Red Hat Decision Manager 7.4

KIE API を使った Red Hat Decision Manager の操作

Last Updated: 2019-08-01

Red Hat Decision Manager 7.4 KIE API を使った Red Hat Decision Manager の操作

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

法律上の通知

Copyright © 2019 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, the Red Hat 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 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、KIE API を使用して Red Hat Decision Manager 7.4 の Decision Server、KIE コンテナー、およびビジネスアセットを操作する方法について説明します。

目次

前書き

ビジネスルールの開発者やシステム管理者は、KIE API を使って Red Hat Decision Manager の Decision Server、KIE コンテナー、およびビジネスアセットを操作できます。KIE コンテナーおよびビジネスアセット (ビジネスルールやプロセス、ソルバーなど) には Decision Server REST API と Java クライアント API を、Decision Server テンプレートとインスタンスには Decision Manager コントローラー REST API と Java クライアント API を、Business Central 内のスペースとプロジェクトには Knowledge Store REST API を使って操作します。

DECISION SERVER および DECISION MANAGER コントローラー向け REST API エンドポイント

Decision Server および Decision Manager コントローラー向け REST API エンドポイント一覧は本書とは別個に発行されており、エンドポイントオプションとデータが最新のものに維持されるように、動的にメンテナンスされています。Decision Server および Decision Manager コントローラー REST API で可能になることとその使い方については本書を使用し、特定エンドポイントの詳細については別個にメンテナンスされている REST API エンドポイント一覧を参照してください。

Decision Server REST API エンドポイントの完全一覧と説明については、以下のリソースを参照してください。

- jBPM ドキュメントページ (静的) の Execution Server REST API
- **http://SERVER:PORT/kie-server/docs** (動的。稼働中の Decision Server が必要) ページの Decision Server REST API 用 Swagger UI

Decision Manager コントローラー REST API エンドポイントの完全一覧と説明については、以下のリソースを参照してください。

- jBPM ドキュメントページ (静的) の Controller REST API
- **http://SERVER:PORT/CONTROLLER/docs** (動的。稼働中の Decision Manager コントローラーが必要) ページの Decision Manager コントローラー REST API 用 Swagger UI

前提条件

- Red Hat Decision Manager がインストールされ、実行中である。インストールとスタートアップオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。
- 以下のロールを持つユーザーで Red Hat Decision Manager へのアクセスがある。
  - **kie-server**: Decision Server API 機能へのアクセス、および Business Central なしでヘッドレス Decision Manager コントローラー API 機能にアクセスするため (該当する場合)。
  - **rest-all**: ビルトインの Decision Manager コントローラーおよび Business Central ナレッジストア用の Business Central API 機能にアクセスするため。
  - **admin**: Red Hat Decision Manager への完全な管理者アクセス用。
    各 KIE API ですべてのユーザーロールが必要なわけではありませんが、これらすべてを取得しておくといずれの KIE API にも問題なくアクセスできるようになります。ユーザーロールについての詳細は、『Red Hat Decision Manager インストールのプランニング』を参照してください。

第1章 KIE コンテナーおよびビジネスアセット用の DECISION SERVER REST API

Red Hat Decision Manager は Decision Server REST API を提供し、これを使用することで Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager の KIE コンテナーやビジネスアセット (ビジネスルールやプロセス、ソルバーなど) を操作することができます。この API のサポートにより、Red Hat Decision Manager のリソースをより効率的に維持でき、Red Hat Decision Manager の統合と開発を最適化できるようになります。

Decision Server REST API を使用すると、以下のアクションが可能になります。

- KIE コンテナーのデプロイまたは破棄
- KIE コンテナー情報の取得および更新
- Decision Server ステータスおよび基本的情報の確認
- ビジネスアセット情報の取得および更新
- ビジネスアセット (ルールやプロセス) の実行

Decision Server REST API 要求には以下のコンポーネントが必要です。

認証

Decision Server REST API は、ユーザーロール **kie-server** に HTTP の Basic 認証またはトークンベースの認証を必要とします。お使いの Red Hat Decision Manager に設定されているユーザーロールを表示するには、~/**$SERVER_HOME/standalone/configuration/application-roles.properties** と ~/**application-users.properties** に移動します。
ユーザーに **kie-server** ロールを追加するには、~/**$SERVER_HOME/bin** に移動して以下のコマンドを実行します。

```
$ ./add-user.sh -a --user <USERNAME> --password <PASSWORD> --role kie-server
```

ユーザーロールと Red Hat Decision Manager のインストールオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。

HTTP ヘッダー

Decision Server REST API は、API 要求に以下の HTTP ヘッダーを必要とします。

- **Accept**: 要求元のクライアントが受け付けるデータ形式:
  - **application/json** (JSON)
  - **application/xml** (XML、JAXB または XSTREAM 用)
- **Content-Type**: **POST** または **PUT** API 要求データ向けのデータ形式:
  - **application/json** (JSON)
  - **application/xml** (XML、JAXB または XSTREAM 用)
- **X-KIE-ContentType**: **application/xml** XSTREAM API 要求および応答に必要なヘッダー:
  - **XSTREAM**

HTTP メソッド

Decision Server REST API は、API 要求に以下の HTTP メソッドを必要とします。

- **GET**: 指定したリソースのエンドポイントから指定した情報を取得する
- **POST**: リソースまたはリソースインスタンスを更新する
- **PUT**: リソースまたはリソースインスタンスを更新もしくは作成する
- **DELETE**: リソースまたはリソースインスタンスを削除する

ベース URL

Decision Server REST API 要求のベース URL は **http://SERVER:PORT/kie-server/services/rest/** で、たとえば **http://localhost:8080/kie-server/services/rest/** となります。

Endpoints (エンドポイント)

特定の KIE コンテナーにおける **/server/containers/{containerId}** など、Decision Server REST API のエンドポイントは、Decision Server REST API ベース URL に追記する URI で、Red Hat Decision Manager の対応するリソースやリソースタイプにアクセスするためのものです。

/server/containers/{containerId} エンドポイントの要求 URL 例

http://localhost:8080/kie-server/services/rest/server/containers/MyContainer

要求パラメーターおよび要求データ

多くの Decision Server REST API 要求では、特定リソースを特定またはフィルタリングし、特定のアクションを実行するために、要求 URL パスで特定のパラメーターを必要とします。URL パラメーターは、**?<PARAM>=<VALUE>&<PARAM>=<VALUE>** の形式でエンドポイントに追記します。

GET 要求 URL のパラメーター例

http://localhost:8080/kie-server/services/rest/server/containers?groupId=com.redhat&artifactId=Project1&version=1.0&status=STARTED

HTTP **POST** と **PUT** の要求は、さらに要求の本文もしくはデータのあるファイルが必要になる場合があります。

POST 要求 URL と JSON 要求の本文データの例

http://localhost:8080/kie-server/services/rest/server/containers/MyContainer/release-id

```
{
  "release-id": {
    "artifact-id": "Project1",
    "group-id": "com.redhat",
    "version": "1.1"
  }
}
```

1.1. REST クライアントまたは CURL ユーティリティーを使用した DECISION SERVER REST API による要求送信

Decision Server REST API を使用すると、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager の KIE コンテナーやビジネスアセット (ビジネスルールやプロセス、ソルバーな

ど) を操作することができます。Decision Server REST API 要求は、REST クライアントまたは curl ユーティリティーを使って送信できます。

前提条件

- Decision Server をインストールし、実行している。
- **kie-server** ユーザーロールで Decision Server にアクセスできる。

手順

1. **[GET] /server/containers** など、要求の送信先に適した API endpoint を特定し、Decision Server から KIE コンテナーを取得します。
2. REST クライアントまたは curl ユーティリティーで、**/server/containers** への **GET** 要求に以下のコンポーネントを記入します。ご自分のユースケースに合わせて、要求詳細を調整します。
   REST クライアントの場合:
   - **Authentication**: **kie-server** ロールを持つ Decision Server ユーザーのユーザー名とパスワードを入力します。
   - **HTTP Headers**: 以下のヘッダーを設定します。
     - **Accept**: **application/json**
   - **HTTP method**: **GET** に設定します。
   - **URL**: Decision Server REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/server/containers** となります。

   curl ユーティリティーの場合:
   - **-u**: **kie-server** ロールを持つ Decision Server ユーザーのユーザー名とパスワードを入力します。
   - **-H**: 以下のヘッダーを設定します。
     - **accept**: **application/json**
   - **-X**: **GET** に設定します。
   - **URL**: Decision Server REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/server/containers** となります。

   ```
   curl -u 'baAdmin:password@1' -H "accept: application/json" -X GET
   "http://localhost:8080/kie-server/services/rest/server/containers"
   ```
3. 要求を実行し、Decision Server の応答を確認します。
   サーバー応答の例 (JSON):

   ```
   {
     "type": "SUCCESS",
     "msg": "List of created containers",
     "result": {
       "kie-containers": {
         "kie-container": [
   ```

```
        {
          "container-id": "itorders_1.0.0-SNAPSHOT",
          "release-id": {
            "group-id": "itorders",
            "artifact-id": "itorders",
            "version": "1.0.0-SNAPSHOT"
          },
          "resolved-release-id": {
            "group-id": "itorders",
            "artifact-id": "itorders",
            "version": "1.0.0-SNAPSHOT"
          },
          "status": "STARTED",
          "scanner": {
            "status": "DISPOSED",
            "poll-interval": null
          },
          "config-items": [],
          "container-alias": "itorders"
        }
      ]
    }
  }
}
```

4. この例では、プロジェクトの **group-id**、**artifact-id**、および **version** (GAV) のデータを応答で返されたデプロイ済み KIE コンテナーのいずれかからコピーするか、書き留めます。

5. REST クライアントまたは curl ユーティリティーで、**/server/containers/{containerId}** への **PUT** 要求を以下のコンポーネントで送信し、コピーしたプロジェクトの GAV データで新規 KIE コンテナーをデプロイします。ご自分のユースケースに合わせて、要求詳細を調整します。REST クライアントの場合:

   - **Authentication**: **kie-server** ロールを持つ Decision Server ユーザーのユーザー名とパスワードを入力します。
   - **HTTP Headers**: 以下のヘッダーを設定します。
     - **Accept**: **application/json**
     - **Content-Type**: **application/json**
   - **HTTP method**: **PUT** に設定します。
   - **URL**: Decision Server REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/server/containers/MyContainer** となります。
   - **要求の本文**: 新規 KIE コンテナー用の設定アイテムのある JSON 要求の本文を追加します。

```
{
  "config-items": [
    {
      "itemName": "RuntimeStrategy",
      "itemValue": "SINGLETON",
      "itemType": "java.lang.String"
```

```
    },
    {
      "itemName": "MergeMode",
      "itemValue": "MERGE_COLLECTIONS",
      "itemType": "java.lang.String"
    },
    {
      "itemName": "KBase",
      "itemValue": "",
      "itemType": "java.lang.String"
    },
    {
      "itemName": "KSession",
      "itemValue": "",
      "itemType": "java.lang.String"
    }
  ],
  "release-id": {
    "group-id": "itorders",
    "artifact-id": "itorders",
    "version": "1.0.0-SNAPSHOT"
  },
  "scanner": {
    "poll-interval": "5000",
    "status": "STARTED"
  }
}
```

curl ユーティリティーの場合:

- **-u**: **kie-server** ロールを持つ Decision Server ユーザーのユーザー名とパスワードを入力します。
- **-H**: 以下のヘッダーを設定します。
  - **accept**: **application/json**
  - **content-type**: **application/json**
- **-X**: **PUT** に設定します。
- **URL**: Decision Server REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/server/containers/MyContainer** となります。
- **-d**: 新規 KIE コンテナー用の設定アイテムのある JSON 要求の本文またはファイル (**@file.json**) を追加します。

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type:
application/json" -X PUT "http://localhost:8080/kie-
server/services/rest/server/containers/MyContainer" -d "{ \"config-items\": [ { \"itemName\":
\"RuntimeStrategy\", \"itemValue\": \"SINGLETON\", \"itemType\": \"java.lang.String\" }, {
\"itemName\": \"MergeMode\", \"itemValue\": \"MERGE_COLLECTIONS\", \"itemType\":
\"java.lang.String\" }, { \"itemName\": \"KBase\", \"itemValue\": \"\", \"itemType\":
\"java.lang.String\" }, { \"itemName\": \"KSession\", \"itemValue\": \"\", \"itemType\":
```

```
\"java.lang.String\" } ], \"release-id\": { \"group-id\": \"itorders\", \"artifact-id\": \"itorders\",
\"version\": \"1.0.0-SNAPSHOT\" }, \"scanner\": { \"poll-interval\": \"5000\", \"status\":
\"STARTED\" }}"
```

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type:
application/json" -X PUT "http://localhost:8080/kie-
server/services/rest/server/containers/MyContainer" -d @my-container-configs.json
```

6. 要求を実行し、Decision Server の応答を確認します。
サーバー応答の例 (JSON):

```
{
  "type": "SUCCESS",
  "msg": "Container MyContainer successfully deployed with module itorders:itorders:1.0.0-
SNAPSHOT.",
  "result": {
    "kie-container": {
      "container-id": "MyContainer",
      "release-id": {
        "group-id": "itorders",
        "artifact-id": "itorders",
        "version": "1.0.0-SNAPSHOT"
      },
      "resolved-release-id": {
        "group-id": "itorders",
        "artifact-id": "itorders",
        "version": "1.0.0-SNAPSHOT"
      },
      "status": "STARTED",
      "scanner": {
        "status": "STARTED",
        "poll-interval": 5000
      },
      "config-items": [],
      "messages": [
        {
          "severity": "INFO",
          "timestamp": {
            "java.util.Date": 1540584717937
          },
          "content": [
            "Container MyContainer successfully created with module itorders:itorders:1.0.0-
SNAPSHOT."
          ]
        }
      ],
      "container-alias": null
    }
  }
}
```

エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて要求を調整します。

1.2. SWAGGER インターフェースを使用した DECISION SERVER REST API による要求送信

Decision Server REST API は Swagger web インターフェースをサポートしています。スタンドアロンの REST クライアントや curl ユーティリティーの代わりのこれを使用すると、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager の KIE コンテナーやビジネスアセット (ビジネスルールやプロセス、ソルバーなど) を操作することができます。

注記

デフォルトでは、**org.kie.swagger.server.ext.disabled=false** のシステムプロパティーが指定されており、Decision Server の Swagger Web インターフェースが有効になっています。Decision Server で Swagger Web インターフェースを無効にするには、このシステムプロパティーを **true** に設定してください。

前提条件

- Decision Server をインストールし、実行している。
- **kie-server** ユーザーロールで Decision Server にアクセスできる。

手順

1. Web ブラウザーで **http://SERVER:PORT/kie-server/docs** を開きます。たとえば、**http://localhost:8080/kie-server/docs** となります。**kie-server** ロールを持つ Decision Server ユーザーのユーザー名とパスワードでログインします。
2. Swagger ページで、要求の送信先となる関連 API エンドポイントを選択します。たとえば、**KIE Server and KIE containers→ [GET] /server/containers** で KIE コンテナーを Decision Server から取得します。
3. **Try it out** をクリックして、結果のフィルタリングに使用するオプションのパラメーターを提供します。
4. **Response content type** ドロップダウンメニューで、サーバー応答のフォーマットを選択します。たとえば、JSON フォーマットでは **application/json** となります。
5. **Execute** をクリックし、Decision Server の応答を確認します。
サーバー応答の例 (JSON):


```
{
  "type": "SUCCESS",
  "msg": "List of created containers",
  "result": {
    "kie-containers": {
      "kie-container": [
        {
          "container-id": "itorders_1.0.0-SNAPSHOT",
          "release-id": {
            "group-id": "itorders",
            "artifact-id": "itorders",
            "version": "1.0.0-SNAPSHOT"
          },
          "resolved-release-id": {
            "group-id": "itorders",
```

```
            "artifact-id": "itorders",
            "version": "1.0.0-SNAPSHOT"
          },
          "status": "STARTED",
          "scanner": {
            "status": "DISPOSED",
            "poll-interval": null
          },
          "config-items": [],
          "container-alias": "itorders"
        }
      ]
    }
  }
}
```

6. この例では、プロジェクトの **group-id**、**artifact-id**、および **version** (GAV) のデータを応答で返されたデプロイ済み KIE コンテナーのいずれかからコピーするか、書き留めます。

7. Swagger ページで **KIE Server and KIE containers→ [PUT] /server/containers/{containerId}** エンドポイントに移動し、コピーしたプロジェクト GAV データで新規 KIE コンテナーをデプロイするための別の要求を送信します。ご自分のユースケースに合わせて、要求詳細を調整します。

8. **Try it out** をクリックして、以下の要求のコンポーネントを入力します。

   - **containerId**: 新規 KIE コンテナーの ID を入力します。例: **MyContainer**

   - **body**: **Parameter content type** を任意の要求の本文 (JSON の場合は **application/json** など) に設定し、要求の本文に新規 KIE コンテナーの設定アイテムを追加します。

```
{
  "config-items": [
    {
      "itemName": "RuntimeStrategy",
      "itemValue": "SINGLETON",
      "itemType": "java.lang.String"
    },
    {
      "itemName": "MergeMode",
      "itemValue": "MERGE_COLLECTIONS",
      "itemType": "java.lang.String"
    },
    {
      "itemName": "KBase",
      "itemValue": "",
      "itemType": "java.lang.String"
    },
    {
      "itemName": "KSession",
      "itemValue": "",
      "itemType": "java.lang.String"
    }
  ],
  "release-id": {
    "group-id": "itorders",
```

```
    "artifact-id": "itorders",
    "version": "1.0.0-SNAPSHOT"
   },
   "scanner": {
    "poll-interval": "5000",
    "status": "STARTED"
   }
  }
```

9. **Response content type** ドロップダウンメニューで、サーバー応答のフォーマットを選択します。たとえば、JSON フォーマットでは **application/json** となります。

