JBoss Enterprise BRMS Platform 5

BRMS 管理者ガイド

JBoss 管理者向け

エディッション 5.3.1

Last Updated: 2017-11-22

JBoss Enterprise BRMS Platform 5 BRMS 管理者ガイド

JBoss 管理者向け
エディッション 5.3.1

Red Hat Content Services

法律上の通知

概要

本書は、管理者が JBoss Enterprise BRMS Platform の設定、カスタマイズを行う際に必要な手順が説明されています。

目次

前書き

第1章 はじめに

1.1. JBOSS ENTERPRISE BUSINESS RULES MANAGEMENT SYSTEM PLATFORM

JBoss Enterprise BRMS Platform は、ビジネスルールやビジネスプロセスの管理、格納、作成、変更、デプロイを行うためのビジネスルール管理システムです。JBoss Developer Studio の Web ベースのユーザーインターフェースやプラグインでは、ユーザーに異なるロールを与えて、ニーズに適した環境を提供します。JBoss Enterprise BRMS は、ビジネスアナリスト、ルールエキスパート、開発者、ルール管理者に特化した環境を提供しています。

JBoss Enterprise BRMS Platform は、各種オペレーティングシステム、 Java 仮想マシン (JVM)、データベース設定にも対応しています。全認定済みの設定および互換性のある設定のすべてが、http://www.redhat.com/resourcelibrary/articles/jboss-enterprise-brms-supported-configurations から一覧で確認いただけます。

バグを報告する

第2章 データベースの設定

2.1. JBOSS ENTERPRISE BRMS および APACHE JACKRABBIT

JBoss Enterprise BRMS Platform は、Content Repository API for Java (JCR) 仕様を用いてアセットの保存やトラッキングを行います。Apache Jackrabbit は JBoss BRMS に同梱されている JCR 実装です。

デフォルトの Apache Jackrabbit レポジトリは、評価目的でのみ提供されている Derby データベースを使用しており、実稼働環境での使用には対応していません。対応データベースの完全一覧は、https://access.redhat.com/knowledge/articles/119933 を参照してください。

バグを報告する

2.2. レポジトリの場所の指定

デフォルトでは、レポジトリは、 JBoss Enterprise BRMS Platform が初回実行されると作成されます。場所が指定されていない場合、レポジトリは実行コマンドが発行されたディレクトリに作成されます。

レポジトリは、セキュアな場所に保存されバックアップも行われます。

レポジトリの場所の指定は、JBoss Seam **components.xml** 設定ファイルを編集して行います。

手順2.1 レポジトリの場所の指定

1. アプリケーションサーバーを終了します。
2. **deploy/jboss-brms.war/WEB-INF/** にある **components.xml** ファイルを開き、repository.root.directory Key-Value 属性をアンコメントします。

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

3. レポジトリの場所を repository.root.directory Key-Value 属性部分に追加します (このディレクトリは予め存在していなければなりません)。

```
<property
name="properties"><key>org.drools.repository.configurator</key><valu
e>org.drools.repository.jackrabbit.JackrabbitRepositoryConfigurator<
/value><!--  the root directory for the repo storage the directory
must exist. --
><key>repository.root.directory</key><value>/BRMSRulesRepositoryLoca
```

```
tion</value></property>
```

4. JBoss Enterprise BRMS Platform は、すでにデータストアがない場合はこの場所に新しく作成します。既存のデータストアを継続して使用するには、アプリケーションサーバーを再起動する前に既存のファイルを新しい場所にコピーします。

5. アプリケーションサーバーを再起動します。既存のデータストアを新しい場所に移動していない場合は、新しいデータストアが作成されます。

バグを報告する

2.3. 外部の RDBMS の設定

JBoss Enterprise BRMS Platform を設定して外部 RDBMS をデータソースとして使用することができます。BRMS Repository Configuration ツールを使って必要な RDBMS を設定します。

手順2.2 Repository Configuration Tool を使った外部 RDBMS の設定

1. JBoss Enterprise BRMS Platform のユーザーインターフェースにログインします。

2. 左側のナビゲーションパネルから、**Administration** → **Repository Configuration** を選択します。

3. **`Select RDBMS type:`** のドロップダウンメニューから RDBMS タイプを選択します。

4. JNDI が設定されている場合、**`USE JNDI`** を確認して、JNDI 名を入力します。

5. JNDI が設定されていない場合、必要なデータベース情報を入力します。

   - ドライバー
   - URL
   - ユーザー
   - パスワード

6. **`Generate repository config`** をクリックして **`repository.xml`** ファイルを生成します。

7. **`Save Configuration`** をクリックして、生成した **`repository.xml`** ファイルをダウンロードして、**`components.xml`** ファイルで指定した場所にある **`repository.xml`** を置き換えます。

バグを報告する

2.4. 外部キーのインデックス作成

Oracle や Postgres などのデータベースは、外部キーごとのインデックスの自動作成は行いません。そのため、デッドロックが発生します。この状況を回避するには、外部キー制約で参照されている列の一部にインデックスを作成する必要があります。

以下の列にインデックスを作成して、デッドロックを回避しクエリのパフォーマンスを向上します。

ヒューマンタスクスキーマ:

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

コアエンジンスキーマ:

- eventtables.instanceid

バグを報告する

2.5. JBOSS ENTERPRISE WEB SERVER のデータソースの設定

デフォルト設定では、実稼働環境に適していないか、サポートされていない組み込みデータベースを使用しています。実稼働環境へのデプロイの前に、この環境を対応のデータベースに変更するようにしてください。JBoss Enterprise Web Server をお使いの場合は **jbpm-human-task.war** と **business-central-server.war** を設定する必要があります。

以下のサンプルでは、MySQL データベースを使用します。

手順2.3 JBoss Enterprise Web Server のデータソースの設定

1. MySQL ドライバーを **tomcat6/lib/** にコピーします。
2. **tomcat6/conf/context.xml** にあるデータソースリソースの設定:

```
<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. **tomcat6/webapps/business-central-server.war/WEB-INF/Classes/META-INF/persistence.xml** ファイルを編集して、**context.xml** からの一意名を含めます。

