



## **Red Hat Decision Manager 7.0**

**Creating and running a Red Hat Business  
Optimizer solver for employee rostering using  
Decision Central**



# Red Hat Decision Manager 7.0 Creating and running a Red Hat Business Optimizer solver for employee rostering using Decision Central

---

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 create and run the employee rostering example for Red Hat Business Optimizer using Decision Central in Red Hat Decision Manager 7.0.

## Table of Contents

<b>PREFACE</b>	<b>3</b>
<b>CHAPTER 1. DEPLOY THE EMPLOYEE ROSTERING SAMPLE PROJECT IN DECISION CENTRAL</b>	<b>4</b>
<b>CHAPTER 2. SET UP THE EMPLOYEE ROSTERING SAMPLE PROJECT</b>	<b>5</b>
<b>CHAPTER 3. PROBLEM FACTS AND PLANNING ENTITIES</b>	<b>6</b>
<b>CHAPTER 4. CREATE THE DATA MODEL FOR THE EMPLOYEE ROSTERING PROJECT</b>	<b>7</b>
4.1. CREATE THE EMPLOYEE ROSTER PLANNING ENTITY	8
4.2. CREATE THE EMPLOYEE ROSTER PLANNING SOLUTION	9
<b>CHAPTER 5. EMPLOYEE ROSTERING CONSTRAINTS</b>	<b>11</b>
5.1. DRL RULES	11
5.2. CONSTRAINT DEFINITION FOR EMPLOYEE ROSTERING USING THE DRL DESIGNER	11
<b>CHAPTER 6. CREATE RULES FOR EMPLOYEE ROSTERING USING GUIDED RULES</b>	<b>13</b>
6.1. GUIDED RULES	13
6.2. CREATE A GUIDED RULE TO BALANCE EMPLOYEE SHIFT NUMBERS	13
6.3. CREATE A GUIDED RULE FOR NO MORE THAN ONE SHIFT PER DAY	14
6.4. CREATE A GUIDED RULE TO MATCH SKILLS TO SHIFT REQUIREMENTS	16
6.5. CREATE A GUIDED RULE TO MANAGE DAY OFF REQUESTS	17
<b>CHAPTER 7. CONFIGURE THE SOLVER FOR EMPLOYEE ROSTERING</b>	<b>19</b>
7.1. CONFIGURE SOLVER TERMINATION FOR THE EMPLOYEE ROSTERING PROJECT	19
7.2. REGISTER THE SOLVER USING THE REST API	20
<b>APPENDIX A. VERSIONING INFORMATION</b>	<b>25</b>



## PREFACE

As a business rules developer, you can test and interact with the Red Hat Business Optimizer functionality using the preconfigured **employee-rostering** sample project included in the Red Hat Decision Manager distribution.

The **employee-rostering** sample project can be built and deployed in Decision Central. The project is designed to demonstrate how to create each of the Decision Central assets required to solve the shift rostering planning problem and use Red Hat Business Optimizer to find the best possible solution. Use this document to deploy the preconfigured **employee-rostering** project in Decision Central, or to create the project yourself using Decision Central.



### NOTE

The **employee-rostering** sample project in Decision Central does not include a data set.

### Prerequisites

- Red Hat JBoss Enterprise Application Platform 7.1.0 installed. See [Red Hat JBoss EAP 7.1.0 Installation Guide](#).
- Red Hat Decision Manager installed. For more information, see [Installing Red Hat Decision Manager on premise](#).
- Red Hat Decision Manager is running and you can log in to Decision Central with the **admin** role. For more information, see [Installing Red Hat Decision Manager on premise](#).
- Red Hat Decision Manager is configured for Red Hat Business Optimizer. For more information about the required configuration, see [Installing and configuring Red Hat Business Optimizer](#).
- If you are modifying this example to optimize your own data, a data set is required in order to run an optimization task on the Decision Server after the modified project has been deployed from Decision Central to the server.

# CHAPTER 1. DEPLOY THE EMPLOYEE ROSTERING SAMPLE PROJECT IN DECISION CENTRAL

Decision Central includes a number of sample projects that you can use to get familiar with the product and its features. The employee rostering sample project has been designed and created to demonstrate the shift rostering use case for Red Hat Business Optimizer. Use the following procedure to deploy and run the employee rostering sample in Decision Central.

## Prerequisites

- Red Hat Decision Manager has been downloaded and installed.
- You have started the decision server and logged in to Decision Central with a user that has **admin** permissions.
- For more information about installation see [Installing Red Hat Decision Manager on premise](#).
- For more information about getting started, see [Getting started with decision services](#).