10. **Execute** をクリックし、Decision Server の応答を確認します。
サーバー応答の例 (JSON):

```
{
  "type": "SUCCESS",
  "msg": "Container MyContainer successfully deployed with module itorders:itorders:1.0.0-
SNAPSHOT.",
  "result": {
    "kie-container": {
      "container-id": "MyContainer",
      "release-id": {
        "group-id": "itorders",
        "artifact-id": "itorders",
        "version": "1.0.0-SNAPSHOT"
      },
      "resolved-release-id": {
        "group-id": "itorders",
        "artifact-id": "itorders",
        "version": "1.0.0-SNAPSHOT"
      },
      "status": "STARTED",
      "scanner": {
        "status": "STARTED",
        "poll-interval": 5000
      },
      "config-items": [],
      "messages": [
        {
          "severity": "INFO",
          "timestamp": {
            "java.util.Date": 1540584717937
          },
          "content": [
            "Container MyContainer successfully created with module itorders:itorders:1.0.0-
SNAPSHOT."
          ]
        }
      ],
      "container-alias": null
    }
  }
}
```

エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて要求を調整します。

1.3. サポート対象の DECISION SERVER REST API エンドポイント

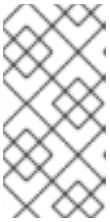
Decision Server REST API は、Red Hat Decision Manager で以下のタイプのリソースにエンドポイントを提供します。

- Decision Server および KIE コンテナー
- KIE セッションアセット (ランタイムコマンド用)
- DMN アセット
- プラニングソルバー

Decision Server REST API のベース URL は **http://SERVER:PORT/kie-server/services/rest/** です。ユーザーロール **kie-server** では、すべての要求で HTTP Basic 認証またはトークンベースの認証が必要です。

Decision Server REST API エンドポイントの完全一覧と説明については、以下のリソースを参照してください。

- jBPM ドキュメントページ (静的) の Execution Server REST API
- **http://SERVER:PORT/kie-server/docs** (動的。稼働中の Decision Server が必要) ページの Decision Server REST API 用 Swagger UI



注記

デフォルトでは、**org.kie.swagger.server.ext.disabled=false** のシステムプロパティーが指定されており、Decision Server の Swagger Web インターフェースが有効になっています。Decision Server で Swagger Web インターフェースを無効にするには、このシステムプロパティーを **true** に設定してください。

第2章 KIE コンテナーおよびビジネスアセット用の DECISION SERVER JAVA クライアント API

Red Hat Decision Manager は Decision Server Java クライアント API を提供し、これを使用することで Java クライアントアプリケーションから REST プロトコルを使って Decision Server に接続できるようになります。Decision Server REST API の代わりに Decision Server Java クライアント API を使って、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager の KIE コンテナーやビジネスアセット (ビジネスルールやプロセス、ソルバーなど) を操作することができます。この API のサポートにより、Red Hat Decision Manager のリソースをより効率的に維持でき、Red Hat Decision Manager の統合と開発を最適化できるようになります。

Decision Server Java クライアント API を使用すると、Decision Server REST API でもサポートされている以下のアクションが実行可能になります。

- KIE コンテナーのデプロイまたは破棄
- KIE コンテナー情報の取得および更新
- Decision Server ステータスおよび基本的情報の確認
- ビジネスアセット情報の取得および更新
- ビジネスアセット (ルールやプロセス) の実行

Decision Server Java クライアント API 要求には以下のコンポーネントが必要です。

認証

Decision Server Java クライアント API は、ユーザーロール **kie-server** に HTTP の Basic 認証を必要とします。お使いの Red Hat Decision Manager に設定されているユーザーロールを表示するには、**~/$SERVER_HOME/standalone/configuration/application-roles.properties** と **~/application-users.properties** に移動します。
ユーザーに **kie-server** ロールを追加するには、**~/$SERVER_HOME/bin** に移動して以下のコマンドを実行します。

```
$ ./add-user.sh -a --user <USERNAME> --password <PASSWORD> --role kie-server
```

ユーザーロールと Red Hat Decision Manager のインストールオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。

プロジェクトの依存関係

Decision Server Java クライアント API には、Java プロジェクト内の適切なクラスパスに、以下の依存関係が必要です。

```
<!-- For remote execution on Decision Server -->
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-client</artifactId>
  <version>${rhdm.version}</version>
</dependency>

<!-- For runtime commands -->
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-compiler</artifactId>
```

```
  <scope>runtime</scope>
  <version>${rhdm.version}</version>
</dependency>

<!-- For debug logging (optional) -->
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
</dependency>
```

Red Hat Decision Manager 依存関係の **<version>** は、プロジェクトで現在使用する Red Hat Decision Manager の Maven アーティファクトバージョンです (例: 7.23.0.Final-redhat-00002)。

注記

個別の依存関係に対して Red Hat Decision Manager **<version>** を指定するのではなく、Red Hat Business Automation 部品表 (BOM) の依存関係をプロジェクトの **pom.xml** ファイルに追加することを検討してください。Red Hat Business Automation BOM は、Red Hat Decision Manager と Red Hat Process Automation Manager の両方に適用します。BOM ファイルを追加すると、指定の Maven リポジトリーからの一時的な依存関係の内、正しいバージョンが、このプロジェクトに追加されます。

BOM 依存関係の例:

```
<dependency>
  <groupId>com.redhat.ba</groupId>
  <artifactId>ba-platform-bom</artifactId>
  <version>7.4.0.GA-redhat-00002</version>
  <scope>import</scope>
  <type>pom</type>
</dependency>
```

Red Hat Business Automation BOM (Bill of Materials) についての詳細情報は、「What is the mapping between RHDM product and maven library version? 」を参照してください。

クライアント要求の設定

Decision Server Java クライアント API による Java クライアント要求はすべて、少なくとも以下のサーバー通信コンポーネントを定義する必要があります。

- **kie-server** ユーザーの認証情報
- Decision Server の場所。たとえば、**http://localhost:8080/kie-server/services/rest/server**
- API 要求および応答のマーシャリングフォーマット (JSON、JAXB、または XSTREAM)
- **KieServicesConfiguration** オブジェクトおよび **KieServicesClient** オブジェクト。これらは、Java クライアント API を使用してサーバー通信を開始するためのエントリーポイントの役割を果たします。
- REST プロトコルおよびユーザーアクセスを定義する **KieServicesFactory** オブジェクト

- 使用される他のクライアントサービス。**RuleServicesClient**、**ProcessServicesClient**、**QueryServicesClient** など

以下は、これらのコンポーネントを使用した基本的および高度なクライアント設定の例です。

基本的クライアント設定の例

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.client.KieServicesClient;
import org.kie.server.client.KieServicesConfiguration;
import org.kie.server.client.KieServicesFactory;

public class MyConfigurationObject {

  private static final String URL = "http://localhost:8080/kie-server/services/rest/server";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

  private static final MarshallingFormat FORMAT = MarshallingFormat.JSON;

  private static KieServicesConfiguration conf;
  private static KieServicesClient kieServicesClient;

  public static void initialize() {
    conf = KieServicesFactory.newRestConfiguration(URL, USER, PASSWORD);

    //If you use custom classes, such as Obj.class, add them to the configuration.
    Set<Class<?>> extraClassList = new HashSet<Class<?>>();
    extraClassList.add(Obj.class);
    conf.addExtraClasses(extraClassList);

    conf.setMarshallingFormat(FORMAT);
    kieServicesClient = KieServicesFactory.newKieServicesClient(conf);
  }
}
```

追加のクライアントサービスを使用した高度なクライアント設定の例

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.client.CaseServicesClient;
import org.kie.server.client.DMNServicesClient;
import org.kie.server.client.DocumentServicesClient;
import org.kie.server.client.JobServicesClient;
import org.kie.server.client.KieServicesClient;
import org.kie.server.client.KieServicesConfiguration;
import org.kie.server.client.KieServicesFactory;
import org.kie.server.client.ProcessServicesClient;
import org.kie.server.client.QueryServicesClient;
import org.kie.server.client.RuleServicesClient;
import org.kie.server.client.SolverServicesClient;
import org.kie.server.client.UIServicesClient;
import org.kie.server.client.UserTaskServicesClient;
import org.kie.server.api.model.instance.ProcessInstance;
import org.kie.server.api.model.KieContainerResource;
import org.kie.server.api.model.ReleaseId;
```

```
public class MyAdvancedConfigurationObject {

    // REST API base URL, credentials, and marshalling format
    private static final String URL = "http://localhost:8080/kie-server/services/rest/server";
    private static final String USER = "baAdmin";
    private static final String PASSWORD = "password@1";;

    private static final MarshallingFormat FORMAT = MarshallingFormat.JSON;

    private static KieServicesConfiguration conf;

    // KIE client for common operations
    private static KieServicesClient kieServicesClient;

    // Rules client
    private static RuleServicesClient ruleClient;

    // Process automation clients
    private static CaseServicesClient caseClient;
    private static DocumentServicesClient documentClient;
    private static JobServicesClient jobClient;
    private static ProcessServicesClient processClient;
    private static QueryServicesClient queryClient;
    private static UIServicesClient uiClient;
    private static UserTaskServicesClient userTaskClient;

    // DMN client
    private static DMNServicesClient dmnClient;

    // Planning client
    private static SolverServicesClient solverClient;

    public static void main(String[] args) {
        initializeKieServerClient();
        initializeDroolsServiceClients();
        initializeJbpmServiceClients();
        initializeSolverServiceClients();
    }

    public static void initializeKieServerClient() {
        conf = KieServicesFactory.newRestConfiguration(URL, USER, PASSWORD);
        conf.setMarshallingFormat(FORMAT);
        kieServicesClient = KieServicesFactory.newKieServicesClient(conf);
    }

    public static void initializeDroolsServiceClients() {
        ruleClient = kieServicesClient.getServicesClient(RuleServicesClient.class);
        dmnClient = kieServicesClient.getServicesClient(DMNServicesClient.class);
    }

    public static void initializeJbpmServiceClients() {
        caseClient = kieServicesClient.getServicesClient(CaseServicesClient.class);
        documentClient = kieServicesClient.getServicesClient(DocumentServicesClient.class);
        jobClient = kieServicesClient.getServicesClient(JobServicesClient.class);
        processClient = kieServicesClient.getServicesClient(ProcessServicesClient.class);
```

```
        queryClient = kieServicesClient.getServicesClient(QueryServicesClient.class);
        uiClient = kieServicesClient.getServicesClient(UIServicesClient.class);
        userTaskClient = kieServicesClient.getServicesClient(UserTaskServicesClient.class);
    }

    public static void initializeSolverServiceClients() {
        solverClient = kieServicesClient.getServicesClient(SolverServicesClient.class);
    }
}
```

2.1. DECISION SERVER JAVA クライアント API を使った要求送信

Decision Server Java クライアント API を使用すると、Java クライアントアプリケーションから REST プロトコルを使って Decision Server に接続できるようになります。Decision Server REST API の代わりに Decision Server Java クライアント API を使って、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager の KIE コンテナーやビジネスアセット (ビジネスルールやプロセス、ソルバーなど) を操作することができます。

前提条件

- Decision Server をインストールし、実行している。
- **kie-server** ユーザーロールで Decision Server にアクセスできる。
- Red Hat Decision Manager リソースを使った Java プロジェクトがある。

手順

1. クライアントアプリケーションで、Java プロジェクト内の適切なクラスパスに以下の依存関係が追加されていることを確認します。

```
<!-- For remote execution on Decision Server -->
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-client</artifactId>
  <version>${rhdm.version}</version>
</dependency>

<!-- For runtime commands -->
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-compiler</artifactId>
  <scope>runtime</scope>
  <version>${rhdm.version}</version>
</dependency>

<!-- For debug logging (optional) -->
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
</dependency>
```

2. Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution**をダ

ウンロードし、~/**rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-remote/kie-server-client/src/main/java/org/kie/server/client** に移動して Decision Server Java クライアントにアクセスします。

3. ~/**kie/server/client** ディレクトリーで、Decision Server の KIE コンテナーや他のアセット用のクライアントサービスにアクセスするために、**KieServicesClient** などの送信する要求用の関連 Java クライアントを特定します。

4. クライアントアプリケーションで、API 要求用の **.java** クラスを作成します。クラスには、Decision Server の場所とユーザー認証情報、**KieServicesClient** オブジェクト、**KieServicesClient** クライアントからの **createContainer** や **disposeContainer** などの実行するクライアントメソッドを含める必要があります。ご自分のユースケースに合わせて、設定詳細を調整します。

コンテナーの作成および破棄

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.client.KieServicesClient;
import org.kie.server.client.KieServicesConfiguration;
import org.kie.server.client.KieServicesFactory;
import org.kie.server.api.model.KieContainerResource;
import org.kie.server.api.model.ServiceResponse;

public class MyConfigurationObject {

  private static final String URL = "http://localhost:8080/kie-server/services/rest/server";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

  private static final MarshallingFormat FORMAT = MarshallingFormat.JSON;

  private static KieServicesConfiguration conf;
  private static KieServicesClient kieServicesClient;

  public static void initialize() {
    conf = KieServicesFactory.newRestConfiguration(URL, USER, PASSWORD);

  public void disposeAndCreateContainer() {
      System.out.println("== Disposing and creating containers ==");

      // Retrieve list of KIE containers
      List<KieContainerResource> kieContainers =
kieServicesClient.listContainers().getResult().getContainers();
      if (kieContainers.size() == 0) {
          System.out.println("No containers available...");
          return;
      }

      // Dispose KIE container
      KieContainerResource container = kieContainers.get(0);
      String containerId = container.getContainerId();
      ServiceResponse<Void> responseDispose =
kieServicesClient.disposeContainer(containerId);
      if (responseDispose.getType() == ResponseType.FAILURE) {
          System.out.println("Error disposing " + containerId + ". Message: ");
          System.out.println(responseDispose.getMsg());
```

```
            return;
        }
        System.out.println("Success Disposing container " + containerId);
        System.out.println("Trying to recreate the container...");

        // Re-create KIE container
        ServiceResponse<KieContainerResource> createResponse =
kieServicesClient.createContainer(containerId, container);
        if(createResponse.getType() == ResponseType.FAILURE) {
            System.out.println("Error creating " + containerId + ". Message: ");
            System.out.println(responseDispose.getMsg());
            return;
        }
        System.out.println("Container recreated with success!");
        }
    }
}
```

org.kie.server.api.model.ServiceResponse<T> オブジェクトを使ってサービス応答を定義します。ここでの **T** は、返される応答のタイプを表します。**ServiceResponse** オブジェクトには以下の属性があります。

- **String message**: 応答メッセージを返します。
- **ResponseType type**: **SUCCESS** または **FAILURE** を返します。
- **T result**: 要求されたオブジェクトを返します。

この例ではコンテナーを破棄する際に、**ServiceResponse** が **Void** 応答を返します。コンテナーの作成時には、**ServiceResponse** が **KieContainerResource** オブジェクトを返します。

注記

クライアントとクラスター環境内の特定の Decision Server コンテナーとの対話は、一意の **conversationID** でセキュリティーが保たれます。**conversationID** は **X-KIE-ConversationId** REST ヘッダーを使って送信されます。コンテナーを更新する場合は、以前の **conversationID** の設定を解除します。**KieServiesClient.completeConversation()** を使って Java API の **conversationID** を設定解除します。

5. 設定済みの **.java** クラスをプロジェクトディレクトリーから実行して、要求を出し、Decision Server の応答を確認します。
デバッグのロギングが有効になっている場合は、JSON などの設定済みマーシャリングフォーマットに従って、Decision Server が詳細を返します。

   新規 KIE コンテナーのサーバー応答の例 (ログ):

```
10:23:35.194 [main] INFO  o.k.s.a.m.MarshallerFactory - Marshaller extensions init
10:23:35.396 [main] DEBUG o.k.s.client.balancer.LoadBalancer - Load balancer
RoundRobinBalancerStrategy{availableEndpoints=[http://localhost:8080/kie-
server/services/rest/server]} selected url 'http://localhost:8080/kie-server/services/rest/server'
10:23:35.398 [main] DEBUG o.k.s.c.i.AbstractKieServicesClientImpl - About to send GET
request to 'http://localhost:8080/kie-server/services/rest/server'
10:23:35.440 [main] DEBUG o.k.s.c.i.AbstractKieServicesClientImpl - About to deserialize
content:
```

```
 '{
  "type" : "SUCCESS",
  "msg" : "Kie Server info",
  "result" : {
    "kie-server-info" : {
      "id" : "default-kieserver",
      "version" : "7.11.0.Final-redhat-00003",
      "name" : "default-kieserver",
      "location" : "http://localhost:8080/kie-server/services/rest/server",
      "capabilities" : [ "KieServer", "BRM", "BPM", "CaseMgmt", "BPM-UI", "BRP", "DMN",
"Swagger" ],
      "messages" : [ {
        "severity" : "INFO",
        "timestamp" : {
  "java.util.Date" : 1540814906533
},
        "content" : [ "Server KieServerInfo{serverId='default-kieserver', version='7.11.0.Final-
redhat-00003', name='default-kieserver', location='http://localhost:8080/kie-
server/services/rest/server', capabilities=[KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP,
DMN, Swagger], messages=null}started successfully at Mon Oct 29 08:08:26 EDT 2018" ]
      } ]
    }
  }
}'
 into type: 'class org.kie.server.api.model.ServiceResponse'
10:23:35.653 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - KieServicesClient
connected to: default-kieserver version 7.11.0.Final-redhat-00003
10:23:35.653 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Supported capabilities by
the server: [KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP, DMN, Swagger]
10:23:35.653 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability KieServer
10:23:35.653 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - No builder found for
'KieServer' capability
10:23:35.654 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability BRM
10:23:35.654 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.DroolsServicesClientBuilder@6b927fb' for capability 'BRM'
10:23:35.655 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.RuleServicesClient=org.kie.server.client.impl.RuleServicesClientImpl@4a94
ee4}
10:23:35.655 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability BPM
10:23:35.656 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.JBPMServicesClientBuilder@4cc451f2' for capability 'BPM'
10:23:35.672 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.JobServicesClient=org.kie.server.client.impl.JobServicesClientImpl@1189d
52, interface
org.kie.server.client.admin.ProcessAdminServicesClient=org.kie.server.client.admin.impl.Proces
sAdminServicesClientImpl@36bc55de, interface
org.kie.server.client.DocumentServicesClient=org.kie.server.client.impl.DocumentServicesClient
mpl@564fabc8, interface
org.kie.server.client.admin.UserTaskAdminServicesClient=org.kie.server.client.admin.impl.User
TaskAdminServicesClientImpl@16d04d3d, interface
org.kie.server.client.QueryServicesClient=org.kie.server.client.impl.QueryServicesClientImpl@4
```

