Red Hat JBoss BPM Suite 6.0

Administration And Configuration Guide

The Administration and Configuration Guide for Red Hat JBoss BPM Suite

Last Updated: 2017-11-17

Red Hat JBoss BPM Suite 6.0 Administration And Configuration Guide

The Administration and Configuration Guide for Red Hat JBoss BPM Suite

Red Hat Content Services

Legal Notice

Abstract

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

Table of Contents

PART I. INTRODUCTION

CHAPTER 1. BUSINESS PROCESS MODEL AND NOTATION

Business Process Model and Notation (BPMN) is a standard notation for business process modeling. It aspires to link the gap between business analysts and programmers by providing a workflow language that can be clearly understood by both.

Report a bug

1.1. ABOUT RED HAT JBOSS BPM SUITE

Red Hat JBoss BPM Suite is an open source business process management suite that combines Business Process Management and Business Rules Management and enables business and IT users to create, manage, validate, and deploy Business Processes and Rules.

Red Hat JBoss BRMS and Red Hat JBoss BPM Suite use a centralized repository where all resources are stored. This ensures consistency, transparency, and the ability to audit across the business. Business users can modify business logic without requiring assistance from IT personnel.

To accommodate Business Rules component, Red Hat JBoss BPM Suite includes integrated Red Hat JBoss BRMS.

Business Resource Planner is included as a technical preview with this release.

Report a bug

1.2. COMPONENTS

Red Hat JBoss BPM Suite integrates multiple components to support business processes throughout their entire life cycle and to provide process management features and tools for business analysts, developers, and business users. The product can be deployed on various JEE-compliant servers; the recommended option is Red Hat JBoss Enterprise Application Platform 6.

Red Hat JBoss BPM Suite consists of the following main components:

- **`Execution Engine`** - provides the runtime environment for Processes and Business Rules. It encompasses a workflow library that can be embedded into a user web application. Runtime manager is the root object and contains the following components:
  - **`Runtime Engine`** - implements the core behavior of the computer language and it is provided by the runtime manager.
    - **`Process Engine`** - is the environment for business process model execution.
    - **`Task Service`** - handles human task lifecycles.
  - **`Rule Engine`** - can be used with the process engine or on its own.
    - **`Rules Evaluation`** - executes business rules on the provided set of facts.
    - **`Complex Event Processing`** - applies business rules on incoming stream of events.
- **`Business Central`** - a web-based application that accommodates tooling for asset creation, management, and monitoring by providing an integrated web environment.
  - **`Asset Repository`** - is the central sharing location (Knowledge Store) for business

assets, processes, rules, forms, etc. Users access this repository through the Project Explorer view of Business Central via **Authoring → Project Authoring**. By default, the product initializes a local GIT repository as its Asset Repository. However, other repositories may be added or removed as necessary.

- **`Artifact Repository`** - is a Maven based repository for storage of project jar artifacts.
- **`Execution Server`** - provides an execution environment for business process instances and tasks.
- **`Business Activity Monitor`** - provides customizable view on business performance.

NOTE

Red Hat JBoss BRMS comes with its own Business Central application that is a subset of the Business Central application in Red Hat JBoss BPM Suite.

Report a bug

1.3. USE CASE: PROCESS-BASED SOLUTIONS IN THE LOAN INDUSTRY

Red Hat JBoss BPM Suite (BPMS) can be deployed to automate business processes, such as automating the loan approval process at a retail bank. This is a typical 'Specific Process-Based' deployment that might be the first step in a wider adoption of BPM throughout an enterprise. It leverages both the BPM and business rules features of BPMS.

A retail bank offers several types of loan products each with varying terms and eligibility requirements. Customers requiring a loan must file a loan application with the bank, which then processes the application in several steps, verifying eligibility, determining terms, checking for fraudulent activity, and determining the most appropriate loan product. Once approved, the bank creates and funds a loan account for the applicant, who can then access funds. The bank must be sure to comply with all relevant banking regulations at each step of the process, and needs to manage its loan portfolio to maximize profitability. Policies are in place to aid in decision making at each step, and those policies are actively managed to optimize outcomes for the bank.

Business analysts at the bank model the loan application processes using the BPMN2 authoring tools (Process Designer) in BPM Suite:

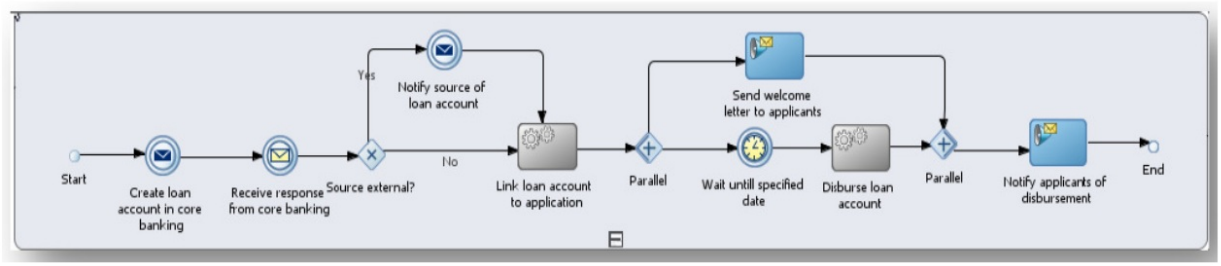


Figure 1.1. High-level loan application process flow

Business rules are developed with the rule authoring tools in BPM Suite to enforce policies and make decisions. Rules are linked with the process models to enforce the correct policies at each process step.

The bank's IT organization deploys the BPM Suite so that the entire loan application process can be automated.

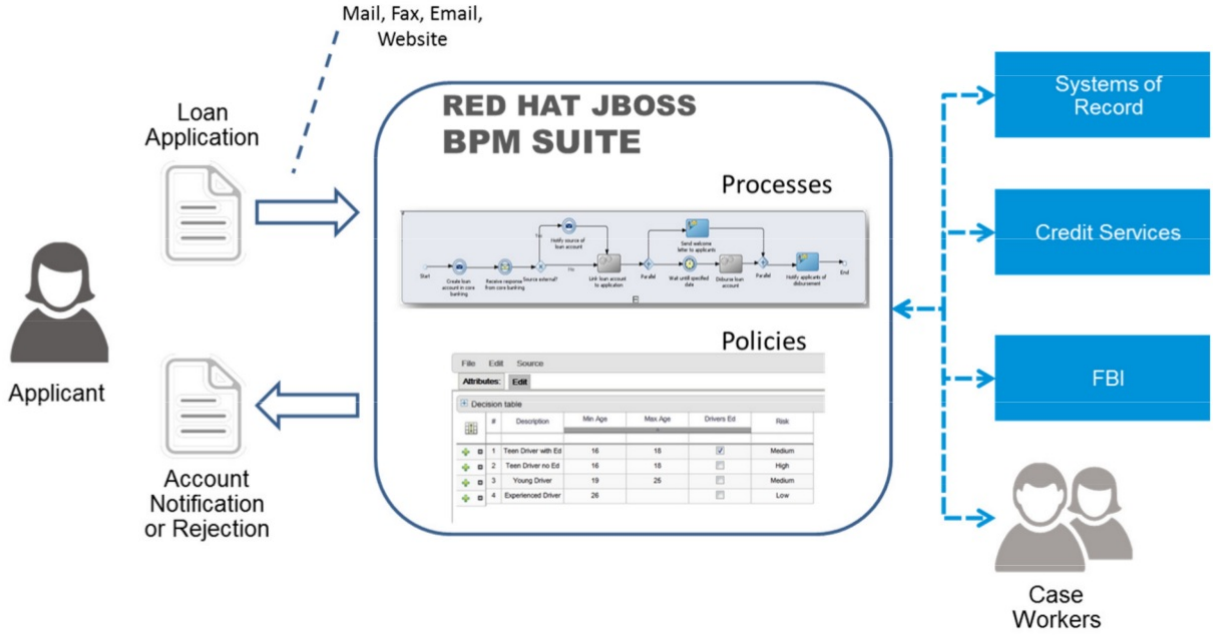


Figure 1.2. Loan Application Process Automation

The entire loan process and rules can be modified at any time by the bank's business analysts. The bank is able to maintain constant compliance with changing regulations, and is able to quickly introduce new loan products and improve loan policies in order to compete effectively and drive profitability.

Report a bug

PART II. CONFIGURATION

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 *Red Hat JBoss BPMS Installation Guide*).

Report a bug

2.1. ACCESS CONTROL

The access control mechanism includes authorization and authentication. In the unified environment of Red Hat JBoss BPM Suite, users are able to update the default user roles located within ***$JBOSS_HOME*/standalone/deployments/business-central.war/WEB-INF/classes/userinfo.properties**.

To grant a user access to BPMS, the user needs to have the respective role assigned:

- **admin**: administrates BPMS system and has full access rights to make any changes necessary including the ability to add and remove users from the system.
- **developer**: implements code required for processes to work and has access to everything except administration tasks.
- **analyst**: creates and designs processes and forms and instantiates the processes. This role is the similar to a developer, without access to asset repository and deployments.
- **user**: claims, performs, and invokes other actions (such as, escalation, rejection, etc.) on the assigned Tasks and has no access to authoring functions.
- **manager**: monitors the system and its statistics and only has access to the dashboard.
- **business user**: takes action on business tasks that are required for processes to continue forward. Works primarily with the task list.

If using Red Hat JBoss EAP, to create a user with particular roles, run the ***$JBOSS_HOME*/add-user.sh** script and create an Application User in the **ApplicationRealm** with the respectives roles.

Workbench Configuration

Within the Red Hat JBoss BPM Suite, 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* business-central.war/WEB-INF/classes/workbench-policy.propeties**

Authentication in Human Tasks

Every Task that needs to be executed is assigned to one or multiple roles or groups, so that any user with the given role or the given group assigned can claim the Task instance and execute it. Tasks can also be assigned to one or multiple users directly. Such roles are created by the application container.

