Red Hat Process Automation Manager 7.8

Business Central でのカスタムタスクとワークアイテムハンドラー

Last Updated: 2020-09-18

Red Hat Process Automation Manager 7.8 Business Central でのカスタムタスクとワークアイテムハンドラー

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

法律上の通知

概要

本書は、Red Hat Process Automation Manager 7.8 の Business Central でカスタムタスクやワークアイテムハンドラーを作成し、カスタマイズして管理する方法を説明します。

目次

前書き

ビジネスルール開発者は、Business Central でカスタムタスクやワークアイテムハンドラーを作成して、プロセスフロー内でカスタムコードを実行し、Red Hat Process Automation Manager で使用できるように操作を拡張することができます。カスタムタスクを使用して、Red Hat Process Automation Manager に直接含まれていない操作を開発して、プロセスダイアグラムに追加できます。

Business Central では、プロセスダイアグラムの各タスクには **WorkItem** Java クラスと、関連付けられた **WorkItemHandler** Java クラスがあります。ワークアイテムハンドラーには、Business Central に登録された Java コードが含まれており、**org.kie.api.runtime.process.WorkItemHandler** を実装します。

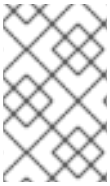
タスクがトリガーされると、ワークアイテムハンドラーの Java コードが実行されます。ワークアイテムハンドラーをカスタマイズして登録し、カスタムタスクで独自の Java コードを実行できます。

前提条件

- Business Central がデプロイされ、Web または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。
- Maven がインストールされている。
- ホストからインターネットにアクセスできる。ビルドプロセスは、インターネットを使用して、外部のリポジトリーから Maven パッケージをダウンロードします。
- お使いのシステムから Red Hat の Maven リポジトリーにローカルまたはオンラインでアクセスできる。

第1章 BUSINESS CENTRAL でのサービスタスクの管理

サービスタスク (ワークアイテム) とは、複数のビジネスプロセスまたは Business Central の全プロジェクトの間でカスタマイズして再利用できるタスクのことです。Red Hat Process Automation Manager は、Business Central のサービスタスクのリポジトリーで、サービスタスクセットを提供します。デフォルトのサービスタスクを有効化または無効化して、カスタムのサービスタスクを 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 Process Automation Manager に含まれるサポート対象のカスタムタスクには限りがあり、Red Hat Process Automation 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