```
9ec71f8, interface
org.kie.server.client.ProcessServicesClient=org.kie.server.client.impl.ProcessServicesClientImp
@1d2adfbe, interface
org.kie.server.client.UserTaskServicesClient=org.kie.server.client.impl.UserTaskServicesClientI
mpl@36902638}
10:23:35.672 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability CaseMgmt
10:23:35.672 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.CaseServicesClientBuilder@223d2c72' for capability 'CaseMgmt'
10:23:35.676 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.admin.CaseAdminServicesClient=org.kie.server.client.admin.impl.CaseAdm
nServicesClientImpl@2b662a77, interface
org.kie.server.client.CaseServicesClient=org.kie.server.client.impl.CaseServicesClientImpl@7f0
eb4b4}
10:23:35.676 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability BPM-UI
10:23:35.676 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.JBPMUIServicesClientBuilder@5c33f1a9' for capability 'BPM-UI'
10:23:35.677 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.UIServicesClient=org.kie.server.client.impl.UIServicesClientImpl@223191a
}
10:23:35.678 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability BRP
10:23:35.678 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.OptaplannerServicesClientBuilder@49139829' for capability
'BRP'
10:23:35.679 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.SolverServicesClient=org.kie.server.client.impl.SolverServicesClientImpl@7
7fbd92c}
10:23:35.679 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability DMN
10:23:35.679 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Builder
'org.kie.server.client.helper.DMNServicesClientBuilder@67c27493' for capability 'DMN'
10:23:35.680 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Capability implemented by
{interface
org.kie.server.client.DMNServicesClient=org.kie.server.client.impl.DMNServicesClientImpl@35e
2d654}
10:23:35.680 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - Building services client for
server capability Swagger
10:23:35.680 [main] DEBUG o.k.s.c.impl.KieServicesClientImpl - No builder found for
'Swagger' capability
10:23:35.681 [main] DEBUG o.k.s.client.balancer.LoadBalancer - Load balancer
RoundRobinBalancerStrategy{availableEndpoints=[http://localhost:8080/kie-
server/services/rest/server]} selected url 'http://localhost:8080/kie-server/services/rest/server'
10:23:35.701 [main] DEBUG o.k.s.c.i.AbstractKieServicesClientImpl - About to send PUT
request to 'http://localhost:8080/kie-server/services/rest/server/containers/employee-
rostering3' with payload '{
  "container-id" : null,
  "release-id" : {
    "group-id" : "employeerostering",
    "artifact-id" : "employeerostering",
    "version" : "1.0.0-SNAPSHOT"
  },
```

```
  "resolved-release-id" : null,
  "status" : null,
  "scanner" : null,
  "config-items" : [ ],
  "messages" : [ ],
  "container-alias" : null
}'
10:23:38.071 [main] DEBUG o.k.s.c.i.AbstractKieServicesClientImpl - About to deserialize
content:
 '{
  "type" : "SUCCESS",
  "msg" : "Container employee-rostering3 successfully deployed with module
employeerostering:employeerostering:1.0.0-SNAPSHOT.",
  "result" : {
    "kie-container" : {
      "container-id" : "employee-rostering3",
      "release-id" : {
        "group-id" : "employeerostering",
        "artifact-id" : "employeerostering",
        "version" : "1.0.0-SNAPSHOT"
      },
      "resolved-release-id" : {
        "group-id" : "employeerostering",
        "artifact-id" : "employeerostering",
        "version" : "1.0.0-SNAPSHOT"
      },
      "status" : "STARTED",
      "scanner" : {
        "status" : "DISPOSED",
        "poll-interval" : null
      },
      "config-items" : [ ],
      "messages" : [ {
        "severity" : "INFO",
        "timestamp" : {
  "java.util.Date" : 1540909418069
},
        "content" : [ "Container employee-rostering3 successfully created with module
employeerostering:employeerostering:1.0.0-SNAPSHOT." ]
      } ],
      "container-alias" : null
    }
  }
}'
 into type: 'class org.kie.server.api.model.ServiceResponse'
```

エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて Java 設定を調整します。

2.2. サポート対象の DECISION SERVER JAVA クライアント

以下は、Red Hat Decision Manager の **org.kie.server.client** パッケージで利用可能な Java クライアントサービスの一部です。これらのサービスを使用して、Decision Server REST API と同様に Decision Server の関連リソースを操作できます。

- **KieServicesClient**: 他の Decision Server Java クライアントのエントリーポイントとととして使用し、KIE コンテナーを操作します。
- **JobServicesClient**: ジョブ要求のスケジュール、キャンセル、キューへの再追加、および取得を行うために使用されます。
- **RuleServicesClient**: ルール関連の操作を実行するためにサーバーにコマンドを送信するために使用されます (ルールの実行、KIE セッションへのオブジェクト挿入など)。
- **SolverServicesClient**: ソルバーステータスや最適解の取得、またはソルバーの破棄など、Red Hat Business Optimizer のすべての操作を実行するために使用されます。

getServicesClient メソッドは、これらのクライアントのいずれかへのアクセスを提供します。

```
RuleServicesClient rulesClient = kieServicesClient.getServicesClient(RuleServicesClient.class);
```

利用可能な Decision Server Java クライアントの完全一覧については、Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution**をダウンロードして、**~/rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-remote/kie-server-client/src/main/java/org/kie/server/client** に移動してください。

2.3. DECISION SERVER JAVA クライアント API を使った要求の例

以下は、Decision Server との基本的な対話のための Decision Server Java クライアント API 要求の例です。利用可能な Decision Server Java クライアントの完全一覧については、Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution**をダウンロードして、**~/rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-remote/kie-server-client/src/main/java/org/kie/server/client** に移動してください。

Decision Server の機能一覧

org.kie.server.api.model.KieServerInfo オブジェクトを使うと、サーバー機能を特定できます。**KieServicesClient** クライアントが適切にサービスクライアントを作り出すには、サーバー機能の情報が必要です。機能は **KieServicesConfiguration** でグローバルに指定することができます。指定しない場合は、Decision Server から自動的に取得されます。

Decision Server 機能を返す要求の例

```
public void listCapabilities() {

  KieServerInfo serverInfo = kieServicesClient.getServerInfo().getResult();
  System.out.print("Server capabilities:");

  for (String capability : serverInfo.getCapabilities()) {
    System.out.print(" " + capability);
  }

  System.out.println();
}
```

Decision Server での KIE コンテナーの一覧

KIE コンテナーは **org.kie.server.api.model.KieContainerResource** オブジェクトで表されます。リソース一覧は、**org.kie.server.api.model.KieContainerResourceList** オブジェクトで表されます。

Decision Server から KIE コンテナーを返す要求の例

```
public void listContainers() {
    KieContainerResourceList containersList = kieServicesClient.listContainers().getResult();
    List<KieContainerResource> kieContainers = containersList.getContainers();
    System.out.println("Available containers: ");
    for (KieContainerResource container : kieContainers) {
        System.out.println("\t" + container.getContainerId() + " (" + container.getReleaseId() + ")");
    }
}
```

org.kie.server.api.model.KieContainerResourceFilter クラスのインスタンスを使って KIE コンテナーの結果をフィルタリングすることもできます。これは **org.kie.server.client.KieServicesClient.listContainers()** メソッドに渡されます。

リリース ID とステータスごとの KIE コンテナーを返す要求の例

```
public void listContainersWithFilter() {

    // Filter containers by releaseId "org.example:container:1.0.0.Final" and status FAILED
    KieContainerResourceFilter filter = new KieContainerResourceFilter.Builder()
            .releaseId("org.example", "container", "1.0.0.Final")
            .status(KieContainerStatus.FAILED)
            .build();

    // Using previously created KieServicesClient
    KieContainerResourceList containersList = kieServicesClient.listContainers(filter).getResult();
    List<KieContainerResource> kieContainers = containersList.getContainers();

    System.out.println("Available containers: ");

    for (KieContainerResource container : kieContainers) {
        System.out.println("\t" + container.getContainerId() + " (" + container.getReleaseId() + ")");
    }
}
```

Decision Server での KIE コンテナーの作成および破棄

KieServicesClient で **createContainer** および **disposeContainer** メソッドを使用すると、KIE コンテナーの作成と破棄ができます。この例では、コンテナーを破棄すると、**ServiceResponse** が **Void** 応答を返します。コンテナーを作成すると、**ServiceResponse** が **KieContainerResource** オブジェクトを返します。

KIE コンテナーを破棄して再作成する要求の例

```
public void disposeAndCreateContainer() {
    System.out.println("== Disposing and creating containers ==");

    // Retrieve list of KIE containers
    List<KieContainerResource> kieContainers =
kieServicesClient.listContainers().getResult().getContainers();
    if (kieContainers.size() == 0) {
        System.out.println("No containers available...");
        return;
    }

    // Dispose KIE container
```

```
    KieContainerResource container = kieContainers.get(0);
    String containerId = container.getContainerId();
    ServiceResponse<Void> responseDispose = kieServicesClient.disposeContainer(containerId);
    if (responseDispose.getType() == ResponseType.FAILURE) {
        System.out.println("Error disposing " + containerId + ". Message: ");
        System.out.println(responseDispose.getMsg());
        return;
    }
    System.out.println("Success Disposing container " + containerId);
    System.out.println("Trying to recreate the container...");

    // Re-create KIE container
    ServiceResponse<KieContainerResource> createResponse =
kieServicesClient.createContainer(containerId, container);
    if(createResponse.getType() == ResponseType.FAILURE) {
        System.out.println("Error creating " + containerId + ". Message: ");
        System.out.println(responseDispose.getMsg());
        return;
    }
    System.out.println("Container recreated with success!");
}
```

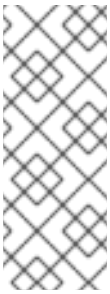
Decision Server でのランタイムコマンドの実行

Red Hat Decision Manager はランタイムコマンドをサポートしています。これは、KIE セッションでオブジェクトを挿入したり取り消したり、全ルールを実行するなどのアセット関連の操作のために Decision Server に送信するものです。サポートされるランタイムコマンドの全一覧は、Red Hat Decision Manager インスタンスの **org.drools.core.command.runtime** パッケージにあります。コマンドの挿入には **org.kie.api.command.KieCommands** クラスを使用し、**KieCommands** クラスのインスタンス化には **org.kie.api.KieServices.get().getCommands()** を使用することもできます。複数のコマンドを追加するには、**BatchExecutionCommand** ラッパーを使用します。

オブジェクトの挿入および全ルール実行の要求の例

```
import org.kie.api.command.Command;
import org.kie.api.command.KieCommands;
import org.kie.server.api.model.ServiceResponse;
import org.kie.server.client.RuleServicesClient;
import org.kie.server.client.KieServicesClient;
import org.kie.api.KieServices;

import java.util.Arrays;

...

public void executeCommands() {

  String containerId = "hello";
  System.out.println("== Sending commands to the server ==");
  RuleServicesClient rulesClient = kieServicesClient.getServicesClient(RuleServicesClient.class);
  KieCommands commandsFactory = KieServices.Factory.get().getCommands();

  Command<?> insert = commandsFactory.newInsert("Some String OBJ");
  Command<?> fireAllRules = commandsFactory.newFireAllRules();
  Command<?> batchCommand = commandsFactory.newBatchExecution(Arrays.asList(insert,
fireAllRules));
```

```
  ServiceResponse<String> executeResponse = rulesClient.executeCommands(containerId,
batchCommand);

  if(executeResponse.getType() == ResponseType.SUCCESS) {
    System.out.println("Commands executed with success! Response: ");
    System.out.println(executeResponse.getResult());
  } else {
    System.out.println("Error executing rules. Message: ");
    System.out.println(executeResponse.getMsg());
  }
}
```



注記

クライアントとクラスター環境内の特定の Decision Server コンテナーとの対話は、一意の **conversationID** でセキュリティーが保たれます。**conversationID** は **X-KIE-ConversationId** REST ヘッダーを使って送信されます。コンテナーを更新する場合は、以前の **conversationID** の設定を解除します。**KieServiesClient.completeConversation()** を使って Java API の **conversationID** を設定解除します。

第3章 RED HAT DECISION MANAGER での DECISION SERVER および KIE コンテナーのコマンド

Red Hat Decision Manager は、サーバー情報を取得したり、コンテナーの作成や削除など、サーバー関連またはコンテナー関連の操作のために Decision Server に送信するサーバーコマンドをサポートしています。サポートされる Decision Server 設定コマンドの全一覧は、Red Hat Decision Manager インスタンスの **org.kie.server.api.commands** パッケージにあります。

Decision Server REST API では、**org.kie.server.api.commands** コマンドを **http://SERVER:PORT/kie-server/services/rest/server/config** への **POST** 要求の本文として使用します。Decision Server REST API の使用に関する詳細情報は、*1章KIE コンテナーおよびビジネスアセット用の Decision Server REST API* を参照してください。

Decision Server Java クライアント API では、親 **KieServicesClient** Java クライアントで対応するメソッドを Java アプリケーションで埋め込み API 要求として使用します。Decision Server コマンドはすべて Java クライアント API で提供されるメソッドが実行するので、実際の Decision Server コマンドを Java アプリケーションに埋め込む必要はありません。Decision Server Java クライアント API の使用に関する詳細情報は、*2章KIE コンテナーおよびビジネスアセット用の Decision Server Java クライアント API* を参照してください。

3.1. DECISION SERVER および KIE コンテナーのコマンドサンプル

以下は、Decision Server で Decision Server REST API または Java クライアント API のサーバー関連もしくはコンテナー関連操作に使用可能な Decision Server コマンドのサンプルです。

- **GetServerInfoCommand**
- **GetServerStateCommand**
- **CreateContainerCommand**
- **GetContainerInfoCommand**
- **ListContainersCommand**
- **CallContainerCommand**
- **DisposeContainerCommand**
- **GetScannerInfoCommand**
- **UpdateScannerCommand**
- **UpdateReleaseIdCommand**

サポートされる Decision Server 設定および管理コマンドの全一覧は、Red Hat Decision Manager インスタンスの **org.kie.server.api.commands** パッケージにあります。

Decision Server コマンドは個別に実行するか、バッチ REST API 要求またはバッチ Java API 要求として実行することができます。

KIE コンテナー (JSON) を作成、呼び出し、破棄するバッチ REST API 要求

```
{
  "commands": [
```

```
    {
      "create-container": {
        "container": {
          "status": "STARTED",
          "container-id": "command-script-container",
          "release-id": {
            "version": "1.0",
            "group-id": "com.redhat",
            "artifact-id": "Project1"
          }
        }
      }
    },
    {
      "call-container": {
        "payload": "{\n  \"commands\" : [ {\n    \"fire-all-rules\" : {\n      \"max\" : -1,\n      \"out-identifier\" :
null\n    }\n  } ]\n}",
        "container-id": "command-script-container"
      }
    },
    {
      "dispose-container": {
        "container-id": "command-script-container"
      }
    }
  ]
}
```

KIE コンテナーを取得、破棄、再作成するバッチ Java API 要求

```
public void disposeAndCreateContainer() {
    System.out.println("== Disposing and creating containers ==");

    // Retrieve list of KIE containers
    List<KieContainerResource> kieContainers =
kieServicesClient.listContainers().getResult().getContainers();
    if (kieContainers.size() == 0) {
        System.out.println("No containers available...");
        return;
    }

    // Dispose KIE container
    KieContainerResource container = kieContainers.get(0);
    String containerId = container.getContainerId();
    ServiceResponse<Void> responseDispose = kieServicesClient.disposeContainer(containerId);
    if (responseDispose.getType() == ResponseType.FAILURE) {
        System.out.println("Error disposing " + containerId + ". Message: ");
        System.out.println(responseDispose.getMsg());
        return;
    }
    System.out.println("Success Disposing container " + containerId);
    System.out.println("Trying to recreate the container...");

    // Re-create KIE container
    ServiceResponse<KieContainerResource> createResponse =
kieServicesClient.createContainer(containerId, container);
```

```
    if(createResponse.getType() == ResponseType.FAILURE) {
        System.out.println("Error creating " + containerId + ". Message: ");
        System.out.println(responseDispose.getMsg());
        return;
    }
    System.out.println("Container recreated with success!");
}
```

本セクションの各コマンドには、Decision Server REST API 用の REST 要求の本文例 (JSON) と Decision Server Java クライアント API 用の **KieServicesClient** Java クライアントからの埋め込みメソッド例が含まれています。

GetServerInfoCommand

Decision Server についての情報を返します。

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "get-server-info" : { }
  } ]
}
```

Java クライアントメソッドの例

```
KieServerInfo serverInfo = kieServicesClient.getServerInfo();
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Kie Server info",
      "result": {
        "kie-server-info": {
          "id": "default-kieserver",
          "version": "7.11.0.Final-redhat-00001",
          "name": "default-kieserver",
          "location": "http://localhost:8080/kie-server/services/rest/server",
          "capabilities": [
            "KieServer",
            "BRM",
            "BPM",
            "CaseMgmt",
            "BPM-UI",
            "BRP",
            "DMN",
            "Swagger"
          ],
          "messages": [
            {
              "severity": "INFO",
              "timestamp": {
```

```
            "java.util.Date": 1538502533321
          },
          "content": [
            "Server KieServerInfo{serverId='default-kieserver', version='7.11.0.Final-redhat-00001',
name='default-kieserver', location='http://localhost:8080/kie-server/services/rest/server',
capabilities=[KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP, DMN, Swagger],
messages=null}started successfully at Tue Oct 02 13:48:53 EDT 2018"
          ]
        }
      ]
    }
  }
 ]
}
```

GetServerStateCommand

Decision Server の現在の状態と設定についての情報を返します。

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "get-server-state" : { }
  } ]
}
```

Java クライアントメソッドの例

