



Red Hat Decision Manager 7.0

Red Hat Business Optimizer のインストールおよび設定

Red Hat Decision Manager 7.0 Red Hat Business Optimizer のインストールおよび設定

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

本書では、Red Hat Decision Manager 7.0 に Red Hat Business Optimizer をインストールして設定する方法について説明します。

目次

前書き	4
パート I. BUSINESS OPTIMIZER を使用したプランニングの問題の解決	5
第1章 RED HAT BUSINESS OPTIMIZER とは	6
1.1. プランニングの問題は NP 完全または NP 困難	6
第2章 プランニングの問題とは	7
2.1. プランニングの問題では検索する領域が膨大に	7
2.2. プランニングの問題における制約	8
パート II. BUSINESS OPTIMIZER のダウンロードおよびインストール	9
第3章 RED HAT BUSINESS OPTIMIZER のダウンロード	10
第4章 アプリケーションへの RED HAT BUSINESS OPTIMIZER エンジンのインストール	11
パート III. サンプルの実行	12
第5章 BUSINESS OPTIMIZER サンプルの実行	13
5.1. IDE (INTELLIJ、ECLIPSE、または NETBEANS) での RED HAT BUSINESS OPTIMIZER サンプルの実行	14
パート IV. DECISION SERVER の設定	16
第6章 DECISION SERVER	17
第7章 DECISION SERVER のデプロイ	18
7.1. ブートストラップスイッチ	18
第8章 管理対象 DECISION SERVER	23
8.1. DECISION CENTRAL により管理される DECISION SERVER の設定	23
8.2. 非管理対象 DECISION SERVER	25
第9章 コンテナの作成	26
第10章 DECISION CENTRAL でのコンテナ管理	28
10.1. DECISION CENTRAL でのコンテナ起動、停止、および削除	28
10.2. DECISION CENTRAL でのコンテナ更新	28
10.3. 複数コンテナの管理	29
パート V. RED HAT BUSINESS OPTIMIZER REST API の設定	30
第11章 ナレッジストア REST API	31
11.1. ジョブコール	31
11.2. 組織単位コール	32
11.3. リポジトリコール	34
パート VI. DECISION SERVER を実行するための REST API	36
第12章 DECISION SERVER 用 REST API	37
12.1. RED HAT DECISION MANAGER コマンド	38
第13章 RED HAT BUSINESS OPTIMIZER REST API の設定	40
13.1. [GET] /CONTAINERS	40
13.2. [PUT] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	41
13.3. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS	41

13.4. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	42
13.5. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/SOLVING	42
13.6. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/STATE/TERMINATING-EARLY	44
13.7. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/BESTSOLUTION	44
13.8. [POST] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMFACTCHANGES	45
13.9. [GET] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}/PROBLEMFACTCHANGES/PROCESSED	46
13.10. [DELETE] /CONTAINERS/{CONTAINERID}/SOLVERS/{SOLVERID}	46
パート VII. オーサープランニング用アセットおよびソルバーの作成	47
第14章 DECISION CENTRAL でのソルバー作成	48
14.1. ソルバー終了の設定	48
第15章 RED HAT BUSINESS OPTIMIZER ドメインエディターおよびオーサープランニングアセット	50
第16章 プランニングエンティティ難易度比較子の設定	52
第17章 ガイド付きルールデザイナーを使用したスコア制約の定義	54
付録A バージョン情報	55

前書き

ビジネスルールの作成者として、Red Hat Business Optimizer を使用して、限られたリソースや個別の制約の中でプランニングの問題に対する最適解を見つけ出すことができます。

本書を使用して、Red Hat Business Optimizer をセットアップしてその機能を設定します。

前提条件

Business Optimizer 機能のデモとして、Decision Central に **従業員勤務表** サンプルプロジェクトを作成している。このプロジェクトを表示するには、Red Hat Decision Manager がインストールされている必要があります。Red Hat Decision Manager および Decision Central のインストールおよびセットアップの詳細は『[デシジョンサービスの使用ガイド](#)』を参照してください。

パート I. BUSINESS OPTIMIZER を使用したプランニングの問題 の解決

第1章 RED HAT BUSINESS OPTIMIZER とは

Red Hat Business Optimizer は組み込み可能な軽量プランニングエンジンで、プランニングの問題を最適化します。最適化のためのヒューリスティック法およびメタヒューリスティック法を、非常に効率的なスコア計算と組み合わせ、一般的な Java プログラマーがプランニングの問題を効率的に解決できるようにします。

Red Hat Business Optimizer は、さまざまなユースケースの解決に役立ちます。以下にその例を示します。

- **従業員勤務表/患者一覧:** 看護師の勤務シフト作成を容易にし、病床管理を追跡します。
- **教育機関の時間割:** 授業、コース、試験、および会議の計画を容易にします。
- **工場の計画:** 自動車の組み立てライン、機械の操業計画、および作業員のタスク計画を追跡します。
- **在庫の削減:** 紙や金属などの資源の消費を削減し、無駄を最小限に抑えます。

どの組織も、制約のある限定されたリソース (従業員、資産、時間、資金) を使用して製品やサービスを提供するといった、プランニングの問題に直面しています。

Red Hat Business Optimizer は、Apache Software License 2.0 を使用するオープンソースソフトウェアです。100% Pure Java に認定されており、ほとんどの Java 仮想マシンで稼働します。

1.1. プランニングの問題は NP 完全または NP 困難

例に挙げたユースケースは **通常 NP 完全または NP 困難** であり、以下のことが言えます。

- 問題に対する解を実用的な時間内に検証することが容易である。
- 問題に対する最適解を実用的な時間内に見つけ出す確実な方法がない。

この場合、一般的な 2 つの手法では不十分なので、問題を解くのが予想より困難だと考えられます。

- 力まかせアルゴリズムでは (より高度な類似アルゴリズムであっても)、時間がかかり過ぎる。
- (**ビンパッキング問題** において) **容量が最大のビンから順に詰めてゆく** クイックアルゴリズムでは、最適解からは程遠い解しか得られない。

高度な最適化アルゴリズムを用いる Business Optimizer であれば、このようなプランニングの問題に対する適切な解を、実用的な時間内に見つけ出すことができます。

第2章 プランニングの問題とは

プランニングの問題では、限られたリソースや個別の制約の中で最適なゴールを見つけ出します。最適なゴールは、以下に示すように何らかの物の数となります。

- 最大の利益: 最適なゴールにより、可能な限り高い利益が得られます。
- 経済活動の最小フットプリント: 最適なゴールでは、環境負荷が最小となります。
- スタッフ/顧客の最大満足: 最適なゴールでは、スタッフ/顧客のニーズが優先されます。

これらのゴールに到達できるかどうかは、利用できるリソースの数に依存します。たとえば、以下のようリソースには制限があります。

- 要員の人数
- 時間
- 予算
- 装置、車両、コンピューター、施設などの物理資産

これらのリソースに関連する個別の制約についても考慮する必要があります。たとえば、要員が働くことのできる時間数、特定の装置を使用することのできる技能、または機器同士の互換性などです。

Business Optimizer は、最適化のためのヒューリスティック法およびメタヒューリスティック法を効率的なスコア計算と組み合わせ、Java プログラマーが制約充足問題を効率的に解くことができるようにします。

2.1. プランニングの問題では検索する領域が膨大に

プランニングの問題には、多数の解が存在します。

以下に示すように、解は複数のカテゴリーに分類されます。

可能解

可能解とは、制約に違反するかどうかは問わず、あらゆる解を指します。通常、プランニングの問題には膨大な数の可能解が存在します。ただし、それらの解の多くは、無価値なものです。

