



Red Hat Decision Manager 7.0

Installing and configuring Red Hat Business Optimizer

Red Hat Decision Manager 7.0 Installing and configuring Red Hat Business Optimizer

Red Hat Customer Content Services
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 describes how to install and configure Red Hat Business Optimizer in Red Hat Decision Manager 7.0.

Table of Contents

PREFACE	4
PART I. SOLVE PLANNING PROBLEMS USING BUSINESS OPTIMIZER	5
CHAPTER 1. WHAT IS RED HAT BUSINESS OPTIMIZER	6
1.1. A PLANNING PROBLEM IS NP-COMPLETE OR NP-HARD	6
CHAPTER 2. WHAT IS A PLANNING PROBLEM?	7
2.1. A PLANNING PROBLEM HAS A HUGE SEARCH SPACE	7
2.2. A PLANNING PROBLEM HAS CONSTRAINTS	8
PART II. DOWNLOAD AND INSTALL BUSINESS OPTIMIZER	9
CHAPTER 3. DOWNLOAD RED HAT BUSINESS OPTIMIZER	10
CHAPTER 4. INSTALLING THE RED HAT BUSINESS OPTIMIZER ENGINE IN YOUR APPLICATION	11
PART III. RUN THE EXAMPLES	12
CHAPTER 5. RUNNING THE BUSINESS OPTIMIZER EXAMPLES	13
5.1. RUNNING THE RED HAT BUSINESS OPTIMIZER EXAMPLES IN AN IDE (INTELLIJ, ECLIPSE, OR NETBEANS)	14
PART IV. CONFIGURE THE DECISION SERVER	16
CHAPTER 6. DECISION SERVER	17
CHAPTER 7. DEPLOYING DECISION SERVER	18
7.1. BOOTSTRAP SWITCHES	18
CHAPTER 8. MANAGED DECISION SERVER	23
8.1. CONFIGURING DECISION SERVER MANAGED BY DECISION CENTRAL	23
8.2. UNMANAGED DECISION SERVER	25
CHAPTER 9. CREATING CONTAINERS	27
CHAPTER 10. MANAGING CONTAINERS IN DECISION CENTRAL	29
10.1. STARTING, STOPPING, AND DELETING CONTAINERS IN DECISION CENTRAL	29
10.2. UPGRADING CONTAINERS IN DECISION CENTRAL	29
10.3. MANAGING MULTIPLE CONTAINERS	30
PART V. CONFIGURE THE RED HAT BUSINESS OPTIMIZER REST API	31
CHAPTER 11. KNOWLEDGE STORE REST API	32
11.1. JOB CALLS	32
11.2. ORGANIZATIONAL UNIT CALLS	33
11.3. REPOSITORY CALLS	35
PART VI. THE REST API FOR DECISION SERVER EXECUTION	37
CHAPTER 12. THE REST API FOR DECISION SERVER	38
12.1. RED HAT DECISION MANAGER COMMANDS	39
CHAPTER 13. CONFIGURE THE RED HAT BUSINESS OPTIMIZER REST API	41
13.1. [GET] /CONTAINERS	41
13.2. [PUT] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	42
13.3. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS	42

13.4. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	43
13.5. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/SOLVING	43
13.6. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/TERMINATING-EARLY	45
13.7. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/BESTSOLUTION	45
13.8. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMACTCHANGES	46
13.9. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMACTCHANGES/PROCESSED	47
13.10. [DELETE] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	47
PART VII. AUTHOR PLANNING ASSETS AND CREATE SOLVERS	48
CHAPTER 14. CREATE A SOLVER IN DECISION CENTRAL	49
14.1. CONFIGURE SOLVER TERMINATION	49
CHAPTER 15. AUTHOR PLANNING ASSETS WITH THE RED HAT BUSINESS OPTIMIZER DOMAIN EDITOR	51
CHAPTER 16. CONFIGURE A PLANNING ENTITY DIFFICULTY COMPARATOR	53
CHAPTER 17. DEFINE SCORE CONSTRAINTS USING THE GUIDED RULES DESIGNER	55
APPENDIX A. VERSIONING INFORMATION	56

PREFACE

As a business rules developer, you can use the Red Hat Business Optimizer to find the optimal solution to planning problems based on a set of limited resources and under specific constraints.

Use this document to set up and configure Red Hat Business Optimizer functionality.

Prerequisite

The **Employee Rostering** sample project is created in Decision Central to demonstrate Business Optimizer capabilities. To view this project, you must have Red Hat Decision Manager installed. For more information about setting up and installing Red Hat Decision Manager and Decision Central, see [Getting started with decision services](#).

PART I. SOLVE PLANNING PROBLEMS USING BUSINESS OPTIMIZER

CHAPTER 1. WHAT IS RED HAT BUSINESS OPTIMIZER

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

For example, Red Hat Business Optimizer helps solve various use cases:

- *Employee/Patient Rosters*: It helps create timetables for nurses and keeps track of patient bed management.
- *Educational Timetables*: It helps schedule lessons, courses, exams, and conference presentations.
- *Shop Schedules*: It tracks car assembly lines, machine queue planning, and workforce task planning.
- *Cutting Stock*: It minimizes waste by reducing the consumption of resources such as paper and steel.

Every organization faces planning problems; that is, they provide products and services with a limited set of constrained resources (employees, assets, time, and money).

Red Hat Business Optimizer is open source software under the Apache Software License 2.0. It is 100% pure Java and runs on most Java virtual machines.

1.1. A PLANNING PROBLEM IS NP-COMPLETE OR NP-HARD

The provided use cases are *probably* [NP-complete or NP-hard](#), which means the following statements apply:

- It is easy to verify a given solution to a problem in reasonable time.
- There is no simple way to find the optimal solution of a problem in reasonable time.

The implication is that solving your problem is probably harder than you anticipated, because the two common techniques will not suffice:

- A brute force algorithm (even a more advanced variant) will take too long.
- A quick algorithm, for example in the [bin packing problem](#), *putting in the largest items first* will return a solution that is far from optimal.

By using advanced optimization algorithms, the Business Optimizer does find a good solution in reasonable time for such planning problems.

CHAPTER 2. WHAT IS A PLANNING PROBLEM?

A planning problem has an optimal goal, based on limited resources and under specific constraints. Optimal goals can be any number of things, such as:

- Maximized profits - the optimal goal results in the highest possible profit.
- Minimized ecological footprint - the optimal goal has the least amount of environmental impact.
- Maximized satisfaction for employees or customers - the optimal goal prioritizes the needs of employees or customers.

The ability to achieve these goals relies on the number of resources available. For example, the following resources might be limited:

- The number of people
- Amount of time
- Budget
- Physical assets, for example, machinery, vehicles, computers, buildings, and so on

You must also take into account the specific constraints related to these resources, such as the number of hours a person works, their ability to use certain machines, or compatibility between pieces of equipment.

The Business Optimizer helps Java programmers solve constraint satisfaction problems efficiently. It combines optimization heuristics and metaheuristics with efficient score calculation.

2.1. A PLANNING PROBLEM HAS A HUGE SEARCH SPACE

A planning problem has a number of solutions.

There are several categories of solutions:

Possible solution

A possible solution is any solution, whether or not it breaks any number of constraints. Planning problems often have an incredibly large number of possible solutions. Many of those solutions are worthless.

Feasible solution

A feasible solution is a solution that does not break any (negative) hard constraints. The number of feasible solutions tends to be relative to the number of possible solutions. Sometimes there are no feasible solutions. Every feasible solution is a possible solution.