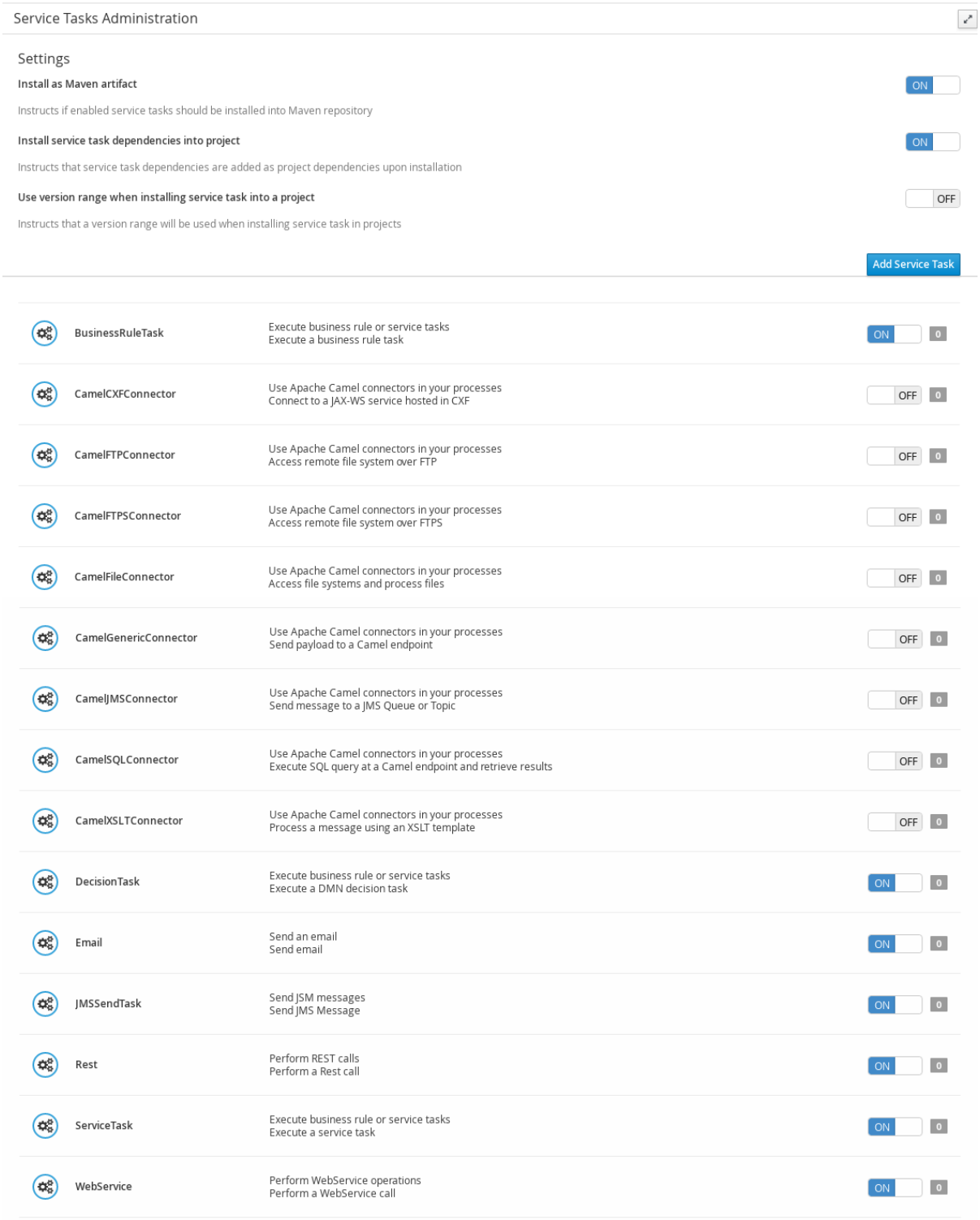
1. Business Central で、右上隅の  をクリックして、**Service Tasks Administration** を選択します。
   このページは、サービスタスクのインストール設定や、Business Central 全体にあるプロジェクトのプロセスで利用可能なサービスタスクを表示します。このページで有効にしたサービスタスクは、プロジェクトレベルの設定で利用できます。プロジェクトレベルの設定で、プロセスで使用する各サービスタスクをインストールできます。サービスタスクをプロジェクトにインストールする方法は、**Service Tasks Administration** ページの **Settings** で有効化または無効化したグローバル設定により決まります。

2. **Settings** で、各設定を有効化または無効化して、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、利用可能なサービスタスクを実装する方法を決定します。
   以下のサービス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には、サービスタスクの JAR ファイルを Maven リポジトリーにアップロードし、Business Central で設定します。
   - **Install service task dependencies into project** サービス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 service task into project** プロジェクトの依存関係として追加するサービス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。例: **7.16.0.Final** ではなく **[7.16,)**

3. 必要に応じて利用可能なサービスタスクを有効化または無効化します (**ON** または **OFF** に設定)。有効化したサービスタスクは、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1.1 サービスタスクおよびサービスタスク設定の有効化



4. カスタムのサービスタスクを追加するには、**Add Service Task**をクリックして、適切な JAR ファイルを参照し、**アップロード** アイコンをクリックします。JAR ファイルには、**@Wid** のアノテーションを指定した作業アイテムハンドラーの実装を含める必要があります。