## Procedure

1. In Decision Central, click **Menu** → **Design** → **Projects**.
2. In the preconfigured **myteam** space, click **Try Samples**.
3. Select **employee-rostering** from the list of sample projects and click **Ok** in the upper-right corner to import the project.
4. After the asset list has compiled, click **Build & Deploy** to deploy the employee rostering example.

The rest of this document explains each of the project assets and their configuration.



## CHAPTER 2. SET UP THE EMPLOYEE ROSTERING SAMPLE PROJECT

The employee rostering sample project is a preconfigured project available in Decision Central. You can learn about how to deploy this project in [Chapter 1, \*Deploy the employee rostering sample project in Decision Central\*](#).

This chapter describes how to set up the employee rostering example. You can use the workflow in this example to create a similar project of your own in Decision Central.

### Prerequisites

- Red Hat Decision Manager has been downloaded and installed.
- You have deployed Decision Central and logged in with a user that has the **admin** role.

### Procedure

1. Create a new project in Decision Central by clicking **Menu** → **Design** → **Projects** → **Add Project**.
2. In the **Add Project** window, fill out the following fields:
  - **Name:** `employee-rostering`
  - **Description**(optional): Employee rostering problem optimization using Business Optimizer. Assigns employees to shifts based on their skill.

Optionally, click **Show Advanced Options** to populate the **Group ID**, **Artifact ID**, and **Version** information.

- **Group ID:** `employeerostering`
  - **Artifact ID:** `employeerostering`
  - **Version:** `1.0.0-SNAPSHOT`
3. Click **Add** to add the project to the Decision Central project repository.

## CHAPTER 3. PROBLEM FACTS AND PLANNING ENTITIES

Each of the domain classes in the employee rostering planning problem can be categorized as one of the following:

- A *unrelated class*: not used by any of the score constraints. From a planning standpoint, this data is obsolete.
- A *problem fact* class: used by the score constraints, but does not change during planning (as long as the problem stays the same), for example, **Shift** and **Employee**. All the properties of a problem fact class are problem properties.
- A *planning entity* class: used by the score constraints and changes during planning, for example, **ShiftAssignment**. The properties that change during planning are *planning variables*. The other properties are problem properties.

Ask yourself the following questions:

- *What class changes during planning?*
- *Which class has variables that I want the **Solver** to change?*

That class is a planning entity.

A planning entity class needs to be annotated with the **@PlanningEntity** annotation, or defined in Decision Central using the Red Hat Business Optimizer dock in the domain designer.

Each planning entity class has one or more *planning variables*, and should also have one or more defining properties.

Most use cases have only one planning entity class, and only one planning variable per planning entity class.

## CHAPTER 4. CREATE THE DATA MODEL FOR THE EMPLOYEE ROSTERING PROJECT

Use this section to create the data objects required to run the employee rostering sample project in Decision Central.

### Prerequisite

You have completed the project set up described in [Chapter 2, Set up the employee rostering sample project](#).

### Procedure

1. With your new project, either click **Data Object** in the project perspective, or click **Create New Asset** → **Data Object** to create a new data object.
2. Name the first data object **Timeslot**, and select **employee rostering.employee rostering** as the **Package**. Click **Ok**.
3. In the **Data Objects** perspective, click **+add field** to add fields to the **Timeslot** data object.
4. In the **id** field, type **endTime**.
5. Click the drop-down menu next to **Type** and select **LocalDateTime**.
6. Click **Create and continue** to add another field.
7. Add another field with the **id** **startTime** and **Type** **LocalDateTime**.
8. Click **Create**.
9. Click **Save** in the upper-right corner to save the **Timeslot** data object.
10. Click the **x** in the upper-right corner to close the **Data Objects** perspective and return to the **Assets** menu.
11. Using the previous steps, create the following data objects and their attributes:

**Table 4.1. Skill**

id	Type
<b>name</b>	<b>String</b>

**Table 4.2. Employee**

id	Type
<b>name</b>	<b>String</b>
<b>skills</b>	<b>employee rostering.employee rostering.Skill[List]</b>

Table 4.3. Shift

id	Type
requiredSkill	employee rostering.employee rostering.Skill
timeslot	employee rostering.employee rostering.Timeslot

Table 4.4. DayOffRequest

id	Type
date	LocalDate
employee	employee rostering.employee rostering.Employee

Table 4.5. ShiftAssignment

id	Type
employee	employee rostering.employee rostering.Employee
shift	employee rostering.employee rostering.Shift

For more information about creating data objects, see [Getting started with decision services](#).

