Red Hat Process Automation Manager 7.3

Installing and configuring Process Server on Oracle WebLogic Server

Last Updated: 2020-05-04

Red Hat Process Automation Manager 7.3 Installing and configuring Process Server on Oracle WebLogic Server

Red Hat Customer Content Services
brms-docs@redhat.com

Legal Notice

Abstract

This document describes how to configure Oracle WebLogic Server for Process Server and how to install Process Server on that Oracle server instance.

Table of Contents

PREFACE

As a system administrator, you can configure your Oracle WebLogic Server for Red Hat Process Server and install Process Server on that Oracle server instance.

Prerequisites

- An Oracle WebLogic Server instance version 12.2.1.3.0 or later is installed. For complete installation instructions, see the Oracle WebLogic Server product page .
- You have access to the Oracle WebLogic Server Administration Console, usually at **http://<HOST>:7001/console**.

CHAPTER 1. PROCESS SERVER

Process Server is the server where the rules and other artifacts for Red Hat Process Automation Manager are stored and run. Process Server is a standalone built-in component that can be used to instantiate and execute rules through interfaces available for REST, Java Message Service (JMS), or Java client-side applications, as well as to manage processes, jobs, and Red Hat Business Optimizer functionality through solvers.

Created as a web deployable WAR file, Process Server can be deployed on any web container. The current version of the Process Server is included with default extensions for both Red Hat Decision Manager and Red Hat Process Automation Manager.

Process Server has a low footprint with minimal memory consumption and therefore can be deployed easily on a cloud instance. Each instance of this server can open and instantiate multiple containers, which enables you to execute multiple rule services in parallel.

Process Server can be integrated with other application servers, such as Oracle WebLogic Server or IBM WebSphere Application Server, to streamline Red Hat Process Automation Manager application management.

CHAPTER 2. ORACLE WEBLOGIC SERVER

Oracle WebLogic Server is a Java EE application server that provides a standard set of APIs for creating distributed Java applications that can access a wide variety of services, such as databases, messaging services, and connections to external enterprise systems. User clients access these applications using web browser clients or Java clients.

CHAPTER 3. INSTALLING AND RUNNING ORACLE WEBLOGIC SERVER

Oracle WebLogic Server must be installed and running for you to apply many of the configurations that accommodate Process Server. This section describes how to install and start Oracle WebLogic Server in a standalone Oracle WebLogic Server domain.

For the most up-to-date and detailed installation instructions, see the Oracle WebLogic Server product page.

NOTE

If you are already running an instance of Oracle WebLogic Server that uses the same listener port as the one to be used by the server you are starting, you must stop the first server before starting the second server.

Procedure

1. Download Oracle WebLogic Server 12.2.1.3.0 or later from the Oracle WebLogic Server Downloads page.
2. Sign in to the target system and verify that a certified JDK already exists on your system. The installer requires a certified JDK. For system requirements, see Oracle Fusion Middleware Systems Requirements and Specifications. To download the JDK, see the 'About JDK Requirements for an Oracle Fusion Middleware Installation' section in the Planning an Installation of Oracle Fusion Middleware.
3. Go to the directory where you downloaded the installation program.
4. Launch the installation program by running **java -jar** from the JDK directory on your system. See the following examples:
   On UNIX-based operating systems:

   ```
   /home/Oracle/jdk/jdk1.8.0_131/bin/java -jar fmw_12.2.1.3.0_wls_generic.jar
   ```

   On Windows operating systems:

   ```
   C:\Program Files\Java\jdk1.8.0_131\bin\java -jar fmw_12.2.1.3.0_wls_generic.jar
   ```

   Be sure to replace the JDK location in these examples with the actual JDK location on your system.
5. Follow the installation wizard prompts to complete the installation.
6. After the installation is complete, navigate to the domain directory in the command terminal, ***WLS_HOME*/user_projects/<DOMAIN_NAME>**. For example:

   ```
   WLS\user_projects\mydomain
   ```

7. Enter one of the following commands to start Oracle WebLogic Server:
   On UNIX-based operating systems:

   ```
   startWebLogic.sh
   ```

On Windows operating systems:

```
startWebLogic.cmd
```

The startup script displays a series of messages, and finally displays a message similar to the following:

```
<Dec 8, 2017 3:50:42 PM PDT> <Notice> <WebLogicServer> <000360> <Server started in RUNNING mode>
```

8. Open the following URL in a web browser:

```
http://<HOST>:<PORT>/console
```

<HOST> is the system name or IP address of the host server.

<PORT> is the address of the port on which the host server is listening for requests (7001 by default).

For example, to start the Administration Console for a local instance of Oracle WebLogic Server running on your system, enter the following URL in a web browser:

```
http://localhost:7001/console/
```

If you started the Administration Console using secure socket layer (SSL), you must add **s** after **http**, as follows: **https://<HOST>:<PORT>/console**

9. When the login page of the WebLogic Administration Console appears, enter your administrative credentials.

CHAPTER 4. CONFIGURING ORACLE WEBLOGIC SERVER FOR PROCESS SERVER

Before you deploy Process Server with Oracle WebLogic Server, you must configure system properties, security settings, JMS requirements, and other properties on Oracle WebLogic Server. These configurations promote an optimal integration with Process Server.

Prerequisites

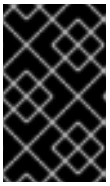
- Oracle WebLogic Server is installed and running.
- You are logged in to the WebLogic Administration Console.

4.1. CONFIGURING THE PROCESS SERVER GROUP AND USERS

You must assign users to a **kie-server** group in the WebLogic Administration Console to enable the container-managed authentication mechanisms in Oracle WebLogic Server.

Procedure

1. In the WebLogic Administration Console, click **Security Realms**.
2. Choose your desired security realm or click **New** to create a new security realm.
3. Navigate to **Users and Groups** → **Groups** → **New** and create the **kie-server** group.
4. Navigate to **Users** → **New** and create a new user.
5. Enter a user, such as **server-user**, and a password for this new user and click **OK**.



IMPORTANT

Make sure that the selected user name does not conflict with any known title of a role or a group. For example, if there is a role called **kie-server**, then do not create a user with the user name **kie-server**.

6. Click the newly created user, then return to the **Groups** tab.
7. Use the selection tool to move the **kie-server** group from the **Available** field to the **Chosen** field, and click **Save**.

4.2. CONFIGURING JDBC DATA SOURCES IN ORACLE WEBLOGIC 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 Oracle WebLogic Server to ensure proper data exchange between the servers and the designated database.

Prerequisite

The JDBC drivers that you want to use to create database connections are installed on all servers on

which you want to deploy the data source. Some JDBC drivers are installed with Oracle WebLogic Server, such as WebLogic-branded Data Direct JDBC drivers for DB2, Informix, MS SQL Server, and Sybase. For more information about JDBC drivers, see Using JDBC Drivers with WebLogic Server in the Oracle Help Center.