```
  <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. **tomcat6/webapps/jbpm-human-task.war/WEB-INF/classes/META-INF/persistence.xml** ファイルを編集して、**context.xml** からの一意名を含めます。

```
<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 にはトランザクションマネージャーが含まれていません。以下の例は、Bitronix 向けの設定で、評価目的のためだけに提供されています。Bitronix は http://docs.codehaus.org/display/BTM/Home からダウンロードしていただけます。

```
 #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. Bitronix トランザクションマネージャーを設定するには、**tomcat6/conf/btm-config.properties** ファイルを作成します。

```
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
```

以下の行を **tomcat6/conf/context.xml** に追加します。

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

以下の行を **tomcat6/conf/server.xml** に追加します。

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

設定プロパティファイルを作成します。Unix ベースのシステムでは、このファイルは **tomcat6/setenv.sh**、Windows システムでは **tomcat6/setenv.bat** としてください。

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

バグを報告する

2.6. JBOSS ENTERPRISE APPLICATION PLATFORM 6 のデータソース設定

デフォルト設定では、実稼働環境に適していないか、サポートされていない組み込みデータベースを使用しています。実稼働環境へのデプロイの前に、この環境を対応のデータベースに変更するようにしてください。

対応データベースの一覧については、http://www.redhat.com/resourcelibrary/articles/jboss-enterprise-brms-supported-configurations を参照してください。

以下の手順では、例として PostgreSQL データベースを使用しています。

手順2.4 JBoss Enterprise Platform 6 のデータソース設定

1. JBoss Enterprise Application Platform 6 ディレクトリ構造ににデータベースのディレクトリを追加します。このディレクトリは、使用予定のデータベース向けに名前を設定してください。例えば、PostgreSQL データベースには、以下の構造で作成してください。

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

   MySQL データベースに対しては、以下の構造で作成してください。

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

2. データベースに適した JDBC ドライバーをダウンロードします。

3. データベース用に作成したディレクトリに、**main** ディレクトリを作成します。

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

JBDC ドライバーを **jboss-eap-6.0/modules/org/postgresql/main/** ディレクトリにコピーします。

4. **module.xml** ファイルを作成して、以下の内容を含めて **jboss-eap-6.0/modules/org/postgresql/main/** に保存してください。

```
<?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. プロファイル設定ファイルを編集します。例えば、**jboss-eap-6.0/standalone/configuration/standalone.xml** は **brmsDS** と呼ばれる PostgreSQL データベース、ユーザー名 brms、パスワード brms を設定しています。

```
<subsystem xmlns="urn:jboss:domain:datasources:1.1"><datasources>
<datasource jta="true" jndi-name="java:jboss/datasources/brmsDS"
pool-name="java:jboss/datasources/brmsDS_Pool" enabled="true" use-
java-context="true" use-ccm="true"><connection-
url>jdbc:postgresql://localhost:5432/brms</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>brms</user-name><password>brms</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. **jboss-eap-6.0/standalone/deployments/business-central-server.war/WEB-INF/classes/META-INF/persistence.xml** ファイルを編集して、**DefaultDS** から **brmsDS** へデータソース名を、Hibernate 方言を PostgreSQL に変更します。

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

7. **jboss-eap-6.0/standalone/deployments/jbpm-human-task.war/WEB-INF/classes/META-INF/persistence.xml** ファイルを編集して、**DefaultDS** から **brmsDS** へデータソース名を、Hibernate 方言を PostgreSQL に変更します。

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

バグを報告する

2.7. JACKRABBIT での JBOSS BRMS のクラスタリング

JBoss Enterprise BRMS を設定して、クラスターノード間で入ってくるリクエストを分散するロードバランサープロキシ経由でレポジトリにアクセスすることで、クラスター内の各 JBoss BRMS ノードを用いて JBoss Enterprise Application Platform クラスターで実行し、高可用性とフェールオーバーを提供することができます。

手順2.5 Jackrabbit でのクラスタリングの設定

1. JBoss Enterprise Application Platform クラスターの設定については、『管理設定ガイド』にある JBoss Enterprise Application Platform 5 『クラスタリングガイド』を参照してください。

2. ロードバランサーの設定については、JBoss Enterprise Application Platform 5 『HTTP コネクター負荷分散ガイド』 を参照してください。

3. jboss-brms.war のコピーをアプリケーションサーバークラスターの各ノード **$JBOSS_HOME/server/*nodeX*/deploy/** にデプロイします。

4. デフォルト設定では、実稼働環境に適していないか、サポートされていない組み込みデータベースを使用しています。実稼働環境へのデプロイの前に、この環境を対応のデータベースに変更するようにしてください。設定に関する説明は、本章の別のセクションを参照してください。

5. クラスター内のノードごとに、個別のワークスペース、バージョン、検索インデックスを

repository.xml ファイルに設定する必要があります。JBoss Enterprise BRMS ユーザーインターフェースにログインして、**Administration** → **Repository Configuration** を選択し、必要なデータベース向けに **repository.xml** を生成します。

6. **repository.xml** ファイルをクラスターの各ノードの **jboss-brms.war/WEB-INF/components.xml** で指定したレポジトリの場所にコピーします。

7. クラスターのノードごとに一意の ID を設定する必要があります。これは、**repository.xml** ファイルの各ノードのコピーに設定します。例：

```
<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. **web-app** の子要素として **<distributable/>** 要素を **jboss-brms.war/WEB-INF/web.xml** ファイルに追加します。

9. JBoss Enterprise BRMS ユーザーインターフェースにログインすることで、クラスターノードが機能しているかを検証します。ターミナルの出力を検証して、どのノードがセッションを処理しているか確認します。そのノードを終了し、しばらくすると、クラスター内の別のノードがこのセッションを引き継ぐはずです。

Jackrabbit でのクラスタリングに関する詳細情報は、http://wiki.apache.org/jackrabbit/Clustering を参照してください。

バグを報告する

2.8. JCR としての MODESHAPE の使用

ModeShape は Apache Jackrabbit がある場所で利用可能な Java Content Repository (JCR) です。JBoss Enterprise BRMS 5.2 と 5.3 にテクニカルプレビューとして同梱されています。

ModeShape のインストール前に、以下の要件を満たす必要があります。

- BRMS スタンドアローンパッケージのダウンロードとインストールが済んでいること。方法については、『JBoss Enterprise BRMS Getting Started Guide』 を参照してください。
- Apache Ant がインストールされていること。インストール方法については、本章の最後の付録を参照してください。
- 実稼働デプロイには、対応データベースサーバーが必要。対応データベースサーバーの完全一覧は、https://access.redhat.com/knowledge/articles/119933 を参照してください。
    - 使用予定のデータベースインスタンスがすでに作成されていること。

- データベースを変更できるパーミッションを持つデータベースユーザーが存在すること。
- データベースの JDBC ドライバー JAR ファイルがサーバー設定の **lib/** ディレクトリにあること。

手順2.6 ModeShape の設定

1. ModeShape ディレクトリから ant インストーラーを実行して、ModeShape のインストール先のサーバープロファイルを指定します。

