



## **Red Hat Decision Manager 7.0**

**Decision Central で Red Hat Business  
Optimizer の従業員の勤務表問題を作成し解決す  
る方法**



# Red Hat Decision Manager 7.0 Decision Central で Red Hat Business Optimizer の従業員勤務表問題を作成し解決する方法

---

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

## 法律上の通知

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.

## 概要

本書は、Red Hat Decision Manager 7.0 の Decision Central で、Red Hat Business Optimizer の従業員の勤務表サンプルを作成して実行する方法を説明します。

---

目次

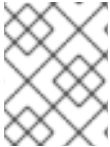
前書き .....	3
第1章 DECISION CENTRAL への従業員勤務表サンプルプロジェクトのデプロイメント .....	4
第2章 従業員の勤務表サンプルプロジェクトの設定 .....	5
第3章 プロジェクトファクトおよびプランニングエンティティ .....	6
第4章 従業員の勤務表プロジェクトへのデータモデルの作成 .....	7
4.1. 従業員の勤務表プランニングエンティティの作成 .....	8
4.2. 従業員の勤務表プランニングソリューションの作成 .....	9
第5章 従業員勤務表の制約 .....	11
5.1. DRL ルール .....	11
5.2. DRL デザイナーを使用した従業員の勤務表の制約定義 .....	11
第6章 ガイド付きルールを使用して従業員の勤務表にルールの作成 .....	13
6.1. ガイド付きルール .....	13
6.2. 従業員のシフト数のバランスを取るガイド付きルールの作成 .....	13
6.3. 同じ日に複数のシフトを割り当てないようにするガイド付きルールの作成 .....	14
6.4. シフト要件にスキルを一致させるガイド付きルールの作成 .....	16
6.5. 休暇申請を管理するガイド付きルールの作成 .....	17
第7章 従業員の勤務表への SOLVER の設定 .....	19
7.1. 従業員の勤務表プロジェクトに対する SOLVER の終了設定 .....	19
7.2. REST API を使用した SOLVER の登録 .....	20
付録A バージョン情報 .....	25



## 前書き

ビジネスルールの作成者は、Red Hat Decision Manager ディストリビューションに同梱されている、事前設定済みの **employee-rostering** サンプルプロジェクトを使用して、Red Hat Business Optimizer の機能をテストおよび操作できます。

**employee-rostering** サンプルプロジェクトは、Decision Central にビルドしてデプロイできます。このプロジェクトは、シフト勤務の計画問題を解決するのに必要な Decision Central の各アセットを作成し、Red Hat Business Optimizer を使用して実現可能な最適解を見つける方法を示します。本書を参考に、事前設定した **employee-rostering** プロジェクトを Decision Central にデプロイするか、Decision Central を使用してプロジェクトを作成します。



### 注記

Decision Central の **employee-rostering** サンプルプロジェクトには、データセットが含まれていません。

### 前提条件

- Red Hat JBoss Enterprise Application Platform 7.1.0 がインストールされている。『[Red Hat JBoss EAP 7.1.0 インストールガイド](#)』を参照してください。
- Red Hat Decision Manager がインストールされている。詳細は『[Red Hat Decision Manager のオンプレミスインストール](#)』を参照してください。
- Red Hat Decision Manager が実行していて、**admin** ロールで Decision Central にログインできる。詳細は『[Red Hat Decision Manager のオンプレミスインストール](#)』を参照してください。
- Red Hat Business Optimizer に Red Hat Decision Manager が設定されている。必要な設定については『[Red Hat Business Optimizer のインストールおよび設定](#)』を参照してください。
- このサンプルを修正してデータを最適化する場合は、修正したプロジェクトを Decision Central からサーバーにデプロイし、Decision Server で最適なタスクを実行するためのデータセットが必要です。

## 第1章 DECISION CENTRAL への従業員勤務表サンプルプロジェクトのデプロイメント

Decision Central には、製品と機能に慣れるために使用できるサンプルプロジェクトが多数あります。従業員の勤務表サンプルプロジェクトは、Red Hat Business Optimizer でシフト勤務のユースケースを示すために計画され作成されました。以下の手順に従って、従業員の勤務表サンプルを Decision Central にデプロイして実行します。