## 4.1. CREATE THE EMPLOYEE ROSTER PLANNING ENTITY

In order to solve the employee rostering planning problem, you must create a planning entity and a solver. The planning entity is defined in the domain designer using the attributes available in the Red Hat Business Optimizer dock.

Use the following procedure to define the **ShiftAssignment** data object as the planning entity for the employee rostering example.

### Prerequisite

- You have created the relevant data objects and planning entity required to run the employee rostering example by completing the procedures in [Chapter 4, Create the data model for the employee rostering project](#).

### Procedure

- From the project **Assets** menu, open the **ShiftAssignment** data object.

2. In the **Data Objects** perspective, open the Red Hat Business Optimizer dock by clicking the



on the right.

3. Select **Planning Entity**.
4. Select **employee** from the list of fields under the **ShiftAssignment** data object.
5. In the Red Hat Business Optimizer dock, select **Planning Variable**.  
In the **Value Range Id** input field, type **employeeRange**. This adds the **@ValueRangeProvider** annotation to the planning entity, which you can view by clicking the **Source** tab in the designer.

The value range of a planning variable is defined with the **@ValueRangeProvider** annotation. A **@ValueRangeProvider** annotation always has a property **id**, which is referenced by the **@PlanningVariable** property **valueRangeProviderRefs**.

6. Close the dock and click **Save** to save the data object.

## 4.2. CREATE THE EMPLOYEE ROSTER PLANNING SOLUTION

The employee roster problem relies on a defined planning solution. The planning solution is defined in the domain designer using the attributes available in the Red Hat Business Optimizer dock.

### Prerequisite

- You have created the relevant data objects and planning entity required to run the employee rostering example by completing the procedures in [Chapter 4, Create the data model for the employee rostering project](#) and [Section 4.1, “Create the employee roster planning entity”](#).

### Procedure

1. Create a new data object with the identifier **EmployeeRoster**.
2. Create the following fields:

**Table 4.6. EmployeeRoster**

id	Type
<b>dayOffRequestList</b>	<code>employee rostering.EmployeeRoster.dayOffRequest[List]</code>
<b>shiftAssignmentList</b>	<code>employee rostering.EmployeeRoster.shiftAssignment[List]</code>
<b>shiftList</b>	<code>employee rostering.EmployeeRoster.shift[List]</code>
<b>skillList</b>	<code>employee rostering.EmployeeRoster.skill[List]</code>

id	Type
<b>timeslotList</b>	<b>employee rostering.employee rostering.Timeslot[List]</b>

- In the **Data Objects** perspective, open the Red Hat Business Optimizer dock by clicking the



on the right.

- Select **Planing Solution**.
- Leave the default **Hard soft score** as the **Solution Score Type**. This automatically generates a **score** field in the **EmployeeRoster** data object with the solution score as the type.
- Add a new field with the following attributes:

id	Type
<b>employeeList</b>	<b>employee rostering.employee rostering.Employee[List]</b>

- With the **employeeList** field selected, open the Red Hat Business Optimizer dock and select the **Planning Value Range Provider** box.  
In the **id** field, type **employeeRange**. Close the dock.
- Click **Save** in the upper-right corner to save the asset.

## CHAPTER 5. EMPLOYEE ROSTERING CONSTRAINTS

Employee rostering is a planning problem. All planning problems include constraints that must be satisfied in order to find an optimal solution.

The employee rostering sample project in Decision Central includes the following hard and soft constraints:

### Hard constraint

- Employees are only assigned one shift per day.
- All shifts that require a particular employee skill are assigned an employee with that particular skill.

### Soft constraints

- All employees are assigned a shift.
- If an employee requests a day off, their shift can be reassigned to another employee.

Hard and soft constraints can be defined in Decision Central using either the free-form DRL designer, or using guided rules.

For more information about hard and soft constraints, see [Installing and configuring Red Hat Business Optimizer](#).

## 5.1. DRL RULES

DRL rules are business rules that you define directly in **.drl** text files. These DRL files are the source in which all other rule assets in Decision Central are ultimately rendered. You can create and manage DRL files within the Decision Central interface, or create them externally using Red Hat Developer Studio, Java objects, or Maven archetypes. A DRL file can contain one or more rules that define at minimum the rule conditions (**when**) and actions (**then**). The DRL designer in Decision Central provides syntax highlighting for Java, DRL, and XML.

