



Red Hat JBoss BRMS 6.2

Administration And Configuration Guide

The Administration and Configuration Guide for Red Hat JBoss BRMS

Red Hat JBoss BRMS 6.2 Administration And Configuration Guide

The Administration and Configuration Guide for Red Hat JBoss BRMS

Kanchan Desai
kadesai@redhat.com

Doug Hoffman

Eva Kopalova

Red Hat Content Services

Gemma Sheldon
Red Hat Engineering Content Services
gsheldon@redhat.com

Joshua Wulf
jwulf@redhat.com

Legal Notice

Copyright © 2015 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

A guide for administrators and advanced users dealing with Red Hat JBoss BRMS setup, configuration, and advanced usage.

Table of Contents

CHAPTER 1. ASSET REPOSITORY	4
1.1. CREATING AN ORGANIZATIONAL UNIT	4
1.2. CREATING A REPOSITORY	6
1.3. CLONING A REPOSITORY	8
1.4. REMOVING A REPOSITORY	11
1.5. MANAGING ASSETS	11
1.6. MAVEN REPOSITORY	17
1.7. CONFIGURING DEPLOYMENT TO A REMOTE NEXUS REPOSITORY	18
1.8. SYSTEM CONFIGURATION	19
CHAPTER 2. BUSINESS CENTRAL CONFIGURATION	21
2.1. ACCESS CONTROL	21
2.2. BRANDING THE BUSINESS CENTRAL APPLICATION	21
2.3. EXTENDING BUSINESS CENTRAL	24
2.4. CONFIGURING TABLE COLUMNS	30
CHAPTER 3. LOGGING	33
3.1. LOGBACK FUNCTIONALITY	33
3.2. CONFIGURING LOGGING	33
CHAPTER 4. REPOSITORY HOOKS	35
4.1. CONFIGURING GIT HOOKS	35
CHAPTER 5. COMMAND LINE CONFIGURATION	38
5.1. STARTING THE KIE-CONFIG-CLI TOOL IN ONLINE MODE	38
5.2. STARTING THE KIE-CONFIG-CLI TOOL IN OFFLINE MODE	38
5.3. COMMANDS AVAILABLE FOR THE KIE-CONFIG-CLI TOOL	39
CHAPTER 6. MIGRATION	41
6.1. DATA MIGRATION	41
6.2. API AND BACKWARDS COMPATIBILITY	42
PART I. INTEGRATION	44
CHAPTER 7. INTEGRATING RED HAT JBOSS BRMS WITH RED HAT JBOSS FUSE	45
7.1. CORE JBOSS BPM SUITE AND JBOSS BRMS FEATURES	45
7.2. ADDITIONAL FEATURES FOR SWITCHYARD AND CAMEL INTEGRATION	46
7.3. INSTALL/UPDATE CORE INTEGRATION FEATURES	47
7.4. INSTALL ADDITIONAL INTEGRATION FEATURES	48
7.5. CONFIGURING DEPENDENCIES	49
7.6. INSTALL JBOSS FUSE INTEGRATION QUICKSTART APPLICATIONS	49
CHAPTER 8. INTEGRATION WITH SPRING	52
8.1. CONFIGURING RED HAT JBOSS BRMS WITH SPRING	52
CHAPTER 9. LOCALIZATION AND CUSTOMIZATION	54
9.1. AVAILABLE LANGUAGES	54
9.2. CHANGING LANGUAGE SETTINGS	54
9.3. RUNNING THE JVM WITH UTF-8 ENCODING	54
9.4. CONFIGURING TABLE COLUMNS	54
CHAPTER 10. PROCESS EXECUTION SERVER CONFIGURATION	57
10.1. ASSIGNMENT RULES	57
10.2. MAIL SESSION	58

CHAPTER 11. MONITORING 60

11.1. JBOSS OPERATIONS NETWORK 60

11.2. DOWNLOADING RED HAT JBOSS 60

11.3. INSTALLING THE JBOSS BRMS PLUG-IN INTO JBOSS ON 60

11.4. MONITORING KIE BASES AND KIE SESSIONS 61

11.5. THE JBOSS RULES KIE BASE MONITORING SERVICE 61

11.6. THE JBOSS RULES KIE SESSION MONITORING SERVICE 61

APPENDIX A. REVISION HISTORY 63

CHAPTER 1. ASSET REPOSITORY

Business Rules and other assets and resources created in Business Central are stored in asset repository, which is otherwise known as the Knowledge Store.

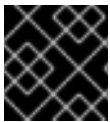
Knowledge Store is a centralized repository for your business knowledge. It connects with the Git repository that allows you to store different kinds of knowledge assets and artifacts at a single location. Business Central provides a web front-end that allows users to view and update the stored content. You can access it using the **Project Explorer** from the unified environment of Red Hat JBoss BRMS.

All business assets are stored in repositories. These repositories are then saved in directories called organizational units. By default, the Artifact repository does not contain any organizational unit. Therefore, to be able to create your own business assets, you need to create an organizational unit and a repository first.

1.1. CREATING AN ORGANIZATIONAL UNIT

Is possible to create an organizational unit either in the **Administration** perspective of Business Central, using the **kie-config-cli** tool or the REST API calls.

Creating an Organizational Unit in Business Central



IMPORTANT

Note that only users with the **admin** role can create organizational units.

Procedure 1.1. Using Business Central to Create an Organizational Unit

1. In Business Central, go to **Authoring** → **Administration**.
2. On the perspective menu, click **Organizational Units** → **Manage Organizational Units**.
3. In the **Organization Unit Manager** view, click **Add**. The **Add New Organizational Unit** dialog window opens.

Add New Organizational Unit ×

Organizational Unit Information * is required

* Name

* Default Group ID

 ?

Owner

Figure 1.1. Add New Organizational Unit Dialog Window

4. Enter the two mandatory parameters (name and default group ID) and click **Ok**.

Creating an Organizational Unit Using the `kie-config-cli` Tool

Organizational units can be created using the `kie-config-cli` tool as well. To do so, run the `create-org-unit` command. The tool then guides you through the entire process of creating an organizational unit by asking for other required parameters. Type `help` for a list of all commands.

For more information about the `kie-config-cli` tool, see [Chapter 5, Command line configuration](#).

Creating an Organizational Unit Using the REST API

To create an organizational unit in Knowledge Store, issue the **POST** REST API call. Details of the organizational unit are defined by the JSON entity.

Input parameter of the call is a `OrganizationalUnit` instance. Call returns a `CreateOrganizationalUnitRequest` instance.

Example 1.1. Creating an Organizational Unit Using the Curl Utility

Example JSON entity containing details of an organizational unit to be created:

```
{
  "name"      : "helloWorldUnit",
  "description" : "Organizational unit for the helloworldrepo
repository.",
  "owner"     : "tester",
  "repositories" : ["helloworldrepo"]
}
```

Execute the following command:

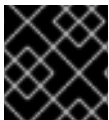
```
curl -X POST 'localhost:8080/business-central/rest/organizationalunits/'
-u USERNAME:PASSWORD -H 'Accept: application/json' -H 'Content-Type:
application/json' -d
'{"name":"helloWorldUnit","description":"Organizational unit for the
helloworldrepo repository.","owner":"tester","repositories":
["helloworldrepo"]}'
```

For further information, refer to the *Red Hat JBoss BPM Suite Development Guide*, chapter [Knowledge Store REST API](#), section *Organizational Unit Calls*.

1.2. CREATING A REPOSITORY

There are three ways to create a repository: using the **Administration** perspective of Business Central, the **kie-config-cli** tool or the REST API calls.

Creating a Repository in Business Central



IMPORTANT

Note that only users with the **admin** role can create repositories.

Procedure 1.2. Using Business Central to Create a Repository

1. In Business Central, go to **Authoring** → **Administration**.
2. On the perspective menu, click **Repositories** → **New repository**.
3. The **New Repository** pop-up window is displayed.

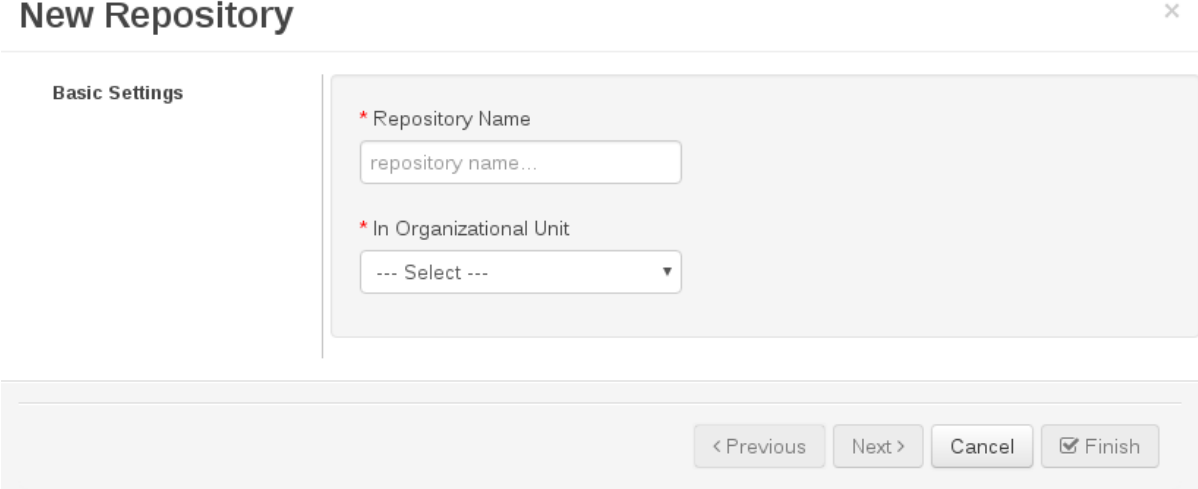


Figure 1.2. New Repository Dialog Window

4. Specify the two mandatory parameters:

- repository name



NOTE

Make sure that the repository name is a valid file name. Avoid using a space or any special character that might lead to an invalid name.

- organizational unit: specifies the location of the newly created repository.

5. Click **Finish**.

The new repository can be viewed either in the **File Explorer** or **Project Explorer** views.

Creating a Repository Using the `kie-config-cli` Tool

To create a new Git repository using the `kie-config-cli` tool, run the `create-repo` command. The tool then guides you through the entire process of creating a repository by asking for other required parameters. Type `help` for a list of all commands.

For more information about the `kie-config-cli` tool, see [Chapter 5, Command line configuration](#).

Creating a Repository Using the REST API

To create a repository in the Knowledge Store, issue the **POST** REST API call. Details of the repository are defined by the JSON entity. Make sure you established an authenticated HTTP session before executing this call.

Input parameter is a **RepositoryRequest** instance. Returns a **CreateOrCloneRepositoryRequest** instance.