### 前提条件

- Red Hat Decision Manager がダウンロードされ、インストールされている。
- デシジョンサーバーを起動し、**admin** 権限を持つユーザーで Decision Central にログインしている。
- インストールの詳細は『[Red Hat Decision Manager のオンプレミスインストール](#)』を参照してください。
- 使用方法の詳細は『[デシジョンサービスの使用ガイド](#)』を参照してください。

### 手順

1. Decision Central で **Menu** → **Design** → **Projects** の順にクリックします。
2. 事前に設定した **myteam** スペースで **Try Samples** をクリックします。
3. サンプルプロジェクトの一覧から **employee-rostering** を選択し、右上の **OK** をクリックして、プロジェクトをインポートします。
4. アセットリストをコンパイルし、**Build & Deploy** をクリックして、従業員の勤務表サンプルをデプロイします。

本書は、各プロジェクトアセットと、その設定を説明します。



## 第2章 従業員の勤務表サンプルプロジェクトの設定

従業員の勤務表サンプルプロジェクトは、Decision Central で使用できる事前設定のプロジェクトです。このプロジェクトをデプロイする方法は「[1章 Decision Central への従業員勤務表サンプルプロジェクトのデプロイメント](#)」を参照してください。

本章は、従業員の勤務表サンプルを設定する方法を説明します。このワークフローに従って、Decision Central に同じようなプロジェクトを作成できます。

### 前提条件

- Red Hat Decision Manager がダウンロードされ、インストールされている。
- Decision Central をデプロイし、**admin** ロールを持つユーザーでログインしている。

### 手順

1. **Menu** → **Design** → **Projects** → **Add Project** をクリックして、Decision Central に新しいプロジェクトを作成します。
2. **Add Project** ウィンドウで、以下のフィールドに入力します。

- **Name:** `employee-rostering`
- **Description**(任意): Business Optimizer を使用した従業員の勤務表問題の最適化。スキルに基づいて、従業員をシフトに割り当てます。

任意で、**Show Advanced Options** をクリックして、**Group ID**、**Artifact ID**、および **Version** に情報を追加します。

- **Group ID:** `employeeerostering`
  - **Artifact ID:** `employeeerostering`
  - **Version:** `1.0.0-SNAPSHOT`
3. **Add** をクリックして、Decision Central プロジェクトリポジトリにプロジェクトを追加します。

## 第3章 プロジェクトファクトおよびプランニングエンティティ

従業員勤務表の計画問題の各ドメインクラスは、以下のいずれかに分類できます。

- 関連性のないクラス: どのスコア制約にも使用されません。計画に関して言えば、このデータは使用されません。
- **問題ファクト** クラス: スコア制約に使用されますが、(問題が変わらない限り) 計画時には変化しません (例: **Shift**、**Employee** など)。問題ファクトクラスのプロパティはすべて問題のプロパティです。
- **プランニングエンティティ** クラス: スコア制約に使用され、計画時に変化します (例: **ShiftAssignment**)。計画時に変更するプロパティは **プランニング変数** です。その他のプロパティは問題プロパティです。  
以下の点についてお考え下さい。
- どのクラスを、計画時に変更しますか?
- どのクラスに、**Solver** で変更する変数がありますか?  
そのクラスが、**プランニングエンティティ** です。

プランニングエンティティークラスは、**@PlanningEntity** アノテーションでアノテートする必要があります。または、ドメインデザイナーで Red Hat Business Optimizer ドックを使用して Decision Central に定義する必要があります。

各プランニングエンティティークラスには、1 つ以上の **プランニング変数** があり、1 つ以上の定義プロパティが必要です。

多くのユースケースには、プランニングエンティティークラスが 1 つだけあり、1 つのプランニングエンティティークラスに対してプランニング変数が 1 つだけ含まれます。