実行可能解

実行可能解とは、いずれの (負の) ハード制約にも違反しない解を指します。実行可能解の数は、可能解の数に比例する傾向にあります。実行可能解が存在しないケースもあります。実行可能解は、可能解の部分集合です。

最適解

最適解とは、最高のスコアを持つ解を指します。通常、プランニングの問題には数個の最適解が存在します。実行可能解が存在せず、最適解が現実的ではない場合でも、プランニングの問題には少なくとも 1 つの最適解が必ず存在します。

見つかった最善解

見つかった最善解とは、与えられた時間内に実施した検索で見つかった最高スコアの解を指します。通常、見つかった最善解は現実的で、十分な時間があれば最適解を見つけることができます。

直観には反していますが、小規模なデータセットの場合であっても、膨大な数の可能解が存在します (正しく計算された場合)。

planner-engine ディストリビューションフォルダーのサンプルでも、ほとんどのインスタンスには膨大な数の可能解が存在します。最適解を確実に見つけ出すことのできる方法は存在しないため、いかなる実行方法も、これらすべての可能解の部分集合を評価することしかできません。

膨大な数の可能解全体を効率的に網羅するために、Business Optimizer はさまざまな最適化アルゴリズムをサポートしています。

ユースケースによっては、ある最適化アルゴリズムが他のアルゴリズムより優れることがあります。ただし、それを事前に予測することは不可能です。Business Optimizer では、XML またはコード中のソルバー設定を数行変更するだけで、最適化アルゴリズムを切り替えることができます。

2.2. プランニングの問題における制約

通常、プランニングの問題には、少なくとも 2 つの制約レベルがあります。

- **(負の) ハード制約** は、絶対に違反してはならない。
例: 1 人の教師は同時に 2 つの講義を受け持つことはできない。
- **(負の) ソフト制約** は、避けることが可能であれば違反してはならない。
例: 教師 A は金曜日の午後に講義を受け持ちたくない。

正の制約を持つ問題もあります。

- **正のソフト制約 (ボーナス)** は、可能であれば満たす必要がある。
例: 教師 B は月曜日の午前中に講義を受け持つことを希望している。

基本的な問題の中には、ハード制約しか持たないものや、3 つ以上の制約レベル (例: ハード、ミディアム、およびソフト制約) を持つものもあります。

これらの制約により、プランニングの問題における **スコア計算方法** (または **適合度関数**) が定義されます。プランニングの問題の解は、それぞれスコアで等級付けすることができます。Business Optimizer のスコア制約は、Java コード等のオブジェクト指向言語または Drools ルールで記述されます。

このタイプのコードは柔軟で、スケーラビリティに優れます。

パート II. BUSINESS OPTIMIZER のダウンロードおよびインストール

第3章 RED HAT BUSINESS OPTIMIZER のダウンロード

Red Hat Business Optimizer を使用して、Decision Central、ご自分の IDE、または OpenShift でビジネス上のソリューションを最適化することができます。

手順

1. [Red Hat カスタマーポータル](#) から Red Hat Decision Manager をダウンロードしてインストールします。Red Hat Decision Manager の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。
2. [Red Hat カスタマーポータル](#) の Red Hat Decision Manager セクションから **rhdm-7.0.0.GA-add-ons.zip** ファイルをダウンロードします。
3. **rhdm-7.0.0.GA-planner-engine.zip** ファイルから Red Hat Business Optimizer エンジンを展開し、Red Hat Business Optimizer エンジンの JAR ファイルセットにアクセスします。
4. [サンプルを実行します](#)。

以前のバージョンの Red Hat Business Optimizer からの移行については、『[Red Hat JBoss BRMS 6.4 から Red Hat Decision Manager 7.0 への移行](#)』を参照してください。

第4章 アプリケーションへの RED HAT BUSINESS OPTIMIZER エンジンのインストール

Maven またはその他のアプリケーション (Gradle、Ivy、または Buildr 等) を使用して、Red Hat Business Optimizer をインストールすることができます。

前提条件

[Red Hat カスタマーポータル](#) から Red Hat Decision Manager をダウンロードしてインストールしている。Red Hat Decision Manager のインストールの詳細は『[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. [Red Hat JBoss Maven リポジトリ](#) から Red Hat Business Optimizer の **optaplanner-core** JAR を取得します。
2. [Red Hat JBoss Maven リポジトリ](#) を確認して最新のバージョンを探します。
3. お使いのプロジェクトの **pom.xml** に **optaplanner-core** への依存関係を追加します。

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

4. [このステップは必須ではありません] その他の必要な Red Hat Business Optimizer エンジンモジュール (**optaplanner-persistence-jpa** または **optaplanner-persistence-xstream** 統合モジュール等) をすべて追加します。他の Java テクノロジーと共に Red Hat Business Optimizer を使用するための設定方法については、[Optaplanner](#) のドキュメントを参照してください。

注記

optaplanner-benchmark モジュールはエンジンの一部として含まれていますが、[Employee Rostering モジュール](#) で説明するように、独立したモジュールとして使用することを推奨します。これは、**.war** ファイルが不必要な **optaplanner-benchmark** の依存関係を取得するのを防ぐためです。

あるいは、Ant (Ivy なし) を使用している場合は、ダウンロードした ZIP ファイルの **binaries** ディレクトリーからすべての JAR をコピーして、エンジンをインストールすることもできます。クラスパスに重複した JAR が含まれていないことを手作業で確認してください。

注記

ダウンロードした **.zip** ファイルの **binaries** ディレクトリーには、実際に **optaplanner-core** が使用する以外の JAR が多数含まれています。ここには、**optaplanner-benchmark** 等の他のモジュールが使用する JAR も含まれています。

Maven リポジトリの **pom.xml** ファイルを確認して、特定モジュールの特定バージョンで最低限必要な依存関係のセットを判断します。

パート III. サンプルの実行

第5章 BUSINESS OPTIMIZER サンプルの実行

Red Hat Business Optimizer には、さまざまなユースケースのデモとして多数のサンプルが含まれています。

前提条件

[Red Hat カスタマーポータル](#) から Red Hat Decision Manager をダウンロードしてインストールしている。

手順

1. [Red Hat カスタマーポータル](#) の Red Hat Decision Manager セクションから **rhdm-7.0.0.GA-add-ons.zip** ファイルをダウンロードします。
2. **rhdm-7.0.0.GA-planner-engine.zip** ファイルから Red Hat Business Optimizer エンジンを展開します。
3. **rhdm-7.0.0.GA-planner-engine** フォルダーで **examples** ディレクトリーを開き、適切なスクリプトを使用してサンプルを実行します。