Example 1.2. Creating a Repository Using the Curl Utility

Example JSON entity containing details of a repository to be created:

```
{
  "name"           : "newRepository",
  "description"    : "Repository for the Hello World project.",
```

```
"userName"           : null,  
"password"           : null,  
"gitURL"             : null,  
"requestType"        : "new",  
"organizationalUnitName" : "helloWorldUnit"  
}
```

Execute the following command:

```
curl -X POST 'localhost:8080/business-central/rest/repositories/' -u  
USERNAME:PASSWORD -H 'Accept: application/json' -H 'Content-Type:  
application/json' -d '{"name":"newRepository","description":"Repository  
for the Hello World  
project.", "username":null, "password":null, "requestType":"new", "gitURL":n  
ull, "organizationalUnitName":"helloWorldUnit"}'
```

For further information, refer to the *Red Hat JBoss BPM Suite Development Guide*, section [Repository Calls](#).

1.3. CLONING A REPOSITORY

It is possible to clone a repository either in Business Central or using the REST API calls. The **kie-config-cli** tool cannot be used to clone arbitrary repositories - run **git clone** or one of the following options instead.

Cloning a Repository in Business Central



IMPORTANT

Note that only users with the **admin** role can clone repositories.

Procedure 1.3. Using Business Central to Clone a Repository

1. In Business Central, go to **Authoring** → **Administration**.
2. On the perspective menu, choose **Repositories** → **Clone repository**.
3. The **Clone Repository** pop-up window is displayed.

Clone Repository



Repository Information * is required

* Repository Name

* Organizational Unit

* Git URL

User Name

Password

Figure 1.3. Clone Repository Dialog Window

4. In the **Clone Repository** dialog window, enter the repository details:
 - a. Enter the **Repository Name** to be used as the repository identifier in the Asset repository and select the **Organizational Unit** it should be added to.
 - b. Enter the URL of the Git repository:
 - for a local repository, use `file:///PATH_TO_REPOSITORY/REPOSITORY_NAME`;
 - for a remote or preexisting repository, use `https://github.com/USERNAME/REPOSITORY_NAME.git` or `git://HOST_NAME/REPOSITORY_NAME`.

**IMPORTANT**

It is important to use the HTTPS or Git protocol instead of a SCP-style SSH URL. Business Central does not support the basic SSH URL and fails with *Invalid URL format*.

**NOTE**

The file protocol is only supported for READ operations. WRITE operations are *not* supported.

- c. If applicable, enter the **User Name** and **Password** of your Git account to be used for authentication.
5. Click **Clone**.
6. A confirmation prompt with the notification that the repository was created successfully is displayed. After clicking **Ok**, the repository is being indexed. Some workbench features may be unavailable until indexing has completed.

The cloned repository can be viewed either in the **File Explorer** or **Project Explorer**.

Cloning a Repository Using the REST API

To clone a repository, issue the **POST** REST API call. This call creates or clones (according to the value of the **requestType** parameter) the repository defined by the JSON entity.

Input parameter is a **RepositoryRequest** instance. Returns a **CreateOrCloneRepositoryRequest** instance.

Example 1.3. Cloning a Repository Using the Curl Utility

Example JSON entity containing details of a repository to be cloned:

```
{
  "name"           : "clonedRepository",
  "description"    : null,
  "userName"       : null,
  "password"       : null,
  "requestType"    : "clone",
  "gitURL"         : "git://localhost:9418/example-repository",
  "organizationalUnitName" : "helloWorldUnit"
}
```

Execute the following command:

```
curl -X POST 'localhost:8080/business-central/rest/repositories/' -u
USERNAME:PASSWORD -H 'Accept: application/json' -H 'Content-Type:
application/json' -d
'{"name":"clonedRepository","description":null,"username":null,"password
":null,"requestType":"clone","gitURL":"git://localhost:9418/example-
repository","organizationalUnitName":"helloWorldUnit"}'
```

For further information, refer to the *Red Hat JBoss BPM Suite Development Guide*, section [Repository Calls](#).

1.4. REMOVING A REPOSITORY

Repositories can be removed using any of the following procedures.

Removing a Repository in Business Central

The simplest way to remove a repository is using the **RepositoryEditor** in Business Central.

Procedure 1.4. Using Business Central to Remove a Repository

1. In Business Central, go to **Authoring** → **Administration**.
2. Select **Repositories** from the tree menu on the left.
3. In the **RepositoryEditor** on the right side of the page, locate the repository you want to delete from the list of available repositories.
4. From the drop-down menu, select **master** and click **Delete**.
5. The following message will appear:

Are you sure you want to remove Repository "REPOSITORY_NAME"? Some editors may become inoperable if their content is inaccessible.

Press **OK** to delete the repository.

Removing a Repository Using the `kie-config-cli` Tool

Repositories can be removed using the `kie-config-cli` tool as well. To do so, run the `remove-repo` command.

For further information about the `kie-config-cli` tool, see [Chapter 5, Command line configuration](#).

Removing a Repository Using the REST API

To remove a repository from the Knowledge Store, issue the **DELETE** REST API call. Make sure you established an authenticated HTTP session before executing this command.

Returns a **RemoveRepositoryRequest** instance.

Example 1.4. Removing a Repository Using the Curl Utility

Execute the following command:

```
curl -X DELETE 'localhost:8080/business-
central/rest/repositories/REPOSITORY_NAME' -u USERNAME:PASSWORD -H
'Accept: application/json' -H 'Content-Type: application/json'
```

More information about repository calls to the Knowledge Store can be found in *Red Hat JBoss BPM Suite Development Guide*, section [Repository Calls](#).

1.5. MANAGING ASSETS



NOTE

The content in this section is classified as Technical Preview for the 6.1 release of Red Hat JBoss BRMS. It is provided as is and no support is provided.

To activate and use the featured described here, you will need to login to Business Central with a user that has been given the special role of **kiemgmt**.

To make management of projects easier, Red Hat JBoss BRMS now provides a way to manage multiple projects based on standards. This allows you to create repository structures using industry standard best practices for maintenance, versioning and distribution of your projects.

To start with, repositories can now be managed or unmanaged.

Managed and Unmanaged Repositories

Unmanaged Repositories are the repository structures that you are used to. They can contain multiple unrelated projects.

Managed Repositories, on the other hand, provide version control at the project level and project branches for managing the release cycle. Further, Managed Repositories can be restricted to just a single project or encompass multiple projects. When a Managed Repository is created the asset management configuration process is automatically launched in order to create the repository branches, and the corresponding project structure is also created.

To create a Managed or Unmanaged Repository, open up the screen for creating a new repository. This is achieved by going to **Authoring** → **Administration** and then clicking on **Repositories** → **New Repository**. This will bring up the **New Repository** screen.

New Repository

Basic Settings
Managed Repository Settings

* Repository Name

* In Organizational Unit

☒ Managed Repository
 A managed repository provides project-level version control and project branches for managing the release cycle.

The Unmanaged Repository creation is the same as before; Enter the name of the repository and select the organizational unit that it belongs to and click the **Finish** button.

To create a Managed Repository, select the **Managed Repository** checkbox, after giving the repository a name and the organizational unit it belongs to. Click the **Next** button to enter details of this Managed Repository.

New Repository ✕

✓ Basic Settings

✓ **Managed Repository Settings**

Repository Type:

☐ Single-project Repository
Create a single managed project in this repository. Use this option for simple or self-contained projects.

☒ Multi-project Repository
Integrate multiple projects to create a larger application. The projects in this repository will be managed together, and will all increment version numbers together.

Project Branches:

☒ Automatically Configure Branches (master/dev/release)

Project Settings:

* Name

Description

* Group

* Artifact

< Previous

Next >

Cancel

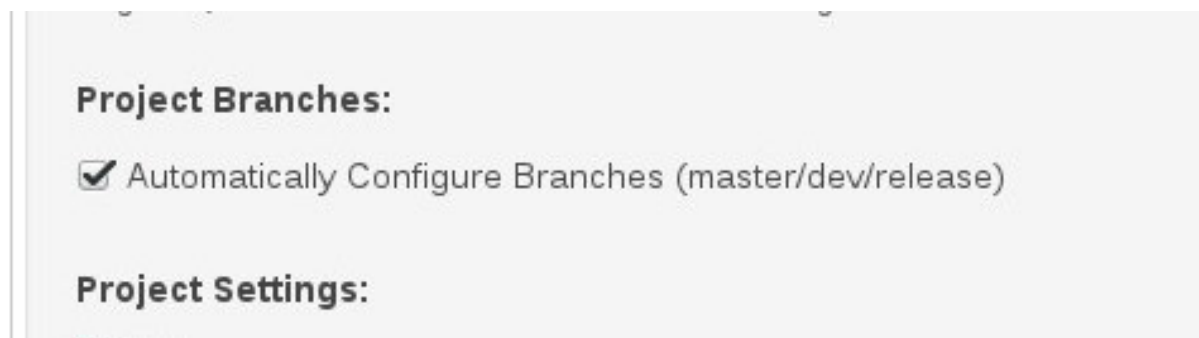
☒ Finish

Select the **Single Project** label if the project you are creating is a simple project and is self-contained. Enter the details of the managed project, along with the GAV details. You will not be able to add more projects to this repository later.

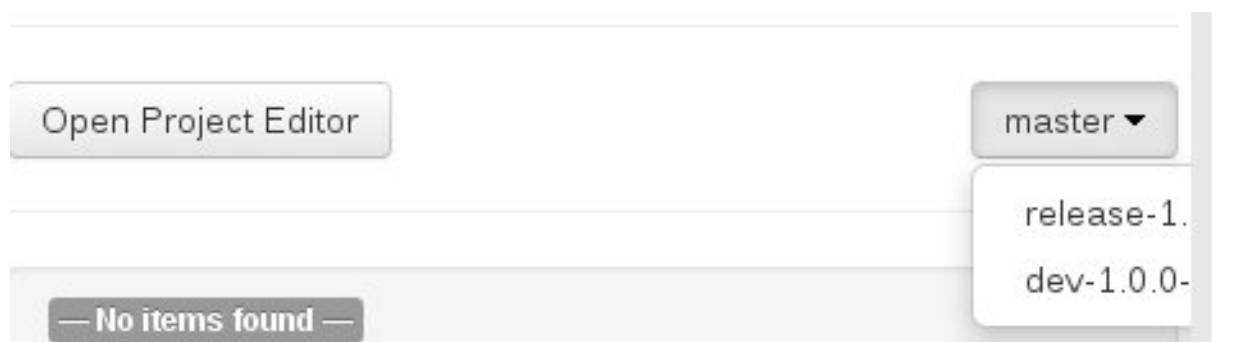
For more complex projects, where there is likely to be a parent project that encompasses other smaller projects, select the **Multi-Project** repository. All Projects created in a multi-project repository will be managed together, with their version numbers being incremented together as well. Also enter the details of the parent project and the GAV, which will be inherited by all future projects that you create in this Managed Repository.

Managed Branches

With Managed Repositories comes the added advantage of Managed Branches. As in GIT, you can choose to work on different branches of your project (for example: master, dev and release). This process of branching can also be automated for you, by selecting the checkbox while creating a new Managed Repository (for both single and multi-projects).

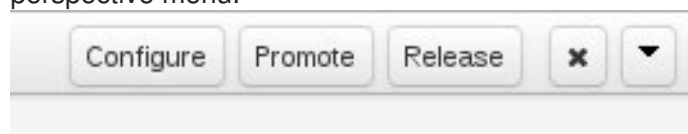


You can switch between branches by selecting the desired branch while working in the Project Explorer.