## 第4章 従業員の勤務表プロジェクトへのデータモデルの作成

このセクションでは、Decision Central で従業員の勤務表サンプルプロジェクトを実行するのに必要なデータオブジェクトを作成します。

### 前提条件

「[2章 従業員の勤務表サンプルプロジェクトの設定](#)」に従ってプロジェクトを設定している。

### 手順

1. 新規プロジェクトで、プロジェクトパースペクティブの **Data Object** をクリックするか、**Create New Asset** → **Data Object** をクリックして、新しいデータオブジェクトを作成します。
2. 最初のデータオブジェクトの名前を **Timeslot** とし、**パッケージ** で **employeeerostring.employeeerostring** を選択します。  
**OK** をクリックします。
3. **Data Objects** パースペクティブで **+add field** をクリックして、**Timeslot** データオブジェクトにフィールドを追加します。
4. **id** フィールドで **endTime** と入力します。
5. **Type** の横にあるドロップダウンメニューをクリックし、**LocalDateTime** を選択します。
6. **Create and continue** を別のフィールドに追加します。
7. **id startTime** および **Type LocalDateTime** を使用して、フィールドを追加します。
8. **Create** をクリックします。
9. 右上の **Save** をクリックして、**Timeslot** データオブジェクトを保存します。
10. 右上の **x** をクリックして、**Data Objects** パースペクティブを閉じ、**Assets** メニューに戻ります。
11. 前述の手順で、以下のデータオブジェクトとその属性を作成します。

表4.1 Skill

id	タイプ
name	String

表4.2 Employee

id	タイプ
name	String
skills	employeeerostring.employeeerostring.Skill[List]

表4.3 Shift

id	タイプ
requiredSkill	employeeerostring.employeeerostring.Skill
timeslot	employeeerostring.employeeerostring.Timeslot

表4.4 DayOffRequest

id	タイプ
date	LocalDate
employee	employeeerostring.employeeerostring.Employee

表4.5 ShiftAssignment

id	タイプ
employee	employeeerostring.employeeerostring.Employee
shift	employeeerostring.employeeerostring.Shift

データオブジェクトの詳細は『[デシジョンサービスの使用ガイド](#)』を参照してください。

## 4.1. 従業員の勤務表プランニングエンティティの作成

従業員勤務表の計画問題を解決するには、プランニングエンティティと Solver を作成する必要があります。プランニングエンティティは、Red Hat Business Optimizer ドックで利用可能な属性を使用して、ドメインデザイナーに定義します。

以下の手順に従って、従業員の勤務表サンプルに、**ShiftAssignment** データオブジェクトをプランニングエンティティとして定義します。

### 前提条件

- 従業員の勤務表サンプルを実行するには、「[4章 従業員の勤務表プロジェクトへのデータモデルの作成](#)」の手順に従って、関連するデータオブジェクトとプランニングエンティティを作成する必要があります。

### 手順

- プロジェクトの **Assets** メニューから、**ShiftAssignment** データオブジェクトを開きます。



2. **Data Objects** パースペクティブで、右側の  をクリックして Red Hat Business Optimizer ドックを開きます。
3. **Planning Entity** を選択します。
4. **ShiftAssignment** データオブジェクトのフィールドリストで **employee** を選択します。
5. Red Hat Business Optimizer ドックで **Planning Variable** を選択します。  
**Value Range Id** 入力フィールドに **employeeRange** を入力します。これにより、**@ValueRangeProvider** アノテーションがプランニングエンティティに追加され、デザイナーの **Source** タブをクリックすると表示されます。

プランニング変数の値の範囲は **@ValueRangeProvider** アノテーションで定義されます。**@ValueRangeProvider** アノテーションには **id** プロパティが常にあり、**@PlanningVariable** の **valueRangeProviderRefs** プロパティから参照されます。

6. ドックを閉じ、**Save** をクリックして、データオブジェクトを保存します。

## 4.2. 従業員の勤務表プランニングソリューションの作成

従業員勤務表の問題は、定義したプランニングソリューションに依存します。プランニングソリューションは、Red Hat Business Optimizer ドックで利用可能な属性を使用してドメインデザイナーで定義されます。

### 前提条件

- 「[4章 従業員の勤務表プロジェクトへのデータモデルの作成](#)」および「[従業員の勤務表プランニングエンティティの作成](#)」の手順に従って、従業員の勤務表サンプルを実行するのに必要なデータオブジェクトおよびプランニングエンティティを作成している。


### 手順

1. 識別子 **EmployeeRoster** でデータオブジェクトを新規作成します。
2. 以下のフィールドを作成します。

表4.6 EmployeeRoster

id	タイプ
<b>dayOffRequestList</b>	<b>employee rostering.employee rostering.DayOffRequest[<a href="#">List</a>]</b>
<b>shiftAssignmentList</b>	<b>employee rostering.employee rostering.ShiftAssignment[<a href="#">List</a>]</b>
<b>shiftList</b>	<b>employee rostering.employee rostering.Shift[<a href="#">List</a>]</b>
<b>skillList</b>	<b>employee rostering.employee rostering.Skill[<a href="#">List</a>]</b>

id	タイプ
<code>timeslotList</code>	<code>employee rostering.employee rostering.Timeslot[List]</code>

