
</property>
```

3. Add the desired location of the repository to the repository.root.directory Key-Value Attribute. (This directory must already exist.)

```
<property name="properties">
   <key>org.drools.repository.configurator</key>

<value>org.drools.repository.jackrabbit.JackrabbitRepositoryConfigur
ator</value>
   <!--  the root directory for the repo storage the directory must
exist. -->
```

```
    <key>repository.root.directory</key>
    <value>/BRMSRulesRepositoryLocation</value>
</property>
```

4. The JBoss Enterprise BRMS Platform will create a new datastore at that location if there is not already one there. To keep an existing datastore, copy the existing files to the new location before restarting the application server.

5. Restart the application server. If an existing datastore was not moved to the new location, a new datastore will be created.

Report a bug

2.3. CONFIGURING AN EXTERNAL RDBMS

JBoss Enterprise BRMS Platform can be configured to use an external RDBMS as the datasource. Use the BRMS Repository Configuration tool to configure the required RDBMS.

Procedure 2.2. Configure an External RDBMS with the Repository Configuration Tool

1. Log on to the JBoss Enterprise BRMS Platform User Interface.

2. From the navigation panel on the left select **Administration** → **Repository Configuration**.

3. Select the RDBMS type from the **`Select RDBMS type:`** drop-down menu.

4. If JNDI is configured, check **`USE JNDI`** and enter the JNDI Name.

5. If JNDI is not configured, enter the required database information:

    - Driver
    - URL
    - User
    - Password

6. Click **`Generate repository config`** to generate the **`repository.xml`** file.

7. Click **`Save Configuration`** to download the generated **`repository.xml`** file and replace the existing **`repository.xml`** in the location specified in the **`components.xml`** file.

Report a bug

2.4. INDEXING FOREIGN KEYS

Some databases, for instance Oracle and Postgres, do not automatically create an index for each foreign key. This can result in deadlocks occurring. To avoid this situation it is necessary to create indexes on some of the columns that are referenced in foreign key constraints.

Indexes should be created on the following columns to avoid possible deadlocks and improve query performance:

Human-Task schema:

- Task.processinstanceid
- task.processid
- task.status
- task.archived
- task.workitem
- i18ntext.language

Core engine schema:

- eventtables.instanceid

Report a bug

2.5. CONFIGURING A DATASOURCE FOR JBOSS ENTERPRISE WEB SERVER

The default configurations use embedded databases that are not suitable or supported for production environments. Before deploying into a production environment, this configuration must be changed to a supported database. JBoss Enterprise Web Server users must configure **`jbpm-human-task.war`** and **`business-central-server.war`**.

The following example uses a MySQL database.

Procedure 2.3. Configuring a datasource for JBoss Enterprise Web Server

1. Copy the MySQL driver to **`tomcat6/lib/`**
2. Configure the datasource resources in **`tomcat6/conf/context.xml`**:

```
<Resource name="jdbc/jbpmDS" auth="Container"
type="javax.sql.DataSource"
factory="bitronix.tm.resource.ResourceObjectFactory"
uniqueName="jdbc/jbpmDS"/>

<Resource name="jdbc/jbpmTasksDS" auth="Container"
type="javax.sql.DataSource"
factory="bitronix.tm.resource.ResourceObjectFactory"
uniqueName="jdbc/jbpmTasksDS"/>
```

3. Edit the **`tomcat6/webapps/business-central-server.war/WEB-INF/Classes/META-INF/persistence.xml`** file to include the unique name from **`context.xml`**:

```
  <jta-data-source>java:/comp/env/jdbc/jbpmDS</jta-data-source>
...
<property name="hibernate.dialect"
```

```
value="org.hibernate.dialect.MySQLDialect"/>
<property name="hibernate.transaction.manager_lookup_class"
value="org.hibernate.transaction.BTMTransactionManagerLookup" />
```

4. Edit the **tomcat6/webapps/jbpm-human-task.war/WEB-INF/classes/META-INF/persistence.xml** file to include the unique name from **context.xml**:

```
<jta-data-source>java:/comp/env/jdbc/jbpmTasksDS</jta-data-source>
...
<property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect"/>
<property name="hibernate.transaction.manager_lookup_class"
value="org.hibernate.transaction.BTMTransactionManagerLookup" />
```

5. JBoss Enterprise Web Server does not include a transaction manager. The following example shows the configuration for Bitronix, and it is provided for evaluation purposes only. Bitronix can be downloaded from http://docs.codehaus.org/display/BTM/Home.

```
 #file resources.properties
 resource.ds1.className=com.mysql.jdbc.jdbc2.optional.MysqlXADataSou
rce
 resource.ds1.uniqueName=jdbc/jbpmDS
 resource.ds1.minPoolSize=0
 resource.ds1.maxPoolSize=10
 resource.ds1.driverProperties.user=guest
 resource.ds1.driverProperties.password=guest
 resource.ds1.driverProperties.URL=jdbc:mysql://localhost:3306/jbpmp
rocess
 resource.ds1.allowLocalTransactions=true

 resource.ds2.className=com.mysql.jdbc.jdbc2.optional.MysqlXADataSou
rce
 resource.ds2.uniqueName=jdbc/jbpmTasksDS
 resource.ds2.minPoolSize=0
 resource.ds2.maxPoolSize=10
 resource.ds2.driverProperties.user=guest
 resource.ds2.driverProperties.password=guest
 resource.ds2.driverProperties.URL=jdbc:mysql://localhost:3306/jbpmt
asks
 resource.ds2.allowLocalTransactions=true
```

6. To configure the Bitronix transaction manager, create a **tomcat6/conf/btm-config.properties** file:

```
bitronix.tm.serverId=tomcat-btm-node0
bitronix.tm.journal.disk.logPart1Filename=${btm.root}/work/btm1.tlog
bitronix.tm.journal.disk.logPart2Filename=${btm.root}/work/btm2.tlog
bitronix.tm.resource.configuration=${btm.root}/conf/resources.proper
ties
```

Add the following line to **tomcat6/conf/context.xml**:

```
<Transaction
factory="bitronix.tm.BitronixUserTransactionObjectFactory"/>
```

-

Add the following line to **tomcat6/conf/server.xml**

```
<Listener
className="bitronix.tm.integration.tomcat55.BTMLifecycleListener"/>
```

Create a configuration properties file. On unix based systems the file should be **tomcat6/setenv.sh**, and on Windows systems the file should be **tomcat6/setenv.bat**.