```
KieServerStateInfo serverStateInfo = kieServicesClient.getServerState();
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Successfully loaded server state for server id default-kieserver",
      "result": {
        "kie-server-state-info": {
          "controller": [
            "http://localhost:8080/decision-central/rest/controller"
          ],
          "config": {
            "config-items": [
              {
                "itemName": "org.kie.server.location",
                "itemValue": "http://localhost:8080/kie-server/services/rest/server",
                "itemType": "java.lang.String"
              },
              {
                "itemName": "org.kie.server.controller.user",
                "itemValue": "controllerUser",
                "itemType": "java.lang.String"
```

```
            },
            {
              "itemName": "org.kie.server.controller",
              "itemValue": "http://localhost:8080/decision-central/rest/controller",
              "itemType": "java.lang.String"
            }
          ]
        },
        "containers": [
          {
            "container-id": "employee-rostering",
            "release-id": {
              "group-id": "employeerostering",
              "artifact-id": "employeerostering",
              "version": "1.0.0-SNAPSHOT"
            },
            "resolved-release-id": null,
            "status": "STARTED",
            "scanner": {
              "status": "STOPPED",
              "poll-interval": null
            },
            "config-items": [
              {
                "itemName": "KBase",
                "itemValue": "",
                "itemType": "BPM"
              },
              {
                "itemName": "KSession",
                "itemValue": "",
                "itemType": "BPM"
              },
              {
                "itemName": "MergeMode",
                "itemValue": "MERGE_COLLECTIONS",
                "itemType": "BPM"
              },
              {
                "itemName": "RuntimeStrategy",
                "itemValue": "SINGLETON",
                "itemType": "BPM"
              }
            ],
            "messages": [],
            "container-alias": "employeerostering"
          }
        ]
      }
    }
  ]
}
```


CreateContainerCommand

Decision Server の KIE コンテナーを作成します。

表3.1 コマンドの属性

名前	説明	要件
container	**container-id**、**release-id** データ (グループ ID、アーティファクト ID、バージョン)、**status**、および新規 KIE コンテナーの他のコンポーネントを含むマップ。	必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "create-container" : {
      "container" : {
        "status" : null,
        "messages" : [ ],
        "container-id" : "command-script-container",
        "release-id" : {
          "version" : "1.0",
          "group-id" : "com.redhat",
          "artifact-id" : "Project1"
        },
        "config-items" : [ ]
      }
    }
  } ]
}
```

Java クライアントメソッドの例

```
ServiceResponse<KieContainerResource> response =
kieServicesClient.createContainer("command-script-container", resource);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully deployed with module
com.redhat:Project1:1.0.",
      "result": {
        "kie-container": {
          "container-id": "command-script-container",
          "release-id": {
            "version" : "1.0",
            "group-id" : "com.redhat",
            "artifact-id" : "Project1"
          },
          "resolved-release-id": {
```

```
            "version" : "1.0",
            "group-id" : "com.redhat",
            "artifact-id" : "Project1"
          },
          "status": "STARTED",
          "scanner": {
            "status": "DISPOSED",
            "poll-interval": null
          },
          "config-items": [],
          "messages": [
            {
              "severity": "INFO",
              "timestamp": {
                "java.util.Date": 1538762455510
              },
              "content": [
                "Container command-script-container successfully created with module
com.redhat:Project1:1.0."
              ]
            }
          ],
          "container-alias": null
        }
      }
    }
  ]
}
```

GetContainerInfoCommand

Decision Server の指定された KIE コンテナーについての情報を返します。

表3.2 コマンドの属性

名前	説明	要件
container-id	KIE コンテナーの ID	必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "get-container-info" : {
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
ServiceResponse<KieContainerResource> response =
kieServicesClient.getContainerInfo("command-script-container");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Info for container command-script-container",
      "result": {
        "kie-container": {
          "container-id": "command-script-container",
          "release-id": {
            "group-id": "com.redhat",
            "artifact-id": "Project1",
            "version": "1.0"
          },
          "resolved-release-id": {
            "group-id": "com.redhat",
            "artifact-id": "Project1",
            "version": "1.0"
          },
          "status": "STARTED",
          "scanner": {
            "status": "DISPOSED",
            "poll-interval": null
          },
          "config-items": [

          ],
          "container-alias": null
        }
      }
    }
  ]
}
```

ListContainersCommand

Decision Server で作成された KIE コンテナー一覧を返します。

表3.3 コマンドの属性

名前	説明	要件
kie-container-filter	**release-id-filter**、**container-status-filter**、および結果のフィルタリングに使用する他の KIE コンテナープロパティーを含むオプションのマップ。	オプション

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "list-containers" : {
      "kie-container-filter" : {
        "release-id-filter" : { },
```

```
        "container-status-filter" : {
          "accepted-status" : ["FAILED"]
        }
      }
    }
  } ]
}
```

Java クライアントメソッドの例

```
KieContainerResourceFilter filter = new KieContainerResourceFilter.Builder()
        .status(KieContainerStatus.FAILED)
        .build();

KieContainerResourceList containersList = kieServicesClient.listContainers(filter);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "List of created containers",
      "result": {
        "kie-containers": {
          "kie-container": [
            {
              "container-id": "command-script-container",
              "release-id": {
                "group-id": "com.redhat",
                "artifact-id": "Project1",
                "version": "1.0"
              },
              "resolved-release-id": {
                "group-id": "com.redhat",
                "artifact-id": "Project1",
                "version": "1.0"
              },
              "status": "STARTED",
              "scanner": {
                "status": "STARTED",
                "poll-interval": 5000
              },
              "config-items": [
                {
                  "itemName": "RuntimeStrategy",
                  "itemValue": "SINGLETON",
                  "itemType": "java.lang.String"
                },
                {
                  "itemName": "MergeMode",
                  "itemValue": "MERGE_COLLECTIONS",
                  "itemType": "java.lang.String"
                },
                {
```

```
              "itemName": "KBase",
              "itemValue": "",
              "itemType": "java.lang.String"
            },
            {
              "itemName": "KSession",
              "itemValue": "",
              "itemType": "java.lang.String"
            }
          ],
          "messages": [
            {
              "severity": "INFO",
              "timestamp": {
                "java.util.Date": 1538504619749
              },
              "content": [
                "Container command-script-container successfully created with module
com.redhat:Project1:1.0."
              ]
            }
          ],
          "container-alias": null
        }
      ]
    }
  }
 ]
}
```


CallContainerCommand

KIE コンテナーを呼び出し、1 つ以上のランタイムコマンドを実行します。Red Hat Decision Manager ランタイムコマンドについての情報は、*4章Red Hat Decision Manager のランタイムコマンド*を参照してください。

表3.4 コマンドの属性

名前	説明	要件
container-id	呼び出される KIE コンテナーの ID	必須
payload	KIE コンテナーで実行される **BatchExecutionCommand** ラッパー内の 1 つ以上のコマンド	必須

REST 要求の本文 (JSON) 例


```
{
  "commands" : [ {
    "call-container" : {
      "payload" : "{\n  \"lookup\" : \"defaultKieSession\",\n  \"commands\" : [ {\n    \"fire-all-rules\" : {\n
\"max\" : -1,\n      \"out-identifier\" : null\n    }\n  } ]\n}",
```

```
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
List<Command<?>> commands = new ArrayList<Command<?>>();
      BatchExecutionCommand batchExecution1 =
commandsFactory.newBatchExecution(commands, "defaultKieSession");
      commands.add(commandsFactory.newFireAllRules());

      ServiceResponse<ExecutionResults> response1 =
ruleClient.executeCommandsWithResults("command-script-container", batchExecution1);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": "{\n  \"results\" : [ ],\n  \"facts\" : [ ]\n}"
    }
  ]
}
```

DisposeContainerCommand

Decision Server の指定された KIE コンテナーを破棄します。

表3.5 コマンドの属性

名前	説明	要件
container-id	破棄される KIE コンテナーの ID	必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "dispose-container" : {
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
ServiceResponse<Void> response = kieServicesClient.disposeContainer("command-script-
container");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully disposed.",
      "result": null
    }
  ]
}
```

GetScannerInfoCommand

該当する場合、指定された KIE コンテナー内の自動更新に使用される KIE スキャナーについての情報を返します。

表3.6 コマンドの属性

名前	説明	要件
container-id	KIE スキャナーを使用する KIE コンテナーの ID	必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "get-scanner-info" : {
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
ServiceResponse<KieScannerResource> response =
kieServicesClient.getScannerInfo("command-script-container");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Scanner info successfully retrieved",
      "result": {
        "kie-scanner": {
          "status": "DISPOSED",
          "poll-interval": null
        }
      }
```

```
    }
  ]
}
```

UpdateScannerCommand

更新済み KIE コンテナーデプロイメントのポーリングを制御する KIE スキャナーを起動、停止します。

注記

ビジネスプロセスと KIE スキャナーを併用するのは避けてください。プロセスで KIE スキャナーを使用すると、予期せぬ更新が発生し、プロセスインスタンスの実行と互換性のない変更が加えられた場合に、長時間実行中のプロセスでエラーが発生する可能性があります。

表3.7 コマンドの属性

名前	説明	要件
container-id	KIE スキャナーを使用する KIE コンテナーの ID	必須
status	KIE スキャナーに設定するステータス (**STARTED**、**STOPPED**)	必須
poll-interval	ポーリングの時間 (単位: ミリ秒)	スキャナーの起動時にのみ必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "update-scanner" : {
      "scanner" : {
        "status" : "STARTED",
        "poll-interval" : 10000
      },
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
KieScannerResource scannerResource = new KieScannerResource();
scannerResource.setPollInterval(10000);
scannerResource.setStatus(KieScannerStatus. STARTED);

ServiceResponse<KieScannerResource> response =
kieServicesClient.updateScanner("command-script-container", scannerResource);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Kie scanner successfully created.",
      "result": {
        "kie-scanner": {
          "status": "STARTED",
          "poll-interval": 10000
        }
      }
    }
  ]
}
```

UpdateReleaseIdCommand

指定した KIE コンテナーのリリース ID データ (グループ ID、アーティファクト ID、バージョン) を更新します。

表3.8 コマンドの属性

名前	説明	要件
container-id	更新される KIE コンテナーの ID	必須
releaseId	KIE コンテナーに適用される更新済み GAV (グループ ID、アーティファクト ID、バージョン) データ	必須

REST 要求の本文 (JSON) 例

```
{
  "commands" : [ {
    "update-release-id" : {
      "releaseId" : {
        "version" : "1.1",
        "group-id" : "com.redhat",
        "artifact-id" : "Project1"
      },
      "container-id" : "command-script-container"
    }
  } ]
}
```

Java クライアントメソッドの例

```
ServiceResponse<ReleaseId> response = kieServicesClient.updateReleaseId("command-script-container", "com.redhat:Project1:1.1");
```

サーバーの応答例 (JSON)

```
{
```

```
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Release id successfully updated",
      "result": {
        "release-id": {
          "group-id": "com.redhat",
          "artifact-id": "Project1",
          "version": "1.1"
        }
      }
    }
  ]
}
```

第4章 RED HAT DECISION MANAGER のランタイムコマンド

Red Hat Decision Manager はランタイムコマンドをサポートしています。これは、KIE セッションでオブジェクトを挿入したり取り消したり、全ルールを実行するなど、アセット関連の操作のために Decision Server に送信するものです。サポートされるランタイムコマンドの全一覧は、Red Hat Decision Manager インスタンスの **org.drools.core.command.runtime** パッケージにあります。

Decision Server REST API では、グローバルの **org.drools.core.command.runtime** コマンドまたはルール固有の **org.drools.core.command.runtime.rule** コマンドを **http://SERVER:PORT/kie-server/services/rest/server/containers/instances/{containerId}** への**POST** 要求の本文として使用します。Decision Server REST API の使用方法に関しては、*1章KIE コンテナーおよびビジネスアセット用の Decision Server REST API* を参照してください。

Decision Server Java クライアント API では、関連する Java クライアントの Java アプリケーションにこれらのコマンドを埋め込むことができます。たとえばルール関連のコマンドでは、**RuleServicesClient** Java クライアントを埋め込むコマンドに使用します。Decision Server Java クライアント API の使用に関する詳細情報は、*2章KIE コンテナーおよびビジネスアセット用の Decision Server Java クライアント API* を参照してください。

4.1. RED HAT DECISION MANAGER のランタイムコマンドのサンプル

以下は、Decision Server で Decision Server REST API または Java クライアント API のアセット関連演算に使用可能なランタイムコマンドのサンプルです。

- **BatchExecutionCommand**
- **InsertObjectCommand**
- **RetractCommand**
- **ModifyCommand**
- **GetObjectCommand**
- **GetObjectsCommand**
- **InsertElementsCommand**
- **FireAllRulesCommand**
- **QueryCommand**
- **SetGlobalCommand**
- **GetGlobalCommand**

サポートされるランタイムコマンドの全一覧は、Red Hat Decision Manager インスタンスの **org.drools.core.command.runtime** パッケージにあります。

本セクションの各コマンドには、Decision Server REST API 用の REST 要求の本文例 (JSON) と Decision Server Java クライアント API 用の Java クライアントの埋め込みコマンド例が含まれています。Java の例では、**name** (文字列) と **age** (整数) のフィールドがあるオブジェクト **org.drools.compiler.test.Person** を使用しています。

BatchExecutionCommand

同時に実行する複数のコマンドが含まれています。

表4.1 コマンドの属性

名前	説明	要件
commands	実行されるコマンド一覧	必須
lookup	コマンドの実行対象である KIE セッション ID を設定します。ステートレス KIE セッションの場合、この属性は必須です。ステートフル KIE セッションの場合この属性はオプションで、指定されていないとデフォルトの KIE セッションが使用されます。	ステートレス KIE セッションでは必須、ステートフル KIE セッションではオプション

JSON リクエストボディの例

```
{
  "lookup": "ksession1",
  "commands": [ {
      "insert": {
        "object": {
          "org.drools.compiler.test.Person": {
            "name": "john",
            "age": 25
          }
        }
      }
    },
    {
      "fire-all-rules": {
        "max": 10,
        "out-identifier": "firedActivations"
      }
    }
  ]
}
```

Java コマンドの例

```
BatchExecutionCommand command = new BatchExecutionCommand();
command.setLookup("ksession1");

InsertObjectCommand insertObjectCommand = new InsertObjectCommand(new Person("john",
25));
FireAllRulesCommand fireAllRulesCommand = new FireAllRulesCommand();

command.getCommands().add(insertObjectCommand);
command.getCommands().add(fireAllRulesCommand);

ksession.execute(command);
```

サーバーの応答例 (JSON)

```
{
  "response": [
```

```
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [
            {
              "value": 0,
              "key": "firedActivations"
            }
          ],
          "facts": []
        }
      }
    }
  ]
}
```


InsertObjectCommand

KIE セッションにオブジェクトを挿入します。

表4.2 コマンドの属性

名前	説明	要件
object	挿入するオブジェクト	必須
out-identifier	オブジェクト挿入から作成され、実行結果に追加される **FactHandle** の ID	オプション
return-object	実行結果にオブジェクトを返す必要があるかどうかを決定するブール値 (デフォルト値: **true**)	オプション
entry-point	挿入のエントリーポイント	オプション

JSON リクエストボディの例


```
{
  "commands": [ {
    "insert": {
      "entry-point": "my stream",
      "object": {
        "org.drools.compiler.test.Person": {
          "age": 25,
          "name": "john"
        }
      },
      "out-identifier": "john",
      "return-object": false
    }
```

```
    }
  ]
}
```

Java コマンドの例

```
Command insertObjectCommand =
  CommandFactory.newInsert(new Person("john", 25), "john", false, null);

ksession.execute(insertObjectCommand);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [],
          "facts": [
            {
              "value": {
                "org.drools.core.common.DefaultFactHandle": {
                  "external-form": "0:4:436792766:-
2127720265:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap"
                }
              },
              "key": "john"
            }
          ]
        }
      }
    }
  ]
}
```

RetractCommand

KIE セッションからオブジェクトを撤回します。

表4.3 コマンドの属性

名前	説明	要件
fact-handle	撤回するオブジェクトに関連付けられた **FactHandle**	必須

JSON リクエストボディの例

```
{
  "commands": [ {
```

```
      "retract": {
        "fact-handle": "0:4:436792766:-
2127720265:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap"
      }
    }
  ]
}
```

Java コマンドの使用例: FactHandleFromString

```
RetractCommand retractCommand = new RetractCommand();
retractCommand.setFactHandleFromString("123:234:345:456:567");
```

Java コマンドの使用例: 挿入されたオブジェクトから FactHandle

```
RetractCommand retractCommand = new RetractCommand(factHandle);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container employee-rostering successfully called.",
      "result": {
        "execution-results": {
          "results": [],
          "facts": []
        }
      }
    }
  ]
}
```

ModifyCommand

KIE セッションに以前に挿入されたオブジェクトを修正します。

表4.4 コマンドの属性

名前	説明	要件
fact-handle	修正するオブジェクトに関連付けられた **FactHandle**	必須
setters	オブジェクト修正のセッター一覧	必須

JSON リクエストボディの例

```
{
  "commands": [ {
    "modify": {
```

```
        "fact-handle": "0:4:436792766:-
2127720265:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap",
        "setters": {
          "accessor": "age",
          "value": 25
        }
      }
    }
  ]
}
```

Java コマンドの例

```
ModifyCommand modifyCommand = new ModifyCommand(factHandle);

List<Setter> setters = new ArrayList<Setter>();
setters.add(new SetterImpl("age", "25"));

modifyCommand.setSetters(setters);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container employee-rostering successfully called.",
      "result": {
        "execution-results": {
          "results": [],
          "facts": []
        }
      }
    }
  ]
}
```

GetObjectCommand

KIE セッションからオブジェクトを取得します。

表4.5 コマンドの属性

名前	説明	要件
fact-handle	取得するオブジェクトに関連付けられた **FactHandle**	必須
out-identifier	オブジェクト挿入から作成され、実行結果に追加される **FactHandle** の ID	オプション

JSON リクエストボディの例

```
{
  "commands": [ {
     "get-object": {
       "fact-handle": "0:4:436792766:-
2127720265:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap",
       "out-identifier": "john"
     }
    }
  ]
}
```

Java コマンドの例