WARNING

A group for a Human Task must not be named after an existing user of the system. Doing so causes intermittent issues.

Report a bug

CHAPTER 3. 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 BPM Suite**.

4. From the **Version** drop-down menu, select 6.0.3.

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**.

Report a bug

3.1. 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 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**

Report a bug

3.2. 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.

Report a bug

3.3. 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
- **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

Report a bug

CHAPTER 4. MIGRATION

Migrating your projects from BRMS 5 to BPMS 6 requires careful planning and step by step evaluation of the various issues.

Because BPMS 6 uses GIT for storing assets, artifacts and code repositories including processes and rules, you should start by creating an empty project in BPMS 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 BPMS 6 system is smart enough to pick these changes and apply them correctly. Make sure 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.

Let's get started with the simple data migration step, which migrates your BRMS 5 repository from the JCR Jackrabbit to a GIT backend in the new BPMS 6.

Report a bug

4.1. DATA MIGRATION

To migrate data from 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 BPM Suite Software Downloads section. Click on **`Red Hat JBoss BPM Suite Migration Tool`** to download the zip archive.

2. Unzip the downloaded zip archive in a directory of your choice, navigate to this directory in a command prompt and then run the following command:

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

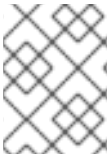
Where:

- **`<source-path>`** is a path to a source JCR repository.
- **`<desintation-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.

Importing the repository in Business Central

The repository can be imported in business central by cloning it. In the Adminstration 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 BPM Suite 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.

Report a bug

4.2. RUNTIME MIGRATION

To run BPMS 5 processes to BPMS 6, do the following:

1. Set the system property **`jbpm.v5.id.strategy`** to true in the BPMS **`standalone.xml`**:

```
<property name="jbpm.v5.id.strategy" value="true"/>
```

2. Load the KieSession like the following:

```
KieSession ksession =
JPAKnowledgeService.loadStatefulKnowledgeSession(sessionID, kbase,
sessionConf, env);
```

3. Continue the normal execution of the process using KieSession methods:

```
ksession.signalEvent("SomeEvent", null);
```

Report a bug

4.3. API AND BACKWARDS COMPATIBILITY

The BPMS 6 system provides backward compatibility with the rule, event and process interactions from 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 can't 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 BPMS and BRMS and is backwards compatible.

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

Report a bug

4.4. MIGRATING TASK SERVICE

BPMS 6 provides support for a locally running task server only. This means that you don't need to setup any messaging service in your project. This differs from BRMS 5 because it provided a task server that was bridged from the core engine by using, most commonly, the messaging system provided by HornetQ.

To help you bridge the gap until you can migrate this in your current architecture, there is a helper or utility method, **`LocalHTWorkItemHandler`**.

Since the TaskService API is part of the public API you will now need to refactor your imports because of package changes and refactor your methods due to API changes themselves.

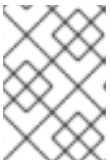
Report a bug

CHAPTER 5. DATA MANAGEMENT

5.1. DATA BACKUPS

When applying a backup mechanism to Red Hat JBoss BPMS make sure you back up the following resources:

- any customized deployment descriptors (such as, **web.xml**, **jboss-web.xml**, **jboss.xml**)
- any customized properties files



NOTE

Consider backing up the entire **business-central.war** and **dashbuilder.war** files.

Report a bug

5.2. SETUP INDEXES

Setup foreign key indexes

Some databases, for instance Oracle and Postgres, do not automatically create an index for each foreign key. This can result in deadlocks occurring. To avoid this situation it is necessary to create an index on all foreign keys, especially in the Oracle database.

Report a bug

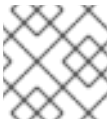
5.3. SETTING UP THE DATABASE

Dashbuilder application requires an existing database, previously created before running the application. To create a database you can use any database client tool and run the following commands:

Postgres

The following sql sentence is used to create a Postgres database:

```
CREATE DATABASE dashbuilder
  WITH ENCODING='UTF8'
         OWNER=dashbuilder
         CONNECTION LIMIT=-1
```



NOTE

The database encoding must be UTF8

DB2

DB2 database can be created using the following sql sentence:

```
CREATE DATABASE dashb PAGESIZE 16384
```

NOTE

The default pagesize for DB2 systems is 4k which is not enough for the dashbuilder table columns size. The pagesize should be forced to 16384 as shown in the above sentence.

Once the database is created, the application server datasource must be configured. You must edit the JBoss EAP configuration file and configure the datasource subsystem as the following examples:

```
<datasource jndi-name="java:jboss/datasources/jbpmDS" enabled="true" use-
java-context="true" pool-name="postgresqlDS">
    <connection-url>jdbc:postgresql://localhost/test</connection-url>
    <driver>postgresql</driver>
    <pool></pool>
    <security>
       <user-name>sa</user-name>
       <password>sa</password>
    </security>
</datasource>
<drivers>
    <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
    </driver>
</drivers>
```

Report a bug

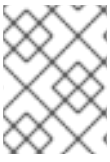
5.4. EDITING THE DATABASE

Dashbuilder requires the JBoss BPM Suite to have history log's database tables. It is mandatory to deploy the Human Task console (or a superset, i.e: kie-wb) first. Otherwise, the Dashboard will not be initialized correctly and it will not be possible to display its key performance indicators.

By default, the application is configured to use a datasource with the following JNDI name:

```
java:jboss/datasources/ExampleDS
```

This is specified in JBoss EAP's configuration file; for example, **`standalone.xml`**.



NOTE

This datasource is intended for development/demo purposes; it is present by default in any JBoss installation.

If you want to deploy on a database different from H2 like Oracle, MySQL, Postgres or MS SQL Server, please perform the following steps:

Procedure 5.1. Changing Database

1. Install the database driver on JBoss (refer to JBoss driver documentation).

2. Create an empty database and a JBoss data source which connects to the database driver.

3. Modify the file **dashbuilder.war/WEB-INF/jboss-web.xml**:

```
<jboss-web>
   <context-root>/dashbuilder</context-root>
   <resource-ref>
       <res-ref-name>jdbc/dashbuilder</res-ref-name>
       <res-type>javax.sql.DataSource</res-type>
       <jndi-name>java:jboss/datasources/myDataSource</jndi-name>
   </resource-ref>
   ...
```

4. Replace the jndi-name parameter value by the JNDI path of the JBoss data source you've just created.

5. Modify the file **dashbuilder.war/WEB-INF/jboss-deployment-structure.xml**

6. Add the following snippet of configuration inside the **deployment** tag, where **jdbcDriverModuleName** is the name of the JBoss JDBC driver module:

```
  <dependencies>
      <module name="jdbcDriverModuleName" />
  </dependencies>
```

Report a bug

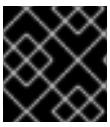
CHAPTER 6. ASSET REPOSITORY

Business Rules, Process definition files 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 BPM Suite.

Report a bug

6.1. CREATING A REPOSITORY



IMPORTANT

Note that only user with the **`ADMIN`** role can create a repository.

Procedure 6.1. Creating a New Repository

1. Open the **`Administration`** perspective: on the main menu, click **Authoring** → **Administration**.
2. On the perspective menu, click **Repositories** → **New Repository**.
3. The **`Create Repository`** pop-up window is displayed.