3. **Data Objects** パースペクティブで、右側の  をクリックして Red Hat Business Optimizer ドックを開きます。

4. **プランニングソリューション** を選択します。

5. **Solution Score Type** は、デフォルトの **Hard soft score** のままにします。これにより、タイプが ソリューションスコアとなる **EmployeeRoster** データオブジェクトに、**score** フィールドが自動的に生成されます。

6. 次の属性で新しいフィールドを追加します。

id	タイプ
<code>employeeList</code>	<code>employee rostering.employee rostering.Employee[List]</code>

7. **employeeList** フィールドを選択した状態で、Red Hat Business Optimizer ドックを開いて、**Planning Value Range Provider** ボックスを選択します。  
**id** フィールドに **employeeRange** を入力し、ドックを閉じます。

8. 右上で **Save** をクリックし、アセットを保存します。

## 第5章 従業員勤務表の制約

従業員の勤務表はプランニングソリューションです。すべての計画問題には、最適解を得るのに必要な制約が含まれます。

Decision Central の従業員の勤務表サンプルプロジェクトには、以下のハード制約およびソフト制約が含まれます。

### ハード制約

- 従業員に割り当てられるシフトの数は、1 日 1 つまで。
- 特別な従業員スキルが必要なすべてのシフトは、そのスキルを持つ従業員に割り当てられる。

### ソフト制約

- すべての従業員がシフトに割り当てられている。
- 従業員が休暇を取った場合は、シフトを別の従業員に再割り当て可能。

ハード制約およびソフト制約は、フリーフォーム DRL デザイナー、またはガイド付きルールを使用して Decision Central で定義できます。

ハード制約およびソフト制約の詳細は「[Red Hat Business Optimizer のインストールおよび設定](#)」を参照してください。

## 5.1. DRL ルール

DRL ルールは、`.drl` テキストファイルに直接定義するビジネスルールです。このような DRL ファイルは、Decision Central の他のすべてのルールアセットが最終的にレンダリングされるソースとなります。DRL ファイルを Decision Central インターフェースで作成および管理したり、外部の Red Hat Developer Studio、Java オブジェクト、Maven アーキタイプを使用して作成したりできます。DRL ファイルには、最低限、ルールの条件 (**when**) およびアクション (**then**) を定義するルールを 1 つ以上追加できます。Decision Central の DRL デザイナーでは、Java、DRL、および XML の構文が強調表示されます。

DRL ルールに関連するデータオブジェクトはすべて、DRL ルールと同じ Decision Central プロジェクトパッケージに置く必要があります。同じパッケージのアセットはデフォルトでインポートされます。その他のパッケージの既存アセットは、DRL ルールを使用してインポートできます。

## 5.2. DRL デザイナーを使用した従業員の勤務表の制約定義

Decision Central でフリーフォーム DRL デザイナーを使用して、従業員の勤務表サンプルに制約の定義を作成できます。

この手順を使用して、シフトが終わってから 10 時間以上経たないと従業員をシステムに割り当てられない **ハード制約** を作成します。

### 手順

1. **Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. **Create New Asset** → **DRL file** をクリックします。