```
GetObjectCommand getObjectCommand = new GetObjectCommand();
getObjectCommand.setFactHandleFromString("123:234:345:456:567");
getObjectCommand.setOutIdentifier("john");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [
            {
              "value": null,
              "key": "john"
            }
          ],
          "facts": []
        }
      }
    }
  ]
}
```

GetObjectsCommand

KIE セッションからすべてのオブジェクトをコレクションとして取得します。

表4.6 コマンドの属性

名前	説明	要件
object-filter	KIE セッションから返されるオブジェクト用のフィルター	オプション
out-identifier	実行結果に使用する識別子	オプション

JSON リクエストボディの例

```
{
  "commands": [ {
     "get-objects": {
      "out-identifier": "objects"
     }
    }
  ]
}
```

Java コマンドの例

```
GetObjectsCommand getObjectsCommand = new GetObjectsCommand();
getObjectsCommand.setOutIdentifier("objects");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [
            {
              "value": [
                {
                  "org.apache.xerces.dom.ElementNSImpl": "<?xml version=\"1.0\" encoding=\"UTF-
16\"?>\n<object xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:type=\"person\">
<age>25</age><name>john</name>\n <\/object>"
                },
                {
                  "org.drools.compiler.test.Person": {
                    "name": "john",
                    "age": 25
                  }
                }
              ],
              "key": "objects"
            }
          ],
          "facts": []
        }
      }
    }
  ]
}
```

InsertElementsCommand

KIE セッションにオブジェクト一覧を挿入します。

表4.7 コマンドの属性

名前	説明	要件
objects	KIE セッションに挿入するオブジェクト一覧	必須
out-identifier	オブジェクト挿入から作成され、実行結果に追加される **FactHandle** の ID	オプション
return-object	実行結果にオブジェクトを返す必要があるかどうかを決定するブール値。デフォルト値: **true**。	オプション
entry-point	挿入のエントリーポイント	オプション

JSON リクエストボディの例

```
{
  "commands": [ {
    "insert-elements": {
      "objects": [
        {
          "containedObject": {
            "@class": "org.drools.compiler.test.Person",
            "age": 25,
            "name": "john"
          }
        },
        {
          "containedObject": {
            "@class": "Person",
            "age": 35,
            "name": "sarah"
          }
        }
      ]
    }
  }
]
}
```

Java コマンドの例

```
List<Object> objects = new ArrayList<Object>();
objects.add(new Person("john", 25));
objects.add(new Person("sarah", 35));

Command insertElementsCommand = CommandFactory.newInsertElements(objects);
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
```

```
    "type": "SUCCESS",
    "msg": "Container command-script-container successfully called.",
    "result": {
      "execution-results": {
        "results": [],
        "facts": [
          {
            "value": {
              "org.drools.core.common.DefaultFactHandle": {
                "external-form": "0:4:436792766:-
2127720265:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap"
              }
            },
            "key": "john"
          },
          {
            "value": {
              "org.drools.core.common.DefaultFactHandle": {
                "external-form": "0:4:436792766:-
2127720266:4:DEFAULT:NON_TRAIT:java.util.LinkedHashMap"
              }
            },
            "key": "sarah"
          }
        ]
      }
    }
  }
 ]
}
```


FireAllRulesCommand

KIE セッションですべてのルールを実行します。

表4.8 コマンドの属性

名前	説明	要件
max	実行するルールの最大数。デフォルト値は **-1** で、実行に制限を課しません。	オプション
out-identifier	実行結果で、使用されたルール数の取得に使用される ID。	オプション
agenda-filter	ルール実行に使用するアジェンダフィルター。	オプション

JSON リクエストボディの例


```
{
  "commands" : [ {
    "fire-all-rules": {
      "max": 10,
      "out-identifier": "firedActivations"
```

```
    }
  } ]
}
```

Java コマンドの例

```
FireAllRulesCommand fireAllRulesCommand = new FireAllRulesCommand();
fireAllRulesCommand.setMax(10);
fireAllRulesCommand.setOutIdentifier("firedActivations");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [
            {
              "value": 0,
              "key": "firedActivations"
            }
          ],
          "facts": []
        }
      }
    }
  ]
}
```

QueryCommand

KIE ベースで定義されたクエリーを実行します。

表4.9 コマンドの属性

名前	説明	要件
name	クエリー名。	必須
out-identifier	クエリー結果の ID。クエリー結果を実行結果に追加する際に、この ID を使用します。	オプション
arguments	クエリーパラメーターとして渡されるオブジェクト一覧。	オプション

JSON リクエストボディの例

```
{
  "commands": [
```

```
    {
      "query": {
        "name": "persons",
        "arguments": [],
        "out-identifier": "persons"
      }
    }
  ]
}
```

Java コマンドの例

```
QueryCommand queryCommand = new QueryCommand();
queryCommand.setName("persons");
queryCommand.setOutIdentifier("persons");
```

サーバーの応答例 (JSON)

```
{
  "type": "SUCCESS",
  "msg": "Container stateful-session successfully called.",
  "result": {
    "execution-results": {
      "results": [
        {
          "value": {
            "org.drools.core.runtime.rule.impl.FlatQueryResults": {
              "idFactHandleMaps": {
                "type": "LIST",
                "componentType": null,
                "element": [
                  {
                    "type": "MAP",
                    "componentType": null,
                    "element": [
                      {
                        "value": {
                          "org.drools.core.common.DisconnectedFactHandle": {
                            "id": 1,
                            "identityHashCode": 1809949690,
                            "objectHashCode": 1809949690,
                            "recency": 1,
                            "object": {
                              "org.kie.server.testing.Person": {
                                "fullname": "John Doe",
                                "age": 47
                              }
                            },
                            "entryPointId": "DEFAULT",
                            "traitType": "NON_TRAIT",
                            "external-form":
"0:1:1809949690:1809949690:1:DEFAULT:NON_TRAIT:org.kie.server.testing.Person"
                          }
                        },
                        "key": "$person"
```

```
                  }
                ]
              }
            ]
          },
          "idResultMaps": {
            "type": "LIST",
            "componentType": null,
            "element": [
              {
                "type": "MAP",
                "componentType": null,
                "element": [
                  {
                    "value": {
                      "org.kie.server.testing.Person": {
                        "fullname": "John Doe",
                        "age": 47
                      }
                    },
                    "key": "$person"
                  }
                ]
              }
            ]
          },
          "identifiers": {
            "type": "SET",
            "componentType": null,
            "element": [
              "$person"
            ]
          }
        }
      },
      "key": "persons"
    }
  ],
  "facts": []
 }
}
}
```


SetGlobalCommand

オブジェクトをグローバルステートに設定します。

表4.10 コマンドの属性

名前	説明	要件
identifier	KIE ベースで定義されるグローバル変数の ID	必須
object	グローバル変数に設定されるオブジェクト	オプション

名前	説明	要件
out	設定したグローバル変数を実行結果から除外するブール値	オプション
out-identifier	グローバル実行結果の ID	オプション

JSON リクエストボディの例

```
{
  "commands": [
    {
      "set-global": {
        "identifier": "helper",
        "object": {
          "org.kie.server.testing.Person": {
            "fullname": "kyle",
            "age": 30
          }
        },
        "out-identifier": "output"
      }
    }
  ]
}
```

Java コマンドの例

```
SetGlobalCommand setGlobalCommand = new SetGlobalCommand();
setGlobalCommand.setIdentifier("helper");
setGlobalCommand.setObject(new Person("kyle", 30));
setGlobalCommand.setOut(true);
setGlobalCommand.setOutIdentifier("output");
```

サーバーの応答例 (JSON)

```
{
  "type": "SUCCESS",
  "msg": "Container stateful-session successfully called.",
  "result": {
    "execution-results": {
      "results": [
        {
          "value": {
            "org.kie.server.testing.Person": {
              "fullname": "kyle",
              "age": 30
            }
          },
          "key": "output"
        }
      ],
```

```
      "facts": []
    }
  }
}
```

GetGlobalCommand

以前に定義したグローバルオブジェクトを取得します。

表4.11 コマンドの属性

名前	説明	要件
identifier	KIE ベースで定義されるグローバル変数の ID	必須
out-identifier	実行結果で使用される ID	オプション

JSON リクエストボディの例

```
{
  "commands": [ {
    "get-global": {
      "identifier": "helper",
      "out-identifier": "helperOutput"
    }
  }
 ]
}
```

Java コマンドの例

```
GetGlobalCommand getGlobalCommand = new GetGlobalCommand();
getGlobalCommand.setIdentifier("helper");
getGlobalCommand.setOutIdentifier("helperOutput");
```

サーバーの応答例 (JSON)

```
{
  "response": [
    {
      "type": "SUCCESS",
      "msg": "Container command-script-container successfully called.",
      "result": {
        "execution-results": {
          "results": [
            {
              "value": null,
              "key": "helperOutput"
            }
          ],
          "facts": []
        }
      }
```

```
    }
  ]
}
```

第5章 DECISION SERVER 用の DECISION MANAGER コントローラー REST API テンプレートおよびインスタンス

Red Hat Decision Manager は Decision Manager コントローラー REST API を提供し、これを使用することで Business Central ユーザーインターフェースを使わずに Decision Server の テンプレート (設定) や Decision Server インスタンス (リモートサーバー)、関連する KIE コンテナー (デプロイメントユニット) を操作することができます。この API のサポートにより、Red Hat Decision Manager のサーバーとリソースをより効率的に維持でき、Red Hat Decision Manager の統合と開発を最適化できるようになります。

Decision Manager コントローラー REST API を使用すると、以下のアクションが可能になります。

- Decision Server テンプレート、インスタンス、および関連する KIE コンテナーについての情報の取得
- Decision Server テンプレートおよびインスタンスに関連付けられ KIE コンテナーの更新、起動、停止
- Decision Server テンプレートの作成、更新、削除
- Decision Server インスタンスの作成、更新、削除

Decision Manager コントローラー REST API への要求には、以下のコンポーネントが必要です。

認証

Decision Manager コントローラー REST API は、コントローラーのタイプによって、以下のユーザーロールに HTTP の Basic 認証またはトークンベースの認証を必要とします。

- Business Central をインストールしていて、ビルトインの Decision Manager コントローラーを使用する場合は、**rest-all** のユーザーロール。
- ヘッドレス Decision Manager コントローラーを Business Central とは別にインストールしている場合は、**kie-server** のユーザーロール。

お使いの Red Hat Decision Manager に設定されているユーザーロールを表示するには、**~/$SERVER_HOME/standalone/configuration/application-roles.properties** と **~/application-users.properties** に移動します。

ユーザーに **kie-server** ロールか **rest-all** ロール、もしくはそれら両方を追加するには、**~/$SERVER_HOME/bin** に移動し、ロールを指定して以下のコマンドを実行します。

```
$ ./add-user.sh -a --user <USERNAME> --password <PASSWORD> --role kie-server,rest-all
```

Decision Manager コントローラーのアクセスで **kie-server** または **rest-all** ユーザーを設定するには、**~/$SERVER_HOME/standalone/configuration/standalone-full.xml** を開き、 (該当する場合) **org.kie.server** プロパティーをコメント解除して、コントローラーユーザーログイン認証情報とコントローラーの位置を (必要に応じて) 追加します。

```
<property name="org.kie.server.location" value="http://localhost:8080/kie-
server/services/rest/server"/>
<property name="org.kie.server.controller" value="http://localhost:8080/decision-
central/rest/controller"/>
<property name="org.kie.server.controller.user" value="baAdmin"/>
<property name="org.kie.server.controller.pwd" value="password@1"/>
<property name="org.kie.server.id" value="default-kieserver"/>
```

ユーザーロールと Red Hat Decision Manager のインストールオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。

HTTP ヘッダー

Decision Manager コントローラー REST API は、API 要求に以下の HTTP ヘッダーを必要とします。

- **Accept**: 要求元のクライアントが受け付けるデータ形式:
  - **application/json** (JSON)
  - **application/xml** (XML, for JAXB)
- **Content-Type**: **POST** または **PUT** API 要求データ向けのデータ形式:
  - **application/json** (JSON)
  - **application/xml** (XML, for JAXB)

HTTP メソッド

Decision Manager コントローラー REST API は、API 要求に以下の HTTP メソッドをサポートしています。

- **GET**: 指定したリソースのエンドポイントから指定した情報を取得する
- **POST**: リソースまたはリソースインスタンスを更新する
- **PUT**: リソースまたはリソースインスタンスを作成します
- **DELETE**: リソースまたはリソースインスタンスを削除する

ベース URL

Decision Manager コントローラー REST API 要求のベース URL は **http://SERVER:PORT/CONTROLLER/rest/** で、Business Central のビルトイン Decision Manager コントローラーを使用している場合は **http://localhost:8080/decision-central/rest/** のようになります。

Endpoints (エンドポイント)

指定した Decision Server テンプレートにおける **/controller/management/servers/{serverTemplateId}** などの Decision Manager コントローラー REST API のエンドポイントは、Decision Manager コントローラー REST API のベース URL に追記する URI で、Red Hat Decision Manager の対応するサーバーリソースやサーバーリソースのタイプにアクセスするためのものです。

/spaces/{serverTemplateId} エンドポイントの要求 URL 例

http://localhost:8080/decision-central/rest/controller/management/servers/default-kieserver

要求パラメーターおよび要求データ

Decision Manager コントローラー REST API 要求のなかには、特定リソースを特定またはフィルタリングし、特定のアクションを実行するために、要求 URL パスで特定のパラメーターを必要とします。URL パラメーターは、**?<PARAM>=<VALUE>&<PARAM>=<VALUE>** の形式でエンドポイントに追記します。

DELETE 要求 URL のパラメーター例

http://localhost:8080/decision-central/rest/controller/server/new-kieserver-instance?location=http://localhost:8080/kie-server/services/rest/server

HTTP **POST** と **PUT** の要求は、さらに要求の本文もしくはデータのあるファイルが必要になる場合があります。

PUT 要求 URL と JSON 要求の本文データの例

http://localhost:8080/decision-central/rest/controller/management/servers/new-kieserver

```
{
  "server-id": "new-kieserver",
  "server-name": "new-kieserver",
  "container-specs": [],
  "server-config": {},
  "capabilities": [
    "RULE",
    "PROCESS",
    "PLANNING"
  ]
}
```

5.1. REST クライアントまたは CURL ユーティリティーを使用した DECISION MANAGER コントローラー REST API による要求送信

Decision Manager コントローラーは REST API を提供し、これを使用することで Business Central ユーザーインターフェースを使わずに Decision Server の テンプレート (設定) や Decision Server インスタンス (リモートサーバー)、関連する KIE コンテナー (デプロイメントユニット) を操作することができます。Decision Manager コントローラー REST API 要求は、REST クライアントや curl ユーティリティーを使って送信することができます。

前提条件

- Decision Server をインストールし、実行している。
- Decision Manager コントローラーもしくはヘッドレス Decision Manager コントローラーがインストールされ、実行している。
- Business Central をインストールしている場合は Decision Manager コントローラーにアクセスする **rest-all** ユーザーロールがあること。もしくは、Business Central とは別にインストールされたヘッドレス Decision Manager コントローラーにアクセスする **kie-server** ユーザーロールがあること。

手順

1. **[GET] /controller/management/servers** など、要求の送信先となるに適した API エンドポイント を特定し、Decision Manager コントローラーから Decision Server テンプレートを取得します。
2. REST クライアントまたは curl ユーティリティーで、**controller/management/servers** への **GET** 要求に以下のコンポーネントを記入します。ご自分のユースケースに合わせて、要求詳細を調整します。
   REST クライアントの場合:

- **Authentication**: **rest-all** ロールのある Decision Manager コントローラーユーザーまたは **kie-server** ロールを持つヘッドレス Decision Manager コントローラーユーザーのユーザー名とパスワードを入力します。
- **HTTP Headers**: 以下のヘッダーを設定します。
  - **Accept**: **application/json**
- **HTTP method**: **GET** に設定します。
- **URL**: Decision Manager コントローラー REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/controller/management/servers** となります。

curl ユーティリティーの場合:

- **-u**: **rest-all** ロールのある Decision Manager コントローラーユーザーまたは **kie-server** ロールを持つヘッドレス Decision Manager コントローラーユーザーのユーザー名とパスワードを入力します。
- **-H**: 以下のヘッダーを設定します。
  - **accept**: **application/json**
- **-X**: **GET** に設定します。
- **URL**: Decision Manager コントローラー REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/controller/management/servers** となります。

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -X GET
"http://localhost:8080/decision-central/rest/controller/management/servers"
```

3. 要求を実行し、Decision Manager コントローラーの応答を確認します。
サーバー応答の例 (JSON):


```
{
  "server-template": [
    {
      "server-id": "default-kieserver",
      "server-name": "default-kieserver",
      "container-specs": [
        {
          "container-id": "employeerostering_1.0.0-SNAPSHOT",
          "container-name": "employeerostering",
          "server-template-key": {
            "server-id": "default-kieserver",
            "server-name": "default-kieserver"
          },
          "release-id": {
            "group-id": "employeerostering",
            "artifact-id": "employeerostering",
            "version": "1.0.0-SNAPSHOT"
          },
          "configuration": {
            "RULE": {
```

```
          "org.kie.server.controller.api.model.spec.RuleConfig": {
            "pollInterval": null,
            "scannerStatus": "STOPPED"
          }
        },
        "PROCESS": {
          "org.kie.server.controller.api.model.spec.ProcessConfig": {
            "runtimeStrategy": "SINGLETON",
            "kbase": "",
            "ksession": "",
            "mergeMode": "MERGE_COLLECTIONS"
          }
        }
      },
      "status": "STARTED"
    },
    {
      "container-id": "mortgage-process_1.0.0-SNAPSHOT",
      "container-name": "mortgage-process",
      "server-template-key": {
        "server-id": "default-kieserver",
        "server-name": "default-kieserver"
      },
      "release-id": {
        "group-id": "mortgage-process",
        "artifact-id": "mortgage-process",
        "version": "1.0.0-SNAPSHOT"
      },
      "configuration": {
        "RULE": {
          "org.kie.server.controller.api.model.spec.RuleConfig": {
            "pollInterval": null,
            "scannerStatus": "STOPPED"
          }
        },
        "PROCESS": {
          "org.kie.server.controller.api.model.spec.ProcessConfig": {
            "runtimeStrategy": "PER_PROCESS_INSTANCE",
            "kbase": "",
            "ksession": "",
            "mergeMode": "MERGE_COLLECTIONS"
          }
        }
      },
      "status": "STARTED"
    }
  ],
  "server-config": {},
  "server-instances": [
    {
      "server-instance-id": "default-kieserver-instance@localhost:8080",
      "server-name": "default-kieserver-instance@localhost:8080",
      "server-template-id": "default-kieserver",
      "server-url": "http://localhost:8080/kie-server/services/rest/server"
    }
  ],
```

```
      "capabilities": [
        "RULE",
        "PROCESS",
        "PLANNING"
      ]
    }
  ]
}
```