All data objects related to a DRL rule must be in the same project package as the DRL rule in Decision Central. Assets in the same package are imported by default. Existing assets in other packages can be imported with the DRL rule.

## 5.2. CONSTRAINT DEFINITION FOR EMPLOYEE ROSTERING USING THE DRL DESIGNER

You can create constraint definitions for the employee rostering example using the free-form DRL designer in Decision Central.

Use this procedure to create a *hard constraint* where no employee can be assigned a shift that begins less than 10 hours after their previous shift ended.

### Procedure

1. Go to **Menu** → **Design** → **Projects** and click the project name.
2. Click **Create New Asset** → **DRL file**.

3. In the **DRL file** name field, type **ComplexScoreRules**.
4. Select the **employee rostering.employee rostering** package.
5. Click **+Ok** to create the DRL file.
6. In the **Editor** tab of the DRL designer, define the **Employee10HourShiftSpace** rule as a DRL file:

```
package employee rostering.employee rostering;

rule "Employee10HourShiftSpace"
    dialect "mvel"
    when
        $shiftAssignment : ShiftAssignment( $employee : employee !=
null, $shiftEndTime : shift.timeslot.endTime)
        ShiftAssignment( this != $shiftAssignment, $employee ==
employee, $shiftEndTime <= shift.timeslot.endTime,
                        $shiftEndTime.until(shift.timeslot.startTime,
java.time.temporal.ChronoUnit.HOURS) <10)
    then
        scoreHolder.addHardConstraintMatch(kcontext, -1);
    end
```

7. Click **Save** to save the DRL file.

For more information about creating DRL files, see [Designing a decision service using DRL rules](#).



## CHAPTER 6. CREATE RULES FOR EMPLOYEE ROSTERING USING GUIDED RULES

You can create rules that define hard and soft constraints for employee rostering using the guided rules designer in Decision Central.

### 6.1. GUIDED RULES

Guided rules are business rules that you create in a UI-based guided rules designer in Decision Central that leads you through the rule-creation process. The guided rules designer provides fields and options for acceptable input based on the data objects for the rule being defined. The guided rules that you define are compiled into Drools Rule Language (DRL) rules as with all other rule assets.


All data objects related to a guided rule must be in the same project package as the guided rule. Assets in the same package are imported by default. After the necessary data objects and the guided rule are created, you can use the **Data Objects** tab of the guided rules designer to verify that all required data objects are listed or to import other existing data objects by adding a **New item**.



### 6.2. CREATE A GUIDED RULE TO BALANCE EMPLOYEE SHIFT NUMBERS

The **BalanceEmployeesShiftNumber** guided rule creates a soft constraint that ensures shifts are assigned to employees in a way that is balanced as evenly as possible. It does this by creating a score penalty that increases when shift distribution is less even. The score formula, implemented by the rule, incentivizes the Solver to distribute shifts in a more balanced way.

The screenshot shows the 'BalanceEmployeesShiftNumber.rdr1 - Guided Rules' window. It has tabs for 'Editor', 'Overview', 'Source', and 'Data Objects'. The 'Editor' tab is active, showing the rule structure. At the top, there are buttons: 'Save', 'Delete', 'Rename', 'Copy', 'Validate', 'Latest Version', and a close button. Below the tabs, there's an 'EXTENDS' dropdown set to '- None -'. The 'WHEN' section contains two conditions: 1. 'There is an Employee [Employee]' and 'There is a Number [ShiftCount]'. 2. 'From Accumulate All ShiftAssignment [ShiftAssignment] with: employee equal to \$employee'. A 'Function' button is visible. The 'THEN' section contains a 'Soft Score' condition with the formula '(\$shiftCount.intValue() \* \$shiftCount.intValue())'. At the bottom, there's a 'Messages' section with a 'Clear' button and a refresh icon.