Procedure

1. In the WebLogic Administration Console, navigate to **Change Center → Lock & Edit**
2. Under **Domain Structure**, click **Services → Data Sources**.
3. On the **Summary of Data Sources** page, click **New → Generic Data Source**.
4. On the **JDBC Data Sources Properties** page, enter or select the following information:
   - **Name:** Enter a name for this JDBC data source. This name is used in the configuration file (**config.xml**) and throughout the Administration Console whenever referring to this data source.
   - **JNDI Name:** Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection.
   - **Database Type:** Select the DBMS of the database that you want to connect to. If your DBMS is not listed, select **Other**.
5. Click **Next** to continue.
6. Select the **Database Driver** that you want to use to connect to the database. The list includes common JDBC drivers for the selected DBMS and any other JDBC drivers that have been installed previously.
7. On the **Transaction Options** page, leave the **Supports Global Transactions** option selected and choose from the available transaction options. You can also clear this check box to disable (ignore) global transactions in this data source. In most cases, you should leave the option selected for optimal data efficiency.
   - **Two-Phase Commit:** Select this option to enable standard XA processing. This option is only available when you select an XA JDBC driver to make database connections.
   - **Logging Last Resource:** Select this option to enable a non-XA JDBC connection to participate in global transactions using the Logging Last Resource (LLR) transaction optimization. This option is recommended in place of Emulate Two-Phase Commit. This option is only available when you select a non-XA JDBC driver to make database connections.
   - **Emulate Two-Phase Commit:** Select this option to enable a non-XA JDBC connection to emulate participation in distributed transactions using JTA. Select this option only if your application can tolerate heuristic conditions. This option is only available when you select a non-XA JDBC driver to make database connections.
   - **One-Phase Commit:** Select this option to enable the non-XA connection to participate in a global transaction as the only transaction participant. This option is only available when you select a non-XA JDBC driver to make database connections.
8. Click **Next** to continue.

9. On the **Connection Properties** page, enter values for the following properties:
    - **Service Name:** Specify the service name of the database to which you want to connect. This must be the same for each data source if more than one is provided. This field is available only if you selected one of the available service-instance connections drivers for Oracle Real Application Clusters (RAC).
    - **Database Name:** Enter the name of the database that you want to connect to. Exact database name requirements vary by JDBC driver and by DBMS.
    - **Host Name:** Enter the DNS name or IP address of the server that hosts the database. If you are creating an Oracle GridLink service-instance connection, this must be the same for each data source if more than one is provided.
    - **Port:** Enter the port on which the database server listens for connection requests.
    - **Database User Name:** Enter the database user account name that you want to use for each connection in the data source.
    - **Password/Confirm Password:** Enter the password for the database user account.
    - **oracle.jdbc.DRCPConnectionClass:** Optionally, enter the Database Resident Connection Pooling (DCRP) connection class if required by your environment.
10. Click **Next** to continue.
11. On the **Test Database Connection** page, review the connection parameters and click **Test Configuration**.
    Oracle WebLogic Server attempts to create a connection from the Administration Server to the database. Results from the connection test are displayed at the top of the page. If the test is unsuccessful, correct any configuration errors and retry the test.
12. Click **Next** to continue or to skip this step if the JDBC driver you selected is not installed on the Administration Server.
13. On the **Select Targets** page, select the servers or clusters on which you want to deploy the data source and click **Finish**.
14. Return to the main menu of the WebLogic Administration Console and select **Change Center → Activate Changes**.

For more information about Oracle WebLogic Server data sources, see JDBC Data Sources for Oracle WebLogic Server in the Oracle Help Center.

4.3. CONFIGURING JAVA MESSAGE SERVICE (JMS)

The Java Message Service (JMS) is a Java API that Process Server uses to exchange messages with other application servers such as Oracle WebLogic Server and IBM WebSphere Application Server. You must configure your application server to send and receive JMS messages through Process Server to ensure proper collaboration between the two servers.

4.3.1. Create a JMS server

You must create a JMS server in order to use JMS.

Procedure

1. In the WebLogic Administration Console, navigate to **Services → Messaging → JMS Servers**.
2. Click **New** to create a new JMS server.
3. Enter a name for your JMS server and click **Next**.
4. Select the target server chosen for the Process Server deployment.
5. Click **Finish**.

4.3.2. Create a JMS module

You must create a JMS module to store your JMS resources, such as connection factories and queues.

Prerequisite

You have created a JMS server.

