



# **Red Hat Process Automation Manager 7.0**

**Managing and monitoring Process Server**



# Red Hat Process Automation Manager 7.0 Managing and monitoring Process Server

---

Red Hat Customer Content Services  
[brms-docs@redhat.com](mailto:brms-docs@redhat.com)

## Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## Abstract

This document explains how install, configure, and performance tune Red Hat Process Automation Manager 7.0.

## Table of Contents

<b>PREFACE</b>	<b>4</b>
<b>CHAPTER 1. RED HAT PROCESS AUTOMATION MANAGER COMPONENTS</b>	<b>5</b>
<b>CHAPTER 2. SYSTEM INTEGRATION WITH MAVEN</b>	<b>6</b>
2.1. PRE-EMPTIVE AUTHENTICATION FOR LOCAL PROJECTS	6
2.2. DUPLICATE GAV DETECTION IN BUSINESS CENTRAL	7
2.3. MANAGING DUPLICATE GAV DETECTION SETTINGS IN BUSINESS CENTRAL	7
<b>CHAPTER 3. APPLYING PATCH UPDATES TO RED HAT PROCESS AUTOMATION MANAGER</b>	<b>8</b>
<b>CHAPTER 4. CONFIGURING AND STARTING PROCESS SERVER</b>	<b>11</b>
<b>CHAPTER 5. CONFIGURING JDBC DATA SOURCES FOR PROCESS SERVER</b>	<b>13</b>
<b>CHAPTER 6. CONFIGURING PROCESS SERVER WITH THE INTEGRATED PROCESS AUTOMATION MANAGER CONTROLLER</b>	<b>15</b>
<b>CHAPTER 7. INSTALLING AND RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER</b>	<b>17</b>
7.1. USING THE INSTALLER TO CONFIGURE PROCESS SERVER WITH THE PROCESS AUTOMATION MANAGER CONTROLLER	17
7.2. INSTALLING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER	18
7.2.1. Creating a controller user	19
7.2.2. Configuring Process Server and the headless Process Automation Manager controller	19
7.3. RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER	21
7.4. CLUSTERING WITH THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER	21
7.5. CONFIGURING THE HEADLESS RED HAT PROCESS AUTOMATION MANAGER CONTROLLER	23
<b>CHAPTER 8. CONFIGURING A PROCESS SERVER TO CONNECT TO BUSINESS CENTRAL</b>	<b>25</b>
<b>CHAPTER 9. CONFIGURING PROCESS SERVER MANAGED BY BUSINESS CENTRAL</b>	<b>27</b>
<b>CHAPTER 10. MANAGED PROCESS SERVER</b>	<b>30</b>
<b>CHAPTER 11. UNMANAGED PROCESS SERVER</b>	<b>31</b>
<b>CHAPTER 12. DEPLOYMENT DESCRIPTORS</b>	<b>32</b>
12.1. DEPLOYMENT DESCRIPTOR CONFIGURATION	32
What Can You Configure?	32
12.2. MANAGING DEPLOYMENT DESCRIPTORS	34
12.3. RESTRICTING ACCESS TO THE RUNTIME ENGINE	34
<b>CHAPTER 13. ACCESSING RUNTIME DATA FROM BUSINESS CENTRAL</b>	<b>35</b>
<b>CHAPTER 14. EXECUTION ERROR MANAGEMENT</b>	<b>36</b>
14.1. MANAGE EXECUTION ERRORS	36
14.2. THE EXECUTIONERRORHANDLER	37
14.3. EXECUTION ERROR STORAGE	37
14.4. ERROR TYPES AND FILTERS	37
14.5. AUTO ACKNOWLEDGING EXECUTION ERRORS	38
14.6. CLEANING UP THE ERROR LIST	40
<b>CHAPTER 15. CONFIGURING OPENSIFT CONNECTION TIMEOUT</b>	<b>42</b>
<b>CHAPTER 16. PERSISTENCE</b>	<b>43</b>
16.1. CONFIGURING SAFE POINTS	43

16.2. SESSION PERSISTENCE ENTITIES	44
16.3. PROCESS INSTANCE PERSISTENCE ENTITIES	44
16.4. WORK ITEM PERSISTENCE ENTITIES	45
16.5. CORRELATION KEY ENTITIES	45
16.6. CONTEXT MAPPING ENTITY	46
16.7. PESSIMISTIC LOCKING SUPPORT	47
<b>CHAPTER 17. DEFINE THE LDAP LOGIN DOMAIN</b>	<b>48</b>
<b>CHAPTER 18. AUTHENTICATING THIRD-PARTY CLIENTS THROUGH RH-SSO</b>	<b>49</b>
18.1. BASIC AUTHENTICATION	49
<b>CHAPTER 19. BOOTSTRAP SWITCHES</b>	<b>50</b>
<b>CHAPTER 20. SUPPORTED PROPERTIES</b>	<b>55</b>
<b>CHAPTER 21. RELATED INFORMATION</b>	<b>57</b>
<b>APPENDIX A. VERSIONING INFORMATION</b>	<b>58</b>



## PREFACE

As a systems administrator, you want to be able to 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

- Installed Red Hat JBoss Enterprise Application Platform 7.1.0. See [Red Hat JBoss EAP 7.1.0 Installation Guide](#).
- Installed Red Hat Process Automation Manager. For more information, see the [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 the [Planning a Red Hat Process Automation Manager installation](#).



# CHAPTER 1. RED HAT PROCESS AUTOMATION MANAGER COMPONENTS

Red Hat Process Automation Manager is made up of Business Central and Process 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 needing to deploy it to an application server.

- Process Server is the server where processes, rules, and other artifacts are executed. It is used to instantiate and execute processes and rules and solve planning problems. You can install Process Server in a Red Hat JBoss EAP instance, on OpenShift, in an Oracle WebLogic server instance, or an IBM WebSphere Application Server instance.  
You can configure Process Server to run in managed or unmanaged mode. If Process Server is unmanaged, you must manually create and maintain containers. A container is a specific version of a project. If Process Server is managed, the Process Automation Manager controller manages the Process Server configuration and you interact with the controller to create and maintain containers.

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

Red Hat Business Optimizer is integrated in Business Central and Process Server. It is a lightweight, embeddable planning engine that optimizes planning problems. Red Hat Business Optimizer helps Java programmers solve planning problems efficiently, and it combines optimization heuristics and metaheuristics with efficient score calculations.

## 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. PRE-EMPTIVE 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:

```
<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:

```
<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.

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.

### 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 TO RED HAT PROCESS AUTOMATION MANAGER

Patch updates of Red Hat Process Automation Manager typically include the latest security updates and bug fixes. Scheduled patch updates contain all previously released patch updates for that minor version of the product, so you do not need to apply each patch update incrementally in order to apply the latest update. For example, you can update Red Hat Process Automation Manager 7.0.0 or 7.0.1 to Red Hat Process Automation Manager 7.0.2. However, to keep your Red Hat Process Automation Manager distribution current with the latest fixes, apply patch updates to Red Hat Process Automation Manager as they become available in the Red Hat Customer Portal.