3. **DRL file** 名前フィールドに、**ComplexScoreRules** と入力します。
4. **employeeerostering.employeeerostering** パッケージを選択します。
5. **+Ok** をクリックして DRL ファイルを作成します。
6. DRL デザイナーの **Editor** タブで、**Employee10HourShiftSpace** ルールを DRL ファイルとして定義します。

```
package employeeerostering.employeeerostering;

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. **Save** をクリックして、DRL ファイルを保存します。

DRL ファイルの作成方法は「[DRL ルールを使用したデシジョンサービスの作成](#)」を参照してください。



## 第6章 ガイド付きルールを使用して従業員の勤務表にルールの作成

Decision Central でガイド付きルールデザイナーを使用して、従業員の勤務表にハード制約およびソフト制約を定義するルールを作成できます。

### 6.1. ガイド付きルール

ガイド付きルールは、ルール作成のプロセスを提供する、Decision Central の UI ベースのガイド付きルールデザイナーで作成するビジネスルールです。ガイド付きルールデザイナーを使用すると、ルールを定義するデータオブジェクトに基づいて、可能なインプットにフィールドおよびオプションを提供します。定義したガイド付きルールは、その他のすべてのルールアセットとともに Drools Rule Language (DRL) ルールにコンパイルされます。

ガイド付きルールに関連するすべてのデータオブジェクトは、ガイド付きルールと同じプロジェクトパッケージに置く必要があります。同じパッケージに含まれるアセットはデフォルトでインポートされます。必要なデータオブジェクトとガイド付きルールを作成したら、ガイド付きルールデザイナーの **Data Objects** タブから、必要なデータオブジェクトがすべてリストされていることを検証したり、**新規アイテム** を追加してその他の既存データオブジェクトをインポートしたりできます。

### 6.2. 従業員のシフト数のバランスを取るガイド付きルールの作成

ガイド付きルール **BalanceEmployeesShiftNumber** は、可能な限りバランスを取るよう従業員にシフトを割り当てるソフト制約を作成します。これは、シフトの分配が平等でなくなると増えるスコアペナルティを作成することで行います。ルールによって実装されたスコア式により、Solver がよりバランスの取れるようにシフトを分散させます。

BalanceEmployeesShiftNumber.rdr1 - Guided Rules

Save Delete Rename Copy Validate Latest Version

Editor Overview Source Data Objects

EXTENDS - None -

WHEN

1. There is an Employee [Employee]  
There is a Number [ShiftCount]  
From Accumulate  
All ShiftAssignment [ShiftAssignment] with:
2. employee equal to \$employee  
Custom Code Function  
Function: count(\$ShiftAssignment)



THEN

1. Soft Score - (\$ShiftCount.intValue() \* \$ShiftCount.intValue())  
(show options...)

Messages Clear

#### 手順

1. **Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. **Create New Asset** → **Guided Rule** をクリックします。
3. **Guided Rule** 名に **BalanceEmployeesShiftNumber** を入力し、**Package** で **employee rostering.employee rostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで **+** をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule** ウィンドウで **Employee** を選択し、**+Ok** をクリックします。

7. **Employee** 条件でクリックして制約を修正し、変数名 **\$employee** を追加します。
8. **WHEN** 条件 **From Accumulate** を追加します。
  - a. **From Accumulate** 条件の上で **click to add pattern** をクリックし、ドロップダウンリストでファクトタイプ **Number** を選択します。
  - b. 変数名 **\$shiftCount** を **Number** 条件に追加します。
  - c. **From Accumulate** 条件の下で **click to add pattern** をクリックして、ドロップダウンリストで **ShiftAssignment** ファクトタイプを選択します。
  - d. 変数名 **\$shiftAssignment** を **ShiftAssignment** ファクトタイプに追加します。
  - e. **ShiftAssignment** 条件を再度クリックし、**Add a restriction on a field** ドロップダウンリストで **employee** を選択します。
  - f. **employee** 制約の横にあるドロップダウンリストで **equal to** を選択します。
  - g. ドロップダウンボタンの横の  アイコンをクリックして変数を追加し、**Field value** ウィンドウで **Bound variable** をクリックします。
  - h. ドロップダウンリストで **\$employee** を選択します。
  - i. **Function** ボックスに **count(\$shiftAssignment)** と入力します。
9. **THEN** フィールドで  をクリックして、**THEN** 条件を追加します。
