Red Hat Process Automation Manager
7.12

Managing Red Hat Process Automation Manager and KIE Server settings
Red Hat Process Automation Manager 7.12 Managing Red Hat Process Automation Manager and KIE Server settings
Abstract

This document describes how to modify Red Hat Process Automation Manager and KIE Server settings and properties to meet your business needs.
Table of Contents

PREFACE .................................................................................................................. 7

MAKING OPEN SOURCE MORE INCLUSIVE ............................................................... 8

PART I. MANAGING AND MONITORING KIE SERVER ............................................. 9

CHAPTER 1. RED HAT PROCESS AUTOMATION MANAGER COMPONENTS ............. 10

CHAPTER 2. SYSTEM INTEGRATION WITH MAVEN .................................................... 11
  2.1. PREEMPTIVE AUTHENTICATION FOR LOCAL PROJECTS .............................. 11
  2.2. DUPLICATE GAV DETECTION IN BUSINESS CENTRAL ............................... 12
  2.3. MANAGING DUPLICATE GAV DETECTION SETTINGS IN BUSINESS CENTRAL 12

CHAPTER 3. APPLYING PATCH UPDATES AND MINOR RELEASE UPGRADES TO RED HAT PROCESS AUTOMATION MANAGER ................................................................. 14

CHAPTER 4. CONFIGURING AND STARTING KIE SERVER ....................................... 19

CHAPTER 5. CONFIGURING JDBC DATA SOURCES FOR KIE SERVER ....................... 21

CHAPTER 6. MANAGED KIE SERVER ........................................................................ 24

CHAPTER 7. UNMANAGED KIE SERVER .................................................................... 25

CHAPTER 8. CONFIGURING THE ENVIRONMENT MODE IN KIE SERVER AND BUSINESS CENTRAL ............................................................. 26

CHAPTER 9. CONFIGURING KIE SERVER TO CONNECT TO BUSINESS CENTRAL .......... 27

CHAPTER 10. INSTALLING AND RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER .............................................................................. 30
  10.1. USING THE INSTALLER TO CONFIGURE KIE SERVER WITH THE PROCESS AUTOMATION MANAGER CONTROLLER ...................................................... 30
  10.2. INSTALLING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER .......................................................... 31
    10.2.1. Creating a headless Process Automation Manager controller user ............... 32
    10.2.2. Configuring KIE Server and the headless Process Automation Manager controller 33
  10.3. RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER .......................................................... 34
  10.4. CLUSTERING KIE SERVERS WITH THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER ............................................................ 35

CHAPTER 11. CONFIGURING SMART ROUTER FOR TLS SUPPORT ............................. 37

CHAPTER 12. ACTIVATING OR DEACTIVATING A KIE CONTAINER ON KIE SERVER .......... 38

CHAPTER 13. DEPLOYMENT DESCRIPTORS .................................................................. 39
  13.1. DEPLOYMENT DESCRIPTOR CONFIGURATION ............................................. 39
    What Can You Configure? .............................................................................. 39
  13.2. MANAGING DEPLOYMENT DESCRIPTORS .................................................. 41
  13.3. RESTRICTING ACCESS TO THE RUNTIME ENGINE ......................................... 41

CHAPTER 14. ACCESSING RUNTIME DATA FROM BUSINESS CENTRAL ...................... 43

CHAPTER 15. PROMETHEUS METRICS MONITORING IN RED HAT PROCESS AUTOMATION MANAGER ................................................................. 44
  15.1. CONFIGURING PROMETHEUS METRICS MONITORING FOR KIE SERVER .......................................................... 44
  15.2. CONFIGURING PROMETHEUS METRICS MONITORING FOR KIE SERVER ON RED HAT OPENSHEET CONTAINER PLATFORM .................................................. 51
  15.3. EXTENDING PROMETHEUS METRICS MONITORING IN KIE SERVER WITH CUSTOM METRICS .......................................................... 55
42.1. CREATING POST-COMMIT GIT HOOKS
42.2. IMPORTING REMOTE GIT REPOSITORIES
42.3. CONFIGURING GIT HOOKS FOR EXISTING REMOTE GIT PROJECT REPOSITORIES
42.4. CONFIGURING GIT HOOKS AS A SYSTEM PROPERTY FOR BUSINESS CENTRAL
42.5. INTEGRATING REMOTE GIT REPOSITORIES
42.6. GIT HOOK EXIT CODES
42.7. CUSTOMIZING GIT HOOK NOTIFICATIONS
   42.7.1. Git hook notifications in Business Central
   42.7.2. Git hook notification internationalization support

CHAPTER 43. ROLE-BASED ACCESS CONTROL FOR BRANCHES IN BUSINESS CENTRAL ............... 159
  43.1. CUSTOMIZING ROLE-BASED BRANCH ACCESS 159

CHAPTER 44. VIEWING PROCESS INSTANCE LOGS ............................................................. 160

CHAPTER 45. BUSINESS CENTRAL SYSTEM PROPERTIES .................................................. 161

CHAPTER 46. PERFORMANCE TUNING CONSIDERATIONS WITH BUSINESS CENTRAL ............ 167

PART III. USING STANDALONE PERSPECTIVES IN BUSINESS CENTRAL ......................... 168

CHAPTER 47. STANDALONE PERSPECTIVES IN BUSINESS CENTRAL ............................... 169

CHAPTER 48. USING THE STANDALONE LIBRARY PERSPECTIVE ..................................... 170

CHAPTER 49. USING THE STANDALONE EDITOR PERSPECTIVE ....................................... 171

CHAPTER 50. USING THE STANDALONE CONTENT MANAGER PERSPECTIVE ....................... 172

CHAPTER 51. USING STANDALONE CUSTOM PAGES (DASHBOARDS) ................................. 173

PART IV. CREATING CUSTOM PAGES IN BUSINESS CENTRAL ......................................... 174

CHAPTER 52. BUSINESS CENTRAL CUSTOM DASHBOARDS ............................................ 175

CHAPTER 53. DASHBUILDER RUNTIME AND DASHBUILDER STANDALONE ..................... 176
  53.1. INSTALLING DASHBUILDER RUNTIME ON RED HAT JBOSs EAP 176
     53.1.1. Dashbuilder Runtime system properties 177
  53.2. DEPLOYING DASHBUILDER STANDALONE ON RED HAT OPENSHIFT CONTAINER PLATFORM 178
     53.2.1. Dashbuilder Standalone environment variables 180

CHAPTER 54. DATA SETS AUTHORING ............................................................... 183
  54.1. ADDING DATA SETS 183
  54.2. EDITING DATA SETS 184
  54.3. DATA REFRESH 184
  54.4. CACHING DATA
     Client cache 185
     Back-end cache 185
  54.5. KIE SERVER DATA SETS WITH DASHBUILDER RUNTIME AND DASHBUILDER STANDALONE 185

CHAPTER 55. PAGE AUTHORING .......................................................... 187
  55.1. CREATING PAGES 187
  55.2. SAVING, DELETING, RENAMING, OR COPYING PAGES 187
  55.3. NAVIGATION TREE
     55.3.1. Creating a navigation tree 188
     55.3.2. Editing a navigation tree 188
     55.3.3. Adding groups, dividers, and page entries to a navigation tree 189
     55.3.4. Reordering a navigation tree 189
Reordering navigation trees
Reordering the entries of a navigation tree
55.3.5. Renaming a navigation tree
55.3.6. Deleting a navigation tree
55.3.7. Deleting the entries of a navigation tree
55.4. COMPONENTS
55.4.1. Core components
55.4.2. Navigation components
55.4.3. Reporting components
55.4.4. Component properties
55.4.5. Placing components on the page editor to create a page
55.4.6. Using the Preview tab to preview pages
55.4.7. Adding a time series chart component on a page
55.5. HEATMAP COMPONENTS
55.5.1. Creating heatmap components for a process
55.5.2. Creating heatmap components for multiple processes
55.5.3. Executing a heatmap component
55.6. EXTERNAL COMPONENTS
55.6.1. Creating external components

CHAPTER 56. SECURITY MANAGEMENT ................................................................. 204
56.1. SECURITY MANAGEMENT PROVIDERS ................................................. 204
56.1.1. Configuring the Red Hat JBoss EAP security management provider based on property files 204
56.1.2. Configuring the Red Hat JBoss EAP security management provider based on property files and CLI mode 205
56.2. PERMISSIONS AND SETTINGS ................................................................. 206
56.2.1. Changing permissions for groups and roles in Business Central 207
56.2.2. Changing the Business Central home page 208
56.2.3. Setting priorities 208

CHAPTER 57. EXPORTING, IMPORTING, AND DEPLOYING DASHBOARDS ................. 209
57.1. EXPORTING BUSINESS CENTRAL DASHBOARD DATA ......................... 209
57.2. IMPORTING BUSINESS CENTRAL DASHBOARD DATA ........................ 210
57.3. DEPLOYING DASHBOARDS FROM BUSINESS CENTRAL ON DASHBUILDER RUNTIME 211

APPENDIX A. VERSIONING INFORMATION ...................................................... 212

APPENDIX B. CONTACT INFORMATION ......................................................... 213
PREFACE

As a developer or system administrator, you can modify Red Hat Process Automation Manager and KIE Server settings and properties to meet your business needs. You can modify the behavior of the Red Hat Process Automation Manager runtime, the Business Central interface, or the KIE Server.
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PART I. MANAGING AND MONITORING KIE SERVER
As a system administrator, you can install, configure, and upgrade Red Hat Process Automation Manager for production environments, quickly and easily troubleshoot system failures, and ensure that systems are running optimally.

Prerequisites

- Red Hat JBoss Enterprise Application Platform 7.4 is installed. For more information, see Red Hat JBoss Enterprise Application Platform 7.4 Installation Guide.

- Red Hat Process Automation Manager is installed. For more information, see Planning a Red Hat Process Automation Manager installation.

- Red Hat Process Automation Manager is running and you can log in to Business Central with the admin role. For more information, see Planning a Red Hat Process Automation Manager installation.

NOTE
For information about managing and monitoring business processes that run on KIE Server, see Managing and monitoring business processes in Business Central.
CHAPTER 1. RED HAT PROCESS AUTOMATION MANAGER COMPONENTS

Red Hat Process Automation Manager is made up of Business Central and KIE Server.

- Business Central is the graphical user interface where you create and manage business rules. You can install Business Central in a Red Hat JBoss EAP instance or on the Red Hat OpenShift Container Platform (OpenShift).
  Business Central is also available as a standalone JAR file. You can use the Business Central standalone JAR file to run Business Central without deploying it to an application server.

- KIE Server is the server where rules and other artifacts are executed. It is used to instantiate and execute rules and solve planning problems. You can install KIE Server in a Red Hat JBoss EAP instance, in a Red Hat JBoss EAP cluster, on OpenShift, in an Oracle WebLogic server instance, in an IBM WebSphere Application Server instance, or as a part of Spring Boot application.
  You can configure KIE Server to run in managed or unmanaged mode. If KIE Server is unmanaged, you must manually create and maintain KIE containers (deployment units). A KIE container is a specific version of a project. If KIE Server is managed, the Process Automation Manager controller manages the KIE Server configuration and you interact with the Process Automation Manager controller to create and maintain KIE containers.
CHAPTER 2. SYSTEM INTEGRATION WITH MAVEN

Red Hat Process Automation Manager is designed to be used with Red Hat JBoss Middleware Maven Repository and Maven Central repository as dependency sources. Ensure that both the dependencies are available for projects builds.

Ensure that your project depends on specific versions of an artifact. LATEST or RELEASE are commonly used to specify and manage dependency versions in your application.

- LATEST refers to the latest deployed (snapshot) version of an artifact.
- RELEASE refers to the last non-snapshot version release in the repository.

By using LATEST or RELEASE, you do not have to update version numbers when a new release of a third-party library is released, however, you lose control over your build being affected by a software release.

2.1. PREEMPTIVE AUTHENTICATION FOR LOCAL PROJECTS

If your environment does not have access to the internet, set up an in-house Nexus and use it instead of Maven Central or other public repositories. To import JARs from the remote Maven repository of Red Hat Process Automation Manager server to a local Maven project, turn on pre-emptive authentication for the repository server. You can do this by configuring authentication for guvnor-m2-repo in the pom.xml file as shown below:

```xml
<server>
  <id>guvnor-m2-repo</id>
  <username>admin</username>
  <password>admin</password>
  <configuration>
    <wagonProvider>httpclient</wagonProvider>
    <httpConfiguration>
      <all>
        <usePreemptive>true</usePreemptive>
      </all>
    </httpConfiguration>
  </configuration>
</server>
```

Alternatively, you can set Authorization HTTP header with Base64 encoded credentials:

```xml
<server>
  <id>guvnor-m2-repo</id>
  <configuration>
    <httpHeaders>
      <property>
        <name>Authorization</name>
        <!-- Base64-encoded “admin:admin” -->
        <value>Basic YWRtaW46YWRtaW4=</value>
      </property>
    </httpHeaders>
  </configuration>
</server>
```
2.2. DUPLICATE GAV DETECTION IN BUSINESS CENTRAL

In Business Central, all Maven repositories are checked for any duplicated GroupId, ArtifactId, and Version (GAV) values in a project. If a GAV duplicate exists, the performed operation is canceled.

NOTE
Duplicate GAV detection is disabled for projects in Development Mode. To enable duplicate GAV detection in Business Central, go to project Settings → General Settings → Version and toggle the Development Mode option to OFF (if applicable).

Duplicate GAV detection is executed every time you perform the following operations:

- Save a project definition for the project.
- Save the pom.xml file.
- Install, build, or deploy a project.

The following Maven repositories are checked for duplicate GAVs:

- Repositories specified in the <repositories> and <distributionManagement> elements of the pom.xml file.
- Repositories specified in the Maven settings.xml configuration file.

2.3. MANAGING DUPLICATE GAV DETECTION SETTINGS IN BUSINESS CENTRAL

Business Central users with the admin role can modify the list of repositories that are checked for duplicate GroupId, ArtifactId, and Version (GAV) values for a project.

NOTE
Duplicate GAV detection is disabled for projects in Development Mode. To enable duplicate GAV detection in Business Central, go to project Settings → General Settings → Version and toggle the Development Mode option to OFF (if applicable).

Procedure

1. In Business Central, go to Menu → Design → Projects and click the project name.

2. Click the project Settings tab and then click Validation to open the list of repositories.

3. Select or clear any of the listed repository options to enable or disable duplicate GAV detection. In the future, duplicate GAVs will be reported for only the repositories you have enabled for validation.
NOTE

To disable this feature, set the `org.guvnor.project.gav.check.disabled` system property to `true` for Business Central at system startup:

```
$ ~/EAP_HOME/bin/standalone.sh -c standalone-full.xml
-Dorg.guvnor.project.gav.check.disabled=true
```
CHAPTER 3. APPLYING PATCH UPDATES AND MINOR RELEASE UPGRADES TO RED HAT PROCESS AUTOMATION MANAGER

Automated update tools are often provided with both patch updates and new minor versions of Red Hat Process Automation Manager to facilitate updating certain components of Red Hat Process Automation Manager, such as Business Central, KIE Server, and the headless Process Automation Manager controller. Other Red Hat Process Automation Manager artifacts, such as the decision engine and standalone Business Central, are released as new artifacts with each minor release and you must reinstall them to apply the update.

You can use the same automated update tool to apply both patch updates and minor release upgrades to Red Hat Process Automation Manager 7.12. Patch updates of Red Hat Process Automation Manager, such as an update from version 7.12 to 7.12.1, include the latest security updates and bug fixes. Minor release upgrades of Red Hat Process Automation Manager, such as an upgrade from version 7.10.x to 7.11, include enhancements, security updates, and bug fixes.

**NOTE**

Only updates for Red Hat Process Automation Manager are included in Red Hat Process Automation Manager update tools. Updates to Red Hat JBoss EAP must be applied using Red Hat JBoss EAP patch distributions. For more information about Red Hat JBoss EAP patching, see the [Red Hat JBoss EAP patching and upgrading guide](#).

**Prerequisites**

- Your Red Hat Process Automation Manager and KIE Server instances are not running. Do not apply updates while you are running an instance of Red Hat Process Automation Manager or KIE Server.

**Procedure**

1. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options.
   If you are upgrading to a new minor release of Red Hat Process Automation Manager, such as an upgrade from version 7.10.x to 7.11, first apply the latest patch update to your current version of Red Hat Process Automation Manager and then follow this procedure again to upgrade to the new minor release.

2. Click Patches, download the Red Hat Process Automation Manager [VERSION] Update Tool and extract the downloaded rhpam-$VERSION-update.zip file to a temporary directory.
   This update tool automates the update of certain components of Red Hat Process Automation Manager, such as Business Central, KIE Server, and the headless Process Automation Manager controller. Use this update tool first to apply updates and then install any other updates or new release artifacts that are relevant to your Red Hat Process Automation Manager distribution.

3. If you want to preserve any files from being updated by the update tool, navigate to the extracted rhpam-$VERSION-update folder, open the blacklist.txt file, and add the relative paths to the files that you do not want to be updated.
   When a file is listed in the blacklist.txt file, the update script does not replace the file with the new version but instead leaves the file in place and in the same location adds the new version with a .new suffix. If you block files that are no longer being distributed, the update tool creates an empty marker file with a .removed suffix. You can then choose to retain, merge, or delete these new files manually.
Example files to be excluded in blacklist.txt file:

- WEB-INF/web.xml // Custom file
- styles/base.css // Obsolete custom file kept for record

The contents of the blocked file directories after the update:

- `ls WEB-INF`
  - `web.xml`
  - `web.xml.new`

- `ls styles`
  - `base.css`
  - `base.css.removed`

4. In your command terminal, navigate to the temporary directory where you extracted the rhpam-$VERSION-update.zip file and run the apply-updates script in the following format:

```
$ ls WEB-INF
web.xml web.xml.new

$ ls styles
base.css base.css.removed
```

**IMPORTANT**

Make sure that your Red Hat Process Automation Manager and KIE Server instances are not running before you apply updates. Do not apply updates while you are running an instance of Red Hat Process Automation Manager or KIE Server.

On Linux or Unix-based systems:

```
$ ./apply-updates.sh $DISTRO_PATH $DISTRO_TYPE
```

On Windows:

```
$ .\apply-updates.bat $DISTRO_PATH $DISTRO_TYPE
```

The $DISTRO_PATH portion is the path to the relevant distribution directory and the $DISTRO_TYPE portion is the type of distribution that you are updating with this update.

The following distribution types are supported in Red Hat Process Automation Manager update tool:

- **rhpam-business-central-eap7-deployable**: Updates Business Central (*business-central.war*)
- **rhpam-kie-server-ee8**: Updates KIE Server (*kie-server.war*)

**NOTE**

The update tool will update and replace Red Hat JBoss EAP EE7 to Red Hat JBoss EAP EE8. Red Hat JBoss EAP EE7 is used for WebLogic and WebSphere, whereas version EE8 is used for Red Hat JBoss EAP. Make sure that KIE Server on WebLogic and WebSphere is not updated by the update tool.

- **rhpam-kie-server-jws**: Updates KIE Server on Red Hat JBoss Web Server (*kie-server.war*)
**rhpam-controller-ee7**: Updates the headless Process Automation Manager controller (controller.war)

**rhpam-controller-jws**: Updates the headless Process Automation Manager controller on Red Hat JBoss Web Server (controller.war)

Example update to Business Central and KIE Server for a full Red Hat Process Automation Manager distribution on Red Hat JBoss EAP:

```bash
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/business-central.war rhpam-business-central-eap7-deployable
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/kie-server.war rhpam-kie-server-ee8
```

Example update to headless Process Automation Manager controller, if used:

```bash
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/controller.war rhpam-controller-ee7
```

The update script creates a **backup** folder in the extracted `rhpam-$VERSION-update` folder with a copy of the specified distribution, and then proceeds with the update.

5. After the update tool completes, return to the **Software Downloads** page of the Red Hat Customer Portal where you downloaded the update tool and install any other updates or new release artifacts that are relevant to your Red Hat Process Automation Manager distribution. For files that already exist in your Red Hat Process Automation Manager distribution, such as `.jar` files for the decision engine or other add-ons, replace the existing version of the file with the new version from the Red Hat Customer Portal.

6. If you use the standalone **Red Hat Process Automation Manager 7.12.0 Maven Repository** artifact (`rhpam-7.12.0-maven-repository.zip`), such as in air-gap environments, download **Red Hat Process Automation Manager 7.12.x Maven Repository** and extract the downloaded `rhpam-7.12.x-maven-repository.zip` file to your existing `~/.maven-repository` directory to update the relevant contents.

Example Maven repository update:

```bash
$ unzip -o rhpam-7.12.x-maven-repository.zip 'rhba-7.12.1.GA-maven-repository/maven-repository/*' -d /tmp/rhbaMavenRepoUpdate
$ mv /tmp/rhbaMavenRepoUpdate/rhba-7.12.0.GA-maven-repository/maven-repository/$REPO_PATH/
```

**NOTE**

You can remove the `/tmp/rhbaMavenRepoUpdate` folder after you complete the update.

7. Optional: If you are changing Red Hat Process Automation Manager from using property-based user storage to file-based user storage, complete the following steps:

   a. Navigate to the `$JBOSS_HOME` directory and run one of the following commands:

      * On Linux or Unix-based systems:
$ ./bin/standalone.sh --admin-only -c standalone-full.xml

$ ./bin/jboss-cli.sh --connect --file=rhpam-$VERSION-update/elytron/add-kie-fs-realm.cli

• On Windows:

$ ./bin/standalone.bat --admin-only -c standalone-full.xml

$ ./bin/jboss-cli.bat --connect --file=rhpam-$VERSION-update/elytron/add-kie-fs-realm.cli

b. Run the following command:

• On Linux or Unix-based systems:

$ ./bin/elytron-tool.sh filesystem-realm --users-file standalone/configuration/application-users.properties --roles-file standalone/configuration/application-roles.properties --output-location standalone/configuration/kie-fs-realm-users --filesystem-realm-name kie-fs-realm-users

• On Windows:

$ ./bin/elytron-tool.bat filesystem-realm --users-file standalone/configuration/application-users.properties --roles-file standalone/configuration/application-roles.properties --output-location standalone/configuration/kie-fs-realm-users --filesystem-realm-name kie-fs-realm-users

c. Navigate to the directory where you extracted the rhpam-$VERSION-update.zip file and run one of the following commands to apply the kie-fs-realm patch:

• On Linux or Unix-based systems:

$ ./elytron/kie-fs-realm-patch.sh ~/$JBOSS_HOME/standalone/configuration/kie-fs-realm-users/

• On Windows:

$ ./elytron/kie-fs-realm-patch.bat ~/$JBOSS_HOME/standalone/configuration/kie-fs-realm-users/

8. After you finish applying all relevant updates, start Red Hat Process Automation Manager and KIE Server and log in to Business Central.

9. Verify that all project data is present and accurate in Business Central, and in the top-right corner of the Business Central window, click your profile name and click About to verify the updated product version number.

   If you encounter errors or notice any missing data in Business Central, you can restore the contents in the backup folder within the rhpam-$VERSION-update folder to revert the update tool changes. You can also reinstall the relevant release artifacts from your previous version of
Red Hat Process Automation Manager in the Red Hat Customer Portal. After restoring your previous distribution, you can try again to run the update.
CHAPTER 4. CONFIGURING AND STARTING KIE SERVER

You can configure your KIE Server location, user name, password, and other related properties by defining the necessary configurations when you start KIE Server.

Procedure

Navigate to the Red Hat Process Automation Manager 7.12 bin directory and start the new KIE Server with the following properties. Adjust the specific properties according to your environment.

```
$ ~/EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml
- Dorg.kie.server.id=myserver
- Dorg.kie.server.user=kie_server_username
- Dorg.kie.server.pwd=kie_server_password
- Dorg.kie.server.controller.user=controller_username
- Dorg.kie.server.controller.pwd=controller_password
- Dorg.kie.server.location=http://localhost:8080/kie-server/services/rest/server
- Dorg.kie.server.persistence.dialect=org.hibernate.dialect.PostgreSQLDialect
- Dorg.kie.server.persistence.ds=java:jboss/datasources/psjbpmDS
```

1. Start command with standalone-full.xml server profile
2. Server ID that must match the server configuration name defined in Business Central
3. User name to connect with KIE Server from the Process Automation Manager controller
4. Password to connect with KIE Server from the Process Automation Manager controller
5. Process Automation Manager controller location, Business Central URL with /rest/controller suffix
6. User name to connect to the Process Automation Manager controller REST API
7. Password to connect to the Process Automation Manager controller REST API
8. KIE Server location (on the same instance as Business Central in this example)
9. Hibernate dialect to be used
10. JNDI name of the data source used for your previous Red Hat JBoss BPM Suite database
NOTE

If Business Central and KIE Server are installed on separate application server instances (Red Hat JBoss EAP or other), use a separate port for the KIE Server location to avoid port conflicts with Business Central. If a separate KIE Server port has not already been configured, you can add a port offset and adjust the KIE Server port value accordingly in the KIE Server properties.

Example:

-Dboss.socket.binding.port-offset=150
-Dorg.kie.server.location=http://localhost:8230/kie-server/services/rest/server

If the Business Central port is 8080, as in this example, then the KIE Server port, with a defined offset of 150, is 8230.

KIE Server connects to the new Business Central and collects the list of deployment units (KIE containers) to be deployed.

NOTE

When you use a class inside a dependency JAR file to access KIE Server from KIE Server client, you get the **ConversionException** and **ForbiddenClassException** in Business Central. To avoid generating these exceptions in Business Central, do one of the following:

- If the exceptions are generated on the client-side, add following system property to the kie-server client:

  ```
  System.setProperty("org.kie.server.xstream.enabled.packages", "org.example.");
  ```

- If the exceptions are generated on the server-side, open **standalone-full.xml** from the Red Hat Process Automation Manager installation directory, set the following property under the `<system-properties>` tag:

  ```
  <property name="org.kie.server.xstream.enabled.packages" value="org.example.">
  ```

- Set the following JVM property:

  ```
  -Dorg.kie.server.xstream.enabled.packages=org.example.
  ```

It is expected that you do not configure the classes that exists in KJAR using these system property. Ensure that only known classes are used in the system property to avoid any vulnerabilities.

The **org.example** is an example package, you can define any package that you want to use. You can specify multiple packages separated by comma, for example, **org.example1.*, org.example2.*, org.example3.***.

You can also add specific classes, for example, **org.example1.Mydata1, org.example2.Mydata2**.
A data source is an object that enables a Java Database Connectivity (JDBC) client, such as an application server, to establish a connection with a database. Applications look up the data source on the Java Naming and Directory Interface (JNDI) tree or in the local application context and request a database connection to retrieve data. You must configure data sources for KIE Server to ensure correct data exchange between the servers and the designated database.

Typically, solutions using Red Hat Process Automation Manager manage several resources within a single transaction. JMS for asynchronous jobs, events, and timers, for example. Red Hat Process Automation Manager requires an XA driver in the datasource when possible to ensure data atomicity and consistent results. If transactional code for different schemas exists inside listeners or derives from hooks provided by the jBPM engine, an XA driver is also required.

Do not use non-XA datasources unless you are positive you do not have multiple resources participating in single transactions.

**NOTE**

For production environments, specify an actual data source. Do not use the example data source in production environments.

**Prerequisites**

- The JDBC providers that you want to use to create database connections are configured on all servers on which you want to deploy KIE Server, as described in the "Creating Datasources" and "JDBC Drivers" sections of the *Red Hat JBoss Enterprise Application Server Configuration Guide*.


**Procedure**

1. Complete the following steps to prepare your database:
   a. Extract *rhpam-7.12.0-add-ons.zip* in a temporary directory, for example *TEMP_DIR*.
   c. Change your current directory to the *TEMP_DIR/rhpam-7.12.0-migration-tool/ddl-scripts* directory. This directory contains DDL scripts for several database types.
   d. Import the DDL script for your database type into the database that you want to use.

   The following example creates jBPM database structures in PostgreSQL:

   ```
   psql jbpm < /ddl-scripts/postgresql/postgresql-jbpm-schema.sql
   ```
NOTE
If you are using PostgreSQL or Oracle in conjunction with Spring Boot, you must import the respective Spring Boot DDL script, for example /ddl-scripts/oracle/oracle-springboot-jbpm-schema.sql or /ddl-scripts/postgresql/postgresql-springboot-jbpm-schema.sql.

NOTE
The PostgreSQL DDL scripts create the PostgreSQL schema with auto-incrementing integer value (OID) columns for entity attributes annotated with @LOB. To use other binary column types such as BYTEA instead of OID, you must create the PostgreSQL schema with the postgresql-bytea-jbpm-schema.sql script and set the Red Hat Process Automation Manager org.kie.persistence.postgresql.useBytea=true flag. Do not use the postgresql-jbpm-lo-trigger-clob.sql script when creating a BYTEA-based schema. Red Hat Process Automation Manager does not provide a migration tool to change from an OID-based to a BYTEA-based schema.

2. Open EAP_HOME/standalone/configuration/standalone-full.xml in a text editor and locate the <system-properties> tag.

3. Add the following properties to the <system-properties> tag where <DATASOURCE> is the JNDI name of your data source and <HIBERNATE_DIALECT> is the hibernate dialect for your database.

```xml
<property name="org.kie.server.persistence.ds" value="<DATASOURCE>">
<property name="org.kie.server.persistence.dialect" value="<HIBERNATE_DIALECT>">
```

The default value of the org.kie.server.persistence.ds property is java:jboss/datasources/ExampleDS. The default value of the org.kie.server.persistence.dialect property is org.hibernate.dialect.H2Dialect.

```xml
<system-properties>
  <property name="org.kie.server.repo" value="${jboss.server.data.dir}"/>
  <property name="org.kie.example" value="true"/>
  <property name="org.jbpm.designer.perspective" value="full"/>
  <property name="designerdataobjects" value="false"/>
  <property name="org.kie.server.user" value="rhpamUser"/>
  <property name="org.kie.server.pwd" value="rhpam123!"/>
  <property name="org.kie.server.location" value="http://localhost:8080/kie-server/services/rest/server"/>
  <property name="org.kie.server.controller" value="http://localhost:8080/business-central/rest/controller"/>
  <property name="org.kie.server.controller.user" value="kieserver"/>
  <property name="org.kie.server.controller.pwd" value="kieserver1!"/>
  <property name="org.kie.server.id" value="local-server-123"/>
</system-properties>
```
The following dialects are supported:

- **DB2**: `org.hibernate.dialect.DB2Dialect`
- **MSSQL**: `org.hibernate.dialect.SQLServer2012Dialect`
- **MySQL**: `org.hibernate.dialect.MySQL5InnoDBDialect`
- **MariaDB**: `org.hibernate.dialect.MySQL5InnoDBDialect`
- **Oracle**: `org.hibernate.dialect.Oracle10gDialect`
- **PostgreSQL**: `org.hibernate.dialect.PostgreSQL82Dialect`
- **PostgreSQL plus**: `org.hibernate.dialect.PostgresPlusDialect`
- **Sybase**: `org.hibernate.dialect.SybaseASE157Dialect`
CHAPTER 6. MANAGED KIE SERVER

A managed instance requires an available Process Automation Manager controller to start KIE Server.

A Process Automation Manager controller manages KIE Server configuration in a centralized way. Each Process Automation Manager controller can manage multiple configurations at once, and there can be multiple Process Automation Manager controllers in the environment. Managed KIE Server can be configured with a list of Process Automation Manager controllers, but will only connect to one at a time.

IMPORTANT

All Process Automation Manager controllers should be synchronized to ensure that the same set of configuration is provided to the server, regardless of the Process Automation Manager controller to which it connects.

When KIE Server is configured with a list of Process Automation Manager controllers, it will attempt to connect to each of them at startup until a connection is successfully established with one of them. If a connection cannot be established, the server will not start, even if there is a local storage available with configuration. This ensures consistency and prevents the server from running with redundant configuration.

NOTE

To run KIE Server in standalone mode without connecting to Process Automation Manager controllers, see Chapter 7, Unmanaged KIE Server.
An unmanaged KIE Server is a standalone instance, and therefore must be configured individually using REST/JMS API from KIE Server itself. The configuration is automatically persisted by the server into a file and that is used as the internal server state, in case of restarts.

The configuration is updated during the following operations:

- Deploy KIE container
- Undeploy KIE container
- Start KIE container
- Stop KIE container

**NOTE**

If KIE Server is restarted, it will attempt to re-establish the same state that was persisted before shutdown. Therefore, KIE containers (deployment units) that were running will be started, but the ones that were stopped will not.
CHAPTER 8. CONFIGURING THE ENVIRONMENT MODE IN KIE SERVER AND BUSINESS CENTRAL

You can set KIE Server to run in production mode or in development mode. Development mode provides a flexible deployment policy that enables you to update existing deployment units (KIE containers) while maintaining active process instances for small changes. It also enables you to reset the deployment unit state before updating active process instances for larger changes. Production mode is optimal for production environments, where each deployment creates a new deployment unit.

In a development environment, you can click Deploy in Business Central to deploy the built KJAR file to a KIE Server without stopping any running instances (if applicable), or click Redeploy to deploy the built KJAR file and replace all instances. The next time you deploy or redeploy the built KJAR, the previous deployment unit (KIE container) is automatically updated in the same target KIE Server.

In a production environment, the Redeploy option in Business Central is disabled and you can click only Deploy to deploy the built KJAR file to a new deployment unit (KIE container) on a KIE Server.

Procedure

1. To configure the KIE Server environment mode, set the org.kie.server.mode system property to org.kie.server.mode=development or org.kie.server.mode=production.

2. To configure the deployment behavior for a project in Business Central, go to project Settings → General Settings → Version and toggle the Development Mode option.

NOTE

By default, KIE Server and all new projects in Business Central are in development mode.

You cannot deploy a project with Development Mode turned on or with a manually added SNAPSHOT version suffix to a KIE Server that is in production mode.
CHAPTER 9. CONFIGURING KIE SERVER TO CONNECT TO BUSINESS CENTRAL

WARNING
This section provides a sample setup that you can use for testing purposes. Some of the values are unsuitable for a production environment, and are marked as such.

If a KIE Server is not configured in your Red Hat Process Automation Manager environment, or if you require additional KIE Servers in your Red Hat Process Automation Manager environment, you must configure a KIE Server to connect to Business Central.

NOTE
If you are deploying KIE Server on Red Hat OpenShift Container Platform, see the Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform 4 using Operators document for instructions about configuring it to connect to Business Central.

KIE Server can be managed or unmanaged. If KIE Server is unmanaged, you must manually create and maintain KIE containers (deployment units). If KIE Server is managed, the Process Automation Manager controller manages the KIE Server configuration and you interact with the Process Automation Manager controller to create and maintain the KIE containers.

NOTE
Make the changes described in this section if KIE Server is managed by Business Central and you have installed Red Hat Process Automation Manager from the ZIP files. If you have installed Business Central, you can use the headless Process Automation Manager controller to manage KIE Server, as described in Chapter 10, Installing and running the headless Process Automation Manager controller.

Prerequisites

- Business Central and KIE Server are installed in the base directory of the Red Hat JBoss EAP installation (EAP_HOME).

NOTE
You must install Business Central and KIE Server on different servers in production environments. In this sample situation, we use only one user named controllerUser, containing both rest-all and the kie-server roles. However, if you install KIE Server and Business Central on the same server, for example in a development environment, make the changes in the shared standalone-full.xml file as described in this section.

- Users with the following roles exist:
  - In Business Central, a user with the role rest-all
On KIE Server, a user with the role kie-server

Procedure

1. In your Red Hat Process Automation Manager installation directory, navigate to the standalone-full.xml file. For example, if you use a Red Hat JBoss EAP installation for Red Hat Process Automation Manager, go to $EAP_HOME/standalone/configuration/standalone-full.xml.