Automated patch tools are often provided with patch releases to facilitate updating certain components of Red Hat Process Automation Manager, such as Business Central, Process Server, and the headless Process Automation Manager controller. Other Red Hat Process Automation Manager artifacts, such as the process engine and standalone Business Central, are released as new artifacts with each patch update and you must re-install them to apply the update.



### NOTE

Only updates for Red Hat Process Automation Manager are included in Red Hat Process Automation Manager patch updates. Patches 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](#).

### Prerequisite

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

Example:

- **Product:** Process Automation Manager
  - **Version:** 7.0.1
2. Click **Patches**, download **Red Hat Process Automation Manager [VERSION] Patch Update**, and extract the downloaded **rhpm-\$VERSION-patch.zip** file to a temporary directory.  
This patch update tool automates the update of certain components of Red Hat Process Automation Manager, such as Business Central, Process Server, and the headless Process Automation Manager controller. Use this update tool first to apply updates and then install any other patch 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 patch update tool, navigate to the extracted **rhpm-\$VERSION-patch** folder, open the **blacklisted.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 patch 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 blacklist files that are no longer being distributed, the patch

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 **blacklisted.txt** file:

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

The contents of the blacklisted 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 **rhcam-\$VERSION-patch.zip** file and run the **apply-updates** script in the following format:



### IMPORTANT

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

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

- **business-central-eap7-deployable**: Updates Business Central (**business-central.war**)
- **kie-server-ee7**: Updates Process Server (**kie-server.war**)
- **controller-ee7**: Updates the headless Process Automation Manager controller (**controller.war**)

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

```
./apply-updates.sh ~EAP_HOME/standalone/deployments/business-
central.war business-central-eap7-deployable
```

```
./apply-updates.sh ~EAP_HOME/standalone/deployments/kie-server.war
kie-server-ee7
```

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

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

The patch update script creates a **backup** folder in the extracted **rhpm-\$VERSION-patch** 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 patch update tool and install any other patch 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 process 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 7.0.0 Maven Repository** artifact (**rhpm-7.0.0-maven-repository.zip**), such as in air-gap environments, download **Red Hat Process Automation Manager [VERSION] Incremental Maven Repository** and extract the downloaded **rhpm-\$VERSION-incremental-maven-repository.zip** file to your existing **~/maven-repository** directory to update the relevant contents.

Example:

```
$ unzip -o rhpm-7.0.1-incremental-maven-repository.zip -d
$REPO_PATH/rhpm-7.0.0-maven-repository/maven-repository/
```

7. After you finish applying all relevant updates, start Red Hat Process Automation Manager and Process Server and log in to Business Central.
8. 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 any patching errors or notice any missing data in Business Central, you can restore the contents in the **backup** folder within the **rhpm-\$VERSION-patch** folder to revert the patch update tool changes. You can also re-install 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 patch update.

## CHAPTER 4. CONFIGURING AND STARTING PROCESS SERVER

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

### Procedure

Navigate to the Red Hat Process Automation Manager 7.0 **bin** directory and start the new Process Server with the following properties. Adjust the specific properties according to your environment.

```
$ ~/EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml 1
-Dorg.kie.server.id=myserver 2
-Dorg.kie.server.user=process_server_username 3
-Dorg.kie.server.pwd=process_server_password 4
-Dorg.kie.server.controller=http://localhost:8080/business-
central/rest/controller 5
-Dorg.kie.server.controller.user=controller_username 6
-Dorg.kie.server.controller.pwd=controller_password 7
-Dorg.kie.server.location=http://localhost:8080/kie-
server/services/rest/server 8
-
Dorg.kie.server.persistence.dialect=org.hibernate.dialect.PostgreSQLDialect 9
-Dorg.kie.server.persistence.ds=java:jboss/datasources/psjbpmDS 10
```

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

Example:

```
-Djboss.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 Process Server port, with a defined offset of 150, is 8230.

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



## CHAPTER 5. CONFIGURING JDBC DATA SOURCES FOR PROCESS SERVER

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 Process Server to ensure proper data exchange between the servers and the designated database.

### Prerequisite

The JDBC providers that you want to use to create database connections are configured on all servers on which you want to deploy Process Server.

### Procedure

1. Open **EAP\_HOME/standalone/configuration/standalone-full.xml** in a text editor and locate the **<system-properties>** tag.
2. Add the following properties to the **<system-properties>** tag where **<DATASOURCE>** is the name of your data source and **<HIBERNATE\_DIALECT>** is the hibernate dialect for your database.



### NOTE

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**.

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

For example:

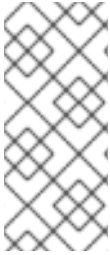
```
<system-properties>
  <property name="org.kie.server.repo"
value="${jboss.server.data.dir}"/>
  <property name="org.kie.example" value="true"/>
  <property name="org.jbpm.designer.perspective" value="full"/>
  <property name="designerdataobjects" value="false"/>
  <property name="org.kie.server.user" 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"/>
```

```
<!-- Data source properties. -->
<property name="org.kie.server.persistence.ds"
value="java:jboss/datasources/KieServerDS"/>
<property name="org.kie.server.persistence.dialect"
value="org.hibernate.dialect.PostgreSQLDialect"/>
</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. CONFIGURING PROCESS SERVER WITH THE INTEGRATED PROCESS AUTOMATION MANAGER CONTROLLER



### NOTE

Only make the changes described in this section if Process Server will be managed by Business Central and you installed Red Hat Process Automation Manager from the ZIP files. If you did not install Business Central, you can use the headless Process Automation Manager controller to manage Process Server, as described in [Chapter 7, Installing and running the headless Process Automation Manager controller](#).

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

If you installed Red Hat Process Automation Manager from the ZIP files, you must edit the **standalone-full.xml** file in both the Process Server and Business Central installations to configure Process Server with the integrated Process Automation Manager controller.

### Prerequisites

- Business Central and Process Server are installed in the base directory of the Red Hat JBoss EAP installation (**EAP\_HOME**).



### NOTE

You should install Business Central and Process Server on different servers in production environments. However, if you install Process Server and Business Central on the same server, for example in a development environment, make the changes described in this section in the shared **standalone-full.xml** file.

- On Business Central server nodes, a user with the **rest-all** role exists.