Procedure

1. In the WebLogic Administration Console, navigate to **Services → Messaging → JMS Modules**.
2. Click **New** to create a module.
3. Enter a module name and click **Next**.
4. Select the target server chosen for the Process Server deployment and click **Finish**.
5. Click the newly created module name and then click **Subdeployments**.
6. Click **New** to create a subdeployment for your module.
7. Give your subdeployment a name and click **Next**.
8. Select the check box to choose the previously created JMS server.
9. Click **Finish** to complete the subdeployment configuration.

4.3.3. Create JMS connection factories

To enable messaging with Process Server, you must create certain JMS connection factories for sending and receiving messages.

Prerequisites

- You have created a JMS server.
- You have created a JMS module.

Procedure

1. In the WebLogic Administration Console, navigate to **Services → Messaging → JMS Modules** to see a list of JMS modules.
2. Select your previously created module and click **New** to create a new JMS resource.

3. Select **Connection Factory** and click **Next**.
4. For each of the following required connection factories, enter the name of the connection factory (for example, **KIE.SERVER.REQUEST**) and the JNDI name (for example, **jms/cf/KIE.SERVER.REQUEST**) and click **Next**. The connection factory automatically selects the servers assigned to the JMS Module as the default.
5. Click **Finish** to add the connection factory, and repeat for each required factory.

4.3.3.1. JMS connection factories for Process Server

The following are the required Java Message Service (JMS) connection factories that enable JMS messaging with Process Server:

Table 4.1. Required JMS connection factories for Process Server

Name	Default value	Used for
KIE.SERVER.REQUEST	**jms/cf/KIE.SERVER.REQUEST**	Sending all requests to Process Server
KIE.SERVER.RESPONSE	**jms/cf/KIE.SERVER.RESPONSE**	Receiving all responses produced by Process Server
KIE.SERVER.EXECUTOR	**jms/cf/KIE.SERVER.EXECUTOR**	Process Server executor services

4.3.4. Create JMS queues

JMS queues are the destination end points for point-to-point messaging. You must create certain JMS queues to enable JMS messaging with Process Server.

Prerequisites

- You have created a JMS server.
- You have created a JMS module.

Procedure

1. In the WebLogic Administration Console, navigate to **Services → Messaging → JMS Modules** to see the list of JMS modules.
2. Select your previously created module, then click **New** to create a new JMS resource.
3. Select **Queue** and click **Next**.
4. For each of the following required queues, enter the name of the queue (for example, **KIE.SERVER.REQUEST**) and the JNDI name (for example, **jms/KIE.SERVER.REQUEST**) and then click **Next**.
5. Choose the JMS module subdeployment that connects to the JMS server.
6. Click **Finish** to add the queue, and repeat for each required queue.

4.3.4.1. JMS queues for Process Server

The following are the required Java Message Service (JMS) queues that enable JMS messaging with Process Server:

Table 4.2. Required JMS queues for Process Server

Name	Default value	Used for
KIE.SERVER.REQUEST	**jms/KIE.SERVER.REQUEST**	Sending all requests to Process Server
KIE.SERVER.RESPONSE	**jms/KIE.SERVER.RESPONSE**	Receiving all responses produced by Process Server
KIE.SERVER.EXECUTOR	**jms/KIE.SERVER.EXECUTOR**	Process Server executor services

4.4. SETTING SYSTEM PROPERTIES IN ORACLE WEBLOGIC SERVER

Set the system properties listed in this section on your Oracle WebLogic Server before you deploy Process Server.

Procedure

1. Set the following system property to increase the Java Virtual Machine (JVM) memory size:

```
USER_MEM_ARGS=-Xms512m -Xmx1024m
```

   If you do not increase the JVM memory size, Oracle WebLogic Server freezes or causes deployment errors when deploying Process Server.

2. Specify the following system properties for Process Server on the Oracle WebLogic Server instance:

   Table 4.3. System properties for Process Server

Name	Value	Description
kie.server.jms.queues.response	**jms/queue/KIE.SERVER.RESPONSE**	The JNDI name of JMS queue for responses used by the Process Server.
org.kie.server.domain	**OracleDefaultLoginConfiguration**	JAAS **LoginContext** domain used to authenticate users when using JMS.
org.kie.server.persistence.ds	**jdbc/jbpm**	Data source JNDI name for Process Server.