```
[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. ユーザーアカウント **admin** と **mailman** を追加して、以下で指定したロールを割り当てます。デフォルト設定はテキストファイルを使用してユーザー名、パスワード、割り当てたロールを保存します。ここでは、認証設定を変更していないとの前提で説明を進めています。認証設定を変更した場合は、その設定に従いユーザーを追加してください。

   a. テキストエディターで ***PROFILE*/conf/props/brms-users.properties** ファイルを開き、改行してユーザー**admin** と **mailman** を **username=password** の構文で追加します。

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

   b. 以下で指定したロールをこれら 2 つのユーザーに割り当てるには、テキストエディターで ***PROFILE*/conf/props/brms-roles.properties** を開き、ユーザーごとに **username=role1, role2, role3** の構文で新しく行を追加していきます。

   admin ユーザーには、**JBossAdmin**、**HttpInvoker**、**user**、**admin** のロールを割り当てる必要があります。

   mailman ユーザーには、**JBossAdmin** と **readwrite** のロールを割り当てる必要があります。

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

   c. 任意で、プレーンテキストではなくエンコード形式でパスワードを保存することができます。方法は、アプリケーションサーバーのディレクトリ **(jboss-as** または **jboss-as-web)** で以下のコマンドを実行して、エンコード形式のパスワードを作成します。*PASSWORD* は実際のパスワードで置き換えてください。

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

- 次に、***PROFILE*/conf/props/brms-users.properties** に格納されているパスワードをエンコードしたもので置き換えます。

3. ModeShape のデフォルト設定では、実稼働環境に適していないか、サポートされていないデータベースを使用しています。実稼働環境へのデプロイの前に、この環境を対応のデータベースに変更するようにしてください。

   デフォルトの Modeshape レポジトリ設定は、JNDI データソース **–ModeShapeBRMSRepo–** をデータソースに使用します。これは、***PROFILE*/deploy/modeshape-brms-store-ds.xml** ファイルで設定します。このファイルを編集して、Hypersonice から任意のデータベースにデータソースの設定を変更します。このファイルは、標準の JBoss データソース設定ファイルです。

   JBoss データソース設定の詳細については、アプリケーションサーバー文書のデータソース設定の章を参照してください。

4. **jboss-brms.war/WEB-INF/components.xml** ファイルを編集して、Apache Jackrabbit 設定を削除し ModeShape 設定を追加します。

   Apache Jackrabbit 設定を削除するには、以下のセクションをコメントアウトしてください。

```
<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>
```

   デフォルト設定では、ModeShape 設定はコメントアウトされています。以下のようにこの設定をアンコメントしてください。

```
<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 には、2 種のユーザーアカウント **admin** と **mailman** が必要です。これらのアカウントのパスワードは、以前の手順の ModeShape 設定にて指定しておく必要があります。

```
<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. サーバーを再起動します。

バグを報告する

2.9. 検索とインデックス作成

Apache Lucene (http://lucene.apache.org/) には、検索とインデックス作成が含まれています。

デフォルトでは、検索インデックスはローカルファイルシステムに保存されています。これは、パフォーマンスを高めるためです。Red Hat では、特別な要件がない限り、このデフォルト設定の変更を推奨していません。

検索インデックスの場所の設定は、**repository.xml** ファイルの **<SearchIndex>** 要素を変更してください。

```
<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>
```

バグを報告する

第3章 データ管理

3.1. データバックアップ

JBoss Enterprise BRMS では、バックアップソリューションが提供されていません。データベースは、データベースベンダー提供のツールを使用してバックアップしてください。

バックアップのリストアを行う際は、Red Hat は、データベースインデックスが再構築され新規データ向けに最適化されるように、まずこれらのインデックスを削除するように推奨しています。

バグを報告する

3.2. データのインポートおよびエクスポート

3.2.1. データ移行

JBoss Enterprise BRMS では、データベースの移行時に JCR 規格のエクスポート/インポート機能を使用できます。JCR 規格で定義されているように、エクスポートしたレポジトリは XML ファイルに書き込まれます。

警告

インポート/エクスポート機能は、バックアップソリューションとしての使用を目的としておらず、Red Hat ではバックアップソリューションとしてのサポートはしていません。データベースベンダー提供のバックアップシステムを必ず使用するようにしてください。

インポート/エクスポートの操作を行う場合は、以下の点に注意してください。

- インポート時は、データベース内の既存のコンテンツをすべて削除されます。
- データベースの使用中には、エクスポート/インポートの操作は行わないようにしてください。
- サーバーで利用できるメモリやデータベースのサイズにより、パフォーマンスは大きく左右されます。インポートは、メモリがかなり消費されるプロセスとなっています。
- レポジトリのバージョン履歴はエクスポートされず、ルールは Creation Date 属性がインポートされた日付で再設定されます。

バグを報告する

3.2.2. レポジトリのエクスポート

手順3.1 レポジトリのエクスポート

1. BRMS ユーザーインターフェースの左側のナビゲーションパネルから、**Administration** → **Import Export** を選択します。
2. **Export** をクリックします。

ブラウザーにて XML ファイルのレポジトリが含まれる zip ファイルがダウンロードされます。

レポジトリのサイズにより、この操作は時間がかかる場合があります。

バグを報告する

3.2.3. レポジトリのインポート

レポジトリをインポートすると、インポートした XML ファイルのコンテンツにより、レポジトリ内のコンテンツが置き換えられます。

手順3.2 レポジトリのインポート

1. ナビゲーションパネルから **Administration** → **Import Export** を選択します。
2. エクスポートデータを含む XML ファイルを選択するには、**Browse** ボタンをクリックして、ローカルファイルシステムを参照します。
3. **Import** をクリックして、確認ダイアログで**OK** を選択します。

