



Red Hat Decision Manager 7.13

Red Hat DecisionManager および KIE Server 設定の管理

法律上の通知

Copyright © 2023 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

このドキュメントでは、ビジネスニーズに合わせて Red Hat DecisionManager と KIE Server の設定とプロパティを変更する方法を説明します。

目次

前書き	6
多様性を受け入れるオープンソースの強化	7
パート I. KIE SERVER の管理とモニタリング	8
第1章 RED HAT DECISION MANAGER のコンポーネント	9
第2章 MAVEN を使用したシステム統合	10
2.1. ローカルプロジェクトの PREEMPTIVE (先行) 認証	10
2.2. BUSINESS CENTRAL における重複した GAV の検出	11
2.3. BUSINESS CENTRAL における重複した GAV 検出設定の管理	11
第3章 RED HAT PROCESS AUTOMATION MANAGER へのパッチ更新およびマイナーリリースアップグレードの適用	13
第4章 KIE SERVER の設定と起動	18
第5章 マネージドの KIE SERVER	20
第6章 マネージド外の KIE SERVER	21
第7章 KIE SERVER および BUSINESS CENTRAL での環境モードの設定	22
第8章 BUSINESS CENTRAL に接続する KIE SERVER の設定	23
第9章 ヘッドレス PROCESS AUTOMATION MANAGER コントローラーのインストールおよび実行	26
9.1. インストーラーでの PROCESS AUTOMATION MANAGER コントローラーを使用する KIE SERVER の設定	26
9.2. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーのインストール	27
9.3. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーの実行	30
9.4. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーを使用した KIE SERVER のクラスタリング	31
第10章 デプロイメント記述子	34
10.1. デプロイメント記述子の設定	34
10.2. デプロイメント記述子の管理	36
10.3. ランタイムエンジンへのアクセス制限	36
第11章 RED HAT DECISION MANAGER の PROMETHEUS メトリクスのモニタリング	38
11.1. KIE SERVER のモニタリングを行う PROMETHEUS メトリクスの設定	38
11.2. RED HAT OPENSIFT CONTAINER PLATFORM の KIE SERVER の PROMETHEUS メトリクスモニタリングの設定	42
11.3. カスタムのメトリクスを使用した KIE SERVER の PROMETHEUS メトリクスモニタリングの拡張	46
第12章 OPENSIFT 接続タイムアウトの設定	52
第13章 LDAP ログインドメインの定義	53
第14章 RH-SSO を使用したサードパーティークライアントの認証	54
14.1. BASIC 認証	54
第15章 KIE SERVER のシステムプロパティ	55
第16章 KIE SERVER の機能と拡張	62
16.1. カスタム REST API エンドポイントを使用した既存の KIE SERVER 機能の拡張	63
16.2. カスタムデータトランスポートを使用するための KIE SERVER の拡張	69

16.3. カスタムクライアント API を使用した KIE SERVER のクライアント拡張	76
第17章 KIE SERVER の使用時のパフォーマンスチューニングに関する考慮点	81
第18章 関連情報	82
パート II. BUSINESS CENTRAL 設定およびプロパティの設定	83
第19章 ユーザーおよびグループの管理	84
19.1. ユーザーの作成	84
19.2. ユーザーの編集	85
19.3. グループの作成	85
19.4. グループの編集	86
第20章 セキュリティー管理	87
20.1. セキュリティー管理プロバイダー	87
20.2. パーミッションおよび設定	89
第21章 アーティファクトの管理	93
21.1. アーティファクトの表示	93
21.2. アーティファクトのダウンロード	93
21.3. アーティファクトのアップロード	93
第22章 データソースおよびデータベースドライバの管理	95
22.1. データソースの追加	95
22.2. データソースの編集	95
22.3. データソースの削除	96
22.4. データベースドライバの追加	96
22.5. データベースドライバの編集	96
22.6. データベースドライバの削除	96
第23章 データセットのオーサリング	98
23.1. データセットの追加	98
23.2. データセットの編集	99
23.3. データの再読み込み	99
23.4. データのキャッシュ	99
第24章 アーキタイプの管理	101
24.1. アーキタイプの表示	101
24.2. アーキタイプの追加	101
24.3. アーキタイプの追加機能の管理	101
24.4. アーキタイプを使用したプロジェクトの作成	102
24.5. BUSINESS CENTRAL のスペース設定を使用したアーキタイプの管理	103
第25章 プロジェクト設定のカスタマイズ	104
第26章 アーティファクトリーポジトリのカスタマイズのプロパティ	107
第27章 言語のカスタマイズ設定	108
第28章 プロセス管理のカスタマイズ	109
第29章 PROCESS DESIGNER のカスタマイズ	110
第30章 SSH キー	111
30.1. デフォルトの SSH キーストア	111
30.2. カスタムの SSH キーストア	111
30.3. SSH キーの作成	112

30.4. SSH キーストアを使用した SSH 公開鍵の登録	112
30.5. SSH キーの削除	113
第31章 BUSINESS CENTRAL でのカスタムタスクの管理	114
第32章 LDAP 接続	118
32.1. LDAP USERGROUPCALLBACK 実装	119
第33章 データベース接続	121
33.1. データベースの USERGROUPCALLBACK 実装	121
第34章 SETTINGS.XML ファイルを使用した MAVEN の設定	123
関連情報	123
第35章 GAV チェック管理	124
35.1. GAV チェックおよび子の GAV エディションの設定	124
35.2. 全プロジェクトの GAV チェックの設定	124
第36章 KIE SERVER および BUSINESS CENTRAL での環境モードの設定	126
第37章 GIT フックおよびリモート GIT リポジトリの統合	127
37.1. POST-COMMIT の GIT フックの作成	127
37.2. リモート GIT リポジトリのインポート	128
37.3. 既存のリモート GIT プロジェクトリポジトリ用の GIT フックの設定	129
37.4. BUSINESS CENTRAL のシステムプロパティとしての GIT フックの設定	130
37.5. リモート GIT リポジトリの統合	131
37.6. GIT フックの終了コード	134
37.7. GIT フック通知のカスタマイズ	134
第38章 BUSINESS CENTRAL のブランチでのロールベースアクセス制御	137
38.1. ロールベースのブランチアクセスのカスタマイズ	137
第39章 プロセスインスタンスログの表示	138
第40章 BUSINESS CENTRAL システムプロパティ	139
第41章 BUSINESS CENTRAL 使用時のパフォーマンスチューニングに関する考慮点	145
パート III. BUSINESS CENTRAL のスタンドアロンパースペクティブの使用	146
第42章 BUSINESS CENTRAL のスタンドアロンパースペクティブ	147
第43章 スタンドアロンのライブラリーパースペクティブの使用	148
第44章 スタンドアロンのエディターパースペクティブの使用	149
第45章 スタンドアロンのコンテンツマネージャーパースペクティブの使用	150
第46章 カスタムページ (ダッシュページ) の使用	151
パート IV. BUSINESS CENTRAL でのカスタムページの作成	152
第47章 BUSINESS CENTRAL カスタムダッシュボード	153
第48章 データセットのオーサリング	154
48.1. データセットの追加	154
48.2. データセットの編集	155
48.3. データの再読み込み	155
48.4. データのキャッシュ	155

第49章 セキュリティー管理	157
49.1. セキュリティー管理プロバイダー	157
49.2. パーミッションおよび設定	159
第50章 ダッシュボードのエクスポート、インポート、およびデプロイ	163
50.1. ダッシュボードデータのエクスポート	163
50.2. BUSINESS CENTRAL ダッシュボードのインポート	163
付録A バージョン情報	165
付録B お問い合わせ先	166

前書き

開発者またはシステム管理者は、ビジネスニーズに合わせて Red Hat Decision Manager と KIE Server の設定およびプロパティーを変更できます。Red Hat Decision Manager ランタイム、Business Central インターフェイス、または KIE Server の動作を変更できます。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みにより、これらの変更は今後の複数のリリースに対して段階的に実施されます。詳細は、[弊社の CTO である Chris Wright のメッセージ](#) を参照してください。

パート I. KIE SERVER の管理とモニタリング

システム管理者として、Red Hat Decision Manager を実稼働環境にインストール、設定、およびアップグレードし、システム障害にすばやくかつ容易に対応できるようになり、システムが最適に稼働するようにできます。

前提条件

- Red Hat JBoss Enterprise Application Platform 7.4 がインストールされている。インストールの詳細は、[Red Hat JBoss Enterprise Application Platform 7.4 Installation Guide](#)を参照してください。
- Red Hat Decision Manager がインストールされている。詳細は、[Planning a Red Hat Decision Manager installation](#) を参照してください。
- Red Hat Decision Manager が稼働し、**admin** ロールで Business Central にログインできる。詳細は、[Planning a Red Hat Decision Manager installation](#)を参照してください。

第1章 RED HAT DECISION MANAGER のコンポーネント

この製品は、Business Central と KIE Server で設定されています。

- Business Central は、ビジネスルールを作成して管理するグラフィカルユーザーインターフェイスです。Business Central は、Red Hat JBoss EAP インスタンスまたは Red Hat OpenShift Container Platform (OpenShift) にインストールできます。
Business Central は、スタンドアロンの JAR ファイルとしても使用できます。Business Central スタンドアロンの JAR ファイルとして使用して、アプリケーションサーバーにデプロイせずに Business Central を実行できます。
- KIE Server では、ルール、およびその他のアーティファクトが実行されます。これは、ルールをインスタンス化して実行し、計画の問題を解決するために使用されます。KIE Server は、Red Hat JBoss EAP インスタンス、Red Hat JBoss EAP クラスター、OpenShift、Oracle WebLogic Server インスタンス、IBM WebSphere Application Server インスタンスに、または Spring Boot アプリケーションの一部としてインストールできます。
KIE Server は、管理モードまたは非管理モードで動作するように設定できます。KIE Server が非管理モードの場合は、手動で KIE コンテナ (デプロイメントユニット) を作成および維持する必要があります。KIE コンテナは、プロジェクトの特定のバージョンです。KIE Server が管理されている場合は、Process Automation Manager コントローラーが KIE Server の設定を管理し、ユーザーはコントローラーと対話形式で KIE コンテナを作成、維持します。

第2章 MAVEN を使用したシステム統合

Red Hat Decision Manager は、[Red Hat JBoss Middleware Maven Repository](#) と Maven Central リポジトリを依存関係ソースとして使用するようになっています。これら両方の依存関係がプロセスビルドに利用可能になるようにしてください。

ご自分のプロジェクトがアーティファクトの特定バージョンに依存していることを確認してください。**LATEST** または **RELEASE** は、一般的に、アプリケーションの依存関係バージョンの特定と管理に使用されます。

- **LATEST** は、アーティファクトの最新デプロイ (スナップショット) バージョンになります。
- **RELEASE** は、リポジトリ内の最新の非スナップショットバージョンリリースになります。

LATEST または **RELEASE** を使用することで、サードパーティーのライブラリーの新リリース時にバージョン番号を更新する必要がなくなります。ただし、ソフトウェアリリースに影響を受けるビルドに対するコントロールができなくなるようになります。

2.1. ローカルプロジェクトの PREEMPTIVE (先行) 認証

お使いの環境にインターネットアクセスがない場合には、Maven Central や他のパブリックリポジトリの代わりに社内リポジトリを設定します。Red Hat Decision Manager サーバーのリモート Maven リポジトリからローカル Maven プロジェクトに JAR をインポートするには、リポジトリサーバーの先行認証をオンにします。`pom.xml` ファイルの `guvnor-m2-repo` 用の認証を設定することでこれが実行できます。以下に例を示します。

```
<server>
  <id>guvnor-m2-repo</id>
  <username>admin</username>
  <password>admin</password>
  <configuration>
    <wagonProvider>httpclient</wagonProvider>
    <httpConfiguration>
      <all>
        <usePreemptive>true</usePreemptive>
      </all>
    </httpConfiguration>
  </configuration>
</server>
```

別の方法では、Authorization HTTP ヘッダーを Base64 でエンコードされた認証情報で設定できます。

```
<server>
  <id>guvnor-m2-repo</id>
  <configuration>
    <httpHeaders>
      <property>
        <name>Authorization</name>
        <!-- Base64-encoded "admin:admin" -->
        <value>Basic YWRtaW46YWRtaW4=</value>
      </property>
    </httpHeaders>
  </configuration>
</server>
```

2.2. BUSINESS CENTRAL における重複した GAV の検出

Business Central のすべての Maven リポジトリで、プロジェクトの **GroupId**、**ArtifactId**、および **Version** (GAV) の各値が重複しているかどうかを確認されます。GAV が重複していると、実行された操作が取り消されます。



注記

重複する GAV の検出は、**Development Mode** のプロジェクトでは無効になっていません。Business Central で重複する GAV 検出を有効にするには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを **OFF** (該当する場合) に切り替えます。

重複した GAV の検出は、以下の操作を実行するたびに実行されます。

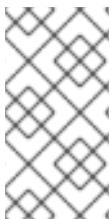
- プロジェクトのプロジェクト定義の保存。
- **pom.xml** ファイルの保存。
- プロジェクトのインストール、ビルド、またはデプロイメント。

以下の Maven リポジトリで重複の GAV が確認されます。

- **pom.xml** ファイルの **<repositories>** 要素および **<distributionManagement>** 要素で指定されたリポジトリ。
- Maven の **settings.xml** 設定ファイルに指定されたリポジトリ。

2.3. BUSINESS CENTRAL における重複した GAV 検出設定の管理

admin ロールを持つ Business Central ユーザーは、プロジェクトで **GroupId** 値、**ArtifactId** 値、および **Version** 値 (GAV) が重複しているかどうかを確認するリポジトリのリストを修正できます。



注記

重複する GAV の検出は、**Development Mode** のプロジェクトでは無効になっていません。Business Central で重複する GAV 検出を有効にするには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを **OFF** (該当する場合) に切り替えます。

手順

1. Business Central で、**Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. プロジェクトの **Settings** タブをクリックし、**Validation** をクリックしてリポジトリのリストを開きます。
3. リスト表示したリポジトリオプションの中から選択するか選択を解除して、重複した GAV の検出を有効または無効にします。
今後、重複した GAV の報告は、検証を有効にしたリポジトリに対してのみ行われます。



注記

この機能を無効にするには、システムの起動時に Business Central の **org.guvnor.project.gav.check.disabled** システムプロパティを **true** に設定します。

```
$ ~/EAP_HOME/bin/standalone.sh -c standalone-full.xml  
-Dorg.guvnor.project.gav.check.disabled=true
```


第3章 RED HAT PROCESS AUTOMATION MANAGER へのパッチ更新およびマイナーリリースアップグレードの適用

大抵の場合は、Business Central、KIE Server、ヘッドレス Process Automation Manager コントローラーなど、Red Hat Process Automation Manager の特定コンポーネントの更新を容易にする自動更新ツールが Red Hat Process Automation Manager のパッチ更新と新規マイナーバージョンで提供されます。デシジョンエンジンやスタンドアロンの Business Central など、その他の Red Hat Process Automation Manager アーティファクトは、各マイナーリリースが含まれる新しいアーティファクトとしてリリースされるため、再インストールして更新を適用する必要があります。

同じ自動更新ツールを使用して、Red Hat Process Automation Manager 7.13 にパッチ更新とマイナーリリースアップグレードの両方を適用することができます。バージョン 7.13 から 7.13.4 への更新といった Red Hat Process Automation Manager のパッチ更新には、最新のセキュリティー更新とバグ修正が含まれています。バージョン 7.12.x から 7.13 へのアップグレードといった Red Hat Process Automation Manager のマイナーリリースアップグレードには、機能強化、セキュリティー更新、バグ修正が含まれます。



注記

Red Hat Process Automation Manager への更新だけが、Red Hat Process Automation Manager パッチ更新に含まれます。Red Hat JBoss EAP への更新は、Red Hat JBoss EAP パッチ配信を使用して適用する必要があります。詳細は、[Red Hat JBoss EAP パッチおよびアップグレードガイド](#) を参照してください。

前提条件

- Red Hat Process Automation Manager インスタンスおよび KIE Server インスタンスを実行していない。Red Hat Process Automation Manager または KIE Server のインスタンスを実行している間は更新を適用しないでください。

手順

1. Red Hat カスタマーポータルでの [Software Downloads](#) ページに移動し (ログインが必要)、ドロップダウンオプションから製品およびバージョンを選択します。
バージョン 7.12.x から 7.13 などのように、Red Hat Process Automation Manager の新たなマイナーリリースにアップグレードする場合は、お使いの Red Hat Process Automation Manager に最新のパッチ更新を適用してから、以下の手順にしたがって新たなマイナーリリースにアップグレードしてください。
2. **Patches** をクリックし、**Red Hat Process Automation Manager [VERSION] Patch Update** をダウンロードし、ダウンロードした **rhpm-\$VERSION-update.zip** ファイルを一時ディレクトリーにデプロイメントします。
この更新ツールは、Business Central、KIE Server、およびヘッドレス Process Automation Manager コントローラーなど、Red Hat Process Automation Manager の一定のコンポーネントの更新を自動化します。この更新ツールを使用して最初に更新を適用し、Red Hat Process Automation Manager ディストリビューションに関連するその他の更新、または新しいリリースアーティファクトをインストールします。
3. 更新ツールにファイルが更新されないようにするには、デプロイメントした **rhpm-\$VERSION-update** フォルダーに移動し、**blacklist.txt** ファイルを開き、更新しないファイルの相対パスを追加します。
ファイルが **blacklist.txt** ファイルのリストに追加されていると、更新スクリプトは、そのファイルを新しいバージョンに置き換えずにそのまま残し、新しいバージョンのファイルに **.new** 接尾辞を付けて追加します。ブロックファイルが配布されなくなると、更新ツール

は、**.removed** 接尾辞の付いた、空のマーカーファイルを作成します。次に、これらの新しいファイルを手動で保持、マージ、または削除することを選択できます。

blacklist.txt ファイルで除外されるファイルの例:

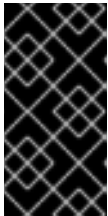
```
WEB-INF/web.xml // Custom file
styles/base.css // Obsolete custom file kept for record
```

更新後の、ブロックされたファイルディレクトリー内のコンテンツ:

```
$ ls WEB-INF
web.xml web.xml.new
```

```
$ ls styles
base.css base.css.removed
```

4. コマンドの端末で、**rhpm-\$VERSION-update.zip** ファイルからデプロイメントした一時ディレクトリーに移動し、以下の形式で **apply-updates** スクリプトを実行します。



重要

更新を行う前に、Red Hat Process Automation Manager インスタンスおよび KIE Server インスタンスが実行していないことを確認します。Red Hat Process Automation Manager または KIE Server のインスタンスを実行している間は更新を適用しないでください。

Linux システムまたは Unix ベースのシステムの場合:

```
$ ./apply-updates.sh $DISTRO_PATH $DISTRO_TYPE
```

Windows の場合:

```
$ .\apply-updates.bat $DISTRO_PATH $DISTRO_TYPE
```

\$DISTRO_PATH の部分は、関連するディストリビューションディレクトリーへのパスで、**\$DISTRO_TYPE** の部分は、更新しているディストリビューションの種類となります。

Red Hat Process Automation Manager 更新ツールでは、以下のディストリビューションの種類がサポートされます。

- **rhpm-business-central-eap7-deployable**: Business Central (**business-central.war**) を更新します。
- **rhpm-kie-server-ee8**: KIE Server (**kie-server.war**) を更新します。



注記

この更新ツールで、Red Hat JBoss EAP EE7 から Red Hat JBoss EAP EE8 に更新および置き換えられます。Red Hat JBoss EAP EE7 は WebLogic および WebSphere に使用されますが、バージョン EE8 は Red Hat JBoss EAP に使用されます。更新ツールでは、WebLogic および WebSphere の KIE Server が更新されていないことを確認します。

- **rhpm-kie-server-jws**: Red Hat JBoss Web Server で KIE Server を更新します (**kie-server.war**)。
- **rhpm-controller-ee7**: ヘッドレス Process Automation Manager controller (**controller.war**) を更新します。
- **rhpm-controller-jws**: Red Hat JBoss Web Server でヘッドレスの Process Automation Manager コントローラーを更新します (**controller.war**)。
Red Hat JBoss EAP で、Red Hat Process Automation Manager の完全ディストリビューションに対する Business Central および KIE Server への更新の例:

```
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/business-central.war
rhpm-business-central-eap7-deployable
```

```
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/kie-server.war rhpm-kie-
server-ee8
```

ヘッドレス Process Automation Manager コントローラーへの更新例 (使用している場合):

```
$ ./apply-updates.sh ~EAP_HOME/standalone/deployments/controller.war rhpm-
controller-ee7
```

この更新スクリプトは、デプロイメントした **rhpm-\$VERSION-update** ディレクトリーに、指定したディストリビューションのコピーを含む **backup** ディレクトリーを作成してから、更新を行います。

- 更新ツールが完了したら、更新ツールをダウンロードした、Red Hat カスタマーポータル **Software Downloads** ページに戻り、Red Hat Process Automation Manager ディストリビューションに関するその他の更新または新しいリリースアーティファクトをインストールします。デジジョンエンジンまたはその他のアドオンに関する **.jar** ファイルなど、Red Hat Process Automation Manager ディストリビューションにすでに存在しているファイルについては、ファイルの既存のバージョンを Red Hat カスタマーポータルから取得した新しいバージョンに置き換えます。
- エアギャップ環境など、スタンドアロンの Red Hat Process Automation Manager 7.13.4 Maven リポジトリー アーティファクト (**rhpm-7.13.4-maven-repository.zip**) を使用する場合は、Red Hat Process Automation Manager 7.13.4 Maven リポジトリーをダウンロードして、ダウンロードした **rhpm-7.13.4-maven-repository.zip** ファイルを既存の **~/maven-repository** ディレクトリーに展開して、関連するコンテンツを更新します。

Maven リポジトリーの更新例:

```
$ unzip -o rhpm-7.13.4-maven-repository.zip 'rhba-7.13.4.GA-maven-repository/maven-
repository/*' -d /tmp/rhbaMavenRepoUpdate
```

```
$ mv /tmp/rhbaMavenRepoUpdate/rhba-7.13.4.GA-maven-repository/maven-repository/
$REPO_PATH/
```



注記

更新が完了したら **/tmp/rhbaMavenRepoUpdate** ディレクトリーを削除してください。

7. オプション: Red Hat Process Automation Manager をプロパティベースのユーザーストレージの使用からファイルベースのユーザーストレージに変更する場合、以下の手順を実行します。

a. **\$JBOSS_HOME** ディレクトリーに移動し、以下のコマンドのいずれかを実行します。

- Linux システムまたは Unix ベースのシステムの場合:

```
$. /bin/standalone.sh --admin-only -c standalone-full.xml
```

```
$. /bin/jboss-cli.sh --connect --file=rhpam-$VERSION-update/elytron/add-kie-fs-realm.cli
```

- Windows の場合:

```
$. /bin/standalone.bat --admin-only -c standalone-full.xml
```

```
$. /bin/jboss-cli.bat --connect --file=rhpam-$VERSION-update/elytron/add-kie-fs-realm.cli
```

b. 以下のコマンドを実行します。

- Linux システムまたは Unix ベースのシステムの場合:

```
$. /bin/elytron-tool.sh filesystem-realm --users-file  
standalone/configuration/application-users.properties --roles-file  
standalone/configuration/application-roles.properties --output-location  
standalone/configuration/kie-fs-realm-users --filesystem-realm-name kie-fs-realm-  
users
```

- Windows の場合:

```
$. /bin/elytron-tool.bat filesystem-realm --users-file  
standalone/configuration/application-users.properties --roles-file  
standalone/configuration/application-roles.properties --output-location  
standalone/configuration/kie-fs-realm-users --filesystem-realm-name kie-fs-realm-  
users
```

c. **rhpam-\$VERSION-update.zip** ファイルを展開したディレクトリーに移動し、以下のいずれかのコマンドを実行して **kie-fs-realm** パッチを適用します。

- Linux システムまたは Unix ベースのシステムの場合:

```
$. /elytron/kie-fs-realm-patch.sh ~/$JBOSS_HOME/standalone/configuration/kie-fs-  
realm-users/
```

- Windows の場合:

```
$. /elytron/kie-fs-realm-patch.bat ~/$JBOSS_HOME/standalone/configuration/kie-fs-  
realm-users/
```

8. 関連する更新をすべて適用したら、Red Hat Process Automation Manager および KIE Server を起動して、Business Central にログインします。

9. Business Central 内のすべてのプロジェクトデータが存在して正確であることを確認し、Business Central ウィンドウの右上隅でプロファイル名をクリックし、**About** をクリックして、更新した製品バージョン番号を確認します。
Business Central でエラーが発生したり、データが不足していることが通知されたら、**rhpm-\$VERSION-update** ディレクトリーの **backup** ディレクトリーにコンテンツを復元し、更新ツールへの変更を戻します。Red Hat カスタマーポータルで Red Hat Process Automation Manager の以前のバージョンから、関連するリリースアーティファクトを再インストールできます。以前のディストリビューションを復元したら、更新を再実行してください。

第4章 KIE SERVER の設定と起動

KIE Server の場所、ユーザー名、パスワード、その他の関連プロパティは、KIE Server の起動時に必要な設定を定義することで設定できます。

手順

Red Hat Decision Manager 7.13 の **bin** ディレクトリーに移動し、以下のプロパティで新しい KIE Server を起動します。お使いの環境に応じて特定のプロパティを調整します。

```
$ ~/EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml ①
-Dorg.kie.server.id=myserver ②
-Dorg.kie.server.user=kie_server_username ③
-Dorg.kie.server.pwd=kie_server_password ④
-Dorg.kie.server.controller=http://localhost:8080/business-central/rest/controller ⑤
-Dorg.kie.server.controller.user=controller_username ⑥
-Dorg.kie.server.controller.pwd=controller_password ⑦
-Dorg.kie.server.location=http://localhost:8080/kie-server/services/rest/server ⑧
```

- ① **standalone-full.xml** サーバープロファイルの開始コマンド
- ② サーバー ID (Business Central で定義したサーバー設定名に一致させる必要がある)
- ③ Process Automation Manager コントローラーから KIE Server に接続する際のユーザー名
- ④ Process Automation Manager コントローラーから KIE Server に接続する際のパスワード
- ⑤ Process Automation Manager コントローラーの場所 (**rest/controller** 接尾辞が付いた Business Central URL)
- ⑥ Process Automation Manager コントローラー REST API に接続するユーザー名
- ⑦ Process Automation Manager コントローラー REST API に接続するパスワード
- ⑧ KIE Server の場所 (この例では Business Central と同じ場所)

注記

Business Central と KIE Server が別々のアプリケーションサーバーインスタンス (Red Hat JBoss EAP など) にインストールされている場合は、Business Central とポートが競合しないように、KIE Server の場所には別のポートを使用します。別の KIE Server ポートが設定されていない場合は、ポートオフセットを追加して、KIE Server プロパティに従って KIE Server のポート値を調整します。

以下に例を示します。

```
-Djboss.socket.binding.port-offset=150
-Dorg.kie.server.location=http://localhost:8230/kie-server/services/rest/server
```

この例のように、Business Central ポートが 8080 で、オフセットを 150 に定義した場合、KIE Server ポートは 8230 になります。

KIE Server は、新しい Business Central に接続し、デプロイするデプロイメントユニット (KIE コンテナ) のリストを収集します。

注記

依存関係の JAR ファイルでクラスを使用して KIE Server クライアントから KIE Server にアクセスすると、Business Central では **ConversionException** および **ForbiddenClassException** が発生します。Business Central でこれらの例外を発生させないようにするには、次のいずれかを実行します。

- クライアント側で例外が発生する場合は、kie-server クライアントに次のシステムプロパティを追加します。

```
System.setProperty("org.kie.server.xstream.enabled.packages", "org.example.**");
```

- サーバー側で例外が発生する場合は、Red Hat Decision Manager インストールディレクトリーから **standalone-full.xml** を開き、<system-properties> タグに以下のプロパティを設定します。

```
<property name="org.kie.server.xstream.enabled.packages" value="org.example.**"/>
```

- 以下の JVM プロパティを設定します。

```
-Dorg.kie.server.xstream.enabled.packages=org.example.**
```

KJAR に存在するクラスは、これらのシステムプロパティを使用して設定しないように想定されています。システムプロパティでは既知のクラスのみを使用し、脆弱性を回避するようにしてください。

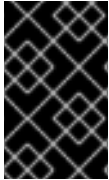
org.example はパッケージ例で、使用するパッケージを何でも定義できます。**org.example1.****, **org.example2.****, **org.example3.**** などのように、コンマ区切りで、複数のパッケージを指定できます。

org.example1.Mydata1, **org.example2.Mydata2** など、特定のクラスも追加できます。

第5章 マネージドの KIE SERVER

管理対象インスタンスには、KIE Server を起動するのに利用可能な Process Automation Manager コントローラーが必要です。

Process Automation Manager コントローラーは、KIE Server の設定を一元的に管理します。各 Process Automation Manager コントローラーは複数の設定を一度に管理でき、環境内に複数の Process Automation Manager コントローラーを配置できます。マネージドの KIE Server に複数の Process Automation Manager コントローラーを設定できますが、一度に接続できるのは1台だけです。



重要

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

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



注記

Process Automation Manager コントローラーに接続せずにスタンドアロンモードで KIE Server を実行する方法については、[6章 マネージド外の KIE Server](#) を参照してください。

第6章 マネージド外の KIE SERVER

マネージド外の KIE Server はスタンドアロンインスタンスであるため、KIE Server 自体から REST/JMS API を使用して個別に設定する必要があります。再起動時には、サーバーが自動的に設定をファイルに永続化し、そのファイルが内部のサーバーの状態として使用されます。

以下の操作を実行中に、設定が更新されます。

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



注記

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

第7章 KIE SERVER および BUSINESS CENTRAL での環境モードの設定

KIE Server は、**production** (実稼働) モードと **development** (開発) モードでの実行が設定可能です。開発モードでは、柔軟な開発ポリシーが提供され、小規模な変更の場合はアクティブなプロセスインスタンスを維持しながら、既存のデプロイメントユニット (KIE コンテナ) を更新できます。また、大規模な変更の場合は、アクティブなプロセスインスタンスを更新する前に、デプロイメントユニットの状態をリセットすることも可能です。実稼働モードは、各デプロイメントで新規デプロイメントユニットが作成される実稼働環境に最適です。

開発環境では、Business Central で **Deploy** をクリックすると、(該当する場合に) 実行中のインスタンスを中止することなくビルドした KJAR ファイルを KIE Server にデプロイすることができます。または、**Redeploy** をクリックすると、ビルドされた KJAR ファイルをデプロイしてすべてのインスタンスを置き換えることができます。次回、ビルドされた KJAR ファイルをデプロイまたは再デプロイすると、以前のデプロイメントユニット (KIE コンテナ) が同じターゲット KIE Server で自動的に更新されます。

実稼働環境では、Business Central の **Redeploy** オプションが無効になり、**Deploy** をクリックして、ビルドした KJAR ファイルを KIE Server 上の新規デプロイメントユニット (KIE コンテナ) にデプロイすることのみが可能です。

手順

1. KIE Server の環境モードを設定するには、**org.kie.server.mode** システムプロパティを **org.kie.server.mode=development** または **org.kie.server.mode=production** に設定します。
2. Business Central のプロジェクトにデプロイメントの動作を設定するには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを切り替えます。



注記

デフォルトでは、KIE Server および Business Central のすべての新規プロジェクトは開発モードになっています。

Development Mode をオンにしたプロジェクトをデプロイしたり、実稼働モードになっている KIE Server に手動で **SNAPSHOT** バージョンの接尾辞を追加したプロジェクトをデプロイしたりすることはできません。

第8章 BUSINESS CENTRAL に接続する KIE SERVER の設定



警告

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

KIE Server を Red Hat Process Automation Manager 環境に設定していない場合、または Red Hat Process Automation Manager 環境に KIE Server を追加する必要がある場合は、KIE Server を設定して Business Central に接続する必要があります。



注記

Red Hat OpenShift Container Platform に KIE Server をデプロイする場合は、[Operator を使用した Red Hat OpenShift Container Platform 4 への Red Hat Decision Manager 環境のデプロイメント](#) ドキュメントで、Business Central に接続する設定手順を参照してください。

KIE Server は管理モードにすることも、非管理モードにすることもできます。KIE Server が非管理モードの場合は、手動で KIE コンテナ (デプロイメントユニット) を作成および維持する必要があります。管理モードの場合は、Process Automation Manager コントローラーが KIE Server の設定を管理し、ユーザーはコントローラーと対話形式で KIE コンテナを作成および維持します。



注記

本セクションの変更は、KIE Server を Business Central で管理し、Red Hat Decision Manager を ZIP ファイルからインストールしている場合にのみ実行してください。Business Central をインストールしている場合は、[9章 ヘッドレス Process Automation Manager コントローラーのインストールおよび実行](#)の記載通りに、ヘッドレス Process Automation Manager コントローラーを使用して KIE Server を管理することができます。

前提条件

- Business Central と KIE Server が Red Hat JBoss EAP インストールのベースディレクトリー (**EAP_HOME**) にインストールされている。



注記

実稼働環境では、Business Central と KIE Server は異なるサーバーにインストールする必要があります。このサンプルでは、**rest-all** と **kie-server** の両ロールを持つ **controllerUser** という名前のユーザー1人のみを使用します。ただし、開発環境などで、KIE Server と Business Central を同じサーバーにインストールする場合は、本セクションの説明に従って、共有の **standalone-full.xml** ファイルを変更します。