Name	Value	Description
org.kie.server.persistence.tm	**org.hibernate.service.jta.platform.internal.WeblogicJtaPlatform**	Transaction manager platform for setting Hibernate properties.
org.kie.server.persistence.dialect	Example: **org.hibernate.dialect.H2Dialect**	Specifies the Hibernate dialect to be used. Set according to data source.
org.kie.executor.jms.queue	**jms/queue/KIE.SERVER.EXECUTOR**	Job executor JMS queue for Process Server.
org.kie.executor.jms.cf	**jms/cf/KIE.SERVER.EXECUTOR**	Job executor JMS connection factory for Process Server.
org.kie.server.router	Example: **http://localhost:9000**	(Optional) Specifies one or more URLs for one or more Process Server routers (Smart Routers) that the application server is part of in a clustered Process Server environment.

3. Set the same property values in the **JAVA_OPTIONS** environment variable:

```
JAVA_OPTIONS="-Dkie.server.jms.queues.response=jms/queue/KIE.SERVER.RESPONSE
 -Dorg.kie.server.domain=OracleDefaultLoginConfiguration
 -Dorg.kie.executor.jms.cf=jms/cf/KIE.SERVER.EXECUTOR
 -Dorg.kie.executor.jms.queue=jms/queue/KIE.SERVER.EXECUTOR
 -Dorg.kie.server.persistence.ds=jdbc/jbpm
 -Dorg.kie.server.persistence.tm=org.hibernate.service.jta.platform.internal.WeblogicJtaPlatform
 -Dorg.kie.server.persistence.dialect=org.hibernate.dialect.H2Dialect
// Optional server router, for clustered server environment
 -Dorg.kie.server.router=http://localhost:9000
```

4.5. STOPPING AND RESTARTING ORACLE WEBLOGIC SERVER

After you have configured all required system properties in Oracle WebLogic Server, stop and restart the Oracle server to ensure that the configurations are applied.

Procedure

1. In the WebLogic Administration Console, navigate to **Change Center → Lock & Edit**
2. Under **Domain Structure**, click **Environment → Servers → Control**.
3. Select the server that you want to stop and click **Shutdown**.

4. Select **When Work Completes** to gracefully shut down the server or select **Force Shutdown Now** to stop the server immediately without completing ongoing tasks.

5. On the **Server Life Cycle Assistant** pane, click **Yes** to complete the shutdown.

6. After the shutdown is complete, navigate to the domain directory in the command terminal, ***WLS_HOME*/user_projects/<DOMAIN_NAME>**. For example:

```
WLS\user_projects\mydomain
```

7. Enter one of the following commands to restart Oracle WebLogic Server to apply the new configurations:
   On UNIX-based operating systems:

```
startWebLogic.sh
```

   On Windows operating systems:

```
startWebLogic.cmd
```

8. Open the Administration Console in a web browser (for example, **http://localhost:7001/console/**) and log in with your credentials.

CHAPTER 5. INSTALLING PROCESS SERVER WITH ORACLE WEBLOGIC SERVER

After you have configured all required system properties in Oracle WebLogic Server, you can install Process Server with Oracle WebLogic Server to streamline Red Hat Process Automation Manager application management.