Repository Structure

If you don't select automatic branch management while creating a repository, you can create branches manually afterwards. For Managed Repositories, you can do so by using the **Configure** button. This button, along with **Promote** and **Release** buttons, is provided in the **Repository Structure** view. You can access this view, by clicking on **Repository** → **Repository Structure** in the Project Explorer perspective menu.



Clicking on the **Configure** button allows you to create branches or edit automatically created ones.

Configure Repository



Repository

ManagedRepo-Single-Manual

Source Branch

master

* Dev Branch

dev

The branch will be called (dev)-1.0.0-SNAPSHOT

* Release Branch

release

The branch will be called (release)-1.0.0-SNAPSHOT

* Version

1.0.0-SNAPSHOT

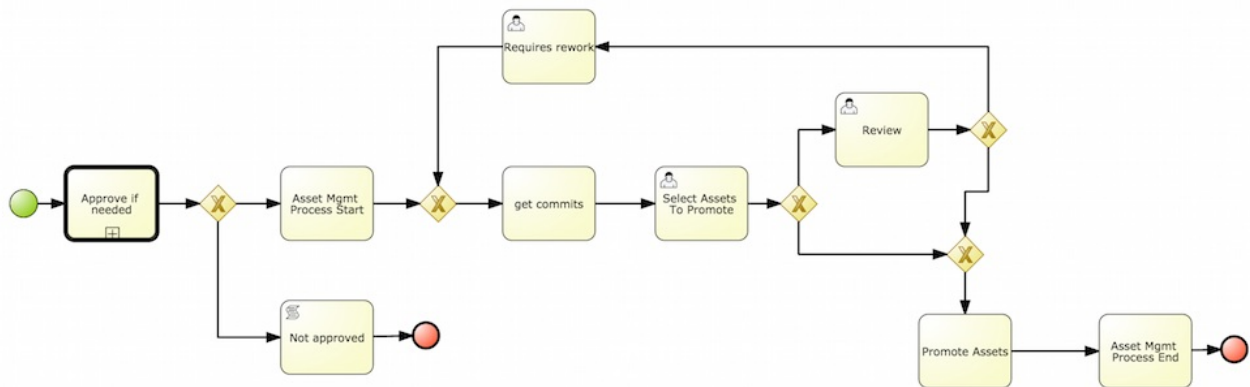
The current repository version is: 1.0.0-SNAPSHOT

+ Ok

Cancel

You can promote assets from the master branch to other branches using the **Promote** button. Similarly, you can Release branches and deploy them on the server using the **Release** button.

Both these functions are controlled internally by the use of pre-defined processes that are deployed on your instance. For example, when you click on **Promote** button after having done work on your development branch, a Promote Changes process is started in the background. A user, with the role of **kiemgmt** will have a user task appear in this task list to review the assets being promoted. This user can claim this task, and decide to promote all, some or none of the assets. The underlying process will cherry-pick the commits selected by the user to a release branch. This user can also request another review of these assets and this process can be repeated multiple times till all the assets are ready for release. The flow for this process is shown below:



Similarly, when you click on the **Release** button, a release process flow is initiated. This process flow builds the project and updates all the Maven artifacts to the next version, and deploys the project to the runtime, if runtime deployment details are supplied.



WARNING

Project branches to be released, must start with the keyword **release**

Release Configuration



Repository

ManagedRepo2

Source Branch

release-1.0.0

* Release Version

1.0.0

The current repository version is: 1.0.0-SNAPSHOT

* Deploy To Runtime



* User Name

vikrambpms

* Password

••••••••

* Server URL

http://localhost:8080/business-ce

+ Ok

Cancel

1.6. MAVEN REPOSITORY

Maven is a software project management tool which uses a project object model (POM) file to manage:

- Builds
- Documentation
- Reporting

- Dependencies
- Releases
- SCMs
- Distribution

A Maven repository is used to hold or store the build artifacts and project dependencies and is generally of two types:

- Local: refers to a local repository where all the project dependencies are stored and is located with the current installation in the default folder as "m2". It is a cache of the remote downloads, and also contains the temporary build artifacts which have not yet been released.
- Remote: refers to any other type of repository that can be accessed by a variety of protocols such as file:// or http://. These repositories can be at a remote location set up by a third-party for downloading of artifacts or an internal repository set up on a file or HTTP server, used to share private artifacts between the development teams for managing internal releases.

1.7. CONFIGURING DEPLOYMENT TO A REMOTE NEXUS REPOSITORY

Nexus is a repository manager frequently used in organizations to centralize storage and management of software development artifacts. It is possible to configure your project so that artifacts produced by every build are automatically deployed to a repository on a remote Nexus server.

To configure your project to deploy artifacts to a remote Nexus repository, add a **distributionManagement** element to your project's **pom.xml** file as demonstrated in the code example below.

```
<distributionManagement>
  <repository>
    <id>deployment</id>
    <name>Internal Releases</name>

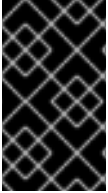
    <url>http://your_nexus_host:8081/nexus/content/repositories/releases</url>
  </repository>
  <snapshotRepository>
    <id>deployment</id>
    <name>Internal Releases</name>

    <url>http://your_nexus_host:8081/nexus/content/repositories/snapshots/</url>
  </snapshotRepository>
</distributionManagement>
```

Replace the URLs in the example with real URLs of your Nexus repositories. The repository specified in the **snapshotRepository** element is used when the **-SNAPSHOT** qualifier is appended to the project's current version number. In other cases the repository specified in the **repository** element is used.

If your Nexus server requires authentication, you will also need to modify your projects Maven settings to add your credentials in the **settings-security.xml** file, using a master password. By default, this file is in **~/.m2** folder, unless you have changed its location by modifying the **kie.maven.settings.custom** system property.

```
<servers>
  <server>
    <id>deployment</id>
    <username>admin</username>
    <password>admin.123</password>
  </server>
</servers>
```



IMPORTANT

Note that keeping your server authentication credentials (for example the passwords) as a plain text in the **settings.xml** file is *not* recommended. All the information should be hashed with a master password in the **settings-security.xml** file.

With this configuration in place, clicking the **Build & Deploy** button in Business Central executes a Maven build and deploys the built artifacts both to the local repository and to one of the Nexus repositories specified in the **pom.xml** file.

1.8. SYSTEM CONFIGURATION

In JBoss EAP, to change a Business Central property, such as the configuration for SSH, do the following:

Procedure 1.5. Changing System Properties

1. Edit the file **\$JBOSS_HOME/domain/configuration/host.xml**
2. Locate the XML elements `server` that belong to the `main-server-group` and add the system property. For example:

```
<system-properties>
  <property name="org.uberfire.nio.git.dir" value="..." boot-
time="false"/>
  ...
</system-properties>
```

Here is a list of all the available system properties:

- **org.uberfire.nio.git.dir**: Location of the directory `.niogit`. Default: working directory
- **org.uberfire.nio.git.daemon.enabled**: Enables/disables GIT daemon. Default: true
- **org.uberfire.nio.git.daemon.host**: If GIT daemon enabled, uses this property as the localhost identifier. Default: localhost
- **org.uberfire.nio.git.daemon.port**: If GIT daemon is enabled, uses this property as the port number. Default: 9418
- **org.uberfire.nio.git.ssh.enabled**: Enables/Disables SSH daemon. Default: true
- **org.uberfire.nio.git.ssh.host**: If SSH daemon is enabled, uses this property as the localhost identifier. Default: localhost

- **org.uberfire.nio.git.ssh.port**: If SSH daemon is enabled, uses this property as the port number. Default: 8001
- **org.uberfire.nio.git.ssh.cert.dir**: Location of the **.security** directory where local certificates will be stored. Default: working directory
- **org.uberfire.metadata.index.dir**: Location of the **.index** folder for Lucene. Default: working directory
- **org.uberfire.cluster.id**: Name of the Helix cluster, for example: kie-cluster
- **org.uberfire.cluster.zk**: Connection string to Zookeeper. This is of the form **host1:port1,host2:port2,host3:port3**. For example: **localhost:2188**.
- **org.uberfire.cluster.local.id**: Unique id of the Helix cluster node. Note that ':' is replaced with '_'. For example: node1_12345.
- **org.uberfire.cluster.vfs.lock**: Name of the resource defined on the Helix cluster, for example: kie-vfs
- **org.uberfire.cluster.autostart**: Delays VFS clustering until the application is fully initialized to avoid conflicts when all cluster members create local clones. Default: false
- **org.uberfire.sys.repo.monitor.disabled**: Disable configuration monitor (do not disable unless you know what you're doing). Default: false
- **org.uberfire.secure.key**: Secret password used by password encryption. Default: org.uberfire.admin
- **org.uberfire.secure.alg**: Crypto algorithm used by password encryption. Default: PBESWithMD5AndDES
- **org.guvnor.m2repo.dir**: Place where Maven repository folder will be stored. Default: working-directory/repositories/kie
- **org.kie.example.repositories**: Folder from where demo repositories will be cloned. The demo repositories need to have been obtained and placed in this folder. This system property takes precedence over org.kie.demo and org.kie.example properties. Default: Not used.
- **org.kie.demo**: Enables external clone of a demo application from GitHub. This system property takes precedence over org.kie.example. Default: true.
- **org.kie.example**: Enables example structure composed by Repository, Organization Unit and Project. Default: false

CHAPTER 2. BUSINESS CENTRAL CONFIGURATION

As Business Central is a web application, any configuration settings are loaded from **`DEPLOY_DIRECTORY/business-central.war/WEB-INF/web.xml`** and the referenced files, and if deployed on Red Hat JBoss EAP 6, also in **`jboss-web.xml`** and **`jboss-deployment-structure.xml`**.

Note that the entire application can be run in different profiles (refer to the *Red Hat JBoss BRMS Installation Guide*).

2.1. ACCESS CONTROL

Workbench Configuration

Within Red Hat JBoss BRMS, users may set up roles using LDAP to modify existing roles. Users may modify the roles in the workbench configuration to ensure the unique LDAP based roles conform to enterprise standards by editing the deployments directory located at **`$JBOSS_HOME/standalone/deployments/business-central.war/WEB-INF/classes/workbench-policy.properties`**.

If authenticating user via LDAP over GIT, administrators must set system property `org.uberfire.domain` to the name of login module it should use to authenticate users via the GIT service. This must be set in the **`standalone.xml`** file in EAP.

2.2. BRANDING THE BUSINESS CENTRAL APPLICATION

The Business Central web application enables you to customize its look and feel by allowing you to override some of its default styles. The ability to customize the Business Central branding allows you to get a consistent appearance across all your applications thereby improving the user experience. It also helps in cases when multiple teams are using the application. Each team can develop their own customized user interface. The customizable elements are built using cascading style sheets (CSS), images, and HTML files, providing an easy and flexible approach to customize without having to recompile the code.