```
CATALINA_OPTS="-Dbtm.root=$CATALINA_HOME -
Dbitronix.tm.configuration=$CATALINA_HOME/conf/btm-
config.properties"
```

Report a bug

2.6. CONFIGURING A DATASOURCE FOR JBOSS ENTERPRISE APPLICATION PLATFORM 6

The default configuration uses embedded databases that are not suitable or supported for production environments. Before deploying into a production environment this configuration must be changed to a supported database.

For a list of supported database please see http://www.redhat.com/resourcelibrary/articles/jboss-enterprise-brms-supported-configurations

The following procedure uses a PostgreSQL database as an example.

Procedure 2.4. Configuring a Datasource for JBoss Enterprise Platform 6

1. Add a directory for the database to the JBoss Enterprise Application Platform 6 directory structure. The directory should be named for the database that will be used. For instance, for a PostgreSQL database create the following structure:

   jboss-eap-6.0/modules/org/postgresql/

   For a MySQL database, create the following structure:

   jboss-eap-6.0/modules/com/mysql/

2. Download the corresponding JBDC driver for the database.

3. Create a **main** directory in the directory created for the database:

   jboss-eap-6.0/modules/org/postgresql/main/

   Copy the JBDC driver into the **jboss-eap-6.0/modules/org/postgresql/main/** directory.

4. Create a **module.xml** file and save it to **jboss-eap-6.0/modules/org/postgresql/main/** with the following content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <module xmlns="urn:jboss:module:1.0" name="org.postgresql">
   ```

```
 <resources>
  <resource-root path="postgresql-8.4-xxx.jdbc4.jar"/>
 </resources>
 <dependencies><module name="javax.api"/></dependencies>
</module>
```

5. Edit the profile configuration file, for instance, **`jboss-eap-6.0/standalone/configuration/standalone.xml`** configured for a PostgreSQL database called **`jbpmDS`**, and a user called sa with a password sa:

```
<subsystem xmlns="urn:jboss:domain:datasources:1.1">
   <datasources>
      <datasource jta="true" jndi-
name="java:jboss/datasources/jbpmDS" pool-
name="java:jboss/datasources/jbpmDS_Pool" enabled="true" use-java-
context="true" use-ccm="true">
         <connection-url>jdbc:postgresql:mem:test;DB_CLOSE_DELAY=-
1</connection-url>
         <driver-class>org.postgresql.Driver</driver-class>
         <driver>postgresql-jdbc4</driver>
         <pool>
            <min-pool-size>2</min-pool-size>
            <max-pool-size>20</max-pool-size>
            <prefill>true</prefill>
         </pool>
         <security>
            <user-name>sa</user-name>
            <password>sa</password>
         </security>
         <validation>
            <check-valid-connection-sql>SELECT 1</check-valid-
connection-sql>
            <validate-on-match>false</validate-on-match>
            <background-validation>false</background-validation>
            <use-fast-fail>false</use-fast-fail>
         </validation>
      </datasource>
      <drivers>
         <driver name="postgresql-jdbc4" module="org.postgresql"/>
      </drivers>
   </datasources>
</subsystem>
```

6. Edit the **`jboss-eap-6.0/standalone/deployments/business-central-server.war/WEB-INF/classes/META-INF/persistence.xml`** file and change the hibernate dialect to PostgreSQL:

```
<jta-data-source>java:jboss/datasources/jbpmDS</jta-data-source>
.
.
<property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect"/>
```

7. Edit the **jboss-eap-6.0/standalone/deployments/jbpm-human-task.war/WEB-INF/classes/META-INF/persistence.xml** file and change the hibernate dialect to PostgreSQL:

```
<non-jta-data-source>java:jboss/datasources/jbpmDS</non-jta-data-source>
.
.
<property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect"/>
```

Report a bug

2.7. CLUSTERING JBOSS BRMS WITH JACKRABBIT

JBoss Enterprise BRMS can be configured to run in a JBoss Enterprise Application Platform cluster, providing high availability and failover, with each JBoss BRMS node in the cluster accessing the repository via a load balancer proxy which distributes incoming requests between the cluster nodes.

Procedure 2.5. Setting up Clustering with Jackrabbit

1. Set up a JBoss Enterprise Application Platform cluster, refer to the JBoss Enterprise Application Platform 5 *Clustering Guide* which is located in the *Administration and Configuration Guide*.

2. Set up a load balancer, refer to the JBoss Enterprise Application Platform 5 *HTTP Connectors Load Balancing Guide*.

3. Deploy a copy of jboss-brms.war to each node of the application server cluster **$JBOSS_HOME/server/*nodeX*/deploy/**

4. The Default configuration uses embedded databases that are not suitable or supported for production environments. Before deploying into a production environment this configuration must be changed to a supported database. Please refer to the other sections in this chapter for configuration instructions.

5. Each node in the cluster must have an individual workspace, version, and search index configured in the **repository.xml** file. Generate the **repository.xml** for the required database by logging in to the JBoss Enterprise BRMS user interface and selecting **Administration** → **Repository Configuration**.

6. Copy the **repository.xml** file to the repository location specified in the **jboss-brms.war/WEB-INF/components.xml** in each node of the cluster.

7. Each node in the cluster must have a unique ID, which is set in each node's copy of the **repository.xml** file. For instance:

```
<Cluster id="01" syncDelay="2000">
   <Journal
class="org.apache.jackrabbit.core.journal.DatabaseJournal">
      <param name="revision" value="${rep.home}/revision.log" />
      <param name="driver" value="javax.naming.InitialContext" />
    <param name="url" value="brms-jdbc-ds" />
```

```
    <param name="schema" value="mysql"/>
   </Journal>
</Cluster>
```

8. Add the **<distributable/>** element as a child element of the **web-app** element to the **jboss-brms.war/WEB-INF/web.xml** file.

9. Verify the Cluster Nodes are working by logging into the JBoss Enterprise BRMS user interface. Examine the terminal output to confirm which node is handling the session. Shut down that node, after a short interval another node in the cluster should take over the session.

For further information about clustering with Jackrabbit, refer to http://wiki.apache.org/jackrabbit/Clustering

Report a bug

2.8. USING MODESHAPE AS THE JCR

ModeShape is a Java Content Repository (JCR) that can be used in place of Apache Jackrabbit and has been included as a technical preview in JBoss Enterprise BRMS 5.2 and 5.3.

The following requirements must be met before ModeShape can be installed:

- The BRMS standalone package has been downloaded and installed, refer to the *JBoss Enterprise BRMS Getting Started Guide* for further instructions.
- Apache Ant is installed. Refer to the appendix at the back of this guide for directions.
- A supported database server is required for production deployment. Refer to https://access.redhat.com/knowledge/articles/119933 for a complete list of supported database servers.
  - The database instance that will be used has already been created.
  - A database user with permissions to change the database exists.
  - The JDBC driver JAR file for the database is in the **lib/** directory of the server configuration.

Procedure 2.6. Configure ModeShape

1. Run the ant installer from the ModeShape directory and specify the server profile ModeShape will be installed to.

```
[localhost modeshape]$ ant
Buildfile: /opt/BRMS/5.3.1/brms-standalone-5.3.1/modeshape/build.xml
determine_home:
set-web-home:
set-as-home:
set-soa-home:
init:
     [echo] JBoss Home is ../jboss-as-web
prompt:
    [input] Enter profile to install ModeShape to:  [default]
```

2. Add the user accounts **admin** and **mailman**, and assign the roles specified below. The default configuration uses text files to store the usernames, passwords, and assigned roles. The directions here assume that the authorization configuration has not been changed. If it has been changed, then add the users according to that configuration.

   a. Open the file ***PROFILE*/conf/props/brms-users.properties** in a text editor and add the users **admin** and **mailman** to this file as a new line with the syntax: **username=password**.

   ```
   admin=s3kr3t5
   mailman=53cur3m@1l
   ```

   b. Assign the roles specified below to the two users by opening the ***PROFILE*/conf/props/brms-roles.properties** file in a text editor and adding a new line for each user with the syntax **username=role1, role2, role3**.

   The **admin** user must be assigned the roles of **JBossAdmin**, **HttpInvoker**, **user**, and **admin**.

   The **mailman** user must be assigned the roles of **JBossAdmin**, **readwrite**.