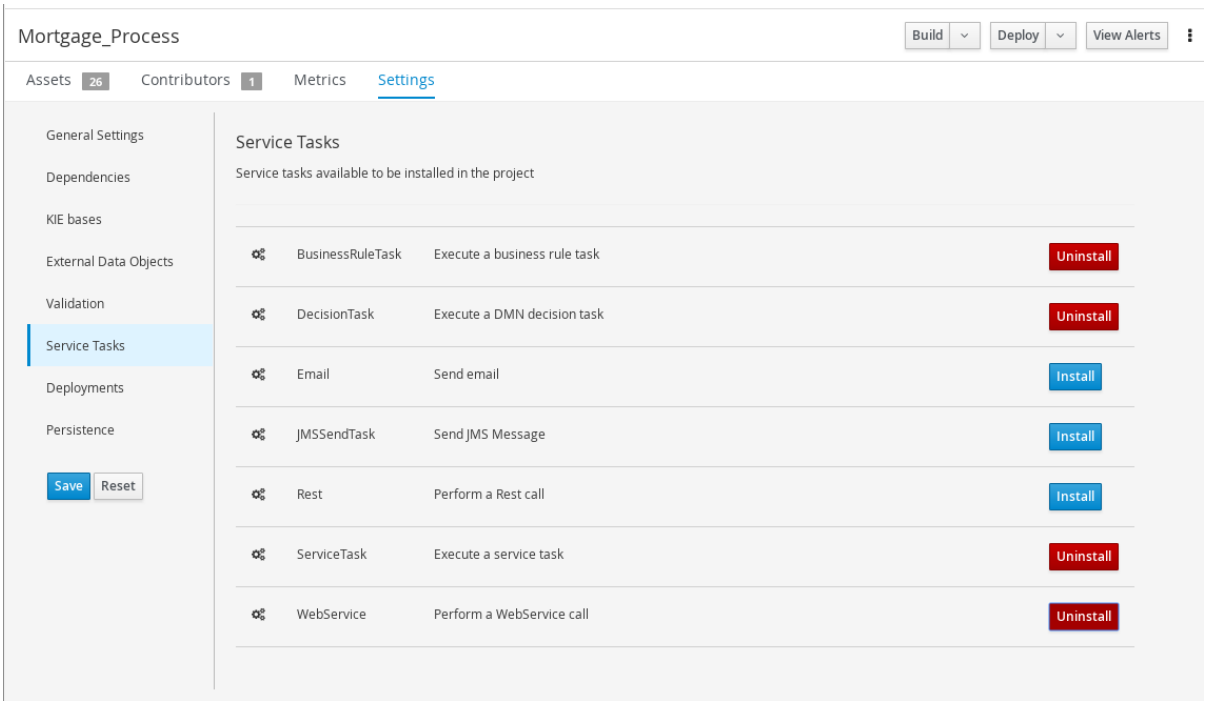
5. オプションでサービスタスクを削除するには、削除するサービスタスクの行にある **remove** をクリックし、**OK** をクリックして削除を確定します。

6. Business Central ですべての必須サービスタスクを設定した後に、プロジェクトの **Settings** → **Service Tasks** ページに移動すると、有効化したサービスタスクで利用可能なものが表示されます。

7. サービスタスクごとに、**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. サービスタスクのインストール時に追加情報を求められた場合には、必要な情報を入力して、もう一度 **Install** をクリックします。
サービス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。他のサービスタスクでは、追加のパラメーターが必要でない場合もあります。