You can modify the following elements in the Business Central application to make it inline with your company's brand:

- Login screen

You can customize the following attributes of the Business Central login screen:

- The background image
- The company logo
- The application logo

- Application header

You can customize the following attributes of the Business Central application header:

- The Business Central header containing the title and banner logo

- Help pop-up windows

You can customize the following attributes of the splash help pop-up windows:

- The splash help images
- The label text

2.2.1. Customizing Business Central Login Page

Procedure 2.1. Changing the Business Central Login Page Background Image

1. Start the EAP server and open <http://localhost:8080/business-central> in a web browser.
2. Copy the new background image to the **\$EAP_HOME/standalone/deployments/business-central.war/images** directory in your JBoss BRMS installation.
3. Navigate to **\$EAP_HOME/standalone/deployments/business-central.war/styles** directory and open the **login-screen.css** file in a text editor.
4. In the **login-screen.css** file, provide the location of your new background image in the following **background-image** attribute.

```
background-image: url("../images/login-screen-background.jpg");
```

The **background-image** attribute points to the default **login-screen-background.jpg** image.

In addition to the background image, you can modify other attributes such as image size, position, and background color in the **login-screen.css** file.

Refresh the Business Central login page to view your changes.

Procedure 2.2. Changing the Business Central Login Page Company Logo and Project Logo

1. Start the EAP server and open <http://localhost:8080/business-central> in a web browser.
2. Navigate to the **\$EAP_HOME/standalone/deployments/business-central.war/images** directory in your JBoss BRMS installation.
3. Replace the default image **login-screen-logo.png** with a new one. This is the company logo that appears on the top right hand corner of the login page.
4. Replace the default image with a new one. This is the project logo that appears on the center left hand side of the login page.

Refresh the Business Central login page to view your changes.

2.2.2. Customizing Business Central Application Header

Procedure 2.3. Changing the Business Central Application Header (Banner)

1. Start the EAP server and open <http://localhost:8080/business-central> in a web browser.
2. Log in to the Business Central application with your user credentials.

3. Copy your new application header image to the **\$EAP_HOME/standalone/deployments/business-central.war/banner** directory in your JBoss BRMS installation.
4. Open **\$EAP_HOME/standalone/deployments/business-central.war/banner/banner.html** file in a text editor.
5. In the **banner.html** file, edit the following `` tag to provide the name of your new header image:

```

```

The default image is **logo.png**.

Refresh the Business Central Home page to view your changes.

2.2.3. Customizing Business Central Splash Help Windows

The **\$EAP_HOME/standalone/deployments/business-central.war/plugins** directory contains the splash pages and the corresponding html files. Each splash page holds the name of the html file, which contains information about the image(s) and the text to be displayed. For example, the **authoring_perspective.splash.js** splash page points to the **authoring_perspective.splash.html** file. The **authoring_perspective.splash.html** contains the names and location of all the image files that appear on the Authoring Perspective splash help and also their captions. You can customize the images and the corresponding captions of the existing splash help pop-up windows.

Procedure 2.4. Changing the Business Central Splash Help Pop-Up Images and Captions

1. Start the EAP server and open <http://localhost:8080/business-central> in a web browser.
2. Log in to the Business Central application with your user credentials.
3. Copy your new splash help image(s) to the **\$EAP_HOME/standalone/deployments/business-central.war/images** directory in your JBoss BRMS installation.
4. Open the corresponding html file from **\$EAP_HOME/standalone/deployments/business-central.war/plugins** directory in a text editor.
5. Edit the html file to point to your new splash help image. For example, to change the first image that appears in the Authoring Perspective splash help, edit the following `` tag in the **authoring_perspective.splash.html** file to add your new image:

```

```

The default image is **authoring_perspective1.png**, which appears on the first page of the Authoring Perspective splash help.

6. To change the image caption that appears on the splash help, edit the `<h4>` and `<p>` tag contents below the `` tag:

```
<h4>Authoring</h4>
<p>Modularized and customizable workbench</p>
```

Refresh the Business Central Home page and access the splash help pop-up windows to view your changes.

2.3. EXTENDING BUSINESS CENTRAL

Starting with version 6.1 of JBoss BRMS, Business Central can be configured to add new screens, menus, editors, splashscreens and perspectives by the Administrator. These elements can extend functionality of Business Central and can be accessed through the **Extensions** menu and are classified under **Plugin Management**.

You can now define your own Javascript and HTML based plugins to extend Business Central and add them without having to worry about copying files in the underlying filesystem. Let's add a new screen in the system to show you the basics of this functionality.


2.3.1. Plugin Management

You access the **Plugin Management** screen by clicking on **Extensions** → **Plugin Management**. This brings up the **Plugin Explorer** screen that lists all the existing plugins under their respective categories: **Perspective Plugin**, **Screen Plugin**, **Editor Plugin**, **Splashscreen Plugin** and **Dynamic Menu**. Open up any of these and you will see the existing plugins in each category, including the uneditable system generated ones.

Let's create a new plugin that echoes "Hello World" when users visit the screen for that plugin. In general, the steps to creating a new plugin are:

- Create a new screen
- Create a new perspective (and add the new screen to it)
- Create a new menu (and add the new perspective to it)
- Apps (optional)

Adding a new Screen

Click on  button and select **New Screen**. You will be prompted to enter the name of this new screen. Enter "HelloWorldJS" and press the **OK** button. The Screen plugin editor will open up, divided into 4 sections: Template, CSS, JavaScript and Media.



NOTE

All manually created elements go into their respective categories in case you want to edit them later. In this case, to open up the Screen plugin editor again if you close it, open up the **Screen Plugin** category and scroll past the system generated screens to your manually created plugin and click on it to open up the Screen plugin editor again.

Template is where your HTML goes, CSS is for styling, JavaScript is for your functions and Media is for uploading and managing images.

Since we are making a simple Hello World plugin, enter the following code in the Template section: **<div>My Hello World Screen</div>**. This can be any HTML code, and you can use the supplied **Angular** and **Knockout** frameworks. For the purposes of this example, we are not using any of those frameworks, but you can choose to by selecting them from the drop down in the Template section.

Enter your JavaScript code in the JavaScript section. Some common methods and properties are defined for you, including **main**, **on_close** and **on_open**. For this demo, select the **on_open** and enter the following: **function () { alert('Hello World'); }**

Click the **Save** button to finish creating the screen. After you save the screen, refresh business central so that the Screen Plugin is listed in the Screen Component of Perspective plugin.

Adding a new Perspective

Once a screen has been created, you need to create a perspective on which this screen will reside. Perspectives can also be created similar to the way a screen is created by clicking on the New button and then selecting, **New Perspective**. You can now provide a name for this perspective, say **HelloWorldPerspective**. This will open up the Perspective plugin editor, similar to the Screen plugin editor. .

The Perspective Editor is like a drag and drop grid builder for screens and HTML components. Remove any existing grids and then drag a **6 6** grid on the right hand side to the left hand side.

Next, open up the **Components** category and drag a Screen Component on the right hand side to the left hand side (in any grid). This will open up the **Edit Component** dialog box that allows you to select the screen created in the previous step (**HelloWorldJS**). Click the **OK** button and then click the **Save** button to save this perspective. To tag your perspective, enter **Home** in the tag name field and click the **Tags** button. Click the **OK** button and save the changes.

You can open this perspective again from the Perspective plugins listed on the left hand side.

Adding a new menu

The final step in creating our plugin is to add a dynamic menu from where the new screen/perspective can be called up. To do so, go to **Extensions** → **Plugin Management** and then click on the New button to select **New Dynamic Menu**. Give this dynamic menu a name (HelloWorldMenu) and then click the **OK** button. The dynamic menu editor opens up.

Enter the perspective name (HelloWorldPerspective) as the **Activity Id** and the name for the drop down menu (HelloWorldMenuDropDown). Click **OK** and then click the **Save** button.

This new menu will be added to your workbench the next time you refresh Business Central. Refresh it now to see **HelloWorldMenu** added to your top level menu. Click on it to reveal **HelloWorldMenuDropDown** which when clicked will open up your perspective/screen with the message **Hello World**.

You have created your first Plugin!

Working with Apps (Optional)

If you create multiple plugins, you can use the Apps directory feature to organize your own components and plugins, instead of having to rely on just the top menu entries.

When you save a new perspective, you can add labels (tags) for them and these labels (tags) are used to associate a perspective with an App directory. You can open up the App directories by clicking on **Extensions** → **Apps**.

The Apps directory provides an alternate way to open up your perspective. When you created your **HelloWorldPerspective**, you entered the tag **Home**. The Apps directory by default contains a single directory called **Home** with which you associated your perspective. This is where you will find it when you open the Apps directory. You can click on it to run the perspective now.

You can create multiple directories and associate perspectives with those directories depending on functional and vertical business requirements. For example, you could create an HR directory and then associate all HR related perspectives with that directory to better manage Apps.



You can create a new directory by clicking on the New button

2.3.2. The JavaScript (JS) API for Extensions

The extensibility of Business Central is achieved by an underlying JavaScript (JS) API which is automatically loaded if it is placed in the **plugins** folder of the Business Central webapp (typically: {INSTALL_DIR}/business-central.war/plugins/) or it can be loaded via regular JavaScript calls.

This API is divided into multiple sets depending on the functionality it performs.

- Register Perspective API: allows for the dynamic creation of perspectives. The example below creates a panel using the **registerPerspective** method:

```
$registerPerspective({
  id: "Home",
  is_default: true,
  panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
  view: {
    parts: [
      {
        place: "welcome",
        min_height: 100,
        parameters: {}
      }
    ],
    panels: [
      {
        width: 250,
        min_width: 200,
        position: "west",
        panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
        parts: [
          {
            place: "YouTubeVideos",
            parameters: {}
          }
        ]
      },
      {
        position: "east",
        panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
        parts: [
          {
```

```

        place: "TodoListScreen",
        parameters: {}
      }
    ],
  },
  {
    height: 400,
    position: "south",
    panel_type:
"org.uberfire.client.workbench.panels.impl.MultiTabWorkbenchPanelPre
senter",
    parts: [
      {
        place: "YouTubeScreen",
        parameters: {}
      }
    ]
  }
]
});

```

- Editor API: allows you to dynamically create editors and associate them with a file type. The example below creates a sample editor and associates it with **filename** file type.

```

$registerEditor({
  "id": "sample editor",
  "type": "editor",
  "templateUrl": "editor.html",
  "resourceType":
"org.uberfire.client.workbench.type.AnyResourceType",
  "on_concurrent_update":function(){
    alert('on_concurrent_update callback')
  }

  $vfs_readAllString(document.getElementById('filename').innerHTML,
  function(a) {
    document.getElementById('editor').value= a;
  });
},
"on_startup": function (uri) {
  $vfs_readAllString(uri, function(a) {
    alert('sample on_startup callback')
  });
},
"on_open":function(uri){
  $vfs_readAllString(uri, function(a) {
    document.getElementById('editor').value=a;
  });
  document.getElementById('filename').innerHTML = uri;
}
});

```

In addition to **on_startup** and **on_open** methods seen in the previous example, the API exposes the following callback events for managing the editor's lifecycle:

- `on_concurrent_update;`
- `on_concurrent_delete;`
- `on_concurrent_rename;`
- `on_concurrent_copy;`
- `on_rename;`
- `on_delete;`
- `on_copy;`
- `on_update;`
- `on_open;`
- `on_close;`
- `on_focus;`
- `on_lost_focus;`
- `on_may_close;`
- `on_startup;`
- `on_shutdown;`