Prerequisite

An Oracle WebLogic Server instance is configured as described in Chapter 4, *Configuring Oracle WebLogic Server for 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:
   - **Product:** Process Automation Manager
   - **Version:** 7.3
2. Download **Red Hat Process Automation Manager 7.3.0 Process Server for All Supported EE7 Containers**.
3. Extract the downloaded **rhpam-7.3.0-kie-server-ee7.zip** file to a temporary directory.
4. In the WebLogic Administration Console, navigate to **Deployments** to view all existing applications.
5. Click **Install**.
6. Navigate to the temporary directory where you downloaded and extracted the **rhpam-7.3.0-kie-server-ee7.zip** file, and go to **rhpam-7.3.0-kie-server-ee7/kie-server.war**.
7. Select the **kie-server.war** file and click **Next** to continue.
8. Select **Install this deployment as an application**as the targeting style and click **Next**.
9. Set the application name to **kie-server** and set the security model to **DD Only**. Leave the remaining options as default and click **Next** to continue.
10. In the **Additional Configuration** section, choose **No, I will review the configuration later**and click **Finish**.

5.1. VERIFYING THE PROCESS SERVER INSTALLATION ON ORACLE WEBLOGIC SERVER

After you have installed Process Server on Oracle WebLogic Server, verify that the installation was successful.

Prerequisites

- An Oracle WebLogic Server instance is configured as described in Chapter 4, *Configuring Oracle WebLogic Server for Process Server*.

- Process Server is installed as described in Chapter 5, *Installing Process Server with Oracle WebLogic Server*.

Procedure

1. Enter the Process Server URL **http://<HOST>:<PORT>/kie-server/services/rest/server** in a web browser.

2. Verify that Process Server is running.
   If Process Server is not running, stop and restart the Oracle WebLogic Server instance and try again to access the Process Server URL.

CHAPTER 6. INSTALLING AND RUNNING THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER WITH ORACLE WEBLOGIC SERVER

To use the Process Server REST API or Java Client API to interact with Process Server, install the headless Process Automation Manager controller with Oracle WebLogic Server. The headless Process Automation Manager controller manages Process Server configuration in a centralized way so that you can use the headless Process Automation Manager controller to create and maintain containers and perform other server-level tasks.

Prerequisites

- The Oracle WebLogic Server instance is configured as described in Chapter 4, *Configuring Oracle WebLogic Server for Process Server*.
- Process Server is installed on the Oracle WebLogic Server instance.
- You have sufficient user permissions to complete the installation.

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:** 7.3
2. Download **Red Hat Process Automation Manager 7.3.0 Add-Ons**
3. Extract the downloaded **rhpam-7.3.0-add-ons.zip** file to a temporary directory.
4. In the WebLogic Administration Console, navigate to **Security Realms → Users and Groups**.
5. In the **kie-server** group that you created previously, create a user for the headless Process Automation Manager controller, such as **controller**, and a password for this new user and click **OK**. For more information about creating groups and users, see Section 4.1, "Configuring the Process Server group and users".
6. Navigate to **Deployments** to view all existing applications.
7. Click **Install**.
8. Navigate to the temporary directory where you downloaded and extracted the **rhpam-7.3.0-add-ons.zip** file, and go to **rhpam-7.3.0-add-ons/rhpam-7.3-controller-ee7.zip/controller.war**.
9. Select the **controller.war** file and click **Next** to continue.
10. Select **Install this deployment as an application** as the targeting style and click **Next**.
11. Keep the application name as **controller** and set the security model to **DD Only**. Leave the remaining options as default and click **Next** to continue.
12. In the **Additional Configuration** section, choose **No, I will review the configuration later** and click **Finish**.

6.1. SETTING SYSTEM PROPERTIES FOR THE HEADLESS PROCESS AUTOMATION MANAGER CONTROLLER

After you install the headless Process Automation Manager controller, set the system properties listed in this section on your application server or servers to enable proper interaction with the headless Process Automation Manager controller.

NOTE

For optimal results, install Process Server and the headless Process Automation Manager controller on different servers in production environments. In development environments, you can install Process Server and the headless Process Automation Manager controller on the same server. In either case, be sure to make these property changes on all application servers where the headless Process Automation Manager controller is installed.

Prerequisite

Process Server and the headless Process Automation Manager controller are installed on the application server instance.