   ```
   admin=JBossAdmin, HttpInvoker, user, admin
   mailman=JBossAdmin, readwrite
   ```

   c. Optionally, the passwords can be stored in encoded form instead of plain text. To do this run the following command in the application server directory (**jboss-as** or **jboss-as-web**) to create the encoded form of the password, replacing *PASSWORD* with the chosen password.

   ```
   [localhost]$ java -cp client/jboss-logging-
   spi.jar:lib/jbosssx.jar
   org.jboss.resource.security.SecureIdentityLoginModule PASSWORD
   Encoded password: 5f78dc15b9a559cbdf8592078de921bc
   ```

   Then replace the passwords stored in ***PROFILE*/conf/props/brms-users.properties** with the encoded versions.

3. The default configuration for Modeshape uses a database that is not suitable or supported in production environments. Before deploying into a production environment, this configuration must be changed to a supported database server.

   The default Modeshape repository configuration uses a JNDI data source — **ModeShapeBRMSRepo** — for data storage which is configured in the ***PROFILE*/deploy/modeshape-brms-store-ds.xml** file. Edit this file to change the data source configuration from Hypersonic to the database of your choice. This file is a standard JBoss data source configuration file.

   Refer to the Datasource Configuration chapter of the Application server documentation for details to configure JBoss data sources.

4. Edit the **jboss-brms.war/WEB-INF/components.xml** file to remove the Apache Jackrabbit configuration and add the ModeShape configuration.

   To remove the Apache Jackrabbit configuration, comment out the following section:

```
<property name="properties">
<key>org.drools.repository.configurator</key>
<value>org.drools.repository.jackrabbit.JackrabbitRepositoryConfigur
ator</value>
<!--  the root directory for the repo storage the directory must
exist. -->
<!--  <key>repository.root.directory</key>
<value>/opt/yourpath</value>  -->
</property>
```

The ModeShape configuration is commented out in the default configuration. Uncomment the configuration as shown below:

```
<property name="properties">
<key>org.drools.repository.configurator</key>
<value>org.drools.repository.modeshape.ModeShapeRepositoryConfigurat
or</value>
<key>org.modeshape.jcr.URL</key><value>jndi:jcr/local?
repositoryName=jcrrepo</value>
<key>org.drools.repository.secure.passwords</key><value>false</value
>
<key>org.drools.repository.admin.password</key><value>admin</value>
<key>org.drools.repository.mailman.password</key><value>mailman</val
ue>
</property>
```

5. ModeShape requires two user accounts to exist, **admin** and **mailman**. The passwords for these accounts must be specified in the ModeShape configuration from the previous step.

```
<key>org.drools.repository.secure.passwords</key><value>false</value
>
<key>org.drools.repository.admin.password</key><value>password</valu
e>
<key>org.drools.repository.mailman.password</key><value>password</va
lue>
```

6. Restart the server.

Report a bug

2.9. SEARCHING AND INDEXING

Searching and Indexing is provided by *Apache Lucene* (http://lucene.apache.org/).

By default, the search index is kept on the local filesystem. This occurs because it provides faster performance. Red Hat does not recommend changing this default setting unless you have a specific requirement to do so.

To configure the location of the search index, modify the **repository.xml** file's **<SearchIndex>** element:

```
<SearchIndex class="org.apache.jackrabbit.core.query.lucene.SearchIndex">
 <param name="path" value="${wsp.home}/index"/>
```

```
  <param name="extractorPoolSize" value="2"/>
  <param name="supportHighlighting" value="true"/>
</SearchIndex>
```

Report a bug

CHAPTER 3. DATA MANAGEMENT

3.1. DATA BACKUPS

JBoss Enterprise BRMS does not provide a backup solution. Database backups should be performed using the tools provided by the database vendor.

When restoring a backup, Red Hat recommends clearing the database indexes first so that they are recreated and optimized for the new data.

Report a bug

3.2. IMPORTING AND EXPORTING DATA

3.2.1. Migrating Data

JBoss Enterprise BRMS provides a JCR-standard export and import feature for use when migrating from one database to another. Exported repositories are written to an XML file as defined by the JCR standard.

WARNING

The Import/Export feature is not intended to be used as a backup solution and is not supported by Red Hat as a backup solution. Always use the backup system provided by the database vendor.

The following points should be considered when performing an import or export operation:

- When importing, all existing content in the database is removed.
- Export or import operations should only be conducted when the database is not in use.
- Performance is extremely dependent on both the size of the database and the memory available to the server. Importing is a very memory-intensive process.
- Repository version history is not exported and rules will have their Creation Date attribute re-set to the date on which they are imported.

Report a bug

3.2.2. Exporting the Repository

Procedure 3.1. Exporting the Repository

1. From the navigation panel on the left hand side of the BRMS User Interface select **Administration** → **Import Export**.

2. Click **Export**.

The browser will download a zip file which contains the repository in an XML file.

Depending on the size of the repository, this operation can take some time.

Report a bug

3.2.3. Importing the Repository

Importing a repository will replace any content in the repository with the contents of the XML file being imported.

Procedure 3.2. Importing the Repository

1. From the navigation panel select **Administration** → **Import Export**.
2. Select the XML file containing the exported data by clicking the **Browse** button and browsing the local filesystem.
3. Click **Import** and select **OK** on the confirmation dialog.

After successfully importing the repository the browser will refresh to show the new content of the repository.

To rebuild the package binaries after importing a repository, select **Knowledge bases** → **Create New** → **Rebuild all package binaries**. This may be necessary if errors are occurring as a result of importing a repository from an older version of JBoss Enterprise BRMS.

Report a bug

CHAPTER 4. SECURITY

4.1. AUTHENTICATION

4.1.1. Authentication

The JBoss Enterprise BRMS Platform uses the *Java Authentication and Authorization Service* for verifying user credentials. This service is supplied by the application server and is used to access a separate authentication system. The separate system could be a Lightweight Directory Access Protocol (LDAP), Active Directory server, or JDBC database.

Report a bug

4.1.2. Configuring Authentication

Configure which authentication method is to be used via the **jboss-brms.war/WEB-INF/components.xml** file. The default configuration has many "commented out" options but the actual settings look like this:

```
<security:identity authenticate-method="#{authenticator.authenticate}"
jaas-config-name="jmx-console"/>
<component name="org.jboss.seam.security.roleBasedPermissionResolver">
   <property name="enableRoleBasedAuthorization">false</property>
</component>
```

NOTE

In JBoss BRMS 5.1 and earlier versions, the components.xml file looks like this:

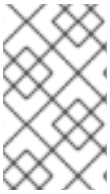
```
<security:identity authenticate-method="#
{authenticator.authenticate}" jaas-config-name="jmx-console"/>
<security:role-based-permission-resolver enable-role-based-
authorization="false"/>
```

IMPORTANT

This default configuration uses the account names, passwords, and roles that are defined in the **jmx-console** authentication policy. Red Hat recommends editing this policy to tailor it for your specific environment.

To configure authentication, follow these steps:

1. Edit the appropriate JBoss login module of the application server.
2. Configure the JBoss Enterprise BRMS Platform to use that module.



NOTE

Many JBoss login modules provide a means of specifying one or more roles for each user. The JBoss Enterprise BRMS Platform has its own mechanism for managing user roles.

WARNING

If *role-based authorization* is disabled, all users effectively have the admin role. This gives them complete access to the JBoss Enterprise BRMS Platform.

Report a bug

4.1.3. Password Configuration for JAAS

When using the default JAAS authentication system, usernames and passwords need to be synchronized between JBoss Enterprise BRMS, the Process Designer, and the Business Central console. If the same usernames and passwords are not used, the different components will not function together.