レポジトリのインポートが正常に完了すると、ブラウザーは更新されレポジトリの新規コンテンツが表示されます。

レポジトリのインポート後にパッケージバイナリを再構築するには、**Knowledge bases** → **Create New** → **Rebuild all package binaries** を選択します。古いバージョンの JBoss Enterprise BRMS からレポジトリをインポートした結果、エラーが発生した場合に、再構築が必要になる可能性があります。

バグを報告する

第4章 セキュリティ

4.1. 認証

4.1.1. 認証

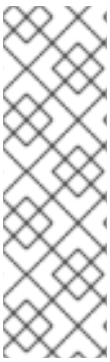
JBoss Enterprise BRMS Platform は JAAS (*Java Authentication and Authorization Service*) を使用して、ユーザー認証情報を確認します。このサービスはアプリケーションサーバーから提供され、別の認証システムにアクセスする際にこのサービスを使用します。この別システムですが、Lightweight Directory Access Protocol (LDAP)、 Active Directory サーバー、または JDBC データベースがお使いいただけます。

バグを報告する

4.1.2. 認証の設定

どの認証方式を使うかについては **jboss-brms.war/WEB-INF/components.xml** ファイルから設定します。デフォルト設定では、多くがコメントアウトされていますが、実際の設定は以下のようになります。

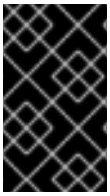
```
<security:identity authenticate-method="#{authenticator.authenticate}"
jaas-config-name="jmx-console"/><component
name="org.jboss.seam.security.roleBasedPermissionResolver"><property
name="enableRoleBasedAuthorization">false</property></component>
```



注記

JBoss BRMS 5.1 以前のバージョンでは、components.xml ファイルは以下のようになっています。

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

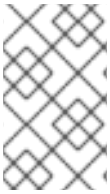


重要

このデフォルト設定は、**jmx-console** 認証ポリシーにて定義されているアカウント名、パスワード、ロールを使用します。Red Hat は、このファイルを編集してお使いの環境に合わせてカスタマイズするよう推奨しています。

認証の設定は、以下の手順に従ってください。

1. アプリケーションサーバーの適切な JBoss ログインモジュールを編集します。
2. このモジュールを使用するように、JBoss Enterprise BRMS Platform を設定します。



注記

JBoss ログインモジュールの多くは、各ユーザーに対して 1 つ以上のロールを指定する手段があります。JBoss Enterprise BRMS Platform には、ユーザーロールの管理に関する独自のメカニズムがあります。

警告

role-based authorization が無効になっている場合は、事実上すべてのユーザーが管理者ロールを持ちます。このロールがあると、JBoss Enterprise BRMS Platform への完全なアクセス権を持つようになります。

バグを報告する

4.1.3. JAAS のパスワード設定

デフォルトの JAAS 認証システムを使用している場合、ユーザー名、パスワードは JBoss Enterprise BRMS、Process Designer、Business Central コンソールの間で同期する必要があります。同じユーザー名とパスワードを使用しないと、各コンポーネントが連携して機能しなくなります。

別のユーザーを **brms-users.properties** ファイルに追加した場合は、Process Designer と Business Central コンソールとも同期する必要があります。

手順4.1 ユーザー名とパスワードの同期

1. Process Designer: Process Designer とは JBoss Enterprise BRMS に統合された別のアプリケーションですが、このアプリケーションのユーザー名とパスワードを編集するには、**designer.war/profiles/jbpm.xml** ファイルを開き、**usr** と **pwd** プロパティを編集します。

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

2. Business Central コンソール: Business Central コンソールのユーザー名とパスワードを編集するには、**business-central-server.war/WEB-INF/classes/jbpm.console.properties** ファイルを開き、**guvnor.usr** と **guvnor.pwd** プロパティを編集します。

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

バグを報告する

4.1.4. 認証の例: UserRolesLoginModule

この例では、**props/brms-users.properties** と **props/brms-roles.properties** ファイルに保存されているユーザーアカウントへアクセスするための

org.jboss.security.auth.spi.UsersRolesLoginModule ログインモジュールの使用方法を説明しています。

手順4.2 認証の例: UserRolesLoginModule

1. **認証システムが正しく設定されていることを確認します。**
   このログインモジュールは、2 つのファイルを使用して、ログイン名、パスワード、各ユーザーに割り当てたロールを保存します。**jboss-as-web/server/*PROFILE*/conf/props/** ディレクトリに **brms-users.properties** と **brms-roles.properties** ファイルを作成して、**username=password** の形式で **brms-users.properties** に少なくとも 1 つ以上のユーザーを指定します (**brms-roles.properties** ファイルは空のままでも構いません)。

2. **シャットダウン**
   アプリケーションサーバーをシャットダウンしてから変更を行います。

3. **JBoss ログインモジュールの設定**
   JBoss ログインモジュールを設定するには、テキストエディターで **jboss-as-web/server/*PROFILE*/conf/login-config.xml** を開きます。これは、XML ファイルで **<policy>** 要素と複数の **<application-policy>** 子要素が含まれています。各 **<application-policy>** 要素は、様々な認証スキームを定義します。以下の **<application-policy>** XML スニペットを **<policy>** 要素の新しい子として追加します。