- 以下のロールを持つユーザーが存在している
 - Business Central: **rest-all** ロールを持つユーザー

- KIE Server: **kie-server** ロールを持つユーザー

手順

1. Red Hat Process Automation Manager インストールディレクトリーで、**standalone-full.xml** ファイルに移動します。たとえば、Red Hat Process Automation Manager に Red Hat JBoss EAP インストールを使用する場合は **\$EAP_HOME/standalone/configuration/standalone-full.xml** に移動します。
2. **standalone-full.xml** ファイルを開き、**<system-properties>** タグの下に、以下の JVM プロパティーを設定します。

表8.1 マネージド KIE Server インスタンスの JVM プロパティー

プロパティー	値	注記
org.kie.server.id	default-kie-server	KIE Server ID。
org.kie.server.controller	http://localhost:8080/business-central/rest/controller	Business Central の場所 Business Central の API に接続する URL。
org.kie.server.controller.user	controllerUser	Business Central にログイン可能な rest-all ロールを持つユーザー名。
org.kie.server.controller.pwd	controllerUser1234;	Business Central にログインできるユーザーのパスワード。
org.kie.server.location	http://localhost:8080/kie-server/services/rest/server	KIE Server の場所KIE Server の API に接続する URL。

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

プロパティー	値	注記
org.kie.server.user	controllerUser	kie-server ロールを持つユーザー名。
org.kie.server.pwd	controllerUser1234;	ユーザーのパスワード。

以下の例は、KIE Server インスタンスを設定する方法を示しています。

```
<property name="org.kie.server.id" value="default-kie-server"/>
<property name="org.kie.server.controller" value="http://localhost:8080/business-central/rest/controller"/>
<property name="org.kie.server.controller.user" value="controllerUser"/>
<property name="org.kie.server.controller.pwd" value="controllerUser1234;"/>
<property name="org.kie.server.location" value="http://localhost:8080/kie-server/services/rest/server"/>
```

以下の例は、Business Central インスタンスに設定する方法を示しています。

```
<property name="org.kie.server.user" value="controllerUser"/>
<property name="org.kie.server.pwd" value="controllerUser1234;"/>
```

3. KIE サーバーが正常に起動したことを確認するには、KIE サーバーが動作しているときに、**http://SERVER:PORT/kie-server/services/rest/server/** に GET リクエストを送信します。KIE サーバー上での Red Hat Process Automation Manager の実行に関する詳細は、[Running Red Hat Process Automation Manager](#)を参照してください。認証に成功すると、以下の例のような XML 応答が返されます。

```
<response type="SUCCESS" msg="Kie Server info">
  <kie-server-info>
    <capabilities>KieServer</capabilities>
    <capabilities>BRM</capabilities>
    <capabilities>BPM</capabilities>
    <capabilities>CaseMgmt</capabilities>
    <capabilities>BPM-UI</capabilities>
    <capabilities>BRP</capabilities>
    <capabilities>DMN</capabilities>
    <capabilities>Swagger</capabilities>
    <location>http://localhost:8230/kie-server/services/rest/server</location>
    <messages>
      <content>Server KieServerInfo{serverId='first-kie-server', version='7.5.1.Final-redhat-1', location='http://localhost:8230/kie-server/services/rest/server', capabilities=[KieServer, BRM, BPM, CaseMgmt, BPM-UI, BRP, DMN, Swagger]}started successfully at Mon Feb 05 15:44:35 AEST 2018</content>
      <severity>INFO</severity>
      <timestamp>2018-02-05T15:44:35.355+10:00</timestamp>
    </messages>
    <name>first-kie-server</name>
    <id>first-kie-server</id>
    <version>7.5.1.Final-redhat-1</version>
  </kie-server-info>
</response>
```

4. 登録が正常に完了したことを確認します。
 - a. Business Central にログインします。
 - b. **Menu** → **Deploy** → **Execution Servers** の順にクリックします。
正常に登録されている場合は、登録されたサーバーの ID が表示されます。

第9章 ヘッドレス PROCESS AUTOMATION MANAGER コントローラーのインストールおよび実行

KIE Server は、管理モードまたは非管理モードで動作するように設定できます。KIE Server が非管理モードの場合は、手動で KIE コンテナ (デプロイメントユニット) を作成および維持する必要があります。KIE Server が管理されている場合は、Process Automation Manager コントローラーが KIE Server の設定を管理し、ユーザーはコントローラーと対話形式で KIE コンテナを作成、維持します。

Business Central には Process Automation Manager コントローラーが組み込まれています。Business Central をインストールしている場合は、**Execution Server** ページを使用して KIE コンテナを作成および維持します。Business Central なしで KIE Server の管理を自動化するには、ヘッドレス Process Automation Manager コントローラーを使用することで可能になります。

9.1. インストーラーでの PROCESS AUTOMATION MANAGER コントローラーを使用する KIE SERVER の設定

KIE Server は、Process Automation Manager コントローラーで管理することも、非管理モードにすることも可能です。KIE Server が非管理モードの場合は、手動で KIE コンテナ (デプロイメントユニット) を作成および維持する必要があります。KIE Server が管理されている場合は、Process Automation Manager コントローラーが KIE Server の設定を管理し、ユーザーはコントローラーと対話形式で KIE コンテナを作成、維持します。

Process Automation Manager コントローラーは Business Central と統合します。Business Central をインストールしている場合は、Business Central の **Execution Server** ページを使用して Process Automation Manager コントローラーと対話します。

インストーラーは対話モードまたは CLI モードで使用し、Business Central と KIE Server をインストールして、Process Automation Manager コントローラーで KIE Server を設定します。

前提条件

- バックアップを作成済みの Red Hat JBoss EAP 7.4 サーバーインストールが設定された 2 台のコンピューターが利用できる。
- インストールを完了するのに必要なユーザーパーミッションが付与されている。

手順

1. 1 台目のコンピューターで、インタラクティブモードまたは CLI モードでインストーラーを実行します。詳細は [Red Hat JBoss EAP 7.4 への Red Hat Decision Manager のインストールおよび設定](#) を参照してください。
2. **Component Selection** ページで、**KIE Server** チェックボックスを外します。
3. Business Central インストールを完了します。
4. 2 台目のコンピューターで、インタラクティブモードまたは CLI モードでインストーラーを実行します。
5. **Component Selection** ページで **Business Central** チェックボックスを外します。
6. **Configure Runtime Environment** ページで **Perform Advanced Configuration** を選択します。
7. **Customize KIE Server properties** を選択し、**Next** をクリックします。

8. Business Central のコントローラー URL を入力し、KIE Server に追加のプロパティを設定します。コントローラー URL は、以下の形式を取ります。<HOST:PORT> は、2 台目のコンピューターの Business Central のアドレスに置き換えます。

```
<HOST:PORT>/business-central/rest/controller
```

9. インストールを完了します。
10. Process Automation Manager コントローラーが Business Central と統合されていることを確認するには、Business Central の **Execution Servers** ページに移動して、設定した KIE Server が **REMOTE SERVERS** に表示されていることを確認します。

9.2. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーのインストール

ヘッドレス Process Automation Manager コントローラーをインストールして、REST API または KIE Server Java Client API を使用して対話します。

前提条件

- バックアップを作成済みの Red Hat JBoss EAP システム (バージョン 7.4) が利用できる。Red Hat JBoss EAP システムのベースディレクトリーを **EAP_HOME** とする。
- インストールを完了するのに必要なユーザーパーミッションが付与されている。

手順

1. Red Hat カスタマーポータルの [Software Downloads](#) ページに移動し (ログインが必要)、ドロップダウンオプションから製品およびバージョンを選択します。
 - **製品:** Process Automation Manager
 - **バージョン:** 7.13.4
2. Red Hat Process Automation Manager 7.13.4 Add Ons(**rhpmam-7.13.4-add-ons.zip** ファイル) をダウンロードします。
3. **rhpmam-7.13.4-add-ons.zip** ファイルを展開します。 **rhpmam-7.13.4-controller-ee7.zip** ファイルは展開したディレクトリーにあります。
4. **rhpmam-7.13.4-controller-ee7.zip** アーカイブを一時ディレクトリーに展開します。以下の例では、この名前を **TEMP_DIR** とします。
5. **TEMP_DIR/rhpmam-7.13.4-controller-ee7/controller.war** ディレクトリーを **EAP_HOME/standalone/deployments/** にコピーします。



警告

コピーするヘッドレス Process Automation Manager コントローラーデプロイメントの名前が、Red Hat JBoss EAP インスタンスの既存デプロイメントと競合しないことを確認します。

6. **TEMP_DIR/rhpam-7.13.4-controller-ee7/SecurityPolicy/** ディレクトリーの内容を **EAP_HOME/bin** にコピーします。
7. ファイルの上書きを求めるプロンプトが出されたら、**Yes** を選択します。
8. **EAP_HOME/standalone/deployments/** ディレクトリーに、**controller.war.dodeploy** という名前で空のファイルを作成します。このファイルにより、サーバーが起動するとヘッドレス Process Automation Manager コントローラーが自動的にデプロイされます。

9.2.1. ヘッドレス Process Automation Manager コントローラーのユーザー作成

ヘッドレス Process Automation Manager コントローラーを使用する前に、**kie-server** ロールを持つユーザーを作成する必要があります。

前提条件

- ヘッドレス Process Automation Manager コントローラーが Red Hat JBoss EAP インストールのベースディレクトリー (**EAP_HOME**) にインストールされている。

手順

1. 端末アプリケーションで **EAP_HOME/bin** ディレクトリーに移動します。
2. 以下のコマンドを入力し、**<USERNAME>** および **<PASSWORD>** を、作成するユーザー名およびパスワードに置き換えます。

```
$ ./bin/jboss-cli.sh --commands="embed-server --std-out=echo,/subsystem=elytron/filesystem-realm=ApplicationRealm:add-identity(identity=<USERNAME>),/subsystem=elytron/filesystem-realm=ApplicationRealm:set-password(identity=<USERNAME>, clear={password='<PASSWORD>'}),/subsystem=elytron/filesystem-realm=ApplicationRealm:add-identity-attribute(identity=<USERNAME>, name=role, value=['kie-server'])"
```



注記

必ず、既存のユーザー、ロール、またはグループとは異なるユーザー名を指定してください。たとえば、**admin** という名前のユーザーは作成しないでください。

パスワードは 8 文字以上で、数字と、英数字以外の文字をそれぞれ 1 文字以上使用する必要があります。ただし & の文字は使用できません。

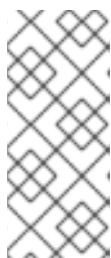
3. ユーザー名とパスワードを書き留めておきます。

9.2.2. KIE Server およびヘッドレス Process Automation Manager コントローラーの設定

KIE Server をヘッドレス Process Automation Manager コントローラーから管理する場合は、KIE Server インストールの **standalone-full.xml** ファイルと、ヘッドレス Process Automation Manager コントローラーインストールの **standalone.xml** ファイルを編集する必要があります。

前提条件

- KIE Server が **EAP_HOME** にインストールされている。
- ヘッドレス Process Automation Manager コントローラーが **EAP_HOME** にインストールされている。



注記

実稼働環境では KIE Server およびヘッドレス Process Automation Manager コントローラーを異なるサーバーにインストールすることを推奨します。ただし、開発環境など、KIE Server およびヘッドレス Process Automation Manager コントローラーを同じサーバーにインストールする場合は、併せて共有の **standalone-full.xml** ファイルを変更します。

- KIE Server ノードに、**kie-server** ロールのあるユーザーが作成されている。
- サーバーノードに、**kie-server** ロールのあるユーザーが作成されている。

手順

1. **EAP_HOME/standalone/configuration/standalone-full.xml** ファイルの **<system-properties>** セクションに以下のプロパティを追加し、**<USERNAME>** および **<USER_PWD>** を、**kie-server** ロールを持つユーザーの認証情報に置き換えます。

```
<property name="org.kie.server.user" value="<USERNAME>"/>
<property name="org.kie.server.pwd" value="<USER_PWD>"/>
```

2. KIE Server の **EAP_HOME/standalone/configuration/standalone-full.xml** ファイルの **<system-properties>** セクションに以下のプロパティを追加します。

```
<property name="org.kie.server.controller.user" value="<CONTROLLER_USER>"/>
<property name="org.kie.server.controller.pwd" value="<CONTROLLER_PWD>"/>
<property name="org.kie.server.id" value="<KIE_SERVER_ID>"/>
<property name="org.kie.server.location" value="http://<HOST>:<PORT>/kie-server/services/rest/server"/>
<property name="org.kie.server.controller" value="<CONTROLLER_URL>"/>
```

3. このファイルで、以下の値を置き換えます。

- **<CONTROLLER_USER>** および **<CONTROLLER_PWD>** を **kie-server** ロールを持つユーザーの認証情報に置き換えます。
- **<KIE_SERVER_ID>** を KIE Server システムの ID または名前に置き換えます (例: **rhpam-7.13.4-kie-server-1**)。

- **<HOST>** を KIE Server ホストの ID または名前に置き換えます (例: **localhost** または **192.7.8.9**)。
- **<PORT>** を KIE Server ホストのポートに置き換えます (例: **8080**)。



注記

org.kie.server.location プロパティで KIE Server の場所を指定します。

- **<CONTROLLER_URL>** をアドレス Process Automation Manager コントローラー の URL で置き換えます。起動中に KIE Server がこの URL に接続します。

9.3. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーの実行

ヘッドレス Process Automation Manager コントローラーを Red Hat JBoss EAP にインストールしたら、以下の手順に従ってヘッドレス Process Automation Manager コントローラーを実行します。

前提条件

- ヘッドレス Process Automation Manager コントローラーが Red Hat JBoss EAP インストールのベースディレクトリー (**EAP_HOME**) にインストールされ設定されている。

手順

1. ターミナルアプリケーションで **EAP_HOME/bin** に移動します。
2. ヘッドレス Process Automation Manager コントローラーを、KIE Server をインストールした Red Hat JBoss EAP インスタンスと同じ Red Hat JBoss EAP インスタンスにインストールしている場合は、以下のいずれかのコマンドを実行します。

- Linux または UNIX ベースのシステムの場合:

```
└─$ ./standalone.sh -c standalone-full.xml
```

- Windows の場合:

```
└─ standalone.bat -c standalone-full.xml
```

3. ヘッドレス Process Automation Manager コントローラーを、KIE Server をインストールした Red Hat JBoss EAP インスタンスとは別の Red Hat JBoss EAP インスタンスにインストールしている場合は、**standalone.sh** スクリプトで Process Automation Manager コントローラーを開始します。



注記

この場合は、**standalone.xml** ファイルに必要な設定変更を加えます。

- Linux または UNIX ベースのシステムの場合:

```
└─$ ./standalone.sh
```

- Windows の場合:

```
standalone.bat
```

4. ヘッドレス Process Automation Manager コントローラーが Red Hat JBoss EAP 上で動作していることを確認するには、以下のコマンドを入力します。<CONTROLLER> はユーザー名で、<CONTROLLER_PWD> はパスワードになります。このコマンドにより、KIE Server インスタンスに関する情報が出力されます。

```
curl -X GET "http://<HOST>:<PORT>/controller/rest/controller/management/servers" -H "accept: application/xml" -u '<CONTROLLER>:<CONTROLLER_PWD>'
```



注記

あるいは、KIE Server Java API Client を使用して、ヘッドレス Process Automation Manager コントローラーにアクセスすることもできます。

9.4. ヘッドレス PROCESS AUTOMATION MANAGER コントローラーを使用した KIE SERVER のクラスタリング

Process Automation Manager コントローラーは Business Central と統合します。ただし、Business Central をインストールしない場合は、ヘッドレス Process Automation Manager コントローラーをインストールし、REST API または KIE Server Java Client API を使用してそのコントローラーと対話します。

前提条件

- バックアップを作成してある Red Hat JBoss EAP システム (バージョン 7.4 またはそれ以降) が利用できる。Red Hat JBoss EAP システムのベースディレクトリーを **EAP_HOME** とする。
- インストールを完了するのに必要なユーザーパーミッションが付与されている。
- [Red Hat JBoss EAP クラスター環境への Red Hat Decision Manager のインストールおよび設定](#) に記載されているように、共有フォルダーを備えた NFS サーバーを利用できる。

手順

1. Red Hat カスタマーポータル [の Software Downloads](#) ページに移動し (ログインが必要)、ドロップダウンオプションから製品およびバージョンを選択します。
 - **PRODUCT:** Process Automation Manager
 - **バージョン:** 7.13.4
2. **Red Hat Process Automation Manager 7.13.4 Add Ons(rhpam-7.13.4-add-ons.zip** ファイル) をダウンロードします。
3. **rhpam-7.13.4-add-ons.zip** ファイルを展開します。 **rhpam-7.13.4-controller-ee7.zip** ファイルは展開したディレクトリーにあります。
4. **rhpam-7.13.4-controller-ee7.zip** アーカイブを一時ディレクトリーに展開します。以下の例では、この名前を **TEMP_DIR** とします。

5. **TEMP_DIR/rhpam-7.13.4-controller-ee7/controller.war** ディレクトリーを **EAP_HOME/standalone/deployments/** にコピーします。



警告

コピーするヘッドレス Process Automation Manager コントローラーデプロイメントの名前が、Red Hat JBoss EAP インスタンスの既存デプロイメントと競合しないことを確認します。

6. **TEMP_DIR/rhpam-7.13.4-controller-ee7/SecurityPolicy/** ディレクトリーの内容を **EAP_HOME/bin** にコピーします。
7. ファイルの上書きを求めるプロンプトが出されたら、**Yes** をクリックします。
8. **EAP_HOME/standalone/deployments/** ディレクトリーに、**controller.war.dodeploy** という名前で空のファイルを作成します。このファイルにより、サーバーが起動するとヘッドレス Process Automation Manager コントローラーが自動的にデプロイされます。
9. テキストエディターで **EAP_HOME/standalone/configuration/standalone.xml** ファイルを開きます。
10. 以下のプロパティーを **<system-properties>** 要素に追加し、**<NFS_STORAGE>** を、テンプレート設定が保存されている NFS ストレージへの絶対パスに置き換えます。

```
<system-properties>
  <property name="org.kie.server.controller.templatefile.watcher.enabled" value="true"/>
  <property name="org.kie.server.controller.templatefile" value="<NFS_STORAGE>"/>
</system-properties>
```

テンプレートファイルには、特定のデプロイメントシナリオのデフォルト設定が含まれます。

org.kie.server.controller.templatefile.watcher.enabled プロパティーの値を **true** に設定すると、別のスレッドが開始してテンプレートファイルの修正を監視します。この確認の間隔はデフォルトで 30000 ミリ秒になり、**org.kie.server.controller.templatefile.watcher.interval** システムプロパティーで制御できます。このプロパティーの値を **false** に設定すると、テンプレートファイルへの変更の検出が、サーバーの再起動時に制限されます。

11. ヘッドレス Process Automation Manager コントローラーを開始するには、**EAP_HOME/bin** に移動して、以下のコマンドを実行します。

- Linux または UNIX ベースのシステムの場合:

```
$ ./standalone.sh
```

- Windows の場合:

```
standalone.bat
```

Red Hat JBoss Enterprise Application Platform のクラスタリング環境で Red Hat Decision Manager を稼働する方法の詳細情報は、[Red Hat JBoss EAP クラスタ環境への Red Hat Decision Manager のインストールおよび設定](#) を参照してください。

第10章 デプロイメント記述子

プロセスとルールは Apache Maven ベースのパッケージに保存され、ナレッジアーカイブ、または KJAR と呼ばれます。ルール、プロセス、アセット、およびその他のプロジェクトアーティファクトは、Maven がビルドおよび管理する JAR ファイルの一部です。**kmodule.xml** と呼ばれる、KJAR の **META-INF** ディレクトリ内に保存されるファイルを使用して、KIE ベースとセッションを定義できます。デフォルトでは、この **kmodule.xml** ファイルは空です。

KIE Server のようなランタイムコンポーネントが KJAR の処理を開始するタイミングで常に **kmodule.xml** を検索して、ランタイム表記を構築します。

デプロイメント記述子は **kmodule.xml** ファイルを補い、デプロイメントにおいてより詳細な制御を提供します。このような記述子は任意で、記述子がなくてもデプロイメントは正常に行われます。記述子を使用して、persistence、auditing、runtime strategy といったメタ値を含む技術的属性を設定することができます。

記述子を使用すると、(サーバーレベルのデフォルト、KJAR ごとに異なるデプロイメント記述子、その他のサーバー設定など) 複数レベルで KIE Server を設定できるようになります。記述子を使用して、デフォルトの KIE Server 設定にシンプルなカスタマイズが可能になります (KJAR ごとなど)。

記述子は **kie-deployment-descriptor.xml** と呼ばれるファイルで定義し、**META-INF** ディレクトリーの **kmodule.xml** ファイルの隣に置くことができます。このデフォルトの場所とファイル名は、システムパラメーターとして指定すると変更できます。

```
-Dorg.kie.deployment.desc.location=file:/path/to/file/company-deployment-descriptor.xml
```

10.1. デプロイメント記述子の設定

デプロイメント記述子を使用すると、ユーザーは以下の複数レベルで実行サーバーを設定することができます。

- **サーバーレベル:** メインのレベルで、サーバーにデプロイされているすべての KJAR に適用されます。
- **KJAR レベル:** このレベルでは、KJAR ベースで記述子を設定できます。
- **デプロイ時レベル:** KJAR のデプロイ時に適用される記述子です。

デプロイメント記述子で指定されたより詳細な設定アイテムは、マージされるコレクションベースの設定アイテムを除いて、サーバーレベルのものよりも優先されます。優先順位は、**デプロイ時設定** > **KJAR 設定** > **サーバー設定** となります。



注記

デプロイ時の設定は、REST API によるデプロイメントに適用されます。

たとえば、サーバーレベルで定義された (設定可能なアイテムの1つである) persistence mode が **NONE** で、同じモードが KJAR レベルでは **JPA** と指定されている場合、その KJAR の実際のモードは **JPA** になります。その KJAR についてデプロイメント記述子で persistence mode に何も指定されていない場合 (またはデプロイメント記述子がない場合) は、サーバーレベルの設定にフォールバックします。このケースでは、**NONE** (またはサーバーレベルのデプロイメント記述子がない場合は **JPA**) になります。

設定内容

デプロイメント記述子では、高度な技術的設定が可能です。以下の表では、設定可能な詳細と、それぞれの許容値とデフォルト値を掲載しています。

表10.1 デプロイメント記述子

設定	XML エントリー	許容値	デフォルト値
ランタイムデータの永続ユニット名	persistence-unit	有効な永続パッケージ名	org.jbpm.domain
監査データの永続ユニット名	audit-persistence-unit	有効な永続パッケージ名	org.jbpm.domain
永続モード	persistence-mode	JPA, NONE	JPA
監査モード	audit-mode	JPA, JMS、または NONE	JPA
ランタイムストラテジー	runtime-strategy	SINGLETON、PER_REQUEST、または PER_PROCESS_INSTANCE	SINGLETON
登録するイベントリスナーリスト	event-listeners	ObjectModel のような有効なリスナークラス名	デフォルト値なし
登録するタスクイベントリスナーリスト	task-event-listeners	ObjectModel のような有効なリスナークラス名	デフォルト値なし
登録する作業アイテムハンドラーリスト	work-item-handlers	NamedObjectHandler のような有効な作業アイテムハンドラークラス	デフォルト値なし
登録するグローバルリスト	globals	NamedObjectModel のような有効なグローバル変数	デフォルト値なし
登録するマーシャリングストラテジー (プラグ可能変数永続)	marshalling-strategies	有効な ObjectModel クラス	デフォルト値なし
KJAR のリソースにアクセス可能となるために必要なロール	required-roles	文字列のロール名	デフォルト値なし
KIE セッションの追加の環境エントリー	environment-entries	有効な NamedObjectModel	デフォルト値なし
KIE セッションの追加の設定オプション	configurations	有効な NamedObjectModel	デフォルト値なし

設定	XML エントリ	許容値	デフォルト値
リモートサービスのシリアル化に使用するクラス	remoteable-class	有効な CustomClass	デフォルト値なし



警告

実稼働環境では、EJB Timer スケジューラー (KIE Server のデフォルトのスケジューラー) を使用した Singleton ランタイムストラテジーを使用しないでください。この組み合わせを使用すると、負荷がかかると、Hibernate で問題が発生する可能性があります。具体的に他のストラテジーを使用する理由がない限り、プロセスインスタンス別のランタイムストラテジーの使用を推奨します。この制約に関する詳細情報は、[Hibernate issues with Singleton strategy and EJBTimerScheduler](#) を参照してください。

10.2. デプロイメント記述子の管理

デプロイメント記述子を設定するには、Business Central で **Menu → Design → \$PROJECT_NAME → Settings → Deployments** と移動します。

プロジェクトが作成されるたびに、ストックの **kie-deployment-descriptor.xml** ファイルがデフォルト値で生成されます。

すべての KJAR で完全なデプロイメント記述子を提供する必要はありません。部分的なデプロイメント記述子の提供は可能で、かつ推奨されるものです。たとえば、異なる監査モードを使用する必要がある場合は、その KJAR のみにそれを指定し、残りの属性はサーバーレベルのデフォルト値で定義します。

OVERRIDE_ALL マージモードの使用時には、すべての設定アイテムを指定する必要があります。関連する KJAR は常に指定された設定を使用し、階層内の他のデプロイメント記述子とマージしないためです。

10.3. ランタイムエンジンへのアクセス制限

required-roles 設定アイテムは、デプロイメント記述子で編集できます。このプロパティーが定義するグループに属するユーザーにのみ特定プロセスへのアクセスを付与することで、KJAR ごとまたはサーバーレベルごとにランタイムエンジンへのアクセスを制限します。

セキュリティロールを使用してプロセス定義へのアクセスを制限したり、ランタイムでのアクセスを制限することができます。

リポジトリの制限に基づいてこのプロパティーに必要なロールを追加するのがデフォルトの動作になります。必要な場合は、セキュリティレールムで定義されている実際のロールに合致するロールを提供することで、このプロパティーを手動で変更できます。

手順

1. プロジェクトのデプロイメント記述子設定を開くには、Business Central で **Menu → Design → \$PROJECT_NAME → Settings → Deployments** の順に選択します。

2. 設定リストから、**Required Roles** をクリックし、次に **Add Required Role** をクリックします。
3. **Add Required Role** ウィンドウで、このデプロイメントにアクセスをするパーミッションのロール名を入力し、**Add** をクリックします。
4. デプロイメントにアクセスする権限を持つロールをさらに追加するには、前の手順を繰り返します。
5. すべてのロールを追加したら、**Save** をクリックします。

第11章 RED HAT DECISION MANAGER の PROMETHEUS メトリクスのモニタリング

Prometheus は、オープンソースのシステムモニタリングツールキットで、Red Hat Decision Manager と連携して、ビジネスルール、プロセス、Decision Model and Notation (DMN) モデル、その他の Red Hat Decision Manager アセットの実行に関するメトリクスを収集して保存できます。KIE Server への REST API 呼び出しや、Prometheus expression browser、Grafana などのデータグラフツールを使用して、保存したメトリックにアクセスできます。

オンプレミスの KIE Server、Spring Boot の KIE Server、Red Hat OpenShift Container Platform の KIE Server デプロイメントに、Prometheus メトリック監視を設定できます。

KIE Server が Prometheus を使用して公開するメトリクスで、利用可能なものの一覧については、[Red Hat カスタマーポータル](#) から [Red Hat Process Automation Manager 7.13.4 Source Distribution](#) をダウンロードし、`~/rhpam-7.13.4-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-services/kie-server-services-prometheus/src/main/java/org/kie/server/services/prometheus` に移動してください。



重要

Prometheus に対する Red Hat のサポートは、Red Hat 製品ドキュメントに記載の設定および設定の推奨事項に限定されます。

11.1. KIE SERVER のモニタリングを行う PROMETHEUS メトリクスの設定

KIE Server インスタンスが Prometheus を使用し、Red Hat Decision Manager でのビジネスアセットアクティビティに関連するメトリックを収集して保存するように設定できます。KIE Server が Prometheus を使用して公開するメトリクスで、利用可能なものの一覧については、[Red Hat カスタマーポータル](#) から [Red Hat Process Automation Manager 7.13.4 Source Distribution](#) をダウンロードし、`~/rhpam-7.13.4-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-services/kie-server-services-prometheus/src/main/java/org/kie/server/services/prometheus` に移動してください。

前提条件

- KIE Server がインストールされている。
- **kie-server** ユーザーロールで KIE Server にアクセスできる。
- Prometheus がインストールされている。Prometheus のダウンロードおよび使用に関する情報は、[Prometheus ドキュメントページ](#) を参照してください。

手順

1. KIE Server インスタンスで、**org.kie.prometheus.server.ext.disabled** システムプロパティを **false** に設定して、Prometheus 拡張機能を有効にします。このプロパティは、KIE Server の起動時、または Red Hat Decision Manager ディストリビューションの **standalone.xml** または **standalone-full.xml** ファイルで定義できます。
2. Spring Boot で Red Hat Decision Manager を実行する場合は、**application.properties** システムプロパティで必要なキーを設定します。

Red Hat Decision Manager および Prometheus の Spring Boot `application.properties` キー

```
kieserver.drools.enabled=true
kieserver.dmn.enabled=true
kieserver.prometheus.enabled=true
```

- Prometheus ディストリビューションの **prometheus.yaml** ファイルで、**scrape_configs** セクションに以下の設定を追加して、Prometheus が KIE Server からメトリックを収集 (scrape) するように設定します。

prometheus.yaml ファイルの Scrape 設定

```
scrape_configs:
  - job_name: 'kie-server'
    metrics_path: /SERVER_PATH/services/rest/metrics
    basicAuth:
      username: USER_NAME
      password: PASSWORD
    static_configs:
      - targets: ["HOST:PORT"]
```

Spring Boot の prometheus.yaml ファイルでの Scrape 設定 (該当する場合)

```
scrape_configs:
  - job_name: 'kie'
    metrics_path: /rest/metrics
    static_configs:
      - targets: ["HOST:PORT"]
```

KIE Server の場所と設定に合わせて、値を置き換えます。

- KIE Server インスタンスを起動します。

Red Hat JBoss EAP での Red Hat Decision Manager の起動コマンド例

```
$ cd ~/EAP_HOME/bin
$ ./standalone.sh --c standalone-full.xml
```

設定済みの KIE Server インスタンスを起動すると、Prometheus はメトリックの収集を開始し、KIE Server はメトリックを REST API エンドポイント

http://HOST:PORT/SERVER/services/rest/metrics (または Spring Boot では **http://HOST:PORT/rest/metrics**) に公開します。

- REST クライアントまたは curl ユーティリティーで、以下のコンポーネントを含む REST API 要求を送信し、KIE Server がメトリックを公開していることを確認します。

REST クライアントの場合:

- **Authentication:** **kie-server** ロールを持つ KIE 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:** KIE Server REST API ベース URL とメトリックエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/metrics** (または Spring Boot では **http://localhost:8080/rest/metrics**) 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 KIE 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:** KIE Server REST API ベース URL とメトリックエンドポイントを入力します。たとえば、**http://localhost:8080/kie-server/services/rest/metrics** (または Spring Boot では **http://localhost:8080/rest/metrics**) となります。

Red Hat JBoss EAP での Red Hat Decision Manager の curl コマンド例

```
curl -u 'baAdmin:password@1' -X GET "http://localhost:8080/kie-server/services/rest/metrics"
```

Spring Boot での Red Hat Decision Manager の curl コマンド例

```
curl -u 'baAdmin:password@1' -X GET "http://localhost:8080/rest/metrics"
```

サーバーの応答例

```
# HELP kie_server_container_started_total Kie Server Started Containers
# TYPE kie_server_container_started_total counter
kie_server_container_started_total{container_id="task-assignment-kjar-1.0"}, 1.0
# HELP solvers_running Number of solvers currently running
# TYPE solvers_running gauge
solvers_running 0.0
# HELP dmn_evaluate_decision_nanosecond DMN Evaluation Time
# TYPE dmn_evaluate_decision_nanosecond histogram
# HELP solver_duration_seconds Time in seconds it took solver to solve the constraint
problem
# TYPE solver_duration_seconds summary
solver_duration_seconds_count{solver_id="100tasks-5employees.xml"}, 1.0
solver_duration_seconds_sum{solver_id="100tasks-5employees.xml"}, 179.828255925
solver_duration_seconds_count{solver_id="24tasks-8employees.xml"}, 1.0
solver_duration_seconds_sum{solver_id="24tasks-8employees.xml"}, 179.995759653
# HELP drl_match_fired_nanosecond Drools Firing Time
# TYPE drl_match_fired_nanosecond histogram
# HELP dmn_evaluate_failed_count DMN Evaluation Failed
# TYPE dmn_evaluate_failed_count counter
# HELP kie_server_start_time Kie Server Start Time
# TYPE kie_server_start_time gauge
kie_server_start_time{name="myapp-kieserver",server_id="myapp-
kieserver",location="http://myapp-kieserver-demo-
monitoring.127.0.0.1.nip.io:80/services/rest/server",version="7.4.0.redhat-20190428"},
1.557221271502E12
# HELP kie_server_container_running_total Kie Server Running Containers
# TYPE kie_server_container_running_total gauge
```

```

kie_server_container_running_total{container_id="task-assignment-kjar-1.0"} 1.0
# HELP solver_score_calculation_speed Number of moves per second for a particular solver
solving the constraint problem
# TYPE solver_score_calculation_speed summary
solver_score_calculation_speed_count{solver_id="100tasks-5employees.xml"} 1.0
solver_score_calculation_speed_sum{solver_id="100tasks-5employees.xml"} 6997.0
solver_score_calculation_speed_count{solver_id="24tasks-8employees.xml"} 1.0
solver_score_calculation_speed_sum{solver_id="24tasks-8employees.xml"} 19772.0

```

KIE Server でメトリックが利用できない場合は、このセクションで説明されている KIE Server および Prometheus の設定を確認します。

また、**http://HOST:PORT/graph** の Prometheus の expression browser で収集したメトリックと対話したり、Prometheus データソースを Grafana などのデータグラフ作成ツールと統合したりすることもできます。

図11.1 Prometheus expression browser と KIE Server メトリック

The screenshot shows the Prometheus expression browser interface. At the top, there are navigation tabs: Prometheus, Alerts, Graph, Status, and Help. Below the navigation, there is a search bar containing the query 'dmn_evaluate_decision_nanosecond_bucket'. To the right of the search bar, it displays 'Load time: 64ms', 'Resolution: 14s', and 'Total time series: 14'. Below the search bar is an 'Execute' button and a dropdown menu showing the query. Below the execute button are two tabs: 'Graph' and 'Console'. The 'Console' tab is active, displaying a table of results.