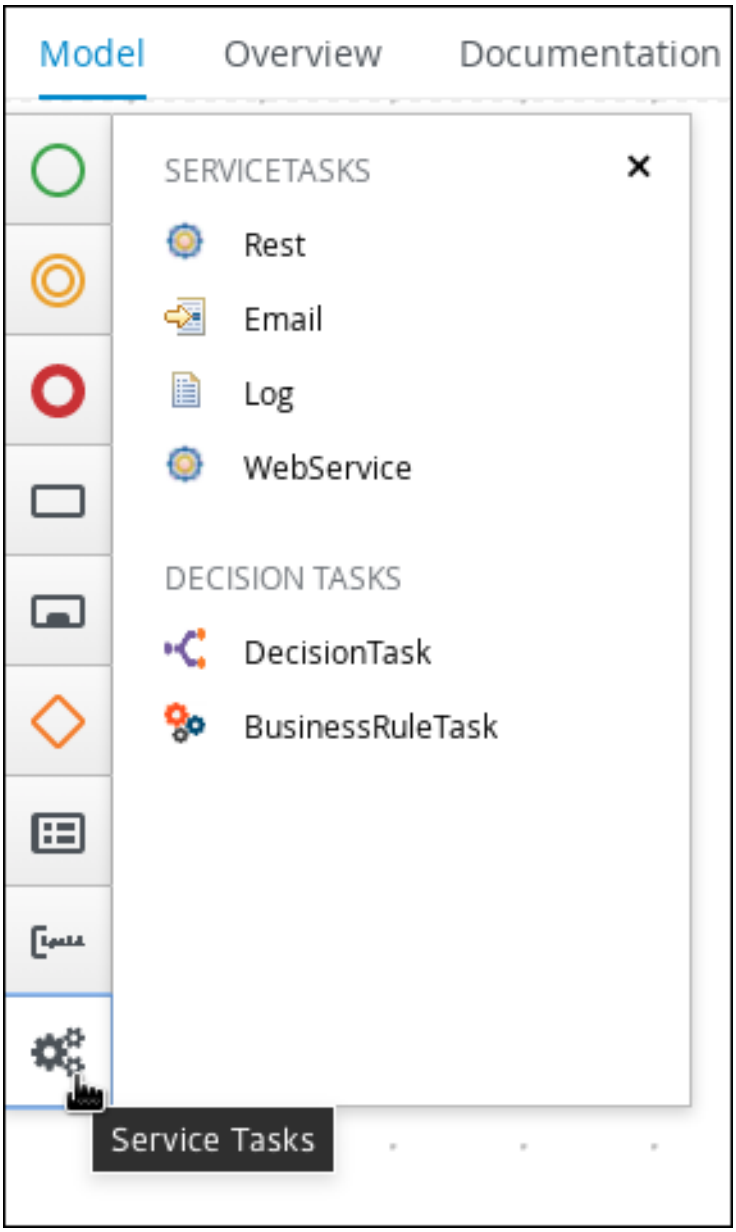
図1.2 プロセスで使用するためのサービスタスクのインストール



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 **Service Tasks** オプションを選択すると、有効化してインストールした、利用可能なサービスタスクが表示されます。

図1.3 プロセスデザイナーでのインストール済みサービスタスクのアクセス



第2章 ワークアイテムハンドラーのプロジェクト作成

カスタムタスクの設定、マッピング、実行可能なコードをすべて含めてソフトウェアプロジェクトを作成します。

最初からワークアイテムハンドラーを作成するか、Maven アーキタイプを使用してサンプルプロジェクトを作成できます。Red Hat Process Automation Manager では、この目的のために、Red Hat Maven リポジトリーから **jbpm-workitems-archetype** を提供します。

手順

1. コマンドラインを開き、**workitem-home** などのワークアイテムハンドラーをビルドするディレクトリーを作成します。

```
$ mkdir workitem-home
```

2. Maven **settings.xml** ファイルを確認して、Red Hat Maven リポジトリーがリポジトリー一覧に含まれていることを確認します。

注記

Maven の設定は、本書の対象外となります。

たとえば、オンラインの Red Hat Maven リポジトリーを Maven **settings.xml** ファイルに追加します。

```
<settings>
  <profiles>
    <profile>
      <id>my-profile</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <id>redhat-ga</id>
          <url>http://maven.repository.redhat.com/ga/</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
        ...
      </repositories>
    </profile>
  </profiles>
  ...
</settings>
```

3. Red Hat ライブラリーバージョンを検索して、以下のタスクの 1 つを実行します。

- オンラインでライブラリーのバージョンを検索する場合には、「What is the mapping between Red Hat Process Automation Manager and the Maven library version?」を参照してください。
- オフラインでライブラリーのバージョンを検索するには、**business-central.war/META-INF/MANIFEST.MF** の **Implementation-Version** か、**kie-server.war/META-INF/MANIFEST.MF** の **Implementation-Version** を確認してください。

4. **workitem-home** ディレクトリーで、以下のコマンドを実行します。

```
$ mvn archetype:generate \
-DarchetypeGroupId=org.jbpm \
-DarchetypeArtifactId=jbpm-workitems-archetype \
-DarchetypeVersion=<redhat-library-version> \
-Dversion=1.0.0-SNAPSHOT \
-DgroupId=com.redhat \
-DartifactId=myworkitem \
-DclassPrefix=MyWorkItem
```