### Procedure

1. In the Business Central **EAP\_HOME/standalone/configuration/standalone-full.xml** file, uncomment the following properties in the **<system-properties>** section and replace **<USERNAME>** and **<USER\_PWD>** with the credentials of a user with the **kie-server** role:

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

2. In the Process Server **EAP\_HOME/standalone/configuration/standalone-full.xml** file, uncomment the following properties in the **<system-properties>** section.

```

    <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. Replace the following values:

- Replace **<CONTROLLER\_USER>** and **<CONTROLLER\_PWD>** with the credentials of a user with the **rest-all** role.
- Replace **<KIE\_SERVER\_ID>** with the ID or name of the Process Server installation, for example, **rhpm700-process-server-1**.
- Replace **<HOST>** with the ID or name of the Process Server host, for example, **localhost** or **192.7.8.9**.
- Replace **<PORT>** with the port of the Process Server host, for example, **8080**.



#### NOTE

The **org.kie.server.location** property specifies the location of Process Server.

- Replace **<CONTROLLER\_URL>** with the URL of Business Central. Process Server connects to this URL during startup.
  - If you installed Business Central using the installer or Red Hat JBoss EAP zip installations, **<CONTROLLER\_URL>** has this format:  
<http://<HOST>:<PORT>/business-central/rest/controller>
  - If you are running Business Central using the **standalone.jar** file, **<CONTROLLER\_URL>** has this format:  
<http://<HOST>:<PORT>/rest/controller>

## CHAPTER 7. INSTALLING AND RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

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

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

### 7.1. USING THE INSTALLER TO CONFIGURE PROCESS SERVER WITH THE PROCESS AUTOMATION MANAGER CONTROLLER

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

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



#### NOTE

If you do not install Business Central, see [Chapter 7, \*Installing and running the headless Process Automation Manager controller\*](#) for information about using the headless Process Automation Manager controller .

#### Prerequisites

- Two computers with backed-up Red Hat JBoss EAP 7.1 or higher 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 7.1](#) for more information.
2. On the **Component Selection** page, clear the **Process 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.

6. On the **Configure Runtime Environment** page, select **Perform Advanced Configuration**.
7. Select **Customize Process Server properties** and click **Next**.
8. On the **Process Server Properties Configuration** page, click **New Server Configuration** to add a Process Server and specify a unique name for that Process Server. This name will appear in Business Central and enable you to distinguish between different Process Servers.

## 7.2. INSTALLING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

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

### Prerequisites

- A backed-up Red Hat JBoss EAP installation version 7.1 or higher 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: Red Hat Process Automation Manager
  - Version: 7.0
2. Download **Red Hat Process Automation Manager 7.0.0 Add Ons** (the **rhcam-7.0.0-add-ons.zip** file).
3. Unzip the **rhcam-7.0.0-add-ons.zip** file. The **rhcam-7.0-controller-ee7.zip** file is in the unzipped directory.
4. Extract the **rhcam-7.0-controller-ee7** archive to a temporary directory. In the following examples this directory is called **TEMP\_DIR**.
5. Copy the **TEMP\_DIR/rhcam-7.0-controller-ee7/controller.war** directory to **EAP\_HOME/standalone/deployments/**.



### WARNING

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

6. Copy the contents of the **TEMP\_DIR/rhcam-7.0-controller-ee7/SecurityPolicy/** directory to **EAP\_HOME/bin**. When asked to overwrite files, select **Yes**.

7. 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.

### 7.2.1. Creating a controller user

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

#### Prerequisite

The 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 ***<USER\_NAME>*** and ***<PASSWORD>*** with the user name and password of your choice.

```
$ ./add-user.sh -a --user <USER_NAME> --password <PASSWORD> --role
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.

### 7.2.2. Configuring Process Server and the headless Process Automation Manager controller

If Process Server will be managed by the headless Process Automation Manager controller, you must edit the ***standalone-full.xml*** file in both the Process Server and headless Process Automation Manager controller installations, as described in this section.

#### Prerequisites

- KIE\_SERVER} is installed in the base directory of the Red Hat JBoss EAP installation (***EAP\_HOME***).
- The Controller is installed in an ***EAP\_HOME***.



## NOTE

You should install Process Server and the headless Process Automation Manager controller on different servers in production environments. However, if you install Process 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 Process Server nodes, a user with the **kie-server** role exists.
- On the controller server nodes, a user with the **kie-server** role exists.

## Procedure

1. In the controller **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:

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

2. In the Process Server **EAP\_HOME/standalone/configuration/standalone-full.xml** file, add the following properties to the **<system-properties>** section:

```
<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 Process Server installation, for example, **rhcam-7.0.0-process\_server-1**.
- Replace **<HOST>** with the ID or name of the Process Server host, for example, **localhost** or **192.7.8.9**.
- Replace **<PORT>** with the port of the Process Server host, for example, **8080**.



## NOTE

The **org.kie.server.location** property specifies the location of Process Server.



- Replace **<CONTROLLER\_URL>** with the URL of the headless Process Automation Manager controller
  1. Process Server connects to this URL during startup.

## 7.3. 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.

### Prerequisite

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. Enter the following command:
  - On Linux or UNIX-based systems:
 

```
$ ./standalone.sh
```
  - On Windows:
 

```
standalone.bat
```
3. To verify that the 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 Process Server instance.

```
curl -X GET "http://<HOST>:  
<PORT>/controller/rest/controller/management/servers" -H "accept:  
application/xml" -u '<CONTROLLER>:<CONTROLLER_PWD>'
```



### NOTE

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

## 7.4. CLUSTERING 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 Process Server Java Client API to interact with it.

### Prerequisites

- A backed-up Red Hat JBoss EAP installation version 7.1 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 mounted partition is available.

## 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: 7.0**
2. Download **Red Hat Process Automation Manager 7.0.0 Add Ons** (the **rhpm-7.0.0-add-ons.zip** file).
3. Unzip the **rhpm-7.0.0-add-ons.zip** file. The **rhpm-7.0-controller-ee7.zip** file is in the unzipped directory.
4. Extract the **rhpm-7.0-controller-ee7.zip** file to a temporary directory. In the following examples this directory is called **TEMP\_DIR**.
5. Repackage the **controller.war** directory:
  - a. Navigate to the **TEMP\_DIR/rhpm-7.0.0-add-ons/rhpm-7.0-controller-ee7/controller.war** directory.
  - b. Select the contents of the **TEMP\_DIR/rhpm-7.0.0-add-ons/rhpm-7.0-controller-ee7/controller.war** directory and create the **controller.zip** file.
  - c. Rename **controller.zip** to **controller.war**. This is the file that you will use to deploy the controller on the cluster nodes.
  - d. If desired, copy the new **controller.war** file to a location that is more convenient to deploy from.
6. If you want to use a security manager with the controller, copy the contents of the **TEMP\_DIR/rhpm-7.0.0-add-ons/rhpm-7.0-controller-ee7/SecurityPolicy** directory to the **EAP\_HOME/bin** directory on each node of the cluster.
7. Add Red Hat JBoss EAP management users to the master node (where you configured the **domain.xml** file) as described in the [Red Hat JBoss EAP 7.1 Configuration Guide](#).
8. On each node of the cluster, add users for the headless Process Automation Manager controller as described in the "Installing the headless Process Automation Manager controller" section of the [Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.1](#).
9. Complete the following steps in the **host.xml** file on the master node and in the **host-slave.xml** file on each slave node:
  - a. Open the **EAP\_HOME/domain/configuration/host.xml** or **EAP\_HOME/domain/configuration/host-slave.xml** file in a text editor.

- b. In the **main-server-group** `<servers>` element, add the servers that will be part of the cluster.
- c. 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>
```

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.