#### Procedure

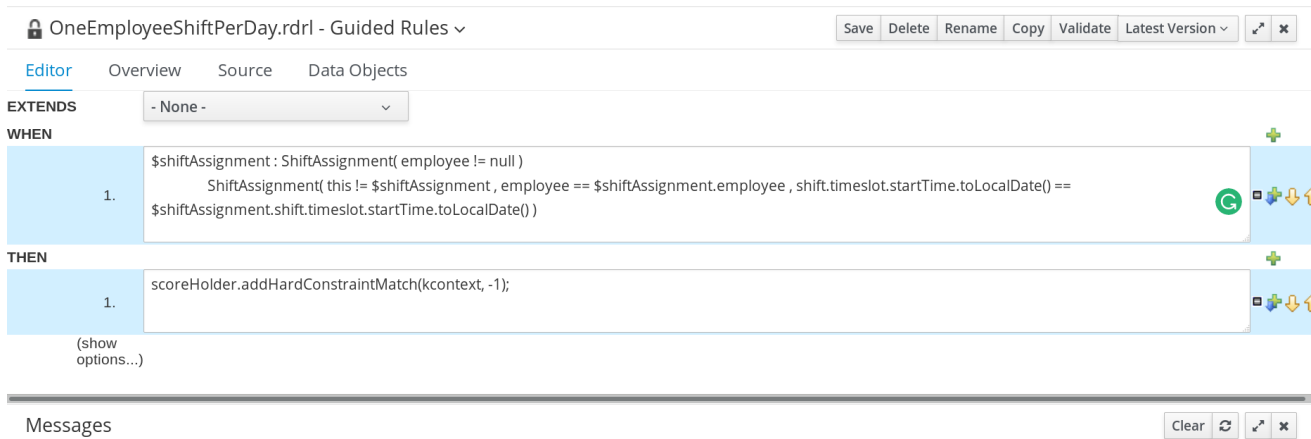
1. Go to **Menu** → **Design** → **Projects** and click the project name.
2. Click **Create New Asset** → **Guided Rule**.
3. Enter **BalanceEmployeesShiftNumber** as the **Guided Rule** name and select the **employee rostering.employee rostering Package**.
4. Click **Ok** to create the rule asset.
5. Add a **WHEN** condition by clicking the  in the **WHEN** field.
6. Select **Employee** in the **Add a condition to the rule window**. Click **+Ok**.

7. Click on the **Employee** condition to modify the constraints and add the variable name **\$employee**.
8. Add the **WHEN** condition **From Accumulate**.
  - a. Above the **From Accumulate** condition, click **click to add pattern** and select **Number** as the fact type from the drop-down list.
  - b. Add the variable name **\$shiftCount** to the **Number** condition.
  - c. Below the **From Accumulate** condition, click **click to add pattern** and select the **ShiftAssignment** fact type from the drop-down list.
  - d. Add the variable name **\$shiftAssignment** to the **ShiftAssignment** fact type.
  - e. Click on the **ShiftAssignment** condition again and from the **Add a restriction on a field** drop-down list, select **employee**.
  - f. Select **equal to** from the drop-down list next to the **employee** constraint.
  - g. Click the  icon next to the drop-down button to add a variable, and click **Bound variable** in the **Field value** window.
  - h. Select **\$employee** from the drop-down list.
  - i. In the **Function** box type **count(\$shiftAssignment)**.
9. Add the **THEN** condition by clicking the  in the **THEN** field.
10. Select **Modify Soft Score** in the **Add a new action** window. Click **+Ok**.
  - a. Type the following expression into the box: -  
`($shiftCount.intValue()*$shiftCount.intValue())`
11. Click **Validate** in the upper-right corner to check all rule conditions are valid. If the rule validation fails, address any problems described in the error message, review all components in the rule, and try again to validate the rule until the rule passes.
12. Click **Save** to save the rule.


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

## 6.3. CREATE A GUIDED RULE FOR NO MORE THAN ONE SHIFT PER DAY

The **OneEmployeeShiftPerDay** guided rule creates a hard constraint that employees are not assigned more than one shift per day. In the employee rostering example, this constraint is created using the guided rule designer.




## Procedure

1. Go to **Menu** → **Design** → **Projects** and click the project name.
2. Click **Create New Asset** → **Guided Rule**.
3. Enter **OneEmployeeShiftPerDay** as the **Guided Rule** name and select the **employee rostering.employee rostering Package**.
4. Click **Ok** to create the rule asset.
5. Add a **WHEN** condition by clicking the  in the **WHEN** field.
6. Select **Free form DRL** from the **Add a condition to the rule** window.
7. In the free form DRL box, type the following condition:

```
$shiftAssignment : ShiftAssignment( employee != null )
    ShiftAssignment( this != $shiftAssignment , employee ==
    $shiftAssignment.employee , shift.timeslot.startTime.toLocalDate()
    == $shiftAssignment.shift.timeslot.startTime.toLocalDate() )
```

This condition states that a shift cannot be assigned to an employee that already has another shift assignment on the same day.