Element	Value
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="+Inf",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26795
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="1.0E9",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26795
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="1000000.0",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26454
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="2.0E9",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26795
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="2000000.0",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26652
dmn_evaluate_decision_nanosecond_bucket[artifact_id="example-prometheus-kjar",container_id="function-definition",decision_name="http://www.omg.org/spec/DMN/20151101",decision_namespace="function-definition",endpoint="web",group_id="com.company",instance="172.17.0.13.8080",job="rhdm-app-metrics",le="3.0E9",namespace="demo-monitoring",pod="myapp-kieserver-1-6cg2",service="rhdm-app-metrics",version="1.0-SNAPSHOT"]	26795

図11.2 Prometheus expression browser と KIE Server ターゲット

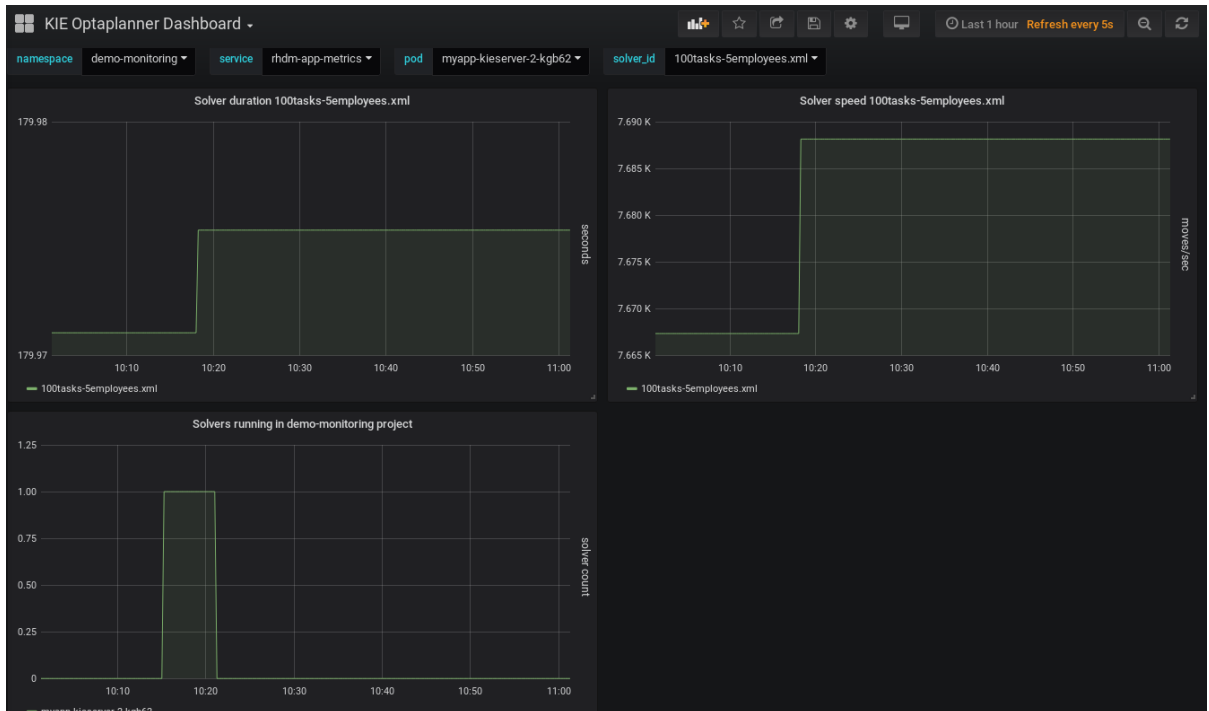
The screenshot shows the Prometheus Targets page. At the top, there are navigation tabs: Prometheus, Alerts, Graph, Status, and Help. Below the navigation, the page title is 'Targets'. There is a status indicator 'All Unhealthy' and a 'Show all' button. Below that, there is a section for 'kie-server (1/1 up)' with a 'Show all' button. Below this section is a table with columns: Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8080/kie-server/services/rest/metrics	UP	instance="localhost:8080"	4.994s ago	1.101ms	

図11.3 Grafana ダッシュボードと DMN モデルの KIE Server メトリック



図11.4 Grafana ダッシュボードとソルバーの KIE Server メトリック



関連情報

- [Getting Started with Prometheus](#)
- [Grafana Support for Prometheus](#)
- [Using Prometheus in Grafana](#)

11.2. RED HAT OPENSIFT CONTAINER PLATFORM の KIE SERVER の PROMETHEUS メトリクスモニタリングの設定

Prometheus を使用して Red Hat Decision Manger でのビジネスアセットアクティビティーに関連するメトリックを収集して保存するように、Red Hat OpenShift Container Platform で KIE Server デプロイメントを設定できます。KIE Server が Prometheus を使用して公開するメトリクスで、利用可能なものの一覧については、[Red Hat カスタマーポータル](#) から Red Hat Process Automation Manager 7.13.4 Source Distribution をダウンロードし、`~/rhpam-7.13.4-sources/src/droolsjbpm-integration-$VERSION/kie-server-parent/kie-server-services/kie-server-services-prometheus/src/main/java/org/kie/server/services/prometheus` に移動してください。

前提条件

- KIE Server が、Red Hat OpenShift Container Platform にインストールおよびデプロイメントされている。OpenShift 上の KIE Server の詳細については、[Red Hat Decision Manager 7.13 の製品ドキュメント](#) で関連する OpenShift デプロイメントオプションを参照してください。
- **kie-server** ユーザーロールで KIE Server にアクセスできる。
- Prometheus Operator がインストールされている。Prometheus Operator のダウンロードと使用についての詳細は、GitHub の [Prometheus Operator](#) プロジェクトを参照してください。

手順

1. OpenShift 上の KIE Server デプロイメントの **DeploymentConfig** オブジェクトで、**PROMETHEUS_SERVER_EXT_DISABLED** 環境変数を **false** に設定して、Prometheus 拡張機能を有効にします。この変数は、OpenShift Web コンソールを使用するか、コマンド端末で **oc** コマンドを使用してこの変数を設定してください。

```
oc set env dc/<dc_name> PROMETHEUS_SERVER_EXT_DISABLED=false -n
<namespace>
```

OpenShift に KIE Server をデプロイしていない場合は、OpenShift デプロイメントに使用予定の OpenShift テンプレート (例: **rhcam713-prod-immutable-kieserver.yaml**) で、**PROMETHEUS_SERVER_EXT_DISABLED** テンプレートパラメーターを **false** に設定して、Prometheus 拡張機能を有効にします。

OpenShift Operator を使用して KIE Server を OpenShift にデプロイする場合は、KIE Server の設定で、**PROMETHEUS_SERVER_EXT_DISABLED** 環境変数を **false** に設定して、Prometheus 拡張機能を有効にします。

```
apiVersion: app.kiegroup.org/v1
kind: KieApp
metadata:
  name: enable-prometheus
spec:
  environment: rhcam-trial
  objects:
    servers:
      - env:
          - name: PROMETHEUS_SERVER_EXT_DISABLED
            value: "false"
```

2. **service-metrics.yaml** ファイルを作成して、KIE Server から Prometheus にメトリックを公開するサービスを追加します。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    description: RHPAM Prometheus metrics exposed
  labels:
    app: myapp-kieserver
    application: myapp-kieserver
    template: myapp-kieserver
    metrics: rhcam
```

```

name: rhpam-app-metrics
spec:
  ports:
    - name: web
      port: 8080
      protocol: TCP
      targetPort: 8080
  selector:
    deploymentConfig: myapp-kieserver
  sessionAffinity: None
  type: ClusterIP

```

3. コマンドターミナルで、**oc** コマンドを使用して、**service-metrics.yaml** ファイルを OpenShift デプロイメントに適用します。

```
oc apply -f service-metrics.yaml
```

4. **metrics-secret** など、OpenShift シークレットを作成して、KIE Server の Prometheus メトリックにアクセスします。シークレットには `username` 要素および `password` 要素と、KIE Server ユーザー認証情報が含まれている必要があります。OpenShift シークレットの詳細は、OpenShift [開発者ガイド](#) の [シークレット](#) の章を参照してください。
5. **ServiceMonitor** オブジェクトを定義する **service-monitor.yaml** ファイルを作成します。サービスモニターにより Prometheus を KIE Server メトリックサービスに接続できます。

```

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: rhpam-service-monitor
  labels:
    team: frontend
spec:
  selector:
    matchLabels:
      metrics: rhpam
  endpoints:
    - port: web
      path: /services/rest/metrics
      basicAuth:
        password:
          name: metrics-secret
          key: password
        username:
          name: metrics-secret
          key: username

```

6. コマンド端末で、**oc** コマンドを使用して、**service-monitor.yaml** ファイルを OpenShift デプロイメントに適用します。

```
oc apply -f service-monitor.yaml
```

上記の設定を完了すると、Prometheus はメトリックの収集を開始し、KIE Server は REST API エンドポイント **http://HOST:PORT/kie-server/services/rest/metrics** にメトリックを公開します。

`http://HOST:PORT/graph` の Prometheus expression browser で収集したメトリックと対話したり、Prometheus データソースを Grafana などのデータグラフ作成ツールと統合したりすることができます。

Prometheus expression browser の場所のホストとポートである `http://HOST:PORT/graph` は、Prometheus Operator をインストールしたときに Prometheus Web コンソールを公開したルートで定義されています。OpenShift ルートの詳細は、OpenShift アーキテクチャドキュメントの [ルート](#) の章を参照してください。

図11.5 Prometheus expression browser と KIE Server メトリック

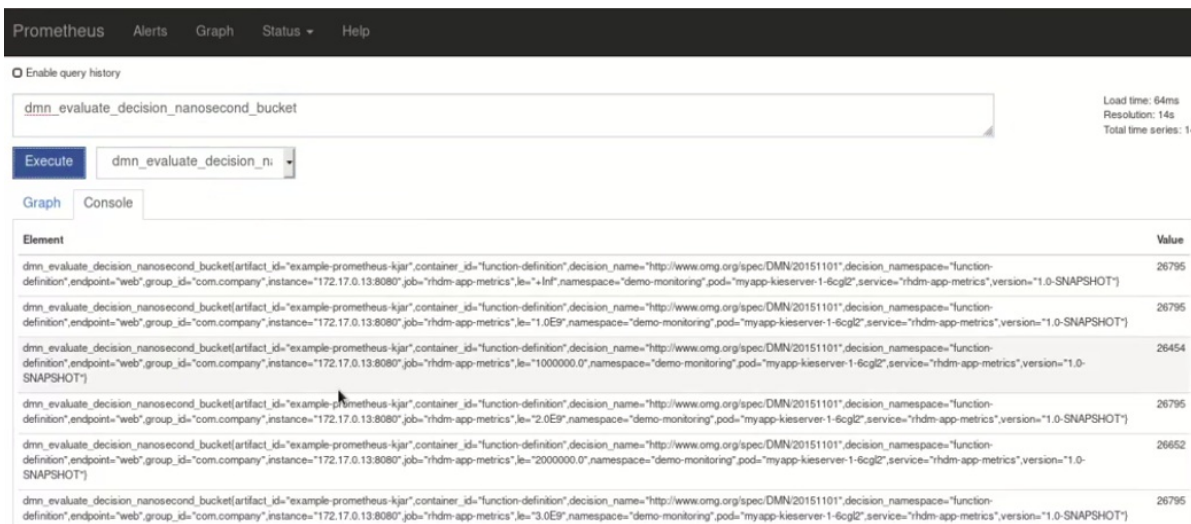


図11.6 Prometheus expression browser と KIE Server ターゲット

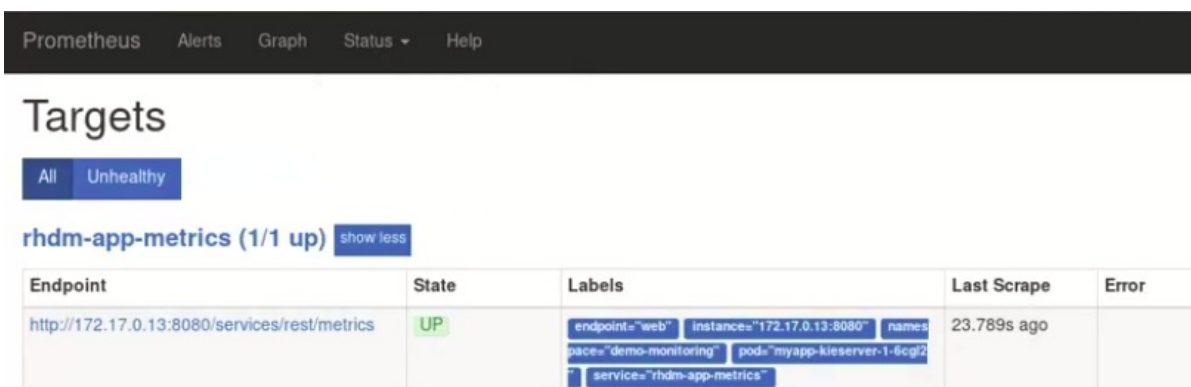
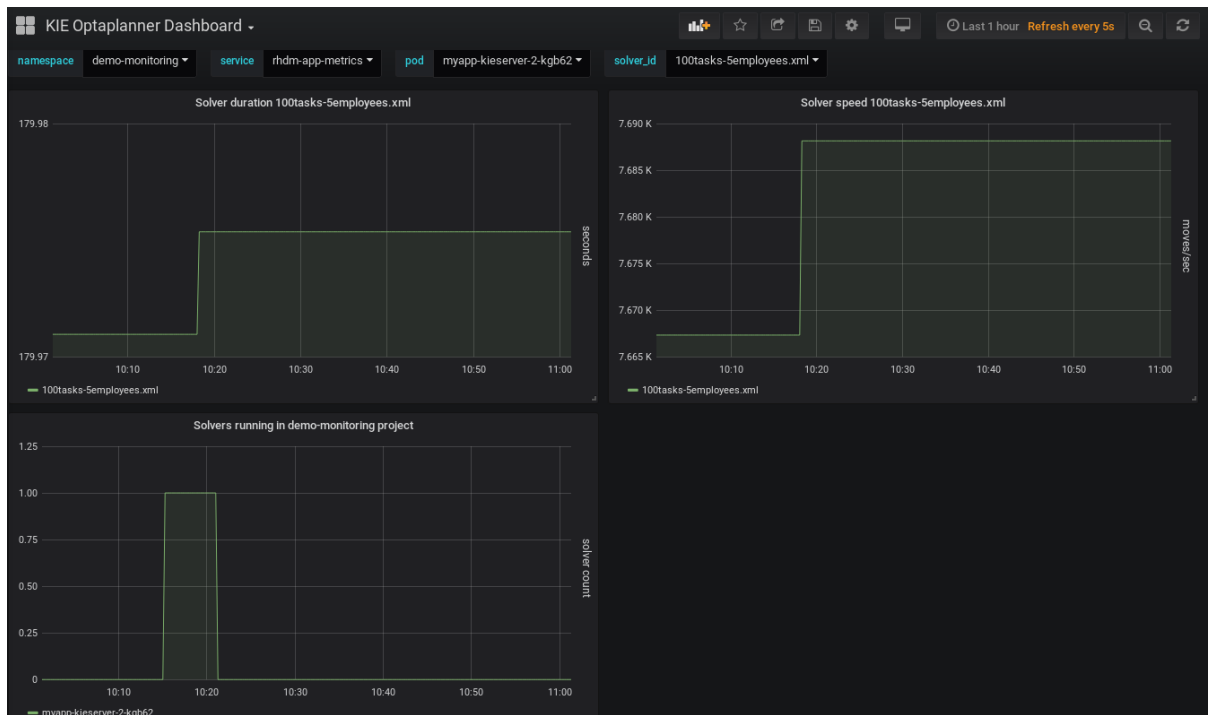


図11.7 Grafana ダッシュボードと DMN モデルの KIE Server メトリック



図11.8 Grafana ダッシュボードとソルバーの KIE Server メトリック



関連情報

- [Prometheus Operator](#)
- [Getting started with the Prometheus Operator](#)
- [Prometheus RBAC](#)
- [Grafana Support for Prometheus](#)
- [Using Prometheus in Grafana](#)
- [Red Hat Decision Manager 7.13 の製品ドキュメント](#) の OpenShift デプロイメントオプション

11.3. カスタムのメトリクスを使用した KIE SERVER の PROMETHEUS メトリクスモニタリングの拡張

KIE Server インスタンスが Prometheus メトリクスモニタリングを使用するように設定後に、ビジネス要件に合わせてカスタムのメトリクスを使用するように、KIE Server の Prometheus 機能を拡張できます。Prometheus は、KIE Server が Prometheus に公開するデフォルトのメトリックと、カスタムのメトリックを収集して、保存します。

たとえば、以下の手順では、Prometheus で収集して保存するように、カスタムの Decision Model and Notation (DMN) メトリックを定義します。

前提条件

- Prometheus メトリクスモニタリングが、KIE Server インスタンス用に設定されている。オンプレミス型 KIE Server を使用した Prometheus 設定に関する詳細は、「[KIE Server のモニタリングを行う Prometheus メトリクスの設定](#)」を参照してください。Red Hat OpenShift Container Platform の KIE Server を使用した Prometheus 設定に関する詳細は、「[Red Hat OpenShift Container Platform の KIE Server の Prometheus メトリクスモニタリングの設定](#)」を参照してください。

手順

1. 空の Maven プロジェクトを作成して、以下のパッケージタイプと依存関係を、プロジェクトの **pom.xml** ファイルに定義します。

サンプルプロジェクトの pom.xml ファイルの例

```
<packaging>jar</packaging>

<properties>
  <version.org.kie>7.67.0.Final-redhat-00024</version.org.kie>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-common</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-drools</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-prometheus</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-dmn-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-dmn-core</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.jbpm</groupId>
    <artifactId>jbpm-services-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.jbpm</groupId>
```

```

    <artifactId>jbpm-executor</artifactId>
    <version>${version.org.kie}</version>
</dependency>
<dependency>
    <groupId>org.optaplanner</groupId>
    <artifactId>optaplanner-core</artifactId>
    <version>${version.org.kie}</version>
</dependency>
<dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient</artifactId>
    <version>0.5.0</version>
</dependency>
</dependencies>

```

- 以下の例にあるように、カスタムの Prometheus メトリックを定義する、カスタムのリスナークラスの一部として、**org.kie.server.services.prometheus.PrometheusMetricsProvider** インターフェイスを実装します。

カスタムのリスナークラス内の DMNRuntimeEventListener リスナーの実装例

```

package org.kie.server.ext.prometheus;

import io.prometheus.client.Gauge;
import org.kie.dmn.api.core.ast.DecisionNode;
import org.kie.dmn.api.core.event.AfterEvaluateBKMEvent;
import org.kie.dmn.api.core.event.AfterEvaluateContextEntryEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionServiceEvent;
import org.kie.dmn.api.core.event.AfterEvaluateDecisionTableEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateBKMEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateContextEntryEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionServiceEvent;
import org.kie.dmn.api.core.event.BeforeEvaluateDecisionTableEvent;
import org.kie.dmn.api.core.event.DMNRuntimeEventListener;
import org.kie.server.api.model.ReleaseId;
import org.kie.server.services.api.KieContainerInstance;

public class ExampleCustomPrometheusMetricListener implements
DMNRuntimeEventListener {

    private final KieContainerInstance kieContainer;

    private final Gauge randomGauge = Gauge.build()
        .name("random_gauge_nanosecond")
        .help("Random gauge as an example of custom KIE Prometheus metric")
        .labelNames("container_id", "group_id", "artifact_id", "version",
"decision_namespace", "decision_name")
        .register();

    public ExampleCustomPrometheusMetricListener(KieContainerInstance
containerInstance) {
        kieContainer = containerInstance;
    }
}

```

```

public void beforeEvaluateDecision(BeforeEvaluateDecisionEvent e) {
}

public void afterEvaluateDecision(AfterEvaluateDecisionEvent e) {
    DecisionNode decisionNode = e.getDecision();
    ReleaseId releaseId = kieContainer.getResource().getReleaseId();
    randomGauge.labels(kieContainer.getContainerId(), releaseId.getGroupId(),
        releaseId.getArtifactId(), releaseId.getVersion(),
        decisionNode.getModelName(), decisionNode.getModelNamespace())
        .set((int) (Math.random() * 100));
}

public void beforeEvaluateBKM(BeforeEvaluateBKMEvent event) {
}

public void afterEvaluateBKM(AfterEvaluateBKMEvent event) {
}

public void beforeEvaluateContextEntry(BeforeEvaluateContextEntryEvent event) {
}

public void afterEvaluateContextEntry(AfterEvaluateContextEntryEvent event) {
}

public void beforeEvaluateDecisionTable(BeforeEvaluateDecisionTableEvent event) {
}

public void afterEvaluateDecisionTable(AfterEvaluateDecisionTableEvent event) {
}

public void beforeEvaluateDecisionService(BeforeEvaluateDecisionServiceEvent event) {
}

public void afterEvaluateDecisionService(AfterEvaluateDecisionServiceEvent event) {
}
}

```

PrometheusMetricsProvider インターフェイスには、Prometheus メトリック収集に必要なリスナーが含まれます。このインターフェイスは、プロジェクトの **pom.xml** ファイルで宣言した **kie-server-services-prometheus** 依存関係に組み込まれます。

以下の例では、**ExampleCustomPrometheusMetricListener** クラスは、(**PrometheusMetricsProvider** インターフェイスからの) **DMNRuntimeEventListener** リスナーを実装し、Prometheus で収集、保存するカスタムの DMN メトリックを定義します。

- 以下の例にあるように、**PrometheusMetricsProvider** インターフェイスとカスタムのリスナーを関連付ける、カスタムのメトリックプロバイダーの一部として **PrometheusMetricsProvider** インターフェイスを実装します。

カスタムメトリックプロバイダークラスの PrometheusMetricsProvider インターフェイスの実装例

```

package org.kie.server.ext.prometheus;

import org.jbpm.executor.AsynchronousJobListener;
import org.jbpm.services.api.DeploymentEventListener;

```

```

import org.kie.api.event.rule.AgendaEventListener;
import org.kie.api.event.rule.DefaultAgendaEventListener;
import org.kie.dmn.api.core.event.DMNRuntimeEventListener;
import org.kie.server.services.api.KieContainerInstance;
import org.kie.server.services.prometheus.PrometheusMetricsProvider;
import org.optaplanner.core.impl.phase.event.PhaseLifecycleListener;
import org.optaplanner.core.impl.phase.event.PhaseLifecycleListenerAdapter;

public class MyPrometheusMetricsProvider implements PrometheusMetricsProvider {

    public DMNRuntimeEventListener createDMNRuntimeEventListener(KieContainerInstance
kContainer) {
        return new ExampleCustomPrometheusMetricListener(kContainer);
    }

    public AgendaEventListener createAgendaEventListener(String kieSessionId,
KieContainerInstance kContainer) {
        return new DefaultAgendaEventListener();
    }

    public PhaseLifecycleListener createPhaseLifecycleListener(String solverId) {
        return new PhaseLifecycleListenerAdapter() {
        };
    }

    public AsynchronousJobListener createAsynchronousJobListener() {
        return null;
    }

    public DeploymentEventListener createDeploymentEventListener() {
        return null;
    }
}

```

以下の例では、**MyPrometheusMetricsProvider** クラスは **PrometheusMetricsProvider** インターフェイスを実装し、このクラスには、カスタムの **ExampleCustomPrometheusMetricListener** リスナークラスが含まれます。

- 新規メトリックプロバイダーを KIE Server で検出できるようにするには、Maven プロジェクトに **META-INF/services/org.kie.server.services.prometheus.PrometheusMetricsProvider** ファイルを作成して、このファイル内に **PrometheusMetricsProvider** 実装クラスの完全修飾クラス名を追加します。以下の例では、このファイルに **org.kie.server.ext.prometheus.MyPrometheusMetricsProvider** の 1 行が含まれています。
- プロジェクトを構築して、作成された JAR ファイルをプロジェクトの **~/kie-server.war/WEB-INF/lib** ディレクトリーにコピーします。たとえば、Red Hat JBoss EAP ではこのディレクトリーへのパスは **EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib** です。Red Hat OpenShift Container Platform に Red Hat Decision Manager をデプロイする場合は、カスタム KIE Server イメージを作成し、この JAR ファイルをイメージに追加します。追加の JAR ファイルを含むカスタム KIE Server イメージの作成に関する詳細は、[Deploying an Red Hat Decision Manager environment on Red Hat OpenShift Container Platform 4 using Operators](#) を参照してください。
- KIE Server を起動して、実行中の KIE Server に構築したプロジェクトをデプロイします。プロジェクトは、Business Central インターフェイスまたは KIE Server REST API (<http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}>) への PUT 要

求) を使用してデプロイできます。

実行中の KIE Server にプロジェクトをデプロイした後に、Prometheus はメトリックの収集を開始し、KIE Server はメトリックを REST API エンドポイント

http://HOST:PORT/SERVER/services/rest/metrics (または Spring Boot では

http://HOST:PORT/rest/metrics) に公開します。

第12章 OPENSIFT 接続タイムアウトの設定

デフォルトでは、OpenShift のルートは 30 秒を超えた HTTP リクエストをタイムアウトするように設定されています。これにより Business Central でセッションタイムアウト問題が発生し、以下の動作につながるおそれがあります。

- "Unable to complete your request.The following exception occurred: (TypeError) : Cannot read property 'indexOf' of null."
- "Unable to complete your request.The following exception occurred: (TypeError) : b is null."
- Business Central で **Project** リンクまたは **Server** リンクをクリックすると、空白ページが表示される。

すべての Business Central テンプレートには拡張タイムアウト設定が含まれています。

Business Central OpenShift ルートのタイムアウトを長く設定するには、ターゲットルートに **haproxy.router.openshift.io/timeout: 60s** のアノテーションを追加します。

```
- kind: Route
  apiVersion: v1
  id: "$APPLICATION_NAME-rhpamcentr-http"
  metadata:
    name: "$APPLICATION_NAME-rhpamcentr"
  labels:
    application: "$APPLICATION_NAME"
  annotations:
    description: Route for Business Central's http service.
    haproxy.router.openshift.io/timeout: 60s
  spec:
    host: "$BUSINESS_CENTRAL_HOSTNAME_HTTP"
    to:
      name: "$APPLICATION_NAME-rhpamcentr"
```

グローバルのルート固有のタイムアウト注釈の完全一覧は、[OpenShift ドキュメント](#) を参照してください。

第13章 LDAP ログインドメインの定義

Red Hat Decision Manager が認証と承認に LDAP を使用するように設定するには、LDAP ログインドメインを定義します。これは、Git SSH 認証が別のセキュリティドメインを使用する可能性があるためです。

LDAP ログインドメインを定義するには、**org.uberfire.domain** システムプロパティを使用します。たとえば、Red Hat JBoss Enterprise Application Platform 上でこのプロパティを以下のように **standalone.xml** ファイルに追加します。

```
<system-properties>
  <!-- other system properties -->
  <property name="org.uberfire.domain" value="LDAPAuth"/>
</system-properties>
```

認証されたユーザーが、LDAP で適切なロール (**admin**、**analyst**、**reviewer**) に関連付けられているようにしてください。

第14章 RH-SSO を使用したサードパーティークライアントの認証

Business Central または KIE Server が提供するさまざまなリモートサービスを使用するには、curl、wget、Web ブラウザー、カスタムの REST クライアントなどのクライアントが、RH-SSO サーバー経由で認証を受け、要求を実行するために有効なトークンを取得する必要があります。リモートのサービスを使用するには、認証済みのユーザーに以下のロールを割り当てる必要があります。

- **rest-all**: Business Central リモートサービスを使用する場合
- **kie-server**: KIE Server のリモートサービスを使用する場合

RH-SSO 管理コンソールを使用してこれらのロールを作成し、リモートサービスを使用するユーザーに割り当てます。

クライアントは、以下のオプションのいずれかを使用して RH-SSO 経由で認証できます。

- クライアントでサポートされている場合は Basic 認証
- トークンベースの認証

14.1. BASIC 認証

Business Central と KIE Server の両方に対して RH-SSO クライアントアダプターの設定で Basic 認証を有効にした場合は、以下の例のようにトークンの付与/更新の呼び出しをせずにサービスを呼び出すことができます。

- Web ベースのリモトリポジトリエンドポイントの場合:

```
curl http://admin:password@localhost:8080/business-central/rest/repositories
```

- KIE Server の場合:

```
curl http://admin:password@localhost:8080/kie-server/services/rest/server/
```

第15章 KIE SERVER のシステムプロパティー

KIE Server では、以下のシステムプロパティー (ブートストラップスイッチ) を使用してサーバーの動作を設定できます。

表15.1 KIE Server の拡張機能を無効にするシステムプロパティー

プロパティー	値	デフォルト	説明
<code>org.drools.server.ext.disabled</code>	true、false	false	true に設定した場合は、(ルールのサポートなど) Business Rule Management (BRM) のサポートが無効になります。
<code>org.optaplanner.server.ext.disabled</code>	true、false	false	true に設定した場合は、Red Hat ビルドの OptaPlanner のサポートが無効になります。
<code>org.kie.prometheus.server.ext.disabled</code>	true、false	true	true に設定した場合は、Prometheus Server 拡張が無効になります。
<code>org.kie.scenariosimulation.server.ext.disabled</code>	true、false	true	true に設定した場合は、テストシナリオサーバー拡張が無効になります。
<code>org.kie.dmn.server.ext.disabled</code>	true、false	false	true に設定した場合は、KIE Server DMN サポートが無効になります。
<code>org.kie.swagger.server.ext.disabled</code>	true、false	false	true に設定した場合は、KIE Server swagger のドキュメントサポートが無効になります。



注記

以下の表に記載した Process Automation Manager コントローラーのプロパティーの中で、必須と印がついているものがあります。Business Central で KIE Server コンテナを作成または削除する場合は、このプロパティーを設定してください。Business Central との対話なしに KIE Server を別個で使用する場合は、必須のプロパティーを設定する必要はありません。

表15.2 Process Automation Manager コントローラーに必要なシステムプロパティー

プロパティー	値	デフォルト	説明
<code>org.kie.server.id</code>	String	該当なし	サーバーに割り当てる任意の ID。ヘッドレス Process Automation Manager コントローラーが Business Central 外に設定されている場合は、この ID を使用して、サーバーはヘッドレス Process Automation Manager コントローラーと接続し、KIE コンテナ設定をフェッチします。指定されていない場合、ID は自動で生成されます。

プロパティ	値	デフォルト	説明
org.kie.server.user	String	kieserver	管理モードで実行する場合に必要な、Process Automation Manager コントローラーから KIE Server への接続に使用するユーザー名。このプロパティは、Business Central のシステムプロパティで設定します。Process Automation Manager コントローラーを使用する場合は、このプロパティを設定します。
org.kie.server.pwd	String	kieserver1!	管理モードで実行する場合に必要な、Process Automation Manager コントローラーから KIE Server への接続に使用するパスワード。このプロパティは、Business Central のシステムプロパティで設定します。Process Automation Manager コントローラーを使用する場合は、このプロパティを設定します。
org.kie.server.token	String	該当なし	このプロパティにより、Process Automation Manager コントローラーと KIE Server 間の認証に、ユーザー名/パスワードを使用する Basic 認証ではなく、トークンベースの認証を使用できます。Process Automation Manager コントローラーは、要求ヘッダーのパラメーターとしてトークンを送信します。トークンは更新されないため、サーバーには有効期限の長いアクセストークンが必要です。
org.kie.server.location	URL	該当なし	Process Automation Manager コントローラーがこのサーバーでコールバックするのに使用する KIE Server インスタンスの URL (例: http://localhost:8230/kie-server/services/rest/server)。Process Automation Manager コントローラーを使用する場合は、このプロパティの設定が必須です。
org.kie.server.controller	コンマ区切りのリスト	該当なし	Process Automation Manager コントローラー REST エンドポイントへの URL のコンマ区切りリスト (例: http://localhost:8080/business-central/rest/controller)。Process Automation Manager コントローラーを使用する場合は、このプロパティの設定が必須です。

プロパティ	値	デフォルト	説明
org.kie.server.controller.user	String	kieserver	Process Automation Manager コントローラー REST API に接続するユーザー名。Process Automation Manager コントローラーを使用する場合は、このプロパティの設定が必須です。
org.kie.server.controller.pwd	String	kieserver1!	Process Automation Manager コントローラー REST API に接続するためのパスワード。Process Automation Manager コントローラーを使用する場合は、このプロパティの設定が必須です。
org.kie.server.controller.token	String	該当なし	このプロパティにより、Process Automation Manager コントローラーと KIE Server との間の認証に、ユーザー名/パスワードを使用する Basic 認証ではなく、トークンベースの認証を使用できます。このサーバーは、要求ヘッダーのパラメーターとしてトークンを送信します。トークンは更新されないため、サーバーには有効期限の長いアクセストークンが必要です。
org.kie.server.controller.connect	Long	10000	サーバーの起動時に KIE Server を Process Automation Manager コントローラーに接続することを試み、次に試行するまでの待機時間 (ミリ秒)。

表15.3 キーストアを読み込むためのシステムプロパティ

プロパティ	値	デフォルト	説明
kie.keystore.keyStoreURL	URL	該当なし	Java Cryptography Extension KeyStore (JCEKS) の読み込みに使用する URL。例: file:///home/kie/keystores/keystore.jceks
kie.keystore.keyStorePwd	String	該当なし	JCEKS に使用するパスワード
kie.keystore.key.server.alias	String	該当なし	パスワードの保存先となる REST サービスのキーのエイリアス名
kie.keystore.key.server.pwd	String	該当なし	REST サービスのエイリアスのパスワード

プロパティ	値	デフォルト	説明
kie.keystore.key.ctrl.alias	String	該当なし	デフォルトの REST Process Automation Manager コントローラー用のキーのエイリアス。
kie.keystore.key.ctrl.pwd	String	該当なし	デフォルトの REST Process Automation Manager コントローラー用のエイリアスのパスワード。

表15.4 トランザクションのコミットを再試行するシステムプロパティ

プロパティ	値	デフォルト	説明
org.kie.optlock.retries	Integer	5	このプロパティでは、完全に失敗となる前に、プロセスエンジンがトランザクションを再試行する回数を記述します。
org.kie.optlock.delay	Integer	50	最初の再試行までの時間 (ミリ秒単位)。
org.kie.optlock.delayFactor	Integer	4	後続の再試行ごとに遅延時間を長くする乗数。デフォルト値の場合には、プロセスエンジンは、1回目の再試行まで 50 ミリ秒、2番目の再試行まで 200 ミリ秒、3番目の再試行まで 800 ミリ秒待機します。

表15.5 その他のシステムプロパティ

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

プロパティ	値	デフォルト	説明
org.kie.server.repo	パス	.	KIE Server の状態ファイルが保存される場所
org.kie.server.sync.deploy	true、false	false	<p>KIE Server に対して、Process Automation Manager コントローラーがコンテナのデプロイメント設定を提供するまでデプロイメントを保持するように指示します。このプロパティは、管理モードで実行するサーバーのみが対象です。以下のオプションが利用できます。</p> <p>* false: Process Automation Manager コントローラーへの接続は非同期です。アプリケーションが起動して、Process Automation Manager コントローラーに接続し、成功すると、コンテナをデプロイします。アプリケーションはコンテナが利用可能になる前でもリクエストを受け付けます。* true: サーバーアプリケーションのデプロイメントは、Process Automation Manager コントローラーの接続スレッドと、メインのデプロイメントを結合し、完了するまで待機します。このオプションを使用すると、複数のアプリケーションが同じサーバー上にある場合に、デッドロックになる可能性があります。1台のサーバーで使用するアプリケーションは1つだけにしてください。</p>
org.kie.server.startup.strategy	ControllerBasedStartupStrategy、LocalContainersStartupStrategy	ControllerBasedStartupStrategy	デプロイした KIE コンテナの制御に使用する KIE Server の起動ストラテジー、およびデプロイする順番
org.kie.server.mgmt.api.disabled	true、false	false	true に設定した場合は、KIE Server 管理 API が無効になります。

プロパティ	値	デフォルト	説明
org.kie.server.xstream.enabled.packages	org.kie.example などの Java パッケージ。 org.kie.example.* のようにワイルドカード表現を指定することも可能です。	該当なし	XStream を使用してマーシャリングするための許可リストに追加パッケージを指定するプロパティ
org.kie.store.services.classes	String	org.drools.persistence.jpa.KnowledgeStoreServiceImpl	KieSession インスタンスのブートストラップを行う KieStoreServices を実装する完全修飾クラス名
org.kie.server.strict.id.format	true 、 false	false	JSON のマーシャリングを使用する際に、プロパティが true に設定されている場合は、適切な JSON 形式で応答を常に返します。たとえば、元の応答に含まれる数値が1つだけの場合、応答は JSON 形式でラップされます。たとえば、 <code>{"value": 1}</code> です。
org.kie.server.json.customObjectDeserializerCNFE Behavior	IGNORE 、 WARN 、 EXCEPTION	IGNORE	JSON アンマーシャリングを使用しているときに、ペイロード内のクラスが見つからない場合、このプロパティを使用して次のように動作を変更できます。 <ul style="list-style-type: none"> ● プロパティが IGNORE に設定されている場合、ペイロードは HashMap に変換されます ● プロパティが WARN に設定されている場合、ペイロードは HashMap に変換され、警告がログに記録されます ● プロパティが EXCEPTION に設定されている場合、KIE サーバーは例外を出力します

プロパティ	値	デフォルト	説明
org.kie.server.strict.jaxb.format	true、false	false	このプロパティの値を true に設定すると、KIE Server は REST API ペイロードのデータのタイプを検証します。たとえば、データフィールドに数字のデータタイプが指定されている場合に、数字以外が含まれているとエラーが発生します。

第16章 KIE SERVER の機能と拡張

KIE Server の機能は、ビジネスニーズに合わせて有効化、無効化、または拡張可能なプラグインにより決まります。KIE Server は以下の機能および拡張をサポートします。

表16.1 KIE Server の機能と拡張

機能名	拡張名	説明
KieServer	KieServer	サーバーインスタンスでの KIE コンテナの作成や削除など、KIE Server のコア機能を提供します。
BRM	Drools	ファクトの挿入やビジネスルールの実行など、ビジネスルール管理 (BRM) 機能を提供します。
BRP	OptaPlanner	ソルバーの実装など、ビジネスリソースプランニング (BRP) 機能を提供します。
DMN	DMN	DMN データ型の管理や DMN モデルの実行など、Decision Model and Notation (DMN) 機能を提供します。
Swagger	Swagger	KIE Server REST API と対話するための Swagger の Web インターフェイス機能を提供します。

実行中の KIE Server インスタンスに対応する拡張を表示するには、以下の REST API エンドポイントに **GET** 要求を送信して、XML または JSON サーバーの要求を確認します。

KIE Server の情報に対する GET 要求のベース URL

```
http://SERVER:PORT/kie-server/services/rest/server
```

KIE Server の情報を含む JSON 応答の例

```
{
  "type": "SUCCESS",
  "msg": "Kie Server info",
  "result": {
    "kie-server-info": {
      "id": "test-kie-server",
      "version": "7.67.0.20190818-050814",
      "name": "test-kie-server",
      "location": "http://localhost:8080/kie-server/services/rest/server",
      "capabilities": [
        "KieServer",
        "BRM",
        "BRP",
        "DMN",
        "Swagger"
      ],
    },
    "messages": [
      {
        "severity": "INFO",
      }
    ]
  }
}
```

```

    "timestamp": {
      "java.util.Date": 1566169865791
    },
    "content": [
      "Server KieServerInfo{serverId='test-kie-server', version='7.67.0.20190818-050814',
name='test-kie-server', location='http://localhost:8080/kie-server/services/rest/server', capabilities=
[KieServer, BRM, BRP, DMN, Swagger]', messages=null, mode=DEVELOPMENT}started
successfully at Sun Aug 18 23:11:05 UTC 2019"
    ]
  }
],
"mode": "DEVELOPMENT"
}
}
}
}
}

```

KIE Server 拡張機能を有効または無効にするには、関連する KIE Server システムプロパティー (`*.server.ext.disabled`) を設定します。たとえば、**BRM** 機能を無効にするには、`org.drools.server.ext.disabled=true` システムプロパティーを設定します。全 KIE Server システムプロパティーについては、[15章 KIE Server のシステムプロパティー](#) を参照してください。

KIE Server 拡張機能は、デフォルトでは REST または JMS データトランスポートで公開され、事前定義済みのクライアント API を使用します。追加の REST エンドポイントで既存の KIE Server 機能を拡張するか、REST または JMS 以外の対応のトランスポートメソッドを拡張するか、KIE Server クライアントの機能を拡張できます。

KIE Server 機能は柔軟であるため、デフォルトの KIE Server 機能にビジネスニーズを合わせるのではなく、KIE Server インスタンスをビジネスニーズに合わせて合わせることができます。



重要

KIE Server 機能を拡張した場合、Red Hat では、カスタムの実装や拡張の一部として使用したカスタムコードをサポートしません。

16.1. カスタム REST API エンドポイントを使用した既存の KIE SERVER 機能の拡張

KIE Server REST API を使用すると、Business Central のユーザーインターフェイスを使用せずに、Red Hat Decision Manager 内の KIE コンテナとビジネスアセット (ビジネスルール、プロセス、ソルバーなど) と対話できます。利用可能な REST エンドポイントは、KIE Server システムプロパティーで有効にした機能により決まります (例: **BRM** 機能は `org.drools.server.ext.disabled=false`)。既存の KIE Server 機能は、カスタムの REST API エンドポイントで拡張し、ビジネスニーズに合わせて KIE Server REST API を適合できます。

たとえば、この手順では、以下のカスタム REST API エンドポイントで **Drools** KIE Server 機能 (**BRM** 機能向け) を拡張します。

カスタム REST API エンドポイントの例

```
/server/containers/instances/{containerId}/ksession/{ksessionId}
```

このカスタムのエンドポイントの例では、デシジョンエンジンの作業メモリーに挿入するファクトリストを受け入れ、自動的に全ルールを実行して、指定の KIE コンテナで KIE セッションからのオブジェクトをすべて取得します。

手順

1. 空の Maven プロジェクトを作成して、以下のパッケージタイプと依存関係を、プロジェクトの **pom.xml** ファイルに定義します。

サンプルプロジェクトの pom.xml ファイルの例

```
<packaging>jar</packaging>

<properties>
  <version.org.kie>7.67.0.Final-redhat-00024</version.org.kie>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-internal</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-common</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-drools</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-rest-common</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-core</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-compiler</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
```

```

<artifactId>slf4j-api</artifactId>
<version>1.7.25</version>
</dependency>
</dependencies>

```

2. 以下の例のように、プロジェクトの Java クラスに **org.kie.server.services.api.KieServerApplicationComponentsService** インターフェイスを実装します。

KieServerApplicationComponentsService インターフェイスの実装例

```

public class CusomtDroolsKieServerApplicationComponentsService implements
KieServerApplicationComponentsService { ❶

    private static final String OWNER_EXTENSION = "Drools"; ❷

    public Collection<Object> getAppComponents(String extension, SupportedTransports
type, Object... services) { ❸
        // Do not accept calls from extensions other than the owner extension:
        if ( !OWNER_EXTENSION.equals(extension) ) {
            return Collections.emptyList();
        }

        RulesExecutionService rulesExecutionService = null; ❹
        KieServerRegistry context = null;

        for( Object object : services ) {
            if( RulesExecutionService.class.isAssignableFrom(object.getClass()) ) {
                rulesExecutionService = (RulesExecutionService) object;
                continue;
            } else if( KieServerRegistry.class.isAssignableFrom(object.getClass()) ) {
                context = (KieServerRegistry) object;
                continue;
            }
        }

        List<Object> components = new ArrayList<Object>(1);
        if( SupportedTransports.REST.equals(type) ) {
            components.add(new CustomResource(rulesExecutionService, context)); ❺
        }

        return components;
    }
}

```

- ❶ アプリケーションの起動時にデプロイされる KIE Server インフラストラクチャーに REST エンドポイントを提供します。
- ❷ この例の **Drools** 拡張など、拡張する機能を指定します。
- ❸ REST コンテナがデプロイする必要のある全リソースを返します。KIE Server インスタンスで有効化した各拡張で、**getAppComponents** メソッドを呼び出して、指定した **OWNER_EXTENSION** 以外の拡張の空のコレクションを、**if (!OWNER_EXTENSION.equals(extension))** の呼び出しで返します。

- 4 この例の **Drools** 拡張の **RulesExecutionService** や **KieServerRegistry** サービスなど、指定の拡張から使用するサービスを表示します。
 - 5 **components** リストの一部としてリソースを返す **CustomResource** クラスと、拡張のトランスポートタイプを **REST** または **JMS** に指定します (この例では **REST**)。
3. 以下の例のように、KIE Server を使用して新規の REST リソースの機能を追加する **CustomResource** クラスを実装します。

CustomResource クラスの実装例

```
// Custom base endpoint:
@Path("server/containers/instances/{containerId}/ksession")
public class CustomResource {

    private static final Logger logger = LoggerFactory.getLogger(CustomResource.class);

    private KieCommands commandsFactory = KieServices.Factory.get().getCommands();

    private RulesExecutionService rulesExecutionService;
    private KieServerRegistry registry;

    public CustomResource() {

    }

    public CustomResource(RulesExecutionService rulesExecutionService, KieServerRegistry
registry) {
        this.rulesExecutionService = rulesExecutionService;
        this.registry = registry;
    }

    // Supported HTTP method, path parameters, and data formats:
    @POST
    @Path("/{ksessionId}")
    @Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public Response insertFireReturn(@Context HttpHeaders headers,
        @PathParam("containerId") String id,
        @PathParam("ksessionId") String ksessionId,
        String cmdPayload) {

        Variant v = getVariant(headers);
        String contentType = getContentType(headers);

        // Marshalling behavior and supported actions:
        MarshallingFormat format = MarshallingFormat.fromType(contentType);
        if (format == null) {
            format = MarshallingFormat.valueOf(contentType);
        }
        try {
            KieContainerInstance kci = registry.getContainer(id);

            Marshaller marshaller = kci.getMarshaller(format);
```

```

List<?> listOfFacts = marshaller.unmarshall(cmdPayload, List.class);

List<Command<?>> commands = new ArrayList<Command<?>>();
BatchExecutionCommand executionCommand =
commandsFactory.newBatchExecution(commands, ksessionId);

for (Object fact : listOfFacts) {
    commands.add(commandsFactory.newInsert(fact, fact.toString()));
}
commands.add(commandsFactory.newFireAllRules());
commands.add(commandsFactory.newGetObjects());

ExecutionResults results = rulesExecutionService.call(kci, executionCommand);

String result = marshaller.marshall(results);

logger.debug("Returning OK response with content '{}'", result);
return createResponse(result, v, Response.Status.OK);
} catch (Exception e) {
    // If marshalling fails, return the `call-container` response to maintain backward
compatibility:
    String response = "Execution failed with error : " + e.getMessage();
    logger.debug("Returning Failure response with content '{}'", response);
    return createResponse(response, v,
Response.Status.INTERNAL_SERVER_ERROR);
}
}
}

```

この例では、カスタムエンドポイントの **CustomResource** クラスで、以下のデータと動作を指定します。

- **server/containers/instances/{containerId}/ksession** のベースポイントを使用します。
- **POST** HTTP メソッドを使用します。
- REST 要求で以下のデータを指定する必要があります。
 - パスの引数として **containerId**
 - パスの引数として **ksessionId**
 - メッセージペイロードとしてファクトのリスト
- 全 KIE Server データ形式をサポートします。
 - XML (JAXB、XStream)
 - JSON
- **List<?>** コレクションにペイロードをアンマーシャリングして、リスト内のアイテムごとに、**InsertCommand** インスタンスを作成し、その後に **FireAllRules** と **GetObject** コマンドを追加します。

- デンジョンエンジンを呼び出す **BatchExecutionCommand** インスタンスに全コマンドを追加します。
4. 新規エンドポイントを KIE Server で検出できるようにするには、Maven プロジェクト内に **META-INF/services/org.kie.server.services.api.KieServerApplicationComponentsService** ファイルを作成して、このファイルに **KieServerApplicationComponentsService** 実装クラスの完全修飾名を追加します。たとえば、このファイルには、**org.kie.server.ext.drools.rest.CusomtDroolsKieServerApplicationComponentsService** の 1 行が含まれます。
 5. プロジェクトを構築して、作成された JAR ファイルをプロジェクトの `~/kie-server.war/WEB-INF/lib` ディレクトリーにコピーします。たとえば、Red Hat JBoss EAP ではこのディレクトリーへのパスは **EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib** です。
 6. KIE Server を起動して、実行中の KIE Server に構築したプロジェクトをデプロイします。プロジェクトは、Business Central インターフェイスまたは KIE Server REST API (**http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}** への **PUT** 要求) を使用してデプロイできます。実行中の KIE Server にプロジェクトを追加した後に、新しい REST エンドポイントとの対話を開始します。

今回の例では、以下の情報を使用して新規エンドポイントを呼び出すことができます。

- 要求 URL 例: **http://localhost:8080/kie-server/services/rest/server/containers/instances/demo/ksession/defaultKieSession**
- HTTP メソッド: **POST**
- HTTP ヘッダー:
 - **Content-Type: application/json**
 - **Accept: application/json**
- メッセージペイロードの例:

```
[
  {
    "org.jbpm.test.Person": {
      "name": "john",
      "age": 25
    }
  },
  {
    "org.jbpm.test.Person": {
      "name": "mary",
      "age": 22
    }
  }
]
```

- サーバーの応答例: **200** (success)
- サーバーのログ出力例:

```
13:37:20,347 INFO [stdout] (default task-24) Hello mary
13:37:20,348 INFO [stdout] (default task-24) Hello john
```


16.2. カスタムデータトランスポートを使用するための KIE SERVER の拡張

デフォルトでは、KIE Server の拡張が REST または JMS データトランスポートを使用して公開されません。KIE Server を拡張して、カスタムのデータトランスポートのサポートを追加し、KIE Server トランスポートプロトコルをビジネスニーズに合わせます。

たとえば、以下の手順では、**Drools** 拡張を使用し、Apache MINA (オープンソースの Java ネットワークアプリケーションフレームワーク) をベースとする KIE Server にカスタムのデータトランスポートを追加します。カスタムの MINA トランスポートの例では、既存のマーシャリング操作に依存し、JSON 形式のみをサポートする文字列ベースのデータを変換します。

手順

1. 空の Maven プロジェクトを作成して、以下のパッケージタイプと依存関係を、プロジェクトの **pom.xml** ファイルに定義します。

サンプルプロジェクトの pom.xml ファイルの例

```
<packaging>jar</packaging>

<properties>
  <version.org.kie>7.67.0.Final-redhat-00024</version.org.kie>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-internal</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-common</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-services-drools</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-core</artifactId>
    <version>${version.org.kie}</version>
```

```

</dependency>
<dependency>
  <groupId>org.drools</groupId>
  <artifactId>drools-compiler</artifactId>
  <version>${version.org.kie}</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.25</version>
</dependency>
<dependency>
  <groupId>org.apache.mina</groupId>
  <artifactId>mina-core</artifactId>
  <version>2.1.3</version>
</dependency>
</dependencies>

```

2. 以下の例のように、プロジェクトの Java クラスに **org.kie.server.services.api.KieServerExtension** インターフェイスを実装します。

KieServerExtension インターフェイスの実装例

```

public class MinaDroolsKieServerExtension implements KieServerExtension {

    private static final Logger logger =
    LoggerFactory.getLogger(MinaDroolsKieServerExtension.class);

    private static final String EXTENSION_NAME = "Drools-Mina";

    private static final Boolean disabled =
    Boolean.parseBoolean(System.getProperty("org.kie.server.drools-mina.ext.disabled",
    "false"));
    private static final String MINA_HOST = System.getProperty("org.kie.server.drools-
    mina.ext.port", "localhost");
    private static final int MINA_PORT =
    Integer.parseInt(System.getProperty("org.kie.server.drools-mina.ext.port", "9123"));

    // Taken from dependency on the `Drools` extension:
    private KieContainerCommandService batchCommandService;

    // Specific to MINA:
    private IoAcceptor acceptor;

    public boolean isActive() {
        return disabled == false;
    }

    public void init(KieServerImpl kieServer, KieServerRegistry registry) {

        KieServerExtension droolsExtension = registry.getServerExtension("Drools");
        if (droolsExtension == null) {
            logger.warn("No Drools extension available, quitting...");
            return;
        }
    }
}

```

```

List<Object> droolsServices = droolsExtension.getServices();
for( Object object : droolsServices ) {
    // If the given service is null (not configured), continue to the next service:
    if (object == null) {
        continue;
    }
    if( KieContainerCommandService.class.isAssignableFrom(object.getClass()) ) {
        batchCommandService = (KieContainerCommandService) object;
        continue;
    }
}
if (batchCommandService != null) {
    acceptor = new NioSocketAcceptor();
    acceptor.getFilterChain().addLast( "codec", new ProtocolCodecFilter( new
    TextLineCodecFactory( Charset.forName( "UTF-8" ) ) ) );

    acceptor.setHandler( new TextBasedIoHandlerAdapter(batchCommandService) );
    acceptor.getSessionConfig().setReadBufferSize( 2048 );
    acceptor.getSessionConfig().setIdleTime( IdleStatus.BOTH_IDLE, 10 );
    try {
        acceptor.bind( new InetSocketAddress(MINA_HOST, MINA_PORT) );

        logger.info("{} -- Mina server started at {} and port {}", toString(), MINA_HOST,
MINA_PORT);
    } catch (IOException e) {
        logger.error("Unable to start Mina acceptor due to {}", e.getMessage(), e);
    }
}

}

public void destroy(KieServerImpl kieServer, KieServerRegistry registry) {
    if (acceptor != null) {
        acceptor.dispose();
        acceptor = null;
    }
    logger.info("{} -- Mina server stopped", toString());
}

public void createContainer(String id, KieContainerInstance kieContainerInstance,
Map<String, Object> parameters) {
    // Empty, already handled by the `Drools` extension
}

public void disposeContainer(String id, KieContainerInstance kieContainerInstance,
Map<String, Object> parameters) {
    // Empty, already handled by the `Drools` extension
}

public List<Object> getAppComponents(SupportedTransports type) {
    // Nothing for supported transports (REST or JMS)
    return Collections.emptyList();
}

```

```

public <T> T getAppComponents(Class<T> serviceType) {

    return null;
}

public String getImplementedCapability() {
    return "BRM-Mina";
}

public List<Object> getServices() {
    return Collections.emptyList();
}

public String getExtensionName() {
    return EXTENSION_NAME;
}

public Integer getStartOrder() {
    return 20;
}

@Override
public String toString() {
    return EXTENSION_NAME + " KIE Server extension";
}
}

```

KieServerExtension インターフェイスは、新規の MINA トランスポートの機能を追加する時に KIE Server が使用する主要な拡張インターフェイスです。このインターフェイスには、以下のコンポーネントが含まれます。

KieServerExtension インターフェイスの概要

```

public interface KieServerExtension {

    boolean isActive();

    void init(KieServerImpl kieServer, KieServerRegistry registry);

    void destroy(KieServerImpl kieServer, KieServerRegistry registry);

    void createContainer(String id, KieContainerInstance kieContainerInstance, Map<String, Object> parameters);

    void disposeContainer(String id, KieContainerInstance kieContainerInstance, Map<String, Object> parameters);

    List<Object> getAppComponents(SupportedTransports type);

    <T> T getAppComponents(Class<T> serviceType);

    String getImplementedCapability(); 1

    List<Object> getServices();
}

```

```
String getExtensionName(); ❷

Integer getStartOrder(); ❸
}
```

- ❶ この拡張で対応している機能を指定します。この機能は、KIE Server 内で一意でなければなりません。
- ❷ 拡張は、人間が解読可能な名前に定義します。
- ❸ 指定した拡張の起動のタイミングを決定します。他の拡張と依存関係がある拡張の場合、この設定は親の設定と競合しないようにしてください。たとえば、今回の場合、このカスタムの拡張は **Drools** 拡張に依存しており、Drool 拡張の **StartOrder** は **0** に設定されているため、このカスタムのアドオン拡張は **0** を超える値でなければなりません (サンプルの実装では **20** に設定)。

このインターフェイスの先程の **MinaDroolsKieServerExtension** 実装例では、**init** メソッドが主に、**Drools** 拡張からサービスを収集して、MINA サーバーをブートストラップ化する要素となっています。**KieServerExtension** インターフェイスの他のメソッドは、標準の実装のままで、インターフェイスの要件を満たします。

TextBasedIoHandlerAdapter クラスは、受信要求に対応する MINA サーバーにあるハンドラーです。

3. 以下の例のように、MINA サーバーの **TextBasedIoHandlerAdapter** ハンドラーを実装します。

TextBasedIoHandlerAdapter ハンドラーの実装例

```
public class TextBasedIoHandlerAdapter extends IoHandlerAdapter {

    private static final Logger logger =
        LoggerFactory.getLogger(TextBasedIoHandlerAdapter.class);

    private KieContainerCommandService batchCommandService;

    public TextBasedIoHandlerAdapter(KieContainerCommandService
        batchCommandService) {
        this.batchCommandService = batchCommandService;
    }

    @Override
    public void messageReceived( IoSession session, Object message ) throws Exception {
        String completeMessage = message.toString();
        logger.debug("Received message '{}'", completeMessage);
        if( completeMessage.trim().equalsIgnoreCase("quit") ||
            completeMessage.trim().equalsIgnoreCase("exit") ) {
            session.close(false);
            return;
        }

        String[] elements = completeMessage.split("\\|");
        logger.debug("Container id {}", elements[0]);
        try {
            ServiceResponse<String> result = batchCommandService.callContainer(elements[0],
```


この例では、Telnet を使用して KIE Server の新しい MINA ベースのデータトランスポートと対話できます。

コマンド端末での Telnet の開始およびポート 9123 での KIE Server の接続

```
telnet 127.0.0.1 9123
```

コマンド端末での KIE Server との対話例

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.

# Request body:
demo|{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"john","age":25}}},"fire-all-rules":""}]

# Server response:
{
  "results" : [ {
    "key" : "",
    "value" : 1
  } ],
  "facts" : [ ]
}

demo|{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"mary","age":22}}},"fire-all-rules":""}]

{
  "results" : [ {
    "key" : "",
    "value" : 1
  } ],
  "facts" : [ ]
}

demo|{"lookup":"defaultKieSession","commands":[{"insert":{"object":{"org.jbpm.test.Person":{"name":"james","age":25}}},"fire-all-rules":""}]

{
  "results" : [ {
    "key" : "",
    "value" : 1
  } ],
  "facts" : [ ]
}
exit
Connection closed by foreign host.
```

サーバーログの出力例

```
16:33:40,206 INFO [stdout] (NioProcessor-2) Hello john
16:34:03,877 INFO [stdout] (NioProcessor-2) Hello mary
16:34:19,800 INFO [stdout] (NioProcessor-2) Hello james
```

16.3. カスタムクライアント API を使用した KIE SERVER のクライアント拡張

KIE Server は、KIE Server サービスの使用時に対話可能な、事前定義済みのクライアント API を使用します。カスタムのクライアント API で KIE Server クライアントを拡張して、ビジネスのニーズに KIE Server サービスを適合させます。

たとえば、以下の手順では、カスタムのクライアント API を KIE Server に追加して、Apache MINA (オープンソースの Java ネットワークアプリケーションフレームワーク) をもとにした、カスタムのデータトランスポートに対応します (このシナリオ向けにすでに設定済み)。

手順

1. 空の Maven プロジェクトを作成して、以下のパッケージタイプと依存関係を、プロジェクトの **pom.xml** ファイルに定義します。

サンプルプロジェクトの pom.xml ファイルの例

```
<packaging>jar</packaging>

<properties>
  <version.org.kie>7.67.0.Final-redhat-00024</version.org.kie>
</properties>

<dependencies>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-api</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.kie.server</groupId>
    <artifactId>kie-server-client</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
  <dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-compiler</artifactId>
    <version>${version.org.kie}</version>
  </dependency>
</dependencies>
```

2. 以下の例のように、プロジェクトの Java クラスに、関連する **ServicesClient** インターフェイスを実装します。

RulesMinaServicesClient インターフェイスの例

```
public interface RulesMinaServicesClient extends RuleServicesClient {

}
```

インターフェイスをもとにクライアントの実装を登録する必要があるため、特定のインターフェイスが必要です。また、指定のインターフェイスには実装は1つしか指定できません。

この例では、カスタムの MINA ベースのデータ転送ポートが **Drools** 拡張を使用し、この **RulesMinaServicesClient** インターフェイスの例は、**Drools** 拡張から、既存の **RuleServicesClient** クライアント API を拡張します。

- 以下の例のように、新規の MINA 転送ポートのクライアント機能を追加するのに KIE Server が使用可能な **RulesMinaServicesClient** インターフェイスを実装します。

RulesMinaServicesClient インターフェイスの実装例

```
public class RulesMinaServicesClientImpl implements RulesMinaServicesClient {

    private String host;
    private Integer port;

    private Marshaller marshaller;

    public RulesMinaServicesClientImpl(KieServicesConfiguration configuration, ClassLoader
classloader) {
        String[] serverDetails = configuration.getServerUrl().split(":");

        this.host = serverDetails[0];
        this.port = Integer.parseInt(serverDetails[1]);

        this.marshaller = MarshallerFactory.getMarshaller(configuration.getExtraJaxbClasses(),
MarshallingFormat.JSON, classloader);
    }

    public ServiceResponse<String> executeCommands(String id, String payload) {

        try {
            String response = sendReceive(id, payload);
            if (response.startsWith("{")) {
                return new ServiceResponse<String>(ResponseType.SUCCESS, null, response);
            } else {
                return new ServiceResponse<String>(ResponseType.FAILURE, response);
            }
        } catch (Exception e) {
            throw new KieServicesException("Unable to send request to KIE Server", e);
        }
    }

    public ServiceResponse<String> executeCommands(String id, Command<?> cmd) {
        try {
            String response = sendReceive(id, marshaller.marshall(cmd));
            if (response.startsWith("{")) {
                return new ServiceResponse<String>(ResponseType.SUCCESS, null, response);
            } else {
                return new ServiceResponse<String>(ResponseType.FAILURE, response);
            }
        } catch (Exception e) {
            throw new KieServicesException("Unable to send request to KIE Server", e);
        }
    }

    protected String sendReceive(String containerId, String content) throws Exception {
```

```

// Flatten the content to be single line:
content = content.replaceAll("\n", "");

Socket minaSocket = null;
PrintWriter out = null;
BufferedReader in = null;

StringBuffer data = new StringBuffer();
try {
    minaSocket = new Socket(host, port);
    out = new PrintWriter(minaSocket.getOutputStream(), true);
    in = new BufferedReader(new InputStreamReader(minaSocket.getInputStream()));

    // Prepare and send data:
    out.println(containerId + "|" + content);
    // Wait for the first line:
    data.append(in.readLine());
    // Continue as long as data is available:
    while (in.ready()) {
        data.append(in.readLine());
    }

    return data.toString();
} finally {
    out.close();
    in.close();
    minaSocket.close();
}
}
}

```

この実装例は、以下のデータおよび動作を指定します。

- ソケットベースの通信を使用して簡素化します。
 - KIE Server クライアントのデフォルト設定に依存し、**ServerUrl** を使用して MINA サーバーのホストとポートを提供します。
 - マーシャリング形式で JSON を指定します。
 - 受信メッセージは左波括弧 { で始まる JSON オブジェクトでなければなりません。
 - 応答の最初の行を待機中に、ブロッキング API と直接、ソケット通信を使用してから、利用可能なすべての行を読み取ります。
 - **ストリームモード** を使用しないため、コマンドの呼び出し後に KIE Server セッションを切断します。
4. 以下の例のように、プロジェクトの Java クラスに **org.kie.server.client.helper.KieServicesClientBuilder** インターフェイスを実装します。

KieServicesClientBuilder インターフェイスの実装例

```

public class MinaClientBuilderImpl implements KieServicesClientBuilder { 1

    public String getImplementedCapability() { 2

```

```

        return "BRM-Mina";
    }

    public Map<Class<?>, Object> build(KieServicesConfiguration configuration, ClassLoader
classLoader) { ③
        Map<Class<?>, Object> services = new HashMap<Class<?>, Object>();

        services.put(RulesMinaServicesClient.class, new
RulesMinaServicesClientImpl(configuration, classLoader));

        return services;
    }
}

```

- ① 一般の KIE Server クライアントインフラストラクチャーにクライアント API を追加できません。
 - ② クライアントが使用する KIE Server 機能 (拡張) を定義します。
 - ③ クライアントの実装のマッピングを提供します。キーはインターフェイス、値は完全な初期実装です。
5. 新規のクライアント API を KIE Server クライアントで検出できるようにするには、Maven プロジェクトで **META-INF/services/org.kie.server.client.helper.KieServicesClientBuilder** ファイルを作成し、このファイルに **KieServicesClientBuilder** 実装クラスの完全修飾名を追加します。たとえば、このファイルには **org.kie.server.ext.mina.client.MinaClientBuilderImpl** の 1 行が含まれます。
 6. プロジェクトを構築して、作成された JAR ファイルをプロジェクトの **~/kie-server.war/WEB-INF/lib** ディレクトリーにコピーします。たとえば、Red Hat JBoss EAP ではこのディレクトリーへのパスは **EAP_HOME/standalone/deployments/kie-server.war/WEB-INF/lib** です。
 7. KIE Server を起動して、実行中の KIE Server に構築したプロジェクトをデプロイします。プロジェクトは、Business Central インターフェイスまたは KIE Server REST API (**http://SERVER:PORT/kie-server/services/rest/server/containers/{containerId}** への **PUT** 要求) を使用してデプロイできます。
 実行中の KIE Server にプロジェクトをデプロイしたあとに、新規の KIE Server クライアントと対話を開始できます。標準の KIE Server クライアントと同じ方法で、クライアント設定とクライアントインスタンスを作成して、タイプ別にサービスクライアントを取得し、クライアントメソッドを呼び出して、新しいクライアントを使用します。

たとえば、**RulesMinaServiceClient** クライアントインスタンスを作成して、MINA トランスポートを使用して KIE Server で操作を呼び出すことができます。

RulesMinaServiceClient クライアント作成の実装例

```

protected RulesMinaServicesClient buildClient() {
    KieServicesConfiguration configuration =
KieServicesFactory.newRestConfiguration("localhost:9123", null, null);
    List<String> capabilities = new ArrayList<String>();
    // Explicitly add capabilities (the MINA client does not respond to `get-server-info`
requests):
    capabilities.add("BRM-Mina");
}