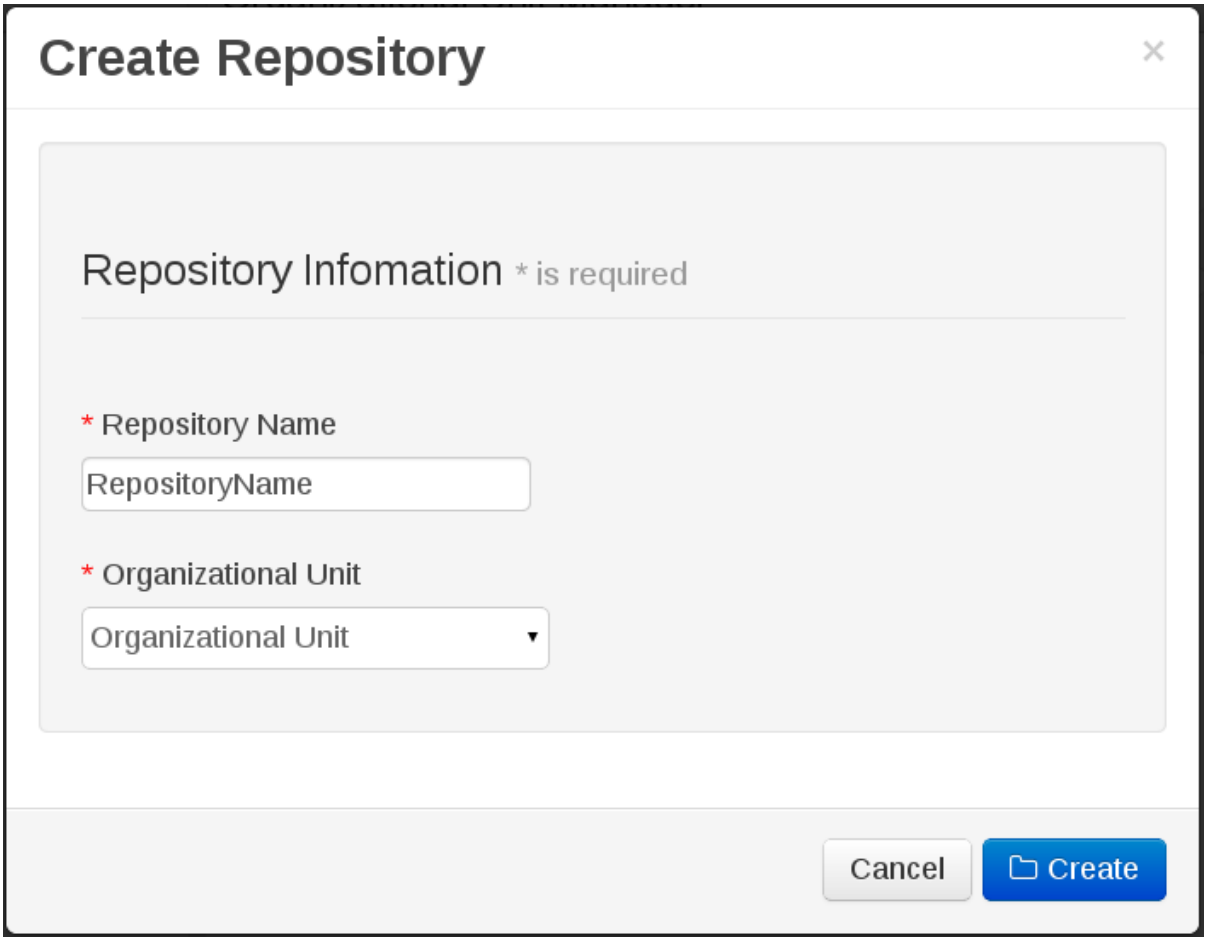


Figure 6.1. Create Repository Pop-up

4. Enter the mandatory details:
   - Repository name.

     NOTE

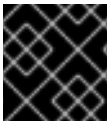
     Note that the repository name should be a valid filename. Avoid using a space or any special character that might lead to an invalid folder name.

   - Select an organizational unit in which the repository is to be created from the **`Organizational Unit`** drop-down option.
5. Click **`Create`**
6. A confirmation prompt with an **`OK`** button is displayed which notifies the user that the repository is created successfully. Click **`OK`**.

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

Report a bug

6.2. CLONING A REPOSITORY



IMPORTANT

Note that only user with the **ADMIN** role can clone a repository.

Procedure 6.2. Cloning a repository

1. Open the **Administration** perspective.
2. On the **Repositories** menu, select **Clone repository**.
3. The **Clone Repository** pop-up window is displayed.

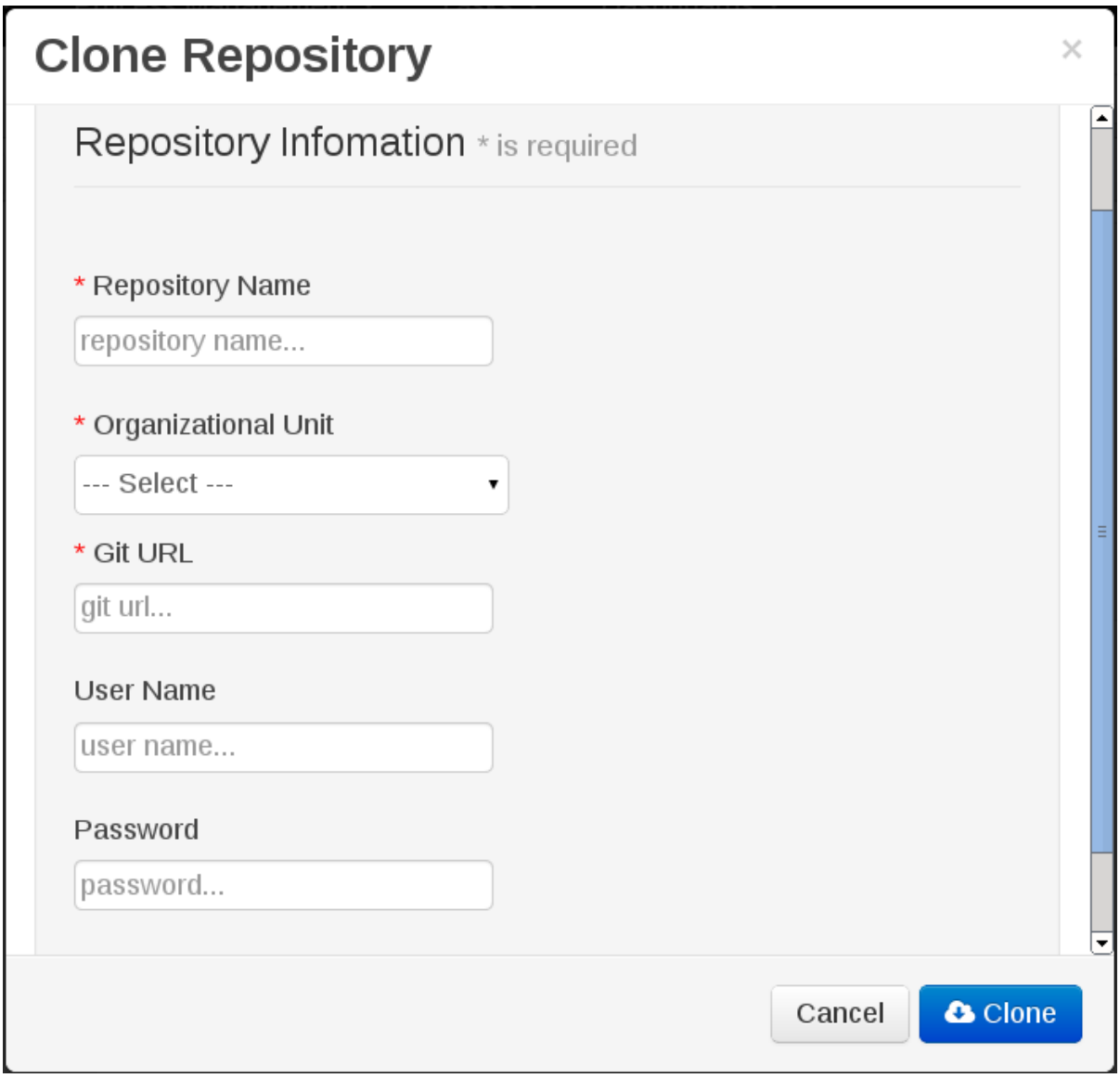


Figure 6.2. Clone Repository Pop-up

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: **file:///*path-to-repository/reponame***

- For a Remote or preexisting Repository: **`git://hostname/reponame`**

NOTE

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

c. If applicable, enter the **`User Name`** and **`Password`** to be used for authentication when cloning the repository.

5. Click **`Clone`**.

6. A confirmation prompt with an **`OK`** button is displayed which notifies the user that the repository is created successfully. Click **`OK`**.

The cloned repository can be checked either in the **`File Explorer`** or **`Project Explorer`** views.

Report a bug

6.3. REMOVING A REPOSITORY

To remove a repository form the Knowledge Store, issue the **`DELETE`** REST API call. This call relies on the user having created an authenticated HTTP session before issuing this command.

Example 6.1. Removing a repository using curl

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

Report a bug

6.4. SYSTEM CONFIGURATION

To change system properties, such as SSH configuration, do the following in WildFly of JBoss EAP cluster:

Procedure 6.3. 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 a system property, for example:

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

Listed below are a list of all 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 local host identifier. Default: localhost
- **`org.uberfire.nio.git.daemon.port`**: If git daemon enabled, uses this property as 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 enabled, uses this property as local host identifier. Default: localhost
- **`org.uberfire.nio.git.ssh.port`**: If ssh daemon enabled, uses this property as port number. Default: 8001
- **`org.uberfire.nio.git.ssh.cert.dir`**: Location of the directory .security where local certtificates will be stored. Default: working directory
- **`org.uberfire.metadata.index.dir`**: Place where Lucene .index folder will be stored. 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 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: PBEWithMD5AndDES
- **`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. Demo repositories can be obtained from the kie-wb-6.0.1.Final-example-repositories.zip artifact. This System Property takes precedence over org.kie.demo and org.kie.example. 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

Report a bug

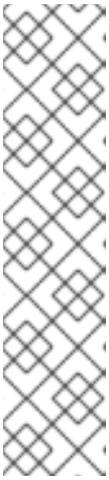
CHAPTER 7. PROCESS EXPORT AND IMPORT

7.1. CREATING A PROCESS DEFINITION

Make sure you have logged in to JBoss BPMS or you are in JBoss Developer Studio with the repository connected.

To create a Process, do the following:

1. Open the Project Authoring perspective (**Authoring** → **Project Authoring**).
2. In **`Project Explorer`** (**Project Authoring** → **Project Explorer**), navigate to the project where you want to create the Process definition (in the **`Project`** view, select the respective repository and project in the drop-down lists; in the **`Repository`** view, navigate to ***`REPOSITORY/PROJECT`*`/src/main/resources/`** directory).



NOTE

It is recommended to create your resources, including your Process definitions, in a package of a Project to allow importing of resources and their referencing. To create a package, do the following:

- In the **`Repository`** view of the Project Explorer, navigate to the ***`REPOSITORY/PROJECT`*`/src/main/resources/`** directory.
- Go to **New Item** → **Package**.
- In the **`New resource`** dialog, define the package name and check the location of the package in the repository.

3. From the perspective menu, go to **New Item** → **Business Process**.
4. In the **`New Processes`** dialog box, enter the Process name and click **`OK`**. Wait until the Process Editor with the Process diagram appears.

Report a bug

7.2. IMPORTING A PROCESS DEFINITION

To import an existing BPMN2 or JSON definition, do the following:

1. In the **`Project Explorer`**, select a Project and the respective package to which you want to import the Process definition.
2. Create a new Business Process to work in by going to **New Item** → **Business Process**.
3. In the Process Designer toolbar, click the **`Import`**  icon in the editor toolbar and pick the format of the imported process definition. Note that you have to choose to overwrite the existing process definition in order to import.
4. From the **`Import`** window, locate the Process file and click **`Import`**.