Optimal solution

An optimal solution is a solution with the highest score. Planning problems usually have a few optimal solutions. They always have at least one optimal solution, even in the case that there are no feasible solutions and the optimal solution is not feasible.

Best solution found

The best solution found is the solution with the highest score found by an implementation in a given amount of time. The best solution found is likely to be feasible and, given enough time, it's an optimal solution.

Counterintuitively, the number of possible solutions is huge (if calculated correctly), even with a small data set.

In the examples provided in the **planner-engine** distribution folder, most instances have a vast number of possible solutions. As there is no guaranteed way to find the optimal solution, any implementation is forced to evaluate at least a subset of all those possible solutions.

The Business Optimizer supports several optimization algorithms to efficiently wade through that incredibly large number of possible solutions.

Depending on the use case, some optimization algorithms perform better than others, but it is impossible to tell in advance. Using the Business Optimizer, you can switch the optimization algorithm by changing the solver configuration in a few lines of XML or code.

2.2. A PLANNING PROBLEM HAS CONSTRAINTS

Usually, a planning problem has at least two levels of constraints:

- A *(negative) hard constraint* must not be broken.
For example, one teacher can not teach two different lessons at the same time.
- A *(negative) soft constraint* should not be broken if it can be avoided.
For example, Teacher A does not like to teach on Friday afternoons.

Some problems also have positive constraints:

- A *positive soft constraint (or reward)* should be fulfilled if possible.
For example, Teacher B likes to teach on Monday mornings.

Some basic problems only have hard constraints. Some problems have three or more levels of constraints, for example, hard, medium, and soft constraints.

These constraints define the *score calculation* (otherwise known as the *fitness function*) of a planning problem. Each solution of a planning problem can be graded with a score. With the Business Optimizer, score constraints are written in an object oriented language, such as Java code or Drools rules.

This type of code is flexible and scalable.

PART II. DOWNLOAD AND INSTALL BUSINESS OPTIMIZER

CHAPTER 3. DOWNLOAD RED HAT BUSINESS OPTIMIZER

Red Hat Business Optimizer can be used to optimize business solutions in Decision Central, in your preferred IDE, or on OpenShift.

Procedure

1. Download and install Red Hat Decision Manager from the [Red Hat Customer Portal](#). See [Installing Red Hat Decision Manager on premise](#) for more information about installing Red Hat Decision Manager.
2. Download the **rhdm-7.0.0.GA-add-ons.zip** file from the Red Hat Decision Manager section of the [Red Hat Customer Portal](#).
3. Extract the Red Hat Business Optimizer engine from the **rhdm-7.0.0.GA-planner-engine.zip** file to access the complete set of Red Hat Business Optimizer engine JAR files.
4. [Run the examples](#).

See [Migrating from Red Hat BRMS 6.4 to Red Hat Decision Manager 7.0](#) for information about migrating from previous versions of Red Hat Business Optimizer.

CHAPTER 4. INSTALLING THE RED HAT BUSINESS OPTIMIZER ENGINE IN YOUR APPLICATION

Red Hat Business Optimizer can be installed using Maven or other applications, such as Gradle, Ivy, or Buildr.

Prerequisite

The Red Hat Decision Manager has been downloaded and installed from the [Red Hat Customer Portal](#). See [Getting started with decision services](#) for more information about installing Red Hat Decision Manager.

Procedure

1. Get the Red Hat Business Optimizer **optaplanner-core** JARs at the [Red Hat JBoss Maven Repository](#).
2. Identify the latest version by checking the [Red Hat JBoss Maven Repository](#).
3. Add a dependency to **optaplanner-core** in your project's **pom.xml**:

```
<dependency>
  <groupId>org.optaplanner</groupId>
  <artifactId>optaplanner-core</artifactId>
  <version>{MAVEN_ARTIFACT_VERSION}</version>
</dependency>
```

4. (Optional) Add any other Red Hat Business Optimizer engine modules that you require, such as the **optaplanner-persistence-jpa** or **optaplanner-persistence-xstream** integration modules. For more information about how to configure Red Hat Business Optimizer for use with other Java technologies, see the [Optaplanner](#) documentation.



NOTE

The **optaplanner-benchmark** module is included as part of the engine, however, it is recommended to be used as a separate module, as demonstrated in the this [Employee Rostering module](#). This is to avoid leaking the **optaplanner-benchmark** into the **.war** file.

Alternatively, if you are using Ant (without Ivy) you can install the engine by copying all of the JARs from the downloaded ZIP's **binaries** directory. Manually verify that your classpath does not contain duplicate JARs.



NOTE

The downloaded **.zip** file **binaries** directory contains far more JARs than **optaplanner-core** actually uses. It also contains the JARs used by other modules, such as **optaplanner-benchmark**.

Check the Maven repository **pom.xml** files to determine the minimal dependency set for a specific version of a specific module.

PART III. RUN THE EXAMPLES

CHAPTER 5. RUNNING THE BUSINESS OPTIMIZER EXAMPLES

Red Hat Business Optimizer includes a number of examples to demonstrate a variety of use cases.

Prerequisite

Download and install Red Hat Decision Manager from the [Red Hat Customer Portal](#).

Procedure

1. Download the **rhdm-7.0.0.GA-add-ons.zip** file from the Red Hat Decision Manager section of the [Red Hat Customer Portal](#).
2. Extract the Red Hat Business Optimizer engine from the **rhdm-7.0.0.GA-planner-engine.zip** file.
3. In the **rhdm-7.0.0.GA-planner-engine** folder, open the **examples** directory and use the appropriate script to run the examples:

Linux or Mac:

```
$ cd examples  
$ ./runExamples.sh
```

Windows:

```
$ cd examples  
$ runExamples.bat
```

Select and run an example from the GUI application window:



NOTE

Red Hat Business Optimizer itself has no GUI dependencies. It runs just as well on a server or a mobile JVM as it does on the desktop.

5.1. RUNNING THE RED HAT BUSINESS OPTIMIZER EXAMPLES IN AN IDE (INTELLIJ, ECLIPSE, OR NETBEANS)

Use the following procedure to run the Red Hat Business Optimizer examples in IntelliJ or Netbeans:

Procedure

1. Open the file `examples/sources/pom.xml` as a new project.
The Maven integration will take care of the rest of the installation.

Use the following procedure to run the Red Hat Business Optimizer examples in Eclipse:

Procedure

1. Open a new project for the directory `examples/sources`.
2. Add all the JARs to the classpath from the directory `binaries` and the directory `examples/binaries`, except for the file `examples/binaries/optaplanner-examples-*.jar`.
3. Add the Java source directory `src/main/java` and the Java resources directory `src/main/resources`.

4. Create a run configuration:

- Main class: **org.optaplanner.examples.app.OptaPlannerExamplesApp**
- VM parameters (optional): **-Xmx512M -server**
- Working directory: **examples/sources**

5. Run the run configuration.

PART IV. CONFIGURE THE DECISION SERVER

CHAPTER 6. DECISION SERVER

The Decision Server is a standalone, built-in component that can be used to instantiate and execute rules through interfaces available for REST, JMS, or a Java client side application, and Red Hat Business Optimizer functionality through solvers. Created as a web deployable WAR file, this server can be deployed on any web container. The current version of the Decision Server is included with default extensions for both Red Hat Decision Manager and Red Hat Business Automation.

This 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 KIE containers, which allows you to execute multiple rule services in parallel.

You can provision the Decision Server instances through Decision Central.