4. REST クライアントまたは curl ユーティリティーで、**/controller/management/servers/{serverTemplateId}** への **PUT** 要求を以下のコンポーネントで送信し、新規の Decision Server テンプレートを作成します。ご自分のユースケースに合わせて、要求詳細を調整します。
REST クライアントの場合:

   - **Authentication**: **rest-all** ロールのある Decision Manager コントローラーユーザーまたは **kie-server** ロールを持つヘッドレス Decision Manager コントローラーユーザーのユーザー名とパスワードを入力します。
   - **HTTP Headers**: 以下のヘッダーを設定します。
     - **Accept**: **application/json**
     - **Content-Type**: **application/json**
   - **HTTP method**: **PUT** に設定します。
   - **URL**: Decision Manager コントローラー REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/controller/management/servers/new-kieserver** となります。
   - **要求の本文**: 新規 Decision Server テンプレート用の設定がある JSON 要求の本文を追加します。

```
{
  "server-id": "new-kieserver",
  "server-name": "new-kieserver",
  "container-specs": [],
  "server-config": {},
  "capabilities": [
    "RULE",
    "PROCESS",
    "PLANNING"
  ]
}
```

   curl ユーティリティーの場合:

   - **-u**: **rest-all** ロールのある Decision Manager コントローラーユーザーまたは **kie-server** ロールを持つヘッドレス Decision Manager コントローラーユーザーのユーザー名とパスワードを入力します。
   - **-H**: 以下のヘッダーを設定します。
     - **accept**: **application/json**

    - **content-type**: **application/json**

- **-X**: **PUT** に設定します。
- **URL**: Decision Manager コントローラー REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/controller/management/servers/new-kieserver** となります。
- **-d**: 新規 Decision Server テンプレート用の設定がある JSON 要求の本文またはファイル (**@file.json**) を追加します。

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type: application/json" -X PUT "http://localhost:8080/decision-central/rest/controller/management/servers/new-kieserver" -d "{ \"server-id\": \"new-kieserver\", \"server-name\": \"new-kieserver\", \"container-specs\": [], \"server-config\": {}, \"capabilities\": [ \"RULE\", \"PROCESS\", \"PLANNING\" ]}"
```

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type: application/json" -X PUT "http://localhost:8080/decision-central/rest/controller/management/servers/new-kieserver" -d @my-server-template-configs.json
```

5. 要求を実行し、Decision Manager コントローラーの応答が正常であることを確認します。エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて要求を調整します。

5.2. SWAGGER インターフェースを使用した DECISION MANAGER コントローラー REST API による要求送信

Decision Manager コントローラー REST API は Swagger web インターフェースをサポートしています。スタンドアロンの REST クライアントや curl ユーティリティーの代わりのこれを使用すると、Business Central ユーザーインターフェースを使わずに Decision Server の テンプレート、インスタンス、関連する KIE コンテナーを Red Hat Decision Manager で操作することができます。

注記

デフォルトでは、**org.kie.workbench.swagger.disabled=false** のシステムプロパティーが指定されており、Decision Manager の Swagger Web インターフェースが有効になっています。Decision Manager で Swagger Web インターフェースを無効にするには、このシステムプロパティーを **true** に設定してください。

前提条件

- Decision Manager コントローラーがインストール済みで実行されている。
- Business Central をインストールしている場合は Decision Manager コントローラーにアクセスする **rest-all** ユーザーロールがあること。もしくは、Business Central とは別にインストールされたヘッドレス Decision Manager コントローラーにアクセスする **kie-server** ユーザーロールがあること。

手順

1. web ブラウザーで **http://SERVER:PORT/CONTROLLER/docs** に移動します。たとえば、**http://localhost:8080/decision-central/docs** などです。**rest-all** ロールのある Decision

Manager コントローラーユーザーまたは **kie-server** ロールを持つヘッドレス Decision Manager コントローラーユーザーのユーザー名とパスワードでログインします。

注記

Business Central に組み込まれている Decision Manager コントローラーを使用している場合、Decision Manager コントローラーに関連付けられている Swagger ページは、Business Central REST サービスでは「Business Central API」と識別されます。Business Central なしでヘッドレス Decision Manager コントローラーを使用している場合は、ヘッドレス Decision Manager コントローラーに関連付けられている Swagger ページは、「Controller API」と識別されます。いずれの場合も、Decision Manager コントローラー REST API エンドポイントは、同じです。

2. Swagger ページで、要求の送信先となる関連 API エンドポイントを選択します。たとえば、**Controller :: KIE Server templates and KIE containers↪ [GET] /controller/management/servers** で Decision Server テンプレートを Decision Manager コントローラーから取得します。

3. 該当する場合は **Try it out** をクリックして、結果のフィルタリングに使用するオプションのパラメーターを提供します。

4. **Response content type** ドロップダウンメニューで、サーバー応答のフォーマットを選択します。たとえば、JSON フォーマットでは **application/json** となります。

5. **Execute** をクリックし、Decision Server の応答を確認します。
サーバー応答の例 (JSON):


```
{
  "server-template": [
    {
      "server-id": "default-kieserver",
      "server-name": "default-kieserver",
      "container-specs": [
        {
          "container-id": "employeerostering_1.0.0-SNAPSHOT",
          "container-name": "employeerostering",
          "server-template-key": {
            "server-id": "default-kieserver",
            "server-name": "default-kieserver"
          },
          "release-id": {
            "group-id": "employeerostering",
            "artifact-id": "employeerostering",
            "version": "1.0.0-SNAPSHOT"
          },
          "configuration": {
            "RULE": {
              "org.kie.server.controller.api.model.spec.RuleConfig": {
                "pollInterval": null,
                "scannerStatus": "STOPPED"
              }
            },
            "PROCESS": {
              "org.kie.server.controller.api.model.spec.ProcessConfig": {
```

```
            "runtimeStrategy": "SINGLETON",
            "kbase": "",
            "ksession": "",
            "mergeMode": "MERGE_COLLECTIONS"
          }
        }
      },
      "status": "STARTED"
    },
    {
      "container-id": "mortgage-process_1.0.0-SNAPSHOT",
      "container-name": "mortgage-process",
      "server-template-key": {
        "server-id": "default-kieserver",
        "server-name": "default-kieserver"
      },
      "release-id": {
        "group-id": "mortgage-process",
        "artifact-id": "mortgage-process",
        "version": "1.0.0-SNAPSHOT"
      },
      "configuration": {
        "RULE": {
          "org.kie.server.controller.api.model.spec.RuleConfig": {
            "pollInterval": null,
            "scannerStatus": "STOPPED"
          }
        },
        "PROCESS": {
          "org.kie.server.controller.api.model.spec.ProcessConfig": {
            "runtimeStrategy": "PER_PROCESS_INSTANCE",
            "kbase": "",
            "ksession": "",
            "mergeMode": "MERGE_COLLECTIONS"
          }
        }
      },
      "status": "STARTED"
    }
  ],
  "server-config": {},
  "server-instances": [
    {
      "server-instance-id": "default-kieserver-instance@localhost:8080",
      "server-name": "default-kieserver-instance@localhost:8080",
      "server-template-id": "default-kieserver",
      "server-url": "http://localhost:8080/kie-server/services/rest/server"
    }
  ],
  "capabilities": [
    "RULE",
    "PROCESS",
    "PLANNING"
  ]
```

```
    }
  ]
}
```

6. Swagger ページで **Controller :: KIE Server templates and KIE containers↦ [GET] /controller/management/servers/{serverTemplateId}** エンドポイントに移動し、新たな Decision Server テンプレートを作成するための別の要求を送信します。ご自分のユースケースに合わせて、要求詳細を調整します。

7. **Try it out** をクリックして、以下の要求のコンポーネントを入力します。

   - **serverTemplateId**: 新規 Decision Server テンプレートの ID を入力します。例: **new-kieserver**

   - **body**: **Parameter content type** を任意の要求本文形式 (JSON の場合は **application/json** など) に設定し、要求の本文に新規 Decision Server テンプレートの設定を追加します。

```
{
  "server-id": "new-kieserver",
  "server-name": "new-kieserver",
  "container-specs": [],
  "server-config": {},
  "capabilities": [
    "RULE",
    "PROCESS",
    "PLANNING"
  ]
}
```

8. **Response content type** ドロップダウンメニューで、サーバー応答のフォーマットを選択します。たとえば、JSON フォーマットでは **application/json** となります。

9. **Execute** をクリックし、Decision Manager コントローラーの応答が正常であることを確認します。
   エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて要求を調整します。

5.3. サポート対象の DECISION MANAGER コントローラー REST API エンドポイント

Decision Manager コントローラー REST API には、Decision Server テンプレート (設定)、Decision Server インスタンス (リモートサーバー)、関連の KIE コンテナー (デプロイメントユニット) を操作するエンドポイントが含まれます。Decision Manager コントローラー REST API のベース URL は、**http://SERVER:PORT/CONTROLLER/rest/** です。すべての要求では、Business Central がインストールされており、ビルトインの Decision Manager コントローラーを使用する場合には、**rest-all** ユーザーロールの HTTP Basic 認証またはトークベースの認証が必要で、Business Central とは別にヘッドレス Decision Manager コントローラーをインストール済みの場合には、**kie-server** ユーザーロールの HTTP Basic 認証またはトークンベースの認証が必要です。

Decision Manager コントローラー REST API エンドポイントの完全一覧と説明については、以下のリソースを参照してください。

- jBPM ドキュメントページ (静的) の Controller REST API

- **http://SERVER:PORT/CONTROLLER/docs** (動的。稼働中の Decision Manager コントローラーが必要) ページの Decision Manager コントローラー REST API 用 Swagger UI

注記

デフォルトでは、**org.kie.workbench.swagger.disabled=false** のシステムプロパティーが指定されており、Decision Manager の Swagger Web インターフェースが有効になっています。Decision Manager で Swagger Web インターフェースを無効にするには、このシステムプロパティーを **true** に設定してください。

Business Central に組み込まれている Decision Manager コントローラーを使用している場合、Decision Manager コントローラーに関連付けられている Swagger ページは、Business Central REST サービスでは「Business Central API」と識別されます。Business Central なしでヘッドレス Decision Manager コントローラーを使用している場合は、ヘッドレス Decision Manager コントローラーに関連付けられている Swagger ページは、「Controller API」と識別されます。いずれの場合も、Decision Manager コントローラー REST API エンドポイントは、同じです。

第6章 DECISION SERVER テンプレートおよびインスタンス用の DECISION MANAGER コントローラー JAVA クライアント API

Red Hat Decision Manager は Decision Manager コントローラー Java クライアント API を提供し、これを使用することで Java クライアントアプリケーションから REST もしくは WebSocket プロトコルを使って Decision Manager コントローラー に接続できるようになります。Decision Manager コントローラー REST API の代わりに Decision Manager コントローラー Java クライアント API を使って、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager で Decision Server の インスタンス (リモートサーバー)、関連する KIE コンテナー (デプロイメントユニット) を操作することができます。この API のサポートにより、Red Hat Decision Manager サーバーのリソースをより効率的に維持でき、Red Hat Decision Manager の統合と開発を最適化できるようになります。

Decision Manager コントローラー Java クライアント API を使用すると、Decision Manager コントローラー REST API でもサポートされている以下のアクションが実行可能になります。

- Decision Server テンプレート、インスタンス、および関連する KIE コンテナーについての情報の取得
- Decision Server テンプレートおよびインスタンスに関連付けられ KIE コンテナーの更新、起動、停止
- Decision Server テンプレートの作成、更新、削除
- Decision Server インスタンスの作成、更新、削除

Decision Manager コントローラー Java クライアント API 要求には以下のコンポーネントが必要です。

認証

Decision Manager コントローラー Java クライアント API は、コントローラーのタイプによって、以下のユーザーロールに HTTP の Basic 認証を必要とします。

- Business Central をインストールしていて、ビルトインの Decision Manager コントローラーを使用する場合は、**rest-all** のユーザーロール。
- ヘッドレス Decision Manager コントローラーを Business Central とは別にインストールしている場合は、**kie-server** のユーザーロール。

お使いの Red Hat Decision Manager に設定されているユーザーロールを表示するには、**~/$SERVER_HOME/standalone/configuration/application-roles.properties** と **~/application-users.properties** に移動します。

ユーザーに **kie-server** ロールか **rest-all** ロール、もしくはそれら両方を追加するには、**~/$SERVER_HOME/bin** に移動し、ロールを指定して以下のコマンドを実行します。

```
$ ./add-user.sh -a --user <USERNAME> --password <PASSWORD> --role kie-server,rest-all
```

Decision Manager コントローラーのアクセスで **kie-server** または **rest-all** ユーザーを設定するには、**~/$SERVER_HOME/standalone/configuration/standalone-full.xml** を開き、 (該当する場合) **org.kie.server** プロパティーをコメント解除して、コントローラーユーザーログイン認証情報とコントローラーの位置を (必要に応じて) 追加します。

```
<property name="org.kie.server.location" value="http://localhost:8080/kie-server/services/rest/server"/>
<property name="org.kie.server.controller" value="http://localhost:8080/decision-central/rest/controller"/>
```

```
<property name="org.kie.server.controller.user" value="baAdmin"/>
<property name="org.kie.server.controller.pwd" value="password@1"/>
<property name="org.kie.server.id" value="default-kieserver"/>
```

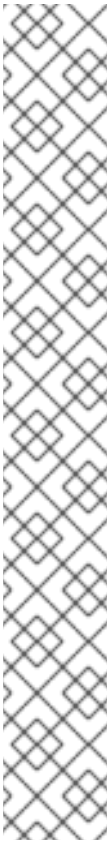
ユーザーロールと Red Hat Decision Manager のインストールオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。

プロジェクトの依存関係

DecisionManager コントローラー Java クライアント API には、Java プロジェクト内の適切なクラスパスに、以下の依存関係を必要です。

```
<!-- For remote execution on controller -->
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-controller-client</artifactId>
  <version>${rhdm.version}</version>
</dependency>

<!-- For REST client -->
<dependency>
  <groupId>org.jboss.resteasy</groupId>
  <artifactId>resteasy-client</artifactId>
  <version>${resteasy.version}</version>
</dependency>

<!-- For WebSocket client -->
<dependency>
  <groupId>io.undertow</groupId>
  <artifactId>undertow-websockets-jsr</artifactId>
  <version>${undertow.version}</version>
</dependency>

<!-- For debug logging (optional) -->
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
</dependency>
```

Red Hat Decision Manager 依存関係の **<version>** は、プロジェクトで現在使用する Red Hat Decision Manager の Maven アーティファクトバージョンです (例: 7.23.0.Final-redhat-00002)。



注記

個別の依存関係に対して Red Hat Decision Manager **<version>** を指定するのではなく、Red Hat Business Automation 部品表 (BOM) の依存関係をプロジェクトの **pom.xml** ファイルに追加することを検討してください。Red Hat Business Automation BOM は、Red Hat Decision Manager と Red Hat Process Automation Manager の両方に適用します。BOM ファイルを追加すると、指定の Maven リポジトリーからの一時的な依存関係の内、正しいバージョンが、このプロジェクトに追加されます。

BOM 依存関係の例:

```
<dependency>
  <groupId>com.redhat.ba</groupId>
  <artifactId>ba-platform-bom</artifactId>
  <version>7.4.0.GA-redhat-00002</version>
  <scope>import</scope>
  <type>pom</type>
</dependency>
```

Red Hat Business Automation BOM (Bill of Materials) についての詳細情報は、「What is the mapping between RHDM product and maven library version? 」を参照してください。

クライアント要求の設定

Decision Manager コントローラー Java クライアント API による Java クライアント要求はすべて、少なくとも以下のコントローラー通信コンポーネントを定義する必要があります。

- Business Central をインストールしている場合は **rest-all** ユーザーの認証情報、またはヘッドレス Decision Manager コントローラーを Business Central とは別にインストールしている場合は **kie-server** のユーザーの認証情報。
- REST もしくは WebSocket プロトコル用 Decision Manager コントローラー の場所。
  - REST URL の例: **http://localhost:8080/decision-central/rest/controller**
  - WebSocket URL の例: **ws://localhost:8080/headless-controller/websocket/controller**
- API 要求および応答のマーシャリングフォーマット (JSON または JAXB)
- **KieServerControllerClient** オブジェクト。これは、Java クライアント API を使用してサーバー通信を開始するためのエントリーポイントの役割を果たします。
- REST プロトコルまたは WebSocket プロトコルおよびユーザーアクセスを定義する **KieServerControllerClientFactory**。
- Decision Manager コントローラークライアントサービス。たとえば、**listServerTemplates**、**getServerTemplate**、**getServerInstances** など。

以下は、これらのコンポーネントを使用した REST および WebSocket クライアントの設定例です。

REST によるクライアント設定例

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.controller.api.model.spec.ServerTemplateList;
```

```
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;

public class ListServerTemplatesExample {

    private static final String URL = "http://localhost:8080/decision-central/rest/controller";
    private static final String USER = "baAdmin";
    private static final String PASSWORD = "password@1";

    private static final MarshallingFormat FORMAT = MarshallingFormat.JSON;

    public static void main(String[] args) {
        KieServerControllerClient client = KieServerControllerClientFactory.newRestClient(URL,
                                                                                           USER,
                                                                                           PASSWORD);

        final ServerTemplateList serverTemplateList = client.listServerTemplates();
        System.out.println(String.format("Found %s server template(s) at controller url: %s",
                                         serverTemplateList.getServerTemplates().length,
                                         URL));
    }
}
```

WebSocket によるクライアント設定例

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.controller.api.model.spec.ServerTemplateList;
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;

public class ListServerTemplatesExample {

    private static final String URL = "ws://localhost:8080/my-controller/websocket/controller";
    private static final String USER = "baAdmin";
    private static final String PASSWORD = "password@1";

    private static final MarshallingFormat FORMAT = MarshallingFormat.JSON;

    public static void main(String[] args) {
        KieServerControllerClient client =
KieServerControllerClientFactory.newWebSocketClient(URL,
                                                                                           USER,
                                                                                           PASSWORD);

        final ServerTemplateList serverTemplateList = client.listServerTemplates();
        System.out.println(String.format("Found %s server template(s) at controller url: %s",
                                         serverTemplateList.getServerTemplates().length,
                                         URL));
    }
}
```