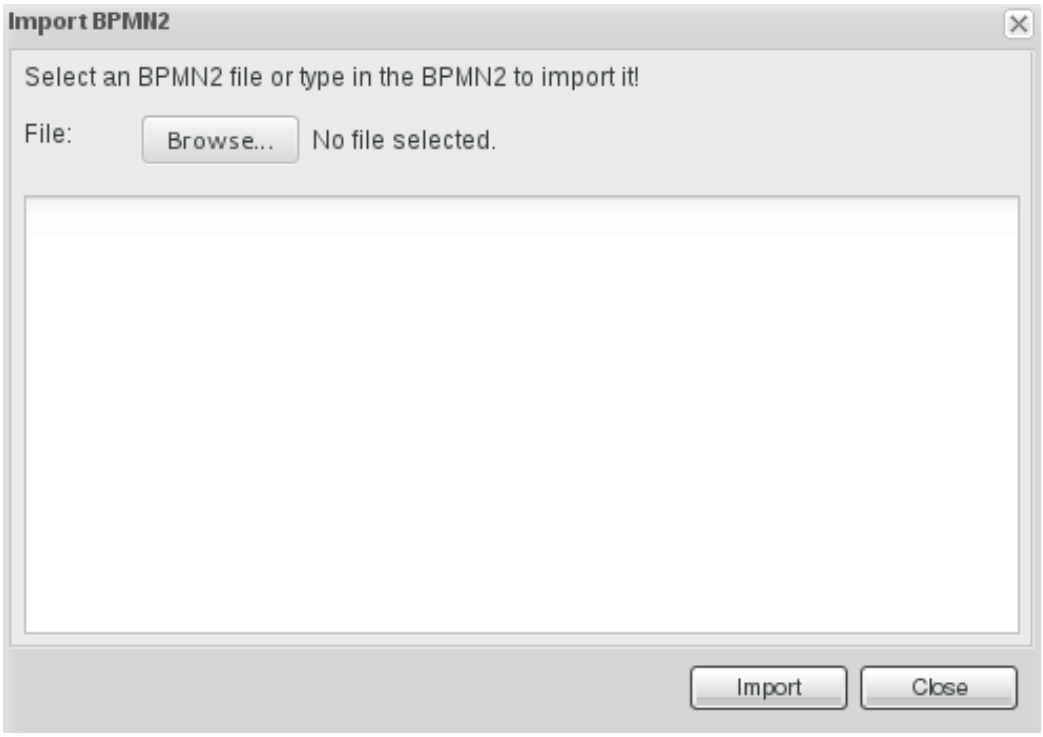


Figure 7.1. Import Window

Whenever a process definition is imported, the existing imported definition is overwritten. Make sure you are not overwriting a process definition you have edited so as not to lose any changes.

A process can also be imported to the git repository in the filesystem by cloning the repository, adding the process files, and pushing the changes back to git. In addition to alternative import methods, you can copy and paste a process or just open a file in the import dialog.

When importing processes, the Process Designer provides visual support for Process elements and therefore requires information on element positions on the canvas. If the information is not provided in the imported Process, you need to add it manually.

Report a bug

7.3. IMPORTING JPDL 3.2 TO BPMN2

To migrate and import a jPDL definition to BPMN2, in the Process Designer, click on the import button then scroll down and select **`Migrate jPDL 3.2 to BPMN2`**.

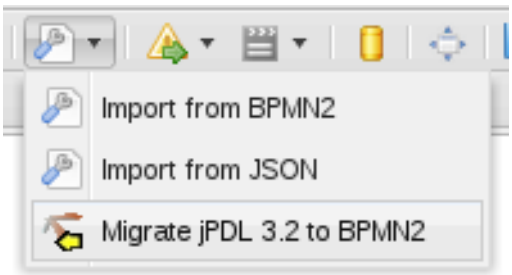


Figure 7.2. Migrate jPDL 3.2 to BPMN2

In the **`Migrate to BPMN2`** dialog box, select the process definition file and the name of the **`gpd`** file. Confirm by clicking the **`Migrate`** button.

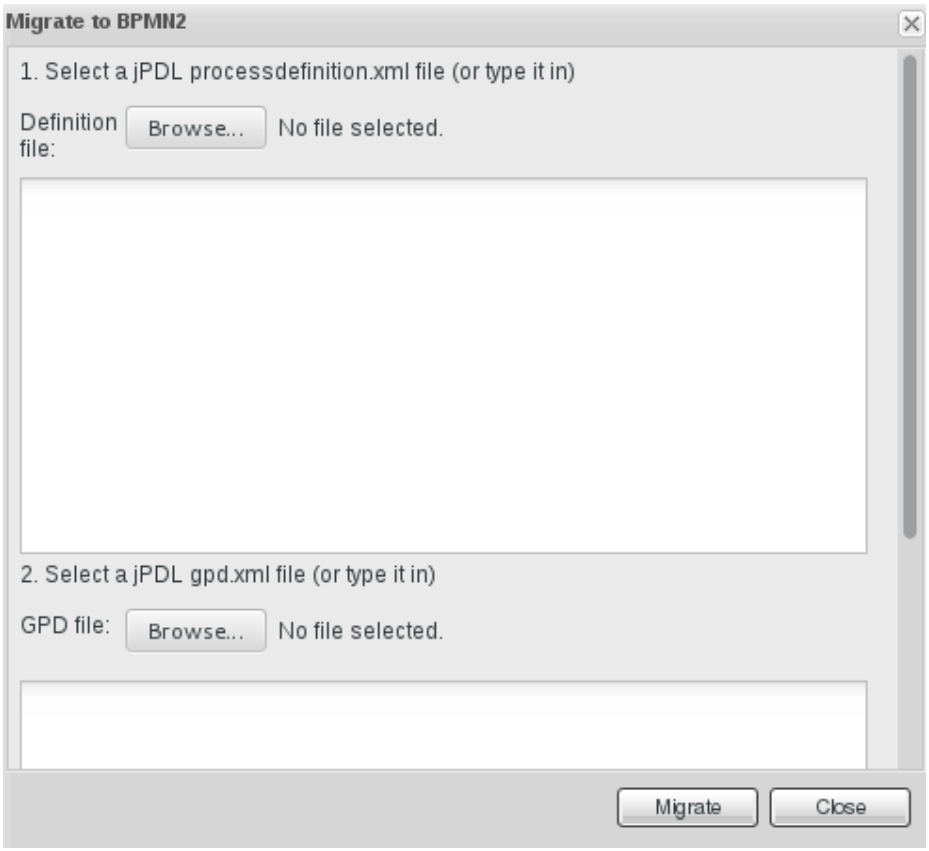


Figure 7.3. Migrate to BPMN2 dialog box

Report a bug

7.4. EXPORTING A PROCESS

Procedure 7.1. Exporting a business process

To export a business process, do the following:

1. Open the **`Project Authoring`** perspective: on the main menu, click **Authoring → Project Authoring**.
2. Select the business process which is to be exported, to view it in the Process Designer.
3. Click on the () button of the process designer toolbar and select **`View Process Sources`** from the drop-down options.
4. The **`Process Sources`** window is displayed
5. Click on the **`Download BPMN2`** button and save the business process at the desired location.

Report a bug

PART III. INTEGRATION

CHAPTER 8. INTEGRATION

Red Hat JBoss BPM Suite provides integration modules for integrating BPMS with other frameworks, namely Spring and Apache Camel. These modules are provided in the generic deployment package. See *Installation Guide* for more details.

This section provides information about these modules and how to configure them.

Report a bug

8.1. CONFIGURING RED HAT JBOSS BPM SUITE WITH APACHE SPRING

Refer to the Red Hat JBoss BPM Suite Installation Guide to download the Apache Spring module. You will need to download the generic deployable version of BPMS.

The Spring module is present in the **`jboss-bpms-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 BPMS 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>
```

Report a bug

8.2. CDI INTEGRATION

To make use of jbpm-kie-services in your system, you will need to provide some mbeans to satisfy all dependencies of the services. There are several mbeans that depend on actual scenarios.

- entity manager and entity manager factory
- user group callback for human tasks
- identity provider to pass authenticated user information to the services

When running in JEE environment, like JBoss Application Server, the mbean should satisfy all requirements of the jbpm-kie-services

```
public class EnvironmentProducer {

    @PersistenceUnit(unitName = "org.jbpm.domain")
    private EntityManagerFactory emf;

    @Inject
```

```
    @Selectable
    private UserGroupCallback userGroupCallback;

    @Produces
    public EntityManagerFactory getEntityManagerFactory() {
        return this.emf;
    }

    @Produces
    @RequestScoped
    public EntityManager getEntityManager() {
        EntityManager em = emf.createEntityManager();
        return em;
    }

    public void close(@Disposes EntityManager em) {
        em.close();
    }

    @Produces
    public UserGroupCallback produceSelectedUserGroupCalback() {
        return userGroupCallback;
    }
    @Produces

    public IdentityProvider produceIdentityProvider {
        return new IdentityProvider() {
             // implement IdentityProvider
        };
    }
}
```

Then **deployments/business-central.war/WEB-INF/beans.xml** file may be configured to change the current settings of the new **usergroupcallback** implementation.

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://docs.jboss.org/cdi/beans_1_0.xsd">

<alternatives>
  <class>org.jbpm.services.task.identity.JAASUserGroupCallbackImpl</class>
</alternatives>

</beans>
```

NOTE

org.jbpm.services.task.identity.JAASUserGroupCallbackImpl is just an example here to demonstrate the settings of the application server regardless of what it actually is (LDAP, DB, etc).

Report a bug

CHAPTER 9. DEPLOYING RED HAT JBOSS BPM SUITE ARTIFACTS TO SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) REPOSITORY

While BPMS and S-RAMP are two independent products, it is possible to move artifacts between them. You can move artifacts from BPMS to S-RAMP using Maven or via a user interface.

This section provides information about these two processes.