CHAPTER 7. DEPLOYING DECISION SERVER

The Decision Server is distributed as a web application archive (WAR) file `kie-server.war`. When you install Red Hat Decision Manager, deploy the `kie-server.war` file for full functionality. For information about how to deploy the Decision Server, see [Installing Red Hat Decision Manager on premise](#).

Prerequisite

The Red Hat Decision Manager has been downloaded and deployed with the `kie-server.war` file.

Procedure

1. Create a user with the role `kie-server` in the web container.
2. Verify that you can access the decision engine.
 - a. In a web browser, open `http://SERVER:PORT/kie-server/services/rest/server/`
 - b. Enter the user name and the password specified in the previous step.
3. Once authenticated, an XML response in the form of engine status opens:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response type="SUCCESS" msg="Kie Server info">
  <kie-server-info>
    <capabilities>KieServer</capabilities>
    <capabilities>BRM</capabilities>
    <capabilities>BPM</capabilities>
    <location>http://localhost:8230/kie-
server/services/rest/server</location>
    <name>first-kie-server</name>
    <id>first-kie-server</id>
    <version>7.5.1.Final-redhat-1</version>
  </kie-server-info>
</response>
```

7.1. BOOTSTRAP SWITCHES

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

Table 7.1. Bootstrap Switches for Disabling Decision 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 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 Business Automation support (for example processes support).

Property	Values	Default	Description
<code>org.optaplanner.server.ext.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	If set to <code>true</code> , disables the 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 Decision Server UI extension.
<code>org.kie.executor.disabled</code>	<code>true</code> , <code>false</code>	<code>false</code>	Disables the Red Hat Decision Manager executor.



NOTE

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

Table 7.2. 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 Decision Server from the controller, required when running in managed mode. Set this property in Decision Central system properties. Set this property when using a controller.
<code>org.kie.server.pwd</code>	String	<code>kieserver1!</code>	The password used to connect with the Decision Server from the controller, required when running in managed mode. Set this property in Decision Central system properties. Set this property when using a controller.

Property	Values	Default	Description
<code>org.kie.server.token</code>	String	N/A	A property that enables you to use token-based authentication between the controller and the Decision 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.
<code>org.kie.server.location</code>	URL	N/A	The URL of the Decision Server instance used by the controller to call back on this server, for example: http://localhost:8230/kie-server/services/rest/server . Setting this property is required when using a controller.
<code>org.kie.server.controller</code>	Comma-separated list	N/A	A comma-separated list of URLs to the controller REST endpoints, for example http://localhost:8080/decision-central/rest/controller . Setting this property is required when using a controller.
<code>org.kie.server.controller.user</code>	String	kieserve r	The user name to connect to the controller REST API. Setting this property is required when using a controller.
<code>org.kie.server.controller.pwd</code>	String	kieserve r1!	The password to connect to the controller REST API. Setting this property is required when using a controller.
<code>org.kie.server.controller.token</code>	String	N/A	A property that enables you to use token-based authentication between the Decision 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.
<code>org.kie.server.controller.connect</code>	Long	10000	The waiting time in milliseconds between repeated attempts to connect the Decision Server to the controller when the server starts.

Table 7.3. Bootstrap Switches for Executor Properties

Property	Values	Default	Description
<code>org.kie.executor.interval</code>	Integer	3	The time between the moment the Red Hat Decision Manager executor finishes a job and the moment it starts a new one, in a time unit specified in the <code>org.kie.executor.timeunit</code> property.
<code>org.kie.executor.timeunit</code>	<code>java.util.concurrent.TimeUnit</code> constant	SECONDS	The time unit in which the <code>org.kie.executor.interval</code> property is specified.
<code>org.kie.executor.pool.size</code>	Integer	1	The number of threads used by the Red Hat Decision Manager executor.
<code>org.kie.executor.retry.count</code>	Integer	3	The number of retries the Red Hat Decision Manager executor attempts on a failed job.

Table 7.4. Other Bootstrap Switches

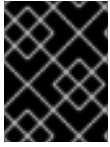
Property	Values	Default	Description
<code>kie.maven.settings.custom</code>	Path	N/A	The location of a custom <code>settings.xml</code> file for Maven configuration.
<code>kie.server.jms.queues.response</code>	String	queue/KIE.SERVER.RESPONSE	The response queue JNDI name for JMS.
<code>org.drools.server.filter.classes</code>	true , false	false	When set to true , the Drools Decision Server extension accepts custom classes annotated by the <code>XmlRootElement</code> or <code>Remotable</code> annotations only.
<code>org.kie.server.domain</code>	String	N/A	The JAAS <code>LoginContext</code> domain used to authenticate users when using JMS.
<code>org.kie.server.repo</code>	Path	.	The location where Decision Server state files will be stored.

Property	Values	Default	Description
<code>org.kie.server.sync.deploy</code>	<code>true</code> , <code>false</code>	<code>false</code>	<p>A property that instructs the Decision 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">• false; 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.• true; 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.

CHAPTER 8. MANAGED DECISION SERVER

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

A controller manages the Decision Server configuration in a centralized way. Each controller can manage multiple configurations at once, and there can be multiple controllers in the environment. Managed Decision 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 Decision 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 Decision Server in standalone mode without connecting to controllers, see [Section 8.2, “Unmanaged Decision Server”](#).

8.1. CONFIGURING DECISION SERVER MANAGED BY DECISION CENTRAL



WARNING

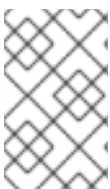
This section provides a sample setup that you can use for testing purposes. Some of the values are unsuitable for a production environment, and are marked as such.

Use this procedure to configure Decision Central to manage a Decision Server instance.

Prerequisite

Users with the following roles exist:

- In Decision Central, a user with the role **rest -all**.
- On the Decision 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 Decision Central and the Decision 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 8.1. JVM Properties for Managed Decision Server Instance

Property	Value	Note
<code>org.kie.server.id</code>	<code>default-kie-server</code>	The Decision Server ID.
<code>org.kie.server.controller</code>	http://localhost:8080/decision-central/rest/controller	The location of Decision Central.
<code>org.kie.server.controller.user</code>	<code>controllerUser</code>	The user name with the role rest-all 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>	http://localhost:8080/kie-server/services/rest/server	The location of the Decision Server.

Table 8.2. JVM Properties for Decision Central Instance

Property	Value	Note
<code>org.kie.server.user</code>	<code>controllerUser</code>	The user name with the role kie-server 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 Decision 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 Decision Central.
- b. Click **Menu** → **Deploy** → **Execution Servers**.
If registration is successful, you can see the registered server ID.

8.2. UNMANAGED DECISION SERVER

An unmanaged Decision Server is a standalone instance, and therefore must be configured individually using REST/JMS API from the Decision 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 Decision 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 9. CREATING CONTAINERS

Containers are self-contained environments that have been provisioned to hold instances of your packaged and deployed rule instances.

Prerequisite

Once the Decision Server is registered, you can start adding containers.

Procedure

1. Log in to Decision Central.
2. In the main menu, click **Menu** → **Deploy** → **Execution Servers**.
3. Click **SERVER TEMPLATES** → **+ New Server Template**.
4. In the **New Server Template** window, click **Container**.
5. Enter the **Group Name**, **Artifact Id**, and **Version** manually.
Alternatively, enter the name of your container and search for the project you want to deploy in the container.
6. Click **Select** next to the project to automatically enter the details of the project.



WARNING

When entering the container's version number, do *not* use the **LATEST** or **RELEASE** keywords. This feature has been deprecated and can cause deployment issues.

7. To deploy the container automatically, select the **Start Container?** box.
8. Click **Finish**.

After the container is successfully created, select the container from the list and click **Start** at the upper right hand corner to start it

To verify that the container is running, send a [GET] request to the endpoint.

Example 9.1. Server Response