10. **Add a new action** ウィンドウで **Modify Soft Score** を選択し、**+Ok** をクリックします。
  - a. ボックスに **-( $\$shiftCount.intValue() * \$shiftCount.intValue()$ )** と入力します。
11. 右上の **Validate** をクリックし、ルール条件がすべて有効であることを確認します。ルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
12. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は『[ガイド付きルールを使用したデシジョンサービスの作成](#)』を参照してください。

### 6.3. 同じ日に複数のシフトを割り当てないようにするガイド付きルールの作成

ガイド付きルール **OneEmployeeShiftPerDay** は、同じ日の複数のシフトに従業員を割り当てないようにするハード制約を作成します。従業員の勤務表サンプルでは、このガイド付きルールデザイナーを使用してこの制約が作成されます。

OneEmployeeShiftPerDay.rdl - Guided Rules

Save Delete Rename Copy Validate Latest Version

Editor Overview Source Data Objects

EXTENDS - None -

WHEN

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

THEN

1. `scoreHolder.addHardConstraintMatch(kcontext, -1);`

(show options...)

Messages

## 手順

1. **Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. **Create New Asset** → **Guided Rule** をクリックします。
3. **Guided Rule** 名に **OneEmployeeShiftPerDay** と入力し、**Package** で **employee rostering.employee rostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで **+** をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule** ウィンドウから **Free form DRL** を選択します。
7. フリーフォームの DRL ボックスに、以下の条件を入力します。

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

この条件は、同じ日に別のシフトがすでに割り当てられている従業員にはシフトを割り当てることができないことを示しています。

8. **THEN** フィールドで **+** をクリックして、**THEN** 条件を追加します。
  9. **Add a new action** ウィンドウから **Add free form DRL** を選択します。
  10. フリーフォームの DRL ボックスに、以下の条件を入力します。
- ```
scoreHolder.addHardConstraintMatch(kcontext, -1);
```
11. 右上の **Validate** をクリックし、ルール条件がすべて有効であることを確認します。ルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
  12. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は『[ガイド付きルールを使用したデシジョンサービスの作成](#)』を参照してください。

## 6.4. シフト要件にスキルを一致させるガイド付きルールを作成

ガイド付きルール **ShiftRequiredSkillsAreMet** は、すべてのシフトが、適切なスキルセットを持つ従業員に割り当てられるのを確認するハード制約を作成します。従業員の勤務表サンプルでは、ガイド付きルールデザイナーを使用してこの制約を作成します。

ShiftRequiredSkillsAreMet.rdl - Guided Rules

Save Delete Rename Copy Validate Latest Version

Editor Overview Source Data Objects

EXTENDS - None -

WHEN

1. There is a ShiftAssignment with:  
employee is not null

2. [\$requiredSkill]:shift.requiredSkill. --- please choose ---

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

THEN

1. Hard Score -1



(show options...)

Messages

Clear

### 手順

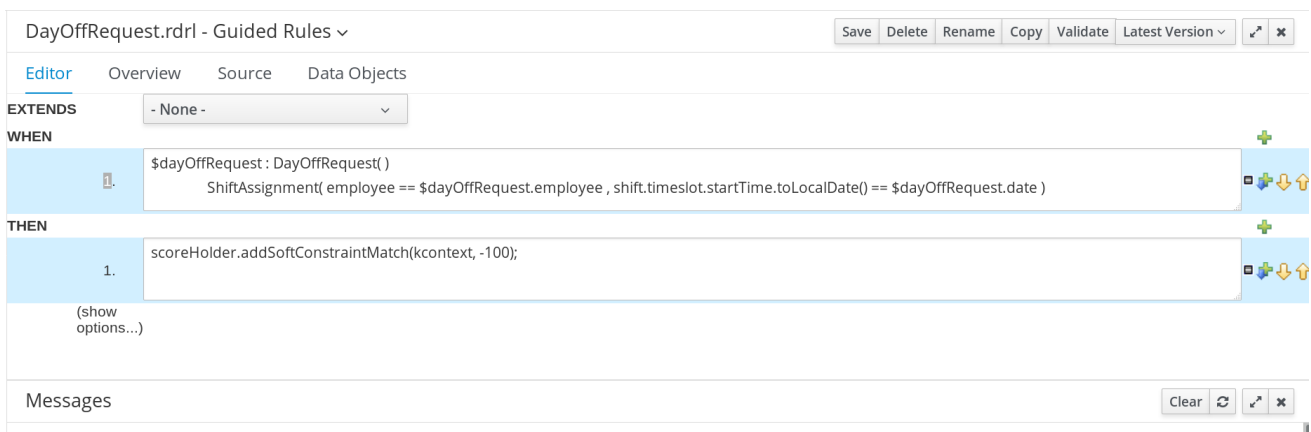
1. **Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. **Create New Asset** → **Guided Rule** をクリックします。
3. **Guided Rule** 名に **ShiftRequiredSkillsAreMet** と入力し、**Package** で **employee rostering.employee rostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで **+** をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule** ウィンドウで **ShiftAssignment** を選択します。**+Ok** をクリックします。
7. **ShiftAssignment** 条件をクリックし、**Add a restriction on a field** ドロップダウンリストで **employee** を選択します。
8. デザイナーで、**employee** の横のドロップダウンリストをクリックし、**is not null** を選択します。
9. **ShiftAssignment** 条件をクリックし、**Expression editor** をクリックします。