表2.1 パラメーターの説明

パラメーター	説明
-DarchetypeGroupId	アーキタイプ固有ですので、変更しないようにしてください。
-DarchetypeArtifactId	アーキタイプ固有ですので、変更しないようにしてください。
-DarchetypeVersion	Maven が **jbpm-workitems-archetype** アーティファクトをダウンロードしようとすると、検索される Red Hat ライブラリーのバージョン
-Dversion	**1.0.0-SNAPSHOT** など、固有のプロジェクトバージョン。
-DgroupId	**com.redhat** など、固有のプロジェクトの Maven グループ。
-DartifactId	**myworkitem** など、固有のプロジェクトの Maven ID。
-DclassPrefix	**MyWorkItem** など、簡単に特定できるように Maven がクラスを生成したときに、Java クラスの最初に追加される文字列。

以下のように、**myworkitem** フォルダーは、**workitem-home** ディレクトリーで作成されます。

```
assembly/
  assembly.xml
src/
  main/
```

```
    java/
      com/
        redhat/
          MyWorkItemWorkItemHandler.java
    repository/
    resources/
  test/
    java/
      com/
        redhat/
          MyWorkItemWorkItemHandlerTest.java
          MyWorkItemWorkItemIntegrationTest.java
    resources/
      com/
        redhat/
pom.xml
```

5. **pom.xml** ファイルに対するワークアイテムハンドラーが必要とする Maven の依存関係を追加します。

6. このプロジェクトのデプロイ可能な JAR を作成するには、pom.xml ファイルが配置されている、親プロジェクトのフォルダーで以下のコマンドを実行します。

```
$ mvn clean package
```

複数のファイルが **target/** ディレクトリーに作成されます。このディレクトリーには主に以下の 2 つのファイルが含まれます。

表2.2 ファイルの説明

パラメーター	説明
myworkitems-<version>.jar	Red Hat Process Automation Manager に直接デプロイする時に使用します。
myworkitems-<version>.zip	サービスリポジトリーを使用するデプロイメントに使用します。

第3章 ワークアイテムハンドラープロジェクトのカスタマイズ

ワークアイテムハンドラープロジェクトのコードをカスタマイズできます。ワークアイテムハンドラーが必要とする Java メソッドは **executeWorkItem** と **abortWorkItem** の 2 つです。

表3.1 Java メソッドの説明

Java メソッド	説明
executeWorkItem(WorkItem workItem, WorkItemManager manager)	ワークアイテムハンドラーの実行時にデフォルトで実行されます。
abortWorkItem(WorkItem workItem, WorkItemManager manager)	ワークアイテムが中断された場合に実行されます。

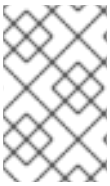
いずれのメソッドでも、**WorkItem** パラメーターには GUI または API 呼び出しでカスタムタスクに入力したパラメーターが含まれており、**WorkItemManager** パラメーターがカスタムタスクの状態を追跡します。

コード構造の例

```
public class MyWorkItemWorkItemHandler extends AbstractLogOrThrowWorkItemHandler {

  public void executeWorkItem(WorkItem workItem, WorkItemManager manager) {
    try {
      RequiredParameterValidator.validate(this.getClass(), workItem);

      // sample parameters
      String sampleParam = (String) workItem.getParameter("SampleParam");
      String sampleParamTwo = (String) workItem.getParameter("SampleParamTwo");

      // complete workitem impl...

      // return results
      String sampleResult = "sample result";
      Map<String, Object> results = new HashMap<String, Object>();
      results.put("SampleResult", sampleResult);
      manager.completeWorkItem(workItem.getId(), results);
    } catch(Throwable cause) {
      handleException(cause);
    }
  }

  @Override
  public void abortWorkItem(WorkItem workItem, WorkItemManager manager) {
    // similar
  }
}
```

表3.2 パラメーターの説明

パラメーター	説明
RequiredParameterValidator.validate(this.getClass(), workItem);	全パラメーターに「required」とマーク付けされていることを確認します。マーク付けされていない場合には、**IllegalArgumentException** が送出されます。
String sampleParam = (String) workItem.getParameter("SampleParam");	**WorkItem** クラスからパラメーターを取得する例。名前は、**WorkItem** など、常に文字列です。この例では、**SampleParam** は常に文字列ですが、これに関連付けられているオブジェクトには多数の型を指定でき、エラーを回避するには、キャストが必要です。
// **complete workitem impl…**	パラメーターの受信時に、カスタムの Java コードが実行されます。
results.put("SampleResult", sampleResult);	カスタムタスクに結果を渡します。この結果は、カスタムタスクのデータ出力エリアに配置されます。
manager.completeWorkItem(workItem.getId(), results);	ワークアイテムハンドラーを完了とマークします。**WorkItemManager** はワークアイテムの状態を制御し、**WorkItem** ID を取得して、取得した結果と正しいカスタムタスクを関連付けます。
abortWorkItem()	カスタムの Java コードを中断します。ワークアイテムが中断されるように設計されていない場合には、空白のままにすることができます。