2. Open the standalone-full.xml file and under the <system-properties> tag, set the following JVM properties:

Table 9.1. JVM Properties for the KIE Server instance

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.kie.server.id</td>
<td>default-kie-server</td>
<td>The KIE Server ID.</td>
</tr>
<tr>
<td>org.kie.server.controller.user</td>
<td>controllerUser</td>
<td>The user name with the role rest-all who can log in to the Business Central.</td>
</tr>
<tr>
<td>org.kie.server.controller.pwd</td>
<td>controllerUser1234;</td>
<td>The password of the user who can log in to the Business Central.</td>
</tr>
</tbody>
</table>

Table 9.2. JVM Properties for the Business Central instance

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.kie.server.user</td>
<td>controllerUser</td>
<td>The user name with the role kie-server.</td>
</tr>
<tr>
<td>org.kie.server.pwd</td>
<td>controllerUser1234;</td>
<td>The password of the user.</td>
</tr>
</tbody>
</table>

The following example shows how to configure a KIE Server instance:

```xml
<property name="org.kie.server.id" value="default-kie-server"/>
<property name="org.kie.server.controller" value="http://localhost:8080/business-central/rest/controller"/>
<property name="org.kie.server.controller.user" value="controllerUser"/>
```
The following example shows how to configure KIE Server for Business Central instance:

```
<property name="org.kie.server.controller.pwd" value="controllerUser1234;"/>
<property name="org.kie.server.location" value="http://localhost:8080/kie-server/services/rest/server"/>
```

3. To verify that KIE Server starts successfully, send a GET request to `http://SERVER:PORT/kie-server/services/rest/server/` when KIE Server is running. For more information about running Red Hat Process Automation Manager on KIE Server, see Running Red Hat Process Automation Manager.

After successful authentication, you receive an XML response similar to the following example:

```
<response type="SUCCESS" msg="Kie Server info">
  <kie-server-info>
    <capabilities>Capabilities</capabilities>
    <capabilities>BRM</capabilities>
    <capabilities>BPM</capabilities>
    <capabilities>CaseMgmt</capabilities>
    <capabilities>BPM-UI</capabilities>
    <capabilities>BRP</capabilities>
    <capabilities>DMN</capabilities>
    <capabilities>Swagger</capabilities>
    <location>http://localhost:8230/kie-server/services/rest/server</location>
    <messages>
      <content>Server KieServerInfo{serverId='first-kie-server', version='7.5.1.Final-redhat-1', location='http://localhost:8230/kie-server/services/rest/server', capabilities=[KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP, DMN, Swagger]} started successfully at Mon Feb 05 15:44:35 AEST 2018</content>
      <severity>INFO</severity>
      <timestamp>2018-02-05T15:44:35.355+10:00</timestamp>
    </messages>
    <name>first-kie-server</name>
    <id>first-kie-server</id>
    <version>7.5.1.Final-redhat-1</version>
  </kie-server-info>
</response>
```

4. Verify successful registration:

   a. Log in to Business Central.

   b. Click **Menu → Deploy → Execution Servers**.
      If registration is successful, you will see the registered server ID.
CHAPTER 10. INSTALLING AND RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

You can configure KIE Server to run in managed or unmanaged mode. If KIE Server is unmanaged, you must manually create and maintain KIE containers (deployment units). If KIE Server is managed, the Process Automation Manager controller manages the KIE Server configuration and you interact with the Process Automation Manager controller to create and maintain KIE containers.

Business Central has an embedded Process Automation Manager controller. If you install Business Central, use the Execution Server page to create and maintain KIE containers. If you want to automate KIE Server management without Business Central, you can use the headless Process Automation Manager controller.

10.1. USING THE INSTALLER TO CONFIGURE KIE SERVER WITH THE PROCESS AUTOMATION MANAGER CONTROLLER

KIE Server can be managed by the Process Automation Manager controller or it can be unmanaged. If KIE Server is unmanaged, you must manually create and maintain KIE containers (deployment units). If KIE Server is managed, the Process Automation Manager controller manages the KIE Server configuration and you interact with the Process Automation Manager controller to create and maintain KIE containers.

The Process Automation Manager controller is integrated with Business Central. If you install Business Central, you can use the Execution Server page in Business Central to interact with the Process Automation Manager controller.

You can use the installer in interactive or CLI mode to install Business Central and KIE Server, and then configure KIE Server with the Process Automation Manager controller.

Prerequisites

- Two computers with backed-up Red Hat JBoss EAP 7.4 server installations are available.
- Sufficient user permissions to complete the installation are granted.

Procedure

1. On the first computer, run the installer in interactive mode or CLI mode. See Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.4 for more information.

2. On the Component Selection page, clear the KIE Server box.

3. Complete the Business Central installation.

4. On the second computer, run the installer in interactive mode or CLI mode.

5. On the Component Selection page, clear the Business Central box.


7. Select Customize KIE Server properties and click Next.
8. Enter the controller URL for Business Central and configure additional properties for KIE Server. The controller URL has the following form where `<HOST:PORT>` is the address of Business Central on the second computer:

```
<HOST:PORT>/business-central/rest/controller
```

9. Complete the installation.

10. To verify that the Process Automation Manager controller is now integrated with Business Central, go to the Execution Servers page in Business Central and confirm that the KIE Server that you configured appears under REMOTE SERVERS.

## 10.2. INSTALLING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

You can install the headless Process Automation Manager controller and use the REST API or the KIE Server Java Client API to interact with it.

### Prerequisites

- A backed-up Red Hat JBoss EAP installation version 7.4 is available. The base directory of the Red Hat JBoss EAP installation is referred to as `EAP_HOME`.

- Sufficient user permissions to complete the installation are granted.

### Procedure

1. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
   - **Product**: Process Automation Manager
   - **Version**: 7.12


4. Extract the `rhpam-7.12.0-controller-ee7.zip` archive to a temporary directory. In the following examples this directory is called `TEMP_DIR`.

5. Copy the `TEMP_DIR/rhpam-7.12.0-controller-ee7/controller.war` directory to `EAP_HOME/standalone/deployments/`. 
WARNING

Ensure that the names of the headless Process Automation Manager controller deployments you copy do not conflict with your existing deployments in the Red Hat JBoss EAP instance.


7. When prompted to overwrite files, select Yes.

8. In the `EAP_HOME/standalone/deployments/` directory, create an empty file named `controller.war.dodeploy`. This file ensures that the headless Process Automation Manager controller is automatically deployed when the server starts.

10.2.1. Creating a headless Process Automation Manager controller user

Before you can use the headless Process Automation Manager controller, you must create a user that has the `kie-server` role.

Prerequisites

- The headless Process Automation Manager controller is installed in the base directory of the Red Hat JBoss EAP installation (`EAP_HOME`).

Procedure

1. In a terminal application, navigate to the `EAP_HOME/bin` directory.

2. Enter the following command and replace `<USERNAME>` and `<PASSWORD>` with the user name and password of your choice.

```
$ ./bin/jboss-cli.sh --commands="embed-server --std-out=echo,/subsystem=elytron/filesystem-realm=ApplicationRealm:add-identity(identity=<USERNAME>),/subsystem=elytron/filesystem-realm=ApplicationRealm:set-password(identity=<USERNAME>, clear={password='<PASSWORD>'}),/subsystem=elytron/filesystem-realm=ApplicationRealm:add-identity-attribute(identity=<USERNAME>, name=role, value=['kie-server'])"
```

NOTE

Make sure that the specified user name is not the same as an existing user, role, or group. For example, do not create a user with the user name `admin`.

The password must have at least eight characters and must contain at least one number and one non-alphanumeric character, but not & (ampersand).

3. Make a note of your user name and password.
10.2.2. Configuring KIE Server and the headless Process Automation Manager controller

If KIE Server will be managed by the headless Process Automation Manager controller, you must edit the `standalone-full.xml` file in KIE Server installation and the `standalone.xml` file in the headless Process Automation Manager controller installation.

Prerequisites

- KIE Server is installed in an `EAP_HOME`.
- The headless Process Automation Manager controller is installed in an `EAP_HOME`.

**NOTE**

You should install KIE Server and the headless Process Automation Manager controller on different servers in production environments. However, if you install KIE Server and the headless Process Automation Manager controller on the same server, for example in a development environment, make these changes in the shared `standalone-full.xml` file.

- On KIE Server nodes, a user with the `kie-server` role exists.
- On the server nodes, a user with the `kie-server` role exists.

Procedure

1. In the `EAP_HOME/standalone/configuration/standalone-full.xml` file, add the following properties to the `<system-properties>` section and replace `<USERNAME>` and `<USER_PWD>` with the credentials of a user with the `kie-server` role:

   ```xml
   <property name="org.kie.server.user" value="<USERNAME/>
   <property name="org.kie.server.pwd" value="<USER_PWD>
   ```

2. In the KIE Server `EAP_HOME/standalone/configuration/standalone-full.xml` file, add the following properties to the `<system-properties>` section:

   ```xml
   <property name="org.kie.server.controller.user" value="<CONTROLLER_USER>
   <property name="org.kie.server.controller.pwd" value="<CONTROLLER_PWD>
   <property name="org.kie.server.id" value="<KIE_SERVER_ID>
   <property name="org.kie.server.location" value="http://<HOST>:<PORT>/kie-server/services/rest/server"/
   <property name="org.kie.server.controller" value="<CONTROLLER_URL>
   ```

3. In this file, replace the following values:

   - Replace `<CONTROLLER_USER>` and `<CONTROLLER_PWD>` with the credentials of a user with the `kie-server` role.
   - Replace `<KIE_SERVER_ID>` with the ID or name of the KIE Server installation, for example, `rhpam-7.12.0-kie-server-1`.
   - Replace `<HOST>` with the ID or name of the KIE Server host, for example, `localhost` or `192.7.8.9`. 

• Replace `<PORT>` with the port of the KIE Server host, for example, 8080.

**NOTE**

The `org.kie.server.location` property specifies the location of KIE Server.

• Replace `<CONTROLLER_URL>` with the URL of the headless Process Automation Manager controller. KIE Server connects to this URL during startup.

### 10.3. RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

After you have installed the headless Process Automation Manager controller on Red Hat JBoss EAP, use this procedure to run the headless Process Automation Manager controller.

**Prerequisites**

• The headless Process Automation Manager controller is installed and configured in the base directory of the Red Hat JBoss EAP installation (`EAP_HOME`).

**Procedure**

1. In a terminal application, navigate to `EAP_HOME/bin`.

2. If you installed the headless Process Automation Manager controller on the same Red Hat JBoss EAP instance as the Red Hat JBoss EAP instance where you installed KIE Server, enter one of the following commands:

   • On Linux or UNIX-based systems:
     
     ```
     $ ./standalone.sh -c standalone-full.xml
     ```

   • On Windows:
     
     ```
     standalone.bat -c standalone-full.xml
     ```

3. If you installed the headless Process Automation Manager controller on a separate Red Hat JBoss EAP instance from the Red Hat JBoss EAP instance where you installed KIE Server, start the headless Process Automation Manager controller with the `standalone.sh` script:

   **NOTE**

   In this case, ensure that you made all required configuration changes to the `standalone.xml` file.

   • On Linux or UNIX-based systems:
     
     ```
     $ ./standalone.sh
     ```

   • On Windows:
     
     ```
     standalone.bat
     ```
4. To verify that the headless Process Automation Manager controller is working on Red Hat JBoss EAP, enter the following command where `<CONTROLLER>` and `<CONTROLLER_PWD>` is the user name and password. The output of this command provides information about the KIE Server instance.

```
```

**NOTE**

Alternatively, you can use the KIE Server Java API Client to access the headless Process Automation Manager controller.

## 10.4. CLUSTERING KIE SERVERS WITH THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

The Process Automation Manager controller is integrated with Business Central. However, if you do not install Business Central, you can install the headless Process Automation Manager controller and use the REST API or the KIE Server Java Client API to interact with it.

### Prerequisites

- A backed-up Red Hat JBoss EAP installation version 7.4 or later is available. The base directory of the Red Hat JBoss EAP installation is referred to as `EAP_HOME`.
- Sufficient user permissions to complete the installation are granted.
- An NFS server with a shared folder is available as described in *Installing and configuring Red Hat Process Automation Manager in a Red Hat JBoss EAP clustered environment*.

### Procedure

1. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
   - **Product**: Process Automation Manager
   - **Version**: 7.12


4. Extract the `rhpam-7.12.0-controller-ee7.zip` archive to a temporary directory. In the following examples this directory is called `TEMP_DIR`.

5. Copy the `TEMP_DIR/rhpam-7.12.0-controller-ee7/controller.war` directory to `EAP_HOME/standalone/deployments/`.
WARNING

Ensure that the names of the headless Process Automation Manager controller deployments you copy do not conflict with your existing deployments in the Red Hat JBoss EAP instance.


7. When prompted to overwrite files, click Yes.

8. In the EAP_HOME/standalone/deployments/ directory, create an empty file named controller.war.dodeploy. This file ensures that the headless Process Automation Manager controller is automatically deployed when the server starts.

9. Open the EAP_HOME/standalone/configuration/standalone.xml file in a text editor.

10. Add the following properties to the <system-properties> element and replace <NFS_STORAGE> with the absolute path to the NFS storage where the template configuration is stored:

```
<system-properties>
  <property name="org.kie.server.controller.templatefile.watcher.enabled" value="true"/>
  <property name="org.kie.server.controller.templatefile" value="<NFS_STORAGE>"/>
</system-properties>
```

Template files contain default configurations for specific deployment scenarios.

If the value of the org.kie.server.controller.templatefile.watcher.enabled property is set to true, a separate thread is started to watch for modifications of the template file. The default interval for these checks is 30000 milliseconds and can be further controlled by the org.kie.server.controller.templatefile.watcher.interval system property. If the value of this property is set to false, changes to the template file are detected only when the server restarts.

11. To start the headless Process Automation Manager controller, navigate to EAP_HOME/bin and enter the following command:

   - On Linux or UNIX-based systems:
     ```
     $ ./standalone.sh
     ```
   - On Windows:
     ```
     standalone.bat
     ```

For more information about running Red Hat Process Automation Manager in a Red Hat JBoss Enterprise Application Platform clustered environment, see Installing and configuring Red Hat Process Automation Manager in a Red Hat JBoss EAP clustered environment.
CHAPTER 11. CONFIGURING SMART ROUTER FOR TLS SUPPORT

You can configure Smart Router (KIE Server Router) for Transport Layer Security (TLS) support to allow HTTPS traffic. In addition, you can disable unsecure HTTP connections to Smart Router.

Prerequisites

- KIE Server is installed on each node of a Red Hat JBoss EAP 7.4 cluster.
- Smart Router is installed and configured. For more information, see Installing and configuring Red Hat Process Automation Manager in a Red Hat JBoss EAP clustered environment.

Procedure

To start Smart Router, use one of the following methods:

- To start Smart Router with TLS support and HTTPS enabled as well as allowing HTTP connections, enter the following command:

  ```java
  java  -Dorg.kie.server.router.tls.keystore = <KEYSTORE_PATH>
        -Dorg.kie.server.router.tls.keystore.password = <KEYSTORE_PASSWORD>
        -Dorg.kie.server.router.tls.keystore.keyalias = <KEYSTORE_ALIAS>
        -Dorg.kie.server.router.tls.port = <HTTPS_PORT>
        -jar rhpam-7.12.0-smart-router.jar
  ```

  In this example, replace the following variables:

  - `<KEYSTORE_PATH>`: The path where the keystore will be stored.
  - `<KEYSTORE_PASSWORD>`: The keystore password.
  - `<KEYSTORE_ALIAS>`: The alias name used to store the certificate.
  - `<HTTPS_PORT>`: The HTTPS port. The default HTTPS port is 9443.

- To start Smart Router with TLS support and HTTPS enabled and with HTTP connections disabled, enter the following command:

  ```java
  java  -Dorg.kie.server.router.tls.keystore = <KEYSTORE_PATH>
        -Dorg.kie.server.router.tls.keystore.password = <KEYSTORE_PASSWORD>
        -Dorg.kie.server.router.tls.keystore.keyalias = <KEYSTORE_ALIAS>
        -Dorg.kie.server.router.tls.port = <HTTPS_PORT>
        -Dorg.kie.server.router.port=0
        -jar rhpam-7.12.0-smart-router.jar
  ```

  When the `org.kie.server.router.port` system property is set to 0, then the HTTP listener is not registered. If TLS is configured and the HTTP listener is not registered, then Smart Router listens only on the HTTPS port.

  **NOTE**

  If TLS is not configured and you disable HTTP by setting `org.kie.server.router.port` to 0, then an error occurs and Smart Router stops.
CHAPTER 12. ACTIVATING OR DEACTIVATING A KIE CONTAINER ON KIE SERVER

You can now stop the creation of new process instances from a given container by deactivating it but at the same time continue working on its existing process instances and tasks. In case the deactivation is temporary, you can activate the container again later. The activation or deactivation of KIE containers do not require restarting of KIE server.

Prerequisites

- A KIE container has been created and configured in Business Central.

Procedure

1. Log in to Business Central.
2. In the main menu, click **Menu → Deploy → Execution Servers**.
3. From the **Server Configurations** pane, which is on the left of the page, select your server.
4. From the **Deployment Units** pane, select the deployment unit you want to activate or deactivate.
5. Click **Activate** or **Deactivate** in the upper-right corner of the deployment unit pane. You cannot create a process instance from a KIE container once it is deactivated.
Processes and rules are stored in Apache Maven based packaging and are known as knowledge archives, or KJAR. The rules, processes, assets, and other project artifacts are part of a JAR file built and managed by Maven. A file kept inside the \texttt{META-INF} directory of the KJAR called \texttt{kmodule.xml} can be used to define the KIE bases and sessions. This \texttt{kmodule.xml} file, by default, is empty.

Whenever a runtime component such as KIE Server is about to process the KJAR, it looks up \texttt{kmodule.xml} to build the runtime representation.

Deployment descriptors supplement the \texttt{kmodule.xml} file and provide granular control over your deployment. The presence of these descriptors is optional and your deployment will proceed successfully without them. You can set purely technical properties using these descriptors, including meta values such as persistence, auditing, and runtime strategy.

These descriptors allow you to configure KIE Server on multiple levels, including server level default, different deployment descriptor per KJAR, and other server configurations. You can use descriptors to make simple customizations to the default KIE Server configuration, possibly per KJAR.

You can define these descriptors in a file called \texttt{kie-deployment-descriptor.xml} and place this file next to your \texttt{kmodule.xml} file in the \texttt{META-INF} folder. You can change this default location and the file name by specifying it as a system parameter:

\begin{verbatim}
-Dorg.kie.deployment.desc.location=file:/path/to/file/company-deployment-descriptor.xml
\end{verbatim}

\section*{13.1. DEPLOYMENT DESCRIPTOR CONFIGURATION}

Deployment descriptors allow the user to configure the execution server on multiple levels:

- \textit{Server level}: The main level and the one that applies to all KJARs deployed on the server.
- \textit{KJAR level}: This enables you to configure descriptors on a per KJAR basis.
- \textit{Deploy time level}: Descriptors that apply while a KJAR is being deployed.

The granular configuration items specified by the deployment descriptors take precedence over the server level ones, except in case of configuration items that are collection based, which are merged. The hierarchy works like this: \textit{deploy time configuration} > \textit{KJAR configuration} > \textit{server configuration}.

\begin{note}
The deploy time configuration applies to deployments done via the REST API.
\end{note}

For example, if the persistence mode (one of the items you can configure) defined at the server level is \texttt{NONE} but the same mode is specified as \texttt{JPA} at the KJAR level, the actual mode will be \texttt{JPA} for that KJAR. If nothing is specified for the persistence mode in the deployment descriptor for that KJAR (or if there is no deployment descriptor), it will fall back to the server level configuration, which in this case is \texttt{NONE} (or to \texttt{JPA} if there is no server level deployment descriptor).

\section*{What Can You Configure?}

High level technical configuration details can be configured via deployment descriptors. The following table lists these along with the permissible and default values for each.

\begin{table}[h]
\centering
\caption{Deployment Descriptors}
\begin{tabular}{|l|l|}
\hline
\textbf{Property} & \textbf{Permissible Values} \\
\hline
Persistence Mode & \texttt{NONE}, \texttt{JPA} \\
\hline
Session Type & \texttt{STATELESS}, \texttt{STATEFUL} \\
\hline
Session Timeout & 10, 60, 300, 600 \\
\hline
\end{tabular}
\end{table}
<table>
<thead>
<tr>
<th>Configuration</th>
<th>XML Entry</th>
<th>Permissible Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistence unit name for runtime data</td>
<td>persistence-unit</td>
<td>Any valid persistence package name</td>
<td>org.jbpm.domain</td>
</tr>
<tr>
<td>Persistence unit name for audit data</td>
<td>audit-persistence-unit</td>
<td>Any valid persistence package name</td>
<td>org.jbpm.domain</td>
</tr>
<tr>
<td>Persistence mode</td>
<td>persistence-mode</td>
<td>JPA, NONE</td>
<td>JPA</td>
</tr>
<tr>
<td>Audit mode</td>
<td>audit-mode</td>
<td>JPA, JMS or NONE</td>
<td>JPA</td>
</tr>
<tr>
<td>Runtime Strategy</td>
<td>runtime-strategy</td>
<td>SINGLETON, PER_REQUEST or PER_PROCESS_INSTANCE</td>
<td>SINGLETON</td>
</tr>
<tr>
<td>List of Event Listeners to be registered</td>
<td>event-listeners</td>
<td>Valid listener class names as <strong>ObjectModel</strong></td>
<td>No default value</td>
</tr>
<tr>
<td>List of Task Event Listeners to be registered</td>
<td>task-event-listeners</td>
<td>Valid listener class names as <strong>ObjectModel</strong></td>
<td>No default value</td>
</tr>
<tr>
<td>List of Work Item Handlers to be registered</td>
<td>work-item-handlers</td>
<td>Valid Work Item Handler classes given as <strong>NamedObjectHandler</strong></td>
<td>No default value</td>
</tr>
<tr>
<td>List of Globals to be registered</td>
<td>globals</td>
<td>Valid Global variables given as <strong>NamedObjectModel</strong></td>
<td>No default value</td>
</tr>
<tr>
<td>Marshalling strategies to be registered (for pluggable variable persistence)</td>
<td>marshalling-strategies</td>
<td>Valid <strong>ObjectModel</strong> classes</td>
<td>No default value</td>
</tr>
<tr>
<td>Required Roles to be granted access to the resources of the KJAR</td>
<td>required-roles</td>
<td>String role names</td>
<td>No default value</td>
</tr>
<tr>
<td>Additional Environment Entries for KIE session</td>
<td>environment-entries</td>
<td>Valid <strong>NamedObjectModel</strong></td>
<td>No default value</td>
</tr>
<tr>
<td>Additional configuration options of KIE session</td>
<td>configurations</td>
<td>Valid <strong>NamedObjectModel</strong></td>
<td>No default value</td>
</tr>
</tbody>
</table>
### 13.2. MANAGING DEPLOYMENT DESCRIPTORS

Deployment descriptors can be configured in Business Central in Menu → Design → $PROJECT_NAME → Settings → Deployments.

Every time a project is created, a stock kie-deployment-descriptor.xml file is generated with default values.

It is not necessary to provide a full deployment descriptor for all KJARs. Providing partial deployment descriptors is possible and recommended. For example, if you need to use a different audit mode, you can specify that for the KJAR only, all other properties will have the default value defined at the server level.

When using OVERRIDE_ALL merge mode, all configuration items must be specified, because the relevant KJAR will always use specified configuration and will not merge with any other deployment descriptor in the hierarchy.

### 13.3. RESTRICTING ACCESS TO THE RUNTIME ENGINE

The required-roles configuration item can be edited in the deployment descriptors. This property restricts access to the runtime engine on a per-KJAR or per-server level by ensuring that access to certain processes is only granted to users that belong to groups defined by this property.

The security role can be used to restrict access to process definitions or restrict access at run time.

The default behavior is to add required roles to this property based on repository restrictions. You can edit these properties manually if required by providing roles that match actual roles defined in the security realm.

**Procedure**

1. To open the project deployment descriptors configuration in Business Central, open Menu → Design → $PROJECT_NAME → Settings → Deployments.

### Configuration

<table>
<thead>
<tr>
<th>Configuration</th>
<th>XML Entry</th>
<th>Permissible Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes used for serialization in the remote services</td>
<td>remoteable-class</td>
<td>Valid CustomClass</td>
<td>No default value</td>
</tr>
</tbody>
</table>

**WARNING**

Do not use the Singleton runtime strategy with the EJB Timer Scheduler (the default scheduler in KIE Server) in a production environment. This combination can result in Hibernate problems under load. Per process instance runtime strategy is recommended if there is no specific reason to use other strategies. For more information about this limitation, see Hibernate issues with Singleton strategy and EJBTimerScheduler.
2. From the list of configuration settings, click **Required Roles**, then click **Add Required Role**.

3. In the **Add Required Role** window, type the name of the role that you want to have permission to access this deployment, then click **Add**.

4. To add more roles with permission to access the deployment, repeat the previous steps.

5. When you have finished adding all required roles, click **Save**.
CHAPTER 14. ACCESSING RUNTIME DATA FROM BUSINESS CENTRAL

The following pages in Business Central allow you to view KIE Server runtime data:

- Process Reports
- Task Reports
- Process Definitions
- Process Instances
- Execution Errors
- Jobs
- Tasks

These pages use the credentials of the currently logged in user to load data from KIE Server. Therefore, to be able to view the runtime data in Business Central, ensure that the following conditions are met:

- The user exists in the KIE container (deployment unit) running the Business Central application. This user must have **admin**, **analyst**, or **developer** roles assigned, in addition to the **kie-server** role, with full access to the runtime data. The **manager** and **process_admin** roles also allow access to runtime data pages in Business Central.
- The user exists in the KIE container (deployment unit) running KIE Server and has the **kie-server** role assigned.
- Communication between Business Central and KIE Server is established. That is, KIE Server is registered in the Process Automation Manager controller, which is part of Business Central.
CHAPTER 15. PROMETHEUS METRICS MONITORING IN RED HAT PROCESS AUTOMATION MANAGER

Prometheus is an open-source systems monitoring toolkit that you can use with Red Hat Process Automation Manager to collect and store metrics related to the execution of business rules, processes, Decision Model and Notation (DMN) models, and other Red Hat Process Automation Manager assets. You can access the stored metrics through a REST API call to the KIE Server, through the Prometheus expression browser, or using a data-graphing tool such as Grafana.

You can configure Prometheus metrics monitoring for an on-premise KIE Server instance, for KIE Server on Spring Boot, or for a KIE Server deployment on Red Hat OpenShift Container Platform.


IMPORTANT

Red Hat support for Prometheus is limited to the setup and configuration recommendations provided in Red Hat product documentation.

15.1. CONFIGURING PROMETHEUS METRICS MONITORING FOR KIE SERVER


Prerequisites

- KIE Server is installed.
- You have kie-server user role access to KIE Server.
- Prometheus is installed. For information about downloading and using Prometheus, see the Prometheus documentation page.

Procedure

1. In your KIE Server instance, set the org.kie.prometheus.server.ext.disabled system property to false to enable the Prometheus extension. You can define this property when you start KIE Server or in the standalone.xml or standalone-full.xml file of Red Hat Process Automation Manager distribution.

2. If you are running Red Hat Process Automation Manager on Spring Boot, configure the required key in the application.properties system property:

   Spring Boot application.properties key for Red Hat Process Automation Manager and Prometheus
In the `prometheus.yaml` file of your Prometheus distribution, add the following settings in the `scrape_configs` section to configure Prometheus to scrape metrics from KIE Server:

```yaml
scrape_configs:
  - job_name: 'kie-server'
    metrics_path: /SERVER_PATH/services/rest/metrics
    basicAuth:
      username: USER_NAME
      password: PASSWORD
    static_configs:
      - targets: ["HOST:PORT"]
```

Scrape configurations in `prometheus.yaml` file for Spring Boot (if applicable)

```yaml
scrape_configs:
  - job_name: 'kie'
    metrics_path: /rest/metrics
    static_configs:
      - targets: ["HOST:PORT"]
```

Replace the values according to your KIE Server location and settings.

Start the KIE Server instance.

**Example start command for Red Hat Process Automation Manager on Red Hat JBoss EAP**

```
$ cd ~/EAP_HOME/bin
$ ./standalone.sh --c standalone-full.xml
```

After you start the configured KIE Server instance, Prometheus begins collecting metrics and KIE Server publishes the metrics to the REST API endpoint `http://HOST:PORT/SERVER/services/rest/metrics` (or on Spring Boot, to `http://HOST:PORT/rest/metrics`).

In a REST client or curl utility, send a REST API request with the following components to verify that KIE Server is publishing the metrics:

For REST client:

- **Authentication**: Enter the user name and password of the KIE Server user with the `kie-server` role.
- **HTTP Headers**: Set the following header:
  - **Accept**: `application/json`
- **HTTP method**: Set to `GET`.
URL: Enter the KIE Server REST API base URL and metrics endpoint, such as http://localhost:8080/kie-server/services/rest/metrics (or on Spring Boot, http://localhost:8080/rest/metrics).

For curl utility:

- **-u**: Enter the user name and password of the KIE Server user with the kie-server role.
- **-H**: Set the following header:
  
  - `accept: application/json`
- **-X**: Set to GET.

**Example curl command for Red Hat Process Automation Manager on Red Hat JBoss EAP**

```
curl -u 'baAdmin:password@1' -X GET "http://localhost:8080/kie-server/services/rest/metrics"
```

**Example curl command for Red Hat Process Automation Manager on Spring Boot**

```
curl -u 'baAdmin:password@1' -X GET "http://localhost:8080/rest/metrics"
```

**Example server response**

```
# HELP kie_server_container_started_total Kie Server Started Containers
# TYPE kie_server_container_started_total counter
kie_server_container_started_total{container_id="task-assignment-kjar-1.0",} 1.0
# HELP solvers_running Number of solvers currently running
# TYPE solvers_running gauge
solvers_running 0.0
# HELP dmn_evaluate_decision_nanosecond DMN Evaluation Time
# TYPE dmn_evaluate_decision_nanosecond histogram
# HELP solver_duration_seconds Time in seconds it took solver to solve the constraint problem
# TYPE solver_duration_seconds summary
solver_duration_seconds_count{solver_id="100tasks-5employees.xml",} 1.0
solver_duration_seconds_sum{solver_id="100tasks-5employees.xml",} 179.828255925
solver_duration_seconds_count{solver_id="24tasks-8employees.xml",} 1.0
solver_duration_seconds_sum{solver_id="24tasks-8employees.xml",} 179.995759653
# HELP drl_match_fired_nanosecond Drools Firing Time
# TYPE drl_match_fired_nanosecond histogram
# HELP dmn_evaluate_failed_count DMN Evaluation Failed
# TYPE dmn_evaluate_failed_count counter
# HELP kie_server_start_time Kie Server Start Time
# TYPE kie_server_start_time gauge
kie_server_start_time{name="myapp-kieserver",server_id="myapp-kieserver",location="http://myapp-kieserver-demo-monitoring.127.0.0.1.nip.io:80/services/rest/server",version="7.4.0.redhat-20190428",} 1.557221271502E12
# HELP kie_server_container_running_total Kie Server Running Containers
```
CHAPTER 15. PROMETHEUS METRICS MONITORING IN RED HAT PROCESS AUTOMATION MANAGER

# TYPE kie_server_container_running_total gauge
kie_server_container_running_total{container_id="task-assignment-kjar-1.0",} 1.0

# HELP solver_score_calculation_speed Number of moves per second for a particular solver solving the constraint problem
# TYPE solver_score_calculation_speed_summary
solver_score_calculation_speed_count{solver_id="100tasks-5employees.xml",} 1.0
solver_score_calculation_speed_sum{solver_id="100tasks-5employees.xml",} 6997.0
solver_score_calculation_speed_count{solver_id="24tasks-8employees.xml",} 1.0
solver_score_calculation_speed_sum{solver_id="24tasks-8employees.xml",} 19772.0

# HELP kie_server_case_started_total Kie Server Started Cases
# TYPE kie_server_case_started_total counter
kie_server_case_started_total{case_definition_id="itorders.orderhardware",} 1.0

# HELP kie_server_case_running_total Kie Server Running Cases
# TYPE kie_server_case_running_total gauge
kie_server_case_running_total{case_definition_id="itorders.orderhardware",} 2.0

# HELP kie_server_data_set_registered_total Kie Server Data Set Registered
# TYPE kie_server_data_set_registered_total gauge
kie_server_data_set_registered_total{name="jbpmProcessInstanceLogs::CUSTOM",uuid="jbpmProcessInstanceLogs",} 1.0
kie_server_data_set_registered_total{name="jbpmRequestList::CUSTOM",uuid="jbpmRequestList",} 1.0
kie_server_data_set_registered_total{name="tasksMonitoring::CUSTOM",uuid="tasksMonitoring",} 1.0
kie_server_data_set_registered_total{name="jbpmHumanTasks::CUSTOM",uuid="jbpmHumanTasks",} 1.0
kie_server_data_set_registered_total{name="jbpmHumanTasksWithUser::FILTERED_PO_TASK",uuid="jbpmHumanTasksWithUser",} 1.0
kie_server_data_set_registered_total{name="jbpmHumanTasksWithVariables::CUSTOM",uuid="jbpmHumanTasksWithVariables",} 1.0
kie_server_data_set_registered_total{name="jbpmProcessInstancesWithVariables::CUSTOM",uuid="jbpmProcessInstances",} 1.0
kie_server_data_set_registered_total{name="jbpmExecutionErrorList::CUSTOM",uuid="jbpmExecutionErrorList",} 1.0
kie_server_data_set_registered_total{name="processesMonitoring::CUSTOM",uuid="processesMonitoring",} 1.0
kie_server_data_set_registered_total{name="jbpmHumanTasksWithAdmin::FILTERED_BA_TASK",uuid="jbpmHumanTasksWithAdmin",} 1.0

# HELP kie_server_execution_error_total Kie Server Execution Errors
# TYPE kie_server_execution_error_total counter

# HELP kie_server_task_completed_total Kie Server Completed Tasks
# TYPE kie_server_task_completed_total counter

# HELP kie_server_container_running_total Kie Server Running Containers
# TYPE kie_server_container_running_total gauge
kie_server_container_running_total{container_id="itorders_1.0.0-SNAPSHOT",} 1.0

# HELP kie_server_job_cancelled_total Kie Server Cancelled Jobs
# TYPE kie_server_job_cancelled_total counter

# HELP kie_server_process_instance_started_total Kie Server Started Process Instances
# TYPE kie_server_process_instance_started_total counter
kie_server_process_instance_started_total{container_id="itorders_1.0.0-SNAPSHOT",process_id="itorders.orderhardware",} 1.0

# HELP solver_duration_seconds Time in seconds it took solver to solve the constraint problem
# TYPE solver_duration_seconds summary

# HELP kie_server_task_skipped_total Kie Server Skipped Tasks
# TYPE kie_server_task_skipped_total counter
# HELP kie_server_data_set_execution_time_seconds Kie Server Data Set Execution Time
# TYPE kie_server_data_set_execution_time_seconds_summary
kie_server_data_set_execution_time_seconds_count{uuid="jbpmProcessInstances",} 8.0
kie_server_data_set_execution_time_seconds_sum{uuid="jbpmProcessInstances",} 0.05600000000000001
# HELP kie_server_job_scheduled_total Kie Server Started Jobs
# TYPE kie_server_job_scheduled_total counter
# HELP kie_server_data_set_execution_total Kie Server Data Set Execution
# TYPE kie_server_data_set_execution_total counter
kie_server_data_set_execution_total{uuid="jbpmProcessInstances",} 8.0
# HELP kie_server_process_instance_completed_total Kie Server Completed Process Instances
# TYPE kie_server_process_instance_completed_total counter
# HELP kie_server_job_running_total Kie Server Running Jobs
# TYPE kie_server_job_running_total gauge
# HELP kie_server_task_failed_total Kie Server Failed Tasks
# TYPE kie_server_task_failed_total counter
# HELP kie_server_task_exited_total Kie Server Exited Tasks
# TYPE kie_server_task_exited_total counter
# HELP dmn_evaluate_decision_nanosecond DMN Evaluation Time
# TYPE dmn_evaluate_decision_nanosecond histogram
# HELP kie_server_data_set_lookups_total Kie Server Data Set Running Lookups
# TYPE kie_server_data_set_lookups_total gauge
dmn_process_instance_duration_seconds{container_id="itorders_1.0.0-SNAPSHOT",process_id="itorders.orderhardware",task_name="Prepare hardware spec",} 1.0
# HELP dmn_evaluate_failed_count DMN Evaluation Failed
# TYPE dmn_evaluate_failed_count counter
# HELP kie_server_task_added_total Kie Server Added Tasks
# TYPE kie_server_task_added_total counter
kie_server_task_added_total{deployment_id="itorders_1.0.0-SNAPSHOT",process_id="itorders.orderhardware",task_name="Prepare hardware spec",} 1.0
# HELP drl_match_fired_nanosecond Drools Firing Time
# TYPE drl_match_fired_nanosecond histogram
# HELP kie_server_container_started_total Kie Server Started Containers
# TYPE kie_server_container_started_total counter
kie_server_container_started_total{container_id="itorders_1.0.0-SNAPSHOT",} 1.0
# HELP kie_server_process_instance_sla_violated_total Kie Server Process Instances SLA Violated
# TYPE kie_server_process_instance_sla_violated_total counter
# HELP kie_server_task_duration_seconds Kie Server Task Duration
# TYPE kie_server_task_duration_seconds summary
# HELP kie_server_job_executed_total Kie Server Executed Jobs
# TYPE kie_server_job_executed_total counter
# HELP kie_server_deployments_active_total Kie Server Active Deployments
# TYPE kie_server_deployments_active_total gauge
kie_server_deployments_active_total{deployment_id="itorders_1.0.0-SNAPSHOT",} 1.0
# HELP kie_server_process_instance_running_total Kie Server Running Process Instances
# TYPE kie_server_process_instance_running_total gauge
# HELP solvers_running Number of solvers currently running
# TYPE solvers_running gauge
solvers_running 0.0
# HELP kie_server_work_item_duration_seconds Kie Server Work Items Duration
# TYPE kie_server_work_item_duration_seconds summary
# HELP kie_server_job_duration_seconds Kie Server Job Duration
# TYPE kie_server_job_duration_seconds summary
# HELP solver_score_calculation_speed Number of moves per second for a particular solver solving the constraint problem
# TYPE solver_score_calculation_speed summary
# HELP kie_server_start_time Kie Server Start Time
# TYPE kie_server_start_time gauge

If the metrics are not available in KIE Server, review and verify the KIE Server and Prometheus configurations described in this section.