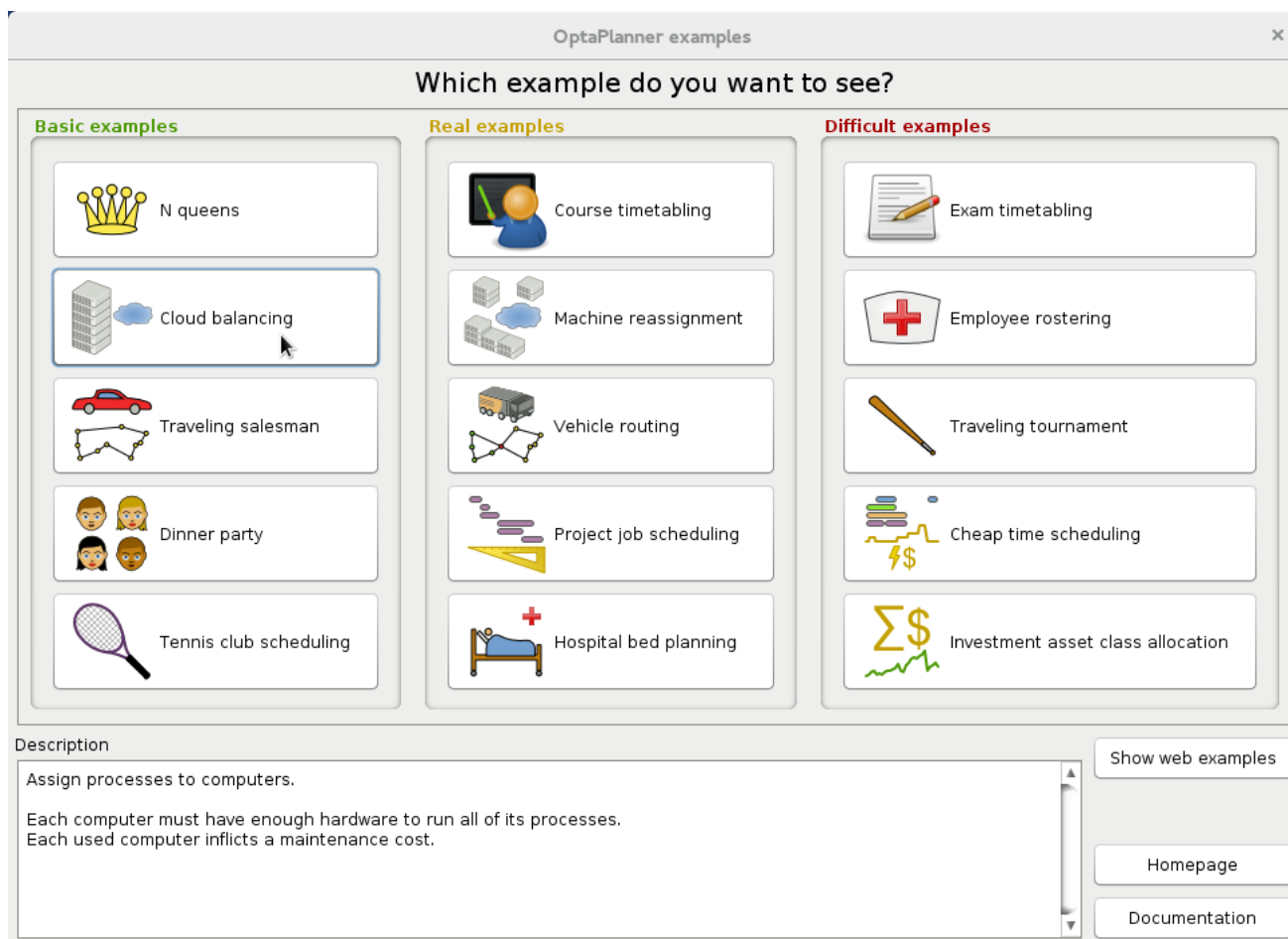
Linux または Mac の場合:

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

Windows の場合:

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

GUI アプリケーションウィンドウからサンプルを選択して実行します。



注記

Red Hat Business Optimizer 自体は GUI に依存していません。サーバーまたはモバイル JVM 上でも、デスクトップとまったく同じように動作します。

5.1. IDE (INTELLIJ、ECLIPSE、または NETBEANS) での RED HAT BUSINESS OPTIMIZER サンプルの実行

IntelliJ または Netbeans で Red Hat Business Optimizer のサンプルを実行するには、以下の手順を使用します。

手順

1. **examples/sources/pom.xml** ファイルを新規プロジェクトとして開きます。
Maven の統合が残りのインストール作業を処理します。

Eclipse で Red Hat Business Optimizer のサンプルを実行するには、以下の手順を使用します。

手順

1. **examples/sources** ディレクトリーで新規プロジェクトを開きます。
2. **binaries** ディレクトリーおよび **examples/binaries** ディレクトリーにあるすべての JAR を、クラスパスに追加します (ファイル **examples/binaries/optaplanner-examples-*.jar** を除く)。

3. Java のソースディレクトリー **src/main/java** と Java のリソースディレクトリー **src/main/resources** を追加します。
4. 実行設定を作成します。
 - メインクラス: **org.optaplanner.examples.app.OptaPlannerExamplesApp**
 - VM パラメーター (任意): **-Xmx512M -server**
 - 作業ディレクトリー: **examples/sources**
5. 実行設定を実行します。

パート IV. DECISION SERVER の設定

第6章 DECISION SERVER

Decision Server はスタンドアロンのビルドインコンポーネントです。これにより、REST、JMS、または Java クライアント側のアプリケーションで利用可能なインターフェース、およびソルバーを通じた Red Hat Business Optimizer 機能を使用して、ルールの実インスタンスを作成して実行できます。Web でデプロイ可能な WAR ファイルとして作成することで、このサーバーは Web コンテナであればどこにでもデプロイできます。Decision Server の現在のバージョンは、Red Hat Decision Manager および Red Hat Business Automation の両方にデフォルトの拡張機能として含まれます。

このサーバーはメモリー消費が最小限でフットプリントが小さいため、クラウドインスタンスに簡単にデプロイできます。このサーバーの各インスタンスでは、複数の KIE コンテナを開いてインスタンスを作成できるので、並行して複数のルールサービスを実行できます。

Decision Central を通じて Decision Server インスタンスをプロビジョニングできます。

第7章 DECISION SERVER のデプロイ

Decision Server は、Web アプリケーションアーカイブ (WAR) ファイル **kie-server.war** として配布されます。Red Hat Decision Manager をインストールする際に、**kie-server.war** ファイルをデプロイしてすべての機能を有効にします。Decision Server のデプロイ方法については、[『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。