You can display this editor via an html template:

```
<div id="sampleEditor">
  <p>Sample JS editor (generated by editor-sample.js)</p>
  <textarea id="editor"></textarea>

  <p>Current file:</p><span id="filename"></span>
  <button id="save" type="button"
onclick="$vfs_write(document.getElementById('filename').innerHTML,
document.getElementById('editor').value,  function(a)
{});">Save</button>
  <br>

  <p>This button change the file content, and uberfire send a
callback to the editor:</p>
  <button id="reset" type="button"
onclick="$vfs_write(document.getElementById('filename').innerHTML,
'Something else',  function(a) {});">Reset File</button>
</div>
```

- PlaceManager API: the methods of this API allow you to request that the Business Central display a particular component associated with a target:
`$goToPlace("componentIdentifier");`
- Register plugin API: the methods of this API allow you to create dynamic plugins (that will be transformed in Business Central screens) via the JS API.


```

$registerPlugin( {
  id: "my_angular_js",
  type: "angularjs",
  templateUrl: "angular.sample.html",
  title: function () {
    return "angular " + Math.floor(Math.random() * 10);
  },
  on_close: function () {
    alert("this is a pure JS alert!");
  }
});

```

The plugin references the **angular.sample.html** template:

```

<div ng-controller="TodoCtrl">
  <span>{{remaining()}} of {{todos.length}} remaining</span>
  [ <a href="" ng-click="archive()">archive</a> ]
  <ul class="unstyled">
    <li ng-repeat="todo in todos">
      <input type="checkbox" ng-model="todo.done">
      <span class="done-{{todo.done}}">{{todo.text}}</span>
    </li>
  </ul>
  <form ng-submit="addTodo()">
    <input type="text" ng-model="todoText" size="30"
placeholder="add new todo here">
    <input class="btn-primary" type="submit" value="add">
  </form>
  <form ng-submit="goto()">
    <input type="text" ng-model="placeText" size="30"
placeholder="place to go">
    <input class="btn-primary" type="submit" value="goTo">
  </form>
</div>

```

A plugin can be hooked to Business Central events via a series of JavaScript callbacks:

- on_concurrent_update;
- on_concurrent_delete;
- on_concurrent_rename;
- on_concurrent_copy;
- on_rename;
- on_delete;
- on_copy;
- on_update;
- on_open;

- on_close;
 - on_focus;
 - on_lost_focus;
 - on_may_close;
 - on_startup;
 - on_shutdown;
- Register splash screens API: use the methods in this API to create splash screens.

```
$registerSplashScreen({
  id: "home.splash",
  templateUrl: "home.splash.html",
  body_height: 325,
  title: function () {
    return "Cool Home Splash " + Math.floor(Math.random() * 10);
  },
  display_next_time: true,
  interception_points: ["Home"]
});
```

- Virtual File System (VFS) API: with this API, you can read and write a file saved in the file system using an asynchronous call.

```
$vfs_readAllString(uri, function(a) {
  //callback logic
});


$vfs_write(uri,content, function(a) {
  //callback logic
})
```

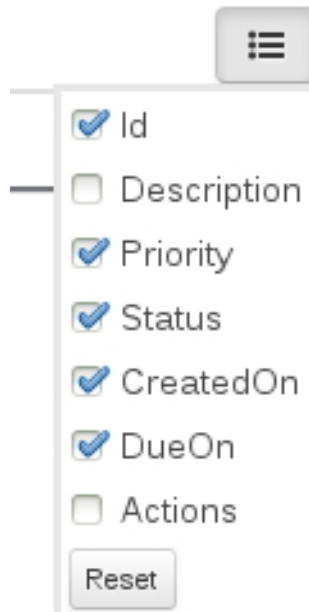
2.4. CONFIGURING TABLE COLUMNS

Business Central allows you to configure views that contain lists of items in the form of tables. You can resize columns, move columns, add or remove the default list of columns and sort the columns. This functionality is provided for all views that contain tables.

Once you make changes to the columns of a table view, these changes are persisted for the current logged in user.

Adding and Removing Columns

Tables that allow columns to be configured have  button in the top right corner. Clicking on this button opens up the list of columns that can be added or removed to the current table with a checkbox next



to each column:

Resizing Columns

To resize columns, place your cursor between the edges of the column header and move in the direction

Priority	Status
0	InProgress

that you want:

Moving Columns

To re-order and drag a column in a different position, hover your mouse over the rightmost area of the column header:

Filters: ▼ Active ▼ Personal ▼ Group ▼ All ▼ Ac				
Status				Due
InProgress				06/01

•

Drop it over the column header that you want to move it to.

To sort columns, click on the desired column's header. To reverse-sort, click on the header again.

To sort columns, click on the desired column's header. To reverse-sort, click on the header again.

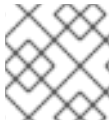
CHAPTER 3. LOGGING

3.1. LOGBACK FUNCTIONALITY

Red Hat JBoss BRMS provides **logback** functionality for logging configuration.

Accordingly, everything configured is logged to the *Simple Logging Facade for Java* [SLF4J](#), which delegates any log to Logback, Apache Commons Logging, Log4j or java.util.logging. Add a dependency to the logging adaptor for your logging framework of choice. If you're not using any logging framework yet, you can use Logback by adding this Maven dependency:

```
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.x</version>
</dependency>
```



NOTE

slf4j-nop and **slf4j-simple** are ideal for a light environment.

3.2. CONFIGURING LOGGING

To configure the logging level of the packages, create a **logback.xml** file in **business-central.war/WEB-INF/classes/logback.xml**. To set the logging level of the **org.drools** package to "debug" for verbose logging, you would need to add the following line to the file:

```
<configuration>

  <logger name="org.drools" level="debug"/>

  ...

</configuration>
```

Similarly, you can configure logging for packages such as the following:

- org.guvnor
- org.jbpm
- org.kie
- org.slf4j
- org.dashbuilder
- org.uberfire
- org.errai
- etc...

If configuring with **log4j**, the **log4j.xml** can be located at **business-central.war/WEB-INF/classes/log4j.xml** and can be configured in the following way:

```
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <category name="org.drools">
        <priority value="debug" />
    </category>

    ...

</log4j:configuration>
```



NOTE

Additional logging can be configured in the individual container. To configure logging for JBoss Enterprise Application Platform, please refer to the Red Hat JBoss Enterprise Application Platform Administration and Configuration Guide.

CHAPTER 4. REPOSITORY HOOKS

In Business Central, it is possible to trigger a chosen action every time a particular event happens. For this purpose, you can configure the repository to use scripts called hooks.

4.1. CONFIGURING GIT HOOKS

Business Central can automatically push changes to a remote repository using the Git hooks. Git hooks support has been introduced with the release of Red Hat JBoss BRMS 6.2.0.



NOTE

Please note that currently only the **post-commit** hook is supported. **Post-commit** hooks are triggered after finishing the entire commit process.

The following procedure shows how to configure the **post-commit** hook to automatically push your changes to the remote repository.

1. In Business Central, go to **Authoring** → **Administration**.
2. Below the main menu, click **Repositories** → **Clone repository**.
3. In the displayed **Clone repository** dialog box, fill in the repository information:
 - Repository Name
 - Organizational Unit
 - Git URL: for example **`https://github.com/USERNAME/REPOSITORY-NAME.git`**



IMPORTANT

It is important to use the HTTPS or Git protocol instead of a SCP-style SSH URL. Business Central does not support the basic SSH URL and fails with *Invalid URL format*.

Clone Repository ×

Repository Information * is required

* Repository Name

* Organizational Unit

* Git URL

git@github.com:user/ExampleRe

Invalid URL format

User Name

Password

Figure 4.1. An invalid SCP-style SSH URL.

- User Name: your Git user name
 - Password: your Git password
4. Go to the created repository:

```
~]$ cd $JBOSS_HOME/bin/.niogit/REPOSITORY-NAME.git
```

5. Change the remote URL:

```
~]$ git remote set-url origin git@github.com:USERNAME/REPOSITORY-NAME.git
```

Make sure that you can access the remote repository through command line using SSH. For example, the private SSH key for the repository should exist under the `~/.ssh/` directory.

6. Verify that the remote repository was successfully added:

```
~]$ git remote -v
```

The command should list the following:

```
origin  git@github.com:USERNAME/REPOSITORY-NAME.git (fetch)
origin  git@github.com:USERNAME/REPOSITORY-NAME.git (push)
```

7. Create a file named **post-commit** with the permissions set to **rw-r--r--** under **\$JBoss_HOME/bin/.nio-git/REPOSITORY-NAME.git/hooks** with the following content:

```
#!/bin/sh
git push origin master
```

8. Make sure that the configuration was successful by creating a new guided rule in Business Central: go to **Authoring** → **Project Authoring** and then **New Item** → **Guided Rule** below. Fill in the required information in the displayed **Create new Guided Rule** window and click **Ok**.

All of the changes should be pushed automatically.

For further information about remote Git repositories, refer to this article: [How to configure the BxMS 6 server to use a remote Git repository for storing assets?](#)

Furthermore, it is also possible to specify the system property **org.uberfire.nio.git.hooks**. Its value determines a directory with default hook files that will be copied to the newly created Git repositories. See the example of a **standalone.xml** file with this setting below:

```
<system-properties>
  <property name="org.uberfire.nio.git.hooks" value="/opt/jboss-as/git-
hooks">
  </property>
  ...
</system-properties>
```

CHAPTER 5. COMMAND LINE CONFIGURATION

The **kie-config-cli** tool is a command line configuration tool that provides capabilities to manage the system repository from the command line and can be used in an online or offline mode.

1. **Online mode** (default and recommended) - on startup, the tool connects to a Git repository using a Git server provided by **kie-wb**. All changes are made locally and published to upstream only after explicitly executing the `push-changes` command. Use the `exit` command to publish local changes. To discard local changes on exit, use the `discard` command.
2. **Offline mode** (a kind of installer style) - creates and manipulates the system repository directly on the server (there is no discard option).

The tool is available on the [Red Hat Customer Portal](#). To download the `kie-config-cli` tool, do the following:

1. Go to the [Red Hat Customer Portal](#) and log in.
2. Click **Downloads** → **Products Downloads**.
3. In the **Product Downloads** page that opens, click **Red Hat JBoss BRMS**.
4. From the **Version** drop-down menu, select 6.2.
5. In the displayed table, navigate to the **Supplementary Tools** row and then click **Download**.

Extract the zip package for supplementary tools you downloaded from the [Red Hat Customer Portal](#). It contains the directory **kie-config-cli-6.MINOR_VERSION-redhat-x-dist** with file **kie-config-cli.sh**.

5.1. STARTING THE KIE-CONFIG-CLI TOOL IN ONLINE MODE

1. To start the `kie-config-cli` tool in online mode, navigate to the **kie-config-cli-6.MINOR_VERSION-redhat-x-dist** directory where you installed the tool and then execute the following command.
2. In a Unix environment run:

```
./kie-config-cli.sh
```

In a Windows environment run:

```
./kie-config-cli.bat
```

By default, the tool starts in online mode and asks for user credentials and a Git URL to connect to (the default value is `git://localhost/system`). To connect to a remote server, replace the host and port with appropriate values. Example: `git://kie-wb-host:9148/system`

5.2. STARTING THE KIE-CONFIG-CLI TOOL IN OFFLINE MODE

To operate in offline mode, append the offline parameter to the command as below.