   ```
   <!--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. **ログインモジュールを使用するための BRMS Platform の設定**
   jboss-as-web/server/*PROFILE*/deploy/JBoss-BRMS.war/WEB-INF/components.xml ファイルを開きます。ここには、**<components>** 要素が 1 つと、**<security:identity>** などの複数の子要素が含まれています。

   矛盾が起こらないように、既存の **<security:identity>** 要素をコメントアウトします。以下の **<security:identity>** 要素を追加します。

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

   jaas-config-name プロパティは、application-policy と同じでなければなりません。以前の手順で application-policy プロパティを変更した場合は、ここで同じになるようにjaas-config-name プロパティを変更します。

5. **再起動**
   アプリケーションサーバーを再起動します。

バグを報告する

4.1.5. 認証の例: LDAP

LDAP は、大企業でよく使用されています。基本的な設定手順は、以前の例と同じですが、詳細の設定が異なります。

手順4.3 認証の例 2: LDAP

1. **LDAP サーバーが正しく設定されていることを確認**
   ファイアウォールとネットワーク構成設定がアプリケーションサーバーと LDAP サーバーの通信を妨害していないことを確認します。

2. **シャットダウン**
   アプリケーションサーバーをシャットダウンしてから変更を行います。

3. **JBoss ログインモジュールの設定**
   JBoss ログインモジュールを設定するには、テキストエディターで **jboss-as-web/server/*PROFILE*/conf/login-config.xml** を開きます。これは、XML ファイルで **<policy>** 要素と複数の **<application-policy>** 子要素が含まれています。各 **<application-policy>** 要素は、様々な認証スキームを定義します。以下の **<application-policy>** XML スニペットを **<policy>** 要素の新しい子として追加します。

```
<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>
```

この設定ファイルの値をお使いの LDAP サーバーに適した値に変更します。

4. **ログインモジュールを使用するための BRMS Platform の設定**
   `jboss-as-web/server/PROFILE/deploy/jboss-brms.war/WEB-INF/components.xml` ファイルを開きます。ここには、**`<components>`** 要素が 1 つと、**`<security:identity>`** などの複数の子要素が含まれています。

   矛盾が起こらないように、既存の **`<security:identity>`** 要素をコメントアウトします。以下の **`<security:identity>`** 要素を追加します。

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

   jaas-config-name プロパティは、application-policy と同じでなければなりません。以前の手順で application-policy プロパティを変更した場合は、ここで同じになるようにjaas-config-name プロパティを変更します。

5. **再起動**
   アプリケーションサーバーを再起動します。

バグを報告する

4.2. ユーザーの管理

4.2.1. ロールベースのパーミッション

JBoss Enterprise BRMS は、ロールベースの認証を使用してユーザーへのパーミッションを割り当てます。デフォルトでは、ロールベース認証は無効になっているため有効にする必要があります (詳細は『JBoss BRMS スタートガイド』 を参照してください)。ロールベースの認証が有効でない場合、全ユーザーに完全な管理者権限が与えられます。

割り当てることのできるロールは、admin、analyst、package のパーミッションです。

Admin 権限

admin ロールを割り当てられたユーザーは、JBoss Enterprise BRMS Platform へのすべてのエリアに完全なアクセス権限を持ち、他のユーザーのパーミッションも管理することができます。

analyst 権限

analyst 権限は、ルールの管理を担当するユーザー向けです。開発者とビジネスアナリストの両方にこの analyst 権限を割り当てるとよいでしょう。analyst 権限は、カテゴリと関連付けられており、analyst 権限を割り当てられたルールユーザーを管理することができます。

analyst 権限に加え、analyst read-only 権限を割り当てることもできます。これは、カテゴリ内のルールを参照できますが、ルールの編集や変更はできません。

Package 権限

Package 権限には 3 種類あります。

Package Administrator

Package Administrator 権限は、パッケージのデプロイなど、指定のパッケージに対して完全な権限を持つことになります。Package Administrator 権限には、パッケージ部分以外の JBoss Enterprise BRMS Platform への管理者権限は含まれていません。

Package Developer

Package Developer は、指定のパッケージ内でアイテムの作成や編集が可能です。つまり、テストの作成や実行はできますが、パッケージのデプロイ権限は含まれていません。

Package Read-only

Package read-only 権限は、パッケージへの読み取り専用権限が割り当てられ、パッケージを参照することができます。

JBoss Enterprise BRMS はユーザー ID を管理しません。BRMS ユーザーとして割り当てられたユーザーのみが、ユーザーインターフェースに表示され、これらのユーザーは手動で追加する必要があります。

バグを報告する

4.2.2. 新規ユーザーの追加

手順4.4 BRMS への新規ユーザーの追加

1. ナビゲーションパネルから **Administration** を選択してから、**User Permissions** を選択します。
2. **Create new user mapping** ボタンをクリックして、ユーザーマッピングを追加します。ユーザー名を表示されたダイアログボックスに入力して、**OK** をクリックします。

注記

ロールに指定されたユーザー名は、認証サービスのユーザー名と一致する必要があり、一致していない場合は、機能しません。

バグを報告する

4.2.3. ユーザーパーミッションの管理

手順4.5 ユーザーパーミッションの管理

1. **navigation pane** から **Administration** を選択してから、**User Permissions** を選択します。
2. ユーザー名の横の **Open** をクリックします。
3. ユーザーパーミッションの割り当ては、プラスのアイコンをクリックして、**Permission type** ドロップダウンメニューから該当のパーミッションを選択します。**OK** を押して確定します。
4. **ユーザーパーミッションの削除**
   ユーザーパーミッションの削除は、削除したいパーミッションの横にあるマイナスアイコンをクリックし、OK で確定します。

管理者ロールを割り当てられたユーザーは、他のロールやパーミッションを変更することができます。

バグを報告する

4.3. ルールパッケージの署名

4.3.1. ルールパッケージの署名

ルールパッケージはデフォルトで署名されていませんが、Red Hat は実稼働環境ではルールパッケージの署名を有効にするよう推奨しています。ルールパッケージの署名は、JBoss Enterprise BRMS Platform サーバーからクライアントアプリケーションにダウンロード中に、パッケージが壊れたり、改ざんされたりできないようにします。

ルールパッケージの署名は、公開鍵の暗号化で実装します。パッケージは秘密鍵で署名され、一致する証明書でなければ照合できません。秘密鍵はキーストアに格納されており、サーバーが自動的に各パッケージを署名する際に使用されます。公開証明書は、トラストストアと呼ばれるキーストアにあり、クライアントアプリケーションが利用できるようになっています。トラストストアの証明書を止揚して、署名パッケージの信ぴょう性を確認します。ダウンロード時に壊れたまたは変更されたルールパッケージは、証明書が合致しないため、クライアントにより却下されます。

バグを報告する

4.3.2. ルールパッケージ署名用のサーバー設定

このプロセスでルールパッケージ署名の設定を行う前に、以下を行う必要があります。

- 秘密署名鍵と対応する公開デジタル証明書の作成
- キーストアに秘密署名鍵と対応の公開デジタル証明書を置き、サーバーから入手できるようにする
- キーストアを使用するサーバーの設定

手順4.6 ルールパッケージ署名の設定

1. **keytool** コマンドを使用して、秘密キーストアを作成します。

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

-alias パラメーターは、キーストアの関連のエンティティをリンクするために使用する名前を指定します。こちらの手順では、それぞれ同じエイリアスを使用してください。エイリアスは大文字、小文字の区別はありません。**-keystore** パラメーターは、秘密キーを保持するために作成されるファイル名を渡します。

keytool は、ID 情報と 2 つのパスワードを入力するようプロンプトを出します。最初のパスワードは *キーストアのパスワード*で、キーストアをセキュアに保ち、 2 つ目のパスワードはキーパスワードで作成したキーをセキュアに保ちます。