前提条件

Red Hat Decision Manager をダウンロードし、**kie-server.war** ファイルと共にデプロイしている。

手順

1. Web コンテナに **kie-server** ロールを持つユーザーを作成します。
2. デシジョンエンジンにアクセスできることを確認します。
 - a. Web ブラウザーで **http://SERVER:PORT/kie-server/services/rest/server/** にアクセスします。
 - b. 前のステップで指定したユーザー名とパスワードを入力します。
3. 認証が済むと、XML の応答がエンジンステータスの形式で表示されます。

```
<?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. ブートストラップスイッチ

Decision Server では、サーバーの動作を設定するためのさまざまなブートストラップスイッチ (システムプロパティ) を使用することができます。

表7.1 Decision Server の拡張機能を無効にするブートストラップスイッチ

プロパティ	値	デフォルト	説明
org.drools.server.ext.disabled	true、false	false	true に設定した場合には、(ルールをサポートなど) Decision Manager のサポートが無効になります。
org.jbpm.server.ext.disabled	true、false	false	true に設定した場合には、(プロセスをサポートなど) Business Automation のサポートが無効になります。

プロパティ	値	デフォルト	説明
<code>org.optaplanner.server.ext.disabled</code>	true、false	false	true に設定した場合には、Business Optimizer のサポートが無効になります。
<code>org.jbpm.ui.server.ext.disabled</code>	true、false	false	true に設定した場合には、Decision Server の UI 拡張が無効になります。
<code>org.kie.executor.disabled</code>	true、false	false	Red Hat Decision Manager エグゼキューターが無効になります。



注記

以下に示すコントローラープロパティの一部は必須と識別されています。Decision Central で Decision Server コンテナの作成や削除を処理する場合には、これらのプロパティを設定します。Decision Central との対話なしに独立して Decision Server を使用する場合には、これらのプロパティを設定する必要はありません。

表7.2 コントローラーの使用に必要なブートストラップスイッチ

プロパティ	値	デフォルト	説明
<code>org.kie.server.id</code>	文字列	該当なし	サーバーに割り当てられる任意の ID。リモートコントローラーが設定されている場合、サーバーはこの ID でコントローラーに接続して KIE コンテナ設定を取得します。指定されていない場合には、ID が自動で生成されます。
<code>org.kie.server.user</code>	文字列	kieserver	コントローラーから Decision Server に接続するのに使用するユーザー名。管理モードで実行する場合には必要です。Decision Central システムプロパティでこのプロパティを設定します。コントローラーを使用する場合には、このプロパティを設定します。
<code>org.kie.server.pwd</code>	文字列	kieserver1!	コントローラーから Decision Server に接続するのに使用するパスワード。管理モードで実行する場合には必要です。Decision Central システムプロパティでこのプロパティを設定します。コントローラーを使用する場合には、このプロパティを設定します。

プロパティ	値	デフォルト	説明
<code>org.kie.server.token</code>	文字列	該当なし	このプロパティにより、コントローラーと Decision Server 間の認証に、ユーザー名/パスワードを使用する基本認証ではなく、トークンベースの認証を使用できます。コントローラーは、リクエストヘッダーのパラメーターとしてトークンを送信します。トークンは更新されないため、サーバーには有効期限の長いアクセストークンが必要です。
<code>org.kie.server.location</code>	URL	該当なし	コントローラーが Decision Server インスタンスをコールバックするのに使用する URL (例: http://localhost:8230/kie-server/services/rest/server)。コントローラーを使用する場合には、このプロパティの設定が必須です。
<code>org.kie.server.controller</code>	コンマ区切りのリスト	該当なし	コントローラー REST エンドポイントへの URL のコンマ区切りリスト(例: http://localhost:8080/decision-central/rest/controller)。コントローラーを使用する場合には、このプロパティの設定が必須です。
<code>org.kie.server.controller.user</code>	文字列	<code>kieserver</code>	コントローラー REST API に接続するためのユーザー名。コントローラーを使用する場合には、このプロパティの設定が必須です。
<code>org.kie.server.controller.pwd</code>	文字列	<code>kieserver1!</code>	コントローラー REST API に接続するためのパスワード。コントローラーを使用する場合には、このプロパティの設定が必須です。
<code>org.kie.server.controller.token</code>	文字列	該当なし	このプロパティにより、Decision Server とコントローラー間の認証に、ユーザー名/パスワードを使用する基本認証ではなく、トークンベースの認証を使用できます。サーバーは、リクエストヘッダーのパラメーターとしてトークンを送信します。トークンは更新されないため、有効期限の長いアクセストークンが必要な点に注意してください。
<code>org.kie.server.controller.connect</code>	Long	<code>10000</code>	サーバーの起動時に Decision Server をコントローラーに接続することを試み、次に試みるまでの待機時間 (ミリ秒)。

表7.3 エグゼキュータープロパティのブートストラップスイッチ

プロパティ	値	デフォルト	説明
<code>org.kie.executor.interval</code>	整数	3	Red Hat Decision Manager エグゼキューターがジョブを完了してから、新しいジョブを開始するまでの時間。時間の単位は <code>org.kie.executor.timeunit</code> プロパティで指定します。
<code>org.kie.executor.timeunit</code>	<code>java.util.concurrent.TimeUnit</code> 定数	SECONDS	<code>org.kie.executor.interval</code> プロパティで指定する時間の単位。
<code>org.kie.executor.pool.size</code>	整数	1	Red Hat Decision Manager エグゼキューターで使用するスレッド数。
<code>org.kie.executor.retry.count</code>	整数	3	Red Hat Decision Manager エグゼキューターが失敗したジョブをリトライする回数。

表7.4 その他のブートストラップスイッチ

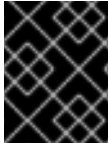
プロパティ	値	デフォルト	説明
<code>kie.maven.settings.custom</code>	パス	該当なし	Maven 設定のカスタム <code>settings.xml</code> ファイルの場所。
<code>kie.server.jms.queues.response</code>	文字列	queue/KIE.SERVER.RESPONSE	JMS に対する応答キューの JNDI 名。
<code>org.drools.server.filter.classes</code>	true 、 false	false	true に設定した場合、Drools Decision Server の拡張機能が受け入れるのは XmlRootElement または Remotable のアノテーションが付いたカスタムクラスのみです。
<code>org.kie.server.domain</code>	文字列	該当なし	JMS を使用する場合にユーザーの認証に使う JAAS LoginContext ドメイン。
<code>org.kie.server.repo</code>	パス	.	Decision Server の状態ファイルが保存される場所。

プロパティ	値	デフォルト	説明
<code>org.kie.server.sync.deploy</code>	<code>true</code> , <code>false</code>	<code>false</code>	<p>Decision Server に対して、コントローラーがコンテナのデプロイメント設定を提供するまでデプロイメントを保持するように指示します。このプロパティは、管理モードで実行するサーバーのみが対象です。このプロパティのオプションは以下のとおりです。</p> <ul style="list-style-type: none">● false: コントローラーへの接続は非同期です。アプリケーションが起動してコントローラーへの接続に成功すると、コンテナをデプロイします。アプリケーションはコンテナが利用可能になる前でもリクエストを受け付けます。● true: メインのサーバーアプリケーションのデプロイメントとコントローラーへの接続スレットを結び付け、その完了を待ちます。同じサーバーインスタンス上に他のアプリケーションがあると、このオプションによりデッドロックが生じる可能性があります。サーバーインスタンス 1 台に対し、アプリケーション (サーバー) は 1 つのみを使用することを推奨します。

第8章 管理対象 DECISION SERVER

管理対象インスタンスには、Decision Server を起動するために利用可能なコントローラーが必要です。

コントローラーは、Decision Server の設定を一元的に管理します。各コントローラーは複数の設定を一度に管理でき、環境内に複数のコントローラーを配置することができます。管理対象 Decision Server に複数のコントローラーを設定できますが、一度に接続することができるのは 1 台だけです。



重要

どのコントローラーに接続されても同じ設定セットがサーバーに提供されるように、コントローラーは同期する必要があります。

Decision Server に複数のコントローラーが設定されている場合には、いずれかのコントローラーとの接続が正常に確立されるまで、起動時に各コントローラーに対して接続を試みます。接続を確立できない場合には、設定でローカルのストレージが利用可能な場合でもサーバーは起動しません。こうすることで、整合性を保ち、冗長設定でサーバーが実行されるのを回避します。



注記

コントローラーに接続せずにスタンドアロンモードで Decision Server を実行する方法については、「[非管理対象 Decision Server](#)」を参照してください。

8.1. DECISION CENTRAL により管理される DECISION SERVER の設定



警告

このセクションでは、テスト目的で使用可能なサンプルの設定を紹介します。一部の値は、実稼働環境には適しておらず、その旨を記載しています。

以下の手順を使用して、Decision Server インスタンスを管理するように Decision Central を設定します。

前提条件

以下のロールを持つユーザーが存在している

- Decision Central: **rest-all** ロールを持つユーザー
- Decision Server: **kie-server** ロールを持つユーザー



注記

実稼働環境では、2 人の異なるユーザーを使用し、それぞれロールを 1 つ割り当ててください。このサンプルでは、**rest-all** と **kie-server** の両ロールを持つ **controllerUser** という名前のユーザー 1 人のみを使用します。

手順

1. 以下の JVM プロパティを設定します。

Decision Central と Decision Server の場所は異なる可能性があります。このような場合には、正しいサーバーインスタンスのプロパティを設定するようにしてください。

- Red Hat JBoss EAP で、以下のファイルの **<system-properties>** セクションを変更します。
 - スタンドアロンモードの場合:
EAP_HOME/standalone/configuration/standalone*.xml
 - ドメインモードの場合: **EAP_HOME/domain/configuration/domain.xml**

表8.1 管理対象 Decision Server インスタンスの JVM プロパティ

プロパティ	値	注記
org.kie.server.id	default-kie-server	Decision Server の ID
org.kie.server.controller	http://localhost:8080/decision-central/rest/controller	Decision Central の場所
org.kie.server.controller.user	controllerUser	前のステップで説明した rest-all ロールを持つユーザーの名前
org.kie.server.controller.pwd	controllerUser1234;	前のステップで説明したユーザーのパスワード
org.kie.server.location	http://localhost:8080/kie-server/services/rest/server	Decision Server の場所

表8.2 Decision Central インスタンスの JVM プロパティ

プロパティ	値	注記
org.kie.server.user	controllerUser	前のステップで説明した kie-server ロールを持つユーザーの名前
org.kie.server.pwd	controllerUser1234;	前のステップで説明したユーザーのパスワード

2. <http://SERVER:PORT/kie-server/services/rest/server/> に GET リクエストを送信して Decision Server が正常に起動したことを確認します。認証が終わると、以下のような XML 応答が返されます。