8. Add the **THEN** condition by clicking the  in the **THEN** field.
9. Select **Add free form DRL** from the **Add a new action** window.
10. In the free form DRL box, type the following condition:
 

```
scoreHolder.addHardConstraintMatch(kcontext, -1);
```
11. Click **Validate** in the upper-right corner to check all rule conditions are valid. If the rule validation fails, address any problems described in the error message, review all components in the rule, and try again to validate the rule until the rule passes.
12. Click **Save** to save the rule.

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

## 6.4. CREATE A GUIDED RULE TO MATCH SKILLS TO SHIFT REQUIREMENTS

The **ShiftRequiredSkillsAreMet** guided rule creates a hard constraint that ensures all shifts are assigned an employee with the correct set of skills. In the employee rostering example, this constraint is created using the guided rule designer.

ShiftRequiredSkillsAreMet.rdr1 - Guided Rules ▾

Save Delete Rename Copy Validate Latest Version ▾

Editor Overview Source Data Objects

EXTENDS - None - ▾

WHEN

There is a ShiftAssignment with:

1. employee is not null

[**\$requiredSkill**] :shift.requiredSkill. Choose... --- please choose ---

[not bound]:employee.skills. Choose... excludes \$requiredSkill

THEN



1. Hard Score -1

(show options...)

Messages Clear

### Procedure

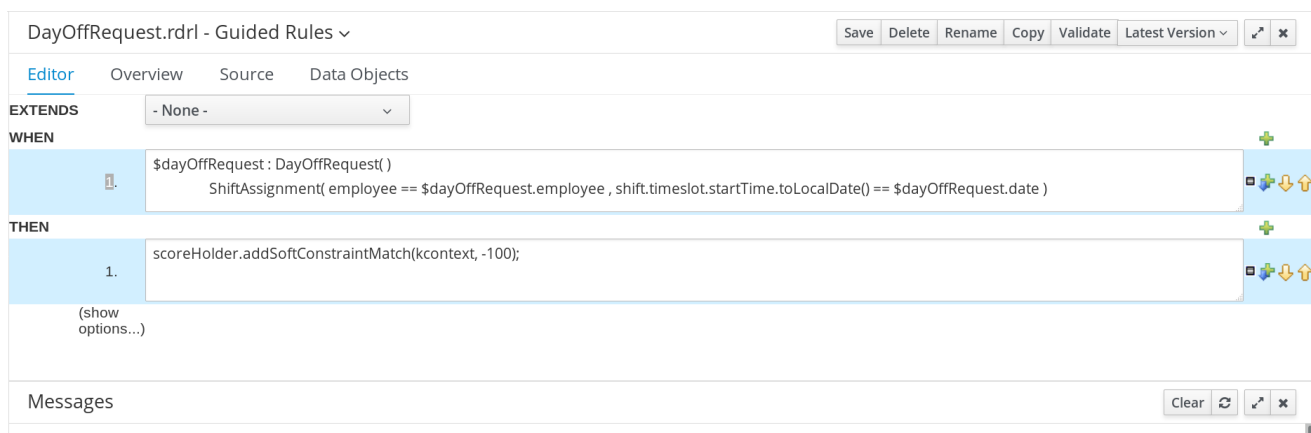
1. Go to **Menu** → **Design** → **Projects** and click the project name.
2. Click **Create New Asset** → **Guided Rule**.
3. Enter **ShiftRequiredSkillsAreMet** as the **Guided Rule** name and select the **employee rostering.employee rostering Package**.
4. Click **Ok** to create the rule asset.
5. Add a **WHEN** condition by clicking the **+** in the **WHEN** field.
6. Select **ShiftAssignment** in the **Add a condition to the rule** window. Click **+Ok**.
7. Click on the **ShiftAssignment** condition, and select **employee** from the **Add a restriction on a field** drop-down list.
8. In the designer, click the drop-down list next to **employee** and select **is not null**.
9. Click on the **ShiftAssignment** condition, and click **Expression editor**.
  - a. In the designer, click **[not bound]** to open the **Expression editor**, and bind the expression to the variable **\$requiredSkill**. Click **Set**.
  - b. In the designer, next to **\$requiredSkill**, select **shift** from the first drop-down list, then **requiredSkill** from the next drop-down list.
10. Click on the **ShiftAssignment** condition, and click **Expression editor**.
  - a. In the designer, next to **[not bound]**, select **employee** from the first drop-down list, then **skills** from the next drop-down list.
  - b. Leave the next drop-down list as **Choose**.