```

```

configuration.setCapabilities(capabilities);
configuration.setMarshallingFormat(MarshallingFormat.JSON);

configuration.addJaxbClasses(extraClasses);

KieServicesClient kieServicesClient =
KieServicesFactory.newKieServicesClient(configuration);

RulesMinaServicesClient rulesClient =
kieServicesClient.getServicesClient(RulesMinaServicesClient.class);

return rulesClient;
}

```

MINA トランスポートを使用して KIE Server 上で操作を呼び出す設定例

```

RulesMinaServicesClient rulesClient = buildClient();

List<Command<?>> commands = new ArrayList<Command<?>>();
BatchExecutionCommand executionCommand =
commandsFactory.newBatchExecution(commands, "defaultKieSession");

Person person = new Person();
person.setName("mary");
commands.add(commandsFactory.newInsert(person, "person"));
commands.add(commandsFactory.newFireAllRules("fired"));

ServiceResponse<String> response = rulesClient.executeCommands(containerId,
executionCommand);
Assert.assertNotNull(response);

Assert.assertEquals(ResponseType.SUCCESS, response.getType());

String data = response.getResult();

Marshaller marshaller = MarshallerFactory.getMarshaller(extraClasses,
MarshallingFormat.JSON, this.getClass().getClassLoader());

ExecutionResultImpl results = marshaller.unmarshall(data, ExecutionResultImpl.class);
Assert.assertNotNull(results);

Object personResult = results.getValue("person");
Assert.assertTrue(personResult instanceof Person);

Assert.assertEquals("mary", ((Person) personResult).getName());
Assert.assertEquals("JBoss Community", ((Person) personResult).getAddress());
Assert.assertEquals(true, ((Person) personResult).isRegistered());

```

第17章 KIE SERVER の使用時のパフォーマンスチューニングに関する考慮点

以下の主要な概念または推奨のプラクティスを使用すると、KIE Server のパフォーマンスの最適化に役立ちます。このセクションではこの概念についてまとめており、随時、他のドキュメントを相互参照して詳細を説明します。本セクションは、Red Hat Decision Manager の新しいリリースで、必要に応じて拡張または変更します。

開発時には、必ず開発モードを有効にする

KIE Server または特定のプロジェクトを Business Central に設定して、**production** モードまたは **development** モードを使用できます。デフォルトでは、KIE Server および Business Central のすべての新規プロジェクトは開発モードになっています。このモードには、プロジェクトの開発ポリシーに柔軟性をもたせるなど、開発が容易にすすむ機能や、重複した GAV の検出を無効化するなど、開発中の KIE Server のパフォーマンスを最適化する機能が含まれます。Red Hat Decision Manager 環境が確立し、実稼働モードを実行できる準備が完全に整うまで、開発モードを使用してください。

環境モードの設定または重複する GAV の検出の詳細は、以下の資料を参照してください。

- [7章 KIE Server および Business Central での環境モードの設定](#)
- [Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ](#)

KIE Server 機能と拡張を特定のニーズに適合すること

KIE Server の機能は、ビジネスニーズに合わせて有効化、無効化、または拡張可能なプラグインにより決まります。KIE Server 拡張機能は、デフォルトでは REST または JMS データトランスポートで公開され、事前定義済みのクライアント API を使用します。追加の REST エンドポイントで既存の KIE Server 機能を拡張するか、REST または JMS 以外の対応のトランスポートメソッドを拡張するか、KIE Server クライアントの機能を拡張できます。

KIE Server 機能は柔軟であるため、デフォルトの KIE Server 機能にビジネスニーズを合わせるのではなく、KIE Server インスタンスをビジネスニーズに合わせることもできます。

KIE Server の機能の有効化、無効化、または拡張に関する詳細は、[16章 KIE Server の機能と拡張](#) を参照してください。

第18章 関連情報

- [Red Hat JBoss EAP 7.4 への Red Hat Decision Manager のインストールおよび設定](#)
- [Red Hat Decision Manager インストールの計画](#)
- [Red Hat JBoss EAP 7.4 への Red Hat Decision Manager のインストールおよび設定](#)
- [Operator を使用した Red Hat OpenShift Container Platform 4 への Red Hat Decision Manager 環境のデプロイメント](#)
- [テンプレートを使用した Red Hat OpenShift Container Platform 3 への Red Hat Decision Manager 環境のデプロイメント](#)

パート II. BUSINESS CENTRAL 設定およびプロパティの設定

管理者は、管理者の **Settings** ページで以下をカスタマイズできます。

- **Roles:** ロールのホームページ、優先順位、パーミッションを設定します。
- **Groups:** グループのホームページ、優先順位、およびパーミッションを設定し、グループの作成および削除を行います。
- **Users:** ユーザーの作成や削除、ユーザーに対するグループおよびロールの追加または削除、およびユーザーパーミッションの表示を行います。
- **Artifacts:** M2 リポジトリーアーティファクトの表示、アーティファクトのアップロード、JAR ファイルの表示とダウンロードを行います。
- **Data Sources:** データソースおよびデータベースドライバーの追加、更新、または削除を行います。
- **Data Sets:** データセットの作成、変更、または削除を行います。
- **Projects:** ファイルエクスポートプロパティ、スペースプロパティ、デフォルト値、GAV 詳細プロパティなどプロジェクトの設定を表示し、編集します。
- **Artifact Repository:** アーティファクトリポジトリプロパティを管理します。
- **Artifact Repository:** Business Central の言語を設定します。
- **Process Administration:** Business Central でデフォルトのページネーションオプションを設定します。
- **Process Designer:** ダイアグラムエディタープロパティを設定します。
- **SSH Keys:** SSH キーを追加または削除します。
- **Custom Tasks Administration:** デフォルトのサービスタスクを有効または無効にして、カスタムのサービスタスクをアップロードします。
- **Dashbuilder データ転送:** Business Central の ZIP ファイルとして Dashbuilder データをインポートおよびエクスポートします。
- **Profiles:** ワークベンチプロファイルを **Planner and Rules** または **Full** に設定します。
- **Archetype:** アーキタイプの表示、追加、検証、デフォルト設定、および削除を行います。Business Central で新規プロジェクトを作成するときにテンプレートとして使用します。

前提条件

- Red Hat JBoss Enterprise Application Platform 7.4.1 がインストールされている。インストールの詳細は、[Red Hat JBoss Enterprise Application Platform 7.4 Installation Guide](#)を参照してください。
- Red Hat Decision Manager がインストールされ、実行中である。詳細は、[Red Hat JBoss EAP 7.4 への Red Hat Decision Manager のインストールおよび設定](#)を参照してください。
- Business Central に **admin** ユーザーロールでログインします。

第19章 ユーザーおよびグループの管理

Business Central は、ユーザー、グループ、およびロールのセキュリティー管理のエンティティーを 3 種類定義します。パーミッションは、ロールにもグループにも両方割り当てることができます。Business Central には以下のロールがあります。

- process-admin
- manager
- admin
- analyst
- rest-all
- developer
- rest-project
- user



注記

ロールレジストリーというアプリケーションのユーザーロールには、ロールの識別子がありますが、ユーザーグループにはありません。

Business Central を使用して、必要数のユーザーとグループを作成し、管理できます。ユーザーは、最低でもユーザー固有のロールを 1 つ以上割り当てて、Business Central にログインできるようにする必要があります。ユーザーの権限は、ユーザーが所属するグループとロールからのパーミッションにより異なります。ユーザーに複数のロールまたはグループが割り当てられている場合は、ロールまたはグループの優先順位が考慮される点に注意してください。

19.1. ユーザーの作成

ユーザーの権限および設定は、ユーザーに割り当てたロールと、ユーザーが属するグループで制御されます。Business Central で、ユーザーをいくつでも作成できます。



注記

プロセスエンジンまたは KIE Server で **unknown** という名前のユーザーは作成しないでください。**unknown** ユーザーアカウントは、superuser のアクセス権限があるシステム名用に予約されています。**unknown** ユーザーアカウントでは、ログインしているユーザーがない場合に、SLA 違反リスナーに関連するタスクを実行します。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Users** を選択します。
2. **New user** をクリックし、ユーザー名を入力し、**Next** をクリックします。
3. ユーザーにロールを割り当てるには、**Roles** タブをクリックして、**Add Roles** をクリックし、任意のロールを選択してから、**Add to selected roles** をクリックします。

4. 必要に応じて、ユーザーにグループを割り当てるには、**Groups** タブをクリックして **Add to groups** をクリックし、任意のグループを選択してから **Add to selected groups** をクリックします。
5. **Create** をクリックします。
6. **Yes** をクリックして、ユーザーにパスワードを設定し、**Change** をクリックします。



注記

ユーザーが Business Central にアクセスするために1つ以上のロールが必要です。

19.2. ユーザーの編集

Business Central **Settings** ページの **Users** オプションを使用して、ユーザーのグループとロールを変更できます。全ユーザーパーミッションは、ユーザーのグループとロールパーミッションを元にしています。ユーザーのパーミッションは、**Permissions** タブから確認できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Users** を選択します。
2. **All users** リストから、編集するユーザーをクリックします。右のペインにユーザーの情報が表示されます。
3. **Edit** をクリックして、以下のタスクを実行します。
 - ユーザーのグループを変更するには、**Groups** タブをクリックして、**Add to groups** をクリックします。次に、ユーザーを所属させるグループを選択して **Add to selected groups** をクリックし、**Save** をクリックします。
 - ユーザーのロールを変更するには、**Roles** タブをクリックし、**Add roles** をクリックします。ユーザーに割り当てるロールを選択して、**Add to selected roles** をクリックし、**Save** をクリックします。
 - ユーザーパーミッションを表示するには、**Permissions** タブをクリックして属性をデプロイメントします。
 - パスワードを変更するには、**Change Password** をクリックして、新規パスワードを入力し、**Change** をクリックします。
 - ユーザーを削除するには、**Delete** をクリックしてから、**Yes** をクリックし、削除を確定します。

19.3. グループの作成

Business Central では、グループを使用してユーザーをまとめて、パーミッションを制御できます。任意の数のグループを作成できますが、グループには最低でもユーザーを1つ所属させる必要があります。



注記

Business Central が Red Hat Single Sign-On (RH-SSO) を使用する場合には、Business Central のグループは RH-SSO のロールから作成されます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Groups** を選択します。
2. **New group** をクリックし、グループ名を入力し、**Next** をクリックします。
3. このグループに追加するユーザーを選択し、**Add selected users** をクリックします。
新規作成したグループは、**All groups** に表示されます。

19.4. グループの編集

必要に応じて、ホームページ、優先順位、パーミッションなどグループの属性を編集できます。Business Central の **Settings** の **Groups** オプションから、グループの変更や削除が可能です。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Groups** を選択します。
2. **All groups** リストから、編集するグループをクリックします。右のペインにユーザーの情報が表示されます。
3. **Home Page** リストからホームページを選択します。
4. **Priority** リストから優先順位を選択します。
5. **Permissions** セクションから、リソース属性をデプロイメントして、パーミッションを変更します。



注記

Pages パーミッション、**Editor** パーミッション、**Spaces** パーミッション、および **Projects** パーミッションに例外を追加できます。

6. **Save** をクリックして変更を適用します。

第20章 セキュリティー管理

セキュリティー管理とは、ユーザー、グループ、パーミッションを管理するプロセスです。Business Central セキュリティー管理ページから Business Central のリソースおよび機能へのアクセスを制御できます。

Business Central は、ユーザー、グループ、およびロールのセキュリティー管理のエンティティを 3 種類定義します。パーミッションは、ロールにもグループにも両方割り当てることができます。ユーザーは、所属するグループおよびロールのパーミッションを継承します。

20.1. セキュリティー管理プロバイダー

レルムは、セキュリティー管理のコンテキストで各種アプリケーションリソースへのアクセスを制限します。レルムには、ユーザー、グループ、ロール、パーミッションに関する情報が含まれます。特定のレルムに対する具体的なユーザーおよびグループ管理サービスの実装は、セキュリティー管理プロバイダーと呼ばれます。

組み込みのセキュリティー管理プロバイダーがアプリケーションセキュリティーレルムの要件を満たさない場合は、独自のセキュリティー管理プロバイダーを構築して登録できます。



注記

セキュリティー管理プロバイダーがインストールされていない場合は、セキュリティーレルムを管理するユーザーインターフェイスは利用できません。セキュリティー管理プロバイダーをインストールして設定した後に、セキュリティー管理ユーザーインターフェイスでユーザーおよびグループの管理機能は自動的に有効になります。

Business Central には、**application-users.properties** または **application-roles.properties** プロパティファイルの内容を基にレルムタイプをサポートする Red Hat JBoss EAP セキュリティー管理プロバイダーが含まれます。

20.1.1. プロパティファイルを基にした Red Hat JBoss EAP セキュリティー管理プロバイダーの設定

独自の Red Hat JBoss EAP セキュリティー管理プロバイダーを構築して登録できます。プロパティファイルを基にして Red Hat JBoss EAP セキュリティー管理プロバイダーを使用するには、以下の手順を行います。

前提条件

- Red Hat JBoss EAP がインストールされている。

手順

1. Red Hat JBoss EAP インスタンスの既存のユーザーまたはロールプロパティファイルを使用するには、以下の例で示すように、**EAP_HOME/standalone/configuration/application-users.properties** および **EAP_HOME/standalone/configuration/application-roles.properties** ファイルに以下のシステムプロパティを含めます。

```
<property name="org.uberfire.ext.security.management.wildfly.properties.realm"
value="ApplicationRealm"/>
<property name="org.uberfire.ext.security.management.wildfly.properties.users-file-path"
```

```
value="/standalone/configuration/application-users.properties"/>
<property name="org.uberfire.ext.security.management.wildfly.properties.groups-file-path"
value="/standalone/configuration/application-roles.properties"/>
```

以下の表は、これらのプロパティの説明とデフォルト値を示しています。

表20.1 プロパティファイルを基にする Red Hat JBoss EAP セキュリティー管理プロバイダー

プロパティ	説明	デフォルト値
org.uberfire.ext.security.management.wildfly.properties.realm	レルムの名前このプロパティは必須ではありません。	ApplicationRealm
org.uberfire.ext.security.management.wildfly.properties.users-file-path	ユーザープロパティファイルの絶対パス。このプロパティは必須です。	./standalone/configuration/application-users.properties
org.uberfire.ext.security.management.wildfly.properties.groups-file-path	グループプロパティファイルの絶対パス。このプロパティは必須です。	./standalone/configuration/application-roles.properties

- アプリケーションのルートディレクトリーに **security-management.properties** ファイルを作成します。たとえば、以下のファイルを作成します。

```
src/main/resources/security-management.properties
```

- security-management.properties** ファイルの値として、以下のシステムプロパティおよびセキュリティープロバイダー名を入力します。

```
<property name="org.uberfire.ext.security.management.api.userManagementServices"
value="WildflyUserManagementService"/>
```

20.1.2. プロパティファイルと CLI モードを基にした Red Hat JBoss EAP セキュリティー管理プロバイダーの設定

プロパティファイルと CLI モードを基に Red Hat JBoss EAP セキュリティー管理プロバイダーを使用するには、以下の手順を行います。

前提条件

- Red Hat JBoss EAP がインストールされている。

手順

- Red Hat JBoss EAP インスタンスの既存のユーザーまたはロールプロパティファイルを使用するには、以下の例で示すように、**EAP_HOME/standalone/configuration/application-users.properties** および **EAP_HOME/standalone/configuration/application-roles.properties** ファイルに以下のシステムプロパティを含めます。

```
<property name="org.uberfire.ext.security.management.wildfly.cli.host" value="localhost"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.port" value="9990"/>
```

```
<property name="org.uberfire.ext.security.management.wildfly.cli.user" value="
<USERNAME>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.password" value="
<USER_PWD>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.realm"
value="ApplicationRealm"/>
```

以下の表は、これらのプロパティーの説明とデフォルト値を示しています。

表20.2 プロパティーファイルと CLI モードを基にする Red Hat JBoss EAP セキュリティー管理プロバイダー

プロパティー	説明	デフォルト値
org.uberfire.ext.security.m anagement.wildfly.cli.host	ネイティブ管理インターフェ イスホスト。	localhost
org.uberfire.ext.security.m anagement.wildfly.cli.port	ネイティブ管理インターフェ イスポート。	9990
org.uberfire.ext.security.m anagement.wildfly.cli.user	ネイティブ管理インターフェ イスのユーザー名。	NA
org.uberfire.ext.security.m anagement.wildfly.cli.pass word	ネイティブ管理インターフェ イスのユーザーのパスワー ド。	NA
org.uberfire.ext.security.m anagement.wildfly.cli.real m	アプリケーションのセキュリ ティーコンテキストで使用さ れるレルム。	ApplicationRealm

2. アプリケーションのルートディレクトリーに **security-management.properties** ファイルを作成します。たとえば、以下のファイルを作成します。

```
src/main/resources/security-management.properties
```

3. **security-management.properties** ファイルの値として、以下のシステムプロパティーおよびセキュリティープロバイダー名を入力します。

```
<property name="org.uberfire.ext.security.management.api.userManagementServices"
value="WildflyCLIUserManagementService"/>
```

20.2. パーミッションおよび設定

パーミッションは、アプリケーション内の特定のリソースに関連するアクションを実行するためにユーザーに付与される権限です。たとえば、以下のパーミッションを指定できます。

- ページを表示する。
- プロジェクトを保存する。
- リポジトリーを削除する。

- ダッシュボードを削除する。

パーミッションは、付与と拒否ができ、グローバルに設定することも、リソースを指定して設定することもできます。パーミッションを使用すると、リソースへのアクセス時のセキュリティーが保護され、アプリケーション内の機能をカスタマイズできます。

20.2.1. Business Central でのグループおよびロールのパーミッションの変更

Business Central では、個人ユーザーに対するパーミッションは変更できません。ただし、グループおよびロールのパーミッションは変更できます。変更したパーミッションは、変更したロールが割り当てられているか、変更したグループに所属するユーザーに適用されます。



注記

ロールまたはグループへの変更は、そのロールまたはグループに関連のあるユーザーに加えられます。

前提条件

- Business Central に **admin** ユーザーロールでログインします。

手順

1. Business Central で **Security management** ページにアクセスするには、画面の右上隅にある **Admin** アイコンを選択します。
2. Business Central **Settings** ページで **Roles**、**Groups**、または **Users** をクリックします。クリックしたアイコンのタブに、**Security management** ページが開きます。
3. リストから編集するロールまたはグループをクリックします。全詳細が右側のペインに表示されます。
4. **Settings** セクションの **Home Page** または **Priority** を設定します。
5. **Permissions** セクションで、Business Central、ページ、エディター、スペース、プロジェクトのパーミッションを設定します。

図20.1パーミッションの設定

admin settings

Home Page ⓘ

Priority ⓘ

Permissions

> Workbench ⓘ

Pages ⓘ

[Add Exception](#)

Editors ⓘ

Spaces ⓘ

Projects ⓘ

6. 変更するパーミッションのリソースタイプの横にある矢印をクリックしてデプロイメントします。
7. 必要に応じて、リソースタイプに例外を追加するには、**Add Exception** をクリックしてから、必要なパーミッションを設定します。



注記

Business Central のリソースタイプには、例外を追加できません。

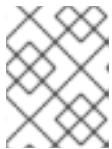
8. **Save** をクリックします。

20.2.2. Business Central ホームページの変更

ホームページは、Business Central にログインすると表示されるページです。デフォルトでは、ホームページは **Home** に設定されます。ロールとグループ別に異なるホームページを指定できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Roles** または **Groups** を選択します。
2. ロールまたはグループを選択します。
3. **Home Page** リストからページを選択します。
4. **Save** をクリックします。



注記

そのロールまたはグループには、ページをホームページにする前に、そのページへの読み取りアクセスが必要です。

20.2.3. 優先順位の設定

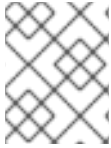
ユーザーは、複数のロールを持ち、複数のグループに所属します。優先順位の設定は、ロールまたはグループの優先順を決定します。

前提条件

- Business Central に **admin** ユーザーロールでログインします。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Roles** または **Groups** を選択します。
2. ロールまたはグループを選択します。
3. 優先順位メニューから優先順位を選択し、**Save** をクリックします。



注記

ユーザーに、設定が競合するロールが割り当てられているか、グループに所属している場合は、一番高い優先順位を持つロールまたはグループを設定します。

第21章 アーティファクトの管理

Business Central の **Artifacts** ページからアーティファクトを管理できます。アーティファクトリーポジトリーは、ローカルの Maven リポジトリーで、インストールごとに Maven リポジトリーは1つのみとなっています。Business Central は、**Sonatype Nexus™**、**Apache Archiva™**、**JFrog Artifactory™** などの Maven リポジトリーソリューションを使用することを推奨します。

Artifacts ページでは、Maven リポジトリーの全アーティファクトをすべて表示します。アーティファクトは、Maven リポジトリーにアップロードできます。



注記

Artifacts リポジトリーにアップロードできるのは、JAR ファイル、KJAR ファイル、および **pom.xml** ファイルのみです。

21.1. アーティファクトの表示

Artifacts ページからローカルの maven リポジトリーのコンテンツをすべて確認できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Artifacts** を選択します。
2. **Open** をクリックしてアーティファクトの詳細を表示します。
3. **Ok** をクリックして **Artifacts** ページに戻ります。

21.2. アーティファクトのダウンロード

Business Central のリポジトリーからプロジェクトのローカルストレージにアーティファクトをダウンロードして保存できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Artifacts** を選択します。
2. **Download** をクリックします。
3. アーティファクトを保存するディレクトリーを参照します。
4. **Save** をクリックします。

21.3. アーティファクトのアップロード

ローカルストレージから Business Central のプロジェクトにアーティファクトをアップロードできます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Artifacts** を選択します。
2. **Upload** をクリックします。

3. **Choose File** をクリックして、アーティファクトをアップロードするディレクトリーに移動して選択します。
4. **Upload** をクリックします。



注記

Maven 以外のアーティファクトを使用する場合は、先に **mvn deploy** コマンドを使用してアーティファクトを Maven リポジトリーにデプロイしてから Business Central のアーティファクトリストを更新します。

第22章 データソースおよびデータベースドライバーの管理

Business Central ではデータソースの管理機能があり、データソースを定義してデータベースにアクセスできます。これらのデータソースは、データセットなど、他の Business Central コンポーネントが使用します。また、データベースドライバーは、データソースと対象のデータベースの間の通信を有効にするために使用します。

Data Source Authoring ページから、Business Central にデータソースとデータベースのドライバーを追加できます。



注記

Business Central にはデフォルトのデータソースが含まれており、これは使用可能ですが、編集または削除することができません。

22.1. データソースの追加

Data Sources Authoring ページから Business Central に新しいデータソースを追加できます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで **Add DataSource** をクリックします。
New data source ウィンドウが開きます。
3. **New data source** ウィンドウで、データソースの **Name**、データベースの **Connection URL**、**User** と **Password**、ならびに **Driver** を入力します。
4. **Test Connection** をクリックして、データベースへの接続を確認します。
5. **Finish** をクリックします。

22.2. データソースの編集

Business Central でデータソースのプロパティを編集し、データベースへの接続をテストできます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで編集するデータソースを選択します。
3. **Data Source Definition** ペインで、**Name** フィールド、**Connection URL** フィールド、**User** フィールド、**Password** フィールド、および **Driver** フィールドに必要な変更を加えます。
4. **Test Connection** をクリックして、データベースへの接続を確認します。
5. **Update** をクリックします。
6. **Save** をクリックします。

22.3. データソースの削除

Business Central の **DataSource Explorer** ペインから既存のデータソースを削除できます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで削除するデータソースを選択します。
3. **Remove** をクリックします。
4. **Delete** をクリックして、データソースの削除を確定します。

22.4. データベースドライバーの追加

Business Central に新規データベースドライバーを追加できます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで **Add Driver** をクリックします。
New driver ウィンドウが開きます。
3. **New driver** ウィンドウで、データベースドライバーの **Name**、**Driver Class Name**、**Group Id**、**Artifact Id**、および **Version** フィールドを入力します。
4. **Finish** をクリックします。

22.5. データベースドライバーの編集

Driver Definition ペインからデータベースドライバーのプロパティを編集できます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで編集するドライバーを選択します。
3. **Driver Definition** ペインで、**Name** フィールド、**Driver Class Name** フィールド、**Group Id** フィールド、**Artifact Id** フィールド、および **Version** フィールドに必要な変更を加えます。
4. **Update** をクリックします。
5. **Yes** をクリックします。

22.6. データベースドライバーの削除

Business Central の **Data Source Definition** ペインからデータベースドライバーを削除できます。

手順

1. Business Central で、画面の上隅にある **Admin** アイコンを選択し、**Data Sources** を選択します。
2. **DataSource Explorer** ペインで削除するドライバーを選択します。
3. **Remove** をクリックします。
4. **Delete** をクリックします。

第23章 データセットのオーサリング

データセットは情報の関連セットの集まりで、データベース、Microsoft Excel、またはメモリーに保存できます。データセット定義は、Business Central メソッドにデータセットへのアクセス、読み取り、および解析を指示します。Business Central はデータを保存しません。データが保存される場所にかかわらず、データセットへのアクセスを定義できます。

たとえば、データベースにデータが保存されると、有効なデータセットには、SQL クエリーの結果として、データベース全体またはデータベースのサブセットなどが含まれます。いずれの場合も、データは、情報を表示する Business Central のレポーティングコンポーネントの入力情報として使用されます。

データセットにアクセスするには、データセット定義を作成および登録する必要があります。このデータセットの定義では、データセットの場所と、その場所へのアクセス、読み取り、および解析のオプション、ならびにデータセットが含まれるコラムを指定します。



注記

Data Sets ページは、admin ロールを持つユーザーにのみ表示されます。

23.1. データセットの追加

外部データソースからデータを取得して、レポーティングコンポーネントでデータを使用するデータセットを作成できます。

手順

1. Business Central で、**Admin** → **Data Sets** に移動します。
Data Sets ページが開きます。
2. **New Data Set** をクリックして、以下のプロバイダタイプから1つ選択します。
 - **Bean**: Java クラスからデータセットを生成します。
 - **CSV**: リモートまたはローカルの CSV ファイルからデータセットを生成します。
 - **SQL**: ANSI-SQL 準拠データベースからデータセットを生成します。
 - **Elastic Search**: Elastic Search ノードからデータセットを生成します。
 - **Prometheus**: Prometheus クエリーを使用してデータセットを生成します。
 - **Kafka**: Kafka ブローカー、コンシューマー、またはプロデューサーからメトリックを使用してデータセットを生成します。



注記

Prometheus、Kafka、および Execution Server オプションに KIE Server を設定する必要があります。

3. **Data Set Creation Wizard** を完了し、**Test** をクリックします。



注記

設定手順は、選択するプロバイダーにより異なります。

4. **Save** をクリックします。

23.2. データセットの編集

既存のデータセットを編集し、レポートコンポーネントに取得したデータが最新になっていることを確認します。

手順

1. Business Central で、**Admin** → **Data Sets** に移動します。
Data Set Explorer ページが開きます。
2. **Data Set Explorer** ペインで、編集するデータセットを検索し、データセットを選択して **Edit** をクリックします。
3. **Data Set Editor** ペインで、適切なタブを使用して必要に応じてデータを編集します。タブは、選択するデータセットプロバイダーの種類によって異なります。
たとえば、**CSV** データプロバイダーの編集には、以下の変更が適用できます。
 - **CSV Configuration**: データセット定義の名前、ソースファイル、区切り記号などのプロパティを変更できます。
 - **Preview**: データのプレビューを使用できます。**CSV Configuration** タブで **Test** をクリックすると、システムはデータセットのルックアップコールを実行し、データが利用可能な場合はプレビューが表示されます。**Preview** タブには2つのサブタブがあります。
 - **Data columns**: どの列をデータセット定義に追加するかを指定できます。
 - **Filter**: 新しいフィルターを追加できます。
 - **Advanced**: 以下の設定を管理できます。
 - **Caching**: 詳細は [キャッシュデータ](#) を参照してください。
 - **Cache life-cycle**: データセット (またはデータ) を再読み込みされるまでの間隔を指定できます。バックエンドデータに変更が加えられると、**Refresh on stale data**機能は、キャッシュしたデータを再読み込みします。
4. 必要な変更を行ったら、**Validate** をクリックします。
5. **Save** をクリックします。

23.3. データの再読み込み

データの再読み込み機能を使用すると、データセット (またはデータ) を再読み込みされるまでの間隔を指定できます。データセットの **Advanced** タブにある **データ更新間隔** 機能にアクセスできます。バックエンドデータに変更が加えられると、**Refresh on stale data**機能は、キャッシュしたデータを再読み込みします。

23.4. データのキャッシュ

Business Central は、インメモリーデータを使用してデータセットを保存し、データ操作を実行するキャッシュメカニズムを提供します。データのキャッシュにより、ネットワークトラフィック、リモートシステムのペイロード、処理時間が減ります。パフォーマンスの問題を回避するには、Decision Central にキャッシュを設定します。

データセットを生成するデータルックアップ呼び出しの場合、キャッシュメソッドは、データルックアップ呼び出しが実行される場所と結果のデータセットが格納される場所を決定します。データのルックアップコールの例としては、ロケールパラメーターを Urban として設定するすべての住宅ローンアプリケーションが挙げられます。

Business Central データセット機能には、キャッシュレベルが2つあります。

- クライアントレベル
- バックエンドレベル

データセットの **Advanced** タブで、**クライアントキャッシュ** および **バックエンドキャッシュ** 設定を指定できます。

クライアントキャッシュ

キャッシュを有効にすると、データセットはルックアップ操作時に Web ブラウザーにキャッシュされ、その後のルックアップ操作ではバックエンドへの要求が実行されません。グループ化、集計、フィルタリング、並べ替えなどのデータセット操作は Web ブラウザーで処理されます。クライアントのキャッシュは、データセットのサイズが小さい場合 (例: データが 10 MB より少ない) にのみ有効になります。データセットが大きい場合は、パフォーマンスの低下や断続的なフリーズなどのブラウザー問題が発生する場合があります。クライアントのキャッシュは、ストレージシステムへの要求などの、バックエンド要求の数を減らします。

バックエンドキャッシュ

キャッシュが有効な場合に、デシジョンエンジンはデータセットをキャッシュします。これにより、リモートのストレージシステムへのバックエンドの要求数が減ります。データセットの全操作は、インメモリーデータを使用してデシジョンエンジンで実行されます。バックエンドキャッシュは、データセットのサイズが頻繁に更新され、インメモリーに保存されて処理される場合に限り有効です。バックエンドキャッシングは、リモートストレージを使用した低レイテンシー接続問題が発生している状況でも有効です。



注記

バックエンドキャッシュの設定は、**Data Set Editor** の **Advanced** タブに常に表示されるわけではありません。これは、インメモリーデシジョンエンジンでデータのルックアップ操作を解決するのに、Java および CSV のデータプロバイダーはバックエンドキャッシュに依存するためです (データセットはメモリー内に存在する必要があります)。

第24章 アーキタイプの管理

Business Central には、アーキタイプの表示、追加、検証、デフォルト設定、削除が可能なアーキタイプの管理機能があります。Business Central の **Archetypes** ページから、アーキタイプを管理できます。アーキタイプは、Apache Maven リポジトリにインストールされるプロジェクトであり、必要に応じてテンプレート構造を設定または作成できます。

アーキタイプに関する最新の詳細情報は、[Introduction to Archetypes page](#) を参照してください。

24.1. アーキタイプの表示

Archetypes ページには、Business Central で追加したアーキタイプがすべて表示されます。このリストには、アーキタイプの **グループ ID**、**アーティファクト ID**、**バージョン**、**作成日**、**ステータス**、および **アクション** の詳細情報が含まれます。

前提条件

- Maven リポジトリからアーキタイプを作成し、Business Central の **Settings** で表示されている。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択して、**Archetypes** を選択します。
Status コラムの緑のアイコンは有効なアーキタイプを、赤のアイコンは無効なアーキタイプを、青のアイコンは対応するアーキタイプが新しいスペースのデフォルトであることを示します。

24.2. アーキタイプの追加

Business Central に新規アーキタイプを追加できます。

前提条件

- Maven リポジトリにアーキタイプをインストールしている。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択して、**Archetypes** を選択します。
2. **Add Archetype** をクリックします。
3. **Add Archetype** パネルで、**Group ID** フィールド、**Artifact ID** フィールド、および **Version** フィールドにそれぞれ GAV 属性を入力します。
4. **追加** をクリックします。

Business Central は、新しく追加したアーキタイプを検証して、全スペースのテンプレートで使用できるようにします。

24.3. アーキタイプの追加機能の管理


Business Central の **Archetypes** ページからアーキタイプの削除、デフォルト設定、および検証を行います。

前提条件

- Maven リポジトリからアーキタイプを作成し、Business Central の **Settings** で表示されています。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択して、**Archetypes** を選択します。

2. **Actions** コラムから、アーキタイプの右側にある  アイコンをクリックします。
 - ドロップダウンメニューから **Delete** を選択して、リストからアーキタイプを削除します。
 - ドロップダウンメニューから **Validate** を選択して、アーキタイプが有効かどうかを検証します。



注記

Business Central の起動時に、登録したアーキタイプがすべて自動的に検証されます。

- ドロップダウンメニューから **Set as default** を選択して、新規スペースのデフォルトとしてアーキタイプを設定します。

24.4. アーキタイプを使用したプロジェクトの作成

アーキタイプを使用して Business Central でプロジェクトを作成します。Business Central でプロジェクトを作成する時に、Red Hat Decision Manager インストールに接続されている Git リポジトリに追加されます。

前提条件

- Maven リポジトリからアーキタイプを作成し、Business Central の **Settings** で表示されています。
- Business Central のスペースでアーキタイプをデフォルトとして設定しました。