6.1. DECISION MANAGER コントローラー JAVA クライアント API を使った要求送信

Decision Manager コントローラー Java クライアント API を使用すると、Java クライアントアプリケーションから REST もしくは WebSocket プロトコルを使って Decision Manager コントローラー に接続できるようになります。Decision Manager コントローラー REST API の代わりに Decision Manager コントローラー Java クライアント API を使って、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager で Decision Server の インスタンス (リモートサーバー)、関連する KIE コンテナー (デプロイメントユニット) を操作することができます。

前提条件

- Decision Server をインストールし、実行している。
- Decision Manager コントローラーもしくはヘッドレス Decision Manager コントローラーがインストールされ、実行している。
- Business Central をインストールしている場合は Decision Manager コントローラーにアクセスする **rest-all** ユーザーロールがあること。もしくは、Business Central とは別にインストールされたヘッドレス Decision Manager コントローラーにアクセスする **kie-server** ユーザーロールがあること。
- Red Hat Decision Manager リソースを使った Java プロジェクトがある。

手順

1. クライアントアプリケーションで、Java プロジェクト内の適切なクラスパスに以下の依存関係が追加されていることを確認します。

```
<!-- For remote execution on controller -->
<dependency>
  <groupId>org.kie.server</groupId>
  <artifactId>kie-server-controller-client</artifactId>
  <version>${rhdm.version}</version>
</dependency>

<!-- For REST client -->
<dependency>
  <groupId>org.jboss.resteasy</groupId>
  <artifactId>resteasy-client</artifactId>
  <version>${resteasy.version}</version>
</dependency>

<!-- For WebSocket client -->
<dependency>
  <groupId>io.undertow</groupId>
  <artifactId>undertow-websockets-jsr</artifactId>
  <version>${undertow.version}</version>
</dependency>

<!-- For debug logging (optional) -->
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>${logback.version}</version>
</dependency>
```

2. Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution**をダ

ウンロードし、~/**rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-controller/kie-server-controller-client/src/main/java/org/kie/server/controller/client** に移動して Decision Manager コントローラー Java クライアントにアクセスします。

3. ~/**kie/server/controller/client** ディレクトリーで、Decision Server テンプレートや KIE コンテナーが REST プロトコルでクライアントサービスにアクセスするために、**RestKieServerControllerClient** 実装などの送信する要求用の関連 Java クライアントを特定します。

4. クライアントアプリケーションで、API 要求用の **.java** クラスを作成します。クラスには、Decision Manager コントローラーの場所とユーザー認証情報、**KieServerControllerClient** オブジェクト、**RestKieServerControllerClient** 実装からの **createServerTemplate** や **createContainer** などの実行するクライアントメソッドを含める必要があります。ご自分のユースケースに合わせて、設定詳細を調整します。

Decision Server テンプレートおよび KIE コンテナーの作成と対話

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.api.model.KieContainerStatus;
import org.kie.server.api.model.KieScannerStatus;
import org.kie.server.api.model.ReleaseId;
import org.kie.server.controller.api.model.spec.*;
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;

public class RestTemplateContainerExample {

  private static final String URL = "http://localhost:8080/decision-central/rest/controller";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

    private static KieServerControllerClient client;

    public static void main(String[] args) {
        KieServerControllerClient client = KieServerControllerClientFactory.newRestClient(URL,
                                                                                           USER,
                                                                                           PASSWORD,
                                                                                           MarshallingFormat.JSON);
        // Create server template and KIE container, start and stop KIE container, and delete
server template
        ServerTemplate serverTemplate = createServerTemplate();
        ContainerSpec container = createContainer(serverTemplate);
        client.startContainer(container);
        client.stopContainer(container);
        client.deleteServerTemplate(serverTemplate.getId());
    }

    // Re-create and configure server template
    protected static ServerTemplate createServerTemplate() {
        ServerTemplate serverTemplate = new ServerTemplate();
        serverTemplate.setId("example-client-id");
```

```
        serverTemplate.setName("example-client-name");
        serverTemplate.setCapabilities(Arrays.asList(Capability.PROCESS.name(),
                                                     Capability.RULE.name(),
                                                     Capability.PLANNING.name()));

        client.saveServerTemplate(serverTemplate);

        return serverTemplate;
    }

    // Re-create and configure KIE containers
    protected static ContainerSpec createContainer(ServerTemplate serverTemplate){
        Map<Capability, ContainerConfig> containerConfigMap = new HashMap();

        ProcessConfig processConfig = new ProcessConfig("PER_PROCESS_INSTANCE",
"kieBase", "kieSession", "MERGE_COLLECTION");
        containerConfigMap.put(Capability.PROCESS, processConfig);

        RuleConfig ruleConfig = new RuleConfig(500l, KieScannerStatus.SCANNING);
        containerConfigMap.put(Capability.RULE, ruleConfig);

        ReleaseId releaseId = new ReleaseId("org.kie.server.testing", "stateless-session-kjar",
"1.0.0-SNAPSHOT");

        ContainerSpec containerSpec = new ContainerSpec("example-container-id", "example-
client-name", serverTemplate, releaseId, KieContainerStatus.STOPPED,
containerConfigMap);
        client.saveContainerSpec(serverTemplate.getId(), containerSpec);

        return containerSpec;
    }
}
```

5. 設定済み **.java** クラスをプロジェクトディレクトリーから実行して、要求を出し、Decision Manager コントローラーの応答を確認します。
デバッグのロギングが有効になっている場合は、JSON などの設定済みマーシャリングフォーマットに従って、Decision Server が詳細を返します。エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて Java 設定を調整します。

6.2. サポート対象の DECISION MANAGER コントローラー JAVA クライアント

以下は、Red Hat Decision Manager の **org.kie.server.controller.client** パッケージで利用可能な Java クライアントサービスの一部です。これらのサービスを使用して、Decision Manager コントローラー REST API と同様に Decision Manager コントローラー の関連リソースを操作できます。

- **KieServerControllerClient**: Decision Manager コントローラーとの通信のエントリーポイントとして使用します。
- **RestKieServerControllerClient**: REST プロトコルでの Decision Server テンプレートと KIE コンテナーとの対話に使用される実装 (~/**org**/**kie**/**server**/**controller**/**client**/**rest** に格納)。
- **WebSocketKieServerControllerClient**: WebSocket プロトコルでの Decision Server テンプレートと KIE コンテナーとの対話に使用される実装 (~/**org**/**kie**/**server**/**controller**/**client**/**websocket** に格納)。

利用可能な Decision Manager コントローラー Java クライアントの完全一覧については、Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution**をダウンロードして、**~/rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-controller/kie-server-controller-client/src/main/java/org/kie/server/controller/client** に移動してください。

6.3. DECISION MANAGER コントローラー JAVA クライアント API を使った要求例

以下は、Decision Manager コントローラー との基本的な対話のための Decision Manager コントローラー Java クライアント API 要求の例です。利用可能な Decision Manager コントローラー Java クライアントの完全一覧については、Red Hat カスタマーポータル から **Red Hat Decision Manager 7.4.0 Source Distribution** をダウンロードして、**~/rhdm-7.4.0-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-controller/kie-server-controller-client/src/main/java/org/kie/server/controller/client** に移動してください。

Decision Server テンプレートおよび KIE コンテナーの作成と対話

REST または WebSocket の Decision Manager コントローラークライアントで **ServerTemplate** および **ContainerSpec** サービスを使用すると、Decision Server テンプレートと KIE コンテナーの作成、破棄、更新が可能で、さらに KIE コンテナーの起動と停止もできます。以下に例を示します。

Decision Server テンプレートおよび KIE コンテナーによる作成と対話要求の例

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.api.model.KieContainerStatus;
import org.kie.server.api.model.KieScannerStatus;
import org.kie.server.api.model.ReleaseId;
import org.kie.server.controller.api.model.spec.*;
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;

public class RestTemplateContainerExample {

  private static final String URL = "http://localhost:8080/decision-central/rest/controller";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

    private static KieServerControllerClient client;

    public static void main(String[] args) {
        KieServerControllerClient client = KieServerControllerClientFactory.newRestClient(URL,
                                                                USER,
                                                                PASSWORD,
                                                                MarshallingFormat.JSON);
        // Create server template and KIE container, start and stop KIE container, and delete server
template
        ServerTemplate serverTemplate = createServerTemplate();
        ContainerSpec container = createContainer(serverTemplate);
        client.startContainer(container);
        client.stopContainer(container);
        client.deleteServerTemplate(serverTemplate.getId());
```

```
    }

    // Re-create and configure server template
    protected static ServerTemplate createServerTemplate() {
        ServerTemplate serverTemplate = new ServerTemplate();
        serverTemplate.setId("example-client-id");
        serverTemplate.setName("example-client-name");
        serverTemplate.setCapabilities(Arrays.asList(Capability.PROCESS.name(),
                                       Capability.RULE.name(),
                                       Capability.PLANNING.name()));

        client.saveServerTemplate(serverTemplate);

        return serverTemplate;
    }

    // Re-create and configure KIE containers
    protected static ContainerSpec createContainer(ServerTemplate serverTemplate){
        Map<Capability, ContainerConfig> containerConfigMap = new HashMap();

        ProcessConfig processConfig = new ProcessConfig("PER_PROCESS_INSTANCE",
"kieBase", "kieSession", "MERGE_COLLECTION");
        containerConfigMap.put(Capability.PROCESS, processConfig);

        RuleConfig ruleConfig = new RuleConfig(500l, KieScannerStatus.SCANNING);
        containerConfigMap.put(Capability.RULE, ruleConfig);

        ReleaseId releaseId = new ReleaseId("org.kie.server.testing", "stateless-session-kjar",
"1.0.0-SNAPSHOT");

        ContainerSpec containerSpec = new ContainerSpec("example-container-id", "example-client-
name", serverTemplate, releaseId, KieContainerStatus.STOPPED, containerConfigMap);
        client.saveContainerSpec(serverTemplate.getId(), containerSpec);

        return containerSpec;
    }
}
```

Decision Server テンプレートの一覧および接続タイムアウトの指定 (REST)

Decision Manager コントローラー Java クライアント API 要求に REST プロトコルを使用すると、独自の **javax.ws.rs.core.Configuration** 仕様で接続タイムアウトなどの基本的な REST クライアント API を変更することができます。

サーバーテンプレートを返し、接続タイムアウトを指定する REST 要求の例

```
import java.util.concurrent.TimeUnit;
import javax.ws.rs.core.Configuration;
import org.jboss.resteasy.client.jaxrs.ResteasyClientBuilder;

import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.controller.api.model.spec.ServerTemplateList;
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;

public class RESTTimeoutExample {
```

```
  private static final String URL = "http://localhost:8080/decision-central/rest/controller";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

  public static void main(String[] args) {

    // Specify connection timeout
    final Configuration configuration =
        new ResteasyClientBuilder()
            .establishConnectionTimeout(10,
                                        TimeUnit.SECONDS)
            .socketTimeout(60,
                           TimeUnit.SECONDS)
             .getConfiguration();
    KieServerControllerClient client = KieServerControllerClientFactory.newRestClient(URL,
                                                                                       USER,
                                                                                       PASSWORD,
                                                                                       MarshallingFormat.JSON,
                                                                                       configuration);

    // Retrieve list of server templates
    final ServerTemplateList serverTemplateList = client.listServerTemplates();
    System.out.println(String.format("Found %s server template(s) at controller url: %s",
                                     serverTemplateList.getServerTemplates().length,
                                     URL));
  }
}
```

Decision Server テンプレートの一覧およびイベント通知の指定 (WebSocket)

Decision Manager コントローラー Java クライアント API 要求に WebSocket プロトコルを使用すると、クライアント API の接続先である特定の Decision Manager コントローラーで発生した変更に基づいてイベントが通知されるようにすることができます。たとえば、Decision Server テンプレートもしくはインスタンスが接続された、または Decision Manager コントローラー内で更新されると、通知を受け取ることができます。

サーバーテンプレートを返し、イベント通知を指定する WebSocket 要求の例

```
import org.kie.server.api.marshalling.MarshallingFormat;
import org.kie.server.controller.api.model.events.*;
import org.kie.server.controller.api.model.spec.ServerTemplateList;
import org.kie.server.controller.client.KieServerControllerClient;
import org.kie.server.controller.client.KieServerControllerClientFactory;
import org.kie.server.controller.client.event.EventHandler;

public class WebSocketEventsExample {

  private static final String URL = "ws://localhost:8080/my-controller/websocket/controller";
  private static final String USER = "baAdmin";
  private static final String PASSWORD = "password@1";

  public static void main(String[] args) {
    KieServerControllerClient client =
KieServerControllerClientFactory.newWebSocketClient(URL,
                                                    USER,
```

```
                                                    PASSWORD,
                                                    MarshallingFormat.JSON,
                                                    new TestEventHandler());

        // Retrieve list of server templates
        final ServerTemplateList serverTemplateList = client.listServerTemplates();
        System.out.println(String.format("Found %s server template(s) at controller url: %s",
                                         serverTemplateList.getServerTemplates().length,
                                         URL));
        try {
            Thread.sleep(60 * 1000);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Set up event notifications
    static class TestEventHandler implements EventHandler {

        @Override
        public void onServerInstanceConnected(ServerInstanceConnected
serverInstanceConnected) {
            System.out.println("serverInstanceConnected = " + serverInstanceConnected);
        }

        @Override
        public void onServerInstanceDeleted(ServerInstanceDeleted serverInstanceDeleted) {
            System.out.println("serverInstanceDeleted = " + serverInstanceDeleted);
        }

        @Override
        public void onServerInstanceDisconnected(ServerInstanceDisconnected
serverInstanceDisconnected) {
            System.out.println("serverInstanceDisconnected = " + serverInstanceDisconnected);
        }

        @Override
        public void onServerTemplateDeleted(ServerTemplateDeleted serverTemplateDeleted) {
            System.out.println("serverTemplateDeleted = " + serverTemplateDeleted);
        }

        @Override
        public void onServerTemplateUpdated(ServerTemplateUpdated serverTemplateUpdated) {
            System.out.println("serverTemplateUpdated = " + serverTemplateUpdated);
        }

        @Override
        public void onServerInstanceUpdated(ServerInstanceUpdated serverInstanceUpdated) {
            System.out.println("serverInstanceUpdated = " + serverInstanceUpdated);
        }

        @Override
        public void onContainerSpecUpdated(ContainerSpecUpdated containerSpecUpdated) {
            System.out.println("onContainerSpecUpdated = " + containerSpecUpdated);
```

```
        }
    }
}
```

第7章 BUSINESS CENTRAL スペースおよびプロジェクト用のナレッジストア REST API

Red Hat Decision Manager はナレッジストア REST API を提供し、これを使用することで Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager のプロジェクトやスペースを操作することができます。ナレッジストアは、Red Hat Decision Manager のアセット用のアーティファクトリポジトリーです。この API のサポートにより、Business Central プロジェクトとスペースの活用と、それらのメンテナンスの自動化が可能になります。

ナレッジストア REST API を使用すると、以下のアクションが可能になります。

- 全プロジェクトおよびスペースに関する情報の取得
- プロジェクトおよびスペースの作成、更新、削除
- プロジェクトのビルド、デプロイおよびテスト
- 以前の REST API 要求または **jobs** についての情報の取得

ナレッジストア REST API 要求には以下のコンポーネントが必要です。

認証

ナレッジストア REST API は、ユーザーロール **rest-all** に HTTP の Basic 認証またはトークンベースの認証を必要とします。お使いの Red Hat Decision Manager に設定されているユーザーロールを表示するには、**~/$SERVER_HOME/standalone/configuration/application-roles.properties** と **~/application-users.properties** に移動します。
ユーザーに **rest-all** ロールを追加するには、**~/$SERVER_HOME/bin** に移動して以下のコマンドを実行します。

```
$ ./add-user.sh -a --user <USERNAME> --password <PASSWORD> --role rest-all
```

ユーザーロールと Red Hat Decision Manager のインストールオプションについての詳細は、『RED HAT DECISION MANAGER インストールのプラニング』を参照してください。

HTTP ヘッダー

ナレッジストア REST API は、API 要求に以下の HTTP ヘッダーを必要とします。

- **Accept**: 要求元のクライアントが受け付けるデータ形式:
  - **application/json** (JSON)
- **Content-Type**: **POST** または **PUT** API 要求データ向けのデータ形式:
  - **application/json** (JSON)

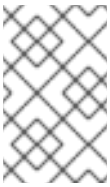
HTTP メソッド

ナレッジストア REST API は、API 要求に以下の HTTP メソッドを必要とします。

- **GET**: 指定したリソースのエンドポイントから指定した情報を取得する
- **POST**: リソースを作成または更新します
- **DELETE**: リソースを削除します

ベース URL

ナレッジストア REST API 要求のベース URL は **http://SERVER:PORT/decision-central/rest/** で、たとえば **http://localhost:8080/decision-central/rest/** となります。



注記

ナレッジストアの REST API のベース URL と Business Central に含まれる Decision Manager コントローラーのものは、両方とも Business Central REST サービスの一部とみなされるので、同じになります。

Endpoints (エンドポイント)

特定のスペースにおける **/spaces/{spaceName}** など、ナレッジストア REST API のエンドポイントは、ナレッジストア REST API ベース URL に追記する URI で、Red Hat Decision Manager の対応するリソースやリソースタイプにアクセスするためのものです。