1. Navigate to the **kie-config-cli-6.MINOR_VERSION-redhat-x-dist** directory where you installed the tool.

2. In a Unix environment, run:

```
./kie-config-cli.sh offline
```

In a Windows environment, run:

```
./kie-config-cli.bat offline
```

Executing this command changes the tool's behaviour and displays a request to specify the folder where the system repository (**.niogit**) is located. If **.niogit** does not yet exist, the folder value can be left empty and a brand new setup is created.

5.3. COMMANDS AVAILABLE FOR THE KIE-CONFIG-CLI TOOL

The following commands are available for managing the GIT repository using the **kie-config-cli** tool:

- **add-deployment** - adds a new deployment unit
- **add-repo-org-unit** - adds a repository to the organizational unit
- **add-role-org-unit** - adds role(s) to an organizational unit
- **add-role-project** - adds role(s) to a project
- **add-role-repo** - adds role(s) to a repository
- **create-org-unit** - creates new organizational unit
- **create-repo** - creates a new git repository
- **discard** - does not publish local changes, cleans up temporary directories and closes the tool
- **exit** - publishes work, cleans up temporary directories and closes the tool
- **fetch-changes** - fetches changes from upstream repository
- **help** - prints available commands with descriptions
- **list-deployment** - lists available deployments
- **list-org-units** - lists available organizational units
- **list-repo** - lists available repositories
- **push-changes** - pushes changes to upstream repository (in online mode only)
- **remove-deployment** - removes existing deployment
- **remove-org-unit** - removes existing organizational unit
- **remove-repo** - removes an existing repository from config only
- **remove-repo-org-unit** - removes a repository from the organizational unit
- **remove-role-org-unit** - removes role(s) from an organizational unit

- **remove-role-project** - removes role(s) from a project
- **remove-role-repo** - removes role(s) from a repository

CHAPTER 6. MIGRATION

Migrating your projects from Red Hat JBoss BRMS 5 to Red Hat JBoss BRMS 6 requires careful planning and step by step evaluation of the various issues. You can plan for migration either manually, or by using automatic processes. Most real world migration will require a combination of these two processes.

Because JBoss BRMS 6 uses GIT for storing assets, artifacts and code repositories including processes and rules, you should start by creating an empty project in JBoss BRMS 6 as the basis for your migration with dummy files as placeholders for the various assets and artifacts. Running a GIT clone of this empty project into your favorite IDE will initiate the migration process.

Based on the placeholder files in your cloned project, you can start adding assets at the correct locations. The JBoss BRMS 6 system is smart enough to pick these changes and apply them correctly. Ensure that when you are importing old rule files that they are imported with the right package name structure.

Since Maven is used for building projects, the projects assets like the rules, processes and models are accessible as a simple jar file.

This section lists the generally accepted step by step ways to migrate your project. These are just guidelines though, and actual migration may vary a lot from this.

In general, you should...

1. Migrate the data first: These are your business assets.
2. Next, migrate your runtime processes.
3. Finally, convert old API calls to new ones one by one.

Let's look at these steps in more detail in the next few sections.

6.1. DATA MIGRATION

To migrate data from Red Hat JBoss BRMS 5, do the following:

1. Download the migration tool by logging in at the [Red Hat Customer Portal](#) and then navigating to Red Hat JBoss BRMS Software Downloads section. Click on **Red Hat JBoss BRMS Migration Tool** to download the zip archive.
2. Unzip the downloaded zip archive in a directory of your choice and navigate to this directory in a command prompt. This directory contains four folders:
 - **bin** - contains the launch scripts.
 - **jcr-exporter-libs** - contains the libs specific to the **export-from-JCR** part of the migration.
 - **vfs-importer-libs** - contains the libs specific to the **import-into-Git** part of the migration.
 - **conf** - contains global migration tool configuration.
3. For production databases, copy the JDBC driver for the database that is used by the JCR repository into the **jcr-exporter-libs** directory of the migration tool.

4. Execute the following command:

```
./bin/runMigration.sh -i <source-path> -o <destination-path> -r  
<repository-name>
```

Where:

- **<source-path>** is a path to a source JCR repository.
- **<destination-path>** is a path to a destination GIT VFS. This folder must not exist already.
- **<repository-name>** an arbitrary name for the new repository.

The repository is migrated at the specified destination.

Besides the **-i** command, you can also use **-h** to print out a help message and **-f** which forces an overwrite of the output directory, thus eliminating the need for manual deletion of this directory.

Importing the repository in Business Central

The repository can be imported in business central by cloning it. In the Administration perspective, click on the **Repositories** menu and then click on **Clone Repository** menu to start the process.



NOTE

Assets can also be migrated manually. After all, they are all just text files. The BPMN2 specification and the DRL syntax did not change between the different versions.

Importing the repository in JBDS

To import the repository in JBoss Developer Studio, do the following

1. Start JBoss Developer Studio.
2. Start the Red Hat JBoss BRMS server (if not already running) by selecting the server from the server tab and click the start icon.
3. Select **File** → **Import...** and navigate to the Git folder. Open the Git folder to select **Projects from Git** and click next.
4. Select the repository source as **Existing local repository** and click next.
5. Select the repository that is to be configured from the list of available repositories.
6. Import the project as a general project in the next window and click next. Name this project and click Finish.

6.2. API AND BACKWARDS COMPATIBILITY

Migrating to Version 6.1

In version 6.1, 5.x APIs are no longer officially supported.

Red Hat JBoss BRMS no longer provides backward compatibility with the rule, event, and process application programming interface (API) from JBoss BRMS 5. The content of the **knowledge-api JAR** file is no longer supported in version 6.1 and is replaced by APIs contained in the **kie-api JAR** file that were introduced in JBoss BRMS 6.0.

If you used the legacy 5.x API (located in **knowledge-api.jar**), please migrate (rewrite) the API calls to the new KIE API. Please be aware that several other APIs have changed between JBoss BRMS 5.x and JBoss BRMS 6.x, namely the task service API and the REST API.

Migrating to Version 6.0

The JBoss BRMS 6 system provides backward compatibility with the rule, event and process interactions from JBoss BRMS 5. You should eventually migrate (rewrite) these interactions to the all new revamped core API because this backward compatibility is likely to be deprecated.

If you cannot migrate your code to use the new API, then you can use the API provided by the purpose built **knowledge-api** jar for backwards compatible code. This API is the public interface for working with JBoss BPM Suite and JBoss BRMS and is backwards compatible.

If you are instead using the REST API in JBoss BRMS 5, note that this has changed as well and there is no mechanism in it for backwards compatibility.

PART I. INTEGRATION

CHAPTER 7. INTEGRATING RED HAT JBOSS BRMS WITH RED HAT JBOSS FUSE

Red Hat JBoss Fuse integration allows users of JBoss Fuse to complement their integration solution with additional features provided by JBoss BPM Suite and JBoss BRMS.

Red Hat JBoss BPMS integration is provided by two **features.xml** files:

- **drools-karaf-features-<version>-features.xml**

This file provides core JBoss BPM Suite and JBoss BRMS features, which defines the OSGi features that can be deployed into JBoss Fuse. This file is a part of the JBoss BPM Suite and JBoss BRMS product. OSGi users can install features from this file in order to install JBoss BRMS engine or JBoss BPM Suite engine into Fuse and use it in their applications.

- **karaf-features-<version>-features.xml**

This file provides additional features used for integrating JBoss BPM Suite and JBoss BRMS with Camel, primarily in Fuse. This file is part of the Integration Pack and it defines OSGi features that enable integration with Camel and SwitchYard. In addition to the karaf-features.xml, the Integration Pack also contains a features.xml file for quickstarts.

7.1. CORE JBOSS BPM SUITE AND JBOSS BRMS FEATURES

Core JBoss BPM Suite and JBoss BRMS features are provided by the **drools-karaf-features-<version>-features.xml** file present in your product Maven repository or the **jboss-brms-bpmsuite<version>-redhat<version>fuse-features.zip** file. It provides the following features:

- drools-common
- drools-module
- drools-templates
- drools-decisiontable
- drools-jpa
- kie
- kie-ci
- kie-spring
- kie-aries-blueprint
- jbpm-commons
- jbpm-human-task
- jbpm
- droolsjbpm-hibernate
- h2

The following table provides example of use cases for some of the features listed above.

Table 7.1. Features and Use Case Examples

Feature	Use Case
drools-module	Use the JBoss BRMS engine for rules evaluation, without requiring persistence, processes, or decision tables.
drools-jpa	Use the JBoss BRMS engine for rules evaluation with persistence and transactions, but without requiring processes or decision tables. The drools-jpa feature already includes drools-module , however you may also need to install the droolsjbpm-hibernate feature, or ensure there is a compatible hibernate bundle installed.
drools-decisiontable	Use the JBoss BRMS engine with decision tables.
jbpm	Use the JBoss BPM Suite (or JBoss BRMS engine with processes). The jbpm feature already includes drools-module and drools-jpa . You may also need to install the droolsjbpm-hibernate feature, or ensure that there is a compatible hibernate bundle installed.
jbpm and jbpm-human-task	Use the JBoss BPM Suite (or JBoss BRMS engine with processes) with Human Task.
Core engine jars and kie-ci .	Use JBoss BRMS or JBoss BPM Suite with KieScanner (KIE-CI) to download kJARs from a Maven repository.
kie-spring	Use KIE-Spring integration.
kie-spring and kie-aries-blueprint .	Use KIE-Aries-Blueprint integration.

7.2. ADDITIONAL FEATURES FOR SWITCHYARD AND CAMEL INTEGRATION

The following additional features for integration with SwitchYard and Camel on JBoss Fuse are provided by the integration pack:

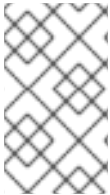
- **fuse-bxms-switchyard-common-knowledge**
- **fuse-bxms-switchyard-rules**
- **fuse-bxms-switchyard-bpm**
- **kie-camel**

- `jbpm-workitems-camel`

The integration pack features are defined in the **karaf-features-<version>-features.xml** file. This file (and supporting repositories) is located in <http://repository.jboss.org/nexus/content/repositories/public>, which is already configured for use on JBoss Fuse 6.2 out of the box in **installDir/etc/org.ops4j.pax.url.mvn.cfg**.

The file can also be downloaded from either the JBoss Fuse 6.2 or JBoss BRMS product page in the Red Hat Customer Portal.

7.3. INSTALL/UPDATE CORE INTEGRATION FEATURES



NOTE

This section refers to features in the **drools-karaf-features-<version>-features.xml** file. For additional integration features, refer to [Section 7.4, “Install Additional Integration Features”](#).

If you have already installed an older version of the core JBoss BPM Suite and JBoss BRMS features (for example, **drools-karaf-features-6.2.0.Final-redhat-6-features.xml**), you need to remove them and all associated files before installing the most recent **features.xml** file.