```
<response type="SUCCESS" msg="Info for container myContainer">
  <kie-container container-id="myContainer" status="STARTED">
    <messages>
      <content>Container myContainer successfully created with module
org.jbpm:CustomersRelationship:1.0.</content>
      <severity>INFO</severity>
      <timestamp>TIMESTAMP</timestamp>
    </messages>
  </release-id>
```

```
<artifact-id>CustomersRelationship</artifact-id>
<group-id>org.jbpm</group-id>
<version>1.0</version>
</release-id>
<resolved-release-id>
  <artifact-id>CustomersRelationship</artifact-id>
  <group-id>org.jbpm</group-id>
  <version>1.0</version>
</resolved-release-id>
<scanner status="DISPOSED"/>
</kie-container>
</response>
```


CHAPTER 10. MANAGING CONTAINERS IN DECISION CENTRAL

Containers within the Decision Server can be started, stopped, and updated from Decision Central.

10.1. STARTING, STOPPING, AND DELETING CONTAINERS IN DECISION CENTRAL

When a container is created in Decision Central, it is stopped by default.

Use the following procedure to start the container:

Prerequisite

A container has been created and configured in Decision Central.

Procedure

1. Log in to Decision Central.
2. In the main menu, click **Menu** → **Deploy** → **Execution Servers**.
3. Select your server from the **SERVER TEMPLATES** section on the left side of the page.
4. Select the container you want to start under the **KIE CONTAINERS** section on the left.
5. Click **Start** at the upper right hand corner.
Alternatively, click **Stop** to stop a running container. Once a container is stopped, you can click **Remove** to remove it.

10.2. UPGRADING CONTAINERS IN DECISION CENTRAL

Deployed containers can be updated without restarting the Decision Server. This might be required in cases where business rule changes cause new versions of packages to be provisioned. You can have multiple versions of the same package provisioned and deployed.

Use the following procedure to upgrade a container:

Prerequisite

The Decision Server is configured with a container running in Decision Central.

Procedure

1. Log in to Decision Central.
2. In the main menu, click **Menu** → **Deploy** → **Execution Servers**.
3. Select your server from the **SERVER TEMPLATES** section on the left side of the page.
4. Select the container you want to upgrade under the **KIE CONTAINERS** section on the left.
5. Click on the **Version Configuration** tab at the top.
6. Enter a new version and click **Upgrade**.

7. Optionally, if you want a deployed container to always have the latest version of your deployment without manually editing it, set the **Version** value to **LATEST** and click **Scan Now**. If a newer version of a container deployment is found in the repository during the scanning, the container is automatically upgraded to this newer version. To start the scanner in the background, click **Start Scanner** and specify a scan interval in milliseconds.

The **Version** value can be set to **LATEST** if you are creating a deployment for the first time.

10.3. MANAGING MULTIPLE CONTAINERS

The Decision Server allows you to create and provision multiple containers. Use the following procedure to create and interact with multiple containers:

Prerequisite

The Decision Server has been registered and you are logged in to Decision Central.

Procedure

1. In the main menu, click **Menu** → **Deploy** → **Execution Servers**.
2. Select your server from the **SERVER TEMPLATES** section on the left side of the page.
3. Select your server under the **REMOTE SERVERS** section to view all containers and their statuses.

PART V. CONFIGURE THE RED HAT BUSINESS OPTIMIZER REST API



NOTE

All REST API calls use the following URL with the endpoint base URL:
`http://SERVER:PORT/decision-central/rest/REQUEST_BODY.`

CHAPTER 11. KNOWLEDGE STORE REST API

REST API calls to the Knowledge Store REST API allow you to manage the organization units, repositories, and projects.

All **POST** and **DELETE** calls return details about the request as well as a job ID that can be used to request the job status and verify whether the job finished successfully. The **GET** calls return information about repositories, projects, and organizational units.

Parameters and results of these calls are provided in the form of JSON entities. Java classes for different entities are available in the `org.guvnor.rest.client` package and are referenced in the following text.

11.1. JOB CALLS

Most Knowledge Store REST calls return a job ID after they are issued. This is necessary as the calls are asynchronous and it is required to be able to reference the job later to check its status as it goes through a job lifecycle.

During its lifecycle, a job can have the following statuses:

Table 11.1. Job Statuses

Status	Description
ACCEPTED	The job was accepted and is being processed.
BAD_REQUEST	The request was not accepted as it contained incorrect content.
RESOURCE_NOT_EXISTS	The requested resource (path) does not exist.
DUPLICATE_RESOURCE	The resource already exists.
SERVER_ERROR	An error on the server side occurred.
SUCCESS	The job finished successfully.
FAIL	The job failed.
APPROVED	The job was approved.
DENIED	The job was denied.

Status	Description
GONE	<p>The job ID could not be found. A job can be GONE in the following cases:</p> <ul style="list-style-type: none"> • The job was explicitly removed. • The job finished and has been deleted from a status cache. A job is removed from a status cache after the cache has reached its maximum capacity. • The job never existed.

The following job calls are provided:

[GET] /jobs/*JOB_ID*

Returns a status of the given *JOB_ID*.

Example 11.1. Formatted Response to GET Job Call on Repository Clone Request

```
{
  "status" : "SUCCESS",
  "jobId" : "1377770574783-27",
  "result" : "Alias: testInstallAndDeployProject, Scheme: git, Uri:
git://testInstallAndDeployProject",
  "lastModified" : 1377770578194,
  "detailedResult" : null
}
```

[DELETE] /jobs/*JOB_ID*

Removes a job with the given *JOB_ID*. If the job is not being processed yet, the call will remove the job from the job queue. However, this call will not cancel or stop an ongoing job.

Both of these job calls return a **JobResult** instance.

11.2. ORGANIZATIONAL UNIT CALLS

Organizational unit calls are calls to the Knowledge Store that allow you to manage its organizational units which are useful to model departments and divisions. An organization unit can hold multiple repositories.

The following organizational unit calls are provided:

[GET] /organizationalunits/

Returns a list of all organizational units.

Example 11.2. Organizational Unit List in JSON Format

```
[ {
  "name" : "EmployeeWage",
  "description" : null,
  "owner" : "Employee",
```

```

    "defaultGroupId" : "org.bpms",
    "repositories" : [ "EmployeeRepo", "OtherRepo" ]
  }, {
    "name" : "OrgUnitName",
    "description" : null,
    "owner" : "OrgUnitOwner",
    "defaultGroupId" : "org.group.id",
    "repositories" : [ "repository-name-1", "repository-name-2" ]
  } ]

```

[GET] /organizationalunits/*ORGANIZATIONAL_UNIT_NAME*

Returns information about a specific organizational unit.

[POST] /organizationalunits/

Creates an organizational unit in the Knowledge Store. The organizational unit is defined as a JSON entity. The call requires an **OrganizationalUnit** instance and returns a **CreateOrganizationalUnitRequest** instance.

Example 11.3. Organizational Unit in JSON Format