```
[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. **keytool** コマンドを使用してデジタル署名を作成します。

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

   以前の手順と同じエイリアスとキーストアを使用します。**-file** パラメーターは作成される新規証明書のファイル名を、**-keystore** パラメーターは秘密キーストアのファイル名を渡します。

   プロンプトでキーストアのパスワードを入力します。

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

3. **keytool** コマンドを使用して、デジタル証明書をキーストアにインポートします。

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

   これにより、新たなキーストア、トラストストアが作成され、デジタル証明書が含まれるようになります。クライアントアプリケーションは、このトラストストアからデジタル証明書を取得できます。

```
[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. この秘密キーストアは、JBoss Enterprise BRMS Platform サーバーのみがアクセスできるセキュアな場所に保存する必要があります。たとえば、同じマシン上か、このマシンからアクセス可能なセキュアなネットワーク上などです。

重要

JBoss Enterprise BRMS Platform はネットワークリソースに認証情報を提供できません。秘密キーストアがネットワーク上のセキュアな場所に保存されている場合、JBoss Enterprise BRMS サーバーの代わりに認証の手順を実行して、秘密キーストアが入手できるようにする必要があります。例えば、JBoss Enterprise BRMS Platform サーバーがアクセスできるローカルディレクトリとして秘密キーストアを格納するファイル共有を、オペレーティングシステムが認証してマウントできるなどです。

5. クライアントアプリケーションからトラストストアにアクセスできるようにしておく必要があります。トラストストアをネットワーク共有や Web サーバー上にホストするなどして、アクセスできるようにします。

6. Drools シリアル化システムのプロパティはサーバー上に設定する必要があります。これらのプロパティは、キーストアにアクセスする際に必要な情報を格納するプロパティです。JBoss Enterprise BRMS Platform にはクライアントのコンポーネントも含まれているため、秘密キーストアとトラストストアのプロパティをサーバー上に設定する必要があります。これらのプロパティは一箇所で設定するだけでよく、どこに設定されていても同じアプリケーションサーバーインスタンス上で実行しているアプリケーションであればどれでも利用できます。

   シリアル化プロパティの設定は、**server/*profile*/deploy/jboss-brms.war/WEB-INF/** にある **preferences.properties** ファイルを以下のプロパティを含めるように編集します。