If the additional users are added to the **`brms-users.properties`** file, they also need to be synchronized for the Process Designer and Business Central Console.

Procedure 4.1. Synchronizing Usernames and Passwords

1. Process Designer: To edit the usernames and passwords for the Process Designer, which is a separate application integrated with JBoss Enterprise BRMS, open the **`designer.war/profiles/jbpm.xml`** file and edit the **`usr`** and **`pwd`** properties:

   ```
   usr="admin" pwd="admin"
   ```

2. Business Central Console. To edit the usernames and passwords for the Business Central Console, open the **`business-central-server.war/WEB-INF/classes/jbpm.console.properties`** file and edit the **`guvnor.usr`** and **`guvnor.pwd`** properties:

   ```
   guvnor.usr=admin
   guvnor.pwd=admin
   ```

Report a bug

4.1.4. Example Authentication: UserRolesLoginModule

This example illustrates the use of the **`org.jboss.security.auth.spi.UsersRolesLoginModule`** login module to access a set of user accounts stored in the **`props/brms-users.properties`** and **`props/brms-roles.properties`**

files.

Procedure 4.2. Authentication Example: `UserRolesLoginModule`

1. **Ensure the Authentication System is Configured Correctly**
   This login module uses two files to store the login name, password, and roles assigned to each user. Create the **`brms-users.properties`** and **`brms-roles.properties`** files in the **`jboss-as-web/server/PROFILE/conf/props/`** directory and then specify at least one user in **`brms-users.properties`** using this format: **`username=password`**. (the **`brms-roles.properties`** file can be left empty.)

2. **Shut Down**
   Shut down the application server before making these changes.

3. **Configure the JBoss Login Module**
   To configure the JBoss Login Modules, open **`jboss-as-web/server/PROFILE/conf/login-config.xml`** in a text editor. It is an XML file containing a **`<policy>`** element with several **`<application-policy>`** child elements. Each **`<application-policy>`** element defines a different authentication scheme. Add the following **`<application-policy>`** XML snippet as a new child of the **`<policy>`** element:

   ```
   <!--BRMS Platform Security Domain-->
   <application-policy name="brms">
      <authentication>
          <login-module
              code="org.jboss.security.auth.spi.UsersRolesLoginModule"
              flag="required">
               <module-option name="usersProperties">
                   props/brms-users.properties
               </module-option>
               <module-option name="rolesProperties">
                   props/brms-roles.properties
               </module-option>
           </login-module>
       </authentication>
   </application-policy>
   ```

4. **Configure the BRMS Platform to use the Login Module**
   Open the **`jboss-as-web/server/PROFILE/deploy/JBoss-BRMS.war/WEB-INF/components.xml`** file. It contains one **`<components>`** element with several child elements, including **`<security:identity>`** .

   Comment out the existing **`<security:identity>`** elements to prevent conflicts. Add the following **`<security:identity>`** element:

   ```
   <security:identity authenticate-
   method="#{authenticator.authenticate}" jaas-config-name="brms"/>
   ```

   The jaas-config-name property must be the same as the application-policy. If the application-policy property was changed in the previous step, modify the jaas-config-name property here to match.

5. **Restart**
   Restart the application server.

Report a bug

4.1.5. Example Authentication: LDAP

LDAP is a popular choice for larger enterprises. The basic configuration steps are the same as the previous example although the details of the configuration will differ.

Procedure 4.3. Authentication Example Two: LDAP

1. **Ensure the LDAP Server is Configured Correctly**
   Check that firewall and network configuration settings are not preventing communication between the application server and the LDAP server.

2. **Shut Down**
   Shut down the application server before making these changes.

3. **Configure the JBoss Login Module**
   To configure the JBoss Login Modules, open **`jboss-as-web/server/PROFILE/conf/login-config.xml`** in a text editor. It is an XML file containing a **`<policy>`** element with several **`<application-policy>`** child elements. Each **`<application-policy>`** element defines a different authentication scheme. Add the following **`<application-policy>`** XML snippet as a new child of the **`<policy>`** element:

```
<application-policy name="brms">
 <authentication>
  <login-module
code="org.jboss.security.auth.spi.LdapExtLoginModule"
      flag="required" >
    <module-option name="java.naming.provider.url">
        ldap://ldap.company.com:389
    </module-option>
    <module-option name="bindDN">DEPARTMENT\someadmin</module-
option>
    <module-option name="bindCredential">password</module-option>
    <module-option name="baseCtxDN">cn=Users,dc=company,dc=com
    </module-option>
    <module-option name="baseFilter">(sAMAccountName={0})</module-
option>
    <module-option name="rolesCtxDN">cn=Users,dc=company,dc=com
    </module-option>
    <module-option name="roleFilter">(sAMAccountName={0})</module-
option>
    <module-option name="roleAttributeID">memberOf</module-option>
    <module-option name="roleAttributeIsDN">true</module-option>
    <module-option name="roleNameAttributeID">cn</module-option>
    <module-option name="roleRecursion">-1</module-option>
    <module-option name="searchScope">ONELEVEL_SCOPE</module-option>
  </login-module>
 </authentication>
</application-policy>
```

   Update the values in this configuration file with those appropriate for your LDAP server.

4. **Configure the BRMS Platform to use the Login Module**

Open the **jboss-as-web/server/*PROFILE*/deploy/jboss-brms.war/WEB-INF/components.xml** file. It contains one **<components>** element with several child elements, including **<security:identity>** .

Comment out the existing **<security:identity>** elements to prevent conflicts. Add the following **<security:identity>** element:

```
<security:identity authenticate-method="#
{authenticator.authenticate}" jaas-config-name="brms"/>
```

The jaas-config-name property must be the same as the application-policy. If the application-policy property was changed in the previous step, modify the jaas-config-name property here to match.

5. **Restart**
   Restart the application server.

Report a bug

4.2. MANAGING USERS

4.2.1. Role-Based Permissions

JBoss Enterprise BRMS uses role-based authorization to assign permissions to users. By default, role-based authorization is disabled and must be enabled (see the *JBoss BRMS Getting Started Guide* for details). If role-based authorization is not enabled, all users have full administrative privileges.

The roles that can be granted are admin, analyst, and package permissions.

Admin Permissions

Users who have been assigned the admin role have complete access to all areas of the JBoss Enterprise BRMS Platform and can manage other user's permissions.

Analyst Permissions

Analyst permissions are intended for the users responsible for maintaining rules. Both developers and business analysts should be granted analyst permissions. Analysts permissions are associated with categories. This makes it possible to manage which rules users are granted analyst permissions for.

In addition to the analyst permissions, it is also possible to grant users Analyst read-only permissions. This allows the user to view the rules in a category, but it does not allow users to edit or change them.

Package Permissions

There are three types of package permissions:

Package Administrator

The package administrator permission grants full control over the specified package, including the right to deploy it. The Package Administrator permission does not grant any administrative rights to any other part of the JBoss Enterprise BRMS Platform.

Package Developer

Package developer permits users to create and edit items within the specified package. This includes being able to create and run tests but does not include the right to deploy the package.

Package Read-only

The package read-only permission grants read-only permissions to a package allowing a user to view the package.

JBoss Enterprise BRMS does not manage the identity of users. Only users who have been assigned as BRMS users will be visible in the user interface, and these users must be added manually.