Report a bug

9.1. DEPLOYING RED HAT JBOSS BPM SUITE ARTIFACTS TO SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) USING MAVEN

Before you can deploy Red Hat JBoss BPM Suite artifacts to S-RAMP using Maven, you will need to enable the S-RAMP Maven Wagon. The Maven Wagon is a key feature that supports the S-RAMP Atom based REST API protocol. By enabling the S-RAMP Maven Wagon, users will be able to access artifacts from the S-RAMP repository as dependencies in a Maven project.

Enable the S-RAMP Maven Wagon by making an edit in the **pom.xml** file as shown below:

```
<build>
  <extensions>
    <extension>
      <groupId>org.overlord.sramp</groupId>
      <artifactId>s-ramp-wagon</artifactId>
      <version>${s-ramp-wagon.version}</version>
    </extension>
  </extensions>
</build>
```

Once the S-RAMP Maven Wagon is enabled, you can deploy the BPMS artifacts to that S-RAMP repository. To do this, follow the steps below:

1. Clone the git repository where you have saved the BPM Suite project by running this command:

   ```
   git clone http://localhost:8001/REPOSITORY_NAME
   ```

2. On the command line, move into the folder that contains the project.

3. Follow the instructions in *Red Hat JBoss Fuse Service Works 6 Development Guide, Volume 3: Governance*, section Deploying to S-RAMP. Use the URL from the example below:

   ```
   <distributionManagement>
     <repository>
       <id>local-sramp-repo</id>
       <name>S-RAMP Releases Repository</name>
       <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/</url>
     </repository>
     <snapshotRepository>
       <id>local-sramp-repo-snapshots</id>
       <name>S-RAMP Snapshots Repository</name>
   ```

```
    <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/</url>
  </snapshotRepository>
</distributionManagement>
```

With these settings, Maven deployments are sent directly to the S-RAMP repository using the S-RAMP API. Note that artifacts are added to the S-RAMP repository with an artifact type based on the Maven type of the project. You can override this behavior by adding a query parameter to the repository URL in the **pom.xml** file. For example:

```
<distributionManagement>
  <repository>
    <id>local-sramp-repo</id>
    <name>S-RAMP Releases Repository</name>
    <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/?
artifactType=KieJarArchive</url>
  </repository>
</distributionManagement>
```

The above example causes the Maven artifact to be uploaded with an S-RAMP artifact type of KieJarArchive.

4. Amend the maven plug-in in file **pom.xml** and add a dependency to it as follows in case the project does not contain decision tables:

```
<plugins>
    <plugin>
      <groupId>org.kie</groupId>
      <artifactId>kie-maven-plugin</artifactId>
      <version>6.0.2-redhat-6</version>
      <extensions>true</extensions>
      <dependencies>
          <dependency>
              <groupId>org.jbpm</groupId>
              <artifactId>jbpm-bpmn2</artifactId>
              <version>6.0.2-redhat-6</version>
          </dependency>
      </dependencies>
     </plugin>
    </plugins>
```

If the project contains decision tables, use this dependency for the kie-maven-plugin instead:

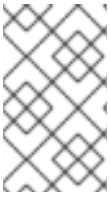
```
<plugins>
    <plugin>
      <groupId>org.kie</groupId>
      <artifactId>kie-maven-plugin</artifactId>
      <version>6.0.2-redhat-6</version>
      <extensions>true</extensions>
      <dependencies>
          <dependency>
               <groupId>org.drools</groupId>
               <artifactId>drools-decisiontables</artifactId>
               <version>6.0.2-redhat-6</version>
           </dependency>
```

```
      </dependencies>
    </plugin>
  </plugins>
```

5. Run a clean Maven deployment using the following command:

```
mvn -s sramp-settings.xml deploy
```

.



NOTE

For the Maven deployment to the S-RAMP repository, it is necessary to have credentials set in the **`settings.xml`** file. For further details on the credentials, refer to *Red Hat JBoss Fuse Service Works* (FSW) documentation on Authentication.

Report a bug

9.2. DEPLOYING RED HAT JBOSS BPM SUITE ARTIFACTS TO SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) USING GRAPHICAL USER INTERFACE (GUI)

To deploy BPMS artifacts to a S-RAMP repository using the user interface, do the following:

1. In a web browser, navigate to http://localhost:8080/s-ramp-ui/. If the user interface has been configured to run from a domain name, substitute **`localhost`** for the domain name. For example http://www.example.com:8080/s-ramp-ui/.
2. Click on **`Artifacts`**.
3. In the **`Manage Artifacts`** section, select **`Import`**.
4. Locate the kie archive you want to deploy. In the dialog that opens, fill out **`KieJarArchive`** as the type, and select **`Import`**.
5. The deployment then creates these entries in the S-RAMP repository:

   `KieJarArchive`, from which it derives:

   - **`KieXmlDocument`** (if the archive contains **`kmodule.xml`**)
   - **`BpmnDocument`** (if the archive contains **`bpmn`** definitions)
   - **`DroolsDocument`** (if the archive contains **`drl`** definitions)

Report a bug

CHAPTER 10. PERSISTENCE

The runtime data of the Process Engine can be persisted in data stores. The persistence mechanism saves the data using marshalling: the runtime data is converted into a binary dataset and the dataset is saved in the data storage.

Note that persistence is not configured by default and the engine runs without persistence.

NOTE

The runtime data is saved using marshalling (binary persistence). The marshalling mechanism is a custom serialization mechanism.

Red Hat JBoss BPMS will persist the following when persistence is configured:

- Session state: this includes the session ID, date of last modification, the session data that business rules would need for evaluation, state of timer jobs.
- Process instance state: this includes the process instance ID, process ID, date of last modification, date of last read access, process instance start date, runtime data (the execution status including the node being executed, variable values, etc.)and the eventtypes .
- Work item runtime state: this includes the work item ID, creation date, name, process instance ID, and the work item state itself.

Based on the persisted data, it is possible to restore the state of execution of all running process instances in case of failure or to temporarily remove running instances from memory and restore them later. By default, no persistence is configured.

To allow persistence, you need to add the jbpm-persistence jar files to the classpath of your application and configure the engine to use persistence. The engine automatically stores the runtime state in the storage when the engine reaches a safe point. Safe points are points where the process instance has paused. When a process instance invocation reaches a safe point in the engine, the engine stores any changes to the process instance as a snapshot of the process runtime data. However, when a process instance is completed, the persisted snapshot of process instance runtime data is automatically deleted.

If a failure occurs and you need to restore the engine runtime from the storage, the process instances are automatically restored and their execution resumes so there is no need to reload and trigger the process instances manually.

The runtime persistence data is to be considered internal to the engine. You should not access persisted runtime data or modify them directly as this might have unexpected side effects.

To obtain information about the current execution state, refer to the history log. Query the database for runtime data only if absolutely necessary.

Report a bug

10.1. SESSION

Sessions are persisted as **`sessioninfo`** entities. These persist the state of the runtime knowledge session, and store the following data:

Table 10.1.

Field	Description	Nullable
id	primary key	false
lastmodificationdate	last saved to data store	N/A
rulesbytearray	binary dataset with session state (binary blob)	false
startdate	session start	
optlock	version number used lock value for optimistic locking	

Report a bug

10.2. PROCESS INSTANCE

Process instances are persisted as **`ProcessInstanceInfo`** entities, which persist the state of a process instance on runtime and store the following data:

Table 10.2.

Field	Description	Nullable
instanceid	primary key	false
lastmodificationdate	last saved to data store	N/A
lastreaddate	last read from data store	N/A
processid	ID of the process the instance is based on	false
processinstancebytearray	binary dataset with process instance state (binary blob)	false
startdate	Process instance start	
optlock	version number used lock value for optimistic locking	
state	Process instance state	false

`ProcessInstanceInfo` has a 1:N relationship to the **`EventTypes`** entity.

The **`EventTypes`** entity contains the following data:

Table 10.3.

Field	Description	Nullable
instanceid	reference to the Process instance (foreign key to the **`processinstanceinfo`**)	false
element	text field related to an event the Process instance has undergone	

Report a bug

10.3. WORK ITEM

Work Items are persisted as **`workiteminfo`** entities, which persist the state of the particular work item instance on runtime and store the following data:

Table 10.4.

Field	Description	Nullable
workitemid	primary key	false
name	work item name	
processinstanceid	parent Process instance id	false
state	integer representing work item state	false
optlock	version number used lock value for optimistic locking	
workitembytearray	binary dataset with work item state (binary blob)	false
creationDate	timestampe on which the work item was created	false

Report a bug

10.4. PERSISTENCE CONFIGURATION

10.4.1. Persistence configuration

Although persistence is not used by default, the dependencies needed are available in the runtime directory as jar files .

Persistence is defined per session and you can define it either using the **`JBPMHelper`** class after you create a session or using the **`JPAKnowledgeService`** to create your session. The latter option provides more flexibility, while **`JBPMHelper`** has a method to create a session, and uses a configuration file to configure this session.

Report a bug

10.4.2. Configuring persistence using JBPMHelper

To configure persistence of your session using JBPMHelper, do the following:

1. Define your application to use an appropriate JBPMHelper session construtor:
   - **`KieSession ksession = JBPMHelper.newKieSession(kbase);`**
   - **`KieSession ksession = JBPMHelper.loadKieSession(kbase, sessionId);`**
2. Configure the persistence in the **`jBPM.properties`** file.

Example 10.1. A sample jBPM.properties file with persistence for the in-memory H2 database

```
# for creating a datasource
persistence.datasource.name=jdbc/jbpm-ds
persistence.datasource.user=sa
persistence.datasource.password=
persistence.datasource.url=jdbc:h2:tcp://localhost/~/jbpm-db
persistence.datasource.driverClassName=org.h2.Driver

# for configuring persistence of the session
persistence.enabled=true
persistence.persistenceunit.name=org.jbpm.persistence.jpa
persistence.persistenceunit.dialect=org.hibernate.dialect.H2Dialec
t

# for configuring the human task service
taskservice.enabled=true
taskservice.datasource.name=org.jbpm.task
taskservice.transport=mina
taskservice.usergroupcallback=org.jbpm.task.service.DefaultUserGro
upCallbackImpl
```

Any invocations on the session will now trigger the persistance process.