手順

1. Business Central で、**Menu** → **Design** → **Projects** に移動します。
2. アーキタイプテンプレートから新しいプロジェクトを追加するスペースを選択または作成します。
3. **Add Project** をクリックします。
4. **Name** フィールドおよび **Description** フィールドにそれぞれプロジェクト名と説明を入力します。

5. **Configure Advanced Options** をクリックします。
6. **Based on template** チェックボックスを選択します。
7. 必要に応じてドロップダウンオプションからアーキタイプを選択します。
デフォルトのアーキタイプはすでにスペースに設定されています。
8. **Add** をクリックします。

選択したアーキタイプのテンプレートをもとに、プロジェクトの **Assets** ビューが開きます。


24.5. BUSINESS CENTRAL のスペース設定を使用したアーキタイプの管理

アーキタイプを Business Central に追加すると、すべてのスペースでそれらをテンプレートとして使用できます。スペースで利用可能な **Settings** タブから全アーキタイプを管理できます。このタブは **admin** ロールが割り当てられたユーザーにしか表示されません。

前提条件

- Maven リポジトリにアーキタイプをインストールしている。
- Maven リポジトリからアーキタイプを作成し、Business Central の **Settings** で表示されている。

手順

1. Business Central で、**Menu** → **Design** → **Projects** に移動します。
2. アーキタイプを管理するスペースを選択または作成します。デフォルトのスペースは **MySpace** です。
3. **Settings** をクリックします。
4. スペースでアーキタイプを追加または除外するには、**Include** チェックボックスを選択します。
5. **Actions** コラムからアーキタイプの右側にある  アイコンをクリックし、ドロップダウンメニューから **Set as default** を選択してスペースのデフォルトとしてアーキタイプを設定します。
6. **Save** をクリックします。

第25章 プロジェクト設定のカスタマイズ

Business Central では、プロジェクトはスペースに含まれており、プロジェクトで関連アセットを保存します。スペースに複数のプロジェクトを追加できます。

たとえば、組織には人事、給与、エンジニアリング、R&D など、多くの部署が含まれます。各部署は、それぞれのプロジェクトを追加して、そのプロジェクトと共に Business Central のスペースにマップできます。

Business Central ではプロジェクト設定をカスタマイズできます。新規プロジェクトを作成するか、既存の Git リポジトリからプロジェクトをクローンできます。

手順

1. Business Central で、上隅にある **Admin** アイコンを選択し、**Projects** を選択します。
2. **Project Preferences** ペインで、変更する設定を選択します。プロジェクト設定には以下が含まれます。
 - **Project Importing:** この設定は以下のプロパティーで設定されます。
 - **Allow multiple projects to be imported on cluster**を選択し、クラスターに複数のプロジェクトをインポートします。
 - **File exporting:** この設定には以下のプロパティーが含まれます。

表25.1 ファイルエクスポートのプロパティー

フィールド	説明
PDF の向き	PDF の向きが横向きか、縦向きかを指定します。
PDF の単位	PDF の単位に PT、MM、CN、または IN のいずれかを指定します。
PDF ページ形式	PDF ページ形式が A[0-10]、B[0-10]、または C[0-10] のいずれかに指定します。

- **Spaces:** この設定には以下のプロパティーが含まれます。

表25.2 スペースのプロパティー

フィールド	説明
名前	スペースのデフォルト名。存在しない場合は自動的に作成されます。
所有者	スペースのデフォルトの所有者。存在しない場合は自動的に作成されます。

フィールド	説明
グループ ID	スペースのデフォルトグループ ID。存在しない場合は自動的に作成されます。
エイリアス (単数)	スペースのカスタマイズエイリアス (単数) を指定します。
エイリアス (複数)	スペースのカスタマイズエイリアス (複数) を指定します。

- **Default values:** この設定には以下のプロパティが含まれます。

表25.3 デフォルト値のプロパティ

フィールド	説明
バージョン	プロジェクトを作成する場合のプロジェクトのデフォルトバージョン番号
説明	プロジェクトを作成する場合のプロジェクトのデフォルトの説明
ブランチ	Git リポジトリを使用時に使用するデフォルトのブランチ
ページあたりのアセット	プロジェクトのページごとにアセット数をカスタマイズするのに使用します。デフォルト値は 15 です。

- **Advanced GAV preferences:** この設定には以下のプロパティが含まれます。

表25.4 GAV 詳細設定のプロパティ

フィールド	説明
GAV の競合チェックを無効にしますか？	GAV の競合チェックを有効化するか、無効化するか指定します。このチェックボックスを無効にすると、プロジェクトに同じ GAV (グループ ID、アーティファクト、バージョン) を含めることができます。
GAV の子エディションを許可しますか？	子/サブプロジェクトに GAV エディションを含めることができるかどうかを指定します。



注記

重複する GAV の検出は、開発モードのプロジェクトでは無効になっています。Business Central のプロジェクトで重複する GAV 検出を有効にするには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを **OFF** (該当する場合) に切り替えます。

3. **Save** をクリックします。

第26章 アーティファクトリポジトリのカスタマイズのプロパティ

場合によっては、ドメインモデルの JAR ファイルをビルドするのに、プロジェクトで外部の依存関係を解決する必要があります。リポジトリには、以下の機能を持つアーティファクトが必要です。

- リポジトリは、Maven リポジトリである
- 全スナップショットにはタイムスタンプが含まれる
- アセットはほぼ、ローカルのハードドライブに保存されている

デフォルトでは、アーティファクトのリポジトリは `$WORKING_DIRECTORY/repositories/kie` に含まれます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Artifact Repository** を選択します。**Artifact Repository** ページが開きます。
2. いずれかを選択して、**Properties** セクションに情報を入力します。
3. **Save** をクリックします。

第27章 言語のカスタマイズ設定

Business Central の **Settings** ページで、言語を変更できます。Business Central は以下の言語をサポートします。

- 英語
- スペイン語
- フランス語
- 日本語

デフォルトの言語は英語です。

手順

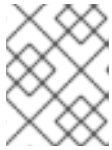
1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Laugages** を選択します。**Language Selector** が開きます。
2. **Language** リストから希望の言語を選択します。
3. **OK** をクリックします。

第28章 プロセス管理のカスタマイズ

Business Central では、**Process Administration** ページの **Default items per page** プロパティを編集して、デフォルトのページオプションをカスタマイズできます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Process Administration** を選択します。
2. **Properties** セクションから **Default items per page** プロパティを更新して、**Save** をクリックします。



注記

ページごとに表示するアイテムは 10、20、50、または 100 のいずれかを指定できます。

第29章 PROCESS DESIGNER のカスタマイズ

Business Central では、**Settings** ページからダイアグラムエディターのプロパティを編集して、Process Designer をカスタマイズできます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Process Designer** を選択します。
2. **Properties** セクションで、以下のプロパティのいずれかを更新します。
 - **Auto hide category panel** チェックボックスを選択して、自動的にカテゴリーツールバーパネルが非表示になるようにします。
 - 描画エリアの幅を設定するには、**Drawing area width** フィールドで、2800 から 5600 までの整数値を入力します。
 - 描画エリアの高さを設定するには、**Drawing area height** フィールドで、1400 から 2800 までの整数値を入力します。
 - 高解像度のディスプレイを使用しており、文字やオブジェクトがぼやけて表示される場合には、**Enable HiDPI** チェックボックスを選択します。このオプションはデフォルトで無効になっています。
3. **Save** をクリックします。

第30章 SSH キー

Business Central には SSH キーストアサービスがあり、ユーザーの SSH 認証を有効にします。また、設定可能なデフォルトの SSH キーストアと拡張可能な API (カスタムの実装) が含まれており、複数の SSH の公開鍵形式をサポートします。

SSH 公開鍵を登録するには、Business Central の **Settings** ページから **SSH Keys** オプションにアクセスできます。

30.1. デフォルトの SSH キーストア

Business Central に含まれるデフォルトの SSH キーストアでは、ユーザーの公開鍵を保存するのにファイルベースのストレージメカニズムが採用されています。デフォルトでは、Business Central は ***.security** ディレクトリーを root ディレクトリーとして使用します。ただし、別のディレクトリーを参照するように、**appformer.ssh.keys.storage.folder** システムプロパティーの値を設定することで、カスタムのストレージパスを使用できます。

SSH 公開鍵は **{securityFolderPath}/pkeys/{userName}/** ディレクトリー構造に保存されます。

各 SSH 公開鍵は以下のファイルで設定されており、これらのファイルはストレージフォルダーに配置されています。

- **{keyId}.pub**: このファイルには SSH 公開鍵のコンテンツが含まれます。ファイル名で、システムのロジックキー ID がわかるので、ファイル名がランタイム時に変更されないようにしてください。以下に例を示します。

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDmak4Wu23RZ6XmN94bOsqecZxuTa4RRhhQm
HmTZjMB7HM57/90u/B/gB/GhsPEu1nAXL0npY56tT/MPQ8vRm2C2W9A7CzN5+z5yyL3W01Y
Zy3kzslk77CjULjfhrcfQSL3b2sPG5jv5E5/nyC/swSytucwT/PE7aXTS9H6cHIKUdYPzlt94SHoBx
WRIK7PJi9d+eLB+hmDzvbVa1ezu5a8yu2kcHi6Nxxf15iRj2rsceDTp0imC1jMoC6ZDfBvZSxL9F\
TMwFdNmTIJveBtv9nAbnAvIWiiS0VOKdj1s3GxBxeZYAcKbcsK9sJzusptk5dxGsG2Z8vInaglN
6OaOQ7b7tcomzCYYwviGQ9gRX8sGsVrw39gsDIGYP2tA4bRr7ecHnlNg1b0HCchA5+QCDk
4Hbz1UrnHmPA2Lg9c3WGm2qedvQdVJXuS3mlwYOqL40aXPs6890PvFJUlpivsZnF50djPnws
MxJZEf1HdTXgZD1Bh54ogZf7czyUNfkNkE69yJDbTHjpQd0cKUQnu9tVxqmBzhX31yF4VcsMe
ADcf2Z8wIA3n4LZnC/GwonYlq5+G93zJpFOkPhme8c2XuPuCXF795lsxyJ8SB/AlwPJAhEtm0y
0s0l1I4eWqxsDxkBOgN+ivU0cZrVMssHJEJb4o0FLf7iHhOW56/iMdD9w== userName
```

- **.{keyId}.pub.meta**: このファイルには、JSON 形式のキーメタデータが含まれます。キーにメタデータが含まれない場合は、新規のメタデータファイルが動的に生成されます。以下に例を示します。

```
{
  "name": "Key",
  "creationDate": "Oct 10, 2018 10:10:50 PM",
  "lastTimeUsed": "Oct 11, 2018 12:11:23 PM"
}
```

30.2. カスタムの SSH キーストア

要件に合わせてデフォルトの SSH キーストアを拡張し、カスタマイズできます。**appformer.ssh.keystore** システムプロパティーで、使用する SSH サービスの Java クラス名を指定してください。このプロパティーが定義されていない場合や、不正な値が含まれる場合には、デフォ

ルトの SSH キーストアが読み込まれます。



注記

SSH キーストアのカスタム実装を作成するには、Java クラスが **uberfire-ssh-api** モジュールで定義した **org.uberfire.ssh.service.backend.keystore.SSHKeyStore** クラスを実装する必要があります。

30.3. SSH キーの作成

SSH キーを Business Central に追加または登録する前に、お使いのシステムで SSH キーを生成する必要があります。

手順

1. システムでコマンド端末を開きます。
2. 以下の例のように、**ssh-keygen** コマンドを実行して、SSH キーを作成します。 **<user_login>** はユーザー名に置き換えてください。

```
ssh-keygen -t rsa -b 4096 -C "<user_login>"
```



注記

Business Central キーストアでサポートされる SSH キーは **ssh-rsa**、**ssh-dss**、**ecdsa-sha2-nistp256**、**ecdsa-sha2-nistp384**、および **ecdsa-sha2-nistp521** です。

3. 以下の例のように、プロンプトが表示されたら、Enter キーを押して、デフォルトのキーファイルの場所を確定します。 **<user_login>** はユーザー名に置き換えてください。

```
Enter a file in which to save the key (/home/<user_login>/.ssh/id_rsa): [Press enter]
```

4. コマンドプロンプトで、パスフレーズを入力して確定します。

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

5. **ssh-agent** を起動します。

```
eval "$(ssh-agent -s)"
Agent pid <any-number-here>
```

6. 新しい SSH 秘密鍵を **ssh-agent** に追加します。違う名前のキーを使用する場合は、コード内の **id_rsa** を置き換えます。

```
ssh-add ~/.ssh/id_rsa
```

30.4. SSH キーストアを使用した SSH 公開鍵の登録

新規作成した SSH 公開鍵は、Business Central キーストアに登録する必要があります。

手順

1. システムでコマンド端末を開きます。
2. 以下の例のように、**cat** コマンドを実行します。**id_rsa** はキーの名前に置き換えます。

```
cat ~/.ssh/id_rsa.pub
```

3. SSH 公開鍵のコンテンツをコピーします。
4. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**SSH Keys** を選択します。
5. **SSH Keys** ページで **Add SSH Key** をクリックします。
6. **Add SSH Key** ウィンドウで **Name** フィールドに名前を入力し、SSH 公開鍵のコンテンツを **Key** フィールドにコピーします。



注記

Name および Key は必須のフィールドです。

7. **Add SSH Key** をクリックしてキーを登録します。

30.5. SSH キーの削除

Business Central の **SSH Keys** ページから SSH キーを削除できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**SSH Keys** を選択します。
2. **SSH Keys** ページで削除する SSH キーの削除アイコンをクリックします。
3. **Delete SSH Key** をクリックして、削除を確定します。

第31章 BUSINESS CENTRAL でのカスタムタスクの管理

カスタムタスク (作業アイテム) は、カスタムロジックを実行できるタスクです。複数のビジネスプロセスまたは Business Central の全プロジェクトでタスクをカスタマイズして再利用できます。また、名前、アイコン、サブカテゴリー、入出力パラメーター、ドキュメントなど、デザイナーパレットにカスタム要素を追加することもできます。Red Hat Decision Manager は、Business Central のカスタムタスクリポジトリ内でカスタムタスクセットを提供します。デフォルトのカスタムタスクを有効化または無効化して、カスタムのタスクを Business Central にアップロードし、適切なプロセスにこのタスクを実装できます。



注記

Red Hat Decision Manager には、サポートされるカスタムタスクの限定セットが含まれています。Red Hat Decision Manager に含まれていないカスタムタスクはサポートされません。

手順

- Business Central で右上隅の  をクリックし、**Custom Task Administration** を選択します。

このページは、カスタムタスクのインストール設定や、Business Central 全体にあるプロジェクトのプロセスで利用可能なカスタムタスクを表示します。このページで有効にしたカスタムタスクは、プロジェクトレベルの設定で利用できます。プロジェクトレベルの設定で、プロセスで使用する各カスタムタスクをインストールできます。カスタムタスクをプロジェクトにインストールする方法は、**Custom Tasks Administration** ページの **Settings** で有効または無効にしたグローバル設定により決まります。
- Settings** で、各設定を有効または無効にして、ユーザーがプロジェクトレベルでインストールするときに、利用可能なカスタムタスクを実装する方法を決定します。

以下のカスタムタスクの設定が利用できます。

 - Install as Maven artifact** ファイルがない場合は、カスタムタスクの JAR ファイルを Maven リポジトリにアップロードし、Business Central で設定します。
 - Install custom task dependencies into project** カスタムタスクの依存関係をプロジェクトの **pom.xml** ファイルに追加します。このファイルでタスクがインストールされます。
 - Use version range when installing custom task into project** プロジェクトの依存関係として追加されるカスタムタスクの固定バージョンではなく、バージョン範囲を使用します。たとえば、**7.16.0.Final** ではなく **[7.16,)** です。
- 必要に応じて利用可能なカスタムタスクを有効または無効にします (**ON** または **OFF** に設定)。有効化したカスタムタスクは、Business Central の全プロジェクトのプロジェクトレベル設定に表示されます。

図31.1 カスタムタスクとカスタムタスク設定の有効化

Custom Tasks Administration

Settings

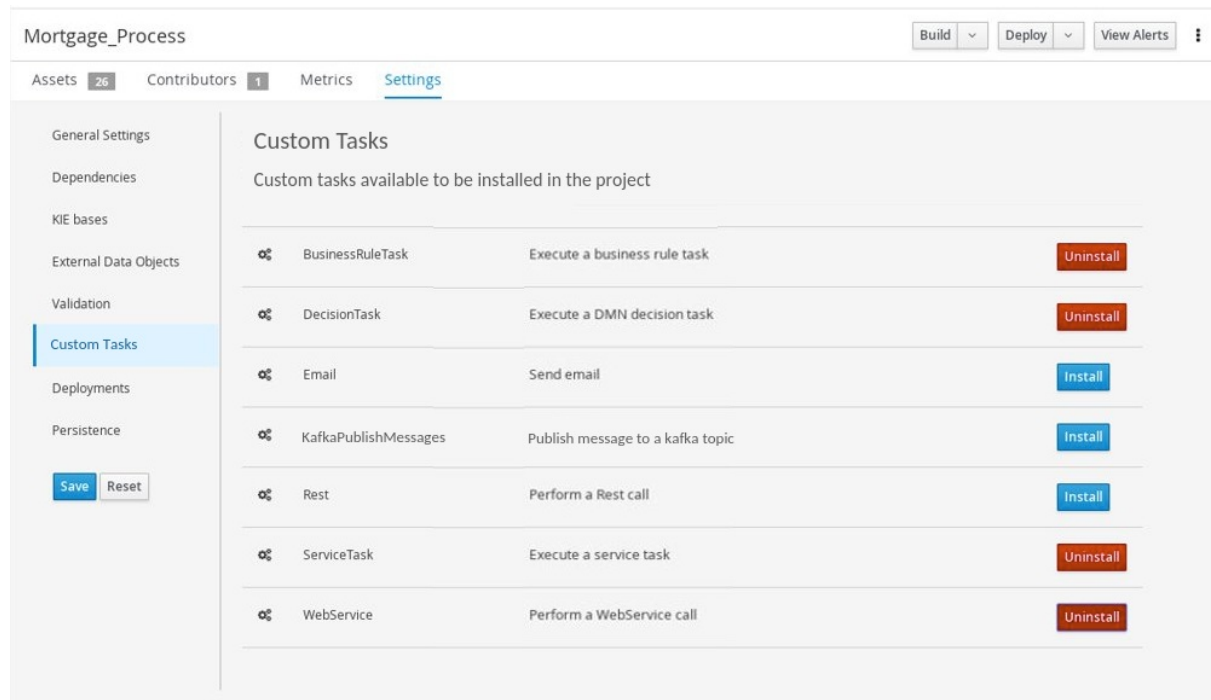
- Install as Maven artifact** ON
Instructs if enabled custom tasks should be installed into Maven repository
- Install custom task dependencies into project** ON
Instructs that custom task dependencies are added as project dependencies upon installation
- Use version range when installing custom task into a project** OFF
Instructs that a version range will be used when installing custom task in projects

[Add Custom Task](#)

	BusinessRuleTask	Execute business rule or service tasks Execute a business rule task	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	CamelXSLTConnector	Use Apache Camel connectors in your processes Process a message using an XSLT template	<input type="checkbox"/> OFF <input type="checkbox"/> ON
	DecisionTask	Execute business rule or service tasks Execute a DMN decision task	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	Email	Send an email Send email	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	KafkaPublishMessages	publish kafka messages from a process Publish message to a kafka topic	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	Rest	Perform REST calls Perform a Rest call	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	ServiceTask	Execute business rule or service tasks Execute a service task	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF
	WebService	Perform Webservice operations Perform a Webservice call	<input checked="" type="checkbox"/> ON <input type="checkbox"/> OFF

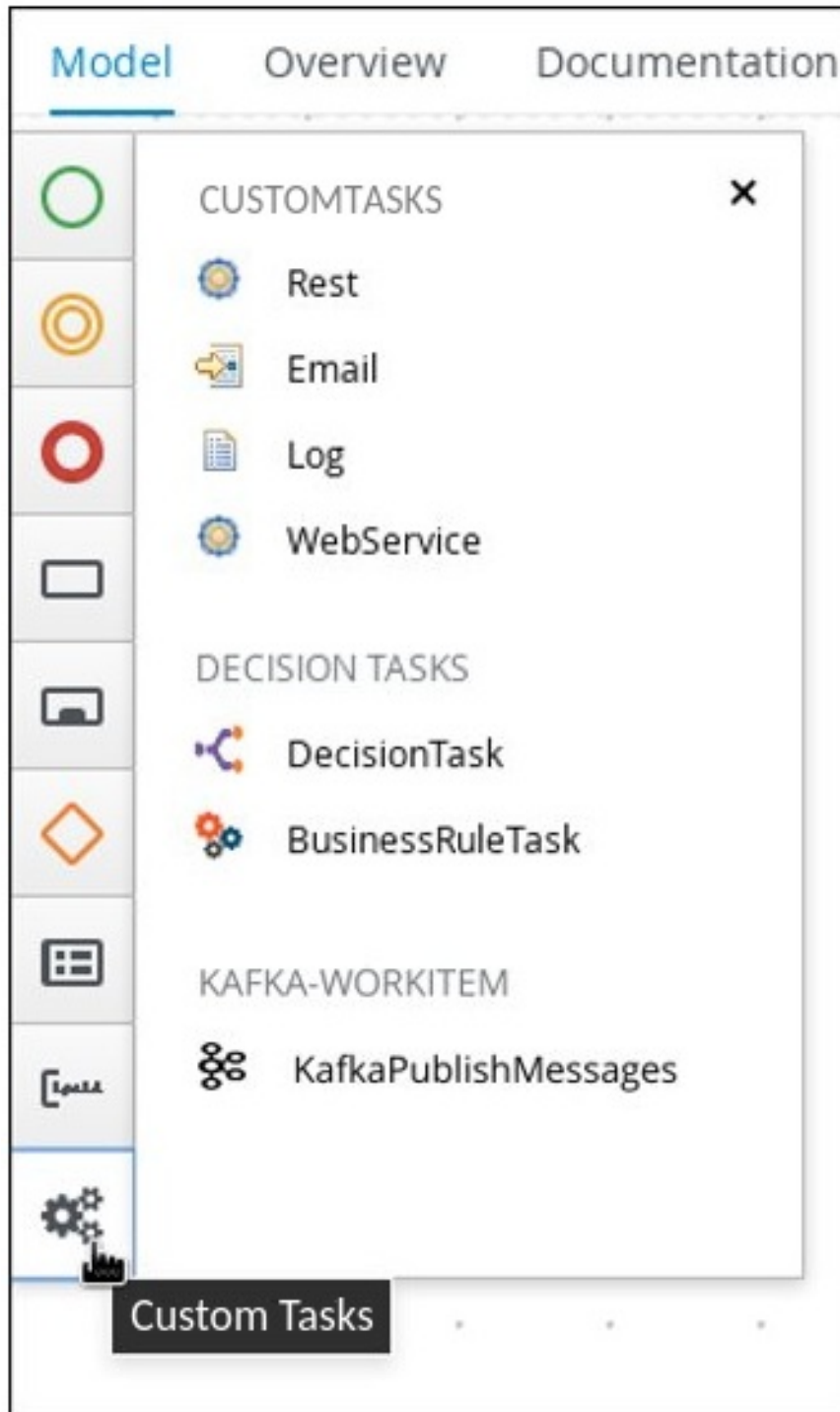
4. カスタムタスクを追加するには、**Add Custom Task**をクリックし、関連する JAR ファイルを参照し、**Upload** アイコンをクリックします。クラスが **WorkItemHandler** を実装する場合は、ファイルを Business Central に個別に追加して、アノテーションを **.wid** ファイルに置き換えることができます。
5. 必要に応じてカスタムタスクを削除するには、削除するカスタムタスクの行にある **remove** をクリックし、**OK** をクリックして削除を確定します。
6. Business Central ですべての必須カスタムタスクを設定した後に、プロジェクトの **Settings** → **Custom Tasks** ページに移動すると、有効化したカスタムタスクで利用可能なものが表示されます。
7. カスタムタスクごとに、**Install** をクリックして、対象のプロジェクトのプロセスでタスクを利用できるようにするか、**Uninstall** をクリックして、プロジェクトのプロセスからタスクを除外します。
8. カスタムタスクのインストール時に追加情報を求められた場合は、必要な情報を入力して、もう一度 **Install** をクリックします。
カスタムタスクの必須パラメーターは、タスクのタイプにより異なります。たとえば、ルールとデシジョンタスクにはアーティファクトの GAV 情報 (グループ ID、アーティファクト ID、およびバージョン) が、メールタスクにはホストとポートアクセスの情報が、REST タスクには API の認証情報が必要です。他のカスタムタスクでは、追加のパラメーターが必要でない場合もあります。

図31.2 プロセスで使用するカスタムタスクのインストール



9. **Save** をクリックします。
10. プロジェクトページに戻り、プロジェクトのビジネスプロセスを選択または追加します。プロセスデザイナーパレットで **Custom Tasks** オプションを選択すると、有効にしてインストールした、利用可能なカスタムタスクが表示されます。

図31.3 プロセスデザイナーでのインストール済みカスタムタスクへのアクセス



第32章 LDAP 接続

Business Central は、ユーザータスクサービスがユーザー、グループ、およびロールの情報を LDAP サービスから直接取得できるように、Red Hat Decision Manager で LDAP サーバー専用の **UserGroupCallback** 実装を提供します。

以下の LDAP **UserGroupCallback** 実装プロパティは、設定可能です。

表32.1 LDAP UserGroupCallback プロパティ

プロパティ	説明
ldap.bind.user	LDAP サーバーへの接続に使用するユーザー名 指定がない場合はこのプロパティは任意となり、LDAP サーバーは匿名アクセスを受け入れます。
ldap.bind.pwd	LDAP サーバーへの接続に使用するパスワード 指定がない場合はこのプロパティは任意となり、LDAP サーバーは匿名アクセスを受け入れます。
ldap.user.ctx	ユーザー情報を含む LDAP のコンテキスト
ldap.role.ctx	グループおよびロールの情報を含む LDAP のコンテキスト
ldap.user.roles.ctx	ユーザーグループおよびロールのメンバーシップ情報を含む LDAP のコンテキスト 指定がない場合はこのプロパティは任意となり、代わりに ldap.role.ctx プロパティを使用します。
ldap.user.filter	ユーザー情報検索用のフィルター このプロパティは通常、代入キー {0} が含まれており、パラメーターと置き換えられます。
ldap.role.filter	グループおよびロールの情報の検索フィルター このプロパティは通常、代入キー {0} が含まれており、パラメーターと置き換えられます。
ldap.user.roles.filter	ユーザーグループおよびロールのメンバーシップ情報を検索するためのフィルター このプロパティは通常、代入キー {0} が含まれており、パラメーターと置き換えられます。

プロパティ	説明
ldap.user.attr.id	LDAP に含まれるユーザー ID の属性名 指定がない場合はこのプロパティは任意となり、代わりに uid プロパティを使用します。
ldap.roles.attr.id	LDAP 内のグループおよびロール ID の属性名 指定がない場合はこのプロパティは任意となり、代わりに cn プロパティを使用します。
ldap.user.id.dn	DN のユーザー ID。ロールの検索前にユーザー DN をクエリーするようにコールバックを指示します。これは任意で、デフォルトは false です。
java.naming.factory.initial	初期コンテキストファクトリークラス名。デフォルトは com.sun.jndi.ldap.LdapCtxFactory
java.naming.security.authentication	認証タイプ。可能な値は、 none 、 simple 、および strong です。デフォルトは simple です。
java.naming.security.protocol	ssl など、使用するセキュリティプロトコル
java.naming.provider.url	LDAP url (デフォルトは ldap://localhost:389 。プロトコルが ssl に設定されている場合は ldap://localhost:636)

32.1. LDAP USERGROUPCALLBACK 実装

以下のいずれかの方法で、該当の LDAP プロパティを設定して、LDAP **UserGroupCallback** 実装を使用できます。

- プログラム: 該当の **LDAPUserGroupCallbackImpl** プロパティでプロパティのオブジェクトをビルドし、同じプロパティオブジェクトをパラメーターとして使用し、**LDAPUserGroupCallbackImpl** を作成します。
以下に例を示します。

```
import org.kie.api.PropertiesConfiguration;
import org.kie.api.task.UserGroupCallback;
...
Properties properties = new Properties();
properties.setProperty(LDAPUserGroupCallbackImpl.USER_CTX, "ou=People,dc=my-domain,dc=com");
properties.setProperty(LDAPUserGroupCallbackImpl.ROLE_CTX, "ou=Roles,dc=my-domain,dc=com");
properties.setProperty(LDAPUserGroupCallbackImpl.USER_ROLES_CTX, "ou=Roles,dc=my-domain,dc=com");
properties.setProperty(LDAPUserGroupCallbackImpl.USER_FILTER, "(uid={0})");
properties.setProperty(LDAPUserGroupCallbackImpl.ROLE_FILTER, "(cn={0})");
```

```
properties.setProperty(LDAPUserGroupCallbackImpl.USER_ROLES_FILTER, "(member={0})");
```

```
UserGroupCallback ldapUserGroupCallback = new
LDAPUserGroupCallbackImpl(properties);
```

```
UserGroupCallbackManager.getInstance().setCallback(ldapUserGroupCallback);
```

- 宣言設定: アプリケーションのルートに **jbpm.usergroup.callback.properties** ファイルを作成するか、システムプロパティとしてファイルの場所を指定します。以下に例を示します。

-Djbpm.usergroup.callback.properties=FILE_LOCATION_ON_CLASSPATH

ユーザータスクサーバーの起動時に LDAP コールバックを登録するようにしてください。

以下に例を示します。

```
#ldap.bind.user=
#ldap.bind.pwd=
ldap.user.ctx=ou\=People,dc\=my-domain,dc\=com
ldap.role.ctx=ou\=Roles,dc\=my-domain,dc\=com
ldap.user.roles.ctx=ou\=Roles,dc\=my-domain,dc\=com
ldap.user.filter=(uid\={0})
ldap.role.filter=(cn\={0})
ldap.user.roles.filter=(member\={0})
#ldap.user.attr.id=
#ldap.roles.attr.id=
```

関連情報

- [ロールおよびユーザー](#)
- [Red Hat Single Sign-On サーバー管理ガイド](#)
- [LDAP ログインドメインの定義](#)
- [Ldap ログインモジュール](#)
- [LdapExtended ログインモジュール](#)
- [AdvancedLDAP ログインモジュール](#)
- [AdvancedAdLdap ログインモジュール](#)
- [LDAP 接続オプション](#)
- [LdapUsers ログインモジュール](#)

第33章 データベース接続

Business Central は、Red Hat Decision Manager でデータベースサーバー用の **UserGroupCallback** 実装を提供し、ユーザータスクサービスを有効にします。ユーザータスクサービスを使用して、ユーザーやグループ (ロール) の情報を直接データベースから取得できるようにします。

以下のデータベースの **UserGroupCallback** 実装プロパティを設定することができます。

表33.1 データベースの UserGroupCallback プロパティ

プロパティ	説明
db.ds.jndi.name	接続に使用するデータソースの JNDI 名
db.user.query	ユーザーの存在を確認する
db.user.roles.query	特定のユーザーのグループを収集する
db.roles.query	グループの存在を確認する

33.1. データベースの USERGROUPCALLBACK 実装

データベースの **UserGroupCallback** 実装では、必須のデータベースを作成する必要があります。以下のいずれかの方法で、該当のデータベースプロパティを設定し、この実装を使用できます。

- プログラム: 該当の **DBUserGroupCallbackImpl** プロパティでプロパティのオブジェクトをビルドし、プロパティオブジェクトで、パラメーターとして **DBUserGroupCallbackImpl** を作成します。

以下に例を示します。

```
import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.DS_JNDI_NAME;
import static
org.jbpm.services.task.identity.DBUserGroupCallbackImpl.PRINCIPAL_QUERY;
import static org.jbpm.services.task.identity.DBUserGroupCallbackImpl.ROLES_QUERY;
import static
org.jbpm.services.task.identity.DBUserGroupCallbackImpl.USER_ROLES_QUERY;
...
props = new Properties();
props.setProperty(DS_JNDI_NAME, "jdbc/jbpm-ds");
props.setProperty(PRINCIPAL_QUERY, "select userId from Users where userId = ?");
props.setProperty(ROLES_QUERY, "select groupId from UserGroups where groupId = ?");
props.setProperty(USER_ROLES_QUERY, "select groupId from UserGroups where userId =
?");

callback = new DBUserGroupCallbackImpl(props);
```

- 宣言設定: アプリケーションのルートに **jbpm.usergroup.callback.properties** ファイルを作成するか、システムプロパティとしてファイルの場所を指定します。以下に例を示します。

-Djbpm.usergroup.callback.properties=FILE_LOCATION_ON_CLASSPATH

ユーザータスクサーバーの起動時にデータベースコールバックを登録するようにしてください。

以下に例を示します。