10. To deploy the **controller.war** file that you created previously into the server group, complete the following steps on the master node:
  - a. Log in to the Red Hat JBoss EAP **Administration** console of your domain as a **management** user.
  - b. Click **Deployments** → **Server Groups** → **main-server-group** and click **Add**.
  - c. In the dialog box, click **Upload a new deployment** and click **Next**.
  - d. In the **Upload Deployments** dialog box, click **Browse**, select the **controller.war** file, and click **Next**.
  - e. Click **Enable** and click **Next**.



#### NOTE

Make sure to check deployment unit readiness with every cluster member.

When a deployment unit is created on a cluster node, it takes some time before it is distributed among all cluster members. You can check the deployment status using the server Administration console or REST API. However, if the query is sent to the node where the deployment was originally issued, the query will return a value of **deployed**. If the query is sent to a node where the deployment has not yet been distributed, the query returns **DeploymentNotFoundException**.

For more information about installing Business Central, see [Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.1](#).

## 7.5. CONFIGURING THE HEADLESS RED HAT PROCESS AUTOMATION MANAGER CONTROLLER

## Prerequisites

- Process Server is installed on each node of a Red Hat JBoss EAP 7.1 cluster.
- An NFS server with a mounted partition accessible to a Red Hat JBoss EAP user is available.

## Procedure

On the java process running Smart Router, do the following:

1. Enable the Process Server watcher service system property:

```
org.kie.server.controller.templatefile.watcher.enabled=true
```

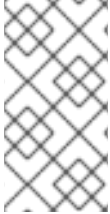
2. In the **org.kie.server.controller.templatefile** property, specify the absolute path to the NFS storage where the memory configuration is stored:

```
org.kie.server.controller.templatefile=  
<absolute_path_to_NFS_storage>
```

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 7.1 clustered environment](#).

## CHAPTER 8. CONFIGURING A PROCESS SERVER TO CONNECT TO BUSINESS CENTRAL

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



### NOTE

If you are deploying Process Server on Red Hat OpenShift Container Platform, see [Deploying a Red Hat Process Automation Manager managed server environment on Red Hat OpenShift Container Platform](#) for instructions about configuring it to connect to Business Central.

### Prerequisite

Process Server is installed. For installation options, see [Planning a Red Hat Process Automation Manager installation](#).

### 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 **standalone-full.xml** and under the **<system-properties>** tag, set the following properties:
  - **org.kie.server.controller.user:** The user name of a user who can log in to the Business Central.
  - **org.kie.server.controller.pwd:** The password of the user who can log in to the Business Central.
  - **org.kie.server.controller:** The URL for connecting to the API of Business Central. Normally, the URL is **http://<centralhost>:<centralport>/business-central/rest/controller**, where **<centralhost>** and **<centralport>** are the host name and port for Business Central. If Business Central is deployed on OpenShift, remove **business-central/** from the URL.
  - **org.kie.server.location:** The URL for connecting to the API of Process Server. Normally, the URL is **http://<serverhost>:<serverport>/kie-server/services/rest/server**, where **<serverhost>** and **<serverport>** are the host name and port for Process Server.
  - **org.kie.server.id:** The name of a server configuration. If this server configuration does not exist in Business Central, it is created automatically when Process Server connects to Business Central.

Example:

```
<property name="org.kie.server.controller.user"
value="central_user"/>
<property name="org.kie.server.controller.password"
```

```
value="central_password"/>
<property name="org.kie.server.controller"
value="http://central.example.com:8080/business-
central/rest/controller"/>
<property name="org.kie.server.location"
value="http://kieserver.example.com:8080/kie-
server/services/rest/server"/>
<property name="org.kie.server.id" value="production-servers"/>
```

3. Start or restart the Process Server.

## CHAPTER 9. CONFIGURING PROCESS SERVER MANAGED BY 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.

Use this procedure to configure Business Central to manage a Process Server instance.

### Prerequisite

Users with the following roles exist:

- In Business Central, a user with the role **rest-all**.
- On the Process Server, a user with the role **kie-server**.



### NOTE

In production environments, use two distinct users, each with one role. In this sample situation, we use only one user named **controllerUser** that has both the **rest-all** and the **kie-server** roles.

### Procedure

1. Set the following JVM properties.  
The location of Business Central and the Process Server may be different. In such case, ensure you set the properties on the correct server instances.
  - On Red Hat JBoss EAP, modify the **<system-properties>** section in:
    - **EAP\_HOME/standalone/configuration/standalone\*.xml** for standalone mode.
    - **EAP\_HOME/domain/configuration/domain.xml** for domain mode.

**Table 9.1. JVM Properties for Process Server Instance**

Property	Value	Note
<b>org.kie.server.id</b>	<b>default-kie-server</b>	The Process Server ID.
<b>org.kie.server.controller</b>	<a href="http://localhost:8080/decision-central/rest/controller">http://localhost:8080/decision-central/rest/controller</a>	The location of Business Central.

Property	Value	Note
<code>org.kie.server.controller.user</code>	<code>controllerUser</code>	The user name with the role <b>rest-all</b> as mentioned in the previous step.
<code>org.kie.server.controller.pwd</code>	<code>controllerUser1234;</code>	The password of the user mentioned in the previous step.
<code>org.kie.server.location</code>	<code>http://localhost:8080/kie-server/services/rest/server</code>	The location of the Process Server.

Table 9.2. JVM Properties for Business Central Instance

Property	Value	Note
<code>org.kie.server.user</code>	<code>controllerUser</code>	The user name with the role <b>kie-server</b> as mentioned in the previous step.
<code>org.kie.server.pwd</code>	<code>controllerUser1234;</code>	The password of the user mentioned in the previous step.

2. Verify the successful start of the Process Server by sending a GET request to `http://SERVER:PORT/kie-server/services/rest/server/`. Once authenticated, you get an XML response similar to this:

```
<response type="SUCCESS" msg="Kie Server info">
  <kie-server-info>
    <capabilities>KieServer</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>
```

3. Verify successful registration:

- a. Log in to Business Central.
- b. Click **Menu** → **Deploy** → **Execution Servers**.  
If registration is successful, you can see the registered server ID.