- c. In the next drop-down box, change **please choose** to **excludes**.
  - d. Click the  icon next to **excludes**, and in the **Field value** window, click the **New formula** button.
  - e. Type **\$requiredSkill** into the formula box.
11. Add the **THEN** condition by clicking the  in the **THEN** field.
  12. Select **Modify Hard Score** in the **Add a new action** window. Click **+Ok**.
  13. Type **-1** into the score actions box.
  14. Click **Validate** in the upper-right corner to check all rule conditions are valid. If the rule validation fails, address any problems described in the error message, review all components in the rule, and try again to validate the rule until the rule passes.
  15. Click **Save** to save the rule.

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

## 6.5. CREATE A GUIDED RULE TO MANAGE DAY OFF REQUESTS

The **DayOffRequest** guided rule creates a soft constraint that allows a shift to be reassigned to another employee in the event the employee who was originally assigned the shift is no longer able to work that day. In the employee rostering example, this constraint is created using the guided rule designer.



DayOffRequest.rdl - Guided Rules

Save Delete Rename Copy Validate Latest Version

Editor Overview Source Data Objects

EXTENDS - None -

WHEN

\$dayOffRequest : DayOffRequest( )  
ShiftAssignment( employee == \$dayOffRequest.employee , shift.timeslot.startTime.toLocalDate() == \$dayOffRequest.date )


THEN

1. scoreHolder.addSoftConstraintMatch(kcontext, -100);

(show options...)

Messages


### Procedure

1. Go to **Menu** → **Design** → **Projects** and click the project name.
2. Click **Create New Asset** → **Guided Rule**.
3. Enter **DayOffRequest** as the **Guided Rule** name and select the **employee rostering.employee rostering Package**.
4. Click **Ok** to create the rule asset.
5. Add a **WHEN** condition by clicking the  in the **WHEN** field.
6. Select **Free form DRL** from the **Add a condition to the rule** window.

7. In the free form DRL box, type the following condition:

```
$dayOffRequest : DayOffRequest( )  
    ShiftAssignment( employee == $dayOffRequest.employee ,  
    shift.timeslot.startTime.toLocalDate() == $dayOffRequest.date )
```

This condition states if a shift is assigned to an employee who has made a day off request, the employee can be unassigned the shift on that day.

8. Add the **THEN** condition by clicking the  in the **THEN** field.
9. Select **Add free form DRL** from the **Add a new action** window.
10. In the free form DRL box, type the following condition:

```
scoreHolder.addSoftConstraintMatch(kcontext, -100);
```
11. Click **Validate** in the upper-right corner to check all rule conditions are valid. If the rule validation fails, address any problems described in the error message, review all components in the rule, and try again to validate the rule until the rule passes.
12. Click **Save** to save the rule.

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

## CHAPTER 7. CONFIGURE THE SOLVER FOR EMPLOYEE ROSTERING

You can create and edit Solver configurations in Decision Central. The Solver configuration designer creates a solver configuration that can be run after the project is deployed.

### Prerequisite

Red Hat Decision Manager has been downloaded and installed. You have created and configured all of the relevant assets for the employee rostering example.

### 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 the name **EmployeeRosteringSolverConfig** for your Solver and click **Ok**.  
This opens the **Solver configuration** designer.
4. In the **Score Director Factory** configuration section, define a knowledge base that contains scoring rule definitions. The employee rostering sample project uses **defaultKieBase**.
  - a. Select one of the knowledge sessions defined within the knowledge base. The employee rostering sample project uses **defaultKieSession**.
5. Click **Validate** in the upper-right corner to check the **Score Director Factory** configuration is correct. If validation fails, address any problems described in the error message, and try again to validate until the configuration passes.
6. Click **Save** to save the Solver configuration.

For more information about configuring a Solver, see [Installing and configuring Red Hat Business Optimizer](#).

### 7.1. CONFIGURE SOLVER TERMINATION FOR THE EMPLOYEE ROSTERING PROJECT

You can configure the Solver to terminate after a specified amount of time. By default, the planning engine is given an unlimited time period to solve a problem instance.

The employee rostering sample project is set up to run for 30 seconds.

### Prerequisite

You have created all relevant assets for the employee rostering project and created the **EmployeeRosteringSolverConfig** solver configuration in Decision Central as described in [Chapter 7, Configure the Solver for employee rostering](#).

### Procedure

1. Open the **EmployeeRosteringSolverConfig** from the **Assets** perspective. This will open the **Solver configuration** designer.