```
<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. 登録が正常に完了したことを確認します。

a. Decision Central にログインします。

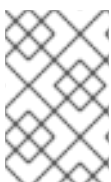
b. **Menu** → **Deploy** → **Execution Servers** の順にクリックします。
正常に登録されている場合には、登録されたサーバーの ID が表示されます。

8.2. 非管理対象 DECISION SERVER

非管理対象 Decision Server はスタンドアロンインスタンスであるため、Decision Server 自体から REST/JMS API を使用して個別に設定する必要があります。コントローラーは使用しません。再起動時には、サーバーが自動的に設定をファイルに永続化し、そのファイルが内部のサーバーの状態として使用されます。

以下の操作を実行中に、設定が更新されます。

- KIE コンテナのデプロイ
- KIE コンテナのデプロイ解除
- KIE コンテナの起動
- KIE コンテナの停止



注記

Decision Server が再起動すると、シャットダウン前に永続化された状態を再度確立しようと試みます。そのため、実行していた KIE コンテナは起動しますが、停止していたコンテナは起動しません。

第9章 コンテナの作成

コンテナとは、パッケージングおよびデプロイされたルールインスタンスを保持するためにプロビジョニングされた独立型の環境です。

前提条件

Decision Server を登録し、コンテナの追加を開始できるようにしている。

手順

1. Decision Central にログインします。
2. メインメニューで、**Menu** → **Deploy** → **Execution Servers** の順にクリックします。
3. **SERVER TEMPLATES** → **+ New Server Template** をクリックします。
4. **New Server Template** ウィンドウで、**Container** をクリックします。
5. 手作業で **Group Name**、**Artifact Id**、および **Version** を入力します。
あるいは、コンテナの名前を入力して、コンテナにデプロイするプロジェクトを検索します。
6. プロジェクトの詳細を自動的に入力するには、プロジェクトの横にある **Select** をクリックします。



警告

コンテナのバージョン番号を入力するときに、**LATEST** または **RELEASE** のキーワードを使用 **しないでください**。この機能は非推奨になっており、デプロイメント時に問題が発生する可能性があります。

7. コンテナを自動的にデプロイするには、**Start Container?** チェックボックスを選択します。
8. **Finish** をクリックします。

コンテナが正常に作成されたら、リストからコンテナを選択し、右上隅の **Start** をクリックして起動します。

コンテナが実行されていることを確認するには、エンドポイントに [GET] リクエストを送信します。

例9.1 サーバーの応答

```
<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>
```

第10章 DECISION CENTRAL でのコンテナ管理

Decision Server 内のコンテナは、Decision Central から起動、停止、更新できます。

10.1. DECISION CENTRAL でのコンテナ起動、停止、および削除

Decision Central で作成されたコンテナは、デフォルトでは停止しています。

コンテナを起動するには、以下の手順を使用します。

前提条件

Decision Central でコンテナが作成および設定されている。

手順

1. Decision Central にログインします。
2. メインメニューで、**Menu** → **Deploy** → **Execution Servers** の順にクリックします。
3. ページの左側の **SERVER TEMPLATES** セクションからサーバーを選択します。
4. 左側の **KIE CONTAINERS** セクションで、起動するコンテナを選択します。
5. 右上隅の **Start** をクリックします。
また、実行中のコンテナを停止するには、**Stop** をクリックします。コンテナが停止したら、**Remove** をクリックして削除できます。

10.2. DECISION CENTRAL でのコンテナ更新

Decision Server を再起動せずにデプロイしたコンテナを更新できます。ビジネスルールの変更により新バージョンのパッケージがプロビジョニングされる場合に、この機能が必要となります。同じパッケージの複数バージョンをプロビジョニングしてデプロイすることができます。

コンテナをアップグレードするには、以下の手順を使用します。

前提条件

Decision Server に Decision Central で実行中のコンテナが設定されている。

手順

1. Decision Central にログインします。
2. メインメニューで、**Menu** → **Deploy** → **Execution Servers** の順にクリックします。
3. ページの左側の **SERVER TEMPLATES** セクションからサーバーを選択します。
4. 左側の **KIE CONTAINERS** セクションで、アップグレードするコンテナを選択します。
5. 上部の **Version Configuration** タブをクリックします。
6. 新しいバージョンを入力して **Upgrade** をクリックします。

7. [このステップは必須ではありません] 手動での編集を行わずに、デプロイしたコンテナに対して常に最新版のデプロイメントが適用されるようにする場合は、**Version** の値を **LATEST** に設定して、**Scan Now** をクリックします。
スキャン中にリポジトリでより新しいバージョンのコンテナデプロイメントが見つかった場合は、コンテナをこの新しいバージョンに自動的にアップグレードします。バックグラウンドでスキャナーを起動するには、**Start Scanner** をクリックし、ミリ秒単位でスキャンの間隔を指定します。

デプロイメントを初めて作成する場合には、**Version** の値を **LATEST** に設定することができます。

10.3. 複数コンテナの管理

Decision Server では、複数のコンテナを作成してプロビジョニングすることができます。複数のコンテナを作成してそれらの操作を行うには、以下の手順を使用します。

前提条件

Decision Server が登録され、Decision Central にログインしている。

手順

1. メインメニューで、**Menu** → **Deploy** → **Execution Servers** の順にクリックします。
2. ページの左側の **SERVER TEMPLATES** セクションからサーバーを選択します。
3. **REMOTE SERVERS** セクションでサーバーを選択して、すべてのコンテナとその状態を表示します。

パート V. RED HAT BUSINESS OPTIMIZER REST API の設定



注記

すべての REST API コールには、エンドポイントベース URL となる以下の URL が使用されます。**`http://SERVER:PORT/decision-central/rest/REQUEST_BODY`**

第11章 ナレッジストア REST API

ナレッジストア REST API に対する REST API コールにより、組織単位、リポジトリ、およびプロジェクトを管理することができます。

すべての **POST** および **DELETE** コールは、リクエストの詳細およびジョブ ID を返します。この ID を使用して、ジョブのステータスを要求してジョブが正常に完了したかどうかを確認できます。**GET** コールは、リポジトリ、プロジェクト、および組織単位に関する情報を返します。