```
System.setProperty("jbpm.usergroup.callback.properties",
"/jbpm.usergroup.callback.db.properties");
callback = new DBUserGroupCallbackImpl(true);
...
db.ds.jndi.name = jdbc/jbpm-ds
db.user.query = select userId from Users where userId = ?
db.roles.query = select groupId from UserGroups where groupId = ?
db.user.roles.query = select groupId from UserGroups where userId = ?
```

関連情報

- [ロールおよびユーザー](#)

第34章 SETTINGS.XML ファイルを使用した MAVEN の設定

Java アプリケーション開発は、Apache Maven ビルド自動化ツールを使用して、ソフトウェアプロジェクトをビルドし、管理します。Maven は Project Object Model (POM) 設定の XML ファイルを使用して、プロジェクトプロパティとビルドプロセスの両方を定義します。

Maven はレポジトリを使用して Java ライブラリー、プラグイン、および他のビルドアーティファクトを格納します。レポジトリはローカルまたはリモートのいずれかになります。ローカルレポジトリは、ローカルマシンにキャッシュされたリモートレポジトリからアーティファクトをダウンロードしたものです。リモートレポジトリは、**http://** (HTTP サーバーにある場合) や **file://** (ファイルサーバーにある場合) などの一般的なプロトコルを使用してアクセスされる他のレポジトリです。デフォルトのレポジトリは、パブリックのリモート Maven 2 Central Repository となっています。Maven は、settings.xml ファイルを変更して設定できます。グローバルの Maven 設定は、**M2_HOME/conf/settings.xml** ファイルで、ユーザーレベルの設定は **USER_HOME/.m2/settings.xml** ファイルで実行可能です。

関連情報

- [Business Central および KIE Server への外部 Maven レポジトリの設定](#)
- [Red Hat Decision Manager プロジェクトの Maven でのパッケージ化およびデプロイ](#)
- [Red Hat Decision Manager の Maven 設定およびレポジトリ](#)
- [Maven を使ったシステム統合](#)
- [Welcome to Apache Maven](#)
- [Apache Maven Project - Introduction to Repositories](#)
- [Apache Maven Parent POMs Reference](#)

第35章 GAV チェック管理

Business Central では、プロジェクトは **グループ ID**、**アーティファクト ID**、および **バージョン (GAV)** の Maven 命名規則で識別されます。GAV の値は、プロジェクトとプロジェクトバージョンを区別し、特定のプロジェクトとの依存関係を識別します。

デフォルトでは、Business Central は GAV の重複を検出します。この機能は、**admin** ロールを持つユーザーにより無効にできます。

35.1. GAV チェックおよび子の GAV エディションの設定

以下の手順では、Business Central での GAV チェックの設定方法を説明します。

手順

1. Business Central で、**Menu** → **Design** → **Projects** に移動して、プロジェクト名をクリックします。
2. プロジェクトウィンドウで、**Settings** タブをクリックします。
3. **General Settings** タブで以下のタスクを実行します。
 - 他のプロジェクトで同じ GAV を使用できるようにするには、**Disable GAV conflict check** チェックボックスを選択します。
 - このプロジェクトに GAV エディションを指定できるようにするには、**Allow child GAV edition** チェックボックスを選択します。
4. **Save** をクリックします。



注記

Reset をクリックして、すべての変更を元に戻すことができます。

5. **Save** をクリックして、変更を確定します。



注記

重複する GAV の検出は、**Development Mode** のプロジェクトでは無効になっていません。Business Central で重複する GAV 検出を有効にするには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを **OFF** (該当する場合) に切り替えます。

35.2. 全プロジェクトの GAV チェックの設定

以下の手順では、Business Central の全プロジェクトに GAV チェックを設定する方法を説明します。また、システムの起動時に GAV チェックを無効にすることも可能です。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Projects** を選択します。**Projects** ウィンドウが開きます。
2. **Advanced GAV preferences** タブで以下のタスクのいずれかを実行します。

- 他のプロジェクトで同じ GAV を使用できるようにするには、**Disable GAV conflict check** チェックボックスを選択します。
- このプロジェクトに GAV エディションを指定できるようにするには、**Allow child GAV edition** チェックボックスを選択します。

3. **Save** をクリックします。



注記

Business Central の起動時に、**org.guvnor.project.gav.check.disabled** システムプロパティを **true** に設定して、重複した GAV の削除機能を無効にすることも可能です。

```
$ ~/EAP_HOME/bin/standalone.sh -c standalone-full.xml  
-Dorg.guvnor.project.gav.check.disabled=true
```

第36章 KIE SERVER および BUSINESS CENTRAL での環境モードの設定

KIE Server は、**production** (実稼働) モードと **development** (開発) モードでの実行が設定可能です。開発モードでは、柔軟な開発ポリシーが提供され、小規模な変更の場合はアクティブなプロセスインスタンスを維持しながら、既存のデプロイメントユニット (KIE コンテナ) を更新できます。また、大規模な変更の場合は、アクティブなプロセスインスタンスを更新する前に、デプロイメントユニットの状態をリセットすることも可能です。実稼働モードは、各デプロイメントで新規デプロイメントユニットが作成される実稼働環境に最適です。

開発環境では、Business Central で **Deploy** をクリックすると、(該当する場合に) 実行中のインスタンスを中止することなくビルドした KJAR ファイルを KIE Server にデプロイすることができます。または、**Redeploy** をクリックすると、ビルドされた KJAR ファイルをデプロイしてすべてのインスタンスを置き換えることができます。次回、ビルドされた KJAR ファイルをデプロイまたは再デプロイすると、以前のデプロイメントユニット (KIE コンテナ) が同じターゲット KIE Server で自動的に更新されます。

実稼働環境では、Business Central の **Redeploy** オプションが無効になり、**Deploy** をクリックして、ビルドした KJAR ファイルを KIE Server 上の新規デプロイメントユニット (KIE コンテナ) にデプロイすることのみが可能です。

手順

1. KIE Server の環境モードを設定するには、**org.kie.server.mode** システムプロパティを **org.kie.server.mode=development** または **org.kie.server.mode=production** に設定します。
2. Business Central のプロジェクトにデプロイメントの動作を設定するには、プロジェクトの **Settings** → **General Settings** → **Version** に移動して、**Development Mode** オプションを切り替えます。



注記

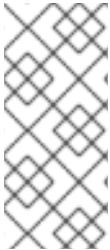
デフォルトでは、KIE Server および Business Central のすべての新規プロジェクトは開発モードになっています。

Development Mode をオンにしたプロジェクトをデプロイしたり、実稼働モードになっている KIE Server に手動で **SNAPSHOT** バージョンの接尾辞を追加したプロジェクトをデプロイしたりすることはできません。

第37章 GIT フックおよびリモート GIT リポジトリの統合

Git フックは、**git commit** や **git push** などの Git イベントの前後に実行するバッシュスクリプトです。Business Central では、Git フックを使用して、イベントが発生するたびに、リポジトリが指定のアクションをトリガーするように設定できます。Git フックの詳細は、[Customizing Git Hooks](#) を参照してください。

Business Central と、リモート Git リポジトリを統合するには、post-commit の Git フックを使用します。こうすることで、Business Central とリモートリポジトリの間のコンテンツの複製を自動化できます。たとえば、Business Central プロジェクトに加えた変更をリモートの Git リポジトリに複製する、リアルタイムのバックアップストラテジーを実装できます。



注記

Business Central は、post-commit の Git フックのみをサポートします。

post-commit の Git フックは、コミットするたびに同期操作として実行します。Business Central は、コミット後の Bash が完了するまで待機して、リポジトリでの他の書き込み操作が行われないようにします。

37.1. POST-COMMIT の GIT フックの作成

post-commit の Git フックスクリプトファイルを作成して、そのファイルに含まれるコードを実行するか、Java プログラムなどの別のファイルからコードを実行できます。

手順

1. **post-commit** Git フックファイルを作成します。

```
$ touch post-commit
```

2. **post-commit** ファイルのパーミッションを **755** に設定します。

```
$ chmod 755 post-commit
```

3. 以下のように、**#!/bin/bash** と必要なコードを **post-commit** ファイルに追加します。

- すべての変更をリモートリポジトリにプッシュするには、以下のコマンドを実行します。

```
#!/bin/bash
git push origin +master
```

- メッセージをログに記録するには、以下を実行します。

```
#!/bin/bash
echo 'Hello World'
```

- 別のファイルのコードを実行するには、以下を実行します。

```
#!/bin/bash
java -jar _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks/git-push.jar
```



注記

post-commit の Git フックを使用して Java コードを実行するには、以下の Java ライブラリーを使用する必要があります。

- [JGit](#): 内部の Business Central Git リポジトリと対話するのに使用します。
- [GitHub API for Java](#): GitHub との通信に使用します。

post-commit の Git フックと Java コードの例に関する情報は、[Business Central post-commit Git Hooks Integration](#) を参照してください。

37.2. リモート GIT リポジトリのインポート

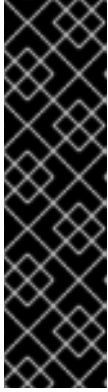
リモートの Git リポジトリを Business Central にインポートし、post-commit の Git フックを設定して、そのリモートリポジトリに変更を自動的にプッシュできます。

前提条件

- Red Hat JBoss EAP 7.4 サーバーインスタンスに Red Hat Process Automation Manager がインストールされている。
- Red Hat Decision Manager プロジェクトが外部の Git リポジトリに存在している。
- 外部 Git リポジトリへの読み取りアクセスに必要な認証情報。
- (Windows の場合) Cygwin は、インストール時に追加される Git パッケージでインストールします。Cygwin `/bin` フォルダへのパスは、お使いの環境の **PATH** 変数に追加されます。たとえば、**C:\cygwin64\bin** です。Cygwin のインストールに関する詳細は、[Installing and Updating Cygwin Packages](#) を参照してください。

手順

1. Business Central で、**Menu** → **Projects** に移動します。
2. Git プロジェクトをインポートするスペースを選択または作成します。
3. 画面の右側の  をクリックして、**Import Project** を選択します。
4. **Import Project** ウィンドウで、Git リポジトリの URL を入力します (例: https://github.com/USERNAME/REPOSITORY_NAME.git および Git リポジトリの認証情報)。
5. **Import** をクリックします。
プロジェクトを Business Central の Git リポジトリに追加すると、スペースで使用できるようになります。



重要

SCP スタイルの SSH URL の代わりに、HTTPS または Git プロトコルを使用します。Business Central は基本的な SSH URL をサポートしないため、この URL を使用する場合はエラーが発生します。

公開 SSH キーを Git プロバイダーに設定する必要があります。

Git リポジトリは、Red Hat Decision Manager のバージョンと互換性のある KJAR が1つだけ含まれる、KJAR プロジェクトでなければなりません。KJAR コンテンツは、リポジトリのルートに配置する必要があります。

6. コマンド端末で、プロジェクトのリポジトリ Git ディレクトリーにある **hooks** ディレクトリーに移動します。以下に例を示します。

```
$ cd _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks
```

7. 以下のように、リモート Git リポジトリに変更をプッシュする **post-commit** ファイルを作成します。以下に例を示します。

```
#!/bin/sh
git push origin +master
```

post-commit の Git フックの作成に関する情報は、「[post-commit の Git フックの作成](#)」を参照してください。

8. 必要に応じて、設定が正常に行われたことを確認するには、Business Central のガイド付きルールを作成します。
 - a. Business Central で **Menu → Projects → Add Asset → Guided Rule** に移動します。
 - b. **Create new Guided Rule** ページで必要な情報を入力します。
 - c. **OK** をクリックします。
Business Central は、リモートリポジトリにすべての変更を自動的にプッシュします。

関連情報

- [Customizing Git - Git Hooks](#)

37.3. 既存のリモート GIT プロジェクトリポジトリ用の GIT フックの設定

既存のリモート Git リポジトリプロジェクトがある場合には、その既存のプロジェクトのリモート Git リポジトリに post-commit の Git フックを作成し、リモート Git リポジトリを Business Central に統合できます。

前提条件

- Red Hat JBoss EAP 7.4 サーバーインスタンスに Red Hat Process Automation Manager がインストールされている。
- Red Hat Decision Manager プロジェクトが外部の Git リポジトリに存在している。
- 外部 Git リポジトリへの読み取りアクセスに必要な認証情報。

- (Windows オペレーティングシステムの場合) インストール時に追加される Git パッケージで Cygwin がインストールされており、Cygwin **/bin** ディレクトリーへのパスがお使いの環境の **PATH** 変数に追加されている。たとえば、**C:\cygwin64\bin** です。Cygwin のインストールに関する詳細は、[Installing and Updating Cygwin Packages](#) を参照してください。

手順

1. コマンド端末で、プロジェクトのリポジトリー Git ディレクトリーにある **hooks** ディレクトリーに移動します。以下に例を示します。

```
$ cd _EAP_HOME_/bin/.niogit/<SPACE>/<PROJECT_NAME>.git/hooks
```

2. 以下のように、リモート Git リポジトリーに変更をプッシュする **post-commit** ファイルを作成します。以下に例を示します。

```
#!/bin/sh
git push origin +master
```

post-commit の Git フックの作成に関する情報は、「[post-commit の Git フックの作成](#)」を参照してください。

3. 必要に応じて、設定が正常に行われたことを確認するには、Business Central のガイド付きルールを作成します。
 - a. Business Central で **Menu** → **Projects** → **Add Asset** → **Guided Rule** に移動します。
 - b. **Create new Guided Rule** ページで必要な情報を入力します。
 - c. **OK** をクリックします。
Business Central は、リモートリポジトリーにすべての変更を自動的にプッシュします。

37.4. BUSINESS CENTRAL のシステムプロパティーとしての GIT フックの設定

既存の Git リポジトリープロジェクトがない場合や、post-commit の Git フックを多数のプロジェクトリポジトリーに適用する場合は、**org.uberfire.nio.git.hooks** システムプロパティーの値に、フックファイルが含まれるディレクトリーを指定できます。このディレクトリーは Git リポジトリーにコピーされます。



注記

org.uberfire.nio.git.hooks システムプロパティーを指定した場合は、すべての Business Central 内部リポジトリーおよびプロジェクトリポジトリーで post-commit の Git フックが使用されます。スクリプトでは完全修飾パスのみを使用する必要があります。

前提条件

- Red Hat JBoss EAP 7.4 サーバーインスタンスに Red Hat Process Automation Manager がインストールされている。
- (Windows オペレーティングシステムの場合) インストール時に追加される Git パッケージで Cygwin がインストールされており、Cygwin **/bin** ディレクトリーへのパスがお使いの環境の **PATH** 変数に追加されている。たとえば、**C:\cygwin64\bin** です。Cygwin のインストールに関する詳細は、[Installing and Updating Cygwin Packages](#) を参照してください。

手順

- ローカルシステムのディレクトリーに post-commit の Git フックを作成します。
post-commit の Git フックの作成に関する情報は、「[post-commit の Git フックの作成](#)」を参照してください。
- org.uberfire.nio.git.hooks** システムプロパティーの値に、フックファイルが含まれるディレクトリーを指定するには、以下のいずれかのタスクを実行します。

- **org.uberfire.nio.git.hooks** システムプロパティーを **standalone.xml** ファイルに追加します。以下に例を示します。

```
<system-properties>
  <property name="org.uberfire.nio.git.hooks" value="_EAP_HOME_/hooks">
  </property>
  ...
</system-properties>
```

- Business Central の実行時に、**-Dorg.uberfire.nio.git.hooks** 環境変数を使用します。以下に例を示します。

```
$ ./standalone.sh -c standalone-full.xml -
Dorg.uberfire.nio.git.hooks=_EAP_HOME_/hooks
```

- Business Central を起動します。
post-commit の Git フックは、Business Central の内部リポジトリとプロジェクトのリポジトリすべてにコピーされます。

関連情報

- [Customizing Git - Git Hooks](#)

37.5. リモート GIT リポジトリの統合

以下の例では、post-commit の Git フックと Java コードを使用して、Business Central とリモートの Git リポジトリを統合します。Java コードの例は、[Business Central post-commit Git Hooks Integration](#) を参照してください。この例では、以下の機能を提供します。

- テンプレート **.gitremote** 設定ファイルを自動生成
- 必須のパラメーターに対する **.gitremote** 設定ファイルを検証
- **.gitremote** ファイルの ignore パラメーターで定義するパターンを Git で無視
- ユーザーにメッセージおよび通知を出力
- GitLab および GitHub トークン認証をサポート
- GitLab グループおよびサブグループプロジェクト作成をサポート
- GitHub 組織リポジトリ作成をサポート

前提条件

- Red Hat JBoss EAP 7.4 サーバーインスタンスに Red Hat Process Automation Manager がインストールされている。
- Java Development Kit (JDK) 8 がインストールされている。
- Maven がインストールされている。

手順

1. 端末ウィンドウで、GitHub リポジトリをシステムにクローンします。

```
$ git clone https://github.com/kiogroup/bc-git-integration-push.git
```

2. クローンしたリポジトリに移動します。

```
$ cd bc-git-integration-push
```

3. Maven の新規インストールを実行します。

```
$ mvn clean install
```

4. **EAP_HOME** ディレクトリに **/hooks** ディレクトリを作成します。

```
$ mkdir -p _EAP_HOME_/hooks/
```

5. **git-push-2.1-SNAPSHOT.jar** を **EAP_HOME/hooks/** ディレクトリにコピーします。

```
$ cp bc-git-integration-push/target/git-push-2.1-SNAPSHOT.jar _EAP_HOME_/hooks/
```

6. 必要に応じて、テンプレートの **.gitremote** 設定ファイルを作成するには **git-push-2.1-SNAPSHOT.jar** を実行します。

```
$ java -jar git-push-2.1-SNAPSHOT.jar
```

テンプレート **.gitremote** 設定ファイルの例

```
#This is an auto generated template empty property file
provider=GIT_HUB
login=
password=
token=
remoteGitUrl=https://api.github.com/
useSSH=false
ignore=.*demo.*, test.*
githubOrg=OrgName
gitlabGroup=Group/subgroup
```

7. **.gitremote** 設定ファイルパラメーターを変更します。

表37.1 **.gitremote** パラメーターの例

パラメーター	説明
provider	Git プロバイダー。許容値は GIT_HUB と GIT_LAB の 2 つのみ。必須。
login	Git プロバイダーのユーザー名。必須。
password	プレーンテキストのパスワードです。 token が指定されている場合は必要ありません。
token	username および password ベースの、安全対策がされていない接続の代替りとなる生成トークン。注記: これが設定されていない場合は、セキュアでない接続を使用している旨の警告が表示されます。 password が指定されている場合は必須ではありません。注記: GitLab はトークン認証のみをサポートします。
remoteGitUrl	パブリックのプロバイダー URL またはプロバイダー用にローカルでホストされたエンタープライズ URL。必須。注記: 公開 GitHub URL は API URL である必要があります。たとえば、api.github.com です。
useSSH	SSH プロトコルがリモートリポジトリに変更をプッシュできるようにするブール型。任意。デフォルト値 = false。注記: このパラメーターはローカルの ~/.ssh/ ディレクトリーを使用して SSH 設定を取得します。
ignore	これらの式のいずれかに一致するプロジェクト名を無視するために、コマ区切りの正規表現。任意。
githubOrg	プロバイダーとして GitHub を使用する場合にリポジトリ組織を定義します。任意。
gitlabGroup	プロバイダーとして GitLab を使用する場合にリポジトリグループおよびサブグループを定義します。任意。

8. **EAP_HOME/hooks** に、 **post-commit** Git フックファイルを作成します。

```
$ touch post-commit
```

9. **post-commit** ファイルのパーミッションを **755** に設定します。

```
$ chmod 755 post-commit
```

10. **#!/bin/bash** と **git-push-2.1-SNAPSHOT.jar** を実行するコードを **post-commit** ファイルに追加します。

```
$ echo "#!/bin/bash\njava -jar $APP_SERVER_HOME/hooks/git-push-2.1-SNAPSHOT.jar" > hooks/post-commit
```

11. **-Dorg.uberfire.nio.git.hooks** 環境変数を設定して、Business Central を起動します。以下に例を示します。

```
$ ./standalone.sh -c standalone-full.xml -Dorg.uberfire.nio.git.hooks=_EAP_HOME_/hooks
```



注記

post-commit の Git フックを使用して Java コードを実行するには、以下の Java ライブラリーを使用する必要があります。

- [JGit](#): 内部の Business Central Git リポジトリと対話するのに使用します。
- [GitHub API for Java](#): GitHub との通信に使用します。

post-commit の Git フックと Java コードの例に関する情報は、[Business Central post-commit Git Hooks Integration](#) を参照してください。

37.6. GIT フックの終了コード

Git フックの終了時には、整数値が返され、Git フックの実行ステータスを判断します。この整数値は、Git フックの終了コードとして知られています。実行ステータスは、成功 (1)、警告 (2 から 30) または失敗 (31 から 255) です。

37.7. GIT フック通知のカスタマイズ

Business Central には、フックの終了コードをもとに、カスタマイズした Git フック通知をユーザーが受信可能なメカニズムがあります。

この通知メカニズムを有効にするには、カスタムのメッセージを含む ***.properties** ファイルを作成してから、**appformer.git.hooks.bundle** システムプロパティーの値として対象のファイルのパスを指定する必要があります。

手順

1. ***.properties** ファイルを作成して、各終了コードに以下の形式の適切なメッセージを 1 行追加してください。

```
<exit_code>=<display_message>
```

<exit_code> は、Git フックの終了コードに、**<display_message>** はユーザーに表示するカスタムメッセージに置き換えます。

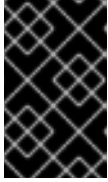
以下に例を示します。