You can also interact with your collected metrics in the Prometheus expression browser at http://HOST:PORT/graph, or integrate your Prometheus data source with a data-graphing tool such as Grafana:

Figure 15.1. Prometheus expression browser with KIE Server metrics

Figure 15.2. Prometheus expression browser with KIE Server target

Figure 15.3. Grafana dashboard with KIE Server metrics for DMN models
Figure 15.3. Grafana dashboard with KIE Server metrics for DMN models

Figure 15.4. Grafana dashboard with KIE Server metrics for solvers

Figure 15.5. Grafana dashboard with KIE Server metrics for processes, cases, and tasks
15.2. CONFIGURING PROMETHEUS METRICS MONITORING FOR KIE SERVER ON RED HAT OPENSOURCE CONTAINER PLATFORM


Prerequisites

- KIE Server is installed and deployed on Red Hat OpenShift Container Platform. For more information about KIE Server on OpenShift, see the relevant OpenShift deployment option in the Product documentation for Red Hat Process Automation Manager 7.12.
- You have kie-server user role access to KIE Server.
- Prometheus Operator is installed. For information about downloading and using Prometheus Operator, see the Prometheus Operator project in GitHub.

Procedure

1. In the DeploymentConfig object of your KIE Server deployment on OpenShift, set the PROMETHEUS_SERVER_EXT_DISABLED environment variable to false to enable the Prometheus extension. You can set this variable in the OpenShift web console or use the oc command in a command terminal:

   oc set env dc/<dc_name> PROMETHEUS_SERVER_EXT_DISABLED=false -n <namespace>

   If you have not yet deployed your KIE Server on OpenShift, then in the OpenShift template that you plan to use for your OpenShift deployment (for example, rhpam712-prod-immutable-kieserver.yaml), you can set the PROMETHEUS_SERVER_EXT_DISABLED template parameter to false to enable the Prometheus extension.

   If you are using the OpenShift Operator to deploy KIE Server on OpenShift, then in your KIE Server configuration, set the PROMETHEUS_SERVER_EXT_DISABLED environment variable to false to enable the Prometheus extension:

   ```yaml
   apiVersion: app.kiegroup.org/v1
   kind: KieApp
   metadata:
     name: enable-prometheus
   spec:
     environment: rhpam-trial
   ```
2. Create a `service-metrics.yaml` file to add a service that exposes the metrics from KIE Server to Prometheus:

```yaml
objects:
  servers:
    - env:
        - name: PROMETHEUS_SERVER_EXT_DISABLED
          value: "false"
```

3. In a command terminal, use the `oc` command to apply the `service-metrics.yaml` file to your OpenShift deployment:

   ```bash
   oc apply -f service-metrics.yaml
   ```

4. Create an OpenShift secret, such as `metrics-secret`, to access the Prometheus metrics on KIE Server. The secret must contain the "username" and "password" elements with KIE Server user credentials. For information about OpenShift secrets, see the Secrets chapter in the OpenShift Developer Guide.

5. Create a `service-monitor.yaml` file that defines the `ServiceMonitor` object. A service monitor enables Prometheus to connect to the KIE Server metrics service.

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: rhpam-service-monitor
  labels:
    team: frontend
spec:
  selector:
    matchLabels:
      metrics: rhpam
  endpoints:
```
6. In a command terminal, use the `oc` command to apply the `service-monitor.yaml` file to your OpenShift deployment:

   ```
   oc apply -f service-monitor.yaml
   ```

   After you complete these configurations, Prometheus begins collecting metrics and KIE Server publishes the metrics to the REST API endpoint `http://HOST:PORT/kie-server/services/rest/metrics`.

   You can interact with your collected metrics in the Prometheus expression browser at `http://HOST:PORT/graph`, or integrate your Prometheus data source with a data-graphing tool such as Grafana.

   The host and port for the Prometheus expression browser location `http://HOST:PORT/graph` was defined in the route where you exposed the Prometheus web console when you installed the Prometheus Operator. For information about OpenShift routes, see the Routes chapter in the OpenShift Architecture documentation.

   Figure 15.6. Prometheus expression browser with KIE Server metrics
Figure 15.7. Prometheus expression browser with KIE Server target

Figure 15.8. Grafana dashboard with KIE Server metrics for DMN models

Figure 15.9. Grafana dashboard with KIE Server metrics for solvers
15.3. EXTENDING PROMETHEUS METRICS MONITORING IN KIE SERVER WITH CUSTOM METRICS

After you configure your KIE Server instance to use Prometheus metrics monitoring, you can extend the Prometheus functionality in KIE Server to use custom metrics according to your business needs. Prometheus then collects and stores your custom metrics along with the default metrics that KIE Server exposes with Prometheus.

As an example, this procedure defines custom Decision Model and Notation (DMN) metrics to be collected and stored by Prometheus.

Prerequisites

- Prometheus metrics monitoring is configured for your KIE Server instance. For information about Prometheus configuration with KIE Server on-premise, see Section 15.1, “Configuring Prometheus metrics monitoring for KIE Server”. For information about Prometheus configuration with KIE Server on Red Hat OpenShift Container Platform, see Section 15.2, “Configuring Prometheus metrics monitoring for KIE Server on Red Hat OpenShift Container Platform”.

Additional resources

- Prometheus Operator
- Getting started with the Prometheus Operator
- Prometheus RBAC
- Grafana Support for Prometheus
- Using Prometheus in Grafana
- OpenShift deployment options in Product documentation for Red Hat Process Automation Manager 7.12
Procedure

1. Create an empty Maven project and define the following packaging type and dependencies in the pom.xml file for the project:

Example pom.xml file in the sample project

```xml
<packaging>jar</packaging>

<properties>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-common</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-drools</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-prometheus</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-dmn-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-dmn-core</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.jbpm</groupId>
    <artifactId>jbpm-services-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
</dependencies>
```
2. Implement the relevant listener from the `org.kie.server.services.prometheus.PrometheusMetricsProvider` interface as part of the custom listener class that defines your custom Prometheus metrics, as shown in the following example:

**Sample implementation of the DMNRuntimeEventListener listener in a custom listener class**

```java
package org.kie.server.ext.prometheus;

import io.prometheus.client.Gauge;
import org.kie.dmn.api.core.ast.DecisionNode;
import org.kie.dmn.api.core.event.AfterEvaluateBKMEvent;
import org.kie.dmn.api.core.event.AfterEvaluateContextEntryEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionServiceEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionTableEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateBKMEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateContextEntryEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionServiceEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionTableEvent;
import org.kie.dmn.api.core.event.DMNRuntimeEventListener;
import org.kie.server.api.model.ReleaseId;
import org.kie.server.services.api.KieContainerInstance;

public class ExampleCustomPrometheusMetricListener implements DMNRuntimeEventListener {

    private final KieContainerInstance kieContainer;

    private final Gauge randomGauge = Gauge.build()
        .name("random_gauge_nanosecond")
        .help("Random gauge as an example of custom KIE Prometheus metric")
        .labelNames("container_id", "group_id", "artifact_id", "version",
            "decision_namespace", "decision_name")
        .register();

    public ExampleCustomPrometheusMetricListener(KieContainerInstance containerInstance) { 
```
The PrometheusMetricsProvider interface contains the required listeners for collecting Prometheus metrics. The interface is incorporated by the kie-server-services-prometheus dependency that you declared in your project pom.xml file.

In this example, the ExampleCustomPrometheusMetricListener class implements the DMNRuntimeEventListener listener (from the PrometheusMetricsProvider interface) and defines the custom DMN metrics to be collected and stored by Prometheus.

3. Implement the PrometheusMetricsProvider interface as part of a custom metrics provider class that associates your custom listener with the PrometheusMetricsProvider interface, as shown in the following example:

Sample implementation of the PrometheusMetricsProvider interface in a custom metrics provider class

```java
package org.kie.server.ext.prometheus;

kieContainer = containerInstance;

public void beforeEvaluateDecision(BeforeEvaluateDecisionEvent e) {
}

public void afterEvaluateDecision(AfterEvaluateDecisionEvent e) {
    DecisionNode decisionNode = e.getDecision();
    ReleaseId releaseId = kieContainer.getResource().getReleaseId();
    randomGauge.labels(kieContainer.getContainerId(), releaseId.getGroupId(),
        releaseId.getArtifactId(), releaseId.getVersion(),
        decisionNode.getModelName(), decisionNode.getModelNamespace())
        .set((int) (Math.random() * 100));
}

public void beforeEvaluateBKM(BeforeEvaluateBKMEvent event) {
}

public void afterEvaluateBKM(AfterEvaluateBKMEvent event) {
}

public void beforeEvaluateContextEntry(BeforeEvaluateContextEntryEvent event) {
}

public void afterEvaluateContextEntry(AfterEvaluateContextEntryEvent event) {
}

public void beforeEvaluateDecisionTable(BeforeEvaluateDecisionTableEvent event) {
}

public void afterEvaluateDecisionTable(AfterEvaluateDecisionTableEvent event) {
}

public void beforeEvaluateDecisionService(BeforeEvaluateDecisionServiceEvent event) {
}

public void afterEvaluateDecisionService(AfterEvaluateDecisionServiceEvent event) {
}
```
In this example, the `MyPrometheusMetricsProvider` class implements the `PrometheusMetricsProvider` interface and includes your custom `ExampleCustomPrometheusMetricListener` listener class.

4. To make the new metrics provider discoverable for KIE Server, create a `META-INF/services/org.kie.server.services.prometheus.PrometheusMetricsProvider` file in your Maven project and add the fully qualified class name of the `PrometheusMetricsProvider` implementation class within the file. For this example, the file contains the single line `org.kie.server.ext.prometheus.MyPrometheusMetricsProvider`.

5. Build your project and copy the resulting JAR file into the `~/kie-server.war/WEB-INF/lib` directory of your project. For example, on Red Hat JBoss EAP, the path to this directory is `EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib`.

If you are deploying Red Hat Process Automation Manager on Red Hat OpenShift Container Platform, create a custom KIE Server image and add this JAR file to the image. For more information about creating a custom KIE Server image with an additional JAR file, see *Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform 4 using Operators*.

6. Start the KIE Server and deploy the built project to the running KIE Server. You can deploy the

```java
import org.jbpm.executor.AsynchronousJobListener;
import org.jbpm.services.api.DeploymentEventListener;
import org.kie.api.event.rule.AgendaEventListener;
import org.kie.api.event.rule.DefaultAgendaEventListener;
import org.kie.dmn.api.core.event.DMNRuntimeEventListener;
import org.kie.server.services.api.KieContainerInstance;
import org.kie.server.services.prometheus.PrometheusMetricsProvider;
import org.optaplanner.core.impl.phase.event.PhaseLifecycleListener;
import org.optaplanner.core.impl.phase.event.PhaseLifecycleListenerAdapter;

public class MyPrometheusMetricsProvider implements PrometheusMetricsProvider {

    public DMNRuntimeEventListener createDMNRuntimeEventListener(KieContainerInstance kContainer) {
        return new ExampleCustomPrometheusMetricListener(kContainer);
    }

    public AgendaEventListener createAgendaEventListener(String kieSessionId, KieContainerInstance kContainer) {
        return new DefaultAgendaEventListener();
    }

    public PhaseLifecycleListener createPhaseLifecycleListener(String solverId) {
        return new PhaseLifecycleListenerAdapter() {
        };
    }

    public AsynchronousJobListener createAsynchronousJobListener() {
        return null;
    }

    public DeploymentEventListener createDeploymentEventListener() {
        return null;
    }
}
```
b. Start the KIE Server and deploy the built project to the running KIE Server. You can deploy the project using the Business Central interface or the KIE Server REST API (a PUT request to http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}). After your project is deployed on a running KIE Server, Prometheus begins collecting metrics and KIE Server publishes the metrics to the REST API endpoint http://HOST:PORT/SERVER/services/rest/metrics (or on Spring Boot, to http://HOST:PORT/rest/metrics).
CHAPTER 16. CONFIGURING OPENSШIFT CONNECTION TIMEOUT

By default, the OpenShift route is configured to time out HTTP requests that are longer than 30 seconds. This may cause session timeout issues in Business Central resulting in the following behaviors:

- "Unable to complete your request. The following exception occurred: (TypeError) : Cannot read property 'indexOf' of null."
- "Unable to complete your request. The following exception occurred: (TypeError) : b is null."
- A blank page is displayed when clicking the Project or Server links in Business Central.

All Business Central templates already include extended timeout configuration.

To configure longer timeout on Business Central OpenShift routes, add the `haproxy.router.openshift.io/timeout: 60s` annotation on the target route:

```yaml
- kind: Route
  apiVersion: v1
  id: "$APPLICATION_NAME-rhpamcentr-http"
  metadata:
    name: "$APPLICATION_NAME-rhpamcentr"
    labels:
      application: "$APPLICATION_NAME"
      annotations:
        description: Route for Business Central's http service.
        haproxy.router.openshift.io/timeout: 60s
  spec:
    host: "$BUSINESS_CENTRAL_HOSTNAME_HTTP"
    to:
      name: "$APPLICATION_NAME-rhpamcentr"
```

For a full list of global route-specific timeout annotations, see the OpenShift Documentation.
CHAPTER 17. PERSISTENCE

Binary persistence, or marshaling, converts the state of the process instance into a binary data set. Binary persistence is a mechanism used to store and retrieve information persistently. The same mechanism is also applied to the session state and work item states.

When you enable persistence of a process instance:

- Red Hat Process Automation Manager transforms the process instance information into binary data. Custom serialization is used instead of Java serialization for performance reasons.

- The binary data is stored together with other process instance metadata, such as process instance ID, process ID, and the process start date.

The session can also store other forms of state, such as the state of timer jobs, or data required for business rules evaluation. Session state is stored separately as a binary data set along with the ID of the session and metadata. You can restore the session state by reloading a session with given ID. Use `ksession.getId()` to get the session ID.

Red Hat Process Automation Manager 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, and other process instance data) and the event types.

- **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, you can 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.

17.1. CONFIGURING KIE SERVER PERSISTENCE

You can configure the KIE Server persistence by passing Hibernate or JPA parameters as system properties.

The KIE Server can acknowledge the system properties with the following prefixes and you can use every Hibernate or JPA parameters with these prefixes:

- `javax.persistence`
- `hibernate`

Procedure

1. To configure KIE Server persistence, complete any of the following tasks:
   - If you want to configure KIE Server persistence using Red Hat JBoss EAP configuration file, complete the following tasks:
     i. In your Red Hat Process Automation Manager installation directory, navigate to the `standalone-full.xml` file. For example, if you use Red Hat JBoss EAP installation for Red Hat Process Automation Manager, go to
Open the standalone-full.xml file and under the <system-properties> tag, set your Hibernate or JPA parameters as system properties.

**Example of configuring KIE Server persistence using Hibernate parameters**

```xml
<system-properties>
  ...
  <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
  ...
</system-properties>
```

**Example of configuring KIE Server persistence using JPA parameters**

```xml
<system-properties>
  ...
  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://mysql.db.server:3306/my_database?
  useSSL=false&serverTimezone=UTC"/>
  ...
</system-properties>
```

If you want to configure KIE Server persistence using command line, complete the following tasks:

i. Pass the parameters directly from the command line using `-Dkey=value` as follows:

**Example of configuring KIE Server persistence using Hibernate parameters:**

```
$EAP_HOME/bin/standalone.sh -Dhibernate.hbm2ddl.auto=create-drop
```

**Example of configuring KIE Server persistence using JPA parameters:**

```
$EAP_HOME/bin/standalone.sh -
javax.persistence.jdbc.url=jdbc:mysql://mysql.db.server:3306/my_database?
useSSL=false&serverTimezone=UTC
```

## 17.2. CONFIGURING SAFE POINTS

To allow persistence, add the jbpm-persistence JAR files to the classpath of your application and configure the process engine to use persistence. The process engine automatically stores the runtime state in the storage when the process 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 process engine, the process 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.

BPMN2 safe point nodes ensure that the process engine saves the state of the process definition at the point where the execution stops and the transaction is committed. The following BPMN2 nodes are considered safe points:
- All intermediate CATCH events
  - Timer Intermediate event
  - Error Intermediate event
  - Conditional Intermediate event
  - Compensation Intermediate event
  - Signal Intermediate event
  - Escalation Intermediate event
  - Message Intermediate event

- User tasks
- Custom (defined by the user) service tasks that do not complete the task in the handler

If a failure occurs and you need to restore the process 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.

Consider the runtime persistence data to be internal to the process engine. You should not access persisted runtime data or modify them directly because this might have unexpected side effects.

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

17.3. SESSION PERSISTENCE ENTITIES

Sessions are persisted as SessionInfo entities. These persist the state of the runtime KIE session, and store the following data:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>lastModificationDate</td>
<td>The last time that entity was saved to a database.</td>
<td></td>
</tr>
<tr>
<td>rulesByteArray</td>
<td>The state of a session.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>startDate</td>
<td>The session start time.</td>
<td></td>
</tr>
<tr>
<td>OPTLOCK</td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
</tbody>
</table>

17.4. PROCESS INSTANCE PERSISTENCE ENTITIES
Process instances are persisted as **ProcessInstanceInfo** entities, which persist the state of a process instance on runtime and store the following data:

**Table 17.2. ProcessInstanceInfo**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>instanceId</code></td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td><code>lastModificationDate</code></td>
<td>The last time that the entity was saved to a database.</td>
<td></td>
</tr>
<tr>
<td><code>lastReadDate</code></td>
<td>The last time that the entity was retrieved from the database.</td>
<td></td>
</tr>
<tr>
<td><code>processId</code></td>
<td>The ID of the process.</td>
<td></td>
</tr>
<tr>
<td><code>processInstanceByteArray</code></td>
<td>The state of a process instance in form of a binary data set.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td><code>startDate</code></td>
<td>The start time of the process.</td>
<td></td>
</tr>
<tr>
<td><code>state</code></td>
<td>An integer representing the state of a process instance.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td><code>OPTLOCK</code></td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
</tbody>
</table>

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

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

**Table 17.3. EventTypes**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>instanceId</code></td>
<td>A reference to the <strong>ProcessInstanceInfo</strong> primary key and foreign key constraint on this column.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td><code>element</code></td>
<td>A finished event in the process.</td>
<td></td>
</tr>
</tbody>
</table>

17.5. WORK ITEM PERSISTENCE ENTITIES

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 17.4. WorkItemInfo**
### Field Description Nullable

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>workItemId</td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>name</td>
<td>The name of the work item.</td>
<td></td>
</tr>
<tr>
<td>processInstanceId</td>
<td>The (primary key) ID of the process. There is no foreign key constraint on this field.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>state</td>
<td>The state of a work item.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>OPTLOCK</td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
<tr>
<td>workitembytearray</td>
<td>The work item state in as a binary data set.</td>
<td>NOT NULL</td>
</tr>
</tbody>
</table>

### 17.6. CORRELATION KEY ENTITIES

The **CorrelationKeyInfo** entity contains information about the correlation key assigned to the given process instance. This table is optional. Use it only when you require correlation capabilities.

Table 17.5. CorrelationKeyInfo

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyId</td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>name</td>
<td>The assigned name of the correlation key.</td>
<td></td>
</tr>
<tr>
<td>processInstanceId</td>
<td>The ID of the process instance which is assigned to the correlation key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>OPTLOCK</td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
</tbody>
</table>

The **CorrelationPropertyInfo** entity contains information about correlation properties for a correlation key assigned to the process instance.

Table 17.6. CorrelationPropertyInfo

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyId</td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
</tbody>
</table>
### Field Description Nullable

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the property.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>The value of the property.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>OPTLOCK</td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
<tr>
<td>correlationKey_keyId</td>
<td>A foreign key mapped to the correlation key.</td>
<td>NOT NULL</td>
</tr>
</tbody>
</table>

### 17.7. CONTEXT MAPPING ENTITY

The **ContextMappingInfo** entity contains information about the contextual information mapped to a **KieSession**. This is an internal part of **RuntimeManager** and can be considered optional when **RuntimeManager** is not used.

**Table 17.7. ContextMappingInfo**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingId</td>
<td>The primary key.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>CONTEXT_ID</td>
<td>The context identifier.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>KSESSION_ID</td>
<td>The <strong>KieSession</strong> identifier.</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>OPTLOCK</td>
<td>A version field containing a lock value.</td>
<td></td>
</tr>
<tr>
<td>OWNER_ID</td>
<td>Holds the identifier of the deployment unit that the given mapping is associated with</td>
<td></td>
</tr>
</tbody>
</table>

### 17.8. PESSIMISTIC LOCKING SUPPORT

The default locking mechanism for persistence of processes is *optimistic*. With multi-thread high concurrency to the same process instance, this locking strategy can result in bad performance.

#### 17.8.1. Configuring pessimistic locking support via code

You can set the locking mechanism on a per process basis and to allow it to be *pessimistic*. You can make the changes at a per KIE Session level or Runtime Manager level as well and not just at the process level.

To set a process to use pessimistic locking, use the following configuration in the runtime environment:

```java
import org.kie.api.runtime.Environment;
```
import org.kie.api.runtime.EnvironmentName;
import org.kie.api.runtime.manager.RuntimeManager;
import org.kie.api.runtime.manager.RuntimeManagerFactory;
...

env.set(EnvironmentName.USE_PESSIMISTIC_LOCKING, true);

RuntimeManager manager =
RuntimeManagerFactory.Factory.get().newPerRequestRuntimeManager(environment);

1 env is an instance of org.kie.api.runtime.Environment.

2 Create your Runtime Manager by using this environment.

17.8.2. Configuring pessimistic locking support in Business Central

Business Central supports pessimistic locking of processes. To set a process to use pessimistic locking in Business Central, use the following procedure:

Prerequisites

- You have sufficient user permissions in Business Central.
- You have created a Business Central project.

Procedure

1. In Business Central, go to Menu → Design → Projects.
2. Select your project.
3. Click Settings tab to access the project settings.
4. Go to Deployments → Environment entries.
5. Click Add Environment Entry.
6. To add environment entries, enter the following values:
   - Name: Enter your environment name.
   - Value: Enter your environment value.
   - Resolver type: Select the MVEL, Reflection, or Spring resolver type as per your requirement.
   - Parameters: Add the parameters.
7. Click Save and click Save again to confirm your changes.

17.9. PERSISTING PROCESS VARIABLES IN A SEPARATE DATABASE SCHEMA IN RED HAT PROCESS AUTOMATION MANAGER
When you create process variables to use within the processes that you define, Red Hat Process Automation Manager stores those process variables as binary data in a default database schema. You can persist process variables in a separate database schema for greater flexibility in maintaining and implementing your process data.

For example, persisting your process variables in a separate database schema can help you perform the following tasks:

- Maintain process variables in human-readable format
- Make the variables available to services outside of Red Hat Process Automation Manager
- Clear the log of the default database tables in Red Hat Process Automation Manager without losing process variable data

**NOTE**

This procedure applies to process variables only. This procedure does not apply to case variables.

**Prerequisites**

- You have defined processes in Red Hat Process Automation Manager for which you want to implement variables.

- If you want to persist variables in a database schema outside of Red Hat Process Automation Manager, you have created a data source and the separate database schema that you want to use. For information about creating data sources, see Configuring Business Central settings and properties.

**Procedure**

1. In the data object file that you use as a process variable, add the following elements to configure variable persistence:

   **Example Person.java object configured for variable persistence**

   ```java
   @javax.persistence.Entity
   @javax.persistence.Table(name = "Person")
   public class Person extends org.drools.persistence.jpa.marshaller.VariableEntity implements java.io.Serializable {

   static final long serialVersionUID = 1L;

   @javax.persistence.GeneratedValue(strategy = javax.persistence.GenerationType.AUTO, generator = "PERSON_ID_GENERATOR")
   @javax.persistence.Id
   @javax.persistence.SequenceGenerator(name = "PERSON_ID_GENERATOR", sequenceName = "PERSON_ID_SEQ")
   private java.lang.Long id;

   private java.lang.String name;

   private java.lang.Integer age;
   ```
Configures the data object as a persistence entity.

Defines the database table name used for the data object.

Creates a separate MappedVariable mapping table that maintains the relationship between this data object and the associated process instance. If you do not need this relationship maintained, you do not need to extend the VariableEntity class. Without this extension, the data object is still persisted, but contains no additional data.

Configures the data object as a serializable object.

Sets a persistence ID for the object.

To make the data object persistable using Business Central, navigate to the data object file in your project, click the Persistence icon in the upper-right corner of the window, and configure the persistence behavior:
2. In the pom.xml file of your project, add the following dependency for persistence support. This dependency contains the VariableEntity class that you configured in your data object.

**Project dependency for persistence**

```xml
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-persistence-jpa</artifactId>
  <version>${rhpam.version}</version>
  <scope>provided</scope>
</dependency>
```

3. In the ~/META-INF/kie-deployment-descriptor.xml file of your project, configure the JPA marshalling strategy and a persistence unit to be used with the marshaller. The JPA marshalling strategy and persistence unit are required for objects defined as entities.

**JPA marshaller and persistence unit configured in the kie-deployment-descriptor.xml file**

```xml
<marshalling-strategy>
  <resolver>mvel</resolver>
  <identifier>
    org.drools.persistence.jpa.marshaller.JPAPlaceholderResolverStrategy("myPersistenceUnit", classLoader)
  </identifier>
  <parameters/>
</marshalling-strategy>
```

4. In the ~/META-INF directory of your project, create a persistence.xml file that specifies in which data source you want to persist the process variable:

```xml
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-persistence-jpa</artifactId>
  <version>${rhpam.version}</version>
  <scope>provided</scope>
</dependency>
```
Example persistence.xml file with data source configuration

```xml
    <persistence-unit name="myPersistenceUnit" transaction-type="JTA">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <jta-data-source>java:jboss/datasources/ExampleDS</jta-data-source>
        <class>org.space.example.Person</class>
        <exclude-unlisted-classes>true</exclude-unlisted-classes>
        <properties>
            <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
            <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.id.new_generator_mappings" value="false"/>
            <property name="hibernate.transaction.jta.platform" value="org.hibernate.service.jta.platform.internal.JBossAppServerJtaPlatform"/>
        </properties>
    </persistence-unit>
</persistence>
```

1 Sets the data source in which the process variable is persisted

To configure the marshalling strategy, persistence unit, and data source using Business Central, navigate to project Settings → Deployments → Marshalling Strategies and to project Settings → Persistence:

Figure 17.2. JPA marshaller configuration in Business Central
Figure 17.3. Persistence unit and data source configuration in Business Central

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>persistence-unit</td>
<td>myPersistenceUnit</td>
</tr>
<tr>
<td>persistence-provider</td>
<td>org.hibernate.boot.HibernatePersistence</td>
</tr>
<tr>
<td>persistence-unit</td>
<td>myPersistenceUnit</td>
</tr>
<tr>
<td>persistence-provider</td>
<td>org.hibernate.boot.HibernatePersistence</td>
</tr>
<tr>
<td>data-source</td>
<td>javax.sql.DataSourceExampleDS</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibernate-dialect</td>
<td>org.hibernate.dialect.H2Dialect</td>
</tr>
<tr>
<td>hibernate-max_fetch_depth</td>
<td>2</td>
</tr>
<tr>
<td>hibernate-null-objects</td>
<td>update</td>
</tr>
</tbody>
</table>
CHAPTER 18. DEFINE THE LDAP LOGIN DOMAIN

When you are setting up Red Hat Process Automation Manager to use LDAP for authentication and authorization, define the LDAP login domain because the Git SSH authentication may use another security domain.

To define the LDAP login domain, use the `org.uberfire.domain` system property. For example, on Red Hat JBoss Enterprise Application Platform, add this property in the `standalone.xml` file as shown:

```xml
<system-properties>
  <!-- other system properties -->
  <property name="org.uberfire.domain" value="LDAPAuth"/>
</system-properties>
```

Ensure that the authenticated user has appropriate roles (admin, analyst, reviewer) associated with it in LDAP.
CHAPTER 19. AUTHENTICATING THIRD-PARTY CLIENTS THROUGH RH-SSO

To use the different remote services provided by Business Central or by KIE Server, your client, such as curl, wget, web browser, or a custom REST client, must authenticate through the RH-SSO server and have a valid token to perform the requests. To use the remote services, the authenticated user must have the following roles:

- **rest-all** for using Business Central remote services.
- **kie-server** for using the KIE Server remote services.

Use the RH-SSO Admin Console to create these roles and assign them to the users that will consume the remote services.

Your client can authenticate through RH-SSO using one of these options:

- Basic authentication, if it is supported by the client
- Token-based authentication

19.1. BASIC AUTHENTICATION

If you enabled basic authentication in the RH-SSO client adapter configuration for both Business Central and KIE Server, you can avoid the token grant and refresh calls and call the services as shown in the following examples:

- For web based remote repositories endpoint:
  
  ```bash
  curl http://admin:password@localhost:8080/business-central/rest/repositories
  ```

- For KIE Server:
  
  ```bash
  curl http://admin:password@localhost:8080/kie-server/services/rest/server/
  ```
CHAPTER 20. KIE SERVER SYSTEM PROPERTIES

KIE Server accepts the following system properties (bootstrap switches) to configure the behavior of the server:

Table 20.1. System properties for disabling KIE Server extensions

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.drools.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the Business Rule Management (BRM) support (for example, rules support).</td>
</tr>
<tr>
<td>org.jbpm.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the Red Hat Process Automation Manager support (for example, processes support).</td>
</tr>
<tr>
<td>org.jbpm.ui.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the Red Hat Process Automation Manager UI extension.</td>
</tr>
<tr>
<td>org.jbpm.case.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the Red Hat Process Automation Manager case management extension.</td>
</tr>
<tr>
<td>org.optaplanner.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the Red Hat build of OptaPlanner support.</td>
</tr>
<tr>
<td>org.kie.prometheus.server.ext.disabled</td>
<td>true, false</td>
<td>true</td>
<td>If set to true, disables the Prometheus Server extension.</td>
</tr>
<tr>
<td>org.kie.scenariosimulation.server.ext.disabled</td>
<td>true, false</td>
<td>true</td>
<td>If set to true, disables the Test scenario Server extension.</td>
</tr>
<tr>
<td>org.kie.dmn.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the KIE Server DMN support.</td>
</tr>
<tr>
<td>org.kie.swagger.server.ext.disabled</td>
<td>true, false</td>
<td>false</td>
<td>If set to true, disables the KIE Server swagger documentation support</td>
</tr>
</tbody>
</table>

**NOTE**

Some Process Automation Manager controller properties listed in the following table are marked as required. Set these properties when you create or remove KIE Server containers in Business Central. If you use KIE Server separately without any interaction with Business Central, you do not need to set the required properties.

Table 20.2. System properties required for Process Automation Manager controller
<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.kie.server.id</td>
<td>String</td>
<td>N/A</td>
<td>An arbitrary ID to be assigned to the server. If a headless Process Automation Manager controller is configured outside of Business Central, this is the ID under which the server connects to the headless Process Automation Manager controller to fetch the KIE container configurations. If not provided, the ID is automatically generated.</td>
</tr>
<tr>
<td>org.kie.server.user</td>
<td>String</td>
<td>kieserver</td>
<td>The user name used to connect with KIE Server from the Process Automation Manager controller, required when running in managed mode. Set this property in Business Central system properties. Set this property when using a Process Automation Manager controller.</td>
</tr>
<tr>
<td>org.kie.server.pwd</td>
<td>String</td>
<td>kieserver1 !</td>
<td>The password used to connect with KIE Server from the Process Automation Manager controller, required when running in managed mode. Set this property in Business Central system properties. Set this property when using a Process Automation Manager controller.</td>
</tr>
<tr>
<td>org.kie.server.token</td>
<td>String</td>
<td>N/A</td>
<td>A property that enables you to use token-based authentication between the Process Automation Manager controller and KIE Server instead of the basic user name and password authentication. The Process Automation Manager controller sends the token as a parameter in the request header. The server requires long-lived access tokens because the tokens are not refreshed.</td>
</tr>
<tr>
<td>org.kie.server.location</td>
<td>URL</td>
<td>N/A</td>
<td>The URL of the KIE Server instance used by the Process Automation Manager controller to call back on this server, for example, <a href="http://localhost:8230/kie-server/services/rest/server">http://localhost:8230/kie-server/services/rest/server</a>. Setting this property is required when using a Process Automation Manager controller.</td>
</tr>
<tr>
<td>org.kie.server.controller</td>
<td>Comma-separated list</td>
<td>N/A</td>
<td>A comma-separated list of URLs to the Process Automation Manager controller REST endpoints, for example, <a href="http://localhost:8080/business-central/rest/controller">http://localhost:8080/business-central/rest/controller</a>. Setting this property is required when using a Process Automation Manager controller.</td>
</tr>
</tbody>
</table>
### Table 20.3. Persistence system properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.kie.server.persistence.ds</td>
<td>String</td>
<td>N/A</td>
<td>A data source JNDI name. Set this property when enabling the BPM support.</td>
</tr>
<tr>
<td>org.kie.server.persistence.tm</td>
<td>String</td>
<td>N/A</td>
<td>A transaction manager platform for Hibernate properties. Set this property when enabling the BPM support.</td>
</tr>
<tr>
<td>org.kie.server.persistence.dialect</td>
<td>String</td>
<td>N/A</td>
<td>The Hibernate dialect to be used. Set this property when enabling the BPM support.</td>
</tr>
<tr>
<td>org.kie.server.persistence.schema</td>
<td>String</td>
<td>N/A</td>
<td>The database schema to be used.</td>
</tr>
</tbody>
</table>

### Table 20.4. Executor system properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.kie.server.controller.user</td>
<td>String</td>
<td>kieserver</td>
<td>The user name to connect to the Process Automation Manager controller REST API. Setting this property is required when using a Process Automation Manager controller.</td>
</tr>
<tr>
<td>org.kie.server.controller.pwd</td>
<td>String</td>
<td>kieserver1</td>
<td>! The password to connect to the Process Automation Manager controller REST API. Setting this property is required when using a Process Automation Manager controller.</td>
</tr>
<tr>
<td>org.kie.server.controller.token</td>
<td>String</td>
<td>N/A</td>
<td>A property that enables you to use token-based authentication between KIE Server and the Process Automation Manager controller instead of the basic user name and password authentication. The server sends the token as a parameter in the request header. The server requires long-lived access tokens because the tokens are not refreshed.</td>
</tr>
<tr>
<td>org.kie.server.controller.connect</td>
<td>Long</td>
<td>10000</td>
<td>The waiting time in milliseconds between repeated attempts to connect KIE Server to the Process Automation Manager controller when the server starts.</td>
</tr>
<tr>
<td>Property</td>
<td>Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>org.kie.executor.interval</code></td>
<td>Integer</td>
<td>0</td>
<td>The time between the moment the Red Hat Process Automation Manager executor finishes a job and the moment it starts a new one, in a time unit specified in the <code>org.kie.executor.timeunit</code> property.</td>
</tr>
<tr>
<td><code>org.kie.executor.timeunit</code></td>
<td><code>java.util.concurrent.TimeUnit</code></td>
<td>SECONDS</td>
<td>The time unit in which the <code>org.kie.executor.interval</code> property is specified.</td>
</tr>
<tr>
<td><code>org.kie.executor.pool.size</code></td>
<td>Integer</td>
<td>1</td>
<td>The number of threads used by the Red Hat Process Automation Manager executor.</td>
</tr>
<tr>
<td><code>org.kie.executor.retry.count</code></td>
<td>Integer</td>
<td>3</td>
<td>The number of retries the Red Hat Process Automation Manager executor attempts on a failed job.</td>
</tr>
<tr>
<td><code>org.kie.executor.jms.queue</code></td>
<td>String</td>
<td><code>queue/KIE.SERVER.EXECUTOR</code></td>
<td>Job executor JMS queue for KIE Server.</td>
</tr>
<tr>
<td><code>org.kie.executor.jms.jobHeader</code></td>
<td><code>true, false</code></td>
<td><code>false</code></td>
<td>If set to <code>true</code>, the request identifier is included in the JMS header as the <code>jobId</code> property.</td>
</tr>
<tr>
<td><code>org.kie.executor.disabled</code></td>
<td><code>true, false</code></td>
<td><code>false</code></td>
<td>If set to <code>true</code>, disables the KIE Server executor.</td>
</tr>
</tbody>
</table>

Table 20.5. Human task system properties
<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.jbpm.ht.callback</td>
<td>mvel, ldap, db, jaas, props, custom</td>
<td>jaas</td>
<td>A property that specifies the implementation of user group callback to be used:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>mvel</strong>: Default; mostly used for testing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>ldap</strong>: LDAP; requires additional configuration in the <code>jbpm.usergroup.callback.properties</code> file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>db</strong>: Database; requires additional configuration in the <code>jbpm.usergroup.callback.properties</code> file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>jaas</strong>: JAAS; delegates to the container to fetch information about user data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>props</strong>: A simple property file; requires additional file that keeps all information (users and groups).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>custom</strong>: A custom implementation; specify the fully qualified name of the class in the <code>org.jbpm.ht.custom.callback</code> property.</td>
</tr>
<tr>
<td>org.jbpm.ht.custom.callback</td>
<td>Fully qualified name</td>
<td>N/A</td>
<td>A custom implementation of the UserGroupCallback interface in case the <code>org.jbpm.ht.callback</code> property is set to <code>custom</code>.</td>
</tr>
<tr>
<td>org.jbpm.task.cleanup.enabled</td>
<td>true, false</td>
<td>true</td>
<td>Enables task cleanup job listener to remove tasks once the process instance is completed.</td>
</tr>
<tr>
<td>org.jbpm.task.bam.enabled</td>
<td>true, false</td>
<td>true</td>
<td>Enables task BAM module to store task related information.</td>
</tr>
<tr>
<td>org.jbpm.ht.admin.user</td>
<td>String</td>
<td>Administrator</td>
<td>User who can access all the tasks from KIE Server.</td>
</tr>
<tr>
<td>org.jbpm.ht.admin.group</td>
<td>String</td>
<td>Administrators</td>
<td>The group that users must belong to in order to view all the tasks from KIE Server.</td>
</tr>
</tbody>
</table>

Table 20.6. System properties for loading keystore
<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>kie.keystore.keyStoreUR L</strong></td>
<td>URL</td>
<td>N/A</td>
<td>The URL is used to load a Java Cryptography Extension KeyStore (JCEKS). For example, file:///home/kie/keystores/keystore.jceks.</td>
</tr>
<tr>
<td><strong>kie.keystore.keyStorePwd</strong></td>
<td>String</td>
<td>N/A</td>
<td>The password is used for the JCEKS.</td>
</tr>
<tr>
<td><strong>kie.keystore.key.server.alias</strong></td>
<td>String</td>
<td>N/A</td>
<td>The alias name of the key for REST services where the password is stored.</td>
</tr>
<tr>
<td><strong>kie.keystore.key.server.pwd</strong></td>
<td>String</td>
<td>N/A</td>
<td>The password of an alias for REST services.</td>
</tr>
<tr>
<td><strong>kie.keystore.key.ctrl.alias</strong></td>
<td>String</td>
<td>N/A</td>
<td>The alias of the key for default REST Process Automation Manager controller.</td>
</tr>
<tr>
<td><strong>kie.keystore.key.ctrl.pwd</strong></td>
<td>String</td>
<td>N/A</td>
<td>The password of an alias for default REST Process Automation Manager controller.</td>
</tr>
</tbody>
</table>

Table 20.7. System properties for retrying committing transactions

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>org.kie.optlock.retries</strong></td>
<td>Integer</td>
<td>5</td>
<td>This property describes how many times the process engine retries a transaction before failing permanently.</td>
</tr>
<tr>
<td><strong>org.kie.optlock.delay</strong></td>
<td>Integer</td>
<td>50</td>
<td>The delay time before the first retry, in milliseconds.</td>
</tr>
<tr>
<td><strong>org.kie.optlock.delayFactor</strong></td>
<td>Integer</td>
<td>4</td>
<td>The multiplier for increasing the delay time for each subsequent retry. With the default values, the process engine waits 50 milliseconds before the first retry, 200 milliseconds before the second retry, 800 milliseconds before the third retry, and so on.</td>
</tr>
</tbody>
</table>

Table 20.8. Other system properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>kie.maven.settings.custom</strong></td>
<td>Path</td>
<td>N/A</td>
<td>The location of a custom settings.xml file for Maven configuration.</td>
</tr>
<tr>
<td>Property</td>
<td>Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>kie.server.jms.queues.response</td>
<td>String</td>
<td>queue/KIE.SER VER.RESPONSE</td>
<td>The response queue JNDI name for JMS.</td>
</tr>
<tr>
<td>org.drools.server.filter.classes</td>
<td>true, false</td>
<td>false</td>
<td>When set to true, the Drools KIE Server extension accepts custom classes annotated by the XmlRootElement or Remotable annotations only.</td>
</tr>
<tr>
<td>org.kie.server.bypass.auth.user</td>
<td>true, false</td>
<td>false</td>
<td>A property that enables you to bypass the authenticated user for task-related operations, for example queries.</td>
</tr>
<tr>
<td>org.jbpm.rule.task.firelimit</td>
<td>Integer</td>
<td>10000</td>
<td>This property specifies the maximum number of executed rules to avoid situations where rules run into an infinite loop and make the server completely unresponsive.</td>
</tr>
<tr>
<td>org.jbpm.ejb.timer.local.cache</td>
<td>true, false</td>
<td>true</td>
<td>This property turns off the EJB Timers local cache.</td>
</tr>
<tr>
<td>org.kie.server.domain</td>
<td>String</td>
<td>N/A</td>
<td>The JAAS LoginContext domain used to authenticate users when using JMS.</td>
</tr>
<tr>
<td>org.kie.server.repo</td>
<td>Path</td>
<td>.</td>
<td>The location where KIE Server state files are stored.</td>
</tr>
<tr>
<td>Property</td>
<td>Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>org.kie.server.sync.deploy</td>
<td>true, false</td>
<td>false</td>
<td>A property that instructs KIE Server to hold the deployment until the Process Automation Manager controller provides the container deployment configuration. This property only affects servers running in managed mode. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* false: The connection to the Process Automation Manager controller is asynchronous. The application starts, connects to the Process Automation Manager controller, and once successful, deploys the containers. The application accepts requests even before the containers are available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>* true: The deployment of the server application joins the Process Automation Manager controller connection thread with the main deployment and awaits its completion. This option can lead to a potential deadlock in case more applications are on the same server. Use only one application on one server instance.</td>
</tr>
<tr>
<td>org.kie.server.startup.strategy</td>
<td>ControllerBasedStartupStrategy, LocalContainerStartupStrategy</td>
<td>ControllerBasedStartupStrategy</td>
<td>The Startup strategy of KIE Server used to control the KIE containers that are deployed and the order in which they are deployed.</td>
</tr>
<tr>
<td>org.kie.server.mgmt.api.disabled</td>
<td>true, false</td>
<td>false</td>
<td>When set to true, disables KIE Server management API.</td>
</tr>
<tr>
<td>org.kie.server.xstream.enabled.packages</td>
<td>Java packages like org.kie.example. You can also specify wildcard expressions like org.kie.example:*</td>
<td>N/A</td>
<td>A property that specifies additional packages to allowlist for marshalling using XStream.</td>
</tr>
<tr>
<td>Property</td>
<td>Values</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>org.kie.store.services.class</td>
<td>String</td>
<td>org.drools.persistence.jpa.KnowledgeStoreServiceImpl</td>
<td>Fully qualified name of the class that implements KieStoreServices that are responsible for bootstrapping KieSession instances.</td>
</tr>
<tr>
<td>org.kie.server.strict.id.format</td>
<td>true, false</td>
<td>false</td>
<td>While using JSON marshalling, if the property is set to true, it will always return a response in the proper JSON format. For example, if the original response contains only a single number, then the response is wrapped in a JSON format. For example, {&quot;value&quot; : 1}.</td>
</tr>
<tr>
<td>org.kie.server.strict.jaxb.format</td>
<td>true, false</td>
<td>false</td>
<td>When the value of this property is set to true, KIE Server validates the data type of the data in the REST API payload. For example, if a data field has the number data type and contains something other than a number, you will receive an error.</td>
</tr>
</tbody>
</table>
CHAPTER 21. KIE SERVER CAPABILITIES AND EXTENSIONS

The capabilities in KIE Server are determined by plug-in extensions that you can enable, disable, or further extend to meet your business needs. KIE Server supports the following default capabilities and extensions:

Table 21.1. KIE Server capabilities and extensions

<table>
<thead>
<tr>
<th>Capability name</th>
<th>Extension name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KieServer</td>
<td>KieServer</td>
<td>Provides the core capabilities of KIE Server, such as creating and disposing KIE containers on your server instance</td>
</tr>
<tr>
<td>BRM</td>
<td>Drools</td>
<td>Provides the Business Rule Management (BRM) capabilities, such as inserting facts and executing business rules</td>
</tr>
<tr>
<td>BPM</td>
<td>jBPM</td>
<td>Provides the Business Process Management (BPM) capabilities, such as managing user tasks and executing business processes</td>
</tr>
<tr>
<td>BPM-UI</td>
<td>jBPM-UI</td>
<td>Provides additional user-interface capabilities related to business processes, such as rendering XML forms and SVG images in process diagrams</td>
</tr>
<tr>
<td>CaseMgmt</td>
<td>Case-Mgmt</td>
<td>Provides the case management capabilities for business processes, such as managing case definitions and milestones</td>
</tr>
<tr>
<td>BRP</td>
<td>OptaPlanner</td>
<td>Provides the Business Resource Planning (BRP) capabilities, such as implementing solvers</td>
</tr>
<tr>
<td>DMN</td>
<td>DMN</td>
<td>Provides the Decision Model and Notation (DMN) capabilities, such as managing DMN data types and executing DMN models</td>
</tr>
<tr>
<td>Swagger</td>
<td>Swagger</td>
<td>Provides the Swagger web-interface capabilities for interacting with the KIE Server REST API</td>
</tr>
</tbody>
</table>

To view the supported extensions of a running KIE Server instance, send a **GET** request to the following REST API endpoint and review the XML or JSON server response:

**Base URL for GET request for KIE Server information**

http://SERVER:PORT/kie-server/services/rest/server

**Example JSON response with KIE Server information**

```json
{
    "type": "SUCCESS",
    "msg": "Kie Server info",
    "result": {
        "kie-server-info": {
            "id": "test-kie-server",
            "version": "7.59.0.20190818-050814",
```
To enable or disable KIE Server extensions, configure the related *.server.ext.disabled KIE Server system property. For example, to disable the BRM capability, set the system property org.drools.server.ext.disabled=true. For all KIE Server system properties, see Chapter 20, KIE Server system properties.

By default, KIE Server extensions are exposed through REST or JMS data transports and use predefined client APIs. You can extend existing KIE Server capabilities with additional REST endpoints, extend supported transport methods beyond REST or JMS, or extend functionality in the KIE Server client.

This flexibility in KIE Server functionality enables you to adapt your KIE Server instances to your business needs, instead of adapting your business needs to the default KIE Server capabilities.

**IMPORTANT**

If you extend KIE Server functionality, Red Hat does not support the custom code that you use as part of your custom implementations and extensions.

### 21.1. Extending an Existing KIE Server Capability with a Custom REST API Endpoint

The KIE Server REST API enables you to interact with your KIE containers and business assets (such as business rules, processes, and solvers) in Red Hat Process Automation Manager without using the Business Central user interface. The available REST endpoints are determined by the capabilities

```json
"name": "test-kie-server",
"location": "http://localhost:8080/kie-server/services/rest/server",
"capabilities": [
  "KieServer",
  "BRM",
  "BPM",
  "CaseMgmt",
  "BPM-UI",
  "BRP",
  "DMN",
  "Swagger"
],
"messages": [
  {
    "severity": "INFO",
    "timestamp": {
      "java.util.Date": 1566169865791
    },
    "content": ["Server KieServerInfo{serverId='test-kie-server', version='7.59.0.20190818-050814', name='test-kie-server', location='http://localhost:8080/kie-server/services/rest/server', capabilities=[KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP, DMN, Swagger]', messages=null', mode=DEVELOPMENT}started successfully at Sun Aug 18 23:11:05 UTC 2019"
  }
],
"mode": "DEVELOPMENT"
}
```
enabled in your KIE Server system properties (for example, org.drools.server.ext.disabled=false for the BRM capability). You can extend an existing KIE Server capability with a custom REST API endpoint to further adapt the KIE Server REST API to your business needs.

As an example, this procedure extends the Drools KIE Server extension (for the BRM capability) with the following custom REST API endpoint:

**Example custom REST API endpoint**

```
/server/containers/instances/{containerId}/ksession/{ksessionId}
```

This example custom endpoint accepts a list of facts to be inserted into the working memory of the decision engine, automatically executes all rules, and retrieves all objects from the KIE session in the specified KIE container.

**Procedure**

1. Create an empty Maven project and define the following packaging type and dependencies in the pom.xml file for the project:

   **Example pom.xml file in the sample project**

   ```xml
   <packaging>jar</packaging>
   
   <properties>
   </properties>
   
   <dependencies>
   <dependency>
   <groupId>org.kie</groupId>
   <artifactId>kie-api</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie</groupId>
   <artifactId>kie-internal</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-api</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-common</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-drools</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-impl</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack-deployment</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack-deployment-drools</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack-deployment-impl</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack-deployment-impl-drools</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   
   <dependency>
   <groupId>org.kie.server</groupId>
   <artifactId>kie-server-services-stack-deployment-impl-drools-control</artifactId>
   <version>${version.org.kie}</version>
   </dependency>
   ```
2. Implement the `org.kie.server.services.api.KieServerApplicationComponentsService` interface in a Java class in your project, as shown in the following example:

### Sample implementation of the `KieServerApplicationComponentsService` interface

```java
public class CustomDroolsKieServerApplicationComponentsService implements KieServerApplicationComponentsService {

    private static final String OWNER_EXTENSION = "Drools";

    public Collection<Object> getAppComponents(String extension, SupportedTransports type, Object... services) {
        // Do not accept calls from extensions other than the owner extension:
        if (!OWNER_EXTENSION.equals(extension)) {
            return Collections.emptyList();
        }

        RulesExecutionService rulesExecutionService = null;
        KieServerRegistry context = null;

        for (Object object : services) {
            if (RulesExecutionService.class.isAssignableFrom(object.getClass())) {
                rulesExecutionService = (RulesExecutionService) object;
                continue;
            } else if (KieServerRegistry.class.isAssignableFrom(object.getClass())) {
                context = (KieServerRegistry) object;
                continue;
            }
        }

        List<Object> components = new ArrayList<Object>();
        if (SupportedTransports.REST.equals(type)) {
            components.add(new CustomResource(rulesExecutionService, context));
        }
    }
}
```
Delivers REST endpoints to the KIE Server infrastructure that is deployed when the application starts.

Specifies the extension that you are extending, such as the Drools extension in this example.

Returns all resources that the REST container must deploy. Each extension that is enabled in your KIE Server instance calls the getAppComponents method, so the if (!OWNER_EXTENSION.equals(extension)) call returns an empty collection for any extensions other than the specified OWNER_EXTENSION extension.

Lists the services from the specified extension that you want to use, such as the RulesExecutionService and KieServerRegistry services from the Drools extension in this example.

Specifies the transport type for the extension, either REST or JMS (REST in this example), and the CustomResource class that returns the resource as part of the components list.

3. Implement the CustomResource class that KIE Server can use to provide the additional functionality for the new REST resource, as shown in the following example:

Sample implementation of the CustomResource class

```java
// Custom base endpoint:
@Path("server/containers/instances/{containerId}/ksession")
public class CustomResource {

    private static final Logger logger = LoggerFactory.getLogger(CustomResource.class);

    private KieCommands commandsFactory = KieServices.Factory.get().getCommands();

    private RulesExecutionService rulesExecutionService;
    private KieServerRegistry registry;

    public CustomResource() {
    }

    public CustomResource(RulesExecutionService rulesExecutionService, KieServerRegistry registry) {
        this.rulesExecutionService = rulesExecutionService;
        this.registry = registry;
    }

    // Supported HTTP method, path parameters, and data formats:
    @POST
    @Path("/{ksessionId}")
    @Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})

    return components;
}
```
In this example, the `CustomResource` class for the custom endpoint specifies the following data and behavior:

- Uses the base endpoint `server/containers/instances/{containerId}/ksession`
- Uses `POST` HTTP method

```java
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public Response insertFireReturn(@Context HttpHeaders headers,
                                 @PathParam("containerId") String id,
                                 @PathParam("ksessionId") String ksessionId,
                                 String cmdPayload) {

    Variant v = getVariant(headers);
    String contentType = getContentType(headers);

    // Marshalling behavior and supported actions:
    MarshallingFormat format = MarshallingFormat.fromType(contentType);
    if (format == null) {
        format = MarshallingFormat.valueOf(contentType);
    }
    try {
        KieContainerInstance kci = registry.getContainer(id);
        Marshaller marshaller = kci.getMarshaller(format);
        List<?> listOffacts = marshaller.unmarshall(cmdPayload, List.class);
        List<Command<?>> commands =
                new ArrayList<Command<?>>();
        BatchExecutionCommand executionCommand =
                commandsFactory.newBatchExecution(commands, ksessionId);
        for (Object fact : listOffacts) {
            commands.add(commandsFactory.newInsert(fact, fact.toString()));
        }
        commands.add(commandsFactory.newFireAllRules());
        commands.add(commandsFactory.newGetObject());
        ExecutionResults results = rulesExecutionService.call(kci, executionCommand);
        String result = marshaller.marshall(results);
        logger.debug("Returning OK response with content '{}'", result);
        return createResponse(result, v, Response.Status.OK);
    } catch (Exception e) {
        // If marshalling fails, return the `call-container` response to maintain backward compatibility:
        String response = "Execution failed with error : " + e.getMessage();
        logger.debug("Returning Failure response with content '{}'", response);
        return createResponse(response, v, Response.Status.INTERNAL_SERVER_ERROR);
    }
}
```

In this example, the `CustomResource` class for the custom endpoint specifies the following data and behavior:

- Uses the base endpoint `server/containers/instances/{containerId}/ksession`
- Uses `POST` HTTP method
- Expects the following data to be given in REST requests:
  - The **containerId** as a path argument
  - The **ksessionId** as a path argument
  - List of facts as a message payload

- Supports all KIE Server data formats:
  - XML (JAXB, XStream)
  - JSON

- Unmarshals the payload into a `List<?>` collection and, for each item in the list, creates an `InsertCommand` instance followed by `FireAllRules` and `GetObject` commands.

- Adds all commands to the `BatchExecutionCommand` instance that calls to the decision engine.

4. To make the new endpoint discoverable for KIE Server, create a `META-INF/services/org.kie.server.services.api.KieServerApplicationComponentsService` file in your Maven project and add the fully qualified class name of the `KieServerApplicationComponentsService` implementation class within the file. For this example, the file contains the single line `org.kie.server.ext.drools.rest.CusomtDroolsKieServerApplicationComponentsService`.

5. Build your project and copy the resulting JAR file into the `~/kie-server.war/WEB-INF/lib` directory of your project. For example, on Red Hat JBoss EAP, the path to this directory is `EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib`.

6. Start KIE Server and deploy the built project to the running KIE Server. You can deploy the project using either the Business Central interface or the KIE Server REST API (a **PUT** request to `http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}`).

After your project is deployed on a running KIE Server, you can start interacting with your new REST endpoint.

For this example, you can use the following information to invoke the new endpoint:

- **Example request URL:** `http://localhost:8080/kie-server/services/rest/server/containers/instances/demo/ksession/defaultKieSession`

- **HTTP method:** POST

- **HTTP headers:**
  - **Content-Type:** application/json
  - **Accept:** application/json

- **Example message payload:**

```json
[
  {
    "org.jbpm.test.Person": {
      "name": "john",
      "age": 25
    }
  }
]
```
20.2. EXTENDING KIE SERVER TO USE A CUSTOM DATA TRANSPORT

By default, KIE Server extensions are exposed through REST or JMS data transports. You can extend KIE Server to support a custom data transport to adapt KIE Server transport protocols to your business needs.

As an example, this procedure adds a custom data transport to KIE Server that uses the Drools extension and that is based on Apache MINA, an open-source Java network-application framework. The example custom MINA transport exchanges string-based data that relies on existing marshalling operations and supports only JSON format.

Procedure

1. Create an empty Maven project and define the following packaging type and dependencies in the pom.xml file for the project:

Example pom.xml file in the sample project

```xml
<pom-revision>5.1.10.Final</pom-revision>
<groupId>org.kie.server</groupId>
<artifactId>kie-server</artifactId>
<version>${version.org.kie}</version>
<packaging>jar</packaging>
<properties>
</properties>
<dependencies>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-internal</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
  </dependency>
</dependencies>
```
2. Implement the `org.kie.server.services.api.KieServerExtension` interface in a Java class in your project, as shown in the following example:

Sample implementation of the `KieServerExtension` interface

```java
public class MinaDroolsKieServerExtension implements KieServerExtension {

    private static final Logger logger = LoggerFactory.getLogger(MinaDroolsKieServerExtension.class);

    public static final String EXTENSION_NAME = "Drools-Mina";

    private static final Boolean disabled = Boolean.parseBoolean(System.getProperty("org.kie.server.drools-mina.ext.disabled", "false"));
    private static final String MINA_HOST = System.getProperty("org.kie.server.drools-mina.ext.port", "localhost");
    private static final int MINA_PORT = Integer.parseInt(System.getProperty("org.kie.server.drools-mina.ext.port", "9123");

    // Taken from dependency on the 'Drools' extension:
    private KieContainerCommandService batchCommandService;
```
private IoAcceptor acceptor;

public boolean isActive() {
    return disabled == false;
}

public void init(KieServerImpl kieServer, KieServerRegistry registry) {

    KieServerExtension droolsExtension = registry.getServerExtension("Drools");
    if (droolsExtension == null) {
        logger.warn("No Drools extension available, quitting...");
        return;
    }

    List<Object> droolsServices = droolsExtension.getServices();
    for (Object object : droolsServices) {
        // If the given service is null (not configured), continue to the next service:
        if (object == null) {
            continue;
        }
        if (KieContainerCommandService.class.isAssignableFrom(object.getClass())) {
            batchCommandService = (KieContainerCommandService) object;
            continue;
        }
    }
    if (batchCommandService != null) {
        acceptor = new NioSocketAcceptor();
        acceptor.getFilterChain().addLast("codec", new ProtocolCodecFilter(new TextLineCodecFactory(Charset.forName("UTF-8"))));
        acceptor.setHandler(new TextBasedIoHandlerAdapter(batchCommandService));
        acceptor.getSessionConfig().setReadBufferSize(2048);
        acceptor.getSessionConfig().setIdleTime(IdleStatus.BOTH_IDLE, 10);
        try {
            acceptor.bind(new InetSocketAddress(MINA_HOST, MINA_PORT));
            logger.info("{} -- Mina server started at {} and port {}", toString(), MINA_HOST, MINA_PORT);
        } catch (IOException e) {
            logger.error("Unable to start Mina acceptor due to {}", e.getMessage(), e);
        }
    }
}

public void destroy(KieServerImpl kieServer, KieServerRegistry registry) {
    if (acceptor != null) {
        acceptor.dispose();
        acceptor = null;
    }
    logger.info("{} -- Mina server stopped", toString());
}

public void createContainer(String id, KieContainerInstance kieContainerInstance,
The **KieServerExtension** interface is the main extension interface that KIE Server can use to provide the additional functionality for the new MINA transport. The interface consists of the following components:

**Overview of the KieServerExtension interface**

```java
public interface KieServerExtension {
    boolean isActive();
    void init(KieServerImpl kieServer, KieServerRegistry registry);
    void destroy(KieServerImpl kieServer, KieServerRegistry registry);
}

Map<String, Object> parameters) {
    // Empty, already handled by the `Drools` extension
}

public void disposeContainer(String id, KieContainerInstance kieContainerInstance,
    Map<String, Object> parameters) {
    // Empty, already handled by the `Drools` extension
}

public List<Object> getAppComponents(SupportedTransports type) {
    // Nothing for supported transports (REST or JMS)
    return Collections.emptyList();
}

public <T> T getAppComponents(Class<T> serviceType) {
    return null;
}

public String getImplementedCapability() {
    return "BRM-Mina";
}

public List<Object> getServices() {
    return Collections.emptyList();
}

public String getExtensionName() {
    return EXTENSION_NAME;
}

public Integer getStartOrder() {
    return 20;
}

@Override
public String toString() {
    return EXTENSION_NAME + " KIE Server extension";
}
```

The **KieServerExtension** interface is the main extension interface that KIE Server can use to provide the additional functionality for the new MINA transport. The interface consists of the following components:
Specifies the capability that is covered by this extension. The capability must be unique within KIE Server.

Defines a human-readable name for the extension.

Determines when the specified extension should be started. For extensions that have dependencies on other extensions, this setting must not conflict with the parent setting. For example, in this case, this custom extension depends on the Drools extension, which has StartOrder set to 0, so this custom add-on extension must be greater than 0 (set to 20 in the sample implementation).

In the previous MinaDroolsKieServerExtension sample implementation of this interface, the init method is the main element for collecting services from the Drools extension and for bootstrapping the MINA server. All other methods in the KieServerExtension interface can remain with the standard implementation to fulfill interface requirements.

The TextBasedIoHandlerAdapter class is the handler on the MINA server that reacts to incoming requests.

3. Implement the TextBasedIoHandlerAdapter handler for the MINA server, as shown in the following example:

Sample implementation of the TextBasedIoHandlerAdapter handler

```java
public class TextBasedIoHandlerAdapter extends IoHandlerAdapter {

    private static final Logger logger = LoggerFactory.getLogger(TextBasedIoHandlerAdapter.class);

    private KieContainerCommandService batchCommandService;

    public TextBasedIoHandlerAdapter(KieContainerCommandService batchCommandService) {
        this.batchCommandService = batchCommandService;
    }
```
In this example, the handler class receives text messages and executes them in the Drools service.

Consider the following handler requirements and behavior when you use the TextBasedIoHandlerAdapter handler implementation:

- Anything that you submit to the handler must be a single line because each incoming transport request is a single line.

- You must pass a KIE container ID in this single line so that the handler expects the format containerID|payload.

- You can set a response in the way that it is produced by the marshaller. The response can be multiple lines.

- The handler supports a stream mode that enables you to send commands without disconnecting from a KIE Server session. To end a KIE Server session in stream mode, send either an exit or quit command to the server.

4. To make the new data transport discoverable for KIE Server, create a META-INF/services/org.kie.server.services.api.KieServerExtension file in your Maven project and add the fully qualified class name of the KieServerExtension implementation class within the file. For this example, the file contains the single line org.kie.server.ext.mina.MinaDroolsKieServerExtension.

5. Build your project and copy the resulting JAR file and the mina-core-2.0.9.jar file (which the
extension depends on in this example) into the ~/kie-server.war/WEB-INF/lib directory of your project. For example, on Red Hat JBoss EAP, the path to this directory is EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib.

6. Start the KIE Server and deploy the built project to the running KIE Server. You can deploy the project using either the Business Central interface or the KIE Server REST API (a PUT request to http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}).

After your project is deployed on a running KIE Server, you can view the status of the new data transport in your KIE Server log and start using your new data transport:

**New data transport in the server log**

Drools-Mina KIE Server extension -- Mina server started at localhost and port 9123
Drools-Mina KIE Server extension has been successfully registered as server extension

For this example, you can use Telnet to interact with the new MINA-based data transport in KIE Server:

**Starting Telnet and connecting to KIE Server on port 9123 in a command terminal**

telnet 127.0.0.1 9123

**Example interactions with KIE Server in a command terminal**

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

# Request body:
demo{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"john","age":25}}}},{"fire-all-rules":""}]

# Server response:
{
  "results" : [ {
    "key" : "",
    "value" : 1
  } ],
  "facts" : [ ]
}

demo{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"mary","age":22}}}},{"fire-all-rules":""}]

{
  "results" : [ {
    "key" : "",
    "value" : 1
  } ],
  "facts" : [ ]
}

demo{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"james","age":25}}}},{"fire-all-rules":""}]

{ "results" : [ {  
```
21.3. EXTENDING THE KIE SERVER CLIENT WITH A CUSTOM CLIENT API

KIE Server uses predefined client APIs that you can interact with to use KIE Server services. You can extend the KIE Server client with a custom client API to adapt KIE Server services to your business needs.

As an example, this procedure adds a custom client API to KIE Server to accommodate a custom data transport (configured previously for this scenario) that is based on Apache MINA, an open-source Java network-application framework.

Procedure

1. Create an empty Maven project and define the following packaging type and dependencies in the pom.xml file for the project:

Example pom.xml file in the sample project

```xml
<packaging>jar</packaging>

<properties>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-client</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-compiler</artifactId>
  </dependency>
</dependencies>
```
2. Implement the relevant `ServicesClient` interface in a Java class in your project, as shown in the following example:

**Sample RulesMinaServicesClient interface**

```java
public interface RulesMinaServicesClient extends RuleServicesClient {
}
```

A specific interface is required because you must register client implementations based on the interface, and you can have only one implementation for a given interface.

For this example, the custom MINA-based data transport uses the Drools extension, so this example `RulesMinaServicesClient` interface extends the existing `RuleServicesClient` client API from the Drools extension.

3. Implement the `RulesMinaServicesClient` interface that KIE Server can use to provide the additional client functionality for the new MINA transport, as shown in the following example:

**Sample implementation of the RulesMinaServicesClient interface**

```java
public class RulesMinaServicesClientImpl implements RulesMinaServicesClient {
    private String host;
    private Integer port;
    private Marshaller marshaller;

    public RulesMinaServicesClientImpl(KieServicesConfiguration configuration, ClassLoader classloader) {
        String[] serverDetails = configuration.getServerUrl().split(":");
        this.host = serverDetails[0];
        this.port = Integer.parseInt(serverDetails[1]);
        this.marshaller = MarshallerFactory.getMarshaller(configuration.getExtraJaxbClasses(), MarshallingFormat.JSON, classloader);
    }

    public ServiceResponse<String> executeCommands(String id, String payload) {
        try {
            String response = sendReceive(id, payload);
            if (response.startsWith("{")) {
                return new ServiceResponse<String>(ResponseType.SUCCESS, null, response);
            } else {
                return new ServiceResponse<String>(ResponseType.FAILURE, response);
            }
        } catch (Exception e) {
            throw new KieServicesException("Unable to send request to KIE Server", e);
        }
    }
}
```
This example implementation specifies the following data and behavior:

- Uses socket-based communication for simplicity
- Relies on default configurations from the KIE Server client and uses `ServerUrl` for providing the host and port of the MINA server
- Specifies JSON as the marshalling format
- Requires received messages to be JSON objects that start with an open bracket {
- Uses direct socket communication with a blocking API while waiting for the first line of the response and then reads all lines that are available
- Does not use stream mode and therefore disconnects the KIE Server session after invoking a command

4. Implement the `org.kie.server.client.helper.KieServicesClientBuilder` interface in a Java class in your project, as shown in the following example:

**Sample implementation of the KieServicesClientBuilder interface**

```java
public class MinaClientBuilderImpl implements KieServicesClientBuilder {
    public String getImplementedCapability() {
        return "BRM-Mina";
    }

    public Map<Class<?>, Object> build(KieServicesConfiguration configuration, ClassLoader classLoader) {
        Map<Class<?>, Object> services = new HashMap<Class<?>, Object>();

        services.put(RulesMinaServicesClient.class, new RulesMinaServicesClientImpl(configuration, classLoader));

        return services;
    }
}
```

1. Enables you to provide additional client APIs to the generic KIE Server client infrastructure
2. Defines the KIE Server capability (extension) that the client uses
3. Provides a map of the client implementations, where the key is the interface and the value is the fully initialized implementation

5. To make the new client API discoverable for the KIE Server client, create a META-INF/services/org.kie.server.client.helper.KieServicesClientBuilder file in your Maven project and add the fully qualified class name of the `KieServicesClientBuilder` implementation class within the file. For this example, the file contains the single line `org.kie.server.ext.mina.client.MinaClientBuilderImpl`.

6. Build your project and copy the resulting JAR file into the `~/kie-server.war/WEB-INF/lib` directory of your project. For example, on Red Hat JBoss EAP, the path to this directory is `EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib`.

7. Start KIE Server and deploy the built project to the running KIE Server. You can deploy the project using either the Business Central interface or the KIE Server REST API (a PUT request to `http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}`).