  - a. デザイナーで、**[not bound]** をクリックし、**Expression editor** を開き、式と変数 **\$requiredSkill** をバインドします。**Set** をクリックします。
  - b. デザイナーの **\$requiredSkill** の横にあるドロップダウンリストで **shift** を選択し、その隣のドロップダウンリストで **requiredSkill** を選択します。
10. **ShiftAssignment** 条件をクリックし、**Expression editor** をクリックします。
  - a. デザイナーで、**[not bound]** の横にあるドロップダウンリストで **employee** を選択し、その隣のドロップダウンリストで **skills** を選択します。
  - b. その隣のドロップダウンリストでは **Choose** を選択したままにします。

- c. その隣のドロップダウンボックスで、**please choose** を **excludes** に変更します。
  - d. **excludes** の隣の  アイコンをクリックし、**Field value** ウィンドウで **New formula** ボタンをクリックします。
  - e. 式ボックスに **\$requiredSkill** を追加します。
11. **THEN** フィールドで  をクリックして、**THEN** 条件を追加します。
  12. **Add a new action** ウィンドウで **Modify Hard Score** を選択し、**+Ok** をクリックします。
  13. スコアアクションボックスに **-1** を入力します。
  14. 右上の **Validate** をクリックし、ルール条件がすべて有効であることを確認します。ルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
  15. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は『[ガイド付きルールを使用したデシジョンサービスの作成](#)』を参照してください。

## 6.5. 休暇申請を管理するガイド付きルールの作成

ガイド付きルール **DayOffRequest** は、そのシフトに元々割り当てられていた従業員がその日に就業できなくなった場合に、別の従業員にシフトを再割り当てできるようにするソフト制約を作成します。従業員の勤務表サンプルでは、この制約はガイド付きルールデザイナーを使用して作成されます。



DayOffRequest.rdlr - 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

Clear


### 手順

1. **Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. **Create New Asset** → **Guided Rule** をクリックします。
3. **Guided Rule** 名に **DayOffRequest** と入力し、**Package** で **employee rostering.employee rostering** を選択します。
4. **OK** をクリックして、ルールアセットを作成します。
5. **WHEN** フィールドで  をクリックして、**WHEN** 条件を追加します。
6. **Add a condition to the rule** ウィンドウから **Free form DRL** を選択します。

7. フリーフォームの DRL ボックスに、以下の条件を入力します。

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

この条件は、休暇申請を行った従業員にシフトを割り当てている場合に、その従業員をその日のシフト割り当てから削除できることを示しています。

8. **THEN** フィールドで  をクリックして、**THEN** 条件を追加します。
9. **Add a new action** ウィンドウから **Add free form DRL** を選択します。
10. フリーフォームの DRL ボックスに、以下の条件を入力します。

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

11. 右上の **Validate** をクリックし、ルール条件がすべて有効であることを確認します。ルールの妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、ルールの全コンポーネントを見直し、エラーが表示されなくなるまでルールの妥当性確認を行います。
12. **Save** をクリックして、ルールを保存します。

ガイド付きルールの作成方法は『[ガイド付きルールを使用したデシジョンサービスの作成](#)』を参照してください。

## 第7章 従業員の勤務表への SOLVER の設定

Decision Central に Solver 設定を作成して編集することができます。Solver 設定デザイナーは、プロジェクトがデプロイされた後に実行できる Solver 設定を作成します。

### 前提条件

Red Hat Decision Manager がダウンロードしてインストールされており、従業員の勤務表サンプルに関連するアセットをすべて作成して設定している。

### 手順

1. Decision Central で、**Menu** → **Projects** をクリックし、プロジェクトをクリックして開きます。
2. **Assets** パースペクティブで、**Create New Asset** → **Solver configuration** をクリックします。
3. **Create new Solver configuration** ウィンドウで、Solver の名前 **EmployeeRosteringSolverConfig** と入力し、**Ok** をクリックします。  
これにより、**Solver configuration** デザイナーが開きます。
4. **Score Director Factory** 設定セクションで、スコアリングルール定義を含むナレッジベースを定義します。従業員の勤務表サンプルプロジェクトは **defaultKieBase** を使用します。
  - a. ナレッジベースに定義したナレッジセクションの中から 1 つ選択します。従業員の勤務表サンプルプロジェクトは **defaultKieSession** を使用します。