注記

Red Hat Process Automation Manager に含まれるサポート対象のカスタムタスクには限りがあり、Red Hat Process Automation 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。

第4章 ワークアイテム定義

Red Hat Process Automation Manager では、 Business Central に表示するデータフィールドを特定して、API 呼び出しを受け入れるのにワークアイテム定義 (WID) ファイルが必要です。WID ファイルで、Red Hat Process Automation Manager のユーザーの操作と、ワークアイテムハンドラーに渡されるデータの間をマッピングします。WID ファイルでは、カスタムタスク名、Business Central のパレットに表示されるカテゴリー、カスタムタスクの指定に使用するアイコン、カスタムタスクがマッピングするワークアイテムハンドラーなど、UI の情報も処理します。

Red Hat Process Automation Manager は、次の 2 つの方法で WID ファイルを作成できます。

- ワークアイテムハンドラーをコード化する時に、**@Wid** アノテーションを使用する
- **definitions-example.wid** など、**.wid** のテキストファイルを作成する。

4.1. @WID アノテーション

Maven アーキタイプを使用してワークアイテムハンドラープロジェクトを生成するときに **@Wid** アノテーションは自動的に作成されます。このアノテーションは、手動でも追加できます。

@Wid の例

```
@Wid(widfile="MyWorkItemDefinitions.wid",
    name="MyWorkItemDefinitions",
    displayName="MyWorkItemDefinitions", icon="",
    defaultHandler="mvel: new
    com.redhat.MyWorkItemWorkItemHandler()",
    documentation = "myworkitem/index.html",
    parameters={
      @WidParameter(name="SampleParam", required = true),
      @WidParameter(name="SampleParamTwo", required = true)
    }, results={
      @WidResult(name="SampleResult")
    },
    mavenDepends={
      @WidMavenDepends(group="com.redhat",
      artifact="myworkitem",
      version="7.26.0.Final-example-00004")
    },
    serviceInfo={
        @WidService(
                category = "myworkitem",
                description = "${description}",
                keywords = "",
                action = @WidAction(title = "Sample Title"),
                authinfo = @WidAuth(required = true, params = {"SampleParam", "SampleParamTwo"},
                        paramsdescription = {"SampleParam", "SampleParamTwo"},
                        referencesite = "referenceSiteURL")
        )
    }
)
```

表4.1 @Wid の説明

	説明
@Wid	WID ファイルを自動生成するトップレベルのアノテーション
widfile	Red Hat Process Automation Manager にデプロイするときに、カスタムタスク用に自動的に作成されるファイルの名前。
name	内部で使用されるカスタムタスク名。この名前は、Red Hat Process Automation Manager にデプロイするカスタムタスクで一意でなければなりません。
displayName	カスタムタスクの表示名。この名前は、Business Central のパレットに表示されます。
icon	**src/main/resources/** から、現在のプロジェクトに配置されているアイコンへのパス。このアイコンは、Business Central のパレットに表示されます。アイコンを指定する場合には、16x16 ピクセルの PNG または GIF ファイルでなければなりません。この値を空白のままにして、デフォルトの「Service Task」アイコンを使用することができます。
description	カスタムタスクの説明
defaultHandler	カスタムタスクにリンクされたワークアイテムハンドラーの Java クラス。このエントリーの形式は、**<language> : <class>** です。Red Hat Process Automation Manager は、この属性の言語の値として **mvel** を使用することを推奨しますが、**java** を使用することも可能です。mvel に関する詳細は、MVEL Documentation を参照してください。
documentation	カスタムタスクの説明が含まれる現在のプロジェクトの HTML ファイルへのパス
@WidParameter	**@Wid** の子アノテーション。Business Central GUI で生成される値または、カスタムタスクのデータ入力として API 呼び出しが必要とする値を指定します。複数の値を指定できます。 **name** - パラメーター名 **注記** この名前は、REST や SOAP などの転送メソッドの API 呼び出しで使用される可能性があるため、スペースや特殊文字を含めないでください。 **required** - パラメーターが、実行するカスタムタスクに必要かどうかを指定するブール値