Report a bug

4.2.2. Adding a New User

Procedure 4.4. Add a New User to BRMS

1. Select **Administration** from the navigation panel, then select **User Permissions**.
2. Add user mapping by clicking on the **Create new user mapping** button. Type the username into the dialogue box that appears, and then click **OK**.

NOTE

The username that is specified for the role must match a username in the authentication service or it will not work.

Report a bug

4.2.3. Managing User Permissions

Procedure 4.5. Manage User Permissions

1. Select **Administration** from the **navigation pane**, then select **User Permissions**.
2. Click **Open** next to the username.
3. Assign user permissions by clicking the plus icon and selecting the appropriate permissions from the **Permission type** drop-down menu. Click **OK** to confirm.
4. **Remove User Permissions**
   Remove user permissions by clicking the minus icon next to the permission being removed, and click OK to confirm.

Users assigned the admin role will be able to modify the roles and permissions of others.

Report a bug

4.3. RULE PACKAGE SIGNING

4.3.1. Rule Package Signing

Rule packages are not signed by default; however, Red Hat recommends enabling rule package signing in production environments. Signing rules packages ensures that the packages cannot be corrupted or tampered with during download from the JBoss Enterprise BRMS Platform server to client applications.

Rule packaging signing is implemented with public key cryptography. Packages are signed with a private key that can only be verified with the matching certificate. The private key is stored in the keystore and is used by the server to automatically sign each package. The public certificate is made available to the client applications in a keystore referred to as a truststore. The certificate in the truststore is used to verify the authenticity of the signed packages. Rule packages that are corrupted or modified during download will be rejected by the client because they no longer match the certificate.

Report a bug

4.3.2. Configuring the Server for Rule Package Signing

Before configuring the server for rule package signing with this process, it is necessary to do the following:

- Create a private signing key and a corresponding public digital certificate.
- Make the private signing key and the digital certificate available to the server in keystores.
- Configure the server to use the keystores.

Procedure 4.6. Configure Rule Package Signing

1. Use the **keytool** command to create the private keystore:

   keytool -genkey -alias *ALIAS* -keyalg RSA -keystore *PRIVATE.keystore*

   The **-alias** parameter specifies the name used to link the related entities in the keystore. Use the same alias for each of these steps. The alias is not case-sensitive. The **-keystore** parameter supplies the name of the file which will be created to hold the private key.

   keytool will prompt you for identifying information as well as two passwords. The first password, the *keystore password*, secures the keystore. The second password, the key password, secures the key that is being created.

   ```
   [localhost ]$ keytool -genkey -alias BRMSKey -keyalg RSA -keystore
   PrivateBRMS.keystore
   Enter keystore password:
   Re-enter new password:
   What is your first and last name?
      [Unknown]:  John Smith
   What is the name of your organizational unit?
      [Unknown]:  Accounts
   What is the name of your organization?
      [Unknown]:  ACME INC
   What is the name of your City or Locality?
      [Unknown]:  Capital City
   ```

```
What is the name of your State or Province?
   [Unknown]:  CC
What is the two-letter country code for this unit?
   [Unknown]:  US
Is CN=John Smith, OU=Accounts, O=ACME INC, L=Capital City, ST=CC,
C=US correct?
   [no]:  yes
Enter key password for <BRMSKey>
   (RETURN if same as keystore password):
Re-enter new password:
```

2. Use the **keytool** command to create a digital certificate:

   keytool -export -alias *ALIAS* -file *CERTIFICATE.crt* -keystore *PRIVATE.keystore*

   Use the same alias and keystore as the previous step. The **-file** parameter is the filename of the new certificate that will be created. The **-keystore** parameter supplies the filename of the private keystore.

   Enter the keystore password at the prompt.

```
[localhost ]$ keytool -export -alias BRMSKey -file BRMSKey.crt -
keystore PrivateBRMS.keystore
Enter keystore password:
Certificate stored in file <BRMSKey.crt>
```

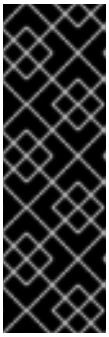
3. Use the **keytool** command to import the digital certificate into a keystore:

   keytool -import -alias *ALIAS* -file *CERTIFICATE.crt* -keystore *PUBLIC.keystore*

   This will create a new keystore, the truststore, which contains the digital certificate. The truststore makes the digital certificate available to client applications.

```
[localhost ]$ keytool -import -alias BRMSKey -file BRMSKey.crt -
keystore PublicBRMS.keystore
Enter keystore password:
Re-enter new password:
Owner: CN=John Smith, OU=Accounts, O=ACME INC, L=Capital City,
ST=CC, C=US
Issuer: CN=John Smith, OU=Accounts, O=ACME INC, L=Capital City,
ST=CC, C=US
Serial number: 4ca0021b
Valid from: Sun Sep 26 22:31:55 EDT 2010 until: Sat Dec 25 21:31:55
EST 2010
Certificate fingerprints:
   MD5:  31:1D:1B:98:59:CC:0E:3C:3F:57:01:C2:FE:F2:6D:C9
   SHA1: 4C:26:52:CA:0A:92:CC:7A:86:04:50:53:80:94:2A:4F:82:6F:53:AD
   Signature algorithm name: SHA1withRSA
   Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

4. The private keystore needs to be kept in a secure location where only the JBoss Enterprise BRMS Platform server is able to access it. This could be on the same machine or in a secured network location that is available to that machine.



IMPORTANT

The JBoss Enterprise BRMS Platform is not able to supply authentication credentials to network resources. If the private keystore is stored in a secure network location, then any authentication procedures must be performed on the behalf of the JBoss Enterprise BRMS server to make the private keystore available to it. For example, the operating system can authenticate and mount a file share that holds the private keystore as a local directory for the JBoss Enterprise BRMS Platform server to access.

5. The truststore needs to be accessible to client applications. This can be done by putting the truststore on network share or hosting it on a webserver.

6. The Drools serialization system properties need to be set on the server. These are the properties that store the information required to access the keystores. Because JBoss Enterprise BRMS Platform also contains client components, both the private keystore and truststore properties have to be set on the server. The properties only need to be set in one location and will be available to all applications running on the same application server instance regardless of where they are set.

   Set the serialization properties by editing the **`preferences.properties`** file, which is located in **`server/`*`profile`*`/deploy/jboss-brms.war/WEB-INF/`** to include the following properties:

   ```
   drools.serialization.sign=true
   drools.serialization.private.keyStoreURL=file:///opt/secure/PrivateB
   RMS.keystore
   drools.serialization.private.keyStorePwd=storepassgoeshere
   drools.serialization.private.keyAlias=BRMSKey
   drools.serialization.private.keyPwd=keypassgoeshere
   drools.serialization.public.keyStoreURL=file:///opt/public/PublicBRM
   S.keystore
   drools.serialization.public.keyStorePwd=keypassgoeshere
   ```

7. The keystore password is currently stored in plain text.

   Refer to https://access.redhat.com/kb/docs/DOC-47247 for instructions to mask the keystore credentials.

Report a bug

4.3.3. Configuring the Client for Rule Package Signing

The procedure below describes the process of configuring the client applications for for Rule Package Signing. This involves setting several properties on the client. The properties can be set programatically using the **`System.setProperty`** method. The class **`org.drools.core.util.KeyStoreHelper`** class contains several constants that represent these properties.

Before performing this task, you require the following:

- A JBoss Enterprise BRMS Platform Server already installed and correctly configured for Rule Package Signing.

- The URL for the truststore that contains the Digital Certificate used by the JBoss Enterprise BRMS Platform Server.
- The password for the truststore, if one is set.

Procedure 4.7. Client Configuration for Rule Package Signing

1. Enable signing by setting the **`drools.serialization.sign`** property to **`true`**.

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

2. Set the **`drools.serialization.public.keyStoreURL`** property to the URL where the TrustStore is located. If the TrustStore is in the classpath of the client then this can be done using the **`getClass().getResource()`** method.

Example 4.1. When the TrustStore is located on the client's classpath