## CHAPTER 10. MANAGED PROCESS SERVER

A managed instance requires an available controller to start the Process Server.

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



### IMPORTANT

Controllers should be synchronized to ensure that the same set of configuration is provided to the server, regardless of the controller to which it connects.

When the Process Server is configured with a list of 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 consistence and prevents the server from running with redundant configuration.



### NOTE

To run the Process Server in standalone mode without connecting to controllers, see [Chapter 11, \*Unmanaged Process Server\*](#).

## CHAPTER 11. UNMANAGED PROCESS SERVER

An unmanaged Process Server is a standalone instance, and therefore must be configured individually using REST/JMS API from the Process Server itself. There is no controller involved. 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 the Process Server is restarted, it will attempt to re-establish the same state that was persisted before shutdown. Therefore, KIE Containers that were running will be started, but the ones that were stopped will not.

## CHAPTER 12. DEPLOYMENT DESCRIPTORS

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 **META-INF** directory of the KJAR called **kmodule.xml** can be used to define the knowledge bases and sessions. This **kmodule.xml** file, by default, is empty.

Whenever a runtime component such as Business Central is about to process the KJAR, it looks up **kmodule.xml** to build the runtime representation.

Deployment descriptors supplement the **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 the Process Server on multiple levels (server level default, different deployment descriptor per KJAR, and other server configurations). This allows you to make simple customizations to the default Process Server configuration (possibly per KJAR).

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

```
-Dorg.kie.deployment.desc.location=file:/path/to/file/company-deployment-descriptor.xml
```

### 12.1. DEPLOYMENT DESCRIPTOR CONFIGURATION

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

- *Server level*: The main level and the one that applies to all KJARs deployed on the server.
- *KJAR level*: This allows you to configure descriptors on a per KJAR basis.
- *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: *deploy time configuration > KJAR configuration > server configuration*.



#### NOTE

The deploy time configuration applies to deployments done via the REST API.

For example, if the persistence mode (one of the items you can configure) defined at the server level is **NONE** but the same mode is specified as **JPA** at the KJAR level, the actual mode will be **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 **NONE** (or to **JPA** if there is no server level deployment descriptor).

#### 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.

Table 12.1. Deployment Descriptors

Configuration	XML Entry	Permissible Values	Default Value
Persistence unit name for runtime data	persistence-unit	Any valid persistence package name	org.jbpm.domain
Persistence unit name for audit data	audit-persistence-unit	Any valid persistence package name	org.jbpm.domain
Persistence mode	persistence-mode	JPA, NONE	JPA
Audit mode	audit-mode	JPA, JMS or NONE	JPA
Runtime Strategy	runtime-strategy	SINGLETON, PER_REQUEST or PER_PROCESS_INSTANCE	SINGLETON
List of Event Listeners to be registered	event-listeners	Valid listener class names as <b>ObjectModel</b>	No default value
List of Task Event Listeners to be registered	task-event-listeners	Valid listener class names as <b>ObjectModel</b>	No default value
List of Work Item Handlers to be registered	work-item-handlers	Valid Work Item Handler classes given as <b>NamedObjectHandler</b>	No default value
List of Globals to be registered	globals	Valid Global variables given as <b>NamedObjectModel</b>	No default value
Marshalling strategies to be registered (for pluggable variable persistence)	marshalling-strategies	Valid <b>ObjectModel</b> classes	No default value
Required Roles to be granted access to the resources of the KJAR	required-roles	String role names	No default value
Additional Environment Entries for Knowledge Session	environment-entries	Valid <b>NamedObjectModel</b>	No default value

Configuration	XML Entry	Permissible Values	Default Value
Additional configuration options of Knowledge Session	configurations	Valid <b>NamedObjectModel</b>	No default value
Classes used for serialization in the remote services	remoteable-class	Valid <b>CustomClass</b>	No default value

## 12.2. 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.

## 12.3. 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**.
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 the previous steps.
5. When you have finished adding all required roles, click **Save**.

## CHAPTER 13. ACCESSING RUNTIME DATA FROM BUSINESS CENTRAL

The following pages in Business Central allow you to view the runtime data of the Process Server:

- **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 the {KIE\_SERVER}. Therefore, to be able to view the runtime data in {central}, ensure that the following conditions are met:

- The user exists in the container 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 container running the Process Server and has **kie-server** role assigned.
- Communication between Business Central and the Process Server is established. That is, the Process Server is registered in the server controller, which is part of Business Central.
- The **deployment.business-central.war** login module is present in the **standalone.xml** configuration of the server running Business Central:

```
<login-module code="org.kie.security.jaas.KieLoginModule"
flag="optional" module="deployment.business-central.war"/>
```

## CHAPTER 14. EXECUTION ERROR MANAGEMENT

When an execution error occurs the process stops and rolls back to the most recent stable state (the closest safe point) and continues its execution. If an error of any kind is not handled by the process the entire transaction rolls back, leaving the process instance in the previous wait state. Any trace of this is only visible in the logs, and usually displayed to the caller who sent the request to the process engine.

Users with process administrator (**process-admin**) or administrator (**admin**) roles are able to access error messages in Business Central, which has the following features:

- Better traceability
- Visibility in case of critical processes
- Reporting and analytics based on error situations
- External system error handling and compensation

Configurable error handling is responsible for receiving any technical errors thrown throughout the process engine execution (including task service). The following technical exceptions apply:

- Anything that extends **java.lang.Throwable**.
- Process level error handling and any other exceptions not previously handled.

There are several components that make up the error handling mechanism and allow a pluggable approach to extend its capabilities.

The process engine entry point for error handling is the **ExecutionErrorManager**. This is integrated with **RuntimeManager**, which is then responsible for providing it to underlying components - **KieSession** and **TaskService**. From the API point of view, **ExecutionErrorManager** gives access to:

- **ExecutionErrorHandler** - the primary mechanism for error handling.
- **ExecutionErrorStorage** - pluggable storage for execution error information.

### 14.1. MANAGE EXECUTION ERRORS

By definition, every error that is caught and stored is unacknowledged, meaning that it is to be handled by someone or something (in case of automatic error recovery). Errors are filtered on the basis of whether or not they have been acknowledged. Acknowledging an error saves the user information and time stamp for traceability.

You can access the **Error Management** view at any time.

1. In Business Central, click **Menu** → **Manage** → **Execution Errors**.
2. Select an error from the list to open the **Details** tab. This displays information about the error or errors.
3. Click the **Acknowledge** button to acknowledge and clear the error. The error can still be viewed later by selecting **Yes** on the **Acknowledged** filter in the **Manage Execution Errors** page.
4. If the error was related to a task, a **Go to Task** button is displayed.  
Click the **Go to Task** button to view the associated job information in the **Manage Tasks** page.



The **Manage Tasks** page allows you to restart, reschedule, or retry the corresponding task.