```
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/PublicBRMS.keystore
drools.serialization.public.keyStorePwd=keypassgoeshere
```

7. キーストアパスワードは、プレーンテキストで現在保存されています。

   キーストアの認証情報の暗号化に関する説明は、https://access.redhat.com/kb/docs/DOC-47247 を参照してください。

バグを報告する

4.3.3. ルールパッケージ署名用のクライアント設定

以下の手順は、ルールパッケージ署名のクライアント設定に関するプロセスを説明しています。ここでは、クライアントに複数のプロパティを設定します。**System.setProperty** メソッドを使用して、プログラムから設定することができます。**org.drools.core.util.KeyStoreHelper** クラスには、プロパティを表現する定数が複数含まれています。

このタスクを行う前に、以下が必要です。

- JBoss Enterprise BRMS Platform がすでにインストールされており、正しくルールパッケージ署名が設定されていること
- JBoss Enterprise BRMS Platform サーバーが使用するデジタル証明書を含むトラストストアの URL
- トラストストアのパスワード (設定されている場合)

手順4.7 ルールパッケージ署名のクライアント設定

1. 署名の有効化は、**drools.serialization.sign** プロパティを **true** にします。

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

2. **drools.serialization.public.keyStoreURL** プロパティにトラストストアが置かれる URL を設定します。トラストストアがクライアントのクラスパスにある場合、これは **getClass().getResource()** メソッドを使用して実行できます。

例4.1 トラストストアがクライアントのクラスパスにある場合：

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

例4.2 トラストストアが Web サーバーにある場合：

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

例4.3 トラストストアがローカルのファイルシステムにある場合：

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

3. **drools.serialization.public.keyStorePwd** プロパティにトラストストアのパスワードを設定します。トラストストアにアクセスする際にパスワードが必要な場合のみ必要です。

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

バグを報告する

第5章 ロギング

5.1. ロギング

JBoss Enterprise BRMS Platform には **log4j** 提供のロギング機能が含まれています。

WAR に含まれえている設定 **WEB-INF/classes/log4j.xml** は、全メッセージを **STDOUT** に送信します。**production** サーバープロファイルをデプロイする場合は、ログメッセージがサーバーログファイル ***PROFILE*/log/server.log** に追加されます。**default** サーバープロファイルにデプロイする場合は、ログメッセージはサーバーコンソールに表示されます。

バグを報告する

5.2. ロギングの設定

ロギングの動作を変更するには、該当のサーバープロファイルの **log4j** 設定ファイル (**jboss-as-web/server/*PROFILE*/conf/jboss-log4j.xml**) を編集します。

以下の XML を **default**サーバープロファイルの **log4j** 設定に追加して、 STDOUT メッセージをサーバーログファイルにダイレクトする新規カテゴリを作成します。

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

バグを報告する

5.3. 余分なログメッセージの削除

デフォルトでは JackRabbit JCR は以下の INFO ログメッセージを生成します。

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

JBoss Enterprise BRMS Platform 5.3.1 のスタンドアローンパッケージは、デフォルトではこれらのメッセージを表示しないように設定されていますが、JBoss Enterprise BRMS 5.3.1 のデプロイ可能パッケージを既存のアプリケーションサーバーにインストールされた方は、このメッセージを表示しないように設定する必要があります。

JBoss Enterprise Application Platform 5 および JBoss Enterprise SOA 5 をお使いの場合は、以下の XML スニペットを **jboss-as/server/*profile*/conf/jboss-log4j.xml** に追加して、このメッセージを削除します。

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

バグを報告する

第6章 カスタマイズ

6.1. 利用可能な言語

JBoss Enterprise BRMS Platform の Web ユーザーインターフェースは、複数の言語で表示することができます。

- アメリカ英語 (**en-US**)
- 日本語 (**ja-JP**)
- 簡体中国語 (**zh-CN**)

言語が指定されていない場合は、デフォルトのアメリカ英語が使用されます。

バグを報告する

6.2. ユーザーインターフェースの言語変更

手順6.1 ユーザーインターフェースの言語変更

1. **jboss-brms.war** ディレクトリにある **index.jsp** ファイルを編集して、別のロケールを指定します。デフォルト **index.jsp** ファイルでは、言語にアメリカ英語が使用されています。

   例6.1 デフォルトの index.jsp

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

   デフォルトの言語として日本語を指定するには、ファイルを以下のように編集します。

   例6.2 デフォルト言語が日本語の index.jsp

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

   デフォルトの言語として簡体中国語を指定するには、ファイルを以下のように編集します。

   例6.3 デフォルト言語が簡体中国語の index.jsp

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

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

2. **preferences.properties の編集**
   preferences.properties ファイルを編集してデフォルトとしたい国、言語コードを指定します。**preferences.properties** ファイルは **jboss-brms.war/WEB-INF/classes/** ディレクトリにあります。デフォルトの **preferences.properties** ファイルでは、言語にアメリカ英語、デフォルトの国にアメリカが使用されています。

   例6.4 デフォルトが US (米国) の preferences.properties

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

   デフォルトの言語として日本語を指定するには、ファイルを以下のように編集します。

   例6.5 デフォルトが日本の preferences.properties

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

   デフォルトの言語として簡体中国語を指定するには、ファイルを以下のように編集します。

   例6.6 デフォルトが簡体中国語の preferences.properties

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

3. **日付形式の変更**
   JBoss BRMS ユーザーインターフェースで複数のインスタンスに対して別の言語を指定してデプロイされる場合、各インスタンスが日付を正しく変換できるように、日付の形式を変更する必要があります。形式の変更をすると既存の日付の値を壊す可能性があるため、以下のプロシージャーは、既存のルールが含まれていない JBoss Enterprise BRMS の新規インストレーションか、日付の値が含まれているテストシナリオのいずれかでのみ実行可能です。

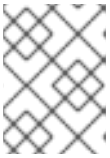
   preferences.properties の日付形式の指定

   日付形式を変更します。

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

   から以下へ

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



注記

形式の変更は、JBoss Enterprise BRMS サーバーを再起動するまで有効にはなりません。

別の言語はセッションベースで、locale パラメーターを含む URL を入力することで手動指定が可能です。

対応言語の完全 URL

日本語

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

簡体中国語

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

アメリカ英語

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

バグを報告する

6.3. UTF-8 エンコーディングでの JVM の実行

JBoss Enterprise BRMS は、UTF-8 エンコーディングと連携するよう設計されています。JVM で別のエンコーディング形式が使用されている場合は、予期せぬエラーが発生する可能性があります。

UTF-8 が JVM で確実に使用されるように、 JVM オプション **"-Dfile.encoding=UTF-8"** を使用してください。

バグを報告する

6.4. ユーザーインターフェースのカスタマイズ

BRMS ユーザーインターフェースは、*GWT フレームワーク*により動的に生成されます。ユーザーインターフェースの外観は、画像や CSS スタイルシートを編集して、ブランディングや統合目的でカスタマイズすることが可能です。

`.css` ファイルや一部の画像は、**`jboss-brms.war/org.drools.guvnor.Guvnor/`** ディレクトリにあります (残りの画像は **`images`** サブディレクトリにあります)。これらのファイルや画像を使用するには、『BRMS スタートガイド』に記載されている通り、展開アーカイブとして WAR ファイルをデプロイする必要があります。

画像と **CSS** ファイルを編集または置き換え、ファイル名は変更せずそのままにします。問題が発生した場合、**WAR** アーカイブからのオリジナルバージョンのファイルをリストアします。

注記

Red Hat は、簡単にメンテナンスが行えるように変更ファイルをバージョン管理システムに追加するよう推奨しています。

最も一般的に行われる変更は、画面上部のロゴや "site favorites" アイコン (それぞれ順に **`hdrlogo_brms.gif`** と **`drools.gif`**) などのブランディング画像の置き換えです。

`Guvnor.css` は、ページ要素の全体的なスタイルを管理します。

警告

GWT コンポーネントは、複数の CSS 追加ファイルを使用しますが、これらは変更しないようにしてください。

BRMS Platform が使用する URL のカスタマイズは、**`jboss-brms.war/WEB-INF/web.xml`** という配備記述子を変更します。他の Java Web アプリケーションについても同じプロセスを使用します。

バグを報告する

第7章 モニタリング

7.1. ナレッジベースとナレッジセッションのモニタリング

JBoss Enterprise BRMS Platform のナレッジベースとナレッジセッションは、JBoss Operations Network でモニタリングが可能です。JBoss BRMS の JBoss Operations Network と JBoss Operations Network のプラグインは、https://access.redhat.com からダウンロードいただけます。

JBoss Enterprise BRMS の JBoss ON サーバーとプラグインのインストール方法については、『JBoss Operations Network インストールガイド』を参照してください。

JBoss Operations Network がナレッジベースとナレッジセッションのモニタリングを行えるように、MBean を有効にする必要があります。

MBean を有効にするには、パラメーターを渡すか、

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

API 経由で行なってください。

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

バグを報告する

7.2. DROOLS ナレッジベースのモニタリングサービス

Drools ナレッジベースのモニタリングサービスは、接続プロパティとしてナレッジベース ID を渡すことでナレッジベースの設定が可能です。

Drools ナレッジベースのモニタリングサービスでは、以下の操作が可能です。

- すべての内部 MBean を開始
- すべての内部 MBean を停止

バグを報告する

7.3. DROOLS ナレッジセッションのモニタリングサービス

Drools ナレッジセッションのモニタリングサービスは、接続プロパティとしてナレッジベース ID とナレッジセッション ID を渡すことでナレッジベースとナレッジセッションの設定が可能です。

Drools ナレッジセッションのモニタリングサービスでは、以下の操作が可能です。

- 全メトリクスカウンターのリセット
- 特定のルールに関する統計を返す
- 特定のプロセスに関する統計を取得

- 特定のプロセスインスタンスに関する統計を取得

Drools ナレッジセッションのモニタリングサービスでは、以下のメトリクスを提供します。

- 作業メモリの合計ファクト数
- 最終リセットから作成された合計アクティベーション数
- 最終リセットから発生した合計アクティベーション数
- 最終リセットからキャンセルされた合計アクティベーション数
- 最終リセットからルールを出すまでにかかる合計時間
- 最終リセットから開始された合計プロセスインスタンス数
- 最終リセットから完了した合計プロセスインスタンス数
- 最終リセット操作のタイムスタンプ

バグを報告する

付録A APACHE ANT

A.1. APACHE ANT のインストール

JBoss ModeShape には Java 構築ツール *Apache Ant* と Trax プラグインが必要となります。JBoss Enterprise BRMS Platform の通常操作やインストールには必要ありません。Apache Ant には、正しくインストールされた Java Runtime Environment (JRE) が必要です。

開発ワークステーションを実行している場合、Apache Ant がすでにインストールされている可能性があります。Apache Ant の Web サイトからの Apache Ant パッケージには、Trax プラグインが含まれていますが、Red Hat Enterprise Linux には別パッケージとしてこのプラグインが含まれています。

手順A.1 Red Hat Enterprise Linux への Apache Ant のインストール

- 以下のコマンドを実行して、Red Hat Enterprise Linux レポジトリに Apache Ant と Trax プラグインをダウンロードして、インストールします。

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

手順A.2 他のオペレーティングシステムへの Apache Ant のインストール

1. **ダウンロードと展開**
   http://ant.apache.org/bindownload.cgi から **Apache Ant** バイナリリリースをダウンロードします。

   ダウンロードしてから、**`c:\Program Files\Apache\Ant\`** または **`/opt/apache-ant-1.8/`** などの、任意の場所に展開します。

2. **ANT_HOME 環境変数の追加**
   ANT_HOME と呼ばれる環境変数を作成します。この変数には、以前の手順で作成したパスを含める必要があります。

   - Red Hat Enterprise Linux では以下の行を **`~/.bash_profile`** ファイルに追加してください。パスは前野手順で作成したパスで置き換えてください。

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

   - Microsoft Windows では、**スタート**メニューをクリックして、**コントロールパネル**を開き、**システム -> 詳細 -> 環境編集**を選択します。

     新しい変数を作成して **ANT_HOME** という名前をつけ、前の手順で作成したディレクトリを参照するように設定します。

3. **PATH に bin を含めます。**
   Ant インストレーションの **bin** ディレクトリを **PATH** 環境変数に追加します。

   - Unix/Linux システムでは、**ANT_HOME** 変数を設定した行の後に、以下の行を **`~/.bash_profile`** ファイルに追加するだけです。

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

- Microsoft Windows では、**コントロールパネル**を開き、**システム -> 詳細 -> 環境変数 -> システム変数**を選択します。**PATH** 変数を編集して、**;%ANT_HOME%\bin** のテキストを追加します。

Apache Ant インストレーションをテストするには、コマンドラインシェルから **ant -version** を実行します。以下のような出力になるはずです。

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

Apache Ant に関する詳細情報は、当プロジェクトの Web サイト http://ant.apache.org を参照してください。

バグを報告する

付録B 永続マネージャー設定

B.1. 永続マネージャーの設定例

ここでは、対応のデータベースに対する永続マネージャーの設定について、複数の例をあげています。当然ながら、JDBC URL や schemaObjectPrefix のように、お使いのデータベースに適した値と、設定値を置き換えてください。

重要

JBoss Enterprise BRMS Platform は、この一覧 http://www.jboss.com/products/platforms/brms/supportedconfigurations/ に記載のデータベースで対応しています。

Apache Jackrabbit 永続マネージャーの追加情報は、http://wiki.apache.org/jackrabbit/PersistenceManagerFAQ を参照してください。

例B.1 BundleDbPersisenceManager を使用した MySQL の一般的な JDBC 設定