2. In the **Termination** section, click **Add** to create new termination element within the selected logical group.
3. Select the **Time spent** termination type from the drop-down list. This is added as an input field in the termination configuration.
4. Use the arrows next to the time elements to adjust the amount of time spent to 30 seconds.
5. Click **Validate** in the upper-right corner to check the **Score Director Factory** configuration is correct. If validation fails, address any problems described in the error message, and try again to validate until the configuration passes.
6. Click **Save** to save the Solver configuration.

For more information about configuring a Solver or Solver termination, see [Red Hat Decision Manager Red Hat Business Optimizer Guide](#).

## 7.2. REGISTER THE SOLVER USING THE REST API

Once the solver configuration is created and the project is deployed to the decision server, you can create a Solver and submit planning problems. This can be done by accessing the project KIE container's capabilities through the decision server's remote APIs. You can create a Solver from the solver configuration file, then submit a planning problem to it and request the best solution at any time.

Each Solver is capable of optimizing one planning problem at a time.

### Prerequisites

- You must have the **kie-server** role in order for the **admin** credentials mentioned in this examples to work. For more information about the **kie-server** role, see [Installing and configuring Red Hat Business Optimizer](#).
- The employee rostering project has been set up and deployed according to the previous chapters in this document.
- The Solver has been configured as instructed in [Chapter 7, Configure the Solver for employee rostering](#) and [Section 7.1, "Configure Solver termination for the employee rostering project"](#).
- *You have successfully built and deployed the employee rostering project.* For more information about deploying the employee rostering sample project in the workbench, see [Chapter 1, Deploy the employee rostering sample project in Decision Central](#).

### Procedure

1. Create a HTTP request using the following header:

```
authorization: admin:admin
X-KIE-ContentType: xstream
content-type: application/xml
```

2. Register the Solver using the following request:

**PUT**



```
http://localhost:8080/kie-
server/services/rest/server/containers/employee rostering_1.0.0-
SNAPSHOT/solvers/EmployeeRosteringSolver
```

#### Request body

```
<solver-instance>
  <solver-config-
file>employee rostering/employee rostering/EmployeeRosteringSolverCo
nfig.solver.xml</solver-config-file>
</solver-instance>
```

3. Submit a request to the Solver:

#### POST

```
http://localhost:8080/kie-
server/services/rest/server/containers/employee rostering_1.0.0-
SNAPSHOT/solvers/EmployeeRosteringSolver/state/solving
```

#### Request body

```
<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 rosteri
```

```

ng.Employee/skills/employee rostering.employee rostering.Skill"/>
  </employee rostering.employee rostering.Shift>
  <employee rostering.employee rostering.Shift>
    <timeslot
reference="../../employee rostering.employee rostering.Shift/timeslo
t"/>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rosteri
ng.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/timeslo
t"/>
    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rosteri
ng.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.Shi
ft/timeslot"/>
</timeslotList>
<dayOffRequestList/>
<shiftAssignmentList/>
</employee rostering.employee rostering.EmployeeRoster>

```

4. Request the best solution to the planning problem:

#### GET

[http://localhost:8080/kie-server/services/rest/server/containers/employee rostering\\_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/bestsolution](http://localhost:8080/kie-server/services/rest/server/containers/employee rostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/bestsolution)

#### Example response

```

<solver-instance>
  <container-id>employee-rostering</container-id>
  <solver-id>solver1</solver-id>
  <solver-config-
file>employee rostering/employee rostering/EmployeeRosteringSolverCo
nfig.solver.xml</solver-config-file>
  <status>NOT_SOLVING</status>

```

```

    <score
scoreClass="org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScore">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"/>
                </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"/>
                    </timeslot>
                    <requiredSkill
reference="../../../../employeeList/employee rostering.employee rostering.Employee[2]/skills/employee rostering.employee rostering.Skill"/>
                    </employee rostering.employee rostering.Shift>
            </employee rostering.employee rostering.Shift>
    </shiftList>
</best-solution>
</employee rostering.employee rostering.EmployeeRoster>

```

```

        </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.Shi
ft/timeslot"/>
    </timeslotList>
    <dayOffRequestList/>
    <shiftAssignmentList/>
    <score>0hard/0soft</score>
</best-solution>
</solver-instance>

```

For more information about creating containers, Solvers, and submitting problems through the decision server REST API, see [Installing and configuring Red Hat Business Optimizer](#).

## APPENDIX A. VERSIONING INFORMATION

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