```
0=Success! All working as expected.
1=Warning! Please check the logs and advise your admin.
.
.
31=Error! Please advise your admin immediately.
```



注記

考えられる終了コードをすべて *.properties ファイルに定義する必要はありません。*.properties ファイルに定義されている終了コードに対する通知のみが表示されます。



重要

通知サービスは、プロパティファイルに設定されている **ISO 8859-1 (LATIN 1)** 文字のみサポートします。拡張文字を使用する場合には、Unicode 文字コードのエスケープシーケンスを使用してください。

- Git フック通知を有効化するには、**appformer.git.hooks.bundle** システムプロパティの値としてファイルへのパスを指定します。

Messages.properties ファイルを参照する設定が含まれている、以下の **standalone.xml** ファイル例を確認してください。

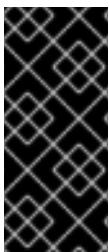
```
<system-properties>
  <property name="appformer.git.hooks.bundle" value="/opt/jboss-as/git-hooks-
  messages/Messages.properties">
  </property>
  ...
</system-properties>
```

37.7.1. Business Central の Git フック通知

Business Central で Git フックの通知を確認できます。Git フックの終了コードの通知タイプは 3 種類あります。

表37.2 Git フックの UI 通知タイプ

終了コード	カスタマイズしたメッセージ	UI での通知の色
0	Success!All working as expected. (成功!すべて想定どおりに機能しています。)	緑
1 から 30	Warning!Please check the logs and advise your admin. (警告!ログを確認して管理者に連絡してください。)	オレンジ
31 から 255	Error!Please advise your admin immediately. (エラー!今すぐ管理者に連絡してください。)	赤



重要

UNIX マシンがサポートするエラーコードは 0 (success) から 255 (error) までで、この範囲外の終了コードは別のコードに変換され、誤った通知メッセージが表示される可能性があります。

Windows マシンには、制限がなく、幅広い終了コードをサポートします。

37.7.2. Git フック通知の国際化サポート

通知メッセージの国際化には、**appformer.git.hooks.bundle** システムプロパティとして指定した元のプロパティファイルと同じパスに、追加でプロパティファイルを配置してください。

各種ローカライズファイルの名前は、**<filename>_<lang>.properties** に指定し、**<filename>** は元のファイルと同じファイル名にしてください。たとえば、システムプロパティが **Messages.properties**

を参照している場合、英語は **Messages_en.properties**、フランス語は **Messages_fr.properties**、またはイタリア語は **Messages_it.properties** として作成できます。

通知サービスは、ユーザー言語をもとにプロパティファイルを選択します。対象言語の翻訳がない場合は、元の **Messages.properties** ファイルからのエントリーを使用します。

第38章 BUSINESS CENTRAL のブランチでのロールベースアクセス制御

Business Central には、特定のコラボレータータイプのターゲットブランチに対するアクセスを制限するオプションがあります。セキュリティーチェックは、**Security Management** の画面とコントリビューターソースの両方を使用してスペースやプロジェクトに対してパーミッションを与えたり、拒否したりします。たとえば、コントリビュータータイプをもとにして、プロジェクトを更新するパーミッションと、対象のブランチへの書き込みパーミッションがある場合は、新規アセットを作成できます。

38.1. ロールベースのブランチアクセスのカスタマイズ

Business Central では、プロジェクトのブランチごとにコントリビューターロールのパーミッションをカスタマイズできます。たとえば、ブランチに割り当てたロールごとに、**Read**、**Write**、**Delete**、および **Deploy** のアクセス権を設定できます。

手順

1. Business Central で、**Menu** → **Design** → **Projects** に移動します。
2. 必要に応じて、新規コントリビューターを追加します。
 - a. プロジェクト名をクリックし、**Contributors** タブをクリックします。
 - b. **Add Contributor** をクリックします。
 - c. テキストフィールドにユーザー名を入力します。
 - d. ドロップダウンリストから **Contributor** のロールタイプを選択します。
 - e. **OK** をクリックします。
3. 関連のコントリビューターのブランチへのアクセス権限をロールベースでカスタマイズします。
 - a. **Settings** → **Branch Management** をクリックします。
 - b. ドロップダウンリストからブランチ名を選択します。
 - c. **Role Access** のセクションで、パーミッションのチェックボックスを選択または選択解除して、利用可能なロールタイプごとにロールベースのブランチアクセスを指定します。
 - d. **Save** をクリックし、再度 **Save** をクリックして変更を確定します。

第39章 プロセスインスタンスログの表示

Logs タブから、インスタンスのプロセスイベントをすべて表示できます。インスタンスログは、現在と過去のプロセスの状態をすべて表示します。Business Central には、**Business** と **Technical** ログの2種類のプロセスインスタンスログがあります。

手順

1. Business Central で、**Menu** → **Manage** → **Process Instances** に移動します。
2. **Manage Process Instances** ページで、表示するログのプロセスインスタンスをクリックします。
3. **Logs** タブを選択します。
 - ビジネスイベントログを表示するには、**Business** をクリックします。
 - テクニカルイベントログを表示するには、**Technical** をクリックします。
 - **Asc** または **Desc** をクリックすると、ログファイルの順序が変わります。

第40章 BUSINESS CENTRAL システムプロパティー

このセクションに記載の Business Central のシステムプロパティーは **standalone*.xml** ファイルに渡されます。

Git ディレクトリー

以下のプロパティーを使用して、Business Central Git ディレクトリーの場所と名前を設定します。

- **org.uberfire.nio.git.dir**: Business Central の Git ディレクトリーの場所。
- **org.uberfire.nio.git.dirname**: Business Central の Git ディレクトリーの名前。デフォルト値は **.niogit** です。
- **org.uberfire.nio.git.ketch**: Git ketch を有効化または無効化。
- **org.uberfire.nio.git.hooks**: Business Central の Git ディレクトリーの場所。

HTTP 経由の Git

次のプロパティーを使用して、HTTP 経由で Git リポジトリーにアクセスできるように設定します。

- **org.uberfire.nio.git.proxy.ssh.over.http**: SSH が HTTP プロキシを使用するかどうかを指定します。デフォルト値は **false** です。
- **http.proxyHost**: HTTP プロキシのホスト名を定義します。デフォルト値は **null** です。
- **http.proxyPort**: HTTP プロキシのホストポート (整数値) を定義します。デフォルト値は **null** です。
- **http.proxyUser**: HTTP プロキシ名を定義します。
- **http.proxyPassword**: HTTP プロキシのユーザーパスワードを定義します。
- **org.uberfire.nio.git.http.enabled**: HTTP デモンを有効または無効にします。デフォルト値は **true** です。
- **org.uberfire.nio.git.http.host**: このデーモンは、HTTP デモンが有効な場合にホストの識別子としてこのプロパティーを使用します。これは、HTTP 経由で Git リポジトリーにアクセスする方法を表示するときに使用する参考属性です。HTTP は、継続してサブレットコンテナに依存します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.http.hostname**: HTTP デモンが有効な場合に、このデーモンはホスト名の識別子としてこのプロパティーを使用します。これは、HTTP 経由で Git リポジトリーにアクセスする方法を表示するときに使用する参考属性です。HTTP は、継続してサブレットコンテナに依存します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.http.port**: このデーモンは、HTTP デモンが有効な場合にポート番号としてこのプロパティーを使用します。これは、HTTP 経由で Git リポジトリーにアクセスする方法を表示するときに使用する参考属性です。HTTP は、継続してサブレットコンテナに依存します。デフォルト値は **8080** です。

HTTPS 経由の Git

次のプロパティーを使用して、HTTPS 経由で Git リポジトリーにアクセスできるように設定します。

- **org.uberfire.nio.git.proxy.ssh.over.https**: SSH が HTTPS プロキシを使用するかどうかを指定します。デフォルト値は **false** です。

- **https.proxyHost**: HTTPS プロキシのホスト名。デフォルト値は **null** です。
- **https.proxyPort**: HTTPS プロキシのポート (整数値)。デフォルト値は **null** です。
- **https.proxyUser**: HTTPS プロキシ名を定義します。
- **https.proxyPassword**: HTTPS プロキシのユーザーパスワードを定義します。
- **user.dir**: ユーザーディレクトリーの場所。
- **org.uberfire.nio.git.https.enabled**: HTTPS デーモンを有効または無効にします。デフォルト値は **false** です。
- **org.uberfire.nio.git.https.host**: このデーモンは、HTTPS デーモンが有効な場合にホストの識別子としてこのプロパティを使用します。これは、HTTPS 経由で Git リポジトリにアクセスする方法を表示するときに使用する参考属性です。HTTPS は、継続してサブレットコンテナに依存します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.https.hostname**: このデーモンは、HTTPS デーモンが有効な場合にホスト名の識別子としてこのプロパティを使用します。これは、HTTPS 経由で Git リポジトリにアクセスする方法を表示するときに使用する参考属性です。HTTPS は、継続してサブレットコンテナに依存します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.https.port**: このデーモンは、HTTPS デーモンが有効な場合にポート番号としてこのプロパティを使用します。これは、HTTPS 経由で Git リポジトリにアクセスする方法を表示するときに使用する参考属性です。HTTPS は、継続してサブレットコンテナに依存します。デフォルト値は **8080** です。

JGit

- **org.uberfire.nio.jgit.cache.instances**: JGit キャッシュサイズを定義します。
- **org.uberfire.nio.jgit.cache.overflow.cleanup.size**: JGit キャッシュオーバーフローのクリーンアップサイズを定義します。
- **org.uberfire.nio.jgit.remove.eldest.iterations**: 最も古い JGit の反復を削除するかどうかを定義します。
- **org.uberfire.nio.jgit.cache.evict.threshold.duration**: JGit 退避のしきい値の期間を定義します。
- **org.uberfire.nio.jgit.cache.evict.threshold.time.unit**: JGit 退避のしきい値の時間単位を定義します。

Git デーモン

次のプロパティを使用して、Git デーモンを有効にして設定します。

- **org.uberfire.nio.git.daemon.enabled**: Git デーモンを有効または無効にします。デフォルト値は **true** です。
- **org.uberfire.nio.git.daemon.host**: Git デーモンが有効な場合は、このプロパティをローカルホストの識別子として使用します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.daemon.hostname**: Git デーモンが有効な場合は、このプロパティをローカルホスト名の識別子として使用します。デフォルト値は **localhost** です。

- **org.uberfire.nio.git.daemon.port**: Git デーモンが有効な場合は、このプロパティをポート番号として使用します。デフォルト値は **9418** です。
- **org.uberfire.nio.git.http.sslVerify**: Git リポジトリを確認する SSL 証明書を有効または無効にします。デフォルト値は **true** です。



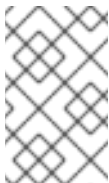
注記

デフォルトポートまたは割り当てられたポートがすでに使用されている場合は、別のポートが自動的に選択されます。ポートが利用可能であることを確認し、詳細についてはログをチェックします。

Git SSH

次のプロパティを使用して、Git SSH デーモンを有効にして設定します。

- **org.uberfire.nio.git.ssh.enabled**: SSH デーモンを有効または無効にします。デフォルト値は **true** です。
- **org.uberfire.nio.git.ssh.host**: SSH デーモンが有効な場合は、このプロパティをローカルホスト識別子として使用します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.ssh.hostname**: SSH デーモンが有効な場合は、このプロパティをローカルホスト名の識別子として使用します。デフォルト値は **localhost** です。
- **org.uberfire.nio.git.SSH.port**: SSH デーモンが有効な場合は、このプロパティをポート番号として使用します。デフォルト値は **8001** です。



注記

デフォルトポートまたは割り当てられたポートがすでに使用されている場合は、別のポートが自動的に選択されます。ポートが利用可能であることを確認し、詳細についてはログをチェックします。

- **org.uberfire.nio.git.ssh.cert.dir**: ローカルの証明書が保存される **.security** ディレクトリーの場所。デフォルトは作業ディレクトリーです。
- **org.uberfire.nio.git.ssh.idle.timeout**: SSH のアイドルタイムアウトを設定します。
- **org.uberfire.nio.git.ssh.passphrase**: SCP スタイルの URL を持つ Git リポジトリのクローンを作成する場合に、オペレーティングシステムの公開キーストアにアクセスするためのパスフレーズ。たとえば、**git@github.com:user/repository.git** です。
- **org.uberfire.nio.git.ssh.algorithm**: SSH で使用されるアルゴリズム。デフォルト値は **RSA** です。
- **org.uberfire.nio.git.gc.limit**: GC の制限を設定します。
- **org.uberfire.nio.git.ssh.ciphers**: コンマ区切りの暗号化の文字列。利用可能な暗号化は **aes128-ctr**、**aes192-ctr**、**aes256-ctr**、**arcfour128**、**arcfour256**、**aes192-cbc**、**aes256-cbc** です。このプロパティを使用しない場合は、すべての暗号化が読み込まれます。
- **org.uberfire.nio.git.ssh.macs**: コンマ区切りのメッセージ認証コード (MAC) の文字列。利用可能な MAC は **hmac-md5**、**hmac-md5-96**、**hmac-sha1**、**hmac-sha1-96**、**hmac-sha2-256**、**hmac-sha2-512** です。このプロパティを使用しない場合は、すべての MAC が読み込まれます。



注記

RSA、または DSA 以外のアルゴリズムを使う場合は、Bouncy Castle JCE ライブラリーを使用するようにアプリケーションサーバーを設定します。

KIE Server ノードおよび Process Automation Manager コントローラー

以下のプロパティーを使用して Process Automation Manager コントローラーから KIE Server ノードへの接続を設定します。

- **org.kie.server.controller:** この URL は Process Automation Manager コントローラーへの接続に使用されます。たとえば、**ws://localhost:8080/business-central/websocket/controller** などです。
- **org.kie.server.user:** Process Automation Manager コントローラーから KIE Server ノードへの接続時に使用するユーザー名。このプロパティーは、この Business Central システムを Process Automation Manager コントローラーとして使用する場合に限り必要になります。
- **org.kie.server.pwd:** Process Automation Manager コントローラーから KIE Server ノードに接続する際に使用するパスワード。このプロパティーは、この Business Central システムを Process Automation Manager コントローラーとして使用する場合に限り必要になります。

Maven など

以下のプロパティーを使用して、Maven などの機能を設定します。

- **kie.maven.offline.force:** Maven のオフライン動作を強制します。true に設定すると、オンラインの依存関係解決が無効になります。デフォルト値は **false** です。



注記

このプロパティーは、Business Central にのみ使用してください。他のコンポーネントとランタイム環境を共有する場合は、設定を分離して、Business Central にだけ適用してください。

- **org.uberfire.gzip.enable: GzipFilter** 圧縮フィルターで Gzip の圧縮を有効にするか、無効にします。デフォルト値は **true** です。
- **org.kie.workbench.profile:** Business Central プロファイルを選択します。許容値は、**FULL** または **PLANNER_AND_RULES** です。プリフィックス **FULL_** で、プロファイルを設定し、管理者設定にこのプロファイルの設定が表示されないようにします。デフォルト値は **FULL** です。
- **org.appformer.m2repo.url:** Business Central は依存関係を検索する時に、Maven リポジトリのデフォルトの場所を使用します。デフォルト値は、**http://localhost:8080/business-central/maven2** など、Business Central 内の Maven リポジトリを参照します。このプロパティーは、Business Central が起動する前に設定してください。デフォルト値は、内部の **m2** リポジトリへのファイルパスです。
- **appformer.ssh.keystore:** クラス名を指定して Business Central で使用する、カスタムの SSH キーストアを定義します。このプロパティーが指定されていない場合はデフォルトの SSH キーストアを使用します。
- **appformer.ssh.keys.storage.folder:** このプロパティーは、デフォルトの SSH キーストアを使用する場合にユーザーの SSH 公開鍵の保存フォルダーを定義します。このプロパティーを指定しないと、この公開鍵は Business Central の **.security** フォルダーに保存され

ます。

- **appformer.experimental.features**: 実験的機能のフレームワークを有効にします。デフォルト値は **false** です。
- **org.kie.demo**: GitHub 外部へのデモアプリケーションのクローン作成を有効にします。
- **org.uberfire.metadata.index.dir**: Lucene の **.index** ディレクトリーが保存される場所。デフォルトは作業ディレクトリーです。
- **org.uberfire ldap.regex.role_mapper**: LDAP プリンシパル名をアプリケーションのロール名にマッピングするのに使用する regex パターン。プリンシパルの値とロール名が一致する場合は、アプリケーションのロール名が変数ロールに置き換えられるため、変数ロールはパターンの一部でなければならない点に注意してください。
- **org.uberfire.sys.repo.monitor.disabled**: 設定モニターを無効にします。無効にした場合の影響を正しく理解していない場合は、無効にしないでください。デフォルト値は **false** です。
- **org.uberfire.secure.key**: パスワードの暗号化で使用するパスワード。デフォルト値は **org.uberfire.admin** です。
- **org.uberfire.secure.alg**: パスワードの暗号化で使用する暗号化アルゴリズム。デフォルト値は **PBEWithMD5AndDES** です。
- **org.uberfire.domain**: uberfire が使用するセキュリティドメイン名。デフォルト値は **ApplicationRealm** です。
- **org.guvnor.m2repo.dir**: Maven リポジトリーディレクトリーが保存される場所。デフォルト値は **<working-directory>/repositories/kie** です。
- **org.guvnor.project.gav.check.disabled**: グループ ID、アーティファクト ID、およびバージョン (GAV) のチェックを無効にします。デフォルト値は **false** です。
- **org.kie.build.disable-project-explorer**: Project Explorer で選択したプロジェクトの自動ビルドを無効にします。デフォルト値は **false** です。
- **org.kie.builder.cache.size**: プロジェクトビルダーのキャッシュサイズを定義します。デフォルト値は **20** です。
- **org.kie.library.assets_per_page**: プロジェクト画面のページごとのアセット数をカスタマイズできます。デフォルト値は **15** です。
- **org.kie.verification.disable-dtable-realtime-verification**: デシジョンテーブルのリアルタイム確認および検証を無効にします。デフォルト値は **false** です。

Process Automation Manager コントローラー

以下のプロパティを使用して、Process Automation Manager コントローラーへの接続方法を設定します。

- **org.kie.workbench.controller**: Process Automation Manager コントローラーとの接続に使用する URL。例: **ws://localhost:8080/kie-server-controller/websocket/controller**
- **org.kie.workbench.controller.user**: Process Automation Manager コントローラーのユーザー。デフォルト値は **kieserver** です。

- **org.kie.workbench.controller.pwd**: Process Automation Manager コントローラーのパスワード。デフォルト値は **kieserver1!** です。
- **org.kie.workbench.controller.token**: Process Automation Manager コントローラーとの接続に使用するトークン文字列

Java Cryptography Extension KeyStore (JCEKS)

JCEKS を設定するには、以下のプロパティを使用します。

- **kie.keystore.keyStoreURL**: Java Cryptography Extension KeyStore (JCEKS) の読み込みに使用する URL。たとえば、**file:///home/kie/keystores/keystore.jceks** です。
- **kie.keystore.keyStorePwd**: JCEKS に使用するパスワード。
- **kie.keystore.key.ctrl.alias**: デフォルトの REST Process Automation Manager コントローラーに使用するキーのエイリアス。
- **kie.keystore.key.ctrl.pwd**: デフォルトの REST Process Automation Manager コントローラーのエイリアスのパスワード

レンダリング

以下のプロパティを使用して、Business Central と KIE Server のレンダリングフォームを切り替えます。

- **org.jbpm.wb.forms.renderer.ext**: Business Central と KIE Server のフォームのレンダリングを切り替えます。デフォルトでは、フォームのレンダリングは Business Central が行います。デフォルト値は **false** です。
- **org.jbpm.wb.forms.renderer.name**: Business Central と KIE Server のレンダリングフォームを切り替えることができます。デフォルト値は **workbench** です。

第41章 BUSINESS CENTRAL 使用時のパフォーマンスチューニングに関する考慮点

以下の主要な概念または推奨のプラクティスを使用すると、Business Central の設定および Red Hat Decision Manager のパフォーマンス最適化に役立ちます。本セクションでこの概念についてまとめており、随時、他のドキュメントを相互参照して詳細を説明します。本セクションは、Red Hat Decision Manager の新しいリリースで、必要に応じて拡張または変更します。

開発時には、必ず開発モードを有効にする

KIE Server または特定のプロジェクトを Business Central に設定して、**production** モードまたは **development** モードを使用できます。デフォルトでは、KIE Server および Business Central のすべての新規プロジェクトは開発モードになっています。このモードには、プロジェクトの開発ポリシーに柔軟性をもたせるなど、開発が容易にすすむ機能や、重複した GAV の検出を無効化するなど、開発中の KIE Server のパフォーマンスを最適化する機能が含まれます。Red Hat Decision Manager 環境が確立し、実稼働モードを実行できる準備が完全に整うまで、開発モードを使用してください。

環境モードの設定または重複する GAV の検出の詳細は、以下の資料を参照してください。

- [36章 KIE Server および Business Central での環境モードの設定](#)
- [Red Hat Decision Manager プロジェクトのパッケージ化およびデプロイ](#)

複雑なガイド付きデシジョンテーブルの検証および妥当性確認の無効化

Business Central のデシジョンテーブルの検証および妥当性確認機能は、デフォルトで有効になっています。この機能を使用すると、ガイド付きデシジョンテーブルの検証が容易になりますが、複雑なガイド付きデシジョンテーブルの場合は、この機能が原因でデシジョンエンジンのパフォーマンスが低下してしまう可能性があります。**org.kie.verification.disable-dtable-realtime-verification** のシステムプロパティを **true** に設定して、この機能を無効にできます。

ガイド付きデシジョンテーブルの詳細は、[ガイド付きデシジョンテーブルを使用したデシジョンサービスの作成](#) を参照してください。

大規模なプロジェクトが多数ある場合は、自動ビルドを無効にする

Business Central では、**Project Explorer** のサイドパネルでプロジェクト間を移動すると、選択したプロジェクトが自動でビルドされて、**Alerts** ウィンドウが更新され、プロジェクトのビルドエラーが表示されます。大規模なプロジェクトがある場合や、頻繁に、開発が実際に行われている多数のプロジェクト間を頻繁に切り替える場合は、この機能を使用すると、Business Central とデシジョンエンジンのパフォーマンスに悪影響を与える可能性があります。

プロジェクトの自動ビルドを無効にするには、**org.kie.build.disable-project-explorer** システムプロパティを **true** に設定します。

パート III. BUSINESS CENTRAL のスタンドアロンパースペクティブの使用

ビジネスルールの開発者は、お使いの Web アプリケーションに Business Central からのスタンドアロンパースペクティブを埋め込み、そのパースペクティブを使用してルール、プロセス、デシジョンテーブルなどのアセットを編集できるようになります。

前提条件

- Business Central がデプロイされ、Web およびアプリケーションサーバーで実行されている。
- Business Central にログインしている。

第42章 BUSINESS CENTRAL のスタンドアロンパースペクティブ

Business Central は、アセットの形式に基づいたオーサリングアセットに特化したエディターを提供します。Business Central には、これらのエディターを個別に使用するための機能があります。この機能は、エディターのスタンドアロンパースペクティブモード、または **スタンドアロンパースペクティブ** と呼ばれています。

ビジネスルール開発者は、これを使用して、お使いの Web アプリケーションにスタンドアロンパースペクティブを組み込み、ルール、プロセス、デシジョンテーブルなどのアセットを編集できます。パースペクティブを組み込んだら、Business Central に切り替えずに、所有するアプリケーションのアセットを編集できるようになります。この機能を使用して Web アプリケーションをカスタマイズできます。スタンドアロンパースペクティブの他にも、アプリケーションに組み込みのスタンドアロンのカスタムページ (ダッシュボード) を組み込むこともできます。

ブラウザーに、**standalone** パラメーターおよび **perspective** パラメーターが含まれる特別な Web アドレスを使用してスタンドアロンパースペクティブにアクセスできます。スタンドアロンパースペクティブの Web アドレスには追加パラメーターが含まれる場合もあります。

第43章 スタンドアロンのライブラリーパースペクティブの使用

Business Central のライブラリーパースペクティブを使用して、編集するプロジェクトを選択します。選択したプロジェクトでオーサリング機能をすべて実行することもできます。

スタンドアロンのライブラリーパースペクティブでは、**header=UberfireBreadcrumbsContainer** パラメーターを **使用する** 方法と **使用しない** 方法のいずれかを選択できます。**header** パラメーターが追加されているアドレスでは、ライブラリーパースペクティブの他にブレッドクラムを表示する点が異なります。このリンクを使用して、プロジェクトに追加 **スペース** を作成できます。

手順

1. Business Central にログインします。
2. Web ブラウザーで、適切な Web アドレスを入力します。
 - a. **header** パラメーターが **含まれない** スタンドアロンのライブラリーパースペクティブにアクセスする場合
http://localhost:8080/business-central/kie-wb.jsp?standalone=true&perspective=LibraryPerspective

ブラウザーで、ブレッドクラムのないスタンドアロンのライブラリーパースペクティブが開きます。
 - b. **header** パラメーターが **含まれる** スタンドアロンのライブラリーパースペクティブにアクセスする場合
http://localhost:8080/business-central/kie-wb.jsp?standalone=true&perspective=LibraryPerspective&header=UberfireBreadcrumbsContainer

ブレッドクラムトレイルを含むスタンドアロンライブラリーパースペクティブがブラウザーで開きます。

第44章 スタンドアロンのエディターパースペクティブの使用

Business Central のスタンドアロンのエディターパースペクティブを使用してアセット固有のエディターにアクセスできます。このパースペクティブを使用してアセットエディターを開き、必要に応じてアセットを変更できます。

アセットのスタンドアロンのエディターパースペクティブにアクセスする Web アドレスには、**standalone** パラメーターおよび **path** パラメーターが含まれます。**path** パラメーターにはアセットへの完全パスが必要で、Web アドレスの末尾は **#StandaloneEditorPerspective** 文字列にできます。また、**path** パラメーターを変更する場合は、スタンドアロンモードで特定のアセットのエディターにアクセスできます。

手順

1. Business Central にログインします。
2. Web ブラウザーで、以下のように、必要に応じて適切な Web アドレスを入力します。
 - a. プロセスを編集する場合:
http://localhost:8080/business-central/kie-wb.jsp?standalone&path=default://master@MySpace/Shop/src/main/resources/com/purchase.bpmn#StandaloneEditorPerspective

スタンドアロンモードで Process Designer が開きます。

- b. フォームを編集する場合:
http://localhost:8080/business-central/kie-wb.jsp?standalone&path=default://master@MySpace/Mortgage_Process/src/main/resources/ApplicationMortgage.frm#StandaloneEditorPerspective

スタンドアロンモードで Form Modeler が開きます。

第45章 スタンドアロンのコンテンツマネージャーパースペクティブの使用

アプリケーションで、スタンドアロンのコンテンツマネージャーパースペクティブを使用すると、アプリケーションのコンテンツとそのナビゲーションメニューを作成し、編集できます。

手順

1. Business Central にログインします。
2. Web ブラウザーのアドレスバーに、以下の Web アドレスを入力します。
`http://localhost:8080/business-central/kie-wb.jsp?standalone=true&perspective=ContentManagerPerspective`

ブラウザーで、スタンドアロンのコンテンツマネージャーパースペクティブを開きます。

第46章 カスタムページ (ダッシュページ) の使用

スタンドアロンパースペクティブの他に、アプリケーションに、ダッシュボードとして知られるカスタムページを組み込むこともできます。アプリケーションからカスタムページにアクセスするには、**perspective** パラメーターの値をカスタムページの名前にします。**perspective** パラメーターでは大文字と小文字が区別されます。

手順

1. Business Central にログインします。
2. Web ブラウザーのアドレスバーに、カスタムページの Web アドレスを入力します。以下は例になります。

**http://localhost:8080/business-central/kie-wb.jsp?
standalone=true&perspective=CustomPageName**

ブラウザーにスタンドアロンのカスタムページが開きます。**CustomPageName** の値を、スタンドアロンモードで使用するカスタムページの名前に置き換えます。

パート IV. BUSINESS CENTRAL でのカスタムページの作成

ビジネス分析者またはビジネスルールの開発者は、Business Central の **Page Editor** を使用して静的ページおよび動的ダッシュボードを作成し、プロジェクトに関する特定の情報を表示します。ダッシュボードは、最低でも1つの動的なレポートコンポーネントが含まれるページのコレクションです。ダッシュボードのレポートコンポーネントをフィードするデータセットを定義できます。ダッシュボードは、Red Hat JBoss EAP のスタンドアロンの Dashbuilder Runtime ダッシュボードビューアーまたは Red Hat OpenShift Container Platform の Dashbuilder Standalone ダッシュボードビューアーにエクスポートできます。

前提条件

- ページを編集する権限のあるユーザーとして Business Central にログインしている。

第47章 BUSINESS CENTRAL カスタムダッシュボード

ダッシュボードは、最低でも1つのレポートコンポーネントが含まれる Business Central ページの集合です。ダッシュボードには多くの場合、データセット、ナビゲーションツリー、およびパーミッションが含まれます。

- **データセットオーサリング:** ページを介してデータにアクセスして表示するデータセットを定義します。詳細は [データセットの追加](#) を参照してください。
- **セキュリティ管理:** このステージでは、Business Central で作業中にユーザーが所有する特権を定義するロールおよびグループのパーミッションが設定されます。詳細は [セキュリティ設定](#) を参照してください。

関連情報

Business Central の以前のバージョンからダッシュボードを移行するには、「[ダッシュボードデータのエクスポート](#)」に記載の Dashbuilder のデータ転送機能を使用します。

第48章 データセットのオーサリング

データセットは情報の関連セットの集まりで、データベース、Microsoft Excel、またはメモリーに保存できます。データセット定義は、Business Central メソッドにデータセットへのアクセス、読み取り、および解析を指示します。Business Central はデータを保存しません。データが保存される場所にかかわらず、データセットへのアクセスを定義できます。

たとえば、データベースにデータが保存されると、有効なデータセットには、SQL クエリーの結果として、データベース全体またはデータベースのサブセットなどが含まれます。いずれの場合も、データは、情報を表示する Business Central のレポーティングコンポーネントの入力情報として使用されます。

データセットにアクセスするには、データセット定義を作成および登録する必要があります。このデータセットの定義では、データセットの場所と、その場所へのアクセス、読み取り、および解析のオプション、ならびにデータセットが含まれるコラムを指定します。



注記

Data Sets ページは、admin ロールを持つユーザーにのみ表示されます。

48.1. データセットの追加

外部データソースからデータを取得して、レポーティングコンポーネントでデータを使用するデータセットを作成できます。

手順

1. Business Central で、**Admin** → **Data Sets** に移動します。
Data Sets ページが開きます。
2. **New Data Set** をクリックして、以下のプロバイダタイプから1つ選択します。
 - **Bean**: Java クラスからデータセットを生成します。
 - **CSV**: リモートまたはローカルの CSV ファイルからデータセットを生成します。
 - **SQL**: ANSI-SQL 準拠データベースからデータセットを生成します。
 - **Elastic Search**: Elastic Search ノードからデータセットを生成します。
 - **Prometheus**: Prometheus クエリーを使用してデータセットを生成します。
 - **Kafka**: Kafka ブローカー、コンシューマー、またはプロデューサーからメトリックを使用してデータセットを生成します。



注記

Prometheus、Kafka、および Execution Server オプションに KIE Server を設定する必要があります。

3. **Data Set Creation Wizard** を完了し、**Test** をクリックします。



注記

設定手順は、選択するプロバイダーにより異なります。

4. **Save** をクリックします。

48.2. データセットの編集

既存のデータセットを編集し、レポートコンポーネントに取得したデータが最新になっていることを確認します。

手順

1. Business Central で、**Admin** → **Data Sets** に移動します。
Data Set Explorer ページが開きます。
2. **Data Set Explorer** ペインで、編集するデータセットを検索し、データセットを選択して **Edit** をクリックします。
3. **Data Set Editor** ペインで、適切なタブを使用して必要に応じてデータを編集します。タブは、選択するデータセットプロバイダーの種類によって異なります。
たとえば、**CSV** データプロバイダーの編集には、以下の変更が適用できます。
 - **CSV Configuration**: データセット定義の名前、ソースファイル、区切り記号などのプロパティを変更できます。
 - **Preview**: データのプレビューを使用できます。**CSV Configuration** タブで **Test** をクリックすると、システムはデータセットのルックアップコールを実行し、データが利用可能な場合はプレビューが表示されます。**Preview** タブには2つのサブタブがあります。
 - **Data columns**: どの列をデータセット定義に追加するかを指定できます。
 - **Filter**: 新しいフィルターを追加できます。
 - **Advanced**: 以下の設定を管理できます。
 - **Caching**: 詳細は [キャッシュデータ](#) を参照してください。
 - **Cache life-cycle**: データセット (またはデータ) を再読み込みされるまでの間隔を指定できます。バックエンドデータに変更が加えられると、**Refresh on stale data**機能は、キャッシュしたデータを再読み込みします。
4. 必要な変更を行ったら、**Validate** をクリックします。
5. **Save** をクリックします。

48.3. データの再読み込み

データの再読み込み機能を使用すると、データセット (またはデータ) を再読み込みされるまでの間隔を指定できます。データセットの **Advanced** タブにある **データ更新間隔** 機能にアクセスできます。バックエンドデータに変更が加えられると、**Refresh on stale data**機能は、キャッシュしたデータを再読み込みします。

48.4. データのキャッシュ

Business Central は、インメモリーデータを使用してデータセットを保存し、データ操作を実行するキャッシュメカニズムを提供します。データのキャッシュにより、ネットワークトラフィック、リモートシステムのペイロード、処理時間が減ります。パフォーマンスの問題を回避するには、Decision Central にキャッシュを設定します。

データセットを生成するデータルックアップ呼び出しの場合、キャッシュメソッドは、データルックアップ呼び出しが実行される場所と結果のデータセットが格納される場所を決定します。データのルックアップコールの例としては、ロケールパラメーターを Urban として設定するすべての住宅ローンアプリケーションが挙げられます。

Business Central データセット機能には、キャッシュレベルが2つあります。

- クライアントレベル
- バックエンドレベル

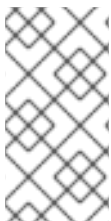
データセットの **Advanced** タブで、**クライアントキャッシュ** および **バックエンドキャッシュ** 設定を指定できます。

クライアントキャッシュ

キャッシュを有効にすると、データセットはルックアップ操作時に Web ブラウザーにキャッシュされ、その後のルックアップ操作ではバックエンドへの要求が実行されません。グループ化、集計、フィルタリング、並べ替えなどのデータセット操作は Web ブラウザーで処理されます。クライアントのキャッシュは、データセットのサイズが小さい場合 (例: データが 10 MB より少ない) にのみ有効になります。データセットが大きい場合は、パフォーマンスの低下や断続的なフリーズなどのブラウザー問題が発生する場合があります。クライアントのキャッシュは、ストレージシステムへの要求などの、バックエンド要求の数を減らします。

バックエンドキャッシュ

キャッシュが有効な場合に、デシジョンエンジンはデータセットをキャッシュします。これにより、リモートのストレージシステムへのバックエンドの要求数が減ります。データセットの全操作は、インメモリーデータを使用してデシジョンエンジンで実行されます。バックエンドキャッシュは、データセットのサイズが頻繁に更新され、インメモリーに保存されて処理される場合に限り有効です。バックエンドキャッシングは、リモートストレージを使用した低レイテンシー接続問題が発生している状況でも有効です。



注記

バックエンドキャッシュの設定は、**Data Set Editor** の **Advanced** タブに常に表示されるわけではありません。これは、インメモリーデシジョンエンジンでデータのルックアップ操作を解決するのに、Java および CSV のデータプロバイダーはバックエンドキャッシュに依存するためです (データセットはメモリー内に存在する必要があります)。

第49章 セキュリティー管理

セキュリティー管理とは、ユーザー、グループ、パーミッションを管理するプロセスです。Business Central セキュリティー管理ページから Business Central のリソースおよび機能へのアクセスを制御できます。

Business Central は、ユーザー、グループ、およびロールのセキュリティー管理のエンティティを 3 種類定義します。パーミッションは、ロールにもグループにも両方割り当てることができます。ユーザーは、所属するグループおよびロールのパーミッションを継承します。

49.1. セキュリティー管理プロバイダー

レルムは、セキュリティー管理のコンテキストで各種アプリケーションリソースへのアクセスを制限します。レルムには、ユーザー、グループ、ロール、パーミッションに関する情報が含まれます。特定のレルムに対する具体的なユーザーおよびグループ管理サービスの実装は、セキュリティー管理プロバイダーと呼ばれます。

組み込みのセキュリティー管理プロバイダーがアプリケーションセキュリティーレルムの要件を満たさない場合は、独自のセキュリティー管理プロバイダーを構築して登録できます。



注記

セキュリティー管理プロバイダーがインストールされていない場合は、セキュリティーレルムを管理するユーザーインターフェイスは利用できません。セキュリティー管理プロバイダーをインストールして設定した後に、セキュリティー管理ユーザーインターフェイスでユーザーおよびグループの管理機能は自動的に有効になります。

Business Central には、**application-users.properties** または **application-roles.properties** プロパティファイルの内容を基にレルムタイプをサポートする Red Hat JBoss EAP セキュリティー管理プロバイダーが含まれます。

49.1.1. プロパティファイルを基にした Red Hat JBoss EAP セキュリティー管理プロバイダーの設定

独自の Red Hat JBoss EAP セキュリティー管理プロバイダーを構築して登録できます。プロパティファイルを基にして Red Hat JBoss EAP セキュリティー管理プロバイダーを使用するには、以下の手順を行います。

前提条件

- Red Hat JBoss EAP がインストールされている。

手順

1. Red Hat JBoss EAP インスタンスの既存のユーザーまたはロールプロパティファイルを使用するには、以下の例で示すように、**EAP_HOME/standalone/configuration/application-users.properties** および **EAP_HOME/standalone/configuration/application-roles.properties** ファイルに以下のシステムプロパティを含めます。

```
<property name="org.uberfire.ext.security.management.wildfly.properties.realm"
value="ApplicationRealm"/>
<property name="org.uberfire.ext.security.management.wildfly.properties.users-file-path"
```

```
value="/standalone/configuration/application-users.properties"/>
<property name="org.uberfire.ext.security.management.wildfly.properties.groups-file-path"
value="/standalone/configuration/application-roles.properties"/>
```

以下の表は、これらのプロパティの説明とデフォルト値を示しています。

表49.1 プロパティファイルを基にする Red Hat JBoss EAP セキュリティー管理プロバイダー

プロパティ	説明	デフォルト値
org.uberfire.ext.security.management.wildfly.properties.realm	レルムの名前このプロパティは必須ではありません。	ApplicationRealm
org.uberfire.ext.security.management.wildfly.properties.users-file-path	ユーザープロパティファイルの絶対パス。このプロパティは必須です。	./standalone/configuration/application-users.properties
org.uberfire.ext.security.management.wildfly.properties.groups-file-path	グループプロパティファイルの絶対パス。このプロパティは必須です。	./standalone/configuration/application-roles.properties

- アプリケーションのルートディレクトリーに **security-management.properties** ファイルを作成します。たとえば、以下のファイルを作成します。

```
src/main/resources/security-management.properties
```

- security-management.properties** ファイルの値として、以下のシステムプロパティおよびセキュリティープロバイダー名を入力します。

```
<property name="org.uberfire.ext.security.management.api.userManagementServices"
value="WildflyUserManagementService"/>
```

49.1.2. プロパティファイルと CLI モードを基にした Red Hat JBoss EAP セキュリティー管理プロバイダーの設定

プロパティファイルと CLI モードを基に Red Hat JBoss EAP セキュリティー管理プロバイダーを使用するには、以下の手順を行います。

前提条件

- Red Hat JBoss EAP がインストールされている。

手順

- Red Hat JBoss EAP インスタンスの既存のユーザーまたはロールプロパティファイルを使用するには、以下の例で示すように、**EAP_HOME/standalone/configuration/application-users.properties** および **EAP_HOME/standalone/configuration/application-roles.properties** ファイルに以下のシステムプロパティを含めます。

```
<property name="org.uberfire.ext.security.management.wildfly.cli.host" value="localhost"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.port" value="9990"/>
```

```
<property name="org.uberfire.ext.security.management.wildfly.cli.user" value="
<USERNAME>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.password" value="
<USER_PWD>"/>
<property name="org.uberfire.ext.security.management.wildfly.cli.realm"
value="ApplicationRealm"/>
```

以下の表は、これらのプロパティーの説明とデフォルト値を示しています。

表49.2 プロパティーファイルと CLI モードを基にする Red Hat JBoss EAP セキュリティー管理プロバイダー

プロパティー	説明	デフォルト値
org.uberfire.ext.security.m anagement.wildfly.cli.host	ネイティブ管理インターフェ イスホスト。	localhost
org.uberfire.ext.security.m anagement.wildfly.cli.port	ネイティブ管理インターフェ イスポート。	9990
org.uberfire.ext.security.m anagement.wildfly.cli.user	ネイティブ管理インターフェ イスのユーザー名。	NA
org.uberfire.ext.security.m anagement.wildfly.cli.pass word	ネイティブ管理インターフェ イスのユーザーのパスワー ド。	NA
org.uberfire.ext.security.m anagement.wildfly.cli.real m	アプリケーションのセキュリ ティーコンテキストで使用さ れるレルム。	ApplicationRealm

2. アプリケーションのルートディレクトリーに **security-management.properties** ファイルを作成します。たとえば、以下のファイルを作成します。

```
src/main/resources/security-management.properties
```

3. **security-management.properties** ファイルの値として、以下のシステムプロパティーおよびセキュリティープロバイダー名を入力します。

```
<property name="org.uberfire.ext.security.management.api.userManagementServices"
value="WildflyCLIUserManagementService"/>
```

49.2. パーミッションおよび設定

パーミッションは、アプリケーション内の特定のリソースに関連するアクションを実行するためにユーザーに付与される権限です。たとえば、以下のパーミッションを指定できます。

- ページを表示する。
- プロジェクトを保存する。
- リポジトリーを削除する。

- ダッシュボードを削除する。

パーミッションは、付与と拒否ができ、グローバルに設定することも、リソースを指定して設定することもできます。パーミッションを使用すると、リソースへのアクセス時のセキュリティーが保護され、アプリケーション内の機能をカスタマイズできます。

49.2.1. Business Central でのグループおよびロールのパーミッションの変更

Business Central では、個人ユーザーに対するパーミッションは変更できません。ただし、グループおよびロールのパーミッションは変更できます。変更したパーミッションは、変更したロールが割り当てられているか、変更したグループに所属するユーザーに適用されます。



注記

ロールまたはグループへの変更は、そのロールまたはグループに関連のあるユーザーに加えられます。

前提条件

- Business Central に **admin** ユーザーロールでログインします。

手順

1. Business Central で **Security management** ページにアクセスするには、画面の右上隅にある **Admin** アイコンを選択します。
2. Business Central **Settings** ページで **Roles**、**Groups**、または **Users** をクリックします。クリックしたアイコンのタブに、**Security management** ページが開きます。
3. リストから編集するロールまたはグループをクリックします。全詳細が右側のペインに表示されます。
4. **Settings** セクションの **Home Page** または **Priority** を設定します。
5. **Permissions** セクションで、Business Central、ページ、エディター、スペース、プロジェクトのパーミッションを設定します。

図49.1パーミッションの設定

admin settings

Home Page ⓘ

Priority ⓘ

Permissions

> Workbench ⓘ

Pages ⓘ

[Add Exception](#)

Editors ⓘ

Spaces ⓘ

Projects ⓘ

6. 変更するパーミッションのリソースタイプの横にある矢印をクリックしてデプロイメントします。
7. 必要に応じて、リソースタイプに例外を追加するには、**Add Exception** をクリックしてから、必要なパーミッションを設定します。



注記

Business Central のリソースタイプには、例外を追加できません。

8. **Save** をクリックします。

49.2.2. Business Central ホームページの変更

ホームページは、Business Central にログインすると表示されるページです。デフォルトでは、ホームページは **Home** に設定されます。ロールとグループ別に異なるホームページを指定できます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Roles** または **Groups** を選択します。
2. ロールまたはグループを選択します。
3. **Home Page** リストからページを選択します。
4. **Save** をクリックします。



注記

そのロールまたはグループには、ページをホームページにする前に、そのページへの読み取りアクセスが必要です。

49.2.3. 優先順位の設定

ユーザーは、複数のロールを持ち、複数のグループに所属します。優先順位の設定は、ロールまたはグループの優先順を決定します。

前提条件

- Business Central に **admin** ユーザーロールでログインします。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Roles** または **Groups** を選択します。
2. ロールまたはグループを選択します。
3. 優先順位メニューから優先順位を選択し、**Save** をクリックします。



注記

ユーザーに、設定が競合するロールが割り当てられているか、グループに所属している場合は、一番高い優先順位を持つロールまたはグループを設定します。

第50章 ダッシュボードのエクスポート、インポート、およびデプロイ

Business Central でダッシュボードを作成したら、Dashboard データをエクスポートし、Business Central の別のインスタンス、Dashbuilder Runtime、または Dashbuilder Standalone でインポートできます。



注記

この機能は管理者ユーザーのみが利用できます。

50.1. ダッシュボードデータのエクスポート

Business Central のデータセットなど、全ダッシュボードデータを Zip ファイルとしてエクスポートできます。

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Dashbuilder Data Transfer** を選択します。
2. **Dashbuilder Data Transfer** ページで **Export all** をクリックします。
すべてのダッシュボードデータを含む **export.zip** ファイルがダウンロードされます。**export.zip** ファイル構造は、以下の例のようにデータタイプで区切られています。

```
dashbuilder/datasets/definitions/dataset-example1.dset
dashbuilder/datasets/definitions/dataset-example2.dset
dashbuilder/datasets/readme.md
VERSION
```

50.2. BUSINESS CENTRAL ダッシュボードのインポート

アーカイブが以下の例と同じように設定されている場合には、Zip ファイルで Dashbuilder データを Business Central にインポートできます。

```
dashbuilder/datasets/definitions/dataset-example1.dset
dashbuilder/datasets/definitions/dataset-example2.dset
dashbuilder/datasets/readme.md
VERSION
```

手順

1. Business Central で、画面の右上隅にある **Admin** アイコンを選択し、**Dashbuilder Data Transfer** を選択します。



警告

既存システムのデータが上書きされないように、新規インストールした Red Hat Decision Manager にのみ、ダッシュボードのデータはインポートするようにしてください。

2. **Dashbuilder Data Transfer** ページで、**Choose File** アイコンをクリックします。
3. インポートする ZIP ファイルに移動し、ファイルを選択します。
4. **Upload** アイコンをクリックします。
5. **Import** をクリックします。

付録A バージョン情報

本書の最終更新日: 2023 年 9 月 5 日 (火)

付録B お問い合わせ先

Red Hat Decision Manager ドキュメントチーム: brms-docs@redhat.com