```
URL trustStoreURL = getClass().getResource(
"BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

Example 4.2. When the TrustStore is located on a webserver

```
URL trustStoreURL = new
URL("http://brms.intranet/resources/BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

Example 4.3. When the TrustStore is located on the local file system

```
URL trustStoreURL = new URL("file:///mnt/fileserve/rules-
server/BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

3. Set the **`drools.serialization.public.keyStorePwd`** property to the password for the truststore. This is only required if a password is required to access the truststore.

```
System.setProperty( KeyStoreHelper.PROP_PUB_KS_PWD,
"sekretPasswordHere" );
```

Report a bug

CHAPTER 5. LOGGING

5.1. LOGGING

JBoss Enterprise BRMS Platform features logging functionality is provided by **log4j**.

The configuration included in the **WAR**, **WEB-INF/classes/log4j.xml**, sends all messages to **STDOUT**. When deployed to the **production** server profile, the log messages are appended to the server log file, ***PROFILE*/log/server.log**. When deployed to the **default** server profile, these messages are displayed on the server console.

Report a bug

5.2. CONFIGURING LOGGING

To change the logging behavior, edit the **log4j** configuration file for the relevant server profile: **jboss-as-web/server/*PROFILE*/conf/jboss-log4j.xml**.

Add the following XML to the **default** server profile's **log4j** configuration to create a new category that directs the STDOUT messages to the server log file:

```
<category name="STDOUT" additivity="false">
   <priority value="INFO" />
   <appender-ref ref="FILE"/>
</category>
```

Report a bug

5.3. REMOVING EXTRA LOG MESSAGES

By default, the JackRabbit JCR generates the following INFO log messages:

```
INFO  [TransientRepository] Session closed
```

The JBoss Enterprise BRMS Platform 5.3.1 standalone package is configured by default not to display these messages; however, users who have installed the JBoss Enterprise BRMS 5.3.1 deployable packages to an existing application server, must configure the server not to display these messages.

JBoss Enterprise Application Platform 5 and JBoss Enterprise SOA 5 users can remove these messages by adding the following XML snippet to **jboss-as/server/*profile*/conf/jboss-log4j.xml**:

```
<!-- Limit the verbose JackRabbit TransientRepository -->
  <category name="org.apache.jackrabbit.core.TransientRepository">
     <priority value="WARN"/>
  </category>
```

Report a bug

CHAPTER 6. CUSTOMIZATION

6.1. AVAILABLE LANGUAGES

The JBoss Enterprise BRMS Platform's web user interface can be viewed in multiple languages:

- United States English (**en-US**)
- Japanese (**ja-JP**)
- Simplified Chinese (**zh-CN**)

If a language is not specified, US English is used by default.

Report a bug

6.2. CHANGING THE USER INTERFACE LANGUAGE

Procedure 6.1. Change the User Interface Language

1. Edit the **index.jsp** file, which is located in the **jboss-brms.war** directory, to specify an alternative locale. The default **index.jsp** file uses United States English as the language.

   Example 6.1. The default index.jsp

   ```
   <%
    String redirectURL = "org.drools.guvnor.Guvnor/Guvnor.html";
    response.sendRedirect(redirectURL);
   %>
   ```

   To specify Japanese as the default language, edit the file as follows:

   Example 6.2. index.jsp with Japanese as the default language

   ```
   <%
    String redirectURL = "org.drools.guvnor.Guvnor/Guvnor.html?
   locale=ja_JP";
    response.sendRedirect(redirectURL);
   %>
   ```

   To specify Simplified Chinese as the default language, edit the file as follows:

   Example 6.3. index.jsp with Simplified Chinese as the default language

   ```
   <%
    String redirectURL = "org.drools.guvnor.Guvnor/Guvnor.html?
   locale=zh_CN";
   ```

```
 response.sendRedirect(redirectURL);
%>
```

2. **Edit preferences.properties**
   Edit the **`preferences.properties`** file to specify the required default country and default language codes. The **`preferences.properties`** file is located in the **`jboss-brms.war/WEB-INF/classes/`** directory. The default **`preferences.properties`** file uses United States English as the language and the United States as the default country.

   Example 6.4. preferences.properties with US defaults

   ```
   drools.defaultlanguage=en
   drools.defaultcountry=US
   ```

   To specify Japanese as the default language, edit the file as follows:

   Example 6.5. preferences.properties with Japanese defaults

   ```
   drools.defaultlanguage=ja
   drools.defaultcountry=JP
   ```

   To specify Simplified Chinese as the default language, edit the file as follows:

   Example 6.6. preferences.properties with Simplified Chinese Defaults

   ```
   drools.defaultlanguage=zh
   drools.defaultcountry=CH
   ```

3. **Change the Date Format**
   If multiple instances of the JBoss BRMS user interface are deployed with different languages specified, the format of the date must be changed so that each instance can correctly interpret the date. The following procedure can only be performed on new installations of JBoss Enterprise BRMS that do not include existing rules or test scenarios that include date values, as changing the format will corrupt existing date values.

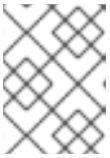
   Specify the date format in **`preferences.properties`**.

   Change the date format from:

   ```
   drools.dateformat=dd-MMM-yyyy
   ```

   To:

   ```
   drools.dateformat=yyyy-MM-dd
   ```



NOTE

The change in format is only effective after the JBoss Enterprise BRMS server has been restarted.

An alternative language can be specified on a per session basis by manually entering the URL with the locale parameter included.

Complete URLs for the Supported Languages

Japanese

http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/Guvnor.html?locale=ja_JP

Simplified Chinese

http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/Guvnor.html?locale=zh_CN

US English

http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/Guvnor.html?locale=en_US

Report a bug

6.3. RUNNING THE JVM WITH UTF-8 ENCODING

JBoss Enterprise BRMS is designed to work with UTF-8 encoding. If a different encoding system is being used by the JVM, unexpected errors might occur.

To ensure UTF-8 is used by the JVM, use the JVM option "-Dfile.encoding=UTF-8".

Report a bug

6.4. CUSTOMIZING THE USER INTERFACE

The BRMS user interface is produced dynamically by the *GWT framework*. The user interface's appearance can be customized for branding or integration purposes by editing the images and CSS stylesheets.

The **`.css`** files and some of the images are found in the **`jboss-brms.war/org.drools.guvnor.Guvnor/`** directory. (The remainder of the images are located in the **`images`** sub-directory.) To access them, the WAR file must be deployed as an exploded archive as described in the *BRMS Getting Started Guide*.

Edit or replace the images and **CSS** files and leave the filenames unchanged. If problems are encountered, restore the original versions of the files from the **WAR** archive.

NOTE

Red Hat recommends adding any modified files to a version control system for easy maintenance.

The change most commonly undertaken is to replace the branding images, these being the logo at the top of the screen and the "site favorites" icon (**hdrlogo_brms.gif** and **drools.gif** respectively.)