これらのコールのパラメーターおよび結果は、JSON エンティティの形式で提供されます。さまざまなエンティティの Java クラスが **org.guvnor.rest.client** パッケージから利用可能です。詳細については、以降のセクションで説明します。

11.1. ジョブコール

ほとんどのナレッジストア REST コールは、発行後にジョブ ID を返します。コールは同期されていないため、後でジョブを参照してライフサイクルの進行に応じてそのステータスを確認する際に、この ID が必要になります。

ライフサイクルの進行に応じて、ジョブは以下のステータスをとります。

表11.1 ジョブのステータス

ステータス	説明
ACCEPTED	ジョブが受け入れられ、処理中である。
BAD_REQUEST	無効な内容が含まれるため、リクエストが受け入れられなかった。
RESOURCE_NOT_EXIST	要求されたリソース (パス) が存在しない。
DUPLICATE_RESOURCE	リソースがすでに存在する。
SERVER_ERROR	サーバー側のエラーが発生した。
SUCCESS	ジョブが正常に完了した。
FAIL	ジョブが失敗した。
APPROVED	ジョブが承認された。
DENIED	ジョブが許否された。

ステータス	説明
GONE	<p>ジョブ ID が見つからなかった。以下の場合にジョブのステータスが GONE となります。</p> <ul style="list-style-type: none"> ジョブが意図的に削除された。 ジョブが完了してステータスキャッシュから削除されている。キャッシュが最大容量に達すると、ジョブはステータスキャッシュから削除されます。 ジョブが存在しなかった。

以下のジョブコールを利用することができます。

[GET] /jobs/JOB_ID

指定した **JOB_ID** のステータスを返します。

例11.1 リポジトリのクローン作成を要求する GET ジョブコールに対するフォーマット済み応答

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

[DELETE] /jobs/JOB_ID

指定した **JOB_ID** のジョブを削除します。ジョブがまだ処理中でなければ、コールによりジョブキューからジョブが削除されます。ただし、このコールでは、処理中のジョブはキャンセルまたは停止されません。

これらのジョブコールは、共に **JobResult** インスタンスを返します。

11.2. 組織単位コール

ナレッジストアに対する組織単位コールにより、部署や部門をモデル化する際に役立つ組織単位を管理することができます。組織単位は複数のリポジトリを持つことができます。

以下の組織単位コールを利用することができます。

[GET] /organizationalunits/

すべての組織単位の一覧を返します。

例11.2 JSON 形式の組織単位一覧

```
[ {
  "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

特定の組織単位に関する情報を返します。

[POST] /organizationalunits/

ナレッジストアに組織単位を作成します。組織単位は JSON エンティティとして定義されます。コールには **OrganizationalUnit** インスタンスが必要で、**CreateOrganizationalUnitRequest** インスタンスを返します。

例11.3 JSON 形式の組織単位

```

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

```

[POST] /organizationalunits/ORGANIZATIONAL_UNIT_NAME

既存組織単位の詳細を更新します。

要求した **UpdateOrganizationalUnit** インスタンスの **name** および **owner** フィールドは、共に空欄のままにすることができます。この操作により、**description** フィールドやリポジトリの関連性を更新することはできません。

例11.4 JSON 形式の組織単位更新用インプット

```

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

```

[DELETE] /organizationalunits/ORGANIZATIONAL_UNIT_NAME

指定した組織単位を削除します。

[POST] /organizationalunits/ORGANIZATIONAL_UNIT_NAME/repositories/REPOSITORY_NAME

組織単位にリポジトリを追加します。

[DELETE]

/organizationalunits/ORGANIZATIONAL_UNIT_NAME/repositories/REPOSITORY_NAME

組織単位からリポジトリを削除します。

11.3. リポジトリコール

ナレッジストアに対するリポジトリコールにより、Git リポジトリおよびそのプロジェクトを管理することができます。

以下のリポジトリコールを利用することができます。

[GET] /repositories

ナレッジストア内のリポジトリ一覧を返します。

例11.5 リポジトリコールの応答

```
[
  {
    "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

特定のリポジトリに関する情報を返します。

[DELETE] /repositories/REPOSITORY_NAME

リポジトリを削除します。

[POST] /repositories/

JSON エンティティーで定義したリポジトリを作成、またはリポジトリのクローンを作成します。

例11.6 クローンを作成するリポジトリの詳細を定義した JSON エンティティー

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

例11.7 作成するリポジトリの詳細を定義した JSON エンティティ

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



重要

リポジトリを作成、またはリポジトリのクローンを作成する前に、クエリーに **organizationalUnitName** key-value ペアが含まれていること、および指定した組織単位が存在することを確認してください。

[GET] /repositories/REPOSITORY_NAME/projects/

特定リポジトリ内のプロジェクト一覧を JSON エンティティとして返します。

例11.8 既存のプロジェクトに関する詳細が含まれる JSON エンティティ

```
[ {
  "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/

リポジトリにプロジェクトを作成します。

例11.9 作成するプロジェクトを定義したリクエストボディ

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

[DELETE] /repositories/REPOSITORY_NAME/projects/PROJECT_NAME

リポジトリ内のプロジェクトを削除します。

パート VI. **DECISION SERVER** を実行するための **REST API**

第12章 DECISION SERVER 用 REST API

REST API を使用して Decision Server と通信することができます。

- リクエストを送信するためのベース URL は、上で定義したエンドポイントです (例: `http://SERVER:PORT/kie-server/services/rest/server/`)。
- すべてのリクエストには、**kie-server** ロールに対する HTTP 基本認証またはトークンベースの認証が必要です。

以下のメソッドでは、3 種類のリクエストフォーマット (JSON、JAXB、および XSTREAM) がサポートされます。以下の HTTP ヘッダーを指定する必要があります。

- **Accept: application/json** または **application/xml** に設定します。
Accept ヘッダーで複数の受け入れ可能コンテンツタイプを指定する場合は、優先修飾子 ([HTML 1.1 規格](#) で定義される `qvalue`) を含める必要があります。これを含めないと、予期せぬ挙動を示す場合があります。複数の受け入れ可能コンテンツタイプを指定した、適切なヘッダーフォーマットの例を以下に示します。

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

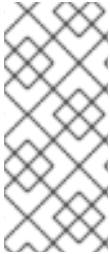
- **X-KIE-ContentType: XSTREAM** マーシャラーを使用する際に必要です。この場合、ヘッダーを **XSTREAM** に設定します。値に **JSON** および **JAXB** を指定することができますが、必須ではありません。**Content-type** を **application/xml** に設定すると、デフォルトでは値に **JAXB** が使用されます。
- **Content-type: application/json** または **application/xml** に設定します。このヘッダーは、お使いのペイロードのフォーマットに対応します。
- **--data:** お使いのペイロードです。ペイロードがファイルにある場合は、名前の始めに **@** 記号を付けます。以下に例を示します。

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

リクエストと応答が共に同じフォーマットになるように、アプリケーションのリクエストには必ず **Content-Type** HTTP ヘッダーと **Accept** HTTP ヘッダーの両方を指定します。指定しないと、サーバーからマーシャリングに関するエラーが送付されます。

例では Curl ユーティリティーが使われていますが、任意の REST クライアントを使用することができます。Curl コマンドには以下のパラメーターが使われます。

- **-u:** Decision Server の認証に `username:password` を指定します。
- **-H:** HTTP ヘッダーを指定します。
- **-X:** リクエストの HTTP メソッドを指定します。メソッドは `[GET]`、`[POST]`、`[PUT]`、または `[DELETE]` のいずれかです。



注記

[Red Hat Decision Manager コマンド](#) は、Decision Server が Red Hat Decision Manager 機能を持つ場合に限り有効です。その他のエンドポイントは、Decision Server が Red Hat Business Automation 機能を持つ場合に限り有効です。**http://SERVER:PORT/kie-server/services/rest/server** の URI にアクセスして、お使いの Decision Server の機能を確認してください。

12.1. RED HAT DECISION MANAGER コマンド

[POST] /containers/instances/CONTAINER_ID

リクエストのタイプ

単一の `org.kie.api.command.Command` コマンド、または `BatchExecutionCommand` ラッパーの複数コマンド。

応答のタイプ

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

説明

指定した **CONTAINER_ID** に送られたコマンドを実行し、コマンドの実行結果を返します。詳細は、以下に示すサポートされるコマンドを参照してください。

サポートされるコマンド一覧:

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

- **DisposeCommand**

コマンドの詳細は、**org.drools.core.command.runtime** パッケージを参照してください。

[POST] Drools コマンドの実行

1. 任意のディレクトリーに移動し、**commandsRequest.json** を作成します。

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

2. 以下のコマンドを実行します。