/spaces/{spaceName} エンドポイントの要求 URL 例

http://localhost:8080/decision-central/rest/spaces/MySpace

要求データ

ナレッジストア REST API の HTTP **POST** 要求は、データに JSON 要求の本文が必要になる場合があります。

POST 要求 URL と JSON 要求の本文データの例

http://localhost:8080/decision-central/rest/spaces/MySpace/projects

```
{
  "name": "Employee_Rostering",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

7.1. REST クライアントまたは CURL ユーティリティーを使用した ナレッジストア REST API による要求送信

ナレッジストア REST API を使用すると、Business Central ユーザーインターフェースを使わずに Red Hat Decision Manager のプロジェクトやスペースを操作することができます。ナレッジストア REST API 要求は、REST クライアントまたは curl ユーティリティーを使って送信できます。

前提条件

- Business Central をインストールし、実行している。
- **rest-all** ユーザーロールで Process Server にアクセスできる。

手順

1. **[GET] /spaces** など、要求の送信先に適した API endpoint を特定し、Business Central からスペースを取得します。

2. REST クライアントまたは curl ユーティリティーで、**/spaces** への **GET** 要求に以下のコンポーネントを記入します。ご自分のユースケースに合わせて、要求詳細を調整します。
REST クライアントの場合:

- **Authentication**: **rest-all** ロールを持つ Business Central ユーザーのユーザー名とパスワードを入力します。
- **HTTP Headers**: 以下のヘッダーを設定します。
  - **Accept**: **application/json**
- **HTTP method**: **GET** に設定します。
- **URL**: ナレッジストア REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/spaces** となります。

curl ユーティリティーの場合:

- **-u**: **rest-all** ロールを持つ Business Central ユーザーのユーザー名とパスワードを入力します。
- **-H**: 以下のヘッダーを設定します。
  - **accept**: **application/json**
- **-X**: **GET** に設定します。
- **URL**: ナレッジストア REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/spaces** となります。

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -X GET
"http://localhost:8080/decision-central/rest/spaces"
```

3. 要求を実行し、Decision Server の応答を確認します。
サーバー応答の例 (JSON):

```
[
  {
    "name": "MySpace",
    "description": null,
    "projects": [
      {
        "name": "Employee_Rostering",
        "spaceName": "MySpace",
        "groupId": "employeerostering",
        "version": "1.0.0-SNAPSHOT",
        "description": "Employee rostering problem optimisation using Planner. Assigns
employees to shifts based on their skill.",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
```

```
        }
       ]
      },
      {
       "name": "Mortgage_Process",
       "spaceName": "MySpace",
       "groupId": "mortgage-process",
       "version": "1.0.0-SNAPSHOT",
       "description": "Getting started loan approval process in BPMN2, decision table, business
rules, and forms.",
       "publicURIs": [
        {
         "protocol": "git",
         "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
        },
        {
         "protocol": "ssh",
         "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
        }
       ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  },
  {
    "name": "MySpace2",
    "description": null,
    "projects": [
     {
       "name": "IT_Orders",
       "spaceName": "MySpace",
       "groupId": "itorders",
       "version": "1.0.0-SNAPSHOT",
       "description": "Case Management IT Orders project",
       "publicURIs": [
        {
         "protocol": "git",
         "uri": "git://localhost:9418/MySpace/example-IT_Orders-1"
        },
        {
         "protocol": "ssh",
         "uri": "ssh://localhost:8001/MySpace/example-IT_Orders-1"
        }
       ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  }
]
```


4. REST クライアントまたは curl ユーティリティーで、**/spaces/{spaceName}/projects** への **POST** 要求を以下のコンポーネントで送信し、スペース内でプロジェクトを作成します。ご自分のユースケースに合わせて、要求詳細を調整します。
   REST クライアントの場合:

- **Authentication**: **rest-all** ロールを持つ Business Central ユーザーのユーザー名とパスワードを入力します。
- **HTTP Headers**: 以下のヘッダーを設定します。
  - **Accept**: **application/json**
  - **Content-Type**: **application/json**
- **HTTP method**: **POST** に設定します。
- **URL**: ナレッジストア REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/spaces/MySpace/projects** となります。
- **要求の本文**: 新規プロジェクト用の ID データのある JSON 要求の本文を追加します。

```
{
  "name": "Employee_Rostering",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

curl ユーティリティーの場合:

- **-u**: **rest-all** ロールを持つ Business Central ユーザーのユーザー名とパスワードを入力します。
- **-H**: 以下のヘッダーを設定します。
  - **accept**: **application/json**
  - **content-type**: **application/json**
- **-X**: **POST** に設定します。
- **URL**: ナレッジストア REST API ベース URL とエンドポイントを入力します。たとえば、**http://localhost:8080/decision-central/rest/spaces/MySpace/projects** となります。
- **-d**: 新規プロジェクト用の ID データのある JSON 要求の本文またはファイル (**@file.json**) を追加します。

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type: application/json" -X POST "http://localhost:8080/decision-central/rest/spaces/MySpace/projects" -d "{ \"name\": \"Employee_Rostering\", \"groupId\": \"employeerostering\", \"version\": \"1.0.0-SNAPSHOT\", \"description\": \"Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.\"}"
```

```
curl -u 'baAdmin:password@1' -H "accept: application/json" -H "content-type: application/json" -X POST "http://localhost:8080/decision-central/rest/spaces/MySpace/projects" -d @my-project.json
```

5. 要求を実行し、Decision Server の応答を確認します。
サーバー応答の例 (JSON):

```
{
  "jobId": "1541017411591-6",
  "status": "APPROVED",
  "spaceName": "MySpace",
  "projectName": "Employee_Rostering",
  "projectGroupId": "employeerostering",
  "projectVersion": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```


エラーが発生した場合は、返されたエラーメッセージを確認して、それに応じて要求を調整します。

7.2. サポートされるナレッジストア REST API エンドポイント

ナレッジストア REST API は、Red Hat Decision Manager 内のスペースおよびプロジェクトを管理し、以前の内 REST API リクエストまたは **jobs** についての情報を取得するエンドポイントを提供します。

7.2.1. スペース

ナレッジストア REST API は Business Central のスペースを管理するための以下のエンドポイントをサポートします。ナレッジストア REST API のベース URL は **http://SERVER:PORT/decision-central/rest/** です。ユーザーロール **rest-all** では、すべての要求で HTTP Basic 認証またはトークンベースの認証が必要です。

[GET] /spaces

Business Central のすべてのスペースを返します。

サーバーの応答例 (JSON)


```
[
  {
    "name": "MySpace",
    "description": null,
    "projects": [
      {
        "name": "Employee_Rostering",
        "spaceName": "MySpace",
        "groupId": "employeerostering",
        "version": "1.0.0-SNAPSHOT",
        "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
          }
        ]
      },
      {
```

```
        "name": "Mortgage_Process",
        "spaceName": "MySpace",
        "groupId": "mortgage-process",
        "version": "1.0.0-SNAPSHOT",
        "description": "Getting started loan approval process in BPMN2, decision table, business
rules, and forms.",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
          }
        ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  },
  {
    "name": "MySpace2",
    "description": null,
    "projects": [
      {
        "name": "IT_Orders",
        "spaceName": "MySpace",
        "groupId": "itorders",
        "version": "1.0.0-SNAPSHOT",
        "description": "Case Management IT Orders project",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-IT_Orders-1"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-IT_Orders-1"
          }
        ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  }
]
```


[GET] /spaces/{spaceName}

指定したスペースに関する情報を返します。

表7.1 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	取得するスペースの名前	文字列	必須

サーバーの応答例 (JSON)


```
{
  "name": "MySpace",
  "description": null,
  "projects": [
    {
      "name": "Mortgage_Process",
      "spaceName": "MySpace",
      "groupId": "mortgage-process",
      "version": "1.0.0-SNAPSHOT",
      "description": "Getting started loan approval process in BPMN2, decision table, business rules, and forms.",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
        }
      ]
    },
    {
      "name": "Employee_Rostering",
      "spaceName": "MySpace",
      "groupId": "employeerostering",
      "version": "1.0.0-SNAPSHOT",
      "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
        }
      ]
    },
    {
      "name": "Evaluation_Process",
      "spaceName": "MySpace",
      "groupId": "evaluation",
      "version": "1.0.0-SNAPSHOT",
      "description": "Getting started Business Process for evaluating employees",
      "publicURIs": [
        {
```

```
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Evaluation_Process"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Evaluation_Process"
        }
      ]
    },
    {
      "name": "IT_Orders",
      "spaceName": "MySpace",
      "groupId": "itorders",
      "version": "1.0.0-SNAPSHOT",
      "description": "Case Management IT Orders project",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-IT_Orders"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-IT_Orders"
        }
      ]
    }
  ],
  "owner": "admin",
  "defaultGroupId": "com.myspace"
}
```


[POST] /spaces

Business Central でスペースを作成します。

表7.2 リクエストパラメーター

名前	説明	タイプ	要件
body	新規スペースの **name**、**description**、**owner**、**defaultGroupId**、およびその他のコンポーネント。	要求の本文	必須

リクエストボディ (JSON) 例


```
{
  "name": "NewSpace",
  "description": "My new space.",
  "owner": "admin",
  "defaultGroupId": "com.newspace"
}
```


サーバーの応答例 (JSON)

```
{
  "jobId": "1541016978154-3",
  "status": "APPROVED",
  "spaceName": "NewSpace",
  "owner": "admin",
  "defaultGroupId": "com.newspace",
  "description": "My new space."
}
```


[DELETE] /spaces/{spaceName}

Business Central からスペースを削除します。

表7.3 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	削除するスペースの名前	文字列	必須

サーバーの応答例 (JSON)


```
{
  "jobId": "1541127032997-8",
  "status": "APPROVED",
  "spaceName": "MySpace",
  "owner": "admin",
  "description": "My deleted space.",
  "repositories": null
}
```


7.2.2. プロジェクト

ナレッジストア REST API は Business Central のプロジェクトを管理、ビルド、デプロイメントするための以下のエンドポイントをサポートします。ナレッジストア REST API のベース URL は **http://SERVER:PORT/decision-central/rest/** です。ユーザーロール **rest-all** では、すべての要求で HTTP Basic 認証またはトークンベースの認証が必要です。

[GET] /spaces/{spaceName}/projects

指定したスペースにあるプロジェクトを返します。

表7.4 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	取得するプロジェクトのスペース名	文字列	必須

サーバーの応答例 (JSON)


```
[
  {
```

```
    "name": "Mortgage_Process",
    "spaceName": "MySpace",
    "groupId": "mortgage-process",
    "version": "1.0.0-SNAPSHOT",
    "description": "Getting started loan approval process in BPMN2, decision table, business rules,
and forms.",
    "publicURIs": [
     {
      "protocol": "git",
      "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
     },
     {
      "protocol": "ssh",
      "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
     }
    ]
  },
  {
    "name": "Employee_Rostering",
    "spaceName": "MySpace",
    "groupId": "employeerostering",
    "version": "1.0.0-SNAPSHOT",
    "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
    "publicURIs": [
     {
      "protocol": "git",
      "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
     },
     {
      "protocol": "ssh",
      "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
     }
    ]
  },
  {
    "name": "Evaluation_Process",
    "spaceName": "MySpace",
    "groupId": "evaluation",
    "version": "1.0.0-SNAPSHOT",
    "description": "Getting started Business Process for evaluating employees",
    "publicURIs": [
     {
      "protocol": "git",
      "uri": "git://localhost:9418/MySpace/example-Evaluation_Process"
     },
     {
      "protocol": "ssh",
      "uri": "ssh://localhost:8001/MySpace/example-Evaluation_Process"
     }
    ]
  },
  {
    "name": "IT_Orders",
    "spaceName": "MySpace",
    "groupId": "itorders",
```

```
    "version": "1.0.0-SNAPSHOT",
    "description": "Case Management IT Orders project",
    "publicURIs": [
      {
        "protocol": "git",
        "uri": "git://localhost:9418/MySpace/example-IT_Orders"
      },
      {
        "protocol": "ssh",
        "uri": "ssh://localhost:8001/MySpace/example-IT_Orders"
      }
    ]
  }
]
```


[GET] /spaces/{spaceName}/projects/{projectName}

指定したスペースにある指定したプロジェクト関する情報を返します。

表7.5 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須
projectName	取得するプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)


```
{
  "name": "Employee_Rostering",
  "spaceName": "MySpace",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
  "publicURIs": [
    {
      "protocol": "git",
      "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
    },
    {
      "protocol": "ssh",
      "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
    }
  ]
}
```


[POST] /spaces/{spaceName}/projects

指定したスペースにプロジェクトを作成します。

表7.6 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	新規プロジェクトが作成されるスペース名	文字列	必須
body	新規プロジェクトの **name**、**groupId**、**version**、**description** 、およびその他のコンポーネント。	要求の本文	必須

リクエストボディ (JSON) 例

```
{
  "name": "Employee_Rostering",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

サーバーの応答例 (JSON)

```
{
  "jobId": "1541017411591-6",
  "status": "APPROVED",
  "spaceName": "MySpace",
  "projectName": "Employee_Rostering",
  "projectGroupId": "employeerostering",
  "projectVersion": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

[DELETE] /spaces/{spaceName}/projects/{projectName}

指定したスペースから指定したプロジェクトを削除します。

表7.7 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須
projectName	削除するプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)

```
{
  "jobId": "1541128617727-10",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

–

[POST] /spaces/{spaceName}/git/clone

指定した Git アドレスから指定したスペースにプロジェクトのクローンを作成します。

表7.8 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのクローンを作成するスペース名	文字列	必須
body	クローンするプロジェクトの **name**、**description**、Git リポジトリーの **userName**、**password**、および **gitURL**。	要求の本文	必須

リクエストボディ (JSON) 例

```
{
  "name": "Employee_Rostering",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
  "userName": "baAdmin",
  "password": "password@1",
  "gitURL": "git://localhost:9418/MySpace/example-Employee_Rostering"
}
```

サーバーの応答例 (JSON)

```
{
  "jobId": "1541129488547-13",
  "status": "APPROVED",
  "cloneProjectRequest": {
    "name": "Employee_Rostering",
    "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
    "userName": "baAdmin",
    "password": "password@1",
    "gitURL": "git://localhost:9418/MySpace/example-Employee_Rostering"
  },
  "spaceName": "MySpace2"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/compile

指定したスペースで指定したプロジェクトをコンパイルします (**mvn compile** と同等)。

表7.9 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須

名前	説明	タイプ	要件
projectName	コンパイルするプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)

```
{
  "jobId": "1541128617727-10",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/test

指定したスペースで指定したプロジェクトをテストします (**mvn test** と同等)。

表7.10 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須
projectName	テストするプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)

```
{
  "jobId": "1541132591595-19",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/install

指定したスペースで指定したプロジェクトをインストールします (**mvn install** と同等)。

表7.11 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須
projectName	インストールするプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)

```
{
  "jobId": "1541132668987-20",
```

```
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```


[POST] /spaces/{spaceName}/projects/{projectName}/maven/deploy

指定したスペースで指定したプロジェクトをデプロイします (**mvn deploy** と同等)。

表7.12 リクエストパラメーター

名前	説明	タイプ	要件
spaceName	プロジェクトのあるスペースの名前	文字列	必須
projectName	デプロイするプロジェクトの名前	文字列	必須

サーバーの応答例 (JSON)


```
{
  "jobId": "1541132816435-21",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```


7.2.3. ジョブ (API リクエスト)

ナレッジストア REST API の **POST** と **DELETE** リクエストはすべて、返されるリクエスト詳細のほかに、各リクエストに関連付けられたジョブ ID を返します。ジョブ ID を使用すると、リクエストステータスを確認したり、送信されたリクエストを削除したりできます。

ナレッジストア REST API リクエストもしくは **ジョブ** には、以下のステータスがあります。

表7.13 ジョブステータス (API リクエストステータス)

ステータス	説明
ACCEPTED	リクエストが受け入れられ、処理中である。
BAD_REQUEST	リクエストに無効なコンテンツが含まれ、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	Decision Server でエラーが発生した。

ステータス	説明
SUCCESS	リクエストが正常に完了した。
FAIL	リクエストが失敗した。
APPROVED	リクエストが承認された。
DENIED	リクエストが拒否された。
GONE	以下のいずれかの理由でリクエストのジョブ ID が見つからなかった。 ● リクエストが明示的に削除された。 ● リクエスト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、リクエストはステータスキャッシュから削除されます。 ● リクエストが元々存在しなかった。

ナレッジストア REST API は、送信済み API リクエストの取得または削除用の以下のエンドポイントをサポートします。ナレッジストア REST API のベース URL は **http://SERVER:PORT/decision-central/rest/** です。ユーザーロール **rest-all** では、すべてのリクエストで HTTP 基本認証またはトークンベースの認証が必要です。

[GET] /jobs/{jobId}

指定されたジョブのステータスを返します (以前に送信された API リクエスト)。

表7.14 リクエストパラメーター

名前	説明	タイプ	要件
jobId	取得するジョブの ID (例: **1541010216919-1**)	文字列	必須

サーバーの応答例 (JSON)

```
{
  "status": "SUCCESS",
  "jobId": "1541010216919-1",
  "result": null,
  "lastModified": 1541010218352,
  "detailedResult": [
    "level:INFO, path:null, text:Build of module 'Mortgage_Process' (requested by system) completed.\n Build: SUCCESSFUL"
  ]
}
```

[DELETE] /jobs/{jobId}

指定したジョブ (以前に送信された API リクエスト) を削除します。ジョブがまだ処理されていない場合は、このリクエストはジョブをジョブキューから削除します。実行中のジョブがキャンセルされたり停止されたりすることはありません。

表7.15 リクエストパラメーター

名前	説明	タイプ	要件
jobId	削除するジョブの ID (例: **1541010216919-1**)	文字列	必須

サーバーの応答例 (JSON)

```
{
  "status": "GONE",
  "jobId": "1541010216919-1",
  "result": null,
  "lastModified": 1541132054916,
  "detailedResult": [
    "level:INFO, path:null, text:Build of module 'Mortgage_Process' (requested by system) completed.\n Build: SUCCESSFUL"
  ]
}
```

第8章 関連資料

- 『Decision Server の管理とモニタリング』
- 『Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ』

付録A バージョン情報

本書の最終更新日: 2019 年 7 月 15 日 (月)