Procedure

1. Specify the following JVM property values on the application server instance where the headless Process Automation Manager controller is installed:

Table 6.1. Required properties for the headless Process Automation Manager controller

Name	Requirement
org.kie.server.user	A user with the **kie-server** role
org.kie.server.pwd	The password for the user specified in the **org.kie.server.user** property

2. Specify the following JVM property values on the application server instance where Process Server is installed:

Table 6.2. Required properties for Process Server when headless Process Automation Manager controller is installed

Name	Requirement
org.kie.server.controller.user	A user with the **kie-server** role
org.kie.server.controller.pwd	The password for the user specified for the **org.kie.server.controller.user** property
org.kie.server.id	The ID or name of the Process Server installation, such as **rhdm700-decision-server-1**

Name	Requirement
org.kie.server.location	The URL of the Process Server, **http://<HOST>:<PORT>/kie-server/services/rest/server**
org.kie.server.controller	The URL of the headless Process Automation Manager controller, **http://<HOST>:<PORT>/controller/rest/controller**

<HOST> is the ID or name of the Process Server host, for example, **localhost** or **192.7.8.9**.

<PORT> is the port of the Process Server host, for example, **7001**.

6.2. VERIFYING THE INSTALLATION

After you install the headless Process Automation Manager controller and define the required system properties and role requirements on the application server, verify that the headless Process Automation Manager controller works correctly.

Prerequisites

- Process Server and the headless Process Automation Manager controller are installed on the application server instance.
- You have set all required system properties and role requirements for the headless Process Automation Manager controller on the application server.

Procedure

In your command terminal, enter the following command to verify that the headless Process Automation Manager controller is working:

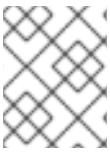
```
curl -X GET "http://<HOST>:<PORT>/controller/rest/controller/management/servers" -H  "accept: application/xml" -u '<CONTROLLER>:<CONTROLLER_PWD>'
```

<HOST> is the ID or name of the Process Server host, for example, **localhost** or **192.7.8.9**.

<PORT> is the port of the Process Server host, for example, **7001**.

<CONTROLLER> and **<CONTROLLER_PWD>** are the user credentials that you created in this section.

The command should return information about the Process Server instance.



NOTE

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

If the headless Process Automation Manager controller is not running, stop and restart the application server instance and try again to access the headless Process Automation Manager controller URL or API.

CHAPTER 7. CONFIGURING AN EMBEDDED PROCESS ENGINE AND DECISION ENGINE IN ORACLE WEBLOGIC SERVER

An embedded engine is a light-weight work flow and rule engine that enables you to execute your decisions and business processes. An embedded engine can be part of a Red Hat Process Automation Manager application or it can be deployed as a service through OpenShift, Kubernetes, and Docker. You can embed an engine in a Red Hat Process Automation Manager application through the API or as a set of contexts and dependency injection (CDI) services.

If you intend to use an embedded engine with your Red Hat Process Automation Manager application, you must add Maven dependencies to your project by adding the Red Hat Business Automation bill of materials (BOM) files to the project's **pom.xml** file. The Red Hat Business Automation BOM applies to both Red Hat Decision Manager and Red Hat Process Automation Manager. For more information about the Red Hat Business Automation BOM, see What is the mapping between Red Hat Process Automation Manager and the Maven library version?.

Procedure

1. Declare the Red Hat Business Automation BOM in the **pom.xml** file:

```
<dependencyManagement>
 <dependencies>
  <dependency>
   <groupId>com.redhat.ba</groupId>
   <artifactId>ba-platform-bom</artifactId>
   <version>7.3.0.GA-redhat-00002</version>
   <type>pom</type>
   <scope>import</scope>
  </dependency>
 </dependencies>
</dependencyManagement>
<dependencies>
<!-- Your dependencies -->
</dependencies>
```

2. Declare dependencies required for your project in the **<dependencies>** tag. After you import the product BOM into your project, the versions of the user-facing product dependencies are defined so you do not need to specify the **<version>** sub-element of these **<dependency>** elements. However, you must use the **<dependency>** element to declare dependencies which you want to use in your project.