```
$ 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
```

このコマンドによりリクエストが生成され、このリクエストがサーバーに Insert Object および Fire All Rules コマンドを送信します。**Lookup** により kjar で設定される ksession が指定されます。lookup 値に null を指定した場合は、デフォルトの KIE セッションが使用されます。

応答の例:

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

第13章 RED HAT BUSINESS OPTIMIZER REST API の設定

Decision Server は、以下の Red Hat Business Optimizer REST API をサポートします。これらの API は、JMS および Java クライアント API から利用することもできます。

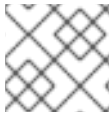
前提条件

Red Hat Decision Manager がインストールされ、**kie-server** ロールに HTTP 基本認証が設定されている。サーバーエンドポイントのベース URL が識別されている (例:

<http://SERVER:PORT/CONTEXT/services/rest/server/>)。

1. 特定のマーシャリングフォーマットが必要な場合は、以下のように、HTTP ヘッダー **Content-Type** と、任意で **X-KIE-ContentType** を HTTP リクエストに追加します。

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



注記

X-KIE-ContentType の値には、**xstream**、**xml**、**json** を設定することができます。

以下の例に示すリクエストおよび応答は、KIE コンテナが Decision Central の従業員勤務表サンプルを使用してビルドされる前提です (**/services/rest/server/containers/employee-rostering** に **PUT** リクエストを呼び出すと、以下の応答が返されます)。

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

13.1. [GET] /CONTAINERS

作成されたコンテナの一覧を返します。

例13.1 サーバー応答の例 (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>employee-rostering</group-id>
        <artifact-id>employee-rostering</artifact-id>
        <version>1.0.0-SNAPSHOT</version>
      </release-id>
      <resolved-release-id>
        <group-id>employee-rostering</group-id>
        <artifact-id>employee-rostering</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}

コンテナ **{containerId}** に、指定の **{solverId}** で新しいソルバーを作成します。リクエストボディはマーシャリングされた **SolverInstance** エンティティになり、ソルバーの設定ファイルを指定する必要があります。

リクエストとそれに対応する応答の例を以下に示します。

例13.2 サーバーリクエストの例 (XStream)

```

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

```

例13.3 サーバー応答の例 (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

コンテナで作成したソルバーの一覧を返します。

例13.4 サーバー応答の例 (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}

コンテナ {containerId} のソルバー {solverId} に関する現在の状態を返します。

例13.5 サーバー応答の例 (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

コンテナ {containerId} のソルバー {solverId} がまだ実行中でなければ、そのソルバーを起動します。リクエストボディは最適化する **PlanningSolution** をマーシャリングしたものです。

以下は、2 台のコンピューターと 6 つのプロセスに関する OptaCloud の問題を解決する例です。ソルバーは同期せずに実行されます。最善解を取得するために、bestsolution URL にリクエストを送信します。

例13.6 サーバリクエストの例 (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.Emp
loyee/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Emp
loyee[3]/skills/employee rostering.employee rostering.Skill"/>
  <employee rostering.employee rostering.Skill
reference="../../../../employeeList/employee rostering.employee rostering.Emp
loyee[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.Employee Roster>

```

13.6. [POST]

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

ソルバーが実行中の場合は、ソルバーに早期の終了をリクエストします。ソルバーは削除されないため、引き続き最善解を取得することができます。

13.7. [GET]

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

リクエストの作成時に見つかった最善解を返します。ソルバーがまだ終了していない (つまり **status** フィールドは **SOLVING** のまま) 場合は、その時点での最善解を返しますが、後のコールでより良い解が返される可能性もあります。

例13.7 サーバー応答の例 (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.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"/>
    <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

リアルタイムプランニング機能。1 つまたは複数の ProblemFactChanges を送信して、現在ソルバーが最適化中のデータセットを更新します。

13.9. [GET]

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

リアルタイムプランニング機能。ソルバーが送信されているすべての ProblemFactChanges を処理したら、true を返します。そうでない場合は、false を返します。

13.10. [DELETE]

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

コンテナ {containerId} のソルバー {solverId} を破棄します。ソルバーがまだ終了していない場合は、終了してから破棄を行います。

パート VII. オーサープランニング用アセットおよびソルバーの作成

第14章 DECISION CENTRAL でのソルバー作成

ソルバーエディターによりソルバー設定を作成し、KJAR のデプロイ後に Execution Solver またはプレーンな Java コードでその設定を実行します。

Decision Central でソルバー設定を編集および作成することができます。

前提条件

Red Hat Decision Manager をダウンロードしてインストールしている。プロジェクトが作成され、すべての関連アセットが設定された状態でデプロイされている。

この例では、ソルバー機能のデモとして同梱された従業員勤務表のサンプルプロジェクトを使用しています。

手順

1. Decision Central で **Menu** → **Projects** をクリックし、使用する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** ウィンドウでソルバーの名前を入力し、**Ok** をクリックします。
これにより、**Solver configuration** エディターが表示されます。
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。
 - a. ナレッジベースに定義したナレッジセッションのいずれかを選択します。
セッションは、Decision Central の **Project** パースペクティブで管理できます (**Menu** → **Projects** をクリックします)。



注記

ナレッジセッションを指定しないと、Red Hat Business Optimizer ではデフォルトのセッションが使用されます。

Score Director Factory

Knowledge base

defaultKieBase

Knowledge session

defaultKieSession

5. **Validate** ボタンをクリックして、ソルバー設定を検証します。この操作で実際にソルバーがビルドされ、デプロイして実行することなくプロジェクトのほとんどの問題を把握することができます。

デフォルトでは、ソルバー設定は自動的にすべてのプランニングエンティティおよびプランニングソリューションクラスを探します。何も見つからない (または多すぎる) と、検証は失敗します。

14.1. ソルバー終了の設定

デフォルトでは、プランニングエンジンが問題のインスタンスを解決する時間に制限はありません。

シナリオによっては (たとえばリアルタイムプランニング)、このデフォルト設定を適用する必要があります。ただし、解決プロセスの総時間をコントロールするメカニズムを用意しておくことが有効です。

サポートされる終了タイプの詳細については、[OptaPlanner のドキュメント](#) を参照してください。



注記

Decision Server REST API を使用して、ソルバーを手動で終了することができます。

前提条件

プロジェクトをデプロイし、Decision Central でソルバーを作成している。

手順

1. **Solver editor** を開きます。
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を選択します。これが、終了設定の入力フィールドとして追加されます。
終了要素は、ツリー構造で整理されます。

エディターでは、論理グループ (終了タイプ **Nested termination** と表示) を定義することができます。複数の終了要素は、論理演算子 (**And/Or**) で結合します。演算子が適用される範囲は、演算子を定義する論理グループに限定されます。