Make sure the datasource is up and running on engine start. If you are running the in-memory H2 database, you can start the database from your application using the **`JBPMHelper.startH2Server();`** method call and register it with the engine using **`JBPMHelper.setupDataSource();`** method call.

Report a bug

10.4.3. Configuring persistence using JPAKnowledgeService

To create your knowledge session and configure its persistence using JPAKnowledgeService, do the following:

1. Define your application to use the knowledge session created by JPAKnowledgeService:

- Define the session based on a knowledge base, a knowledge session configuration, and an environment. The environment must contain a reference to your Entity Manager Factory:

```
// create the entity manager factory and register it in the
environment
EntityManagerFactory emf =
Persistence.createEntityManagerFactory(
"org.jbpm.persistence.jpa" );
Environment env = KnowledgeBaseFactory.newEnvironment();
env.set( EnvironmentName.ENTITY_MANAGER_FACTORY, emf );

// create a new knowledge session that uses JPA to store the
runtime state
KieSession ksession = JPAKnowledgeService.newKieSession( kbase,
null, env );
int sessionId = ksession.getId();

// invoke methods on your method here
ksession.startProcess( "MyProcess" );
ksession.dispose();
```

- Define the session based on a specific session id.

```
// recreate the session from database using the sessionId
ksession = JPAKnowledgeService.loadKieSession(sessionId, kbase,
null, env );
```

2. Configure the persistence in the **`META-INF/persistence.xml`** file: configure JPA to use Hibernate and the respective database.

   Information on how to configure data source on your application server should be available in the documentation delivered with the application server. For this information for JBoss Enterprise Application Platform, see the *Administration and Configuration Guide* for this product.

Example 10.2. A sample persistence.xml file with persistence for an H2 data source `jdbc/jbpm-ds`

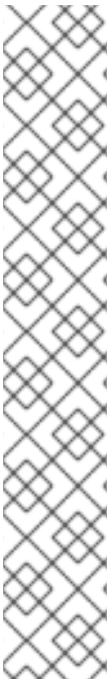
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence
  version="1.0"
  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/persistence
     http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd
     http://java.sun.com/xml/ns/persistence/orm
     http://java.sun.com/xml/ns/persistence/orm_1_0.xsd"
  xmlns:orm="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/persistence">
  <persistence-unit name="org.jbpm.persistence.jpa" transaction-
type="JTA">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>jdbc/jbpm-ds</jta-data-source>
    <mapping-file>META-INF/JBPMorm.xml</mapping-file>
    <class>org.drools.persistence.info.SessionInfo</class>
```

```
<class>org.jbpm.persistence.processinstance.ProcessInstanceInfo</c
lass>
    <class>org.drools.persistence.info.WorkItemInfo</class>
    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.transaction.manager_lookup_class"

value="org.hibernate.transaction.BTMTransactionManagerLookup"/>
    </properties>
  </persistence-unit>
</persistence>
```

Any invocations on the session will now trigger the persistance process.

Make sure the datasource is up and running on engine start. If you are running the in-memory H2 database, you can start the database from your application using the **`JBPMHelper.startH2Server();`** method call and register it with the engine using **`JBPMHelper.setupDataSource();`** method call.



NOTE

If you are running JBoss BPMS in a simple Java environment, your data source configuration will be similar to the following:

```
PoolingDataSource ds = new PoolingDataSource();
ds.setUniqueName("jdbc/jbpm-ds");
ds.setClassName("bitronix.tm.resource.jdbc.lrc.LrcXADataSource"
);
ds.setMaxPoolSize(3);
ds.setAllowLocalTransactions(true);
ds.getDriverProperties().put("user", "sa");
ds.getDriverProperties().put("password", "sasa");
ds.getDriverProperties().put("URL",
"jdbc:h2:tcp://localhost/~/jbpm-db");
ds.getDriverProperties().put("driverClassName",
"org.h2.Driver");
ds.init();
```

Report a bug

CHAPTER 11. LOGGING

The logging mechanism allows you to store information about the execution of a process instance. It is provided by a special event listener that listens to the Process Engine for any relevant events to be logged, so that the information can be stored separately from other non-log information stored either in the server built-in database (h2) or a connected data source using JPA or Hibernate.

The jbpm-audit module provides the event listener and also allows you to store process-related information directly in a database using JPA or Hibernate. The data of the following entities is stored as follows:

- Process instance as **`processinstancelog`**
- Element instance as **`nodeinstancelog`**
- Variable instance as **`variableinstancelog`**

Table 11.1. Fields of the ProcessInstanceLog table

Field	Description	Nullable
id	The primary key of the log entity	No
end_date	The end date of the process instance	Yes
processid	The name (id) of the underlying process	Yes
processinstanceid	The id of the process instance	No
start_date	The start date of the process instance	Yes
status	The status of the process instance	Yes
parentProcessInstanceId	The process instance id of the parent process instance if applicable	Yes
outcome	The outcome of the process instance (details on the process finish, such as error code)	Yes

Table 11.2. Fields of the NodeInstanceLog table

Field	Description	Nullable
id	The primary key of the log entity	No
log_date	The date of the event	Yes

Field	Description	Nullable
nodeid	The node id of the underlying Process Element	Yes
nodeinstanceid	The id of the node instance	Yes
nodename	The name of the underlying node	Yes
processid	The id of the underlying process	Yes
processinstanceid	The id of the parent process instance	No
type	The type of the event (`0` = enter event, `1` = exit event)	No

Table 11.3. Fields of the VariableInstanceLog table

Field	Description	Nullable
id	The primary key of the log entity	No
log_date	The date of the event	Yes
processid	The name (id) of the underlying process	Yes
processinstanceid	The id of the process instance	No
value	The value of the variable at log time	Yes
variableid	The variable id as defined in the process definition	Yes
variableinstanceid	The id of the variable instance	Yes
outcome	The outcome of the process instance (details on the process finish, such as error code)	Yes

If neccessary, define your own data model of custom information and use the process event listeners to extract the information.

Report a bug

11.1. LOGGING EVENTS TO DATABASE

To log an event that occurs on runtime in a Process instance, an Element instance, or a variable instance, you need to do the following:

1. Map the Log classes to the data source, so that the given data source accepts the log entries. On Red Hat JBoss EAP, edit the data source properties in the **`persistence.xml`** file.

Example 11.1. The ProcessInstanceLog, NodeInstanceLog and VariableInstanceLog classes enabled for processInstanceDS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence  version="1.0"  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/persistence
     http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd
     http://java.sun.com/xml/ns/persistence/orm
     http://java.sun.com/xml/ns/persistence/orm_1_0.xsd"
  xmlns:orm="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/persistence">

  <persistence-unit name="org.jbpm.persistence.jpa">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>jdbc/processInstanceDS</jta-data-source>
    <class>org.drools.persistence.info.SessionInfo</class>

<class>org.jbpm.persistence.processinstance.ProcessInstanceInfo</c
lass>
    <class>org.drools.persistence.info.WorkItemInfo</class>
    <class>org.jbpm.process.audit.ProcessInstanceLog</class>
    <class>org.jbpm.process.audit.NodeInstanceLog</class>
    <class>org.jbpm.process.audit.VariableInstanceLog</class>

    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.transaction.manager_lookup_class"

value="org.hibernate.transaction.BTMTransactionManagerLookup"/>
    </properties>
  </persistence-unit>
</persistence>
```

2. Register a logger on your Kie Session.

Example 11.2. Import the Loggers

```
import org.jbpm.process.audit.AuditLogService;
import org.jbpm.process.audit.AuditLoggerFactory;
import org.jbpm.process.audit.AuditLoggerFactory.Type;
import org.jbpm.process.audit.JPAAuditLogService;
...
```

Example 11.3. Registering a Logger to a Kie Session

```
@PersistenceUnit(unitName = PERSISTENCE_UNIT_NAME)
  private EntityManagerFactory emf;

  private AuditLogService auditLogService;
@PostConstruct
  public void configure() {

  auditLogService = new JPAAuditLogService(emf);
  ((JPAAuditLogService)
auditLogService).setPersistenceUnitName(PERSISTENCE_UNIT_NAME);

  if( emf == null ) {
  ((JPAAuditLogService)
auditLogService).setPersistenceUnitName(PERSISTENCE_UNIT_NAME);
  }

  RuntimeEngine runtime =
singletonManager.getRuntimeEngine(EmptyContext.get());
  KieSession ksession = runtime.getKieSession();
  AuditLoggerFactory.newInstance(Type.JPA, ksession, null);

  }
```

3. Optionally, call the method **`addFilter`** on the logger to filter out irrelevant information. Only information accepted by all filters appears in the database.

4. Logger classes can be viewed in the Audit View:

```
<dependency>
 <groupId>org.jbpm</groupId>
 <artifactId>jbpm-audit</artifactId>
 <version>6.0.1.Final</version>
</dependency>
```

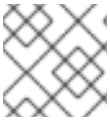
Report a bug

CHAPTER 12. LOCALIZATION AND CUSTOMIZATION

12.1. AVAILABLE LANGUAGES

The Red Hat JBoss BPM Suite 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.

Report a bug

12.2. 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
```

Changing the User Interface Language in Dashbuilder

To change the user interface language in dashbuilder, do the following:

1. Log into the dashbuilder after the server has been successfully started by navigating to http://localhost:8080/dashbuilder in a web browser.
2. Select the language of your choice by clicking on the available locales on the top center of the dashbuilder user interface to change the language.

Setting a Default User Interface Language in Dashbuilder

Following is an example to set the default user interface language in dashbuilder:

Procedure 12.1. Setting the default language as French

1. Navigate to **jboss-eap-6.1/standalone/configuration** and define the following in the **standalone.xml** file.