## 14.2. THE EXECUTIONERRORHANDLER

The **ExecutionErrorHandler** is the primary mechanism for all process error handling. It is bound to the life cycle of **RuntimeEngine**; meaning it is created when a new runtime engine is created, and is destroyed when **RuntimeEngine** is disposed. A single instance of the **ExecutionErrorHandler** is used within a given execution context or transaction. Both **KieSession** and **TaskService** use that instance to inform the error handling about processed nodes/tasks. **ExecutionErrorHandler** is informed about:

- Starting of processing of a given node instance.
- Completion of processing of a given node instance.
- Starting of processing of a given task instance.
- Completion of processing of a given task instance.

This information is mainly used for errors that are of unknown type; that is, errors that do not provide information about the process context. For example, upon commit time, database exceptions do not carry any process information.

## 14.3. EXECUTION ERROR STORAGE

**ExecutionErrorStorage** is a pluggable strategy that permits various ways of persisting information about execution errors. Storage is used directly by the handler that gets an instance of the store when it is created (when **RuntimeEngine** is created). Default storage implementation is based on the database table, which stores every error and includes all of the available information. Some errors may not contain details, as this depends on the type of error and whether or not it is possible to extract specific information.

## 14.4. ERROR TYPES AND FILTERS

Error handling attempts to catch and handle any kind of error, therefore it needs a way to categorize errors. By doing this, it is able to properly extract information from the error and make it pluggable, as some users may require specific types of errors to be thrown and handled in different ways than what is provided by default.

Error categorization and filtering is based on **ExecutionErrorFilters**. This interface is solely responsible for building instances of **ExecutionError**, which are later stored by way of the **ExecutionErrorStorage** strategy. It has following methods:

- **accept**: indicates if given error can be handled by the filter.
- **filter**: where the actual filtering, handling, and so on happens.
- **getPriority**: indicates the priority that is used when calling filters.

As only one filter can process given error, filters use a priority system to avoid having multiple filters returning alternative “views” of the same error. Priority allows more specialized filters to see if the error can be accepted, or otherwise allow another filter to handle it.

**ExecutionErrorFilter** can be provided using the **ServiceLoader** mechanism, which allows the capability of error handling to be easily extended.

Red Hat Process Automation Manager ships with the following **ExecutionErrorFilters** :

**Table 14.1. ExecutionErrorFilters**

Class name	Type	Priority
org.jbpm.runtime.manager.impl.error.filters.ProcessExecutionErrorFilter	Process	100
org.jbpm.runtime.manager.impl.error.filters.TaskExecutionErrorFilter	Task	80
org.jbpm.runtime.manager.impl.error.filters.DBExecutionErrorFilter	DB	200
org.jbpm.executor.impl.error.JobExecutionErrorFilter	Job	100

Filters are given a higher execution order based on the lowest value of the priority. In above table, filters are invoked in following order:

1. Task
2. Process
3. Job
4. DB

## 14.5. AUTO ACKNOWLEDGING EXECUTION ERRORS

When executions errors occur they are unacknowledged by default, and require a manual acknowledgment to be performed otherwise they are always seen as information that requires attention. In case of larger volumes, manual actions can be time consuming and not suitable in some situations.

Auto acknowledgment resolves this issue. It is based on scheduled jobs by way of the **jbpm-executor**, with the following three types of jobs available:

### **org.jbpm.executor.commands.error.JobAutoAckErrorCommand**

Responsible for finding jobs that previously failed but now are either canceled, completed, or rescheduled for another execution. This job only acknowledges execution errors of type **Job**.

### **org.jbpm.executor.commands.error.TaskAutoAckErrorCommand**

Responsible for auto acknowledgment of user task execution errors for tasks that previously failed but now are in one of the exit states (completed, failed, exited, obsolete). This job only acknowledges execution errors of type **Task**.

### **org.jbpm.executor.commands.error.ProcessAutoAckErrorCommand**

Responsible for auto acknowledgment of process instances that have errors attached. It acknowledges errors where the process instance is already finished (completed or aborted), or the task that the error originated from is already finished. This is based on **init\_activity\_id** value.

This job acknowledges any type of execution error that matches the above criteria.

Jobs can be registered on the Process Server. In Business Central you can configure auto acknowledge jobs for errors:

### Prerequisite

1. Execution errors of one or more type have accumulated during processes execution but require no further attention.

### Procedure

1. In Business Central, click **Menu** → **Manage** → **Jobs**.
2. In the top right of the screen, click **New Job**.
3. Type the process correlation key into the **Business Key** field.
4. In the **Type** field, add type of the auto acknowledge job type from the list above.
5. Select a **Due On** time for the job to be completed:
  - a. To run the job immediately, select the **Run now** option.
  - b. To run the job at a specific time, select **Run later**. A date and time field appears next to the **Run later** option. Click the field to open the calendar and schedule a specific time and date for the job.

New Job

×

Basic

Advanced

Business Key \*

0000000001

Due On

☐ Run now
   
☒ Run later
 

24-Aug-2018 16:30:00

Type \*

org.jbpm.executor.commands.error.JobAutoAckErrorCommand

Retries \*

3

Cancel

Create

6. Click **Create** to create the job and return to the **Manage Jobs** page.

The following steps are optional, and allow you to configure auto acknowledge jobs to run either once (**SingleRun**), on specific time intervals (**NextRun**), or using the custom name of an entity manager factory to search for jobs to acknowledge (**EmfName**).

1. Click on the **Advanced** tab.
2. Click the **Add Parameter** button.

3. Enter the configuration parameter you want to apply to the job:

- a. **SingleRun**: **true** or **false**
- b. **NextRun**: time expression, such as 2h, 5d, 1m, and so on.
- c. **EmfName**: custom entity manager factory name.

**New Job**
×

Basic
Advanced

Key	Value	Actions
NextRun	1d	<div style="background-color: #dc3545; color: white; padding: 2px 5px; border-radius: 3px;"> <span>🗑️ Remove</span> </div>

Add Parameter

+ Create

## 14.6. CLEANING UP THE ERROR LIST

The **ExecutionErrorInfo** error list table can be cleaned up to remove redundant information. Depending on the life cycle of the process, errors may remain in the list for some time, and there is no direct API with which to clean up the list. Instead, **jbpmm-executor** commands can be scheduled to periodically clean up errors.

The following options can be used as clean up commands, and are restricted to deleting execution errors of already completed or aborted process instances:

- **DateFormat**
  - Date format for further date related parameters - if not given **yyyy-MM-dd** is used (pattern of **SimpleDateFormat** class).
- **EmfName**
  - Name of the entity manager factory to be used for queries (valid persistence unit name).
- **SingleRun**
  - Indicates if execution should be single run only (**true|false**).
- **NextRun**

- Provides next execution time (valid time expression, for example: 1d, 5h, and so on)
- **OlderThan**
  - Indicates what errors should be deleted - older than given date.
- **OlderThanPeriod**
  - Indicated what errors should be deleted older than given time expression (valid time expression e.g. 1d, 5h, and so on)
- **ForProcess**
  - Indicates errors to be deleted only for given process definition.
- **ForProcessInstance**
  - Indicates errors to be deleted only for given process instance.
- **ForDeployment**
  - Indicates errors to be deleted that are from given deployment ID.