```

{
  "name" : "testgroup",
  "description" : "",
  "owner" : "tester",
  "repositories" : ["testGroupRepository"]
}

```

[POST] /organizationalunits/*ORGANIZATIONAL_UNIT_NAME*

Updates the details of an existing organizational unit.

Both the **name** and **owner** fields in the required **UpdateOrganizationalUnit** instance can be left empty. Neither the **description** field nor the repository association can be updated using this operation.

Example 11.4. Update Organizational Unit Input in JSON Format

```

{
  "owner" : "NewOwner",
  "defaultGroupId" : "org.new.default.group.id"
}

```

[DELETE] /organizationalunits/*ORGANIZATIONAL_UNIT_NAME*

Removes a specified organizational unit.

[POST] /organizationalunits/*ORGANIZATIONAL_UNIT_NAME*/repositories/*REPOSITORY_NAME*

Adds a repository to an organizational unit.

[DELETE]

/organizationalunits/*ORGANIZATIONAL_UNIT_NAME*/repositories/*REPOSITORY_NAME*

Removes a repository from an organizational unit.

11.3. REPOSITORY CALLS

Repository calls are calls to the Knowledge Store that allow you to manage its Git repositories and their projects.

The following repository calls are provided:

[GET] /repositories

Returns a list of repositories in the Knowledge Store.

Example 11.5. Response of Repository Call

```
[
  {
    "name": "bpms-assets",
    "description": "generic assets",
    "userName": null,
    "password": null,
    "requestType": null,
    "gitURL": "git://brms-assets"
  },
  {
    "name": "loanProject",
    "description": "Loan processes and rules",
    "userName": null,
    "password": null,
    "requestType": null,
    "gitURL": "git://loansProject"
  }
]
```

[GET] /repositories/*REPOSITORY_NAME*

Returns information about a specific repository.

[DELETE] /repositories/*REPOSITORY_NAME*

Removes a repository.

[POST] /repositories/

Creates or clones a repository defined by a JSON entity.

Example 11.6. JSON Entity with Details about Repository to Be Cloned

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

Example 11.7. JSON Entity with Details about Repository to Be Created

```
{
  "name": "myCreatedRepository",
  "organizationalUnitName": "example",
  "description": "",
  "userName": "",
  "password": "",
  "requestType": "create",
  "gitURL": "git://localhost/example-repository"
}
```



IMPORTANT

Make sure you always include the **organizationalUnitName** key-value pair in your query and that the specified organization unit exists before you create or clone the repository.

[GET] /repositories/*REPOSITORY_NAME*/projects/

Returns a list of projects in a specific repository as a JSON entity.

Example 11.8. JSON Entity with Details about Existing Projects

```
[ {
  "name" : "my-project-name",
  "description" : "A project to illustrate a REST output.",
  "groupId" : "com.acme",
  "version" : "1.0"
}, {
  "name" : "yet-another-project-name",
  "description" : "Yet another project to illustrate a REST output.",
  "groupId" : "com.acme",
  "version" : "2.2.1"
} ]
```

[POST] /repositories/*REPOSITORY_NAME*/projects/

Creates a project in a repository.

Example 11.9. Request Body That Defines Project to Be Created

```
{
  "name" : "NewProject",
  "description" : "Description of the new project.",
  "groupId" : "org.redhat.test",
  "version" : "1.0.0"
}
```

[DELETE] /repositories/*REPOSITORY_NAME*/projects/*PROJECT_NAME*

Removes a project in a repository.

PART VI. THE REST API FOR DECISION SERVER EXECUTION

CHAPTER 12. THE REST API FOR DECISION SERVER

You can communicate with the Decision Server through the REST API.

- The base URL for sending requests is the endpoint defined earlier, for example `http://SERVER:PORT/kie-server/services/rest/server/`.
- All requests require basic HTTP Authentication or token-based authentication for the role **kie-server**.

Following methods support three formats of the requests: JSON, JAXB, and XSTREAM. You must provide following HTTP headers:

- **Accept**: set to **application/json** or **application/xml**.
When specifying more than one accepted content type in the Accept header, be sure to include the qualifiers of preference (qvalues as defined in the [HTML 1.1 standard](#)). If you do not, unexpected behavior can occur. This is an example of a well-formed header with multiple accepted content types:

```
Accept: application/xml; q=0.5, application/json; q=0.9
```

- **X-KIE-ContentType** is required when using the XSTREAM marshaller. In such case, set the header to **XSTREAM**. Values **JSON** and **JAXB** are allowed, but not required. When you set the **Content-type** to **application/xml**, the **JAXB** value is used by default.
- **Content-type**: set to **application/json** or **application/xml**. This header corresponds with the format of your payload.
- **--data**: your payload. If the payload is in a file, start the name with an ampersand@. For example:

```
--data @commandsRequest.json
```

To ensure both the request and the response are in the same format, always specify both the **Content-Type** and **Accept** HTTP headers in your application's requests. Otherwise, you can receive a marshalling-related error from the server.

The examples use the Curl utility. You can use any REST client. Curl commands use the following parameters:

- **-u**: specifies username:password for the Decision Server authentication.
- **-H**: specifies HTTP headers.
- **-X**: specifies the HTTP method of the request, that is [GET], [POST], [PUT], or [DELETE].



NOTE

[Red Hat Decision Manager commands](#) will work only if your Decision Server has Red Hat Decision Manager capability. The rest of the endpoints will work only if your Decision Server has Red Hat Business Automation capabilities. Check the following URI for capabilities of your Decision Server : `http://SERVER:PORT/kie-server/services/rest/server`.

12.1. RED HAT DECISION MANAGER COMMANDS

[POST] /containers/instances/*CONTAINER_ID*

Request Type

A single `org.kie.api.command.Command` command or multiples commands in `BatchExecutionCommand` wrapper.

Response Type

`org.kie.server.api.model.ServiceResponse<String>`

Description

Executes the commands sent to the specified *CONTAINER_ID* and returns the commands execution results. For more information, see the supported commands below.

List of supported commands:

- `AgendaGroupSetFocusCommand`
- `ClearActivationGroupCommand`
- `ClearAgendaCommand`
- `ClearAgendaGroupCommand`
- `ClearRuleFlowGroupCommand`
- `DeleteCommand`
- `InsertObjectCommand`
- `ModifyCommand`
- `GetObjectCommand`
- `InsertElementsCommand`
- `FireAllRulesCommand`
- `QueryCommand`
- `SetGlobalCommand`
- `GetGlobalCommand`
- `GetObjectsCommand`
- `BatchExecutionCommand`
- `DisposeCommand`

For more information about the commands, see the `org.drools.core.command.runtime` package.

[POST] Drools Commands Execution

1. Change into a directory of your choice and create `commandsRequest.json` :

```

{
  "lookup" : null,
  "commands" : [ {
    "insert" : {
      "object" : "testing",
      "disconnected" : false,
      "out-identifier" : null,
      "return-object" : true,
      "entry-point" : "DEFAULT"
    }
  }, {
    "fire-all-rules" : { }
  } ]
}

```

2. Execute the following command:

```

$ curl -X POST -H 'X-KIE-ContentType: JSON' -H 'Content-type:
application/json' -u 'kieserver:kieserver1!' --data
@commandsRequest.json http://localhost:8080/kie-
server/services/rest/server/containers/instances/myContainer

```

The command generates a request that sends the Insert Object and Fire All Rules commands to the server. **Lookup** specifies a ksession configured in your kjar. If you use a null lookup value, the default KIE session will be used.

An example response:

```

{
  "type" : "SUCCESS",
  "msg" : "Container hello successfully called.",
  "result" : "{\n  \"results\" : [ ],\n  \"facts\" : [ ]\n}"
}

```

CHAPTER 13. CONFIGURE THE RED HAT BUSINESS OPTIMIZER REST API

The Decision Server supports the following Red Hat Business Optimizer REST APIs. These APIs are also available through JMS and the Java client API.

Prerequisite

Red Hat Decision Manager has been installed and configured with basic HTTP Authentication for the **kie-server** role. You have identified the server endpoint base URL, for example, <http://SERVER:PORT/CONTEXT/services/rest/server/>.

1. To get a specific marshalling format, add the HTTP headers **Content-Type** and optional **X-KIE-ContentType** in the HTTP request. For example:

```
Content-Type: application/xml
X-KIE-ContentType: xstream
```



NOTE

X-KIE-ContentType supports the following values: **xstream**, **xml**, **json**.

The requests and responses in the example below assume that a KIE container is built using the Employee Rostering example in Decision Central, by calling a **PUT** on `/services/rest/server/containers/employee-rostering` with this content:

```
<kie-container container-id="employee-rostering">
  <release-id>
    <group-id>employeerostering</group-id>
    <artifact-id>employeerostering</artifact-id>
    <version>1.0.0-SNAPSHOT</version>
  </release-id>
</kie-container>
```

13.1. [GET] /CONTAINERS

Returns the list of created containers.

Example 13.1. Example Server Response (XStream)

```
<response type="SUCCESS" msg="List of created containers">
  <result class="kie-containers">
    <kie-container>
      <container-id>employee-rostering</container-id>
      <release-id>
        <group-id>employeerostering</group-id>
        <artifact-id>employeerostering</artifact-id>
        <version>1.0.0-SNAPSHOT</version>
      </release-id>
      <resolved-release-id>
        <group-id>employeerostering</group-id>
        <artifact-id>employeerostering</artifact-id>
```

```

        <version>1.0.0-SNAPSHOT</version>
      </resolved-release-id>
      <status>STARTED</status>
      <scanner>
        <status>DISPOSED</status>
      </scanner>
    </kie-container>
  </result>
</response>

```

13.2. [PUT] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}

Creates a new solver with the given **{solverId}** in the container **{containerId}**. The request's body is a marshalled **SolverInstance** entity that must specify the solver configuration file.

The following is an example of the request and the corresponding response.

Example 13.2. Example Server Request (XStream)

```

<solver-instance>
  <solver-config-
file>employee-rostering/employee-rostering/EmployeeRosteringSolverConfig.s
olver.xml</solver-config-file>
</solver-instance>

```

Example 13.3. Example Server Response (XStream)

```

<solver-instance>
  <container-id>employee-rostering</container-id>
  <solver-id>solver1</solver-id>
  <solver-config-
file>employee-rostering/employee-rostering/EmployeeRosteringSolverConfig.s
olver.xml</solver-config-file>
  <status>NOT_SOLVING</status>
  <score/>
</solver-instance>

```

13.3. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS

Returns the list of solvers created in the container.

Example 13.4. Example Server Response (XStream)

```

<org.kie.server.api.model.instance.SolverInstanceList>
  <solvers>
    <solver-instance>
      <container-id>employee-rostering</container-id>
      <solver-id>solver1</solver-id>
      <solver-config-

```

```
file>employee rostering/employee rostering/EmployeeRosteringSolverConfig.s
olver.xml</solver-config-file>
  <status>NOT_SOLVING</status>
  <score/>
</solver-instance>
</solvers>
</org.kie.server.api.model.instance.SolverInstanceList>
```

13.4. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}

Returns the current state of the solver {solverId} in container {containerId}.

Example 13.5. Example Server Response (XStream)

```
<solver-instance>
  <container-id>employee-rostering</container-id>
  <solver-id>solver1</solver-id>
  <solver-config-
file>employee rostering/employee rostering/EmployeeRosteringSolverConfig.s
olver.xml</solver-config-file>
  <status>NOT_SOLVING</status>
  <score/>
</solver-instance>
```

13.5. [POST]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/SOLVING

Starts the solver {solverId} in container {containerId} if it is not executing yet. The request's body is a marshalled **PlanningSolution** to be optimized.

The following is an example to solve the OptaCloud problem with 2 computers and 6 processes. The solver runs asynchronously. Send a request to the bestsolution URL to get the best solution.

Example 13.6. Example Server Request (XStream)

```
<employee rostering.employee rostering.EmployeeRoster>
  <employeeList>
    <employee rostering.employee rostering.Employee>
      <name>John</name>
      <skills>
        <employee rostering.employee rostering.Skill>
          <name>reading</name>
        </employee rostering.employee rostering.Skill>
      </skills>
    </employee rostering.employee rostering.Employee>
    <employee rostering.employee rostering.Employee>
      <name>Mary</name>
      <skills>
        <employee rostering.employee rostering.Skill>
          <name>writing</name>
        </employee rostering.employee rostering.Skill>
```

```

    </skills>
  </employee rostering.employee rostering.Employee>
<employee rostering.employee rostering.Employee>
  <name>Petr</name>
  <skills>
    <employee rostering.employee rostering.Skill>
      <name>speaking</name>
    </employee rostering.employee rostering.Skill>
  </skills>
</employee rostering.employee rostering.Employee>
</employeeList>
<shiftList>
  <employee rostering.employee rostering.Shift>
    <timeslot>
      <startTime>2017-01-01T00:00:00</startTime>
      <endTime>2017-01-01T01:00:00</endTime>
    </timeslot>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Emp
loyee/skills/employee rostering.employee rostering.Skill"/>
  </employee rostering.employee rostering.Shift>
<employee rostering.employee rostering.Shift>
  <timeslot
reference="../../../../employee rostering.employee rostering.Shift/timeslot"/>
  <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Emp
loyee[3]/skills/employee rostering.employee rostering.Skill"/>
  </employee rostering.employee rostering.Shift>
<employee rostering.employee rostering.Shift>
  <timeslot
reference="../../../../employee rostering.employee rostering.Shift/timeslot"/>
  <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Emp
loyee[2]/skills/employee rostering.employee rostering.Skill"/>
  </employee rostering.employee rostering.Shift>
</shiftList>
<skillList>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employ
ee/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employ
ee[3]/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employ
ee[2]/skills/employee rostering.employee rostering.Skill"/>
  </skillList>
<timeslotList>
  <employee rostering.employee rostering.Timeslot
reference="../../../../shiftList/employee rostering.employee rostering.Shift/tim
eslot"/>
  </timeslotList>
<dayOffRequestList/>
<shiftAssignmentList/>
</employee rostering.employee rostering.EmployeeRoster>

```


13.6. [POST]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/TERMINATE EARLY

Requests the solver to terminate early, if it is running. This does not delete the solver, the best solution can still be retrieved.

13.7. [GET]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/BESTSOLUTION

Returns the best solution found at the time the request is made. If the solver has not terminated yet (so the **status** field is still **SOLVING**), it will return the best solution found up to then, but later calls can return a better solution.

Example 13.7. Example Server Response (XStream)