```
<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>
```

例B.2 MySqlPersistenceManager を使用した MySQL 設定

```
<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>
```

例B.3 OraclePersistenceManager を使用した Oracle 設定

```
<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>
```

例B.4 PostgreSQLPersistenceManager を使用した PostgreSQL 設定

```
<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>
```

例B.5 MSSqlPersistenceManager を使用した Microsoft SQL Server 2005 設定

```
<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>
```

バグを報告する

付録C CAMEL 統合

C.1. CAMEL 統合の設定例

アプリケーションサーバー内で Camel と JBoss BRMS を実行する場合、Web アプリをバンドルして、クラスパスにある CXF の別バージョンの依存関係間でコンフリクトが起こらないようにする必要があります。

web アプリケーションには、**Drools-camel-server/src/main/resources/** ディレクトリに以下のファイルを含める必要があります。

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

例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>
```

例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>
```

例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>
```

例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>
```

以下に、テスト目的で **test.drl** の例を提示しています。

例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
```

web アプリケーションには、**Drools-camel-server/src/main/webapp/WEB-INF/** ディレクトリに **web.xml** ファイルも含める必要があります。

例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>
```

Apache Camel に関する詳細情報は、『Apache Camel Documentation』 を参照してください。

バグを報告する

付録D 改訂履歴

改訂 5.3.1-1.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

改訂 5.3.1-1 **Thu Mar 07 2013** **L Carlon**

Built from Content Specification: 12749, Revision: 342874 by lcarlon