## CHAPTER 15. CONFIGURING OPENSIFT 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:

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

For a full list of global route-specific timeout annotations, see the [OpenShift Documentation](#).

## CHAPTER 16. 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.

### 16.1. CONFIGURING SAFE POINTS

To allow persistence, add the **jbpmm-persistence** JAR files to the classpath of your application and configure the engine to use persistence. The engine automatically stores the runtime state in the storage when the engine reaches a safe point.

Safe points are points where the process instance has paused. When a process instance invocation reaches a safe point in the engine, the engine stores any changes to the process instance as a snapshot of the process runtime data. However, when a process instance is completed, the persisted snapshot of process instance runtime data is automatically deleted.

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

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

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

## 16.2. SESSION PERSISTENCE ENTITIES

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

**Table 16.1. SessionInfo**

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

## 16.3. 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 16.2. ProcessInstanceInfo**

Field	Description	Nullable
<b>instanceId</b>	The primary key.	NOT NULL
<b>lastModificationDate</b>	The last time that the entity was saved to a database.	
<b>lastReadDate</b>	The last time that the entity was retrieved from the database.	
<b>processId</b>	The ID of the process.	
<b>processInstanceByteArray</b>	The state of a process instance in form of a binary data set.	NOT NULL
<b>startDate</b>	The start time of the process.	
<b>state</b>	An integer representing the state of a process instance.	NOT NULL



Field	Description	Nullable
<b>OPTLOCK</b>	A version field containing a lock value.	

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

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

**Table 16.3. EventTypes**

Field	Description	Nullable
<b>instanceId</b>	A reference to the <b>ProcessInstanceInfo</b> primary key and foreign key constraint on this column.	NOT NULL
<b>element</b>	A finished event in the process.	

## 16.4. 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 16.4. WorkItemInfo**

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

## 16.5. 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 16.5. CorrelationKeyInfo**

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

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

**Table 16.6. CorrelationPropertyInfo**

Field	Description	Nullable
<b>propertyId</b>	The primary key.	NOT NULL
<b>name</b>	The name of the property.	
<b>value</b>	The value of the property.	NOT NULL
<b>OPTLOCK</b>	A version field containing a lock value.	
<b>correlationKey_keyId</b>	A foreign key mapped to the correlation key.	NOT NULL

## 16.6. 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 16.7. ContextMappingInfo**

Field	Description	Nullable
<b>mappingId</b>	The primary key.	NOT NULL

Field	Description	Nullable
<b>CONTEXT_ID</b>	The context identifier.	NOT NULL
<b>KSESSION_ID</b>	The <b>KieSession</b> identifier.	NOT NULL
<b>OPTLOCK</b>	A version field containing a lock value.	
<b>OWNER_ID</b>	Holds the identifier of the deployment unit that the given mapping is associated with	

## 16.7. 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.

This can be changed at runtime to allow the user to set locking on a per process basis and to allow it to be *pessimistic* (the change can be made 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:

```
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); ❶

RuntimeManager manager =
RuntimeManagerFactory.Factory.get().newPerRequestRuntimeManager(enviro
nment); ❷
```

❶ **env** is an instance of **org.kie.api.runtime.Environment**.

❷ Create your Runtime Manager by using this environment.

## CHAPTER 17. 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. This is because the Git SSH authentication may be using another security domain, in which case you may face authentication failure.

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 below:

```
<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 18. AUTHENTICATING THIRD-PARTY CLIENTS THROUGH RH-SSO

To use the different remote services provided by Business Central or by Process 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 Process 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

### 18.1. BASIC AUTHENTICATION

If you enabled basic authentication in the RH-SSO client adapter configuration for both Business Central and Process 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:

```
curl http://admin:password@localhost:8080/decision-  
central/rest/repositories
```

- For Process Server:

```
curl http://admin:password@localhost:8080/kie-execution-  
server/services/rest/server/
```

For more information about configuring Red Hat Process Automation Manager single sign-on and token-based authentication, see [Integrating Red Hat Process Automation Manager with Red Hat Single Sign-On](#).

## CHAPTER 19. BOOTSTRAP SWITCHES

The Process Server accepts a number of bootstrap switches (system properties) to configure the behavior of the server.

**Table 19.1. Bootstrap Switches for Disabling Process Server Extensions**

Property	Values	Default	Description
<code>org.drools.server.ext.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	If set to <code>true</code> , disables the Red Hat Decision Manager support (for example rules support).
<code>org.jbpm.server.ext.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	If set to <code>true</code> , disables the Red Hat Process Automation Manager support (for example processes support).
<code>org.optaplanner.server.ext.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	If set to <code>true</code> , disables the Red Hat Business Optimizer support.
<code>org.jbpm.ui.server.ext.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	If set to <code>true</code> , disables the Process Server UI extension.
<code>org.kie.executor.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	Disables the Red Hat Process Automation Manager executor.



### NOTE

Some controller properties listed below are marked as required. Set these properties when you handle Process Server container creation and removal in Business Central. If you use the Process Server separately without any interaction with Business Central, the properties do not have to be set.

**Table 19.2. Bootstrap Switches Required for Using a Controller**

Property	Values	Default	Description
<code>org.kie.server.id</code>	String	N/A	An arbitrary ID to be assigned to the server. If a remote controller is configured, this is the ID under which the server will connect to the controller to fetch the KIE container configurations. If not provided, the ID is automatically generated.
<code>org.kie.server.user</code>	String	<code>kieserver</code>	The user name used to connect with the Process Server from the controller, required when running in managed mode. Set this property in Business Central system properties. Set this property when using a controller.

Property	Values	Default	Description
<b>org.kie.server.pwd</b>	String	<b>kieserve r1!</b>	The password used to connect with the Process Server from the controller, required when running in managed mode. Set this property in Business Central system properties. Set this property when using a controller.
<b>org.kie.server.token</b>	String	N/A	A property that enables you to use token-based authentication between the controller and the Process Server instead of the basic user name/password authentication. The controller sends the token as a parameter in the request header. The server requires long-lived access tokens as the tokens are not refreshed.
<b>org.kie.server.location</b>	URL	N/A	The URL of the Process Server instance used by the 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 controller.
<b>org.kie.server.controller</b>	Comma-separated list	N/A	A comma-separated list of URLs to the controller REST endpoints, for example <a href="http://localhost:8080/decision-central/rest/controller">http://localhost:8080/decision-central/rest/controller</a> . Setting this property is required when using a controller.
<b>org.kie.server.controller.user</b>	String	<b>kieserve r</b>	The user name to connect to the controller REST API. Setting this property is required when using a controller.
<b>org.kie.server.controller.pwd</b>	String	<b>kieserve r1!</b>	The password to connect to the controller REST API. Setting this property is required when using a controller.
<b>org.kie.server.controller.token</b>	String	N/A	A property that enables you to use token-based authentication between the Process Server and the controller instead of the basic user name/password authentication. The server sends the token as a parameter in the request header. Note that long-lived access tokens are required as the tokens are not refreshed.

Property	Values	Default	Description
<b>org.kie.server.controller.connect</b>	Long	<b>10000</b>	The waiting time in milliseconds between repeated attempts to connect the Process Server to the controller when the server starts.

Table 19.3. Bootstrap Switches for Persistence Properties

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

Table 19.4. Bootstrap Switches for Executor Properties

Property	Values	Default	Description
<b>org.kie.executor.interval</b>	Integer	<b>0</b>	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 <b>org.kie.executor.timeunit</b> property.
<b>org.kie.executor.timeunit</b>	<a href="#">java.util.concurrent.TimeUnit</a> constant	<b>SECONDS</b>	The time unit in which the <b>org.kie.executor.interval</b> property is specified.
<b>org.kie.executor.pool.size</b>	Integer	<b>1</b>	The number of threads used by the Red Hat Process Automation Manager executor.
<b>org.kie.executor.retry.count</b>	Integer	<b>3</b>	The number of retries the Red Hat Process Automation Manager executor attempts on a failed job.

Table 19.5. Callback Bootstrap Switches



Property	Values	Default	Description
<b>org.jbpm.ht.callback</b>	<b>mvel</b> <b>ldap</b> <b>db</b> <b>jaas</b> <b>props</b> <b>custom</b>	<b>jaas</b>	<p>A property that specifies the implementation of user group callback to be used:</p> <ul style="list-style-type: none"> <li>• <b>mvel</b>: Default; mostly used for testing.</li> <li>• <b>ldap</b>: LDAP; requires additional configuration in the <b>jbpm.usergroup.callback.properties</b> file.</li> <li>• <b>db</b>: Database; requires additional configuration in the <b>jbpm.usergroup.callback.properties</b> file.</li> <li>• <b>jaas</b>: JAAS; delegates to the container to fetch information about user data.</li> <li>• <b>props</b>: A simple property file; requires additional file that will keep all information (users and groups).</li> <li>• <b>custom</b>: A custom implementation; specify the fully qualified name of the class in the <b>org.jbpm.ht.custom.callback</b> property.</li> </ul>
<b>org.jbpm.ht.custom.callback</b>	Fully qualified name	N/A	A custom implementation of the <b>UserGroupCallback</b> interface in case the <b>org.jbpm.ht.callback</b> property is set to <b>custom</b> .

Table 19.6. Other Bootstrap Switches

Property	Values	Default	Description
<b>kie.maven.settings.custom</b>	Path	N/A	The location of a custom <b>settings.xml</b> file for Maven configuration.
<b>kie.server.jms.queues.response</b>	String	<b>queue/KIE.SERVER.RESPONSE</b>	The response queue JNDI name for JMS.
<b>org.drools.server.filter.classes</b>	<b>true</b> , <b>false</b>	<b>false</b>	When set to <b>true</b> , the Drools Process Server extension accepts custom classes annotated by the <b>XmlRootElement</b> or <b>Remotable</b> annotations only.

Property	Values	Default	Description
<code>org.kie.server.bypass.auth.user</code>	<b>true</b> , <b>false</b>	<b>false</b>	A property that allows you to bypass the authenticated user for task-related operations, for example queries.
<code>org.kie.server.domain</code>	String	N/A	The JAAS <b>LoginContext</b> domain used to authenticate users when using JMS.
<code>org.kie.server.repo</code>	Path	.	The location where Process Server state files will be stored.
<code>org.kie.server.sync.deploy</code>	<b>true</b> , <b>false</b>	<b>false</b>	<p>A property that instructs the Process Server to hold the deployment until the controller provides the containers deployment configuration. This property only affects servers running in managed mode. The options are as follows:</p> <ul style="list-style-type: none"> <li>• <b>false</b>; the connection to the controller is asynchronous. The application starts, connects to the controller, and once successful, deploys the containers. The application accepts requests even before the containers are available.</li> <li>• <b>true</b>; the deployment of the server application joins the 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 instance. It is recommended that you use only one application (the server) on one server instance.</li> </ul>

## CHAPTER 20. SUPPORTED PROPERTIES

When you install standalone Business Central, you can use the properties listed in this section in the following command:

```
java -jar rhbam-7.0.0-business-central-standalone.jar -s application-  
config.yaml -D<property>=<value> -D<property>=<value>
```

In this command, **<property>** is a property from the following list and **<value>** is a value that you assign to that property:

- **org.uberfire.nio.git.dir**: Location of the Process Server Git directory.
- **org.uberfire.nio.git.dirname**: Name of the Process Server Git directory. Default value: **.niogit**.
- **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.port**: If the Git daemon is enabled, it uses this property as the port number. Default value: **9418**.
- **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.port**: If the SSH daemon is enabled, it uses this property as the port number. Default value: **8001**.
- **org.uberfire.nio.git.ssh.cert.dir**: Location of the **.security** directory where local certificates are stored. Default: the working directory.
- **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: **DSA**.



### 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.

- **org.uberfire.metadata.index.dir**: Place where the Lucene **.index** directory is stored. Default: the 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 part of the pattern because it is substituted by the application role name when matching a principal value to a role

name. Default: Not used.

- **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.example**: Enables external cloning of a demo application from GitHub.
- **org.kie.build.disable-project-explorer**: Disables automatic build of a selected project in Project Explorer. Default value: **false**.
- **org.kie.verification.disable-dtable-realtime-verification**: Disables the real-time validation and verification of decision tables. Default value: **false**.
- **org.kie.server.controller**: URL for connecting with a Kie Server controller, for example: **ws://localhost:8080/business-central/websocket/controller**.
- **org.kie.server.user**: User name used to connect with the Process Server nodes from the controller. This property is only required when using this Business Central installation as a controller.
- **org.kie.server.pwd**: Password used to connect with the Process Server nodes from the controller. This property is only required when using this Business Central installation as a controller.

## CHAPTER 21. RELATED INFORMATION

- *[Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.1](#)*
- *[Planning a Red Hat Process Automation Manager installation](#)*
- *[Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.1](#)*
- *[Deploying a Red Hat Process Automation Manager authoring environment on Red Hat OpenShift Container Platform](#)*

## APPENDIX A. VERSIONING INFORMATION

Documentation last updated on: Monday, October 1, 2018.