The **Guvnor.css** controls the overall styling of the page elements.

WARNING

The GWT components use several additional **CSS** files. Do not change these.

Customize the URLs used by the BRMS Platform by editing the deployment descriptor, namely **jboss-brms.war/WEB-INF/web.xml**. Use the same process as for any other Java web application.

Report a bug

CHAPTER 7. MONITORING

7.1. MONITORING KNOWLEDGE BASES AND KNOWLEDGE SESSIONS

JBoss Enterprise BRMS Platform knowledge bases and knowledge sessions can be monitored with JBoss Operations Network. JBoss Operations Network and the JBoss Operation Network plug-in for JBoss BRMS are available for download from https://access.redhat.com.

Please refer to the *JBoss Operations Network Installation Guide* for installation instructions for the JBoss ON server and the plugin for JBoss Enterprise BRMS.

In order for JBoss Operations Network to monitor knowledge bases and knowledge sessions, MBeans must be enabled.

MBeans can be enabled either by passing the parameter:

```
-Ddrools.mbeans = enabled
```

Or via the API:

```
KnowledgeBaseConfiguration conf =
KnowledgeBaseFactory.newKnowledgeBaseConfiguration();
conf.setOption( MBeansOption.ENABLED );
```

Report a bug

7.2. THE DROOLS KNOWLEDGE BASE MONITORING SERVICE

The Drools knowledge base monitoring service can configure the knowledge base by providing the knowledge base ID as a connection property.

The Drools knowledge base monitoring service provides the following operations:

- Start all internal MBeans
- Stop all internal MBeans

Report a bug

7.3. THE DROOLS KNOWLEDGE SESSION MONITORING SERVICE

The Drools knowledge sessions monitoring service can configure the knowledge base and knowledge session by providing the knowledge base ID and knowledge session ID as connection properties.

The Drools knowledge session monitoring service provides the following operations:

- Reset all metrics counters.
- Return statistics for a specific rules.
- Get statistics for a specific process.

- Get statistics for a specific process instance.

The Drools knowledge session monitoring service provides the following metrics:

- The total number of facts in working memory.
- The total number of activations created since the last reset.
- The total number of activations fired since the last reset.
- The total number of activations canceled since the last reset.
- The total time spent firing rules since the last reset.
- The total number of process instances started since the last reset.
- The total number of process instances completed since the last reset.
- The timestamp of the last reset operation.

Report a bug

APPENDIX A. APACHE ANT

A.1. INSTALLING APACHE ANT

The Java build tool *Apache Ant* and the Trax plugin is a required installation of JBoss ModeShape. It is not required for the installation or normal operation of the JBoss Enterprise BRMS Platform. Apache Ant requires a correctly installed Java Runtime Environment (JRE).

If running a development workstation, Apache Ant may already be installed. The Apache Ant package from the Apache Ant website includes the Trax plugin but Red Hat Enterprise Linux includes it as a separate package.

Procedure A.1. Installing Apache Ant on Red Hat Enterprise Linux

- Download and install Apache Ant and the Trax plugin on Red Hat Enterprise Linux Repository by issuing this command:

```
[localhost]$ sudo yum install ant-trax
```

Procedure A.2. Installing Apache Ant on Other Operating Systems

1. **Download and Extract**
   Download the **Apache Ant** binary release from http://ant.apache.org/bindownload.cgi.

   Once it is downloaded, extract it in a preferred installation location, such as **c:\Program Files\Apache\Ant** or **/opt/apache-ant-1.8/**.

2. **Add the ANT_HOME Environment Variable**
   Create an environment variable called **ANT_HOME**. This variable has to contain the path created in the previous step.

   - Do this on Red Hat Enterprise Linux by adding the following line to the **~/.bash_profile** file, substituting the path with that created in the previous step.

     ```
     export ANT_HOME=/opt/apache-ant-1.8.1
     ```

   - On Microsoft Windows, do this by clicking on the **Start Menu**, opening the **Control Panel** and then selecting **System** -> **Advanced** -> **Environment Variables**.

     Create a new variable, calling it **ANT_HOME**, and configure it to point to the directory created in the previous step.

3. **Include bin in the PATH**
   Append the **bin** directory of the Ant installation to the **PATH** environmental variable.

   - On Unix/Linux systems, one does this simply by adding the following line to the **~/.bash_profile** file after the one which sets the **ANT_HOME** variable:

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

- On Microsoft Windows, do this task by opening the **Control Panel** and selecting **System** -> **Advanced** -> **Environment Variables** -> **System Variables**. Edit the **PATH** variable and append the text **;%ANT_HOME%\bin**.

To test the Apache Ant installation, run **ant -version** from a command line shell. The output should look similar to this:

```
[localhost]$ ant -version
Apache Ant version 1.8 compiled on June 27 2008
```

To learn more about **Apache Ant**, visit the project's website at http://ant.apache.org.

Report a bug

APPENDIX B. PERSISTENCE MANAGER CONFIGURATIONS

B.1. EXAMPLE PERSISTENCE MANAGER CONFIGURATIONS

Here are several example Persistence Manager configurations for the supported databases. You will, of course, have to update the values in the configurations with those that are correct for your database, such as the JDBC URL and schemaObjectPrefix.

IMPORTANT

The JBoss Enterprise BRMS Platform is supported on the databases listed here: http://www.jboss.com/products/platforms/brms/supportedconfigurations/

For additional details about Apache Jackrabbit Persistence Managers you should refer to http://wiki.apache.org/jackrabbit/PersistenceManagerFAQ

Example B.1. Generic JDBC Configuration for MySQL using `BundleDbPersisenceManager`

```
<PersistenceManager class=
"org.apache.jackrabbit.core.persistence.bundle.BundleDbPersistenceManage
r">
    <param name="driver" value="com.mysql.jdbc.Driver"/>
    <param name="url" value="jdbc:mysql://localhost/brms"/>
    <param name="user" value="brms_user"/>
    <param name="password" value="brms_password"/>
    <param name="schema" value="mysql"/>
    <param name="schemaObjectPrefix" value="${wsp.name}_"/>
</PersistenceManager>
```

Example B.2. MySQL Configuration using `MySqlPersistenceManager`

```
<PersistenceManager class=

"org.apache.jackrabbit.core.persistence.bundle.MySqlPersistenceManager">
    <param name="driver" value="com.mysql.jdbc.Driver"/>
    <param name="url" value="jdbc:mysql://localhost:3306/brms"/>
    <param name="user" value="brms_user"/>
    <param name="password" value="brms_password"/>
    <param name="schemaObjectPrefix" value="${wsp.name}_"/>
    <param name="schema" value="mysql"/>
</PersistenceManager>
```

Example B.3. Oracle Configuration using `OraclePersistenceManager`

```
<PersistenceManager class=
 "org.apache.jackrabbit.core.persistence.bundle.OraclePersistenceManager
">
    <param name="driver" value="oracle.jdbc.OracleDriver"/>
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:brms" />
```

```
    <param name="schema" value="oracle"/>
    <param name="user" value="brms_user" />
    <param name="password" value="brms_password" />
    <param name="schemaObjectPrefix" value="${wsp.name}_" />
</PersistenceManager>
```

Example B.4. PostgreSQL Configuration using `PostgreSQLPersistenceManager`

```
<PersistenceManager
class="org.apache.jackrabbit.core.persistence.bundle.
PostgreSQLPersistenceManager">
    <param name="driver" value="org.postgresql.Driver"/>
    <param name="url" value="jdbc:postgresql://localhost:5432/brms" />
    <param name="schema" value="postgresql"/>
    <param name="user" value="brms_user" />
    <param name="password" value="brms_password" />
    <param name="schemaObjectPrefix" value="${wsp.name}_" />
</PersistenceManager>
```