After your project is deployed on a running KIE Server, you can start interacting with your new KIE Server client. You use your new client in the same way as the standard KIE Server client, by creating the client configuration and client instance, retrieving the service client by type, and invoking client methods.
For this example, you can create a `RulesMinaServiceClient` client instance and invoke operations on KIE Server through the MINA transport:

**Sample implementation to create the `RulesMinaServiceClient` client**

```java
protected RulesMinaServicesClient buildClient() {
    KieServicesConfiguration configuration =
        KieServicesFactory.newRestConfiguration("localhost:9123", null, null);
    List<String> capabilities = new ArrayList<String>();
    // Explicitly add capabilities (the MINA client does not respond to `get-server-info` requests):
    capabilities.add("BRM-Mina");

    configuration.setCapabilities(capabilities);
    configuration.setMarshallingFormat(MarshallingFormat.JSON);
    configuration.addJaxbClasses(extraClasses);

    KieServicesClient kieServicesClient =
        KieServicesFactory.newKieServicesClient(configuration);

    RulesMinaServicesClient rulesClient =
        kieServicesClient.getServicesClient(RulesMinaServicesClient.class);

    return rulesClient;
}
```

**Sample configuration to invoke operations on KIE Server through the MINA transport**

```java
RulesMinaServicesClient rulesClient = buildClient();

List<Command<?>> commands = new ArrayList<Command<?>>();
BatchExecutionCommand executionCommand =
    commandsFactory.newBatchExecution(commands, "defaultKieSession");

Person person = new Person();
person.setName("mary");
commands.add(commandsFactory.newInsert(person, "person"));
commands.add(commandsFactory.newFireAllRules("fired"));

ServiceResponse<String> response = rulesClient.executeCommands(containerId,
    executionCommand);
Assert.assertNotNull(response);
Assert.assertEquals(ResponseType.SUCCESS, response.getType());
String data = response.getResult();
Marshaller marshaller = MarshallerFactory.getMarshaller(extraClasses,
    MarshallingFormat.JSON, this.getClass().getClassLoader());
ExecutionResultImpl results = marshaller.unmarshall(data, ExecutionResultImpl.class);
Assert.assertNotNull(results);
```
Object personResult = results.getValue("person");
Assert.assertTrue(personResult instanceof Person);

Assert.assertEquals("mary", ((Person) personResult).getName());
Assert.assertEquals("JBoss Community", ((Person) personResult).getAddress());
Assert.assertEquals(true, ((Person) personResult).isRegistered());
CHAPTER 22. PERFORMANCE TUNING CONSIDERATIONS WITH KIE SERVER

The following key concepts or suggested practices can help you optimize KIE Server performance. These concepts are summarized in this section as a convenience and are explained in more detail in the cross-referenced documentation, where applicable. This section will expand or change as needed with new releases of Red Hat Process Automation Manager.

Ensure that development mode is enabled during development

You can set KIE Server or specific projects in Business Central to use production mode or development mode. By default, KIE Server and all new projects in Business Central are in development mode. This mode provides features that facilitate your development experience, such as flexible project deployment policies, and features that optimize KIE Server performance during development, such as disabled duplicate GAV detection. Use development mode until your Red Hat Process Automation Manager environment is established and completely ready for production mode. For more information about configuring the environment mode or duplicate GAV detection, see the following resources:

- Chapter 8, Configuring the environment mode in KIE Server and Business Central
- Packaging and deploying a Red Hat Process Automation Manager project

Adapt KIE Server capabilities and extensions to your specific needs

The capabilities in KIE Server are determined by plug-in extensions that you can enable, disable, or further extend to meet your business needs. By default, KIE Server extensions are exposed through REST or JMS data transports and use predefined client APIs. You can extend existing KIE Server capabilities with additional REST endpoints, extend supported transport methods beyond REST or JMS, or extend functionality in the KIE Server client.

This flexibility in KIE Server functionality enables you to adapt your KIE Server instances to your business needs, instead of adapting your business needs to the default KIE Server capabilities.

For information about enabling, disabling, or extending KIE Server capabilities, see Chapter 21, KIE Server capabilities and extensions.
CHAPTER 23. ADDITIONAL RESOURCES

- Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.4
- Planning a Red Hat Process Automation Manager installation
- Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.4
- Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform 4 using Operators
- Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform 3 using templates
PART II. CONFIGURING BUSINESS CENTRAL SETTINGS AND PROPERTIES

As an administrator, you can customize the following on the admin Settings page:

- **Roles**: Set the home page, priority, and permissions of a role.

- **Groups**: Set the home page, priority, and permissions of a group as well as create and delete groups.

- **Users**: Create and delete users, add or remove groups and roles from users, and view user permissions.

- **Artifacts**: View M2 repository artifacts, upload artifacts, view, and download JAR files.

- **Data Sources**: Add, update, or delete data sources and database drivers.

- **Data Sets**: Create, modify, or delete data sets.

- **Projects**: View and edit project preferences such as file export properties, space properties, default values, and advanced GAV properties.

- **Artifact Repository**: Manage artifact repository properties.

- **Languages**: Set the Business Central language.

- **Process Administration**: Set the default pagination option in Business Central.

- **Process Designer**: Set diagram editor properties.

- **SSH Keys**: Add or delete SSH keys.

- **Custom Tasks Administration**: Enable or disable default service tasks and upload custom service tasks.

- **Dashbuilder Data Transfer**: Import and export Dashbuilder data as ZIP files in Business Central.

- **Profiles**: Set the workbench profile as Planner and Rules or Full.

- **Archetypes**: View, add, validate, set as default, and delete the archetypes. Used as a template when creating a new project in Business Central.

Prerequisites

- Red Hat JBoss Enterprise Application Platform 7.4.1 is installed. For more information, see Red Hat JBoss Enterprise Application Platform 7.4 Installation Guide.

- Red Hat Process Automation Manager is installed and running. For more information, see Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.4.

- You are logged in to Business Central with the admin user role.
CHAPTER 24. USER AND GROUP MANAGEMENT

Business Central defines three types of entities for security management: users, groups, and roles. You can assign permissions to both roles and groups. You can assign the following roles in Business Central:

- process-admin
- manager
- admin
- analyst
- rest-all
- developer
- rest-project
- user

**NOTE**

User roles in the application Role Registry have a role identifier, whereas user groups do not.

Use Business Central to create and manage as many users and groups as you require. A user must be assigned to at least one user-specific role to log in to Business Central. User privileges depend on permissions from the groups and roles that the user is a member of. Note that the role or group priority is considered if a user has several roles or groups assigned to it.

24.1. CREATING USERS

User privileges and settings are controlled by the roles assigned to a user and the groups that a user belongs to. You can create any number of users in Business Central.

**NOTE**

Do not create a user called unknown in process engine or KIE Server. The unknown user account is a reserved system name with superuser access. The unknown user account performs tasks related to the SLA violation listener when there are no users logged in.

**Procedure**

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Users.

2. Click New user, enter a user name, and then click Next.

3. To assign roles to the user, click the Roles tab, click Add Roles, select the desired roles, and click Add to selected roles.

4. Optional: To assign groups to the user, click the Groups tab, click Add to groups, select the desired groups, and click Add to selected groups.
5. Click **Create**.

6. Click **Yes** to set a password for the user, enter a desired password, and click **Change**.

**NOTE**
The user must have at least one role to access Business Central.

### 24.2. EDITING USERS

You can change the group and role of a user using the **Users** option on the Business Central **Settings** page. All user permissions are based on the group and role permissions of the user. You can view the user permissions from the **Permissions** tab.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Users**.

2. From the **All users** list, click the user you want to edit. The user details display in the right pane.

3. Click **Edit** to perform any of the following tasks:
   - To change the groups of a user, click the **Groups** tab, click **Add to groups**, select the groups you want the user to be part of, click **Add to selected groups**, and click **Save**.
   - To change the roles of a user, click the **Roles** tab, click **Add roles**, select the roles you want to assign to the user, click **Add to selected roles**, and click **Save**.
   - To view the user permissions, click the **Permissions** tab and expand the attributes.
   - To change the password, click **Change Password**, enter the new password, and click **Change**.
   - To delete the user, click **Delete** and then click **Yes** to confirm removal.

### 24.3. CREATING GROUPS

In Business Central, you can use groups to control permissions for a collection of users. You can create as many groups as you want but a group must have at least one user.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Groups**.

2. Click **New group**, enter a group name, and then click **Next**.

3. Select the users that you want to add to this group, and then click **Add selected users**. The newly created group is listed under **All groups**.

### 24.4. EDITING GROUPS
You can edit the attribute of a group such as home page, priority, and permissions according to your requirements. From the Groups option on the Business Central Settings page, you can modify or delete a group.

**Procedure**

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Groups.
2. From the All groups list, click the group that you want to edit. The user details display in the right pane.
3. Select the home page from the Home Page list.
4. Select the priority from the Priority list.
5. In the Permissions section, expand the resource attribute and change its permission.

   **NOTE**

   You can add exceptions to Pages, Editor, Spaces, and Projects permissions.

6. Click Save to apply the changes.
CHAPTER 25. SECURITY MANAGEMENT

Security management is the process of managing users, groups, and permissions. You can control access to Business Central resources and features from the Business Central Security management page.

Business Central defines three types of entities for security management: users, groups, and roles. You can assign permissions to both roles and groups. A user inherits permissions from the groups and roles that the user is a member of.

25.1. SECURITY MANAGEMENT PROVIDERS

In the context of security management, a realm restricts access to different application resources. Realms contain information about users, groups, roles, and permissions. A concrete user and group management service implementation for a specific realm is called a security management provider.

If the built-in security management providers do not meet the requirements of your application security realm, then you can build and register your own security management provider.

NOTE

If the security management provider is not installed, the user interface for managing the security realm is not available. After you install and configure a security management provider, the user and group management features are automatically enabled in the security management user interface.

Business Central includes the Red Hat JBoss EAP security management provider which supports realm types based on the contents of the application-users.properties or application-roles.properties property file.

25.1.1. Configuring the Red Hat JBoss EAP security management provider based on property files

You can build and register your own Red Hat JBoss EAP security management provider. To use the Red Hat JBoss EAP security management provider based on property files, complete the steps in this procedure.

Prerequisites

- Red Hat JBoss EAP is installed.

Procedure

1. To use an existing users or roles property file from the Red Hat JBoss EAP instance, include the following system properties in the EAP_HOME/standalone/configuration/application-users.properties and EAP_HOME/standalone/configuration/application-roles.properties files, as shown in the following example:

```
<property name="org.uberfire.ext.security.management.wildfly.properties.realm" value="ApplicationRealm"/>
<property name="org.uberfire.ext.security.management.wildfly.properties.users-file-path"
```
The following table provides a description and default value for these properties:

Table 25.1. Red Hat JBoss EAP security management provider based on property files

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.realm</td>
<td>The name of the realm. This property is not mandatory.</td>
<td>ApplicationRealm</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.users-file-path</td>
<td>The absolute file path for the users property file. This property is mandatory.</td>
<td>./standalone/configuration/application-users.properties</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.groups-file-path</td>
<td>The absolute file path for the groups property file. This property is mandatory.</td>
<td>./standalone/configuration/application-roles.properties</td>
</tr>
</tbody>
</table>

2. Create the `security-management.properties` file in the root directory of your application. For example, create the following file:

```xml
<src/main/resources/security-management.properties/>
```

3. Enter the following system property and security provider name as a value in the `security-management.properties` file:

```xml
<property name="org.uberfire.ext.security.management.api.userManagementServices" value="WildflyUserManagementService"/>
```

25.1.2. Configuring the Red Hat JBoss EAP security management provider based on property files and CLI mode

To use the Red Hat JBoss EAP security management provider based on property files and CLI mode, complete the steps in this procedure.

**Prerequisites**

- Red Hat JBoss EAP is installed.

**Procedure**

1. To use an existing users or roles property file from the Red Hat JBoss EAP instance, include the following system properties in the `EAP_HOME/standalone/configuration/application-users.properties` and `EAP_HOME/standalone/configuration/application-roles.properties` files, as shown in the following example:

```xml
<property name="org.uberfire.ext.security.management.wildfly.cli.host" value="localhost"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.port" value="9990"/>
```
The following table provides a description and default value for these properties:

### Table 25.2. Red Hat JBoss EAP security management provider based on property files and CLI mode

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.uberfire.ext.security.management.wildfly.cli.host</code></td>
<td>The native administration interface host.</td>
<td>localhost</td>
</tr>
<tr>
<td><code>org.uberfire.ext.security.management.wildfly.cli.port</code></td>
<td>The native administration interface port.</td>
<td>9990</td>
</tr>
<tr>
<td><code>org.uberfire.ext.security.management.wildfly.cli.user</code></td>
<td>The native administration interface username.</td>
<td>NA</td>
</tr>
<tr>
<td><code>org.uberfire.ext.security.management.wildfly.cli.password</code></td>
<td>The native administration interface user's password.</td>
<td>NA</td>
</tr>
<tr>
<td><code>org.uberfire.ext.security.management.wildfly.cli.realm</code></td>
<td>The realm used by the application's security context.</td>
<td>ApplicationRealm</td>
</tr>
</tbody>
</table>

2. Create the `security-management.properties` file in the root directory of your application. For example, create the following file:

```
<property name="org.uberfire.ext.security.management.wildfly.cli.user" value="<USERNAME>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.password" value="<USER_PWD>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.realm" value="ApplicationRealm"/>
```

3. Enter the following system property and security provider name as a value in the `security-management.properties` file:

```
<property name="org.uberfire.ext.security.management.api.userManagementServices" value="WildflyCLIUUserManagementService"/>
```

### 25.2. PERMISSIONS AND SETTINGS

A permission is an authorization granted to a user to perform actions related to a specific resource within the application. For example, a user can have following permissions:

- View a page.
- Save the project.
- View a repository.
- Delete a dashboard.

You can grant or deny a permission and a permission can be global or resource specific. You can use permissions to protect access to resources and customize features within the application.

### 25.2.1. Changing permissions for groups and roles in Business Central

In Business Central, you cannot change permissions for an individual user. However, you can change permissions for groups and roles. The changed permissions apply to users with the role or that belong to a group that you changed.

**NOTE**

Any changes that you make to roles or groups affect all of the users associated with that role or group.

**Prerequisites**

- You are logged in to Business Central with the *admin* user role.

**Procedure**

1. To access the Security management page in Business Central, select the *Admin* icon in the top-right corner of the screen.
2. Click *Roles*, *Groups*, or *Users* on the Business Central *Settings* page. The Security management page opens on the tab for the icon that you clicked.
3. From the list, click the role or group you want to edit. All details are displayed in the right panel.
4. Set the *Home Page* or *Priority* under the *Settings* section.
5. Set the Business Central, page, editor, space, and project permissions under the *Permissions* section.

**Figure 25.1. Setting the permissions**

![Admin settings](image)

6. Click the arrow next to a resource type to expand the resource type whose permissions you want to change.
7. Optional: To add an exception for a resource type, click **Add Exception** and then set the permissions as required.

**NOTE**
You cannot add an exception to the Business Central resource type.

8. Click **Save**.

### 25.2.2. Changing the Business Central home page

The home page is the page that appears after you log in to Business Central. By default, the home page is set to **Home**. You can specify a different home page for each role and group.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Roles** or **Groups**.

2. Select a role or group.

3. Select a page from the **Home Page** list.

4. Click **Save**.

**NOTE**
The role or group must have read access to a page before you can make it the home page.

### 25.2.3. Setting priorities

A user can have multiple roles and belong to multiple groups. The Priority setting determines the order of precedence of a role or group.

**Prerequisites**

- You are logged in to Business Central with the **admin** user role.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Roles** or **Groups**.

2. Select a role or group.

3. Select a priority from the **Priority** menu, and then click **Save**.

**NOTE**
If a user has a role or belongs to a group that has conflicting settings, the settings of the role or group with the highest priority applies.
CHAPTER 26. ARTIFACT MANAGEMENT

You can manage artifacts from the **Artifacts** page in Business Central. The artifact repository is a local Maven repository and there is only one Maven repository for each installation. Business Central recommends using Maven repository solutions like **Sonatype Nexus™**, **Apache Archiva™**, or **JFrog Artifactory™**.

The **Artifacts** page lists all the artifacts in the Maven repository. You can upload artifacts to the Maven repository.

**NOTE**

You can only upload JAR, KJAR, and **pom.xml** files to the **Artifacts** repository.

26.1. VIEWING AN ARTIFACT

You can view all the content of the local maven repository from the **Artifacts** page.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Artifacts**.
2. Click **Open** to view the artifact details.
3. Click **Ok** to go back to the **Artifacts** page.

26.2. DOWNLOADING AN ARTIFACT

You can download and save an artifact from Business Central repository to the local storage of a project.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Artifacts**.
2. Click **Download**.
3. Browse to the directory where you want to save the artifact.
4. Click **Save**.

26.3. UPLOADING AN ARTIFACT

You can upload an artifact from the local storage to a project in Business Central.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Artifacts**.
2. Click **Upload**.
3. Click **Choose File** and browse to the directory from where you want to upload the artifact.

4. Click **Upload**.

**NOTE**

If you are using a non-Maven artifact, first deploy the artifact to the Maven repository using the `mvn deploy` command and then refresh the artifact list in Business Central.
CHAPTER 27. DATA SOURCE AND DATABASE DRIVER MANAGEMENT

Business Central provides data source management features that enable you to define data sources for accessing a database. These data sources are then used by other Business Central components such as data sets. A database driver enables communication between a data source and the targeted database.

From the Data Source Authoring page you can add data sources and database drivers to Business Central.

NOTE

Business Central provides a default data source that can be used but cannot be edited or deleted.

27.1. ADDING A DATA SOURCE

You can add a new data source to Business Central from the Data Sources Authoring page.

Procedure

1. In Business Central, select the Admin icon in the upper-right corner of the screen and select Data Sources.
2. In the DataSource Explorer pane, click Add DataSource. The New data source window opens.
3. In the New data source window, enter the Name, Connection URL, User, Password, and Driver fields of the data source.
4. Click Test Connection to verify the connection to the database.
5. Click Finish.

27.2. EDITING A DATA SOURCE

You can edit the properties of a data source and also test its connection to the database in Business Central.

Procedure

1. In Business Central, select the Admin icon in the upper-right corner of the screen and select Data Sources.
2. In the DataSource Explorer pane, click the data source you want to edit.
3. In the Data Source Definition pane, make the necessary changes to the Name, Connection URL, User, Password, and the Driver fields.
4. Click Test Connection to verify the connection to the database.
5. Click Update.
6. Click Save.
27.3. DELETING A DATA SOURCE
You can delete an existing data source from the **DataSource Explorer** pane in Business Central.

**Procedure**
1. In Business Central, select the **Admin** icon in the upper-right corner of the screen and select **Data Sources**.
2. In the **DataSource Explorer** pane, click the data source you want to delete.
3. Click **Remove**.
4. Click **Delete** to confirm the deletion of the data source.

27.4. ADDING A DATABASE DRIVER
You can add a new database driver to Business Central.

**Procedure**
1. In Business Central, select the **Admin** icon in the upper-right corner of the screen and select **Data Sources**.
2. In the **DataSource Explorer** pane, click **Add Driver**. The **New driver** window opens.
3. In the **New driver** window, enter the **Name**, **Driver Class Name**, **Group Id**, **Artifact Id**, and **Version** fields of the database driver.
4. Click **Finish**.

27.5. EDITING A DATABASE DRIVER
You can edit the properties of a database driver from the **Driver Definition** pane.

**Procedure**
1. In Business Central, select the **Admin** icon in the upper-right corner of the screen and select **Data Sources**.
2. In the **DataSource Explorer** pane, select the driver you want to edit.
3. In the **Driver Definition** pane, make the necessary changes to the **Name**, **Driver Class Name**, **Group Id**, **Artifact Id**, and the **Version** fields.
4. Click **Update**.
5. Click **Yes**.

27.6. DELETING A DATABASE DRIVER
You can remove database drivers from the **Data Source Definition** pane of Business Central.

**Procedure**
Procedure

1. In Business Central, select the Admin icon in the upper-right corner of the screen and select Data Sources.

2. In the DataSource Explorer pane, select the driver you want to delete.

3. Click Remove.

4. Click Delete.
CHAPTER 28. DATA SETS AUTHORING

A data set is a collection of related sets of information and can be stored in a database, in a Microsoft Excel file, or in memory. A data set definition instructs Business Central methods to access, read, and parse a data set. Business Central does not store data. It enables you to define access to a data set regardless of where the data is stored.

For example, if data is stored in a database, a valid data set can contain the entire database or a subset of the database as a result of an SQL query. In both cases the data is used as input for the reporting components of Business Central which then displays the information.

To access a data set, you must create and register a data set definition. The data set definition specifies the location of the data set, options to access it, read it, and parse it, and the columns that it contains.

NOTE

The Data Sets page is visible only to users with the admin role.

28.1. ADDING DATA SETS

You can create a data set to fetch data from an external data source and use that data for the reporting components.

Procedure

1. In Business Central, go to Admin → Data Sets.
   The Data Sets page opens.

2. Click New Data Set and select one of the following provider types:
   - Bean: Generates a data set from a Java class
   - CSV: Generates a data set from a remote or local CSV file
   - SQL: Generates a data set from an ANSI-SQL compliant database
   - Elastic Search: Generates a data set from Elastic Search nodes
   - Prometheus: Generates a data set using the Prometheus query
   - Kafka: Generates a data set using metrics from Kafka broker, consumer, or producer

   NOTE

   You must configure KIE Server for Prometheus, Kafka, and Execution Server options.

3. Complete the Data Set Creation Wizard and click Test.

   NOTE

   The configuration steps differ based on the provider you choose.

4. Click Save.
28.2. EDITING DATA SETS

You can edit existing data sets to ensure that the data fetched to the reporting components is up-to-date.

Procedure

1. In Business Central, go to Admin → Data Sets. The Data Set Explorer page opens.
2. In the Data Set Explorer pane, search for the data set you want to edit, select the data set, and click Edit.
3. In the Data Set Editor pane, use the appropriate tab to edit the data as required. The tabs differ based on the data set provider type you chose. For example, the following changes are applicable for editing a CSV data provider:
   - **CSV Configuration**: Enables you to change the name of the data set definition, the source file, the separator, and other properties.
   - **Preview**: Enables you to preview the data. After you click Test in the CSV Configuration tab, the system executes the data set lookup call and if the data is available, a preview appears. Note that the Preview tab has two sub-tabs:
     - **Data columns**: Enables you to specify what columns are part of your data set definition.
     - **Filter**: Enables you to add a new filter.
   - **Advanced**: Enables you to manage the following configurations:
     - **Caching**: See Caching data for more information.
     - **Cache life-cycle**: Enables you to specify an interval of time after which a data set (or data) is refreshed. The Refresh on stale data feature refreshes the cached data when the back-end data changes.
4. After making the required changes, click Validate.
5. Click Save.

28.3. DATA REFRESH

The data refresh feature enables you to specify an interval of time after which a data set (or data) is refreshed. You can access the Data refresh every feature on the Advanced tab of the data set. The Refresh on stale data feature refreshes the cached data when the back-end data changes.

28.4. CACHING DATA

Business Central provides caching mechanisms for storing data sets and performing data operations using in-memory data. Caching data reduces network traffic, remote system payload, and processing time. To avoid performance issues, configure the cache settings in Business Central.

For any data lookup call that results in a data set, the caching method determines where the data lookup call is executed and where the resulting data set is stored. An example of a data lookup call would be all the mortgage applications whose locale parameter is set as "Urban".
Business Central data set functionality provides two cache levels:

- Client level
- Back-end level

You can set the **Client Cache** and **Backend Cache** settings on the **Advanced** tab of the data set.

**Client cache**
When the cache is turned on, the data set is cached in a web browser during the lookup operation and further lookup operations do not perform requests to the back-end. Data set operations like grouping, aggregations, filtering, and sorting are processed in the web browser. Enable client caching only if the data set size is small, for example, for data sets with less than 10 MB of data. For large data sets, browser issues such as slow performance or intermittent freezing can occur. Client caching reduces the number of back-end requests including requests to the storage system.

**Back-end cache**
When the cache is enabled, the decision engine caches the data set. This reduces the number of back-end requests to the remote storage system. All data set operations are performed in the decision engine using in-memory data. Enable back-end caching only if the data set size is not updated frequently and it can be stored and processed in memory. Using back-end caching is also useful in cases with low latency connectivity issues with the remote storage.

**NOTE**
Back-end cache settings are not always visible in the **Advanced** tab of the **Data Set Editor** because Java and CSV data providers rely on back-end caching (data set must be in the memory) in order to resolve any data lookup operation using the in-memory decision engine.
CHAPTER 29. ARCHETYPE MANAGEMENT

Business Central provides an archetype management feature that enables you to list, add, validate, set as default, and delete the archetypes. You can manage archetypes from the Archetypes page in Business Central. Archetypes are projects that are installed in Apache Maven repositories in which a template structure is set or can be created if required.

For the most up-to-date and detailed information about archetypes, see the Introduction to Archetypes page.

29.1. LISTING ARCHETYPES

The Archetypes page lists all the archetypes that are added in Business Central. This list provides the detailed information about Group ID, Artifact ID, Version, Created Date, Status, and Actions of an archetype.

Prerequisites

- You have created an archetype and listed it in the Business Central Settings from the maven repository.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Archetypes.
   In the Status column, green icon indicates it is a valid archetype, red icon indicates it is an invalid archetype whereas blue icon indicates the corresponding archetype is the default one for the new spaces.

29.2. ADDING AN ARCHETYPE

You can add a new archetype to Business Central.

Prerequisites

- You have installed an archetype in the Maven repository.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Archetypes.

2. Click Add Archetype.

3. In the Add Archetype panel, enter the GAV attributes in the Group ID, Artifact ID, and Version fields respectively.

4. Click Add.

Business Central validates the newly added archetype and make it available to be used as a template in all the spaces.

29.3. MANAGING ADDITIONAL FEATURES OF AN ARCHETYPE
You can delete, set a default, and validate the archetypes from the Archetypes page in Business Central.

Prerequisites

- You have created an archetype and listed in the Business Central Settings from the Maven repository.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Archetypes.

2. From the Actions column, click the icon on the right side of an archetype.
   - Select Delete from the drop-down menu to delete an archetype from the list.
   - Select Validate from the drop-down menu to validate whether the archetype is valid or not.

   **NOTE**
   When the Business Central is starting up, all the registered archetypes are automatically validated.

   - Select Set as default from the drop-down menu to set an archetype as a default for the new spaces.

29.4. CREATING A PROJECT USING ARCHETYPES

You can use archetypes to create a project in Business Central. When you create a project in Business Central, it is added to the Git repository that is connected to your Red Hat Process Automation Manager installation.

Prerequisites

- You have created an archetype and listed it in the Business Central Settings from the Maven repository.

- You have set an archetype as default in your space in the Business Central.

Procedure

1. In Business Central, go to Menu → Design → Projects.

2. Select or create the space into which you want to add a new project from an archetype template.

3. Click Add Project.

4. Type the project name and description in the Name and Description fields respectively.

5. Click Configure Advanced Options.

6. Select the Based on template check box.
7. Select the archetype from drop-down options if required. The default archetype is selected that is already set in the space.

8. Click Add.

The Assets view of the project opens based on the selected archetype template.

**29.5. MANAGING ARCHETYPES USING SPACE SETTINGS IN BUSINESS CENTRAL**

When you add the archetypes to Business Central, they can be used as templates in all the spaces. You can manage all the archetypes from the Settings tab which is available in the space. This tab is visible only to users with the admin role.

**Prerequisites**

- You have installed an archetype in the Maven repository.
- You have created an archetype and listed it in the Business Central Settings from the Maven repository.

**Procedure**

1. In Business Central, go to Menu → Design → Projects.

2. Select or create the space into which you want to manage the archetypes. The default space is MySpace.

3. Click Settings.

4. To include or exclude the archetypes in the space, select the Include check box.

5. From the Actions column, click the icon on the right side of an archetype and select Set as default from the drop-down menu to set an archetype as a default for the space.

6. Click Save.
In Business Central, a project is a part of your space and stores the related assets. You can add multiple projects in a space.

For example, an organization includes various departments, such as HR, Payroll, Engineering, and R&D. You can map each department to a space in Business Central, along with adding respective projects.

You can customize the project settings in Business Central. Also, you can create a new project or clone projects from an existing Git repository.

**Procedure**

1. In Business Central, select the **Admin** icon in the upper-right corner and select **Projects**.
2. In the **Project Preferences** panel, select the preference you want to modify. The project preferences include:
   - **Project Importing**: This preference consists of the following property:
     - Select the **Allow multiple projects to be imported on cluster** to import multiple projects on a cluster.
   - **File exporting**: This preference consists of the following properties:

     | Field                | Description                                                                 |
     |----------------------|-----------------------------------------------------------------------------|
     | PDF orientation      | Determines whether the PDF orientation is portrait or landscape.            |
     | PDF units            | Determines whether the PDF unit is **PT**, **MM**, **CN**, or **IN**.        |
     | PDF page format      | Determines whether the PDF page format is **A[0-10]**, **B[0-10]**, or **C[0-10]**. |

   - **Spaces**: This preference consists of the following properties:

<pre><code> | Field   | Description                                                                 |
 |---------|-----------------------------------------------------------------------------|
 | Name    | The default name of the space that is created automatically if none exists.  |
 | Owner   | The default owner of the space that is created automatically if none exists.  |
 | Group ID| The default group ID of the space that is created automatically if none exists. |
</code></pre>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias (in singular)</td>
<td>Determines the customized alias (singular) of the space.</td>
</tr>
<tr>
<td>Alias (in plural)</td>
<td>Determines the customized alias (plural) of the space.</td>
</tr>
</tbody>
</table>

- **Default values**: This preference consists of the following properties:

  **Table 30.3. Default values properties**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>The default version number of a project when creating projects.</td>
</tr>
<tr>
<td>Description</td>
<td>The default description of a project when creating projects.</td>
</tr>
<tr>
<td>Branch</td>
<td>The default branch to be used when using a Git repository.</td>
</tr>
<tr>
<td>Assets Per Page</td>
<td>Used to customize the number of assets per page in the project. The default value is 15.</td>
</tr>
</tbody>
</table>

- **Advanced GAV preferences**: This preference consists of the following properties:

  **Table 30.4. Advanced GAV preference properties**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable GAV conflict check?</td>
<td>Determines whether to enable or disable the GAV conflict check. Disabling this checkbox enables the projects to contain the same GAV (group ID, artifact, and version).</td>
</tr>
<tr>
<td>Allow child GAV edition?</td>
<td>Determines whether to allow child or subprojects to contain GAV edition.</td>
</tr>
</tbody>
</table>

**NOTE**

Duplicate GAV detection is disabled for projects in the development mode. To enable duplicate GAV detection for a project in Business Central, go to project **Settings → General Settings → Version** and toggle the **Development Mode** option to **OFF** (if applicable).

3. Click **Save**.

128
CHAPTER 31. CUSTOMIZING ARTIFACT REPOSITORY PROPERTIES

In some cases, projects need to resolve external dependencies to build domain model JAR files. A repository contains the needed artifacts and has the following features:

- The repository is a Maven repository.
- All snapshots are time stamped.
- Assets are stored mostly in the local hard drive.

By default, the artifact repository is in `$WORKING_DIRECTORY/repositories/kie`.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Artifact Repository. The Artifact Repository page opens.
2. Make selections and enter information in the Properties section.
3. Click Save.
You can change the language on the Business Central Settings page. Business Central supports the following languages:

- English
- Spanish
- French
- Japanese

The default language is English.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Languages. The Language Selector window opens.

2. Select the desired language from the Language list.

3. Click Ok.
CHAPTER 33. CUSTOMIZING PROCESS ADMINISTRATION

You can customize the default pagination option in Business Central by editing the Default items per page property on the Process Administration page.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Process Administration.

2. From the Properties section, update the Default items per page property and click Save.

NOTE

You can specify 10, 20, 50, or 100 items to display on each page.
CHAPTER 34. CUSTOMIZING THE PROCESS DESIGNER

You can customize the process designer in Business Central by editing the properties of the diagram editor on the Business Central Settings page.

Procedure

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Process Designer.

2. In the Properties section, update any of the following properties:
   - Select the Auto hide category panel check box to automatically hide a category toolbar panel.
   - In the Drawing area width field, enter an integer value between 2800 and 5600 to set the width of the drawing area.
   - In the Drawing area height field, enter an integer value between 1400 and 2800 to set the height of the drawing area.
   - Select the Enable HiDPI check box if you are using a high resolution display and are seeing blurry text and objects. This option is disabled by default.

3. Click Save.
CHAPTER 35. SSH KEYS

Business Central provides an SSH keystore service to enable user SSH authentication. Business Central provides a configurable default SSH keystore, extensible APIs (for custom implementations), and support for multiple SSH public key formats.

You can access the SSH Keys option on the Business Central Settings page to register your SSH public keys.

35.1. DEFAULT SSH KEYSTORE

The default SSH keystore included with Business Central provides a file-based storage mechanism to store a user’s public keys. By default, Business Central uses the *.security folder as the root directory. However, you can also use a custom storage path by setting the value of the appformer.ssh.keys.storage.folder system property to point to a different folder.

The SSH public keys are stored in the {securityFolderPath}/pkeys/{userName}/ folder structure.

Each SSH public key consists of the following files, located in the storage folder:

- `{keyId}.pub`: This file contains the SSH public key content. As the file name determines the logic key ID on the system, ensure that the file name is not modified during run time.
  
  For example:

  ```
  ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAABAQDmak4Wu23RZ6XmN94bOsqecZxuTa4RRhhQm
  HmTZjMB7HM57/90u/B/gBGhsPEn1AXLOnP95JTV/MPQ8yRm2C2W9A7CzN5+z5yyL3W01Y
  Zy3kzslk77CjUl/JhrcfQL3b2sPG5jy5E5/nyC/swSytwcW/T/PE7aXTS9H6cHIKUdYPz194SHoBx
  WRJK7Plj9+eLB+hmd2zvbVa1eszA8y2kchH6NxfH5I5iR2r5ce6DP0imC1jMoC6ZDifbVsZxL9F
  TMwFndNnmTLJeBtv9nAbnAvlWiiS0Vokdj1s3GxBeZYAcKbcsK9sJzu5ptk5dxGsGZ8vlnaqIN
  60aOQ7b7tcomczYywviGO9gRX8sGsVrw39gsDIGYP2A4bRr7ecHnlnG1b0HCcH+a5+QCDk
  4Hbzu1umHmPA2Lg9c3WGM2qedvQdVJKuxS3mlwYOsL40aXPs68580PvFJUmIVsFznF50djPnws
  MxJZEf1HdTXgZD1Bh54ogZ17czyUNflkNkE6odyJDbTHjdpQd0cKUqnuTVqmxBzhX31yF4VcsMe
  ADcif2Zbw4zn4LZnC/GwonYIq5+G93zJpF0KPhme8c2XuPucXf795lxyJ8SB/Aw/PJAhEtm0y
  0s0l114eWqxsDxBOgN+iV0czrVMssHJEJb4o0FLi71HhOW56l/imD9w== userName
  ```

- `{keyId}.pub.meta`: This file contains the key metadata in JSON format. A new metadata file is dynamically generated if a key has no metadata.
  
  For example:

  ```
  {
  "name":"Key",
  "creationDate":"Oct 10, 2018 10:10:50 PM",
  "lastTimeUsed":"Oct 11, 2018 12:11:23 PM"
  }
  ```

35.2. CUSTOM SSH KEYSTORE

You can extend and customize the default SSH keystore according to your requirements. Use the appformer.ssh.keystore system property to specify the Java class name of the SSH service to use. If this property is not defined or it contains an incorrect value, then the default SSH keystore is loaded.
NOTE

To create a custom implementation of the SSH keystore, your Java class must implement the org.uberfire.ssh.service.backend.keystore.SSHKeyStore class defined in the uberfire-ssh-api module.

35.3. CREATING AN SSH KEY

Before you can add or register SSH keys to Business Central, you must generate an SSH key on your system.