```

<solver-instance>
  <container-id>employee-rostering</container-id>
  <solver-id>solver1</solver-id>
  <solver-config-
file>employee-rostering/employee-rostering/EmployeeRosteringSolverConfig.s
olver.xml</solver-config-file>
  <status>NOT_SOLVING</status>
  <score
scoreClass="org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScor
e">0hard/0soft</score>
  <best-solution
class="employee-rostering.employee-rostering.EmployeeRoster">
  <employeeList>
    <employee-rostering.employee-rostering.Employee>
      <name>John</name>
      <skills>
        <employee-rostering.employee-rostering.Skill>
          <name>reading</name>
        </employee-rostering.employee-rostering.Skill>
      </skills>
    </employee-rostering.employee-rostering.Employee>
    <employee-rostering.employee-rostering.Employee>
      <name>Mary</name>
      <skills>
        <employee-rostering.employee-rostering.Skill>
          <name>writing</name>
        </employee-rostering.employee-rostering.Skill>
      </skills>
    </employee-rostering.employee-rostering.Employee>
    <employee-rostering.employee-rostering.Employee>
      <name>Petr</name>
      <skills>
        <employee-rostering.employee-rostering.Skill>
          <name>speaking</name>
        </employee-rostering.employee-rostering.Skill>
      </skills>
    </employee-rostering.employee-rostering.Employee>
  </employeeList>

```