   - For a basic Red Hat Process Automation Manager project, declare the following dependencies, depending on the features that you want to use:

     Embedded process engine dependencies

```
<!-- Public KIE API -->
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-api</artifactId>
</dependency>

<!-- Core dependencies for process engine -->
```

```
<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-flow</artifactId>
</dependency>

<dependency>
  <artifactId>jbpm-flow-builder</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-bpmn2</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-runtime-manager</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-persistence-jpa</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-query-jpa</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-audit</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-kie-services</artifactId>
</dependency>

<!-- Dependency needed for default WorkItemHandler implementations. -->
<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-workitems-core</artifactId>
</dependency>

<!-- Logging dependency. You can use any logging framework compatible with slf4j. -->
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
</dependency>
```

- For a Red Hat Process Automation Manager project that uses CDI, you typically declare the following dependencies:

  CDI-enabled process engine dependencies

```
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-api</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-kie-services</artifactId>
</dependency>

<dependency>
  <groupId>org.jbpm</groupId>
  <artifactId>jbpm-services-cdi</artifactId>
</dependency>
```

Embedded decision engine dependencies

```
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-compiler</artifactId>
</dependency>

<!-- Dependency for persistence support. -->
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-persistence-jpa</artifactId>
</dependency>

<!-- Dependencies for decision tables, templates, and scorecards.
For other assets, declare org.drools:business-central-models-* dependencies. -->
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-decisiontables</artifactId>
</dependency>
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-templates</artifactId>
</dependency>
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-scorecards</artifactId>
</dependency>

<!-- Dependency for loading KJARs from a Maven repository using KieScanner. -->
<dependency>
  <groupId>org.kie</groupId>
  <artifactId>kie-ci</artifactId>
</dependency>
```

- To use the Process Server, declare the following dependencies:

Client application Process Server dependencies

```
<dependency>
  <groupId>org.kie.server</groupId>
```

```
  <artifactId>kie-server-client</artifactId>
</dependency>
```

- To create a remote client for Red Hat Process Automation Manager, declare the following dependency:

  Client dependency

```
<dependency>
  <groupId>org.uberfire</groupId>
  <artifactId>uberfire-rest-client</artifactId>
</dependency>
```

- When creating a JAR file that includes assets, such as rules and process definitions, specify the packaging type for your Maven project as **kjar** and use **org.kie:kie-maven-plugin** to process the **kjar** packaging type located under the **<project>** element. In the following example, **${kie.version}** is the Maven library version listed in What is the mapping between Red Hat Process Automation Manager and the Maven library version?:

```
<packaging>kjar</packaging>
<build>
 <plugins>
  <plugin>
   <groupId>org.kie</groupId>
   <artifactId>kie-maven-plugin</artifactId>
   <version>${kie.version}</version>
   <extensions>true</extensions>
  </plugin>
 </plugins>
</build>
```

3. If you use a process engine or decision engine with persistence support in your project, you must declare the following hibernate dependencies in the **dependencyManagement** secion of your **pom.xml** file by copying the **version.org.hibernate-4ee7** property from the Red Hat Business Automation BOM file:

   Hibernate dependencies

```
<!-- hibernate dependencies -->
<dependencyManagement>
 <dependencies>
  <dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>${version.org.hibernate-4ee7}</version>
  </dependency>

  <dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>${version.org.hibernate-4ee7}</version>
  </dependency>
 </dependencies>
</dependencyManagement>
```

CHAPTER 8. NEXT STEPS

- *Getting started with decision services*
- *Designing a decision service using guided decision tables*

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Wednesday, May 8, 2019.