Procedure

1. Open a command terminal on your system.

2. Run the ssh-keygen command to create the SSH key as shown in the following example, where <user_login> is your user name:

   ```
   ssh-keygen -t rsa -b 4096 -C "<user_login>"
   ```

   **NOTE**

   The SSH key formats supported by Business Central keystore are ssh-rsa, ssh-dss, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

3. When prompted, press Enter and accept the default key file location as shown in the following example, where <user_login> is your user name:

   ```
   Enter a file in which to save the key (/home/<user_login>/.ssh/id_rsa): [Press enter]
   ```

4. At the command prompt, enter and confirm the passphrase:

   ```
   Enter passphrase (empty for no passphrase): [Type a passphrase]
   Enter same passphrase again: [Type passphrase again]
   ```

5. Start the ssh-agent:

   ```
   eval "$\$(ssh-agent -s)"
   Agent pid <any-number-here>
   ```

6. Add the new SSH private key to the ssh-agent. If you have used a different key name, replace id_rsa in that code:

   ```
   ssh-add ~/.ssh/id_rsa
   ```

35.4. REGISTERING YOUR SSH PUBLIC KEY WITH THE SSH KEystore

You must register your newly created SSH public key with the Business Central keystore.

Procedure

1. Open a command terminal on your system.
2. Run the `cat` command as shown in the following example, where `id_rsa` is your key name:

```
    cat ~/.ssh/id_rsa.pub
```

3. Copy the contents of your SSH public key.

4. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **SSH Keys**.

5. On the **SSH Keys** page, click **Add SSH Key**.

6. In the **Add SSH Key** window, enter a name in the **Name** field and copy the contents of the SSH public key to the **Key** field.

   **NOTE**

   The **Name** and the **Key** fields are mandatory.

7. Click **Add SSH Key** to register the key.

### 35.5. DELETING AN SSH KEY

You can delete an SSH key from Business Central by from the **SSH Keys** page.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **SSH Keys**.

2. On the **SSH Keys** page, click the delete icon of the SSH key you want to delete.

3. Click **Delete SSH Key** to confirm the deletion.
CHAPTER 36. MANAGING CUSTOM TASKS IN BUSINESS CENTRAL

Custom tasks (work items) are tasks that can run custom logic. You can customize and reuse custom tasks across multiple business processes or across all projects in Business Central. You can also add custom elements in the designer palette, including name, icon, sub-category, input and output parameters, and documentation. Red Hat Process Automation Manager provides a set of custom tasks within the custom task repository in Business Central. You can enable or disable the default custom tasks and upload custom tasks into Business Central to implement the tasks in the relevant processes.

**NOTE**

Red Hat Process Automation Manager includes a limited set of supported custom tasks. Custom tasks that are not included in Red Hat Process Automation Manager are not supported.

**Procedure**

1. In Business Central, click in the upper-right corner and select Custom Tasks Administration.
   This page lists the custom task installation settings and available custom tasks for processes in projects throughout Business Central. The custom tasks that you enable on this page become available in the project-level settings where you can then install each custom task to be used in processes. The way in which the custom tasks are installed in a project is determined by the global settings that you enable or disable under Settings on this Custom Tasks Administration page.

2. Under Settings, enable or disable each setting to determine how the available custom tasks are implemented when a user installs them at the project level.
   The following custom task settings are available:
   - **Install as Maven artifact** Uploads the custom task JAR file to the Maven repository that is configured with Business Central, if the file is not already present.
   - **Install custom task dependencies into project** Adds any custom task dependencies to the pom.xml file of the project where the task is installed.
   - **Use version range when installing custom task into project** Uses a version range instead of a fixed version of a custom task that is added as a project dependency. Example: [7.16,) instead of 7.16.0.Final

3. Enable or disable (set to ON or OFF) any available custom tasks as needed. Custom tasks that you enable are displayed in project-level settings for all projects in Business Central.
4. To add a custom task, click **Add Custom Task**, browse to the relevant JAR file, and click the **Upload** icon. If a class implements a **WorkItemHandler**, you can replace annotations with a **.wid** file by adding the file to Business Central separately.

5. Optional: To remove a custom task, click **remove** on the row of the custom task you want to remove and click **Ok** to confirm removal.

6. After you configure all required custom tasks, navigate to a project in Business Central and go to the project **Settings → Custom Tasks** page to view the available custom tasks that you enabled.

7. For each custom task, click **Install** to make the task available to the processes in that project or click **Uninstall** to exclude the task from the processes in the project.

8. If you are prompted for additional information when you install a custom task, enter the required information and click **Install** again.
   The required parameters for the custom task depend on the type of task. For example, rule and decision tasks require artifact GAV information (Group ID, Artifact ID, Version), email tasks require host and port access information, and REST tasks require API credentials. Other custom tasks might not require any additional parameters.
9. Click **Save**.

10. Return to the project page, select or add a business process in the project, and in the process designer palette, select the **Custom Tasks** option to view the available custom tasks that you enabled and installed:
Figure 36.3. Access installed custom tasks in process designer
CHAPTER 37. LDAP CONNECTION

Business Central provides a dedicated **UserGroupCallback** implementation for LDAP servers with Red Hat Process Automation Manager to enable the user task service to retrieve information on users, groups, and roles directly from an LDAP service.

You can configure the following LDAP **UserGroupCallback** implementation properties:

**Table 37.1. LDAP UserGroupCallback properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldap.bind.user</td>
<td>User name for connecting to the LDAP server.</td>
</tr>
<tr>
<td></td>
<td>This property is optional if it is not specified and the LDAP server accepts anonymous access.</td>
</tr>
<tr>
<td>ldap.bind.pwd</td>
<td>Password for connecting to the LDAP server.</td>
</tr>
<tr>
<td></td>
<td>This property is optional if it is not specified and the LDAP server accepts anonymous access.</td>
</tr>
<tr>
<td>ldap.user.ctx</td>
<td>Context in LDAP with user information.</td>
</tr>
<tr>
<td>ldap.role.ctx</td>
<td>Context in LDAP with group and role.</td>
</tr>
<tr>
<td>ldap.user.roles.ctx</td>
<td>Context in LDAP with user group and role membership information.</td>
</tr>
<tr>
<td></td>
<td>This property is optional if it is not specified and the ldap.role.ctx property is used instead.</td>
</tr>
<tr>
<td>ldap.user.filter</td>
<td>Filter for searching user information.</td>
</tr>
<tr>
<td></td>
<td>This property usually contains substitution keys {0} that are replaced with parameters.</td>
</tr>
<tr>
<td>ldap.role.filter</td>
<td>Filter for searching group and role information.</td>
</tr>
<tr>
<td></td>
<td>This property usually contains substitution keys {0} that are replaced with parameters.</td>
</tr>
<tr>
<td>ldap.user.roles.filter</td>
<td>Filter for searching user group and role membership information.</td>
</tr>
<tr>
<td></td>
<td>This property usually contains substitution keys {0} that are replaced with parameters.</td>
</tr>
<tr>
<td>ldap.user.attr.id</td>
<td>Attribute name of the user ID in LDAP.</td>
</tr>
<tr>
<td></td>
<td>This property is optional if it is not specified and the uid property is used instead.</td>
</tr>
</tbody>
</table>
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ldap.roles.attr.id</code></td>
<td>Attribute name of the group and role ID in LDAP. This property is optional if it is not specified and the <code>cn</code> property is used instead.</td>
</tr>
<tr>
<td><code>ldap.user.id.dn</code></td>
<td>User ID in a DN, instructs the callback to query for user DN before searching for roles. This is optional and is <code>false</code> by default.</td>
</tr>
<tr>
<td><code>java.naming.factory.initial</code></td>
<td>Initial context factory class name; is <code>com.sun.jndi.ldap.LdapCtxFactory</code> by default.</td>
</tr>
<tr>
<td><code>java.naming.security.authentication</code></td>
<td>Authentication type where the possible values are <code>none</code>, <code>simple</code>, and <code>strong</code>. This is <code>simple</code> by default.</td>
</tr>
<tr>
<td><code>java.naming.security.protocol</code></td>
<td>Security protocol to be used, for example, <code>ssl</code>.</td>
</tr>
<tr>
<td><code>java.naming.provider.url</code></td>
<td>LDAP url (by default <code>ldap://localhost:389</code>; if the protocol is set to <code>ssl</code> then <code>ldap://localhost:636</code>)</td>
</tr>
</tbody>
</table>

### 37.1. LDAP USERGROUPCALLBACK IMPLEMENTATION

You can use the LDAP `UserGroupCallback` implementation by configuring the respective LDAP properties in one of the following ways:

- Programmatically: Build a properties object with the respective `LDAPUserGroupCallbackImpl` properties and create `LDAPUserGroupCallbackImpl` using the same properties object as its parameter.
  
  For example:

  ```java
  import org.kie.api.PropertiesConfiguration;
  import org.kie.api.task.UserGroupCallback;
  ...
  Properties properties = new Properties();
  properties.setProperty(LDAPUserGroupCallbackImpl.USER_CTX, "ou=People,dc=my-domain,dc=com");
  properties.setProperty(LDAPUserGroupCallbackImpl.ROLE_CTX, "ou=Roles,dc=my-domain,dc=com");
  properties.setProperty(LDAPUserGroupCallbackImpl.USER_ROLES_CTX, "ou=Roles,dc=my-domain,dc=com");
  properties.setProperty(LDAPUserGroupCallbackImpl.USER_FILTER, "(uid={0})");
  properties.setProperty(LDAPUserGroupCallbackImpl.ROLE_FILTER, "(cn={0})");
  properties.setProperty(LDAPUserGroupCallbackImpl.USER_ROLES_FILTER, "(member={0})");
  
  UserGroupCallback ldapUserGroupCallback = new LDAPUserGroupCallbackImpl(properties);
  ```
LDAPUserGroupCallbackImpl(properties);
UserGroupCallbackManager.getInstance().setCallback(ldapUserGroupCallback);

- Declaratively: Create the `jbpm.usergroup.callback.properties` file in the root of your application or specify the file location as a system property.
  For example:

  ```
  -Djbpm.usergroup.callback.properties=FILE_LOCATION_ON_CLASSPATH
  ```

  Ensure that you register the LDAP callback when starting the user task server.
  For example:

  ```
  #ldap.bind.user=
  #ldap.bind.pwd=
  ldap.user.ctx=ou=People,dc=my-domain,dc=com
  ldap.role.ctx=ou=Roles,dc=my-domain,dc=com
  ldap.user.roles.ctx=ou=Roles,dc=my-domain,dc=com
  ldap.user.filter=(uid={0})
  ldap.role.filter=(cn={0})
  ldap.user.roles.filter=(member={0})
  #ldap.user.attr.id=
  #ldap.roles.attr.id=
  ```

Additional resources

- Roles and users
- Red Hat Single Sign-On Server Administration Guide
- Defining LDAP login domain
- LDAP login module
- LDAPExtended login module
- AdvancedLDAP login module
- AdvancedAdLDAP login module
- LDAP connectivity options
- LDAPUsers login module
CHAPTER 38. DATABASE CONNECTION

Business Central provides a dedicated UserGroupCallback implementation for database server with Red Hat Process Automation Manager to enable the user task service. The user task service helps in retrieving information on users and groups (roles) directly from databases.

You can configure the following database UserGroupCallback implementation properties:

Table 38.1. Database UserGroupCallback properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db.ds.jndi.name</td>
<td>JNDI name of the data source used for connections</td>
</tr>
<tr>
<td>db.user.query</td>
<td>Verifies the user existence</td>
</tr>
<tr>
<td>db.user.roles.query</td>
<td>Collects the groups for a given user</td>
</tr>
<tr>
<td>db.roles.query</td>
<td>Verifies the group existence</td>
</tr>
</tbody>
</table>

38.1. DATABASE USERGROUPOCALLBACK IMPLEMENTATION

In database UserGroupCallback implementation, you must create the required database. You can use this implementation by configuring the respective database properties in one of the following ways:

- **Programmatically**: Build a properties object with the respective DBUserGroupCallbackImpl properties and create DBUserGroupCallbackImpl using the same properties object as its parameter.
  
  For example:

  ```java
  import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.DS_JNDI_NAME;
  import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.PRINCIPAL_QUERY;
  import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.ROLES_QUERY;
  import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.USER_ROLES_QUERY;
  ...
  props = new Properties();
  props.setProperty(DS_JNDI_NAME, "jdbc/jbpm-ds");
  props.setProperty(PRINCIPAL_QUERY, "select userId from Users where userId = ?>");
  props.setProperty(ROLES_QUERY, "select groupId from UserGroups where groupId = ?>");
  props.setProperty(USER_ROLES_QUERY, "select groupId from UserGroups where userId = ?>");
  callback = new DBUserGroupCallbackImpl(props);
  ```

- **Declaratively**: Create the jbpm.usergroup.callback.properties file in the root of your application or specify the file location as a system property.
  
  For example:

  ```bash
  -Djbpm.usergroup.callback.properties=FILE_LOCATION_ON_CLASSPATH
  ```
Ensure that you register the database callback when starting the user task server.

For example:

```java
System.setProperty("jbpm.usergroup.callback.properties", "/jbpm.usergroup.callback.db.properties");
callback = new DBUserGroupCallbackImpl(true);
...
db.ds.jndi.name = jdbc/jbpm-ds
db.user.query = select userId from Users where userId = ?
db.roles.query = select groupId from UserGroups where groupId = ?
db.user.roles.query = select groupId from UserGroups where userId = ?
```

**Additional resources**

- *Roles and users*
CHAPTER 39. CONFIGURING MAVEN USING SETTINGS.XML FILE

Java application development uses the Apache Maven build automation tool to build and manage software projects. Maven uses Project Object Model (POM) configuration XML files to define both, the project properties and the build process.

Maven uses repositories to store Java libraries, plug-ins, and other build artifacts. Repositories can be either local or remote. A local repository is a download of artifacts from a remote repository cached on a local machine. A remote repository is any other repository accessed using common protocols, such as http:// when located on an HTTP server, or file:// when located on a file server. The default repository is the public remote Maven 2 Central Repository. Configuration of Maven is performed by modifying the settings.xml file. You can either configure global Maven settings in the M2_HOME/conf/settings.xml file, or user-level settings in the USER_HOME/.m2/settings.xml file.

Additional resources

- Configuring an external Maven repository for Business Central and KIE Server
- Packaging and deploying a Red Hat Process Automation Manager project in Maven
- Maven settings and repositories for Red Hat Process Automation Manager
- System integration with Maven
- Welcome to Apache Maven
- Apache Maven Project - Introduction to Repositories
- Apache Maven Parent POMs Reference
CHAPTER 40. GAV CHECK MANAGEMENT

In Business Central, projects are identified by the Group ID, Artifact ID, and Version (GAV) Maven naming convention. GAV values differentiate projects and project versions as well as identify dependencies with particular projects.

By default, Business Central detects duplicate GAVs. This feature can be disabled by users with the admin role.

40.1. CONFIGURING GAV CHECKS AND CHILD GAV EDITION

This procedure describes how to configure GAV checks in Business Central.

Procedure

1. In Business Central, go to Menu → Design → Projects and click the project name.
2. In the project window, click the Settings tab.
3. In the General Settings tab, perform any of the following tasks:
   - To enable other projects to have the same GAV, select the Disable GAV conflict check check box.
   - To enable child projects to have GAV edition, select the Allow child GAV edition check box.
4. Click Save.

   NOTE
   You can click Reset to undo all changes.

5. Click Save to confirm the changes.

   NOTE
   Duplicate GAV detection is disabled for projects in Development Mode. To enable duplicate GAV detection in Business Central, go to project Settings → General Settings → Version and toggle the Development Mode option to OFF (if applicable).

40.2. CONFIGURING GAV CHECKS FOR ALL PROJECTS

This procedure describes how to configure GAV checks for all projects in Business Central. You can also disable GAV checks at system startup.

Procedures

1. In Business Central, select the Admin icon in the top-right corner of the screen and select Projects. The Projects window opens.
2. In the Advanced GAV preferences tab, perform any of the following tasks:
   - To enable other projects to have the same GAV, select the Disable GAV conflict check check box.
check box.

- To enable child projects to have GAV edition, select the **Allow child GAV edition** check box.

3. Click **Save**.

**NOTE**

You can also disable the duplicate GAV detection feature by setting the `org.guvnor.project.gav.check.disabled` system property to `true` for Business Central at system startup:

```bash
$ ~/EAP_HOME/bin/standalone.sh -c standalone-full.xml -Dorg.guvnor.project.gav.check.disabled=true
```
CHAPTER 41. CONFIGURING THE ENVIRONMENT MODE IN KIE SERVER AND BUSINESS CENTRAL

You can set KIE Server to run in production mode or in development mode. Development mode provides a flexible deployment policy that enables you to update existing deployment units (KIE containers) while maintaining active process instances for small changes. It also enables you to reset the deployment unit state before updating active process instances for larger changes. Production mode is optimal for production environments, where each deployment creates a new deployment unit.

In a development environment, you can click Deploy in Business Central to deploy the built KJAR file to a KIE Server without stopping any running instances (if applicable), or click Redeploy to deploy the built KJAR file and replace all instances. The next time you deploy or redeploy the built KJAR, the previous deployment unit (KIE container) is automatically updated in the same target KIE Server.

In a production environment, the Redeploy option in Business Central is disabled and you can click only Deploy to deploy the built KJAR file to a new deployment unit (KIE container) on a KIE Server.

Procedure

1. To configure the KIE Server environment mode, set the org.kie.server.mode system property to org.kie.server.mode=development or org.kie.server.mode=production.

2. To configure the deployment behavior for a project in Business Central, go to project Settings → General Settings → Version and toggle the Development Mode option.

NOTE

By default, KIE Server and all new projects in Business Central are in development mode.

You cannot deploy a project with Development Mode turned on or with a manually added SNAPSHOT version suffix to a KIE Server that is in production mode.
CHAPTER 42. GIT HOOKS AND REMOTE GIT REPOSITORY INTEGRATION

Git hooks are bash scripts that execute before or after Git events such as `git commit` or `git push`. In Business Central, you can use Git hooks to configure repositories to trigger specified actions every time events happen. For more information about Git hooks, see *Customizing Git Hooks*.

You can integrate remote Git repositories with Business Central by using post-commit Git hooks. This enables you to automate content replication between Business Central and remote repositories. For example, you can implement a real-time backup strategy where changes you make to your Business Central projects are replicated to your remote Git repositories.

NOTE

Business Central only supports post-commit Git hooks.

A post-commit Git hook executes after every commit as a sync operation. Business Central waits for the post-commit bash to complete and no other write operation occurs in the repository.

42.1. CREATING POST-COMMIT GIT HOOKS

You can create a post-commit Git hook bash script file that executes code contained in that file or execute code from a different file such as a Java program.

Procedure

1. Create a post-commit Git hook file:

   ```
   $ touch post-commit
   ```

2. Set the permissions of the post-commit file to 755:

   ```
   $ chmod 755 post-commit
   ```

3. Add `#!/bin/bash` and any required code to the post-commit file, for example:

   - To push all changes to a remote repository:
     ```
     #!/bin/bash
     git push origin +master
     ```
   - To log a message:
     ```
     #!/bin/bash
     echo 'Hello World'
     ```
   - To execute code of another file:
     ```
     #!/bin/bash
     java -jar _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks/git-push.jar
     ```
NOTE

To use post-commit Git hooks that execute Java code, you must use the following Java libraries:

- **JGit**: Used to interact with internal Business Central Git repositories.
- **GitHub API for Java**: Used to communicate with GitHub.

For more information about post-commit Git hook and Java code examples, see *Business Central post-commit Git Hooks Integration*.

### 42.2. IMPORTING REMOTE GIT REPOSITORIES

You can import a remote Git repository into Business Central and configure a post-commit Git hook to automatically push changes to that remote repository.

**Prerequisites**

- Red Hat Process Automation Manager is installed in a Red Hat JBoss EAP 7.4 server instance.
- Red Hat Process Automation Manager projects exist in an external Git repository.
- Read access credentials for the external Git repository.
- (For Windows) Cygwin is installed with the Git package added during installation and the path to the Cygwin `/bin` folder is added to your environment `PATH` variable. For example, `C:\cygwin64\bin`. For more information about Cygwin installation, see *Installing and Updating Cygwin Packages*.

**Procedure**

1. In Business Central, go to **Menu → Projects**.
2. Select or create the space that you want to import the Git projects into.
3. Click on the right side of the screen and select **Import Project**.
4. In the **Import Project** window, enter the URL of your Git repository, for example, `https://github.com/USERNAME/REPOSITORY_NAME.git`, and the credentials for the Git repository.
5. Click **Import**. The project is added to the Business Central Git repository and is then available in the space.
IMPORTANT

Use the HTTPS or Git protocol instead of a SCP-style SSH URL. Business Central does not support the basic SSH URL and an error appears if you use this URL.

You must have your public ssh key configured in your Git provider.

The Git repository must be a KJAR project, containing only a single KJAR that is compatible with the Red Hat Process Automation Manager version. The KJAR content must be in the root of the repository.

6. In a command terminal, navigate to the hooks folder located in the repository Git folder of the project. For example:

   $ cd _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks

7. Create a post-commit file that pushes changes to the remote Git repository. For example:

   #!/bin/sh
   git push origin +master

   For more information about creating post-commit Git hooks, see Section 42.1, “Creating post-commit Git hooks”.

8. Optional: To check that the configuration was successful, create a guided rule in Business Central:


   b. On the Create new Guided Rule page, enter the required information.

   c. Click Ok.

      Business Central automatically pushes all changes to the remote repository.

Additional resources

- Customizing Git - Git Hooks

42.3. CONFIGURING GIT HOOKS FOR EXISTING REMOTE GIT PROJECT REPOSITORIES

If you have an existing remote Git repository project you can create a post-commit Git hook in a remote Git repository of that existing project and integrate the remote Git repository with Business Central.

Prerequisites

- Red Hat Process Automation Manager is installed in a Red Hat JBoss EAP 7.4 server instance.

- Red Hat Process Automation Manager projects exist in an external Git repository.

- Read access credentials for the external Git repository.

- (For Windows operating system) Cygwin is installed with the Git package added during
installation and the path to the Cygwin /bin folder is added to your environment PATH variable. For example, C:\cygwin64\bin. For more information about Cygwin installation, see Installing and Updating Cygwin Packages.

Procedure

1. In a command terminal, navigate to the hooks folder located in the repository Git folder of the project. For example:

   $ cd _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks

2. Create a post-commit file that pushes changes to the remote Git repository. For example:

   #!/bin/sh
   git push origin +master

   For more information about creating post-commit Git hooks, see Section 42.1, “Creating post-commit Git hooks”.

3. Optional: To check that the configuration was successful, create a guided rule in Business Central:
   b. On the Create new Guided Rule page, enter the required information.
   c. Click Ok.
      Business Central automatically pushes all changes to the remote repository.

42.4. CONFIGURING GIT HOOKS AS A SYSTEM PROPERTY FOR BUSINESS CENTRAL

If you do not have an existing Git repository project or if you want to apply post-commit Git hooks to a large number of project repositories you can specify a directory containing a hook file for the value of the org.uberfire.nio.git.hooks system property. This directory is copied to the Git repositories.

NOTE

If you specify the org.uberfire.nio.git.hooks system property, all Business Central internal repositories and project repositories use the post-commit Git hook. You should only use fully qualified paths in your script.

Prerequisites

- Red Hat Process Automation Manager is installed in a Red Hat JBoss EAP 7.4 server instance.

- (For Windows operating system) Cygwin is installed with the Git package added during installation and the path to the Cygwin /bin folder is added to your environment PATH variable. For example, C:\cygwin64\bin. For more information about Cygwin installation, see Installing and Updating Cygwin Packages.

Procedure

1. Create a post-commit Git hook in a directory on your local system.
For more information about creating post-commit Git hooks, see Section 42.1, “Creating post-commit Git hooks”.

2. To specify the directory with the hook file for the value of the `org.uberfire.nio.git.hooks` system property, do one of the following tasks:

   • Add the `org.uberfire.nio.git.hooks` system property to the `standalone.xml` file. For example:

   ```xml
   <system-properties>
   <property name="org.uberfire.nio.git.hooks" value="_EAP_HOME_/hooks">
   </property>
   ...
   </system-properties>
   
   • Use the `-Dorg.uberfire.nio.git.hooks` environment variable when executing Business Central. For example:

   ```bash
   $ ./standalone.sh -c standalone-full.xml -Dorg.uberfire.nio.git.hooks=_EAP_HOME_/hooks
   ```

   The post-commit Git hook is copied to all Business Central internal repositories and project repositories.

Additional resources

• Customizing Git - Git Hooks

42.5. INTEGRATING REMOTE GIT REPOSITORIES

In the following example, you use a post-commit Git hook and Java code to integrate Business Central with a remote Git repository. For the Java code example, see Business Central post-commit Git Hooks Integration. The example provides the following functionality:

• Automatic generation of the template `.gitremote` configuration file

• Validation of the `.gitremote` configuration file for required parameters

• Patterns defined in the ignore parameter of the `.gitremote` file are ignored by Git

• Message and notification output to users

• Support for GitLab and GitHub token authentication

• Support for GitLab group and subgroup project creation

• Support for GitHub organization repository creation

Prerequisites

• Red Hat Process Automation Manager is installed in a Red Hat JBoss EAP 7.4 server instance.

• Java Development Kit (JDK) 8 is installed.
• Maven is installed.

Procedure

1. In a terminal window, clone the GitHub repository to your system:

   $ git clone https://github.com/kiegroup/bc-git-integration-push.git

2. Navigate to the cloned repository:

   $ cd bc-git-integration-push

3. Execute a Maven clean install:

   $ mvn clean install

4. Create a /hooks folder in your EAP_HOME directory:

   $ mkdir -p _EAP_HOME_/hooks/

5. Copy the git-push-2.1-SNAPSHOT.jar to the EAP_HOME/hooks/ folder:

   $ cp bc-git-integration-push/target/git-push-2.1-SNAPSHOT.jar _EAP_HOME_/hooks/

6. Optional: To create a template .gitremote configuration file, run git-push-2.1-SNAPSHOT.jar:

   $ java -jar git-push-2.1-SNAPSHOT.jar

Example template .gitremote configuration file

```java
#This is an auto generated template empty property file
provider=GIT_HUB
login=
password=
token=
remoteGitUrl=https://api.github.com/
useSSH=false
ignore=.*demo.*, test.*
githubOrg=OrgName
gitlabGroup=Group/subgroup
```

7. Modify the .gitremote configuration file parameters.

Table 42.1. Example .gitremote parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>The Git provider. Only two values are accepted: GIT_HUB and GIT_LAB. Required</td>
</tr>
<tr>
<td>login</td>
<td>The user name for the Git provider. Required</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>password</td>
<td>A plain text password. Not required if a <strong>token</strong> is provided.</td>
</tr>
<tr>
<td>token</td>
<td>A generated token to replace the <strong>username</strong> and <strong>password</strong> based unsecured connection. Note: If this is not set a warning is displayed that you are using an unsecured connection. Not required if a <strong>password</strong> is provided. Note: GitLab only supports token authentication.</td>
</tr>
<tr>
<td>remoteGitUrl</td>
<td>A public provider URL or a locally hosted enterprise for any provider. Required. Note: The public GitHub URL should be the API URL. For example, api.github.com.</td>
</tr>
<tr>
<td>useSSH</td>
<td>Boolean to allow the SSH protocol to push changes to the remote repository. Optional. Default = false. Note: This parameter uses the local ~/.ssh/ directory to obtain the SSH configuration.</td>
</tr>
<tr>
<td>ignore</td>
<td>A comma separated regular expressions to ignore project names that match any of these expressions. Optional.</td>
</tr>
<tr>
<td>githubOrg</td>
<td>Defines the repository organization if GitHub is used as the provider. Optional.</td>
</tr>
<tr>
<td>gitlabGroup</td>
<td>Defines the repository group and subgroup if GitLab is used as the provider Optional.</td>
</tr>
</tbody>
</table>

8. Create a **post-commit** Git hook file in **EAP_HOME/hooks**:

```
$ touch post-commit
```

9. Set the permissions of the **post-commit** file to **755**:

```
$ chmod 755 post-commit
```

10. Add `#!/bin/bash` and code to execute `git-push-2.1-SNAPSHOT.jar` to the **post-commit** file:

```
$ echo "#!/bin/bash
java -jar $APP_SERVER_HOME/hooks/git-push-2.1-SNAPSHOT.jar" > hooks/post-commit
```

11. Start Business Central with the `-Dorg.uberfire.nio.git.hooks` environment variable set. For example:

```
$ ./standalone.sh -c standalone-full.xml -Dorg.uberfire.nio.git.hooks=_EAP_HOME_/hooks
```
NOTE

To use post-commit Git hooks that execute Java code, you must use the following Java libraries:

- **JGit**: Used to interact with internal Business Central Git repositories.
- **GitHub API for Java**: Used to communicate with GitHub.

For more information about post-commit Git hook and Java code examples, see *Business Central post-commit Git Hooks Integration*.

### 42.6. GIT HOOK EXIT CODES

When a Git hook exits an integer value is returned which determines the status of the Git hook execution. This integer value is known as a Git hook exit code. The execution status can be a success (1), warning (2 to 30) or error (31 to 255).

### 42.7. CUSTOMIZING GIT HOOK NOTIFICATIONS

Business Central provides a mechanism that enables users to receive customized Git hook notifications based on the hook exit codes.

To enable the notification mechanism you must create a `*.properties` file containing the custom messages and then specify the path to that file as the value of the `appformer.git.hooks.bundle` system property.

**Procedure**

1. Create the `*.properties` file and add a line for each exit code with a corresponding message in the following format:
   
   `<exit_code>=<display_message>`

   The `<exit_code>` is the Git hook exit code and the `<display_message>` is the custom message that is displayed to a user.

   For example:

   ```
   0=Success! All working as expected.
   1=Warning! Please check the logs and advise your admin.
   .
   .
   31=Error! Please advise your admin immediately.
   ```

   **NOTE**

   It is not necessary to define all the possible exit codes in the `*.properties` file. Notifications appear only for the exit codes defined in the `*.properties` file.
The notification service only supports the ISO 8859-1 (LATIN 1) character set in the properties file. If you want to use extended characters, please use their escaped Unicode character code sequences.

2. To enable Git hook notifications, specify the path to the file as the value of the appformer.git.hooks.bundle system property.

See the following example of a standalone.xml file with the setting that points to a Messages.properties file:

```xml
<system-properties>
  <property name="appformer.git.hooks.bundle" value="/opt/jboss-as/git-hooks-messages/Messages.properties">
  </property>
  ...
</system-properties>
```

### 42.7.1. Git hook notifications in Business Central

You can view Git hook notifications in Business Central. There are three Git hook exit code notification types.

#### Table 42.2. Git hook UI notification types

<table>
<thead>
<tr>
<th>Exit code</th>
<th>Customized message</th>
<th>UI notification color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success! All working as expected.</td>
<td>Green</td>
</tr>
<tr>
<td>1 to 30</td>
<td>Warning! Please check the logs and advise your admin.</td>
<td>Orange</td>
</tr>
<tr>
<td>31 to 255</td>
<td>Error! Please advise your admin immediately.</td>
<td>Red</td>
</tr>
</tbody>
</table>

**IMPORTANT**

UNIX machines only support error codes between 0 (success) to 255 (error), any exit code outside of this range will end up being converted into a different code which may cause showing a wrong notification message.

Windows machines don’t have this limitation and support a wide range of exit codes.

### 42.7.2. Git hook notification internationalization support

You can internationalize notification messages by placing additional properties files in the same path as the original properties file specified as the appformer.git.hooks.bundle system property.

The name of the different localized files must be `<filename>_<lang>.properties`, where the `<filename>` is the same as the original. For example, where the system property points to Messages.properties, you can create Messages_en.properties for English, Messages_fr.properties for French, or Messages_it.properties for Italian.
The notification service will choose the properties file based on the user’s language, if there are no available translations for that language it will use the entries from the original `Messages.properties` file.
CHAPTER 43. ROLE-BASED ACCESS CONTROL FOR BRANCHES IN BUSINESS CENTRAL

Business Central provides the option for users to restrict the access for a target branch for a specific collaborator type. The security check uses both the Security Management screen and contributors sources to grant or deny permissions to spaces and projects. For example, if a user has the security permission to update a project and has write permission on that branch, based on the contributor type, then they are able to create new assets.

43.1. CUSTOMIZING ROLE-BASED BRANCH ACCESS

You can customize contributor role permissions for each branch of a project in Business Central. For example, you can set Read, Write, Delete, and Deploy access for each role assigned to a branch.

Procedure

1. In Business Central, go to Menu → Design → Projects.

2. If needed, add a new contributor:
   a. Click the project name and then click the Contributors tab.
   b. Click Add Contributor.
   c. Enter user name in the text field.
   d. Select the Contributor role type from the drop-down list.
   e. Click Ok.

3. Customize role-based branch access for the relevant contributor:
   a. Click Settings → Branch Management.
   b. Select the branch name from the drop-down list.
   c. In the Role Access section, select or deselect the permissions check boxes to specify role-based branch access for each available role type.
   d. Click Save and click Save again to confirm your changes.
CHAPTER 44. VIEWING PROCESS INSTANCE LOGS

You can view all the process events of an instance from its Logs tab. The instance logs list all the current and previous process states. Business Central has two types of logs for process instances, Business and Technical logs.

Procedure

1. In Business Central, go to Menu → Manage → Process Instances.

2. On the Manage Process Instances page, click the process instance whose log you want to view.

3. Select the Logs tab:
   - Click Business to view the business events log.
   - Click Technical to view the technical events log.
   - Click Asc or Desc to change the order of the log files.
CHAPTER 45. BUSINESS CENTRAL SYSTEM PROPERTIES

The Business Central system properties listed in this section are passed to `standalone*.xml` files.

**Git directory**

Use the following properties to set the location and name for the Business Central Git directory:

- `org.uberfire.nio.git.dir`: Location of the Business Central Git directory.
- `org.uberfire.nio.git.ketch`: Enables or disables Git ketch.
- `org.uberfire.nio.git.hooks`: Location of the Git hooks directory.

**Git over HTTP**

Use the following properties to configure access to the Git repository over HTTP:

- `org.uberfire.nio.git.proxy.ssh.over.http`: Specifies whether SSH should use an HTTP proxy. Default value: `false`.
- `http.proxyHost`: Defines the host name of the HTTP proxy. Default value: `null`.
- `http.proxyPort`: Defines the host port (integer value) of the HTTP proxy. Default value: `null`.
- `http.proxyUser`: Defines the user name of the HTTP proxy.
- `http.proxyPassword`: Defines the user password of the HTTP proxy.
- `org.uberfire.nio.git.http.enabled`: Enables or disables the HTTP daemon. Default value: `true`.
- `org.uberfire.nio.git.http.host`: If the HTTP daemon is enabled, it uses this property as the host identifier. This is an informative property that is used to display how to access the Git repository over HTTP. The HTTP still relies on the servlet container. Default value: `localhost`.
- `org.uberfire.nio.git.http.hostname`: If the HTTP daemon is enabled, it uses this property as the host name identifier. This is an informative property that is used to display how to access the Git repository over HTTP. The HTTP still relies on the servlet container. Default value: `localhost`.
- `org.uberfire.nio.git.http.port`: If the HTTP daemon is enabled, it uses this property as the port number. This is an informative property that is used to display how to access the Git repository over HTTP. The HTTP still relies on the servlet container. Default value: `8080`.

**Git over HTTPS**

Use the following properties to configure access to the Git repository over HTTPS:

- `org.uberfire.nio.git.proxy.ssh.over.https`: Specifies whether SSH uses an HTTPS proxy. Default value: `false`.
- `https.proxyHost`: Defines the host name of the HTTPS proxy. Default value: `null`.
- `https.proxyPort`: Defines the host port (integer value) of the HTTPS proxy. Default value: `null`. 
• **https.proxyUser**: Defines the user name of the HTTPS proxy.

• **https.proxyPassword**: Defines the user password of the HTTPS proxy.

• **user.dir**: Location of the user directory.

• **org.uberfire.nio.git.https.enabled**: Enables or disables the HTTPS daemon. Default value: false

• **org.uberfire.nio.git.https.host**: If the HTTPS daemon is enabled, it uses this property as the host identifier. This is an informative property that is used to display how to access the Git repository over HTTPS. The HTTPS still relies on the servlet container. Default value: localhost.

• **org.uberfire.nio.git.https.hostname**: If the HTTPS daemon is enabled, it uses this property as the host name identifier. This is an informative property that is used to display how to access the Git repository over HTTPS. The HTTPS still relies on the servlet container. Default value: localhost.

• **org.uberfire.nio.git.https.port**: If the HTTPS daemon is enabled, it uses this property as the port number. This is an informative property that is used to display how to access the Git repository over HTTPS. The HTTPS still relies on the servlet container. Default value: 8080.

**JGit**

• **org.uberfire.nio.jgit.cache.instances**: Defines the JGit cache size.

• **org.uberfire.nio.jgit.cache.overflow.cleanup.size**: Defines the JGit cache overflow cleanup size.

• **org.uberfire.nio.jgit.remove.eldest.iterations**: Enables or disables whether to remove eldest JGit iterations.

• **org.uberfire.nio.jgit.cache.evict.threshold.duration**: Defines the JGit evict threshold duration.

• **org.uberfire.nio.jgit.cache.evict.threshold.time.unit**: Defines the JGit evict threshold time unit.

**Git daemon**

Use the following properties to enable and configure the Git daemon:

• **org.uberfire.nio.git.daemon.enabled**: Enables or disables the Git daemon. Default value: true.

• **org.uberfire.nio.git.daemon.host**: If the Git daemon is enabled, it uses this property as the local host identifier. Default value: localhost.

• **org.uberfire.nio.git.daemon.hostname**: If the Git daemon is enabled, it uses this property as the local host name identifier. Default value: localhost

• **org.uberfire.nio.git.daemon.port**: If the Git daemon is enabled, it uses this property as the port number. Default value: 9418.

• **org.uberfire.nio.git.http.sslVerify**: Enables or disables SSL certificate checking for Git repositories. Default value: true.
NOTE

If the default or assigned port is already in use, a new port is automatically selected. Ensure that the ports are available and check the log for more information.

Git SSH

Use the following properties to enable and configure the Git SSH daemon:

- **org.uberfire.nio.git.ssh.enabled**: Enables or disables the SSH daemon. Default value: **true**.
- **org.uberfire.nio.git.ssh.host**: If the SSH daemon enabled, it uses this property as the local host identifier. Default value: **localhost**.
- **org.uberfire.nio.git.ssh.hostname**: If the SSH daemon is enabled, it uses this property as local host name identifier. Default value: **localhost**.
- **org.uberfire.nio.git.ssh.port**: If the SSH daemon is enabled, it uses this property as the port number. Default value: **8001**.

NOTE

If the default or assigned port is already in use, a new port is automatically selected. Ensure that the ports are available and check the log for more information.

- **org.uberfire.nio.git.ssh.cert.dir**: Location of the .security directory where local certificates are stored. Default value: Working directory.
- **org.uberfire.nio.git.ssh.idle.timeout**: Sets the SSH idle timeout.
- **org.uberfire.nio.git.ssh.passphrase**: Pass phrase used to access the public key store of your operating system when cloning git repositories with SCP style URLs. Example: git@github.com:user/repository.git.
- **org.uberfire.nio.git.ssh.algorithm**: Algorithm used by SSH. Default value: **RSA**.
- **org.uberfire.nio.git.gc.limit**: Sets the GC limit.
- **org.uberfire.nio.git.ssh.ciphers**: A comma-separated string of ciphers. The available ciphers are **aes128-ctr, aes192-ctr, aes256-ctr, arcfour128, arcfour256, aes192-cbc, aes256-cbc**. If the property is not used, all available ciphers are loaded.
- **org.uberfire.nio.git.ssh.macs**: A comma-separated string of message authentication codes (MACs). The available MACs are **hmac-md5, hmac-md5-96, hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512**. If the property is not used, all available MACs are loaded.

NOTE

If you plan to use RSA or any algorithm other than DSA, make sure you set up your application server to use the Bouncy Castle JCE library.

KIE Server nodes and Process Automation Manager controller
Use the following properties to configure the connections with the KIE Server nodes from the Process Automation Manager controller:

- **org.kie.server.controller**: The URL is used to connect to the Process Automation Manager controller. For example, `ws://localhost:8080/business-central/websocket/controller`.

- **org.kie.server.user**: User name used to connect to the KIE Server nodes from the Process Automation Manager controller. This property is only required when using this Business Central installation as a Process Automation Manager controller.

- **org.kie.server.pwd**: Password used to connect to the KIE Server nodes from the Process Automation Manager controller. This property is only required when using this Business Central installation as a Process Automation Manager controller.

**Maven and miscellaneous**

Use the following properties to configure Maven and other miscellaneous functions:

- **kie.maven.offline.force**: Forces Maven to behave as if offline. If true, disables online dependency resolution. Default value: `false`.

  **NOTE**

  Use this property for Business Central only. If you share a runtime environment with any other component, isolate the configuration and apply it only to Business Central.

- **org.uberfire.gzip.enable**: Enables or disables Gzip compression on the `GzipFilter` compression filter. Default value: `true`.

- **org.kie.workbench.profile**: Selects the Business Central profile. Possible values are `FULL` or `PLANNER_AND_RULES`. A prefix `FULL_` sets the profile and hides the profile preferences from the administrator preferences. Default value: `FULL`.

- **org.appformer.m2repo.url**: Business Central uses the default location of the Maven repository when looking for dependencies. It directs to the Maven repository inside Business Central, for example, `http://localhost:8080/business-central/maven2`. Set this property before starting Business Central. Default value: File path to the inner `m2` repository.

- **appformer.ssh.keystore**: Defines the custom SSH keystore to be used with Business Central by specifying a class name. If the property is not available, the default SSH keystore is used.

- **appformer.ssh.keys.storage.folder**: When using the default SSH keystore, this property defines the storage folder for the user’s SSH public keys. If the property is not available, the keys are stored in the Business Central `.security` folder.

- **appformer.experimental.features**: Enables the experimental features framework. Default value: `false`.

- **org.kie.demo**: Enables an external clone of a demo application from GitHub.

- **org.uberfire.metadata.index.dir**: Place where the Lucene `.index` directory is stored. Default value: Working directory.

- **org.uberfire.ldap.regex.role_mapper**: Regex pattern used to map LDAP principal names to the application role name. Note that the variable role must be a part of the pattern as the
application role name substitutes the variable role when matching a principle value and role name.

- **org.uberfire.sys.repo.monitor.disabled**: Disables the configuration monitor. Do not disable unless you are sure. Default value: `false`.

- **org.uberfire.secure.key**: Password used by password encryption. Default value: org.uberfire.admin.

- **org.uberfire.secure.alg**: Crypto algorithm used by password encryption. Default value: PBEWithMD5AndDES.

- **org.uberfire.domain**: Security-domain name used by uberfire. Default value: ApplicationRealm.

- **org.guvnor.m2repo.dir**: Place where the Maven repository folder is stored. Default value: `<working-directory>/repositories/kie`.

- **org.guvnor.project.gav.check.disabled**: Disables group ID, artifact ID, and version (GAV) checks. Default value: `false`.

- **org.kie.build.disable-project-explorer**: Disables automatic build of a selected project in Project Explorer. Default value: `false`.

- **org.kie.builder.cache.size**: Defines the cache size of the project builder. Default value: 20.

- **org.kie.library.assets_per_page**: You can customize the number of assets per page in the project screen. Default value: 15.

- **org.kie.verification.disable-dtable-realtime-verification**: Disables the real-time validation and verification of decision tables. Default value: `false`.

### Process Automation Manager controller

Use the following properties to configure how to connect to the Process Automation Manager controller:

- **org.kie.workbench.controller**: The URL used to connect to the Process Automation Manager controller, for example, `ws://localhost:8080/kie-server-controller/websocket/controller`.

- **org.kie.workbench.controller.user**: The Process Automation Manager controller user. Default value: kieserver.

- **org.kie.workbench.controller.pwd**: The Process Automation Manager controller password. Default value: kieserver1!

- **org.kie.workbench.controller.token**: The token string used to connect to the Process Automation Manager controller.

### Java Cryptography Extension KeyStore (JCEKS)

Use the following properties to configure JCEKS:

- **kie.keystore.keyStoreURL**: The URL used to load a Java Cryptography Extension KeyStore (JCEKS). For example, `file:///home/kie/keystores/keystore.jceks`.

- **kie.keystore.keyStorePwd**: The password used for the JCEKS.
• **kie.keystore.key.ctrl.alias**: The alias of the key for the default REST Process Automation Manager controller.

• **kie.keystore.key.ctrl.pwd**: The password of the alias for the default REST Process Automation Manager controller.

**Rendering**

Use the following properties to switch between Business Central and KIE Server rendered forms:

• **org.jbpm.wb.forms.renderer.ext**: Switches the form rendering between Business Central and KIE Server. By default, the form rendering is performed by Business Central. Default value: **false**.

• **org.jbpm.wb.forms.renderer.name**: Enables you to switch between Business Central and KIE Server rendered forms. Default value: **workbench**.
CHAPTER 46. PERFORMANCE TUNING CONSIDERATIONS WITH BUSINESS CENTRAL

The following key concepts or suggested practices can help you optimize Business Central configuration and Red Hat Process Automation Manager performance. These concepts are summarized in this section as a convenience and are explained in more detail in the cross-referenced documentation, where applicable. This section will expand or change as needed with new releases of Red Hat Process Automation Manager.

Ensure that development mode is enabled during development

You can set KIE Server or specific projects in Business Central to use production mode or development mode. By default, KIE Server and all new projects in Business Central are in development mode. This mode provides features that facilitate your development experience, such as flexible project deployment policies, and features that optimize KIE Server performance during development, such as disabled duplicate GAV detection. Use development mode until your Red Hat Process Automation Manager environment is established and completely ready for production mode. For more information about configuring the environment mode or duplicate GAV detection, see the following resources:

- Chapter 41, Configuring the environment mode in KIE Server and Business Central
- Packaging and deploying a Red Hat Process Automation Manager project

Disable verification and validation of complex guided decision tables

The decision table verification and validation feature of Business Central is enabled by default. This feature helps you validate your guided decision tables, but with complex guided decision tables, this feature can hinder decision engine performance. You can disable this feature by setting the org.kie.verification.disable-dtable-realtime-verification system property value to true. For more information about guided decision table validation, see Designing a decision service using guided decision tables.

Disable automatic builds if you have many large projects

In Business Central, when you navigate between projects in the Project Explorer side panel, the selected project is built automatically so that the Alerts window is updated to show any build errors for the project. If you have large projects or frequently switch between many projects that are under active development, this feature can hinder Business Central and decision engine performance. To disable automatic project builds, set the org.kie.build.disable-project-explorer system property to true.
PART III. USING STANDALONE PERSPECTIVES IN BUSINESS CENTRAL

As a business rules developer, you can embed standalone perspectives from Business Central in your web application and then use them to edit rules, processes, decision tables, and other assets.

Prerequisites

- Business Central is deployed and is running on a web/application server.
- You are logged in to Business Central.
CHAPTER 47. STANDALONE PERSPECTIVES IN BUSINESS CENTRAL

Business Central provides specialized editors for authoring assets based on the asset’s format. Business Central has a feature that enables you to use these editors individually. This feature is known as the standalone perspective mode of the editor or simply the **standalone perspectives**.

As a business rules developer, you can embed a standalone perspective in your web application and then use it to edit rules, processes, decision tables, and other assets. After embedding a perspective you can edit an asset in your own application without switching to Business Central. You can use this feature to customize your web application. In addition to standalone perspectives you can also embed standalone custom pages (dashboards) in your applications.

You can access a standalone perspective by using a specific web address in a browser with the **standalone** and **perspective** parameters. A standalone perspective’s web address may also contain additional parameters.
CHAPTER 48. USING THE STANDALONE LIBRARY PERSPECTIVE

You can use the library perspective of Business Central to select a project you want to edit. You can also perform all the authoring functions on the selected project.

The standalone library perspective can be used in two ways, with and without using the `header=UberfireBreadcrumbsContainer` parameter. The difference is that the address with the `header` parameter will display a breadcrumb trail on top of the library perspective. Using this link you can create additional `Spaces` for your projects.

Procedure

1. Log in to Business Central.

2. In a web browser, enter the appropriate web address:

   a. For accessing the standalone library perspective **without** the `header` parameter
      

      The standalone library perspective without the breadcrumb trail opens in the browser.

   b. For accessing the standalone library perspective **with** the `header` parameter
      

      The standalone library perspective with the breadcrumb trail opens in the browser.
CHAPTER 49. USING THE STANDALONE EDITOR PERSPECTIVE

You can use the standalone editor perspective of Business Central to access an asset's specific editor. By using this perspective, you can open an asset's editor and can modify the asset as required.

The web address for accessing an asset's standalone editor perspective contains the standalone and path parameters. The path parameter must contain the complete path to the asset and the web address can end with #StandaloneEditorPerspective string. Also, by varying the path parameter, you can access a specific asset's editor in the standalone mode.

Procedure

1. Log in to Business Central.

2. In a web browser, enter the appropriate web address as required, for example,
   a. For editing a process:
      
      http://localhost:8080/business-central/kie-wb.jsp?
      standalone&path=default://master@MySpace/Shop/src/main/resources/com/purchase
      .bpmn#StandaloneEditorPerspective
 
      The Process Designer opens in the standalone mode.

   b. For editing a form:
      
      http://localhost:8080/business-central/kie-wb.jsp?
      standalone&path=default://master@MySpace/Mortgage_Process/src/main/resources/
      ApplicationMortgage.frm#StandaloneEditorPerspective
 
      The Form Modeler opens in the standalone mode.
CHAPTER 50. USING THE STANDALONE CONTENT MANAGER PERSPECTIVE

By using the standalone content manager perspective in your application, you can create and edit your application’s content and its navigation menus.

Procedure

1. Log in to Business Central.

2. In a web browser, enter the following web address in the address bar, http://localhost:8080/business-central/kie-wb.jsp?
   standalone=true&perspective=ContentManagerPerspective

   The standalone content manager perspective opens in the browser.
Apart from standalone perspectives, you can also embed custom pages, also known as dashboards, in your application. For accessing the custom pages from your application, provide the name of the custom page as the value of the `perspective` parameter. Note that the `perspective` parameter is case-sensitive.

**Procedure**

1. Log in to Business Central.

2. In a web browser, enter the custom page’s web address in the address bar, for example, 

   ```plaintext
   ```

   The standalone custom page opens in the browser. Replace the value, `CustomPageName`, with the name of the custom page you want to use in the standalone mode.
PART IV. CREATING CUSTOM PAGES IN BUSINESS CENTRAL

As a business analyst or business rules developer, you can use the Page Editor in Business Central to create static pages and dynamic dashboards to display specific information about your project. A dashboard is a collection of pages that contain at least one dynamic reporting component. You can define a data set to feed the reporting components of a dashboard. You can export dashboards to the standalone Dashbuilder Runtime dashboard viewer on Red Hat JBoss EAP or the Dashbuilder Standalone dashboard viewer on Red Hat OpenShift Container Platform.

Prerequisites

- You are logged in to Business Central as a user with permission to edit pages.
A dashboard is a collection of Business Central pages that contain at least one reporting component. Dashboards often contain data sets, navigation trees, and permissions.

There are four stages in the creation of a custom dashboard:

- **Data set authoring**: Define a data set for accessing the data and displaying it through the pages. For more information, see [Adding data sets](#).
- **Page authoring**: Create the dashboard pages. For more information, see [Creating pages](#).
- **Publication**: In this stage, navigation between pages are defined when you create the custom navigation trees or modify the existing default one ([Workbench](#) tree). For more information, see [Creating a navigation tree](#) or [Editing a navigation tree](#).
- **Security management**: In this stage, role and group permissions are set which defines the privileges that are granted to a user when the user is working on Business Central. For more information, see [Security management](#).

**Additional resources**

To migrate dashboards from a previous version of Business Central, use the Dashbuilder Data Transfer feature described in [Section 57.1, “Exporting Business Central dashboard data”](#).
CHAPTER 53. DASHBUILDER RUNTIME AND DASHBUILDER STANDALONE

Dashbuilder Runtime and Dashbuilder Standalone are add-ons that you can use to view dashboards created in and exported from Business Central. This is useful for reviewing business metrics in environments that do not have Business Central. Dashbuilder Runtime is available to install on Red Hat JBoss EAP. You can deploy Dashbuilder Standalone on Red Hat OpenShift Container Platform.

Navigation between the pages of a dashboard in Dashbuilder Runtime and Dashbuilder Standalone is identical to navigation in the Business Central instance where the dashboard was created. If a page belongs to a group, that group is imported to Dashbuilder Runtime or Dashbuilder Standalone as well as the page. If a page is imported to Dashbuilder Runtime or Dashbuilder Standalone but not used in navigation, then the page is added to the Runtime Dashboards menu group. If no navigation is exported then all pages are added to the Runtime Dashboards menu group.

53.1. INSTALLING DASHBUILDER RUNTIME ON RED HAT JBOSS EAP

To install Dashbuilder Runtime, download the Dashbuilder Runtime WAR and create a user with the admin role.

Prerequisites

- You have a Red Hat JBoss EAP installation.
- You have created and exported a dashboard in Business Central. For more information about exporting Dashbuilder data, see the “Exporting and importing Dashbuilder data” section in the Configuring Business Central settings and properties guide.

Procedure

1. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
   - Product: Process Automation Manager
   - Version: 7.12


3. Navigate to the directory that contains the extracted files and extract the rhpam-7.12.0-dashbuilder-runtime.zip file.

4. Copy the contents of the dashbuilder-runtime.zip file that you extracted into the `<EAP_HOME>/standalone/deployments` folder where `<EAP_HOME>` is the Red Hat JBoss EAP home directory that contains your Red Hat JBoss EAP installation.

5. In the Red Hat JBoss EAP home directory, enter the following command to create a user with the admin role and specify a password. In the following example, replace `<USERNAME>` and `<PASSWORD>` with the user name and password of your choice.

```
$ ./bin/jboss-cli.sh --commands="embed-server --std-out=echo,/subsystem=elytron/filesystem-realm=ApplicationRealm:add-identity(identity=<USERNAME>),/subsystem=elytron/filesystem-realm=ApplicationRealm:set-
```
6. In a terminal application, navigate to \texttt{EAP\_HOME}/bin.

7. Enter the following command to start Red Hat JBoss EAP:
   - On Linux or UNIX-based systems:
     \[
     $ ./standalone.sh -c standalone-full.xml
     \]
   - On Windows:
     \[
     standalone.bat -c standalone-full.xml
     \]

8. In a web browser, open the URL \texttt{http://localhost:8080}.

9. Log in using the credentials of the user that you created for Dashbuilder Runtime.

10. When prompted, upload a dashboard that you exported from Business Central. Dashbuilder Runtime uses that dashboard until it is restarted.

### 53.1.1. Dashbuilder Runtime system properties

You can use system properties to customize Dashbuilder Runtime.

**Dashboards Path**

When a dashboard is uploaded it is stored in the filesystem. The path where it is stored is controlled by the system property \texttt{dashbuilder.import.base.dir}. The default is \texttt{/tmp/dashbuilder}.

The system property is the root path for any dashboard model. For example, if there are multiple files on this path, the file can be imported by accessing Dashbuilder Runtime and passing a query parameter import with the name of the file that should be loaded. For example, if you want to load the \texttt{sales_dashboard}, execute \texttt{runtime_host?import=sales_dashboard} and Dashbuilder Runtime will try to load the file \texttt{/tmp/dashbuilder/sales_dashboard.zip}.

**Static Dashboard**

If you want the runtime instance to load a specific dashboard, you can change the system property \texttt{dashbuilder.runtime.import}. Setting the property to a local file path will cause that specific dashboard to be loaded during Runtime startup.

**Controlling upload size**

Application servers control POST request size by default. You can control the allowable size of uploaded dashboards by using the system property \texttt{dashbuilder.runtime.upload.size}. The size should be in KB and by default the value is 96kb, meaning that if someone tries to upload a file larger than 96kb then an error will be displayed and the dashboard won’t be installed.

**Default pages in Dashbuilder Runtime**

Dashboards that are imported in the Dashbuilder Runtime contain a default page. The following list provides a summary of updates of the Dashbuilder Runtime default page:

- When an imported dashboard contains only one page, then it is used as the default page.
- If a page is named as \texttt{index} then it is used as the default page.
In other cases, the generic home page of the Dashbuilder Runtime is used.

Loading external dashboards

A Dashboard that is located at an accessible URL can be accessed by Dashbuilder Runtime. You can access the URL by passing the URL with the import query parameter such as `runtime_host?import=http://filesHost/sales_dashboard.zip`.

**NOTE**

For security reasons this option is disabled by default. You can enable it by setting the system property `dashbuilder.runtime.allowExternal` as true.

53.2. DEPLOYING DASHBUILDER STANDALONE ON RED HAT OPENSHIFT CONTAINER PLATFORM

You can use Dashbuilder Standalone to view dashboards in OpenShift that were created in and exported from Business Central. This is useful for reviewing business metrics in environments that do not have Business Central. Use the Dashbuilder Standalone operator to deploy Dashbuilder Standalone on Red Hat OpenShift Container Platform separately from other services.

**Prerequisites**

- Dashbuilder Standalone is available in the OpenShift registry.
- You have prepared your OpenShift environment as described in [Deploying a Red Hat Process Automation Manager environment on Red Hat OpenShift Container Platform 4 using Operators](#).
- You have created and exported a dashboard in Business Central.

**Procedure**

1. On the Operator **Installation** page, enter a name for your application in the **Application name** field.

2. In the **Environment** field, enter a name for your environment, for example `rhpam-standalone-dashbuilder`.

3. Click **Next**.


5. On the **Components** page, select **Dashbuilder** from the **Components** list.

6. To add a KIE Server data set, complete the following tasks:

   **NOTE**

   You can add additional KIE Server data sets by repeating this step.

   a. Click **Add new KIE Server DataSets**

   b. In the **DataSet name** field, enter `kieserver-1`.
c. In the **Kie Server Location** field, enter the location of your KIE Server, for example `https://my-kie-server:80/services/rest/server`.

d. To set your credentials, complete one of the following tasks:

- If you do not have a token set, in the **Username** and **Password** fields, enter your username and password. Leave the **Token** field blank.

- If you have a token, in the **Token** field, enter your token. Leave the **Username** and **Password** fields blank.

The custom resource example:

```
apiVersion: app.kiegroup.org/v2
kind: KieApp
metadata:
  name: standalone-dashbuilder
spec:
  environment: rhpam-standalone-dashbuilder
  objects:
    dashbuilder:
      config:
        kieServerDataSets:
          - name: kieserver-1
            location: 'https://my-kie-server:80/services/rest/server'
            user: kieserverAdmin
            password: kieserverAdminPwd
            replaceQuery: true
```

7. To add a KIE Server template, complete the following tasks:

   **NOTE**
   You can add additional KIE Server templates by repeating this step.

   a. Click **Add new KIE Server Templates**

   b. In the **Template name** field, enter a name for your template, for example `kieserver-template`.

   c. In the **KIE Server Location** field, enter the location of your KIE Server, for example `https://my-other-kie-server:80/services/rest/server`.

   d. To set your credentials, complete one of the following tasks:

   - If you do not have a token set, in the **Username** and **Password** fields, enter your username and password. Leave the **Token** field blank.

   - If you have a token, in the **Token** field, enter your token. Leave the **Username** and **Password** fields blank.

```
apiVersion: app.kiegroup.org/v2
kind: KieApp
metadata:
  name: standalone-dashbuilder
spec:
```

environment: rhpam-standalone-dashbuilder
objects:
dashbuilder:
  config:
    kieServerDataSets:
      - name: kieserver-1
        location: 'https://my-kie-server:80/services/rest/server'
        user: kieserverAdmin
        password: kieserverAdminPwd
        replaceQuery: true
    kieServerTemplates:
      - name: kieserver-template
        location: 'https://my-another-kie-server:80/services/rest/server'
        user: user
        password: pwd
        replaceQuery: true

8. Optional: To set a custom hostname for the external route, enter a domain in the Custom hostname to be used on the Dashbuilder external Route field, formatted as in the following example:

`dashbuilder.example.com`

**NOTE**

The custom hostname must be valid and resolvable.

To change the custom hostname, you can modify the `routeHostname` property.

### 53.2.1. Dashbuilder Standalone environment variables

When you use the Dashbuilder Container Image within operator, you can configure Dashbuilder by using the environment variables or through Custom Resource.

**Table 53.1. Custom Resource parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Equivalent Environment Variable</th>
<th>Description</th>
<th>Example value</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowExternalFileRegister</td>
<td>DASHBUILDER_ALLOW_EXTERNAL_FILE_REGISTER</td>
<td>Allows downloading of external (remote) files. Default value is false.</td>
<td>False</td>
</tr>
<tr>
<td>componentEnable</td>
<td>DASHBUILDER_COMPONENT_ENABLE</td>
<td>Enables external components.</td>
<td>True</td>
</tr>
<tr>
<td>componentPartition</td>
<td>DASHBUILDER_COMPONENT_PARTITION</td>
<td>Enables partitioning of components by the Runtime Model ID. Default value is true.</td>
<td>True</td>
</tr>
<tr>
<td>Parameter</td>
<td>Equivalent Environment Variable</td>
<td>Description</td>
<td>Example value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>configMapProps</td>
<td>DASHBOARD_CONFIG_MAP_PROPS</td>
<td>Allows the use of the properties file with Dashbuilder configurations. Unique properties are appended and if a property is set more than once, the one from the properties file is used.</td>
<td>True</td>
</tr>
<tr>
<td>dataSetPartition</td>
<td>DASHBOARD_DATASET_PARTITION</td>
<td>Enables partitioning of Dataset IDs by the Runtime Model ID. Default value is true.</td>
<td>True</td>
</tr>
<tr>
<td>enableBusinessCentral</td>
<td>–</td>
<td>Enables integration with Business Central by configuring Business Central and Dashbuilder automatically. Only available on operator.</td>
<td>True</td>
</tr>
<tr>
<td>enableKieServer</td>
<td>–</td>
<td>Enables integration with KIE Server by configuring KIE Server and Dashbuilder automatically. Only available on operator.</td>
<td>True</td>
</tr>
<tr>
<td>externalCompDir</td>
<td>DASHBOARD_EXTERNAL_COMP_DIR</td>
<td>Sets the base directory where dashboard ZIP files are stored. If PersistentConfigs is enabled and ExternalCompDir is not set to an existing path, the /opt/kie/dashbuilder/compone nts directory is used.</td>
<td>–</td>
</tr>
<tr>
<td>importFileLocation</td>
<td>DASHBOARD_IMPORTED_FILE_LOCATION</td>
<td>Sets a static dashboard to run automatically. If this property is set, imports are not allowed.</td>
<td>–</td>
</tr>
<tr>
<td>importsBaseDir</td>
<td>DASHBOARD_IMPORTED_BASE_DIR</td>
<td>Sets the base directory where dashboard ZIP files are stored. If PersistentConfigs is enabled and ImportsBaseDir is not set to an existing path, the /opt/kie/dashbuilder/imports directory is used. If ImportFileLocation is set ImportsBaseDir is ignored.</td>
<td>–</td>
</tr>
<tr>
<td>kieServerDataSets</td>
<td>KIESERVER_DATASETS</td>
<td>Defines the KIE Server data sets access configuration.</td>
<td>–</td>
</tr>
<tr>
<td>Parameter</td>
<td>Equivalent Environment Variable</td>
<td>Description</td>
<td>Example value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>kieServerTemplates</td>
<td>KIESERVER_SERVER_TEMPLATES</td>
<td>Defines the KIE Server Templates access configuration.</td>
<td></td>
</tr>
<tr>
<td>modelFileRemoval</td>
<td>DASHBUILDER_MODEL_FILE_REMOVAL</td>
<td>Enables automatic removal of model file from the file system. Default value is false.</td>
<td>False</td>
</tr>
<tr>
<td>modelUpdate</td>
<td>DASHBUILDER_MODEL_UPDATE</td>
<td>Allows Runtime to check model last update in the file system to update the content. Default value is true.</td>
<td>True</td>
</tr>
<tr>
<td>persistentConfigs</td>
<td></td>
<td>Sets Dashbuilder as not ephemeral. If <code>ImportFileLocation</code> is set <code>PersistentConfigs</code> is ignored. Default value is true. Available only on operator.</td>
<td>True</td>
</tr>
<tr>
<td>runtimeMultipleImport</td>
<td>DASHBUILDER_RUNTIME_MULTIPLE_IMPORT</td>
<td>Allows Runtime to allow imports (multi-tenancy). Default value is false.</td>
<td>False</td>
</tr>
<tr>
<td>uploadSize</td>
<td>DASHBUILDER_UPLOAD_SIZE</td>
<td>Sets the size limit for dashboard uploads (in kb). Default value is 10485760 kb.</td>
<td>10485760</td>
</tr>
<tr>
<td>env</td>
<td></td>
<td>Represents an environment variable present in a Container.</td>
<td></td>
</tr>
</tbody>
</table>

You can use operator to set environment variables by using the `env` property. The following example sets the value of the `DASHBUILDER_UPLOAD_SIZE` property to 1000.

```yaml
apiVersion: app.kiegroup.org/v2
kind: KieApp
metadata:
  name: standalone-dashbuilder
spec:
  environment: rhpam-standalone-dashbuilder
  objects:
    dashbuilder:
      env:
        - name: DASHBUILDER_UPLOAD_SIZE
          value: '1000'
```
A data set is a collection of related sets of information and can be stored in a database, in a Microsoft Excel file, or in memory. A data set definition instructs Business Central methods to access, read, and parse a data set. Business Central does not store data. It enables you to define access to a data set regardless of where the data is stored.

For example, if data is stored in a database, a valid data set can contain the entire database or a subset of the database as a result of an SQL query. In both cases the data is used as input for the reporting components of Business Central which then displays the information.

To access a data set, you must create and register a data set definition. The data set definition specifies the location of the data set, options to access it, read it, and parse it, and the columns that it contains.

**NOTE**
The Data Sets page is visible only to users with the admin role.

### 54.1. ADDING DATA SETS

You can create a data set to fetch data from an external data source and use that data for the reporting components.

**Procedure**

1. In Business Central, go to Admin → Data Sets. The Data Sets page opens.
2. Click **New Data Set** and select one of the following provider types:
   - **Bean**: Generates a data set from a Java class
   - **CSV**: Generates a data set from a remote or local CSV file
   - **SQL**: Generates a data set from an ANSI-SQL compliant database
   - **Elastic Search**: Generates a data set from Elastic Search nodes
   - **Prometheus**: Generates a data set using the Prometheus query
   - **Kafka**: Generates a data set using metrics from Kafka broker, consumer, or producer

   **NOTE**
   You must configure KIE Server for Prometheus, Kafka, and Execution Server options.
3. Complete the Data Set Creation Wizard and click Test.

   **NOTE**
   The configuration steps differ based on the provider you choose.
4. Click **Save**.
54.2. EDITING DATA SETS

You can edit existing data sets to ensure that the data fetched to the reporting components is up-to-date.

Procedure

1. In Business Central, go to Admin → Data Sets.
   The Data Set Explorer page opens.

2. In the Data Set Explorer pane, search for the data set you want to edit, select the data set, and click Edit.

3. In the Data Set Editor pane, use the appropriate tab to edit the data as required. The tabs differ based on the data set provider type you chose.
   For example, the following changes are applicable for editing a CSV data provider:
   - CSV Configuration: Enables you to change the name of the data set definition, the source file, the separator, and other properties.
   - Preview: Enables you to preview the data. After you click Test in the CSV Configuration tab, the system executes the data set lookup call and if the data is available, a preview appears. Note that the Preview tab has two sub-tabs:
     - Data columns: Enables you to specify what columns are part of your data set definition.
     - Filter: Enables you to add a new filter.
   - Advanced: Enables you to manage the following configurations:
     - Caching: See Caching data for more information.
     - Cache life-cycle: Enables you to specify an interval of time after which a data set (or data) is refreshed. The Refresh on stale data feature refreshes the cached data when the back-end data changes.

4. After making the required changes, click Validate.

5. Click Save.

54.3. DATA REFRESH

The data refresh feature enables you to specify an interval of time after which a data set (or data) is refreshed. You can access the Data refresh every feature on the Advanced tab of the data set. The Refresh on stale data feature refreshes the cached data when the back-end data changes.

54.4. CACHING DATA

Business Central provides caching mechanisms for storing data sets and performing data operations using in-memory data. Caching data reduces network traffic, remote system payload, and processing time. To avoid performance issues, configure the cache settings in Business Central.

For any data lookup call that results in a data set, the caching method determines where the data lookup call is executed and where the resulting data set is stored. An example of a data lookup call would be all the mortgage applications whose locale parameter is set as "Urban".
Business Central data set functionality provides two cache levels:

- Client level
- Back-end level

You can set the **Client Cache** and **Backend Cache** settings on the **Advanced** tab of the data set.

**Client cache**

When the cache is turned on, the data set is cached in a web browser during the lookup operation and further lookup operations do not perform requests to the back-end. Data set operations like grouping, aggregations, filtering, and sorting are processed in the web browser. Enable client caching only if the data set size is small, for example, for data sets with less than 10 MB of data. For large data sets, browser issues such as slow performance or intermittent freezing can occur. Client caching reduces the number of back-end requests including requests to the storage system.

**Back-end cache**

When the cache is enabled, the decision engine caches the data set. This reduces the number of back-end requests to the remote storage system. All data set operations are performed in the decision engine using in-memory data. Enable back-end caching only if the data set size is not updated frequently and it can be stored and processed in memory. Using back-end caching is also useful in cases with low latency connectivity issues with the remote storage.

**NOTE**

Back-end cache settings are not always visible in the **Advanced** tab of the **Data Set Editor** because Java and CSV data providers rely on back-end caching (data set must be in the memory) in order to resolve any data lookup operation using the in-memory decision engine.

---

### 54.5. KIE SERVER DATA SETS WITH DASHBUILDER RUNTIME AND DASHBUILDER STANDALONE

A data set is a collection of related information. If you have a KIE Server that contains imported data sets, you can use Dashbuilder Runtime or Dashbuilder Standalone and the KIE Server REST API to run queries on imported data sets.

Because KIE Server uses Business Central as a controller, KIE Server containers are created in Business Central. Data sets are also created in Business Central. The KIE Server configuration is a template that you can refer to when you create data sets or install containers.

Other services, such as Dashbuilder Runtime and Dashbuilder Standalone, use the KIE Server REST API to retrieve KIE Server information. Dashbuilder Runtime and Dashbuilder Standalone access the KIE Server REST API to run queries from data sets.

When a KIE Server data set is created in Business Central, the server template information is provided and it is used by Dashbuilder Runtime and Dashbuilder Standalone to look for the KIE Server information. For example:

```
dashbuilder.kieserver.serverTemplate.{SERVER_TEMPLATE}.location={LOCATION}
dashbuilder.kieserver.serverTemplate.{SERVER_TEMPLATE}.user={USER}
dashbuilder.kieserver.serverTemplate.{SERVER_TEMPLATE}.password={PASSWORD}
dashbuilder.kieserver.serverTemplate.{SERVER_TEMPLATE}.token={TOKEN}
```

You can also setup KIE Server for each data set. For example:
dashbuilder.kieserver.dataset.{DATA_SET_NAME}.location={LOCATION}
dashbuilder.kieserver.dataset.{DATA_SET_NAME}.user={USER}
dashbuilder.kieserver.dataset.{DATA_SET_NAME}.password={PASSWORD}
dashbuilder.kieserver.dataset.{DATA_SET_NAME}.token={TOKEN}

**NOTE**

Token authentication is not used if credentials are provided.

You might want to run the dashboard against another KIE Server installation. When data sets are created on a KIE Server in a development environment, the data sets queries are created on the development KIE Server, for example DEV. If a dashboard is exported to a production environment, for example PROD, with a different KIE Server, the queries that you created in DEV are not available, so an error is thrown. In this case it is possible to port queries from a data set to another KIE Server by using the replace query functionality, either through a server template or a data set:

- **Server template example:**
  
  dashbuilder.kieserver.serverTemplate.{SERVER_TEMPLATE}.replace_query=true

- **Data set example:**
  
  dashbuilder.kieserver.dataset.{DATA_SET_NAME}.replace_query=true

The **replace_query=true property** only needs to be set once so that Dashbuilder Runtime or Dashbuilder Standalone creates the queries. After the queries are created you can remove this system property.

**Additional resources**

- *Interacting with Red Hat Process Automation Manager using KIE APIs*
CHAPTER 55. PAGE AUTHORING

Pages are collections of following components:

- Core components
- Navigational components
- Reporting components
- Heatmaps

Also, a page can have none or any number of components. The Page Editor tool is used to edit the pages.

A page has either the Fluid style or the Page style. The Fluid style is a classical web page with a vertical scroll bar, when the page exceeds the available height. The Page style is a web page that will always fit to the windows height.

55.1. CREATING PAGES

You can use the Pages perspective to create a page that consists of different types of components. After a page has been created and all the components in it defined, use the Page Editor to save, delete, rename, or copy pages as required.

The following procedure describes how to create a page and add the required components to it:

**Procedure**

1. In Business Central, go to Menu → Design → Pages.
2. In the Pages panel, click New. Alternatively, click New Page on the Page Editor pane.
3. In the New Page dialog box, enter a value in the Name field and select the required style.
4. Click Ok. The new page opens in the Page Editor.
5. In the Components pane, expand the components and drag the required component types to the editor canvass.
6. After placing the components on the page, edit its properties from the Properties pane.
7. Click Save and then click Save again.

55.2. SAVING, DELETING, RENAMING, OR COPYING PAGES

After you have created and defined a page, use the Page Editor to save, delete, rename, or copy pages as required.

**Procedure**

1. In Business Central, go to Menu → Design → Pages.
2. Select a page from the Pages panel. The page opens in the Page Editor.
3. Perform the desired operation and select Save, Delete, Rename, or Copy from the upper right corner of the Page Editor.

Figure 55.1. Save, delete, rename, or copy pages

![Save Delete Rename Copy](image)

55.3. NAVIGATION TREE

The Workbench navigation tree contains the entries that are displayed in Business Central's Main menu. Changes to this navigation tree structure will be reflected in the Main menu on the Home page. Such mechanism can be used, for example, to publish new pages.

Also, additional navigation trees can be created. These custom trees can be used to configure the navigation components within a page. You cannot delete the Workbench tree from the Navigation panel but you can edit the Workbench tree hierarchy to suit your requirements. This feature helps users customize the Main menu of Business Central with new pages.

**NOTE**

The Workbench tree, which appears by default in the Navigation panel, is the Main menu of Business Central.

55.3.1. Creating a navigation tree

You can create as many custom navigation trees as required. A custom navigation tree differs from the default Workbench navigation tree in one aspect. The custom navigation trees can be deleted but the default tree can not be removed from Business Central. They can contain the default groups and entries from the Workbench tree as well as user created groups and trees.

**Prerequisites**

- You have sufficient user permissions for creating a navigation tree.

**Procedure**

1. Log in to Business Central and go to Menu → Design → Pages.
2. Select the Navigation panel and then click New.
3. Enter the name of the new navigation tree and click the check mark icon or, press Enter.
4. Click Save.

55.3.2. Editing a navigation tree

You can use the Navigation panel from the Pages perspective to edit the custom navigation trees. You can further customize the trees by adding groups, dividers, and page entries, deleting the specific entries of the tree, and reordering, renaming, or deleting them.

**Prerequisites**

- You have sufficient user permissions for editing a navigation tree.
55.3.3. Adding groups, dividers, and page entries to a navigation tree

You can add groups, dividers, and page entries to a navigation tree.

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.
2. Click the **Navigation** panel and then select the navigation tree you want to add the entries to.
3. Click the gear icon of the tree and then select **New Group**, **New Divider**, or **New Page**.
4. Enter the name of the new group or page and click the check mark icon or press Enter.

   **NOTE**
   Divider entries do not have a name property.

5. Click **Save**.

55.3.4. Reordering a navigation tree

In the **Navigation** panel you can reorder the navigation trees as well as its entries.

**NOTE**

The reordering options of tree entries will differ according to their position in the tree hierarchy.

Reordering navigation trees

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.
2. Click the **Navigation** panel and then select the navigation tree you want to reorder.
3. Click the gear icon of the tree and then move up or down as required.

   **NOTE**
   First and last entries of a navigation tree have only two reordering options available.

4. Click **Save**.

Reordering the entries of a navigation tree

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.
2. Click the **Navigation** panel and then expand the navigation tree.
3. Click the gear icon of the entry you want to reorder and then move up or down as required.

4. Optional: Click **Goto Page** to view the selected page.

5. Click **Save**.

### 55.3.5. Renaming a navigation tree

You can rename all navigation trees except the Workbench tree.

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.

2. Click the **Navigation** panel and then select the custom navigation tree you want to rename.

   **NOTE**

   To rename an entry of a tree, expand the tree and select the entry that you want to rename.

3. Click the edit icon of the tree or the tree entry.

4. Enter the new name of the tree and click the check mark icon.

   **NOTE**

   You cannot rename divider entries.

5. Click **Save**.

### 55.3.6. Deleting a navigation tree

You can delete any navigation tree except the **Workbench** tree from the **Navigation** panel of the Pages perspective.

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.

2. In the **Navigation** panel, select the navigation tree that you want to delete, then click the delete icon.

3. Click **Save**.

### 55.3.7. Deleting the entries of a navigation tree

You can delete the entries of a navigation tree.

**Procedure**

1. In Business Central, go to **Menu → Design → Pages**.
2. Click the **Navigation** panel.

3. Expand the tree that contains the entries you want delete.

4. Click the delete icon of the entry.

5. Click **Save**.

### 55.4. COMPONENTS

A page contains different types of components. You can use the following component types on a page using the **Components** panel:

- Core components
- Navigation components
- Reporting components
- Heatmap components

#### 55.4.1. Core components

You can use the **Core** components to specify custom HTML information or display existing pages. There are three types of core components.

**NOTE**

Core components are not mandatory.

<table>
<thead>
<tr>
<th>Core component sub-types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>This component opens an HTML editor widget that is used to create HTML pages using text, images, tables, links, and colors. You can also customize an HTML page if required.</td>
</tr>
<tr>
<td>Page</td>
<td>This component enables you to add previously created HTML pages to a new dashboard. You can use this component to nest custom pages that you have already created in your dashboard.</td>
</tr>
<tr>
<td>Logo</td>
<td>This component enables you to add images on the page. To add images on a page, you can provide a URL for the image and set the width and height of the image as per your requirements. The default image is the logo of Dashbuilder. The <strong>Logo URL</strong> field is mandatory.</td>
</tr>
</tbody>
</table>

#### 55.4.2. Navigation components
The **Navigation** components are used to navigate between the pages. There are six types of navigation components.

### Table 55.2. Navigation component subtypes

<table>
<thead>
<tr>
<th>Navigation component sub-types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Div</td>
<td>This component displays the entries of the tab list, menu bar, and navigation tree. Also, it keeps the track of the last clicked item.</td>
</tr>
<tr>
<td>Menu Bar</td>
<td>This component displays the entries of a navigation tree in the form of a menu bar. There is no limit to the number of levels supported by Business Central.</td>
</tr>
<tr>
<td>Tile Navigator</td>
<td>This component displays the navigation group in the form of tiles. The groups are displayed as folders but for single entries and the content is displayed instead when selected.</td>
</tr>
<tr>
<td>Tree</td>
<td>This component displays the entries in a vertical tree structure form.</td>
</tr>
<tr>
<td>Carousel</td>
<td>This component displays the selected pages in a carousel or a slide show form.</td>
</tr>
<tr>
<td>Tab List</td>
<td>This component displays the selected menu pages as tabs at the top of the component.</td>
</tr>
</tbody>
</table>

**NOTE**

The **Target Div** setting is not needed for non-target div components such as carousel or tile navigator.

### 55.4.3. Reporting components

The **Reporting** components are used to display the data from data sets in the form of graphs, tables, maps. There are eleven types of reporting components. For more information about the data sets, see Data sets authoring section.

You can configure the reporting components using the **New Displayer** widget, which contains the following tabs:

- **Type**: Enables you to select how to display custom data graphically.
- **Data**: Enables you to select a data set from the list of custom data sets that you created from the Data Sets section available in the Settings menu.
- **Display**: Enables you to select and customize how the content is displayed by adding titles, changing colors, size.
NOTE

For **Time Series Chart** component, the **New Displayer** widget contains the **Data**, **Display**, and **Component Editor** tabs.

### Table 55.3. Reporting component subtypes

<table>
<thead>
<tr>
<th>Reporting component sub-types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar</td>
<td>This component is used to display the data from data sets in a bar chart form.</td>
</tr>
<tr>
<td>Pie</td>
<td>This component is used to display the data from data sets in the form of pie chart form. In a pie chart, the arc length of each slice is proportional to the represented quantity.</td>
</tr>
<tr>
<td>Line</td>
<td>This component is used to display the data from data sets as a series of data points connected by straight line segments on two axes.</td>
</tr>
<tr>
<td>Area</td>
<td>This component combines a line chart and a bar chart to display the data from data sets.</td>
</tr>
<tr>
<td>Bubble</td>
<td>This component displays multiple circles (bubbles) in a two-dimensional plot. It is a generalization of the scatter plot that replaces the dots with bubbles.</td>
</tr>
<tr>
<td>Meter</td>
<td>This component is used to display the data from data sets in a meter form.</td>
</tr>
<tr>
<td>Map</td>
<td>This component enables you to position your data in a context, often geographical, using different layers. The data values are displayed as markers on the map. The data values can act as coordinates or addresses.</td>
</tr>
<tr>
<td>Metric</td>
<td>This component is used to display the data from data sets in a metric form. You can use the <strong>Preview</strong>, <strong>HTML</strong>, or <strong>Javascript</strong> tab to edit your data.</td>
</tr>
<tr>
<td>Table</td>
<td>This component is used to display the data from data sets in a table form. You can hide or display the columns if required.</td>
</tr>
<tr>
<td>Filter</td>
<td>This component allows you to filter the data from data sets.</td>
</tr>
<tr>
<td><strong>Time Series Chart</strong></td>
<td>This component is used to display the data from data sets in a time series form.</td>
</tr>
</tbody>
</table>
55.4.4. Component properties

The components used in a page have different properties associated with them. The Properties panel enables you to customize a component by editing the following properties:

- **Panel** properties: Used to customize component panel properties such as **Width**, **Height**, and **Background Color**.
- **Margin** properties: Used to customize component margin properties such as **Top**, **Bottom**, **Left**, and **Right**.
- **Padding** properties: Used to customize component padding properties such as **Top**, **Bottom**, **Left**, and **Right**.

55.4.5. Placing components on the page editor to create a page

To create pages, you must drag the components to the Editor canvas of the Pages perspective. After placing all of the required components on the page, click **Save**.

55.4.6. Using the Preview tab to preview pages

While creating or editing pages, click the **Preview** tab on the Page Editor to preview a page before saving it.

55.4.7. Adding a time series chart component on a page

You can use the **Time Series Chart** component to represent any time series data. You can create your own dashboards that can be connected to your time series data sets.

You can export the time series components to Dashbuilder Runtime and retrieve the information from a KIE Server or any Prometheus data set. You can also create, edit, and build a dashboard using a time series chart component.

**Prerequisites**

- KIE Server is deployed and connected to Business Central.

**Procedure**

1. Create a new KIE Server data set using the following steps:
   i. In Business Central, go to **Admin → Data Sets**.
      The Data Set Explorer page opens.
   ii. Click **New Data Set** and select any provider type as per your requirements.
      The Data Set Creation Wizard page opens.
   iii. Enter required details in the Data Set Creation Wizard of your selected provider type and click **Test**.
   iv. Click **Save**.

2. In Business Central, go to **Menu → Design → Pages**.

3. In the Pages panel, click **New**.
4. Click OK.

5. In the New Page dialog box, enter a name in the Name field and select the required style. The new page opens in the Page Editor.

6. In the Components panel, expand the Reporting components and drag the Time Series Chart to the Page Editor.

7. In the Display editor wizard, click the Data tab and select the data set that you created.

8. In the Data tab, select the values from the Columns field as per your requirements.

9. Click the Display tab and edit the value of Chart, Margins, Filter, Refresh, and Columns if needed.

10. Click the Component Editor tab and update the following component properties into the Component Properties field:

Table 55.4. Time series component properties

<table>
<thead>
<tr>
<th>Component properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transpose dataset</td>
<td>Define that the provided data set uses the time series as separate columns or as rows.</td>
</tr>
<tr>
<td>Show Area</td>
<td>Select the checkbox to set the type as a line chart or an area chart.</td>
</tr>
<tr>
<td>Date Categories</td>
<td>Select the category, datetime, or numeric option from the drop-down list.</td>
</tr>
<tr>
<td>Labels</td>
<td>Select the checkbox to enable or disable the data labels on data points.</td>
</tr>
<tr>
<td>Zoom type</td>
<td>Select the x, y, or xy option from the drop-down list.</td>
</tr>
<tr>
<td>Zoom enabled</td>
<td>Select the checkbox to enable zooming in axis charts. By default, the checkbox is checked.</td>
</tr>
<tr>
<td>Zoom autoscale Y axis</td>
<td>Select the checkbox to re-scale highs and lows based on visible area.</td>
</tr>
<tr>
<td>Toolbar Auto Selected</td>
<td>Select the zoom, selection, or pan option from the drop-down list.</td>
</tr>
<tr>
<td>Title text</td>
<td>Edit the title of the time series chart component.</td>
</tr>
<tr>
<td>Title align</td>
<td>Select the left, center, or right option from the drop-down list to change the title alignment.</td>
</tr>
</tbody>
</table>
Component properties | Description
--- | ---
Show toolbar | It is a checkbox to enable or disable the toolbar on the upper-right corner of the chart. By default, this checkbox is checked. If this component property is enabled, you can use the zoom in, zoom out, selection zoom, and panning functions of the time series chart component.

Chart Name | Set the chart name as per your requirements. By default, the chart name is set as **Newchart**.

11. Optional: Click the sandwich menu icon in the upper-right corner of the chart to download the data set in CSV, PNG, or SVG format.

12. Click **OK**.

**Figure 55.2. Example time series component**

55.5. HEATMAP COMPONENTS

In Business Central, you can add heatmap components to a page. Heatmap components are used to display heat information over a process diagram. The color on the process diagram nodes is related to the value you assign to each node and based on the assigned value, the color varies on a process diagram. If the assigned value is maximum, the heat intensifies and if a minimum value is assigned, then heat is not displayed on the process diagram.

You can export the heatmap components to Dashbuilder Runtime or Dashbuilder Standalone and retrieve the heat information from a KIE Server data set. You can also create, edit, and build a dashboard using a heatmap component.

55.5.1. Creating heatmap components for a process

You can create heatmap components for a specific process in Business Central.
Prerequisites

- KIE Server is deployed and connected to Business Central.
- You have created a project with at least one business process asset in Business Central.
- A project with a process definition has been deployed in Business Central.
- Sample process instances are created.

Procedure

1. Create a new KIE Server data set using the following steps:
   
i. In Business Central, go to Admin → Data Sets. The Data Set Explorer page opens.
   
   ii. Click New Data Set and select the Execution Server provider type. The Data Set Creation Wizard page opens.
   
   iii. Enter the name of the dataset.
   
   iv. Select the server configuration. The server configuration is available if the project is deployed.
   
   v. Select CUSTOM query target from the list.
   
   vi. In the Query field, enter the following custom SQL query.

```sql
select
  pil.externalId,
  pil.processInstanceId,
  nil.nodeId,
  nil.nodeType,
  nil.nodeName,
  count(nil.nodeId) as total_hits
from
  NodeInstanceLog nil
inner join
  ProcessInstanceLog pil on pil.processInstanceId = nil.processInstanceId
where
  nil.type = 1
group by
  pil.externalId,
  nil.nodeId,
  nil.nodeName
```

   NOTE

You can modify the SQL query according to your database if required.

   vii. Complete the Data Set Creation Wizard and click Test.
   
   viii. Click Save.

2. In Business Central, go to Menu → Design → Pages.
3. In the **Pages** panel, click **New**.

4. In the **New Page** dialog box, enter a value in the **Name** field and select the required style.

5. Click **OK**. 
   The new page opens in the **Page Editor** and you can see that the **Heatmaps** component is available under the **Components** panel.

6. In the **Components** panel, expand the **Heatmaps** components and drag the **Process Heatmap** component type to the **Page Editor**.

7. In the **Displayer editor** wizard, click **Data** tab and select the newly created KIE Server dataset.

8. In the **Data** tab, select **NODEID** and **TOTAL_HITS** from the **Columns** field.

9. Click the **Component Editor** tab and enter the values for the mandatory fields including **Server Template**, **Container ID**, and **Process Definition ID** into the **Component Properties** field.

   **NOTE**
   
   To access the value of **Server Template**, go to **Deploy → Execution Servers → Server Configurations**. For the value of **Container ID**, go to **Manage → Process Instances**, you can click on the process instance you want to use and in the, **Deployment** corresponds to the **Container ID** and the **Definition ID** is the **Process Definition ID**.

10. Click the **Display** tab and edit the value of **Chart**, **Margins**, **Filter**, **Refresh**, and **Columns** as per your requirements.

11. Click **OK**.

**Figure 55.3. Example process heatmap components**

You can see the heat information on the process diagram.

### 55.5.2. Creating heatmap components for multiple processes

You can create heatmap components for multiple processes in Business Central.

**Prerequisites**

- KIE Server is deployed and connected to Business Central.
• You have created multiple projects in Business Central and it contains at least one business process asset.

• A project with a process definition has been deployed in Business Central.

• Sample process instances are created.

Procedure

1. Create a new KIE Server dataset using the following steps:

   i. In Business Central, go to Admin → Data Sets. The Data Set Explorer page opens.

   ii. Click New Data Set and select the Execution Server provider type. The Data Set Creation Wizard page opens.

   iii. Enter the name of the dataset.

   iv. Select the server configuration. The server configuration is available if the project is deployed.

   v. Select CUSTOM query target from the list.

   vi. In the Query field, enter the following custom SQL query.

   ```sql
   select
     pil.externalId,
     pil.processId,
     nil.nodeid,
     nil.nodeType,
     nil.nodeName,
     count(nil.nodeid) as total_hits
   from
     NodeInstanceLog nil
   inner join
     ProcessInstanceLog pil on pil.processInstanceId = nil.processInstanceId
   where
     nil.type = 1
   group by
     pil.externalId,
     nil.nodeid,
     nil.nodeName
   ```

   NOTE
   You can modify the SQL query according to your database if required.

   vii. Complete the Data Set Creation Wizard and click Test.

   viii. Click Save.

2. In Business Central, go to Menu → Design → Pages.

3. In the Pages panel, click New.
4. In the **New Page** dialog box, enter a value in the **Name** field and select the required style.

5. Click **OK**.
   The new page opens in the **Page Editor** and you can see that the **Heatmaps** component is available under the **Components** panel.

6. In the **Components** panel, expand the **Heatmaps** components and drag the **All Processes Heatmaps** component type to the canvas.

7. In the **Display editor** wizard, click **Data** tab and select the newly created KIE Server dataset.

8. In the **Data** tab, select **EXTERNALID, PROCESSID, NODEID, and TOTAL_HITS** from the **Columns** field.

9. In the **Process Selector** box, select the **Container** and **Process** values as per your requirement.

10. Click the **Component Editor** tab and enter the value in the **Server Template** (mandatory) field.

   **NOTE**

   To access the value of **Server Template**, go to **Deploy → Execution Servers → Server Configurations**.

11. Click the **Display** tab and edit the value of **Chart, Margins, Filter, Refresh**, and **Columns** as per your requirements.

12. Click +**OK**.

   **Figure 55.4. Example multiple processes heatmap components**

You can see the heat information on the process diagram.

**55.5.3. Executing a heatmap component**
You can execute the internal heatmap components locally, outside of Business Central for testing purpose only. The same API is used to create external components, which you can use to build your own components. For more information about external components, see Section 55.6, “External components”

To run the specific heatmap component, complete the following steps:

**Prerequisites**

- You have installed npm on your system. For more information about installing npm, see [Downloading and installing Node.js and npm](#).
- You have installed Yarn on your system. For more information about installing Yarn, see [Yarn Installation](#).
- You have cloned the Appformer repository to run the components outside of Business Central.

**Procedure**

1. Go to the `appformer/dashbuilder/dashbuilder-shared/dashbuilder-js` directory.
2. Open the terminal in the `dashbuilder-js` directory and enter the following command:
   
   ```
   yarn run init && yarn run build:fast
   ```
   
   You can see the following components are present in the `dashbuilder-js/packages` directory:
   
   - processes-heatmaps-component
   - process-heatmap-component
   - logo-component
   - heatmap-component
3. Go to the `dashbuilder-js/packages` directory, open the desired heatmap component and enter the following command in the terminal:
   
   ```
   yarn run start
   ```
4. To access the component, enter `http://localhost:9001/` in a web browser. The selected component is displayed on the web browser.

**55.6. EXTERNAL COMPONENTS**

In Business Central, you can add external components to a page. The components are disabled by default. To enable the external components, change the value of `dashbuilder.components.enable system property to `true`.  

The external component location is set and configured with the `dashbuilder.components.dir system property. The default value of this system property is `/tmp/dashbuilder/components`. You must set the component under the components directory with a parent directory, which is used as the component ID. For example, if the component ID is `mycomp` and the component directory is `/tmp/dashbuilder/components`, then the component base directory is `/tmp/dashbuilder/components/mycomp`. 
Business Central checks the `manifest.json` file in the components directory. The `manifest.json` must contain at least one `name` text parameter.

### Table 55.5. `manifest.json` file descriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the component displayed under <code>Components</code> section.</td>
</tr>
<tr>
<td>icon</td>
<td>Icon of the component displayed under <code>Components</code> section.</td>
</tr>
<tr>
<td>noData</td>
<td>A flag that indicates that the component does not require a data set.</td>
</tr>
<tr>
<td>parameters</td>
<td>The list of parameters are using <code>ComponentParameter</code> type. Supported parameter types include <code>name</code>, <code>type</code>, <code>category</code>, <code>defaultValue</code>, <code>label</code>, <code>mandatory</code>, and <code>comboValues</code>.</td>
</tr>
</tbody>
</table>

#### Sample `manifest.json` file

```json
{
   "name": "Heat Map Experiment",
   "icon": "fa fa-bell-o",
   "parameters": [
   {
      "name": "svg",
      "type": "text",
      "defaultValue": "",
      "label": "SVG XML",
      "category": "SVG Content"
      "mandatory": true
   },
   {
      "name": "svgUrl",
      "type": "text",
      "defaultValue": "",
      "label": "SVG URL",
      "category": "SVG URL"
      "mandatory": true
   }
   ,
   {
      "name": "ksProcessId",
      "type": "text",
      "defaultValue": "",
      "label": "Process ID",
      "category": "Kie Server"
      "mandatory": true
   }
   
```
55.6.1. Creating external components

The following procedure describes how to create and add the external components to a page:

Procedure

1. Set the component under components directory with a parent directory.
   For example, if the component ID is `mycomp` and the component directory is `/tmp/dashbuilder/components`, then the component base directory is `/tmp/dashbuilder/components/mycomp`.

2. Create the `manifest.json` file in the component directory.

3. Create `index.html` file with HTML content.

4. In a terminal application, navigate to `EAP_HOME/bin`.

5. To enable the external components, set the value of `dashbuilder.components.enable` system property to `true`:
   ```bash
   $ ~/EAP_HOME/bin/standalone.sh -c standalone-full.xml
   -Ddashbuilder.components.dir={component directory base path} -
   Ddashbuilder.components.enable=true
   ```


7. In the Components pane, expand the External Components and drag the required component types to the canvas.

8. Click Save.
CHAPTER 56. SECURITY MANAGEMENT

Security management is the process of managing users, groups, and permissions. You can control access to Business Central resources and features from the Business Central Security management page.

Business Central defines three types of entities for security management: users, groups, and roles. You can assign permissions to both roles and groups. A user inherits permissions from the groups and roles that the user is a member of.

56.1. SECURITY MANAGEMENT PROVIDERS

In the context of security management, a realm restricts access to different application resources. Realms contain information about users, groups, roles, and permissions. A concrete user and group management service implementation for a specific realm is called a security management provider.

If the built-in security management providers do not meet the requirements of your application security realm, then you can build and register your own security management provider.

NOTE

If the security management provider is not installed, the user interface for managing the security realm is not available. After you install and configure a security management provider, the user and group management features are automatically enabled in the security management user interface.

Business Central includes the Red Hat JBoss EAP security management provider which supports realm types based on the contents of the application-users.properties or application-roles.properties property file.

56.1.1. Configuring the Red Hat JBoss EAP security management provider based on property files

You can build and register your own Red Hat JBoss EAP security management provider. To use the Red Hat JBoss EAP security management provider based on property files, complete the steps in this procedure.

Prerequisites

- Red Hat JBoss EAP is installed.

Procedure

1. To use an existing users or roles property file from the Red Hat JBoss EAP instance, include the following system properties in the EAP_HOME/standalone/configuration/application-users.properties and EAP_HOME/standalone/configuration/application-roles.properties files, as shown in the following example:

   `<property name="org.uberfire.ext.security.management.wildfly.properties.realm" value="ApplicationRealm"/>
   <property name="org.uberfire.ext.security.management.wildfly.properties.users-file-path"`
The following table provides a description and default value for these properties:

### Table 56.1. Red Hat JBoss EAP security management provider based on property files

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.realm</td>
<td>The name of the realm. This property is not mandatory.</td>
<td>ApplicationRealm</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.users-file-path</td>
<td>The absolute file path for the users property file. This property is mandatory.</td>
<td>./standalone/configuration/application-users.properties</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.properties.groups-file-path</td>
<td>The absolute file path for the groups property file. This property is mandatory.</td>
<td>./standalone/configuration/application-roles.properties</td>
</tr>
</tbody>
</table>

2. Create the `security-management.properties` file in the root directory of your application. For example, create the following file:

```xml
<property name="org.uberfire.ext.security.management.wildfly.properties.groups-file-path" value="/standalone/configuration/application-roles.properties"/>
```

3. Enter the following system property and security provider name as a value in the `security-management.properties` file:

```xml
<property name="org.uberfire.ext.security.management.api.userManagementServices" value="WildflyUserManagementService"/>
```

### 56.1.2. Configuring the Red Hat JBoss EAP security management provider based on property files and CLI mode

To use the Red Hat JBoss EAP security management provider based on property files and CLI mode, complete the steps in this procedure.

**Prerequisites**

- Red Hat JBoss EAP is installed.

**Procedure**

1. To use an existing users or roles property file from the Red Hat JBoss EAP instance, include the following system properties in the `EAP_HOME/standalone/configuration/application-users.properties` and `EAP_HOME/standalone/configuration/application-roles.properties` files, as shown in the following example:

```xml
<property name="org.uberfire.ext.security.management.wildfly.cli.host" value="localhost"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.port" value="9990"/>
```
The following table provides a description and default value for these properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.uberfire.ext.security.management.wildfly.cli.host</td>
<td>The native administration interface host.</td>
<td>localhost</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.cli.port</td>
<td>The native administration interface port.</td>
<td>9990</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.cli.user</td>
<td>The native administration interface username.</td>
<td>NA</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.cli.password</td>
<td>The native administration interface user’s password.</td>
<td>NA</td>
</tr>
<tr>
<td>org.uberfire.ext.security.management.wildfly.cli.realm</td>
<td>The realm used by the application’s security context.</td>
<td>ApplicationRealm</td>
</tr>
</tbody>
</table>

2. Create the `security-management.properties` file in the root directory of your application. For example, create the following file:

```xml
<property name="org.uberfire.ext.security.management.wildfly.cli.user" value="<USERNAME>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.password" value="<USER_PWD>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.realm" value="ApplicationRealm"/>
```

3. Enter the following system property and security provider name as a value in the `security-management.properties` file:

```xml
<property name="org.uberfire.ext.security.management.api.userManagementServices" value="WildflyCLIUserManagementService"/>
```

### 56.2. PERMISSIONS AND SETTINGS

A permission is an authorization granted to a user to perform actions related to a specific resource within the application. For example, a user can have following permissions:

- View a page.
- Save the project.
- View a repository.
Delete a dashboard.

You can grant or deny a permission and a permission can be global or resource specific. You can use permissions to protect access to resources and customize features within the application.

### 56.2.1. Changing permissions for groups and roles in Business Central

In Business Central, you cannot change permissions for an individual user. However, you can change permissions for groups and roles. The changed permissions apply to users with the role or that belong to a group that you changed.

**NOTE**

Any changes that you make to roles or groups affect all of the users associated with that role or group.

**Prerequisites**

- You are logged in to Business Central with the **admin** user role.

**Procedure**

1. To access the **Security management** page in Business Central, select the **Admin** icon in the top-right corner of the screen.

2. Click **Roles**, **Groups**, or **Users** on the Business Central **Settings** page. The **Security management** page opens on the tab for the icon that you clicked.

3. From the list, click the role or group you want to edit. All details are displayed in the right panel.

4. Set the **Home Page** or **Priority** under the **Settings** section.

5. Set the Business Central, page, editor, space, and project permissions under the **Permissions** section.

**Figure 56.1. Setting the permissions**

6. Click the arrow next to a resource type to expand the resource type whose permissions you want to change.
7. Optional: To add an exception for a resource type, click **Add Exception** and then set the permissions as required.

**NOTE**
You cannot add an exception to the Business Central resource type.

8. Click **Save**.

### 56.2.2. Changing the Business Central home page

The home page is the page that appears after you log in to Business Central. By default, the home page is set to **Home**. You can specify a different home page for each role and group.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Roles** or **Groups**.

2. Select a role or group.

3. Select a page from the **Home Page** list.

4. Click **Save**.

**NOTE**
The role or group must have read access to a page before you can make it the home page.

### 56.2.3. Setting priorities

A user can have multiple roles and belong to multiple groups. The Priority setting determines the order of precedence of a role or group.

**Prerequisites**

- You are logged in to Business Central with the **admin** user role.

**Procedure**

1. In Business Central, select the **Admin** icon in the top-right corner of the screen and select **Roles** or **Groups**.

2. Select a role or group.

3. Select a priority from the Priority menu, and then click **Save**.

**NOTE**
If a user has a role or belongs to a group that has conflicting settings, the settings of the role or group with the highest priority applies.
CHAPTER 57. EXPORTING, IMPORTING, AND DEPLOYING DASHBOARDS

After you create a dashboard in Business Central, you can export the dashboard data and import it in another instance of Business Central, in Dashbuilder Runtime, or in Dashbuilder Standalone.

NOTE
This feature is only accessible by administrator users.

57.1. EXPORTING BUSINESS CENTRAL DASHBOARD DATA

You can export dashboard data such as data sets, pages, and navigation from Business Central as a ZIP file.

Procedure

1. In Business Central, select the Admin icon in the upper-right corner of the screen and select Dashbuilder Data Transfer.

2. To export the dashboard data, complete any of the following tasks:
   If you want to export all of the dashboard data as a ZIP file, complete the following tasks:
   i. On the Dashbuilder Data Transfer page, click Export all.
      An export.zip file containing all of the dashboard data is downloaded. The export.zip file structure is separated by data type, as shown in the following example:

      ```
      dashbuilder/datasets/definitions/dataset-example1.dset
      dashbuilder/datasets/definitions/dataset-example2.dset
      dashbuilder/datasets/readme.md
      dashbuilder/perspectives/page1/perspective_layout
      dashbuilder/perspectives/page1/perspective_layout.plugin
      dashbuilder/perspectives/page2/perspective_layout
      dashbuilder/perspectives/page2/perspective_layout.plugin
      dashbuilder/perspectives/readme.md
      dashbuilder/navigation/navigation/navtree.json
      dashbuilder/navigation/readme.md
      VERSION
      ```

      If you want to export the customized user created and provide dashboard data as a ZIP file, complete the following tasks:

      i. On the Dashbuilder Data Transfer page, click Custom export

      ii. Select the datasets and pages that you want to include in a ZIP file in the Export Wizard panel and click Next.
      The Export Wizard panel validates the selected data sets and pages. The summary of the data sets and pages is available on the panel.

      NOTE
      Navigation is always included in the exported ZIP file.
iii. If your export is ready, click **Download**. An **export.zip** file containing customized dashboard data is downloaded.

**NOTE**

You must select the associated pages and data sets in the **Export Wizard** panel. If you fail to select both data sets and pages, then an error is generated and you cannot download the **export.zip** file. Therefore, you must select at least one page.

iv. Click **Finish**.

### 57.2. IMPORTING BUSINESS CENTRAL DASHBOARD DATA

You can use a ZIP file to import dashboard data that you created in Business Central to another Business Central instance if the archive is structured as shown in the following example:

```
dashbuilder/datasets/definitions/dataset-example1.dset
dashbuilder/datasets/definitions/dataset-example2.dset
dashbuilder/datasets/readme.md
dashbuilder/perspectives/page1/perspective_layout
dashbuilder/perspectives/page1/perspective_layout.plugin
dashbuilder/perspectives/page2/perspective_layout
dashbuilder/perspectives/page2/perspective_layout.plugin
dashbuilder/perspectives/readme.md
dashbuilder/navigation/navigation/navtree.json
dashbuilder/navigation/readme.md
VERSION
```

**Procedure**

1. In Business Central, select the **Admin** icon in the upper-right corner of the screen and select **Dashbuilder Data Transfer**.

   **WARNING**

   You must import dashboard data to a clean installation of Red Hat Process Automation Manager in order to avoid overwriting data on an existing system.

2. On the **Dashbuilder Data Transfer** page, click the **Choose File** icon.

3. Navigate to the ZIP file you want to import and select the file.

4. Click the **Upload** icon.

5. Click **Import**.
57.3. DEPLOYING DASHBOARDS FROM BUSINESS CENTRAL ON DASHBUILDER RUNTIME

You can automatically deploy the dashboards from Business Central on Dashbuilder Runtime. Business Central is linked to Dashbuilder Runtime using a gradual export feature.

Prerequisites

- Dashbuilder Runtime is configured on your system.
- You have set the `dashbuilder.runtime.multi` system property to `true` in the `standalone.xml` file.
- You have set the value of `dashbuilder.runtime.location` system property to the Dashbuilder Runtime URL, as shown in the following example:
  ```xml
  <property name="dashbuilder.runtime.location" value="http://localhost:8080"
  
  - You have set the `dashbuilder.export.dir` system property to the shared directory where Dashbuilder Runtime reads its model, as shown in the following example:
  ```xml
  <property name="dashbuilder.export.dir" value="/tmp/dashbuilder/models/"

Procedure

1. In Business Central, go to **Menu → Design → Pages**.
2. In the **Components** panel, drag and edit the required component types to the canvas as needed and click **Save** to finish.
3. Select the **Admin** icon in the upper-right corner of the screen and select **Dashbuilder Data Transfer**.
4. On the **Dashbuilder Data Transfer** page, click **Custom export**.
5. Select the pages that you want to include in the ZIP file in the **Export Wizard** panel, and click **Next**.
6. Select **Export Wizard** and click **Open**.
   You can see the Dashbuilder Runtime home page. If you are not logged in, you are redirected to the login page.
7. Go to **Dashboards → Runtime Dashboards** and you can see the pages.
   The selected data is exported and Dashbuilder Runtime updates the model content when it is opened.
APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Monday, February 14, 2022.
APPENDIX B. CONTACT INFORMATION

Red Hat Process Automation Manager documentation team: brms-docs@redhat.com