```

<shiftList>
  <employee rostering.employee rostering.Shift>
    <timeslot>
      <startTime>2017-01-01T00:00:00</startTime>
      <endTime>2017-01-01T01:00:00</endTime>
    </timeslot>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Employee/skills/employee rostering.employee rostering.Skill"/>
    </employee rostering.employee rostering.Shift>
  <employee rostering.employee rostering.Shift>
    <timeslot
reference="../../../../employee rostering.employee rostering.Shift/timeslot"/>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Employee[3]/skills/employee rostering.employee rostering.Skill"/>
    </employee rostering.employee rostering.Shift>
  <employee rostering.employee rostering.Shift>
    <timeslot
reference="../../../../employee rostering.employee rostering.Shift/timeslot"/>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Employee[2]/skills/employee rostering.employee rostering.Skill"/>
    </employee rostering.employee rostering.Shift>
</shiftList>
<skillList>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employee/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employee[3]/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Employee[2]/skills/employee rostering.employee rostering.Skill"/>
</skillList>
<timeslotList>
  <employee rostering.employee rostering.Timeslot
reference="../../../../shiftList/employee rostering.employee rostering.Shift/timeslot"/>
</timeslotList>
<dayOffRequestList/>
<shiftAssignmentList/>
<score>0hard/0soft</score>
</best-solution>
</solver-instance>

```

13.8. [POST]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMFAC

Real-time planning feature. Submits one or multiple ProblemFactChanges to update the dataset the solver currently optimizes.

13.9. [GET]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMFAC

Real-time planning feature. Returns true if the solver processed all ProblemFactChanges that had been submitted. Returns false otherwise.

13.10. [DELETE]

/CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}

Disposes the solver `{solverId}` in container `{containerId}`. If it has not terminated yet, it terminates it first.

PART VII. AUTHOR PLANNING ASSETS AND CREATE SOLVERS

CHAPTER 14. CREATE A SOLVER IN DECISION CENTRAL

The Solver editor creates a solver configuration that can be run in the Execution Solver or plain Java code after the KJAR is deployed.

You can edit and create Solver configurations in Decision Central.

Prerequisite

Red Hat Decision Manager has been downloaded and installed. Your project has been designed and deployed with all relevant assets configured.

This example uses the provided Employee Rostering sample project to demonstrate Solver features.

Procedure

1. In Decision Central, click **Menu** → **Projects**, and click on your project to open it.
2. In the **Assets** perspective, click **Create New Asset** → **Solver configuration**
3. In the **Create new Solver configuration** window, type a name for your Solver and click **Ok**. This opens the **Solver configuration** editor.
4. In the **Score Director Factory** configuration section, define a knowledge base that contains scoring rule definitions.
 - a. Select one of the knowledge sessions defined within the knowledge base. The sessions can be managed in the **Project** perspective in Decision Central by clicking **Menu** → **Projects**.



NOTE

Red Hat Business Optimizer uses a default knowledge session if none is specified.

Score Director Factory

Knowledge base	<input type="text" value="defaultKieBase"/>
Knowledge session	<input type="text" value="defaultKieSession"/>

5. Click the **Validate** button to validate the Solver configuration. This will actually build a Solver, which will notify you of most issues in your project without the need to deploy and run it.

By default, the Solver configuration automatically scans for all planning entities and planning solution classes. If none are found (or too many), validation fails.

14.1. CONFIGURE SOLVER TERMINATION

By default, the planning engine is given an unlimited time period to solve a problem instance.

This might be enforced by some scenarios (for example, real time planning), it is useful to have a mechanism to control total duration of the solving process.

Refer to [OptaPlanner documentation](#) for more information on supported termination types.



NOTE

The Solver can be terminated manually using the Decision Server REST API.

The screenshot shows the 'Termination' configuration panel. It features a tree structure of termination elements. The root element is 'Time spent' with a logical operator of 'Or'. It has three child elements: 'Unimproved time spent' (operator 'And'), 'Best score feasible' (checkbox checked), and another 'Time spent' element (operator 'Or'). Each element has a 'Remove' button.

Prerequisite

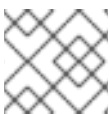
You have deployed a project and created a Solver in Decision Central.

Procedure

1. Open the **Solver editor**,
2. In the **Termination** section, click **Add** to create new termination element within the selected logical group.
3. Select a termination type from the drop-down list. This is added as an input field in the termination configuration.
Termination elements are organized into a tree structure.

The editor supports definition of logical groups (represented by termination type **Nested termination**), which join multiple termination elements using logical operators (**And/Or**). The scope of the operator is limited by the logical group in which it is defined.

Click **Remove** to remove the termination element from the termination tree.




NOTE

Removing the root element of a logical group removes its child elements.

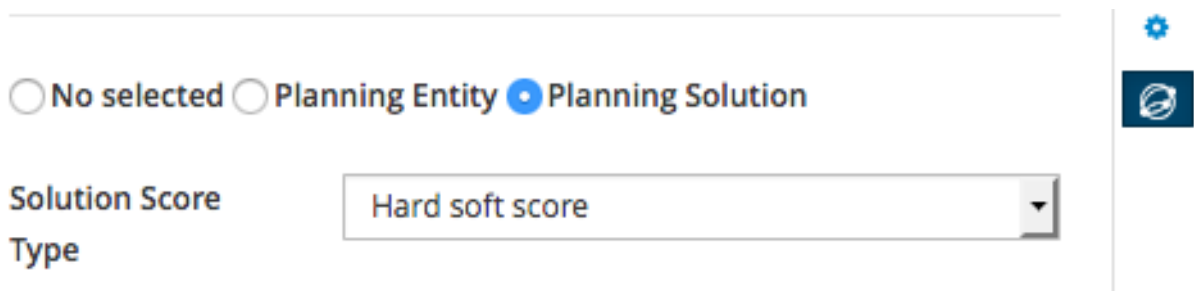
CHAPTER 15. AUTHOR PLANNING ASSETS WITH THE RED HAT BUSINESS OPTIMIZER DOMAIN EDITOR

Red Hat Business Optimizer leverages the data modeller to create a domain model for constraint satisfaction problems.

In addition to its basic functionality (creating data objects and their properties), you can also use the **Data Objects** editor perspective in Decision Central to enhance the data model with Red Hat Business Optimizer-specific data object roles. The **Planning Entity** and **Planning Solution** roles are

available in the Red Hat Business Optimizer dock, accessed by clicking the  on the right side of the **Data Objects** perspective.

These options are available in the Red Hat Business Optimizer domain editor.



The content of the domain editor varies depending on the current selection:

- Selecting a data object displays settings defined on the data object level ([Planning Solution](#), [Planning Entity](#)).
- Selecting properties of the data object displays settings defined on the property level of the data object.

The overview of the Red Hat Business Optimizer domain editor capabilities is shown in the following division:

Data object level

- None
- [Planning Entity](#)
 - [Configure a planning entity difficulty comparator](#)
- [Planning Solution](#)
 - [Solution Score Type](#)

Property level

- None
- [Planning Entity](#)
 - [Planning Variable](#)
 - [valueRanged](#)

- [Planning Solution](#)
 - [Planning Value Range Provider](#)
 - [Planning Value Range Provider id](#)
 - [Planning Entity Collection](#)

CHAPTER 16. CONFIGURE A PLANNING ENTITY DIFFICULTY COMPARATOR

Specified on **Planning Entity** level, a difficulty comparator provides a way to determine which planning entities are more difficult to plan. This helps optimization algorithms to work in an efficient manner. Refer to [OptaPlanner documentation](#) for more details.

Prerequisite

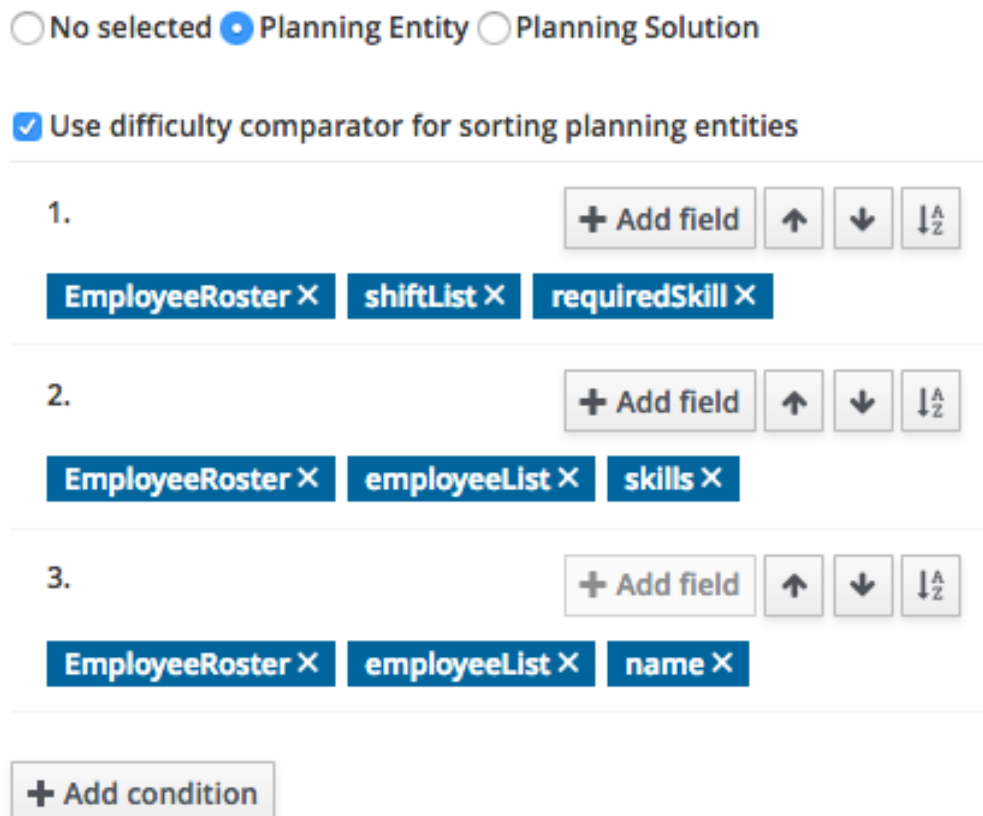
You have selected **Planning Entity** on a data object in the **Data Object** editor and the difficulty comparator definition tool is available in the Red Hat Business Optimizer Domain editor in the **Data Object** perspective.

Procedure

1. Click **Add condition** to add new sorting criteria for given planning entity.
2. Once the condition is added, click **Add field** to select fields that will be used for difficulty comparison.

There are two types of fields:

- **Basic** - value types (for example, number, String)
- **Data object** - complex types having nested attributes
Data object types allow nesting deep into object hierarchy until a basic type is encountered. When this occurs, the **Add field** button is no longer displayed.



The screenshot shows the configuration interface for a difficulty comparator. At the top, there are three radio buttons: "No selected", "Planning Entity" (which is selected), and "Planning Solution". Below this, there is a checked checkbox labeled "Use difficulty comparator for sorting planning entities".

There are three numbered sections for adding sorting criteria:

- 1.** Contains a "+ Add field" button and three sorting icons (up, down, and a combined up/down with a fraction 1/2). Below these are three blue buttons with white text and a close icon: "EmployeeRoster X", "shiftList X", and "requiredSkill X".
- 2.** Contains a "+ Add field" button and three sorting icons. Below these are three blue buttons: "EmployeeRoster X", "employeeList X", and "skills X".
- 3.** Contains a "+ Add field" button and three sorting icons. Below these are three blue buttons: "EmployeeRoster X", "employeeList X", and "name X".


At the bottom of the configuration area, there is a "+ Add condition" button.

Sorting criteria are ordered. The ones defined first are prioritized when Red Hat Business Optimizer engine resolves planning entity difficulty.

3. To remove a field from the sorting criteria, click the **x** icon within the label. If the field is of type **Data object**, removing it will remove all its children as well.
 - Click **Arrow up**, **Arrow down** to change the priority of the criterion by moving it up/down.
 - Select **Sort order** icon to define whether the planning entity criteria is sorted in ascending or descending order.

CHAPTER 17. DEFINE SCORE CONSTRAINTS USING THE GUIDED RULES DESIGNER

To solve an optimization problem, you need to define score constraints that evaluate your solution. Red Hat Business Optimizer integrates with the guided rules designer and provides score modifiers, which are used by the engine during the solving process.


Score modifiers can be accessed in the action selector () of the **THEN** (right-hand side) section of a rule, provided the planning solution is defined within the project.

Use the following procedure to create a new score constraint using the guided rules designer.

Prerequisite

You have defined a [Planning Solution](#) by creating a class in the domain editor and selecting **Planning Solution**. See [Chapter 15, Author planning assets with the Red Hat Business Optimizer domain editor](#) for more information.





Procedure

1. Open the project **Assets** screen by clicking **Menu** → **Projects**, and clicking on your project to open it.
2. Click **Create New Asset** in the upper-right corner.
3. Select **Guided Rule** from the drop-down menu.
4. In the **Create new Guided Rule** window, enter a name for the rule and click **Ok**.
5. If you want to extend a score constraint you have already created, click the drop-down box next to **EXTEND** and select the rule from the list.
6. Click the plus icon () on the right side of the **WHEN** section to add a condition.
7. Once the action is selected, Business Optimizer score input appears on the **THEN** (right-hand side) section of the rule.

The following Red Hat Business Optimizer actions are available in the guided rules designer:

- **Modify a single score level** - use the action to modify only one score component (e.g. hard score)
 - **Modify multiple score levels** - use the action to modify multiple score components at the same time (e.g. hard and soft score)
8. Insert the value of a constraint into the text input.
 9. Click **Validate** to verify the correctness of the inserted value.

THEN

1. Hard Score	-1	    
---------------	----	--

For more information about using the guided rule designer, see [Designing a decision service using guided rules](#).

APPENDIX A. VERSIONING INFORMATION

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