5. 右上の **Validate** をクリックし、**Score Director Factory** 設定が正しいことを確認します。妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、エラーが表示されなくなるまで妥当性確認を行います。
6. **Save** をクリックして、Solver 設定を保存します。

Solver の設定方法は『[Red Hat Business Optimizer のインストールおよび設定](#)』を参照してください。

### 7.1. 従業員の勤務表プロジェクトに対する SOLVER の終了設定

一定期間が過ぎたら Solver が終了できるように設定できます。デフォルトでは、プランニングエンジンには、時間制限なく問題を解決できるように指定されています。

従業員の勤務表サンプルプロジェクトは、30 秒間実行するように設定されています。

### 前提条件

従業員の勤務表プロジェクトに、関連するすべてのアセットを作成し、「[7章 従業員の勤務表への Solver の設定](#)」の手順に従って、Decision Central に Solver 設定 **EmployeeRosteringSolverConfig** を作成している。

### 手順

1. **Assets** パースペクティブで **EmployeeRosteringSolverConfig** を開きます。これにより、**Solver 設定** デザイナーが開きます。
2. **Termination** セクションで **Add** をクリックして、選択した論理グループに新しい終了要素を作成します。



3. ドロップダウンリストから、終了タイプ **Time spent** を選択します。これは、終了条件の入力フィールドとして追加されます。
4. 時間要素の横の矢印を使用して、経過時間を 30 秒に設定します。
5. 右上の **Validate** をクリックし、**Score Director Factory** 設定が正しいことを確認します。妥当性確認に失敗したら、エラーメッセージに記載された問題に対応し、エラーが表示されなくなるまで妥当性確認を行います。
6. **Save** をクリックして、Solver 設定を保存します。

Solver または Solver の終了を設定する方法は『[Red Hat Decision Manager Red Hat Business Optimizer Guide](#)』を参照してください。

## 7.2. REST API を使用した SOLVER の登録

Solver 設定を作成し、プロジェクトをデシジョンサーバーにデプロイしたら、Solver を作成してプランニング問題を送信できます。これは、デシジョンサーバーのリモート API からプロジェクトの KIE コンテナの機能にアクセスして行います。Solver 設定ファイルから Solver を作成し、それにプランニング問題を送信すれば、いつでも最適解をリクエストできます。

各 Solver で、一度に最適化できる計画問題の数は 1 つだけです。

### 前提条件

- このサンプルで使った **admin** 認証情報を有効にするには、**kie-server** ロールが必要。**kie-server** ロールの詳細は『[Red Hat Business Optimizer のインストールおよび設定](#)』を参照してください。
- 前章に従って、従業員の勤務表プロジェクトが設定されデプロイされている。
- 「[7章 従業員の勤務表への Solver の設定](#)」および「[従業員の勤務表プロジェクトに対する Solver の終了設定](#)」に従って Solver が設定されている。
- 従業員の勤務表プロジェクトが適切にビルドされデプロイされている。ワークベンチに従業員の勤務表サンプルプロジェクトをデプロイする方法は「[1章 Decision Central への従業員勤務表サンプルプロジェクトのデプロイメント](#)」を参照してください。

### 手順

1. 以下のヘッダーを使用して HTTP 要求を作成します。

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

2. 以下の要求を使用して Solver を登録します。

#### PUT

```
http://localhost:8080/kie-
server/services/rest/server/containers/employeerostering_1.0.0-
SNAPSHOT/solvers/EmployeeRosteringSolver
```

要求の本文



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

3. Solver に要求を送信します。

## POST

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

## 要求の本文

```
<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 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/timeslo
t"/>
    <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.Shi
ft/timeslot"/>
</timeslotList>
<dayOffRequestList/>
<shiftAssignmentList/>
</employee rostering.employee rostering.EmployeeRoster>

```

4. 計画問題に最適解をリクエストします。

## 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)

## 応答例

```

<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.HardSo
ftScore">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 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/>
  <score>0hard/0soft</score>
</best-solution>
</solver-instance>
```

デシジョンサーバーの REST API から、コンテナ、Solver を作成し、問題を送信する方法は『[Red Hat Business Optimizer のインストールおよび設定](#)』を参照してください。

## 付録A バージョン情報

本ドキュメントの最終更新日: 2018 年 7 月 3 日