Example B.5. Microsoft SQL Server 2005 Configuration using `MSSqlPersistenceManager`

```
<PersistenceManager
class="org.apache.jackrabbit.core.persistence.bundle.
MSSqlPersistenceManager">
    <param name="driver"
        value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
    <param name="url"
        value="jdbc:sqlserver://localhost:3918;DatabaseName=brms" />
    <param name="user" value="brms_user" />
    <param name="password" value="brms_password" />
    <param name="schema" value="mssql"/>
    <param name="schemaObjectPrefix" value="${wsp.name}_" />
</PersistenceManager>
```

Report a bug

APPENDIX C. CAMEL INTEGRATION

C.1. EXAMPLE CONFIGURATION FOR CAMEL INTEGRATION

Users running JBoss BRMS with Camel inside a application server need to bundle their own web app to avoid conflicts between the dependencies of different versions of CXF on the classpath.

The web app should include the following files in the **`Drools-camel-server/src/main/resources/`** directory:

- beans.xml
- camel-server.xml
- knowledge-server.xml
- soap-wsdl
- test.drl

Example C.1. beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd"
    default-autowire="byName">


  <!-- loads the kbases and ksessions into the ApplicationContext -->
  <import resource="classpath:knowledge-services.xml" />

  <!-- Builds the routes that makes the ksessesions available as
services -->
```

```
  <import resource="classpath:camel-server.xml" />

</beans>
```

Example C.2. camel-server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:cxf="http://camel.apache.org/schema/cxf"
       xmlns:jaxrs="http://cxf.apache.org/jaxrs"
       xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://camel.apache.org/schema/cxf
http://camel.apache.org/schema/cxf/camel-cxf.xsd
http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd
http://camel.apache.org/schema/spring
http://camel.apache.org/schema/spring/camel-spring.xsd
">

  <import resource="classpath:META-INF/cxf/cxf.xml" />
  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

  <!--
! If you are running on JBoss you will need to copy a camel-jboss.jar
into the lib and set this classloader configuration
| http://camel.apache.org/camel-jboss.html
<bean id="jbossResolver"
class="org.apache.camel.jboss.JBossPackageScanClassResolver"/>
-->

  <!--
! Define the server end point.
! Copy and paste this element, changing id and the address, to expose
services on different urls.
```

```
! Different Camel routes can handle different end point paths.
-->
  <cxf:rsServer id="rsServer"
                address="/rest"
                serviceClass="org.drools.jax.rs.CommandExecutorImpl">
       <cxf:providers>
           <bean class="org.drools.jax.rs.CommandMessageBodyReader"/>
       </cxf:providers>
  </cxf:rsServer>

  <cxf:cxfEndpoint id="soapServer"
            address="/soap"
             serviceName="ns:CommandExecutor"
             endpointName="ns:CommandExecutorPort"
          wsdlURL="soap.wsdl"
          xmlns:ns="http://soap.jax.drools.org/" >
    <cxf:properties>
      <entry key="dataFormat" value="MESSAGE"/>
      <entry key="defaultOperationName" value="execute"/>
    </cxf:properties>
  </cxf:cxfEndpoint>

  <!-- Leave this, as it's needed to make Camel "drools" aware -->
  <bean id="droolsPolicy"
class="org.drools.camel.component.DroolsPolicy" />

  <camelContext id="camel"
xmlns="http://camel.apache.org/schema/spring">
    <!--
! Routes incoming messages from end point id="rsServer".
! Example route unmarshals the messages with xstream and executes
against ksession1.
! Copy and paste this element, changing marshallers and the 'to' uri, to
target different sessions, as needed.
!-->

    <route>
       <from uri="cxfrs://bean://rsServer"/>
       <policy ref="droolsPolicy">
         <unmarshal ref="xstream" />
         <to uri="drools:node1/ksession1" />
         <marshal ref="xstream" />
       </policy>
    </route>

    <route>
      <from uri="cxf://bean://soapServer"/>
      <policy ref="droolsPolicy">
        <unmarshal ref="xstream" />
        <to uri="drools:node1/ksession1" />
        <marshal ref="xstream" />
      </policy>
    </route>

  </camelContext>
```

```
</beans>
```

Example C.3. knowledge-server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:drools="http://drools.org/schema/drools-spring"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://drools.org/schema/drools-spring http://drools.org/schema/drools-
spring-1.3.0.xsd">

  <drools:grid-node id="node1"/>

  <drools:kbase id="kbase1" node="node1">
    <drools:resources>
      <drools:resource type="DRL" source="classpath:test.drl"/>
    </drools:resources>
  </drools:kbase>

  <drools:ksession id="ksession1" type="stateless" kbase="kbase1"
node="node1"/>

</beans>
```

Example C.4. soap-wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="CommandExecutor"
targetNamespace="http://soap.jax.drools.org/"
```

```
xmlns:ns1="http://cxf.apache.org/bindings/xformat"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://soap.jax.drools.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
<xsd:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://soap.jax.drools.org/"
xmlns:tns="http://soap.jax.drools.org/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="execute" type="tns:execute"/>
<xsd:complexType name="execute">
<xsd:sequence>
<xsd:element minOccurs="0" name="arg0"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="executeResponse" type="tns:executeResponse"/>
<xsd:complexType name="executeResponse">
<xsd:sequence>
<xsd:element minOccurs="0" name="return" type="xsd:anyType"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="execute">
    <wsdl:part element="tns:execute" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="executeResponse">
    <wsdl:part element="tns:executeResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="CommandExecutorPortType">
    <wsdl:operation name="execute">
      <wsdl:input message="tns:execute" name="execute">
    </wsdl:input>
      <wsdl:output message="tns:executeResponse" name="executeResponse">
    </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CommandExecutorSoapBinding"
type="tns:CommandExecutorPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="execute">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="execute">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="executeResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CommandExecutor">
```

```
    <wsdl:port binding="tns:CommandExecutorSoapBinding"
name="CommandExecutorPort">
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

An example test.drl is provided for testing purposes.

Example C.5. test.drl

```
/*
 * Copyright 2010 JBoss Inc
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.test

declare Message
   text : String
end

query "get All Messages"
    $m : Message()
end

rule "echo" dialect "mvel"
when
   $m : Message()
then
    $m.text = "echo:" + $m.text;
end
```

The web app should also include a **web.xml** file in the **Drools-camel-server/src/main/webapp/WEB-INF/** directory:

Example C.6. web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:beans.xml</param-value>
  </context-param>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <servlet>
    <display-name>CXF Servlet</display-name>
    <servlet-name>CXFServlet</servlet-name>
    <servlet-class>
      org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>CXFServlet</servlet-name>
    <url-pattern>/kservice/*</url-pattern>
  </servlet-mapping>

  <session-config>
    <session-timeout>10</session-timeout>
  </session-config>
</web-app>
```

For further details about Apache Camel, please refer to the *Apache Camel Documentation*

Report a bug

APPENDIX D. REVISION HISTORY

Revision 5.3.1-88.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revision 5.3.1-88 **Wed Jan 23 2013** **Red Hat Content Services**

Updated Documentation for the JBoss Enterprise BRMS Platform 5.3.1 release.