終了のツリー構造から終了要素を削除するには、**Remove** をクリックします。



注記


論理グループの親要素を削除すると、その子要素も削除されます。

第15章 RED HAT BUSINESS OPTIMIZER ドメインエディターおよびオーサープランニングアセット

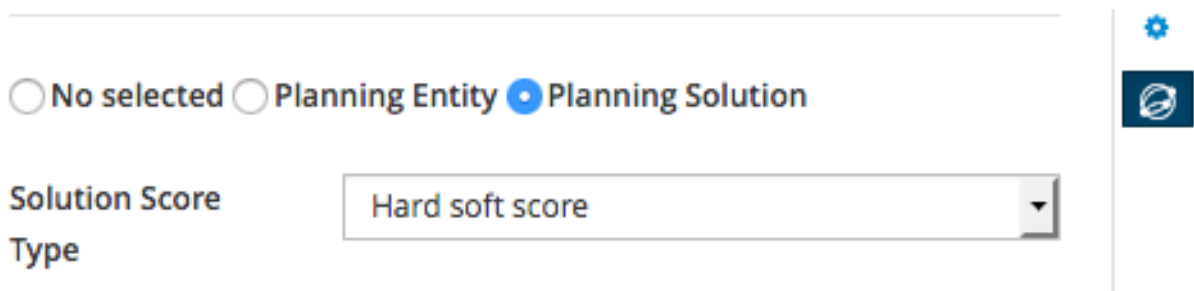
Red Hat Business Optimizer では、データモデラーを活用して制約充足問題のドメインモデルを作成します。

基本機能 (データオブジェクトおよびそのプロパティの作成) に加えて、Decision Central の **Data Objects** エディターパースペクティブを使用して、Red Hat Business Optimizer 固有のデータオブジェクトロールによりデータのモデリングを強化することもできます。**Data Objects** パースペクティブ右



側の  をクリックして Red Hat Business Optimizer ドックを表示し、**Planning Entity** および **Planning Solution** ロールを利用することができます。

これらのオプションは、Red Hat Business Optimizer ドメインエディターから利用することができます。



The screenshot shows a user interface with three radio buttons: 'No selected', 'Planning Entity', and 'Planning Solution'. The 'Planning Solution' button is selected. Below these is a label 'Solution Score Type' and a dropdown menu currently displaying 'Hard soft score'. On the right side of the interface, there are two icons: a gear icon and a square icon with a circular arrow.

ドメインエディターの内容は、その時の選択の状態により異なります。

- データオブジェクトを選択すると、データオブジェクトレベルで定義した設定が表示されます ([Planning Solution](#)、[Planning Entity](#))。
- データオブジェクトのプロパティを選択すると、データオブジェクトのプロパティレベルで定義した設定が表示されます。

Red Hat Business Optimizer ドメインエディター機能の概要については、以下のセクションに説明があります。

データオブジェクトレベル

- なし
- [Planning Entity](#)
 - [プランニングエンティティ難易度比較子の設定](#)
- [Planning Solution](#)
 - [Solution Score Type](#)

プロパティレベル

- なし
- [Planning Entity](#)
 - [Planning Variable](#)

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

第16章 プランニングエンティティの難易度比較子の設定

Planning Entity レベルで指定した難易度比較子を用いて、どのプランニングエンティティの計画がより困難かを設定することができます。これにより、最適化アルゴリズムを効率的に実行することができます。詳細については、[OptaPlanner のドキュメント](#) を参照してください。

前提条件

Data Object エディターのデータオブジェクトで **Planning Entity** を選択しており、**Data Object** パースpekティブの Red Hat Business Optimizer ドメインエディターで難易度比較子定義ツールが利用可能である。

手順

1. **Add condition** をクリックして、指定したプランニングエンティティに、新しい並べ替え条件を追加します。
2. 条件を追加したら、**Add field** をクリックして難易度比較に使用するフィールドを選択します。フィールドには以下の 2 つのタイプがあります。

- **Basic:** 値タイプ (例: 数値、文字列)
- **Data object:** 入れ子属性を持つ複合タイプ
Data object タイプでは、basic タイプに達するまでオブジェクトの階層をネスティングすることができます。basic タイプに達すると、**Add field** ボタンは表示されなくなります。

☐ No selected
 ☒ Planning Entity
 ☐ Planning Solution

☒ Use difficulty comparator for sorting planning entities

1. + Add field ↑ ↓ ↓^A_Z

EmployeeRoster × shiftList × requiredSkill ×

2. + Add field ↑ ↓ ↓^A_Z

EmployeeRoster × employeeList × skills ×

3. + Add field ↑ ↓ ↓^A_Z

EmployeeRoster × employeeList × name ×


+ Add condition

並べ替え条件が配列されます。Red Hat Business Optimizer エンジンがプランニングエンティティの難易度を解決する際、定義された順に条件が優先されます。

3. 並べ替え条件からフィールドを削除するには、ラベル内の **x** アイコンをクリックします。
フィールドのタイプが **Data object** の場合は、フィールドを削除するとその子フィールドもすべて削除されます。
 - 条件の優先順位を変更するには、**上向き矢印**、**下向き矢印** をクリックして条件を上下に移動します。
 - プランニングエンティティの条件の並べ替えを昇順または降順に設定するには、**Sort order** アイコンをクリックします。

第17章 ガイド付きルールデザイナーを使用したスコア制約の定義

最適化の問題を解決するには、解を評価するためのスコア制約を定義する必要があります。Red Hat Business Optimizer にはガイド付きルールデザイナーが組み込まれ、スコア修飾子を利用することができます。エンジンは、解決プロセス時にこの修飾子を使用します。


プロジェクトにプランニングソリューションが定義されている場合には、ルールの **THEN** セクションの右側にあるアクションセクター () から、スコア修飾子を利用することができます。

ガイド付きルールデザイナーを使用して新たなスコア制約を作成するには、以下の手順を使用します。






前提条件

ドメインエディターでクラスを作成して **Planning Solution** を選択し、[プランニングソリューション](#) を定義している。詳細については、「[15章 Red Hat Business Optimizer ドメインエディターおよびオーサープランニングアセット](#)」を参照してください。

手順

1. **Menu** → **Projects** をクリックし、該当するプロジェクトをクリックしてプロジェクトの **Assets** 画面を開きます。
2. 右上隅の **Create New Asset** をクリックします。
3. ドロップダウンメニューから **Guided Rule** を選択します。
4. **Create new Guided Rule** ウィンドウでルールの名前を入力し、**Ok** をクリックします。
5. すでに作成済みのスコア制約を拡張する場合は、**EXTEND** の横のドロップダウンボックスをクリックして、リストからルールを選択します。
6. **WHEN** セクションの右側にあるプラスアイコン () をクリックし、条件を追加します。
7. アクションを選択すると、ルールの **THEN** セクションの右側に Business Optimizer のスコアインプットが表示されます。
ガイド付きルールデザイナーでは、以下の Red Hat Business Optimizer アクションを使用することができます。
 - **Modify a single score level:** 1 つのスコアコンポーネントだけを変更する場合に、このアクションを使用します (例: ハードスコア)
 - **Modify multiple score levels:** 同時に複数のスコアコンポーネントを変更する場合に、このアクションを使用します (例: ハードスコアおよびソフトスコア)
8. テキストの入力フィールドに制約の値を入力します。
9. **Validate** をクリックし、入力した値が正しいことを確認します。

THEN

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

ガイド付きルールデザイナー使用の詳細については、『[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。

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