```
<system-properties>
```

```
    <property
name="org.jboss.dashboard.LocaleManager.installedLocaleIds"
value="en,es,de,fr,ja,pt,zh"/>
    <property
name="org.jboss.dashboard.LocaleManager.defaultLocaleId"
value="fr"/>
</system-properties>
```

2. The default user interface language of the dashbuilder is now set to French.

Defining the Installed Locales in Dashbuilder

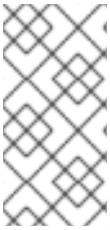
Following is an example to define the installed locales in dashbuilder:

Procedure 12.2. Defining the installed locale

- Navigate to **`jboss-eap-6.1/standalone/configuration`** and define the following in the **`standalone.xml`** file.

```
<system-properties>
    <property
name="org.jboss.dashboard.LocaleManager.installedLocaleIds"
value="en,es,de,fr,ja,pt"/>
    <property
name="org.jboss.dashboard.LocaleManager.defaultLocaleId"
value="fr"/>
</system-properties>
```

In this example, the Chinese language (zh) has been removed from the list of installed locales so users will not be able to switch the dashbuilder to Chinese. Dashbuilder will show content in French, which is the default locale. Users will be able to select other languages that are defined (en, es, de, ja, pt) in this file.



NOTE

Within Business Central, the application server does not need to be restarted after changing locale if you append the "locale" parameter to the URL of Business Central. However, with Dashbuilder, the application server should be restarted after the configuration files have been changed.

Report a bug

12.3. RUNNING THE JVM WITH UTF-8 ENCODING

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

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

Report a bug

PART IV. EXECUTION

CHAPTER 13. EXECUTION SERVER

13.1. 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.

Report a bug

13.1.1. 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.drl** with the rules to be checked when the Human Task is created
   - **default-complete-task.drl** with the rules to be checked when the Human Task is completed
2. Define the rules in the file.

Example 13.1. The `default-add-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 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 13.2. 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**.

Report a bug

13.2. 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 BPMS User Guide*).

Report a bug

13.2.1. 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 13.3. New mail session on localhost

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

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

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

Example 13.4. Outbound socket definition

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

Report a bug

CHAPTER 14. PLUG-IN FOR RED HAT JBOSS DEVELOPER STUDIO

14.1. PLUG-IN

Report a bug

PART V. MONITORING

CHAPTER 15. PROCESS MONITORING

15.1. JBOSS OPERATIONS NETWORK

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

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

Report a bug

15.2. INSTALLING THE BRMS PLUG-IN INTO JBOSS ON

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 15.1. 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-3.2.0.GA. For example:

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

2. Copy the extracted 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-
3.2.0.GA/*.jar /opt/jon/jon-server-3.2.0.GA1/plugins
```

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

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

Procedure 15.2. Uploading the 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 BRMS plug-in for JBoss Operations Network is now uploaded.

Report a bug

15.3. MONITORING KIE BASES AND KIE SESSIONS

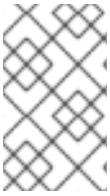
In order for JBoss Operations Network to monitor KieBases and KieSessions, MBeans must be enabled.

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 BRMS 6; for BRMS 5, **`Drools Services`** was the naming convention used and it had different measurements on sessions. For example, **activation** → **match** renaming occured 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.

Report a bug

CHAPTER 16. MANAGING SECURITY FOR RED HAT JBOSS BPM SUITE DASHBUILDER

16.1. ACCESSING RED HAT JBOSS BPM SUITE DASHBUILDER

Dashbuilder is the Red Hat JBoss BPM Suite web-based user interface for Business Activity Monitoring. To access the Dashbuilder from Business Central, go to **Dashboards** → **Process & Task Dashboards**.

The displayed dashboard provides statistics on runtime data selected on the left. You can create your own dashboard in the Dashbuilder. To do so, display the Dashbuilder by clicking **Dashboards** → **Business Dashboards**.

Report a bug

16.2. MANAGING SECURITY

To manage security, you can define custom authorization policies to grant or deny access to workspace, page, or panel instances per role.

Defined below is a list of the available roles for Dashbuilder:

- admin - Administrates the BPMS system. Has full access rights to make any changes necessary. Also has the ability to add and remove users from the system.
- developer - Implements code required for process to work. Mostly uses the JBDS connection to view processes, but may use the web tool occasionally.:
- analyst - Responsible for creating and designing processes into the system. Creates process flows and handles process change requests. Needs to test processes that they create. Also creates forms and dashboards.
- user - Daily user of the system to take actions on business tasks that are required for the processes to continue forward. Works primarily with the task lists.
- manager - Viewer of the system that is interested in statistics around the business processes and their performance, business indicators, and other reporting of the system and people who interact with the system.

Thanks to the permissions system, you can build a workspace structure with several pages, menus, and panels and define what pages and panels within a page will be visible for each role. You can also define special types of users and give them restricted access to certain tooling features, or even restricted access to a page subset.

Report a bug

16.3. WORKSPACE PERMISSIONS

Procedure 16.1. Accessing Workspace Permissions

1. Log into Business Dashboards from Business Central (as described in the Accessing Red Hat JBoss BPM Suite Dashbuilder topic).
2. Select the appropriate Dashboard from the Wokspace drop-down.

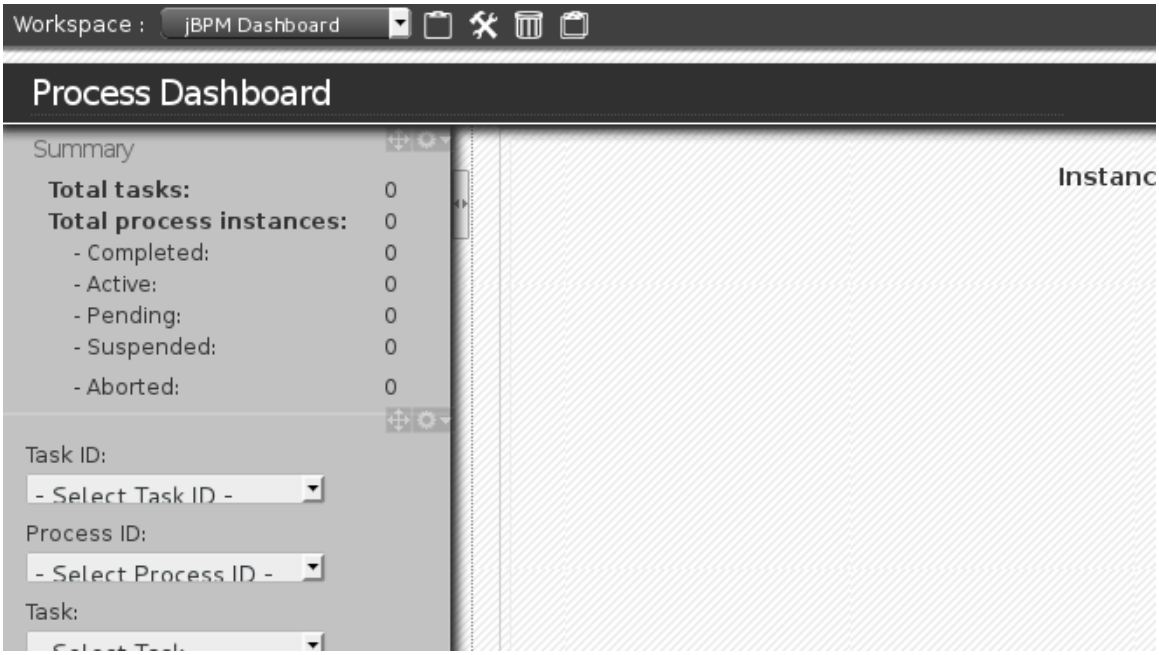


Figure 16.1. Dashbuilder Workspace

3. Click the **`Edit selected workspace properties`**  button to access the Workspace Dashboard.

4. Click the **`Permissions`** label to view the permission management screen.

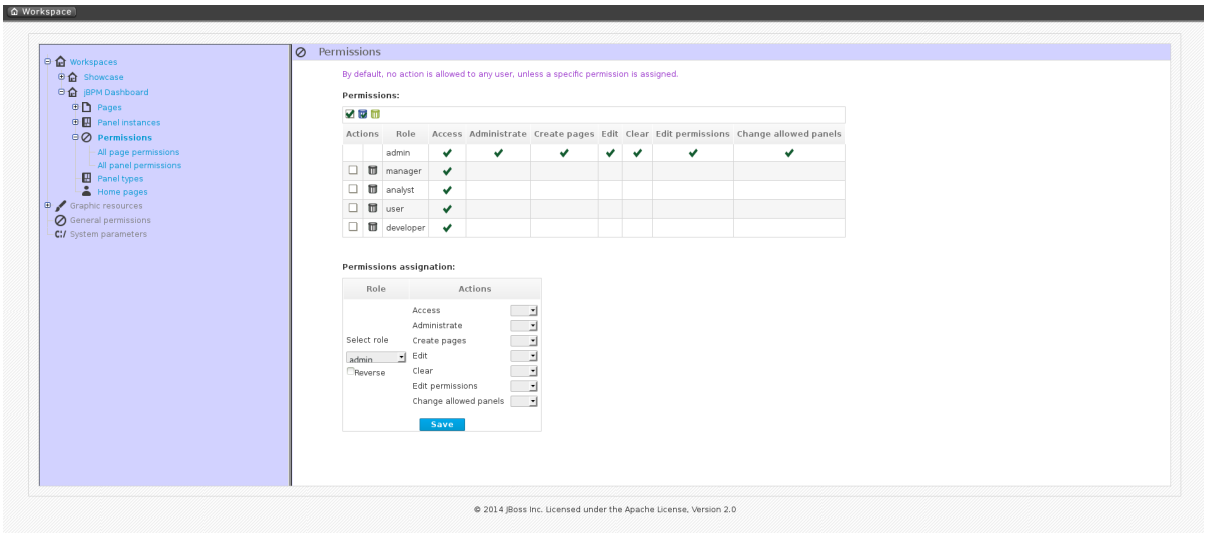


Figure 16.2. Permissions Screen

Under the **`Permissions assignation`** section is a list of allowed actions that are applied to the selected role:

- **Access**: permission to login into the application.
- **Administrate**: permission to access the toolbar and system configuration features.
- **Create pages**: ability to create new project pages.
- **Edit**: permission to change the workspace properties.
- **Clear**: ability to delete the workspace.