Procedure 7.1. Removing an Existing drools-karaf-features Installation

1. Start the Fuse console using:

```
$ ./installDir/bin/fuse
```

2. Uninstall old features/apps that used the previous **features.xml** file. For example:

```
JBossFuse:karaf@root> features:uninstall drools-module
JBossFuse:karaf@root> features:uninstall jbpm
JBossFuse:karaf@root> features:uninstall kie-ci
```

3. Search for references of bundles using drools/kie/jbpm and remove them:

```
JBossFuse:karaf@root> list -t 0 -s | grep drools
JBossFuse:karaf@root> list -t 0 -s | grep kie
JBossFuse:karaf@root> list -t 0 -s | grep jbpm
```

To remove the bundles:

```
karaf@root> osgi:uninstall <BUNDLE_ID>
```

4. Remove the old drools-karaf-features url:

```
karaf@root> features:removeurl mvn:org.drools/drools-karaf-features/6.2.0.Final-redhat-<version>/xml/features
```

5. Restart Fuse.

To install the **drools-karaf-features**:

Procedure 7.2. Install core JBoss BPM Suite and JBoss BRMS features

1. Configure required repositories
 - a. Edit the ***installDir/etc/org.ops4j.pax.url.mvn.cfg*** file in your JBoss Fuse installation and add the following entry to the ***org.ops4j.pax.url.mvn.repositories*** variable, noting that entries are separated by ***' , \'***:
 - ***http://maven.repository.redhat.com/product-ga/@id=bxms-product-repo***

2. Start JBoss Fuse:

```
$ ./installDir/bin/fuse
```

3. Add a reference to the core features file by running the following console command:

```
JBossFuse:karaf@root> features:addurl mvn:org.drools/drools-karaf-features/<version>/xml/features
```

For example:

```
features:addurl mvn:org.drools/drools-karaf-features/6.3.0.Final-redhat-7/xml/features
```

4. You can now install the features provided by this file by running, for example, the following console command:

```
JBossFuse:karaf@root> features:install drools-module
```

7.4. INSTALL ADDITIONAL INTEGRATION FEATURES

Use the following procedure for additional integration with SwitchYard and Camel.

Procedure 7.3. SwitchYard and Camel Integration

1. Download the ***fuse-integration*** package that is aligned with your version of JBoss Fuse.



NOTE

For instance, if you want to use the 6.2.0.redhat-117 version of JBoss Fuse, you need to install the ***fuse-6.2.0.redhat-117*** JBoss Fuse integration features.

2. Add the Remote Maven Repository that contains the fuse dependencies to your ***karaf*** instance:
 - Edit the ***Fuse_home/etc/org.ops4j.pax.url.mvn.cfg***
3. Update the Drools features URL:

```
JBossFuse:karaf@root> features:addurl mvn:org.switchyard.karaf/mvn:org.switchyard.karaf/switchyard/<SWITCH
```

```
YARD_VERSION>/xml/core-features
JBossFuse:karaf@root> features:addurl mvn:org.drools/drools-karaf-
features/<DROOLS_VERSION>/xml/features
JBossFuse:karaf@root> features:addurl
mvn:org.jboss.integration.fuse/karaf-features/1.0.0.redhat-
<version>/xml/features
```

4. You can now install the features provided for SwitchYard and Camel integration by running, for example, the following console command:

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-rules
JBossFuse:karaf@root> features:install kie-camel
JBossFuse:karaf@root> features:install jbpm-workitems-camel
```

7.5. CONFIGURING DEPENDENCIES

When you configure KIE, JBoss BRMS, or Jboss BPM Suite in your application, you can follow one of the following approaches to build your OSGi application bundles:

- Bundle required dependencies into your application bundle. In this approach, you declare all required artifacts as runtime dependencies in your **pom.xml**. Hence, you need not import the packages that provide these artifacts that you have already added as dependencies.
- Import the required dependencies into the application bundle. This is a preferred approach for building OSGi bundles as it adheres to the principles of OSGi framework. In this approach, you declare only the API jars (such as `org.kie:kie-api`) as dependencies in your application bundle. You will need to install the required BRMS and BPM Suite bundles and then import them in your application.

7.6. INSTALL JBOSS FUSE INTEGRATION QUICKSTART APPLICATIONS

The following features for JBoss Fuse integration quickstart applications are provided by **org/jboss/integration/fuse/quickstarts/karaf-features/1.0.0.redhat-<version>/karaf-features-1.0.0.redhat-<version>-features.xml**:

- `fuse-bxms-switchyard-quickstart-bpm-service`
- `fuse-bxms-switchyard-quickstart-rules-camel-cbr`
- `fuse-bxms-switchyard-quickstart-rules-interview`
- `fuse-bxms-switchyard-quickstart-rules-interview-container`
- `fuse-bxms-switchyard-quickstart-rules-interview-dtable`
- `fuse-bxms-switchyard-demo-library`
- `fuse-bxms-switchyard-demo-helpdesk`
- `fuse-bxms-camel-blueprint-drools-decision-table`
- `fuse-bxms-camel-spring-drools-decision-table`

- fuse-bxms-jbpm-workitems-camel-quickstart
- fuse-bxms-spring-jbpm-osgi-example

This file (and supporting repositories) is located in <http://repository.jboss.org/nexus/content/repositories/public>, which is already configured for use on JBoss Fuse 6.2 out of the box in `installDir/etc/org.ops4j.pax.url.mvn.cfg`.

Procedure 7.4. Installing the Quickstart Application

1. Add a reference to the features file by running the following console command:

```
JBossFuse:karaf@root> features:addurl  
mvn:org.jboss.integration.fuse.quickstarts/karaf-  
features/1.0.0.redhat-<version>/xml/features
```

2. You can now install the quickstart applications provided by this features file by running, for example, the following console command:

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-  
quickstart-bpm-service
```

Procedure 7.5. Downloading and Installing the Quickstart ZIP Files

1. Download the quickstart application ZIP file.
2. Unpack the contents of the quickstarts directory into your existing `installDir/quickstarts` directory.
3. Unpack the contents of the system directory into your existing `installDir/system` directory.

7.6.1. Testing Your First Quickstart Application

Procedure 7.6. Testing the Quickstart Application

1. Start JBoss Fuse:

```
$ ./installDir/bin/fuse
```

2. Install and start the **switchyard-bpm-service** by running the following console command:

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-  
quickstart-bpm-service
```



NOTE

Any dependent features specified by the application's features file will be installed automatically.

3. Submit a webservice request to invoke the SOAP gateway.

- a. Open a terminal window and navigate to the associated quickstart directory that was unpacked from the quickstart application ZIP file (in this case, **switchyard-bpm-service**).
- b. Run the following command:

```
$ mvn clean install
```



NOTE

You will need the following repositories configured in your **settings.xml** file:

- <http://maven.repository.redhat.com/ga/>
- <http://repository.jboss.org/nexus/content/repositories/public/>

- c. Run the following command:

```
$ mvn exec:java -Pkaraf
```

4. You will receive the following response:

```
SOAP Reply:
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-
  ENV:Header xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/" /><soap:Body><ns2:sub
  mitOrderResponse xmlns:ns2="urn:switchyard-quickstart:bpm-
  service:1.0">
    <orderId>test1</orderId>
    <accepted>true</accepted>
    <status>Thanks for your order, it has been shipped!</status>
  </ns2:submitOrderResponse></soap:Body></soap:Envelope>
```

CHAPTER 8. INTEGRATION WITH SPRING

8.1. CONFIGURING RED HAT JBOSS BRMS WITH SPRING

Refer to the Red Hat JBoss BRMS Installation Guide to download the Spring module. You will need to download the generic deployable version of JBoss BRMS.

The Spring module is present in the `jboss-brms-engine.zip` file and is called `kie-spring-VERSION-redhat-MINORVERSION.jar`.

How you intend to use the Spring modules in your application affects how you configure them.

As a Self Managed Process Engine

This is the standard way to start using JBoss BRMS in your Spring application. You only configure it once and run as part of the application. Using **RuntimeManager** API, perfect synchronization between process engine and task service is managed internally and the end user does not have to deal with the internal code to make these two work together.

As a Shared Task Service

When you use a single instance of a **TaskService**, you have more flexibility in configuring the task service instance as it is independent of the **RuntimeManager**. Once configured it is then used by the **RuntimeManager** when requested.

To create a **RuntimeEnvironment** from your Spring application, you can use the **org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean** class. This factory class is responsible for producing instances of **RuntimeEnvironment** that are consumed by **RuntimeManager** upon creation. Illustrated below is a configured **RuntimeEnvironment** with the entity manager, transaction manager, and resources for the class

org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean:

```
<bean id="runtimeEnvironment"
class="org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean">
  <property name="type" value="DEFAULT"/>
  <property name="entityManagerFactory" ref="jbpmEMF"/>
  <property name="transactionManager" ref="jbpmTxManager"/>
  <property name="assets">
    <map>
      <entry key-ref="process"><util:constant static-
field="org.kie.api.io.ResourceType.BPMN2"/></entry>
    </map>
  </property>
</bean>
```

The following **RuntimeEnvironment** can be created or configured:

- **DEFAULT** - default (most common) configuration for **RuntimeManager**
- **EMPTY** - completely empty environment to be manually populated
- **DEFAULT_IN_MEMORY** - same as **DEFAULT** but without persistence of the runtime engine
- **DEFAULT_KJAR** - same as **DEFAULT** but knowledge asset are taken from KJAR identified by releaseid or GAV

- `DEFAULT_KJAR_CL` - build directly from classpath that consists `kmodule.xml` descriptor

Depending upon the selected type, there are different mandatory properties that are required. However, at least one of the following knowledge properties must be provided:

- `knowledgeBase`
- `assets`
- `releaseId`
- `groupId`, `artifactId`, `version`

Finally, for `DEFAULT`, `DEFAULT_KJAR`, `DEFAULT_KJAR_CL` types, persistence needs to be configured in the form of values for **entity manager factory** and **transaction manager**. Illustrated below is an example `RuntimeManager` for

`org.kie.spring.factorybeans.RuntimeManagerFactoryBean`:

```
<bean id="runtimeManager"
class="org.kie.spring.factorybeans.RuntimeManagerFactoryBean" destroy-
method="close">
  <property name="identifier" value="spring-rm"/>
  <property name="runtimeEnvironment" ref="runtimeEnvironment"/>
</bean>
```

CHAPTER 9. LOCALIZATION AND CUSTOMIZATION

9.1. AVAILABLE LANGUAGES

The Red Hat JBoss BRMS web user interface can be viewed in multiple languages:

- United States English (**en_US**)
- Spanish (**es_ES**)
- Japanese (**ja_JP**)
- Chinese (**zh_CN**)
- Portuguese (**pt_BR**)
- French (**fr_CA**)
- German (**de_DE**)



NOTE

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

9.2. CHANGING LANGUAGE SETTINGS

Changing the User Interface Language in Business Central

By default, Business Central uses the system locale. If you need to change it, then append the required locale code at the end of the Business Central URL. For example, the following URL will set the locale to Portuguese (pt_BR).

`http://localhost:8080/business-central/?locale=pt_BR`

9.3. RUNNING THE JVM WITH UTF-8 ENCODING

Red Hat JBoss 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".

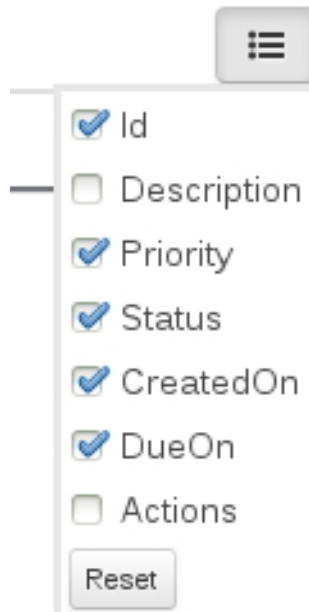
9.4. CONFIGURING TABLE COLUMNS

Business Central allows you to configure views that contain lists of items in the form of tables. You can resize columns, move columns, add or remove the default list of columns and sort the columns. This functionality is provided for all views that contain tables.

Once you make changes to the columns of a table view, these changes are persisted for the current logged in user.

Adding and Removing Columns

Tables that allow columns to be configured have  button in the top right corner. Clicking on this button opens up the list of columns that can be added or removed to the current table with a checkbox next



to each column:

Resizing Columns

To resize columns, place your cursor between the edges of the column header and move in the direction

	Priority	Status
	0	InProgress

that you want:

Moving Columns

To re-order and drag a column in a different position, hover your mouse over the rightmost area of the column header:

Filters: ▼ Active ▼ Personal ▼ Group ▼ All ▼ Ac				
Status				Due
InProgress				06/01

Filters: ▼ Active ▼ Personal ▼ Group ▼ All ▼ Admin			
Status		DueOn	
InProgress		06/01/2015 12:00	

Sorting Columns

To sort columns, click on the desired column's header. To reverse-sort, click on the header again.

CHAPTER 10. PROCESS EXECUTION SERVER CONFIGURATION

10.1. ASSIGNMENT RULES

Assignment rules are rules executed automatically when a Human Task is created or completed. This mechanism can be used, for example, to assign a Human Task automatically to a particular user of a group or prevent a user from completing a Task if data is missing.

10.1.1. Defining assignment rules

To define assignment rules, do the following:

1. Create a file that will contain the rule definition on the Business Central classpath (the recommended location is `$DEPLOY_DIR/standalone/deployments/business-central.war/WEB-INF/classes/`):
 - `default-add-task.dr1` with the rules to be checked when the Human Task is created
 - `default-complete-task.dr1` with the rules to be checked when the Human Task is completed
2. Define the rules in the file.

Example 10.1. The default-add-task.dr1 content

```
package defaultPackage

import org.kie.api.task.model.Task;
import org.kie.api.task.model.User;
import org.kie.api.task.model.Status;
import org.kie.api.task.model.PeopleAssignments;
import org.jbpm.services.task.rule.TaskServiceRequest;
import org.jbpm.services.task.exception.PermissionDeniedException;
import org.jbpm.services.task.impl.model.*;
import java.util.HashMap;
import java.util.List;

global TaskServiceRequest request;

rule "Don't allow Mary to complete task when rejected"
  when
    $task : Task()
    $actualOwner : User( id == 'mary') from
    $task.getTaskData().getActualOwner()
    $params : HashMap(this["approved"] == false)
  then
    request.setAllowed(false);
    request.setExceptionClass(PermissionDeniedException.class);
    request.addReason("Mary is not allowed to complete task with
    approved false");
  end
```

If the potential owners of a Human Task will contain the user **Mary**, the task will be automatically assigned to the user **mary**.

Example 10.2. The default -complete-task.drl content

```
package defaultPackage

import org.kie.api.task.model.Task;
import org.kie.api.task.model.User;
import org.kie.api.task.model.Status;
import org.kie.api.task.model.PeopleAssignments;
import org.jbpm.services.task.rule.TaskServiceRequest;
import org.jbpm.services.task.exception.PermissionDeniedException;
import org.jbpm.services.task.impl.model.*;
import java.util.HashMap;
import java.util.List;

global TaskServiceRequest request;

rule "Don't allow Mary to complete task when rejected"
when
    $task : Task()
    $actualOwner : User( id == 'mary') from
    $task.getTaskData().getActualOwner()
    $params : HashMap(this["approved"] == false)
then
    request.setAllowed(false);
    request.setExceptionClass(PermissionDeniedException.class);
    request.addReason("Mary is not allowed to complete task without
approval.");
end
```

If the potential owners of a Human Task will contain the user **Mary**, the task will be automatically assigned to the user **mary**.

10.2. MAIL SESSION

Mail session defines the mail server properties that are used for sending emails if required by the application, such as, escalation or notification mechanisms (refer to the *Red Hat JBoss BRMS User Guide*).

10.2.1. Setting up mail session

To set up the mail session for your execution engine, do the following:

1. Open the respective profile configuration file (**standalone.xml** or **host.xml**) for editing.
2. Add the mail session to the **urn:jboss:domain:mail:1.1** subsystem.

Example 10.3. New mail session on localhost

.

```
<subsystem xmlns="urn:jboss:domain:mail:1.1">
  <!-- omitted code -->

  <mail-session jndi-name="java:/mail/brmsMailSession"
debug="true" from="brms@company.com">
    <smtp-server outbound-socket-binding-ref="brmsMail"/>
  </mail-session>
</subsystem>
```

3. Define the session outbound socket in the profile configuration file.

Example 10.4. Outbound socket definition

```
<outbound-socket-binding name="brmsMail">
  <remote-destination host="localhost" port="12345"/>
</outbound-socket-binding>
```

CHAPTER 11. MONITORING

11.1. JBOSS OPERATIONS NETWORK

A JBoss Operations Network plug-in can be used to monitor rules sessions for Red Hat JBoss . The plug-in uses Java Management Extensions (JMX) to monitor rules sessions.

Due to a limitation of passing the JVM monitoring arguments via the Maven command line, all `com.sun.management.jmxremote.*` parameters must be passed to the JBoss application via the `pom.xml` configuration file.

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

11.2. DOWNLOADING RED HAT JBOSS

1. Go to the [Red Hat Customer Portal](#) and log in.
2. Click **Downloads** → **Products Downloads**.
3. In the **Product Downloads** page that opens, click **Red Hat JBoss BRMS**.
4. From the **Version** drop-down menu, select version .
5. Select **Red Hat JBoss BRMS 6.2 Deployable for EAP 6.4** and then click **Download**.

11.3. INSTALLING THE JBOSS BRMS PLUG-IN INTO JBOSS ON

Red Hat JBoss BRMS plug-in for JBoss Operations Network can be installed by either copying the plug-in JAR files to the JBoss Operations Network plug-in directory or through the JBoss Operations Network GUI.

The following procedure guides a user to copy the plug-in JAR files to the JBoss Operations Network plug-in directory

Procedure 11.1. Copying the JBoss BRMS plug-in JAR files

1. Extract the JBoss BRMS plug-in pack archive to a temporary location. This creates a subdirectory with the name `jon-plugin-pack-brms-bpms-3.3.0.GA`. For example:

```
[root@server rhq-agent]# unzip jon-plugin-pack-brms-bpms-3.3.0.GA.zip -d /tmp
```

2. Copy the extracted JBoss BRMS plug-in JAR files from the `jon-plugin-pack-brms-bpms-3.2.0.GA/` directory to the JBoss ON server plug-in directory. For example:

```
[root@server rhq-agent]# cp /tmp/jon-plugin-pack-brms-bpms-3.3.0.GA/*.jar /opt/jon/jon-server-3.3.0.GA1/plugins
```

3. Start the JBoss Operations Network server to update the JBoss BRMS plug-in.

To upload the JBoss BRMS plug-in through the JBoss Operations Network GUI, following is the procedure

Procedure 11.2. Uploading the JBoss BRMS plug-in through GUI

1. Start the JBoss Operations Network Server and Log in to access the GUI.
2. In the top navigation of the GUI, open the **Administration** menu.
3. In the **Configuration** area on the left, select the **Server Plugins** link.
4. At the bottom of the list of loaded server plug-ins, click the **Upload a plugin** button and choose the BRMS plugin.
5. The JBoss BRMS plug-in for JBoss Operations Network is now uploaded.

11.4. MONITORING KIE BASES AND KIE SESSIONS

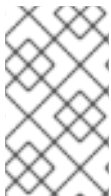
In order for JBoss Operations Network to monitor KieBases and KieSessions, MBeans must be enabled.

MBeans can be enabled either by passing the parameter:

```
-kie.mbeans = enabled
```

Or via the API:

```
KieBaseConfiguration kbconf =
KieServices.Factory.get().newKieBaseConfiguration();
kbconf.setOption(MBeansOption.ENABLED);
```



NOTE

Kie Services have been implemented for JBoss BRMS 6; for JBoss BRMS 5, **Drools Services** was the naming convention used and it had different measurements on sessions. For example, **activation** → **match** renaming occurred in the updated version.

Please refer to the *JBoss Operations Network Resource Monitoring and Operations Reference* guide for information on importing Kie Sessions into the Inventory View for monitoring purposes.

11.5. THE JBOSS RULES KIE BASE MONITORING SERVICE

The JBoss Rules Kie Base Monitoring Service can configure the KieBase by providing the KieBase ID as a connection property.

The JBoss Rules knowledge base monitoring service provides the following operations:

- Start all internal MBeans - starts the internal MBeanServer that will register all the knowledge sessions created from the knowledge base. The resources can be knowledge bases and stateful knowledge sessions. This displays the knowledge base configuration and entry points related to that information.
- Stop all internal MBeans - stops previously started internal MBeans.

11.6. THE JBOSS RULES KIE SESSION MONITORING SERVICE

The JBoss Rules Kie Sessions Monitoring Service can configure the kie base and kie session by providing the kie base ID and kie session ID as connection properties.

The JBoss Rules Kie Session Monitoring Service provides the following operations:

- Reset all metrics counters - resets all the metric statistics with regard to rule/process/process instance specific information.
- Return statistics for a specific rules - provides a description of the total number of fired/created/canceled activations generated by a rule.
- Get statistics for a specific process - provides a description of the total number of fired/created/canceled activations generated by a process.
- Get statistics for a specific process instance - provides a description of the total number of fired/created/canceled activations generated by each process instance.

The JBoss Rules Kie Session Monitoring Service provides the following metrics:

- The total number of facts in working memory.
- The total number of matches created since the last reset.
- The total number of matches fired since the last reset.
- The total number of matches 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.

APPENDIX A. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat JBoss BRMS.

Revision 6.2.0-5 Resolved build failure.	Thu Apr 28 2016	Tomas Radej
Revision 6.2.0-4 Updated with latest fixes.	Thu Apr 28 2016	Tomas Radej
Revision 6.2.0-3 Build for release update 2 of JBoss BRMS.	Tue Mar 29 2016	Tomas Radej
Revision 6.2.0-2 Added note about versions in Revision History, fixed changelog dates.	Mon Nov 30 2015	Tomas Radej
Revision 6.2.0-1 Initial build for release 6.2.0 of JBoss BRMS.	Mon Nov 30 2015	Tomas Radej