- **Edit permissions**: ability to grant/deny permissions.
- **Change allowed panels**: permission to restrict the type of panels that can be used in this workspace.

To assign a permission you must select the target role and the list of actions allowed over the selected resource.

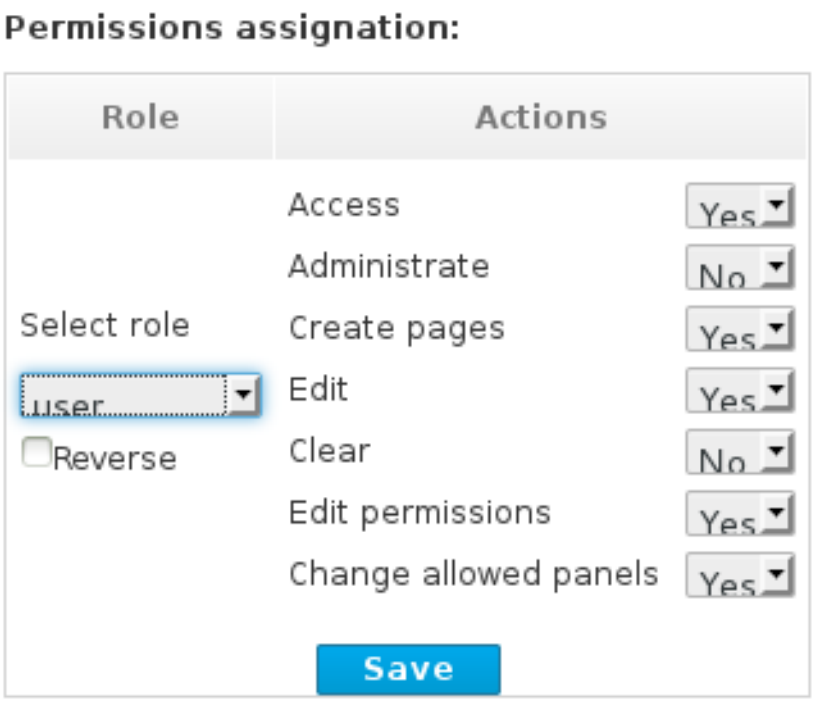


Figure 16.3. Permissions Assignation

- *Target roles (who)*: What user will be granted/denied with the permissions defined.
- *Allowed actions*: depending on the type of the resource we can enable/disable what the user can do on this resource.
- *Reverse (optional)*: when we have a set of roles and we want to grant/deny a permission to all the roles but one.

NOTE

By default, the full set of permissions go to the role *admin*. This makes it easy to create a user that can do everything as long as the role *admin* is assigned.

Report a bug

16.4. PAGE PERMISSIONS

1. To access **`Page permissions`**, locate the **`Pages`** drop-down under the jBPM Dashboard (or whatever Dashboard you selected).
2. After expanding **`Pages`**, expand the **`Process dashboard`** option.

3. Select the **Page permissions** option.

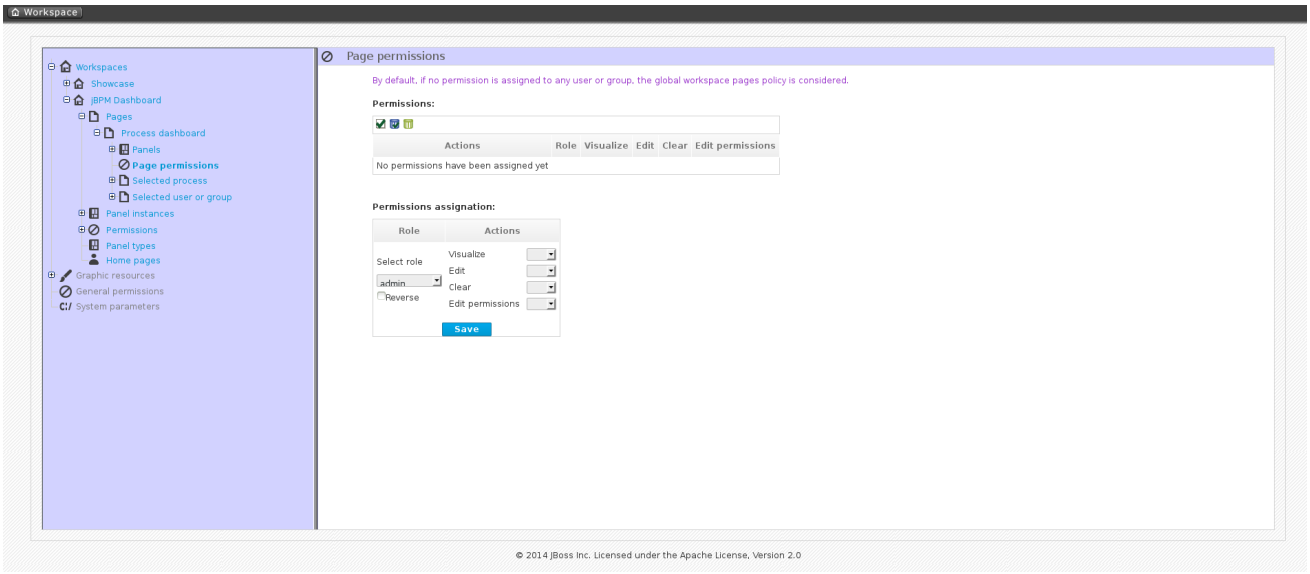


Figure 16.4. Page Permissions

Under the **Permissions assignation** section is a list of allowed actions that are applied to the selected role:

- **Visualize**: permission to make the page visible.
- **Edit**: ability to change the page properties.
- **Clear**: ability to delete the page.
- **Edit permissions**: ability to grant/deny permissions for the page.

Report a bug

16.5. PANEL PERMISSIONS

1. To access the **Panel permissions** page, expand the **Panel instances** option under the jBPM Dashboard (or whatever Dashboard you are using).
2. Expand the **Dashboard** option and then expand the **Process dashboard**.
3. Expand the **Panels** choice and select the appropriate process.
4. Open the **Panel permissions** page.

Below is a screenshot of the permission management screen for a given panel (in this example, the Process dashboard):

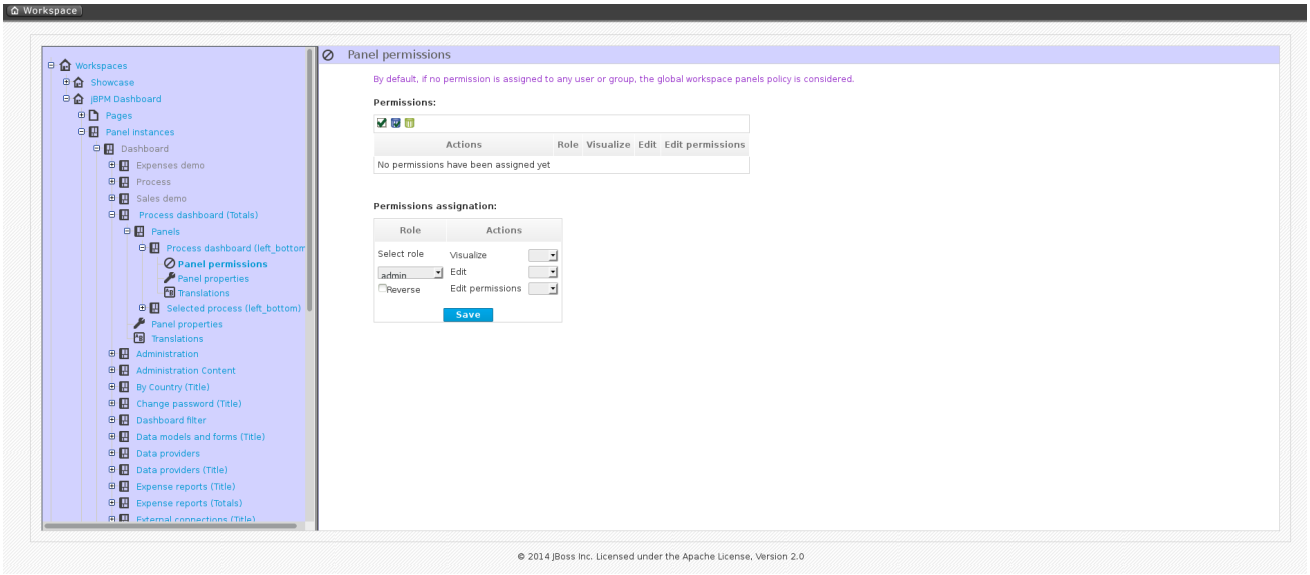


Figure 16.5. Panel permissions configuration screen

Allowed actions are the following:

- **Visualize**: make the panel visible.
- **Edit**: change the panel properties.
- **Edit permissions**: ability to grant/deny permissions for the panel.

Report a bug

APPENDIX A. REVISION HISTORY

Revision 1.0.0-30 **Thu Mar 05 2015** **Vikram Goyal**

Built from Content Specification: 22692, Revision: 745885 by vigoyal