	説明
@WidResult	**@Wid** の子アノテーション。Business Central GUI で生成される値または、カスタムタスクのデータ出力として API 呼び出しが必要とする値を指定します。複数の結果を指定できます。 **name** - 結果の名前  **注記** この名前は、REST や SOAP などの転送メソッドの API 呼び出しで使用される可能性があるため、スペースや特殊文字を含めないでください。 **@WidMavenDepends** - **@Wid** の子アノテーション。ワークアイテムハンドラーが正しく機能するのに必要な Maven の依存関係を指定します。複数の依存関係を指定できます。 **group** - 依存関係の Maven グループ ID **artifact** - 依存関係の Maven アーキタイプ ID **version** - 依存関係の Maven バージョン番号
@WidService	**@Wid** の子アノテーション。サービスリポジトリーで生成される値を指定します。 **category** - ハンドラーが配置される UI パレットのカテゴリー。この値は、**@Wild** の **category** フィールドの値と一致する必要があります。 **description** - サービスリポジトリーに表示されるハンドラーの説明 **keywords** - ハンドラーに適用するキーワードのコンマ区切りの一覧。注: 現在、Business Central サービスリポジトリーでは使用されていません。 **action** - ハンドラーの目的を記述する **@WidAction** オブジェクト。**title** と **description** のフィールドが含まれます。 **authinfo** - 認証要件を定義する **@WidAuth** オブジェクト。任意。**required**、**params**、**paramsdescription**、**referencesite** のフィールドが含まれます。
@WidAction	ハンドラーの目的を記述する **@WidService** のオブジェクト **title** - ハンドラーアクションのタイトル **description** - ハンドラーアクションの説明

	説明
@WidAuth	ハンドラーで必要な認証を定義する **@WidService** オブジェクト **required** - 認証が必要がどうかを判断するブール値 **params** - 必要な認証パラメーターが含まれるアレイ **paramsdescription** - 各認証パラメーターの説明が含まれるアレイ **referencesite** - ハンドラーのドキュメントの場所を指す URL。注: 現在 Business Central リポジトリーでは使用されていません。

4.2. テキストファイル

グローバルな **WorkDefinitions** WID テキストファイルは、ビジネスプロセスが追加されると、新規プロジェクトで自動的に生成されます。WID テキストファイルは、JSON 形式に似ていますが、完全に有効な JSON ファイルではありません。このファイルは、Business Central で開くことができます。既存のプロジェクトから **Add Asset** > **Work item definitions** の順に選択して、追加の WID ファイルを作成できます。

テキストファイルの例

```
[
  [
    "name" : "MyWorkItemDefinitions",
    "displayName" : "MyWorkItemDefinitions",
    "category" : "",
    "description" : "",
    "defaultHandler" : "mvel: new com.redhat.MyWorkItemWorkItemHandler()",
    "documentation" : "myworkitem/index.html",
    "parameters" : [
      "SampleParam" : new StringDataType(),
      "SampleParamTwo" : new StringDataType()
    ],
    "results" : [
      "SampleResult" : new StringDataType()
    ],
    "mavenDependencies" : [
      "com.redhat:myworkitem:7.26.0.Final-example-00004"
    ],
    "icon" : ""
  ]
]
```

ファイルは、JSON のような構造を使用してプレーンテキストファイルとして構成されます。ファイル名の拡張子は、**.wid** です。

表4.2 テキストファイルの説明

	説明

	説明
name	内部で使用されるカスタムタスク名。この名前は、Red Hat Process Automation Manager にデプロイするカスタムタスクで一意でなければなりません。
displayName	カスタムタスクの表示名。この名前は、Business Central のパレットに表示されます。
icon	**src/main/resources/** から、現在のプロジェクトに配置されているアイコンへのパス。このアイコンは、Business Central のパレットに表示されます。アイコンを指定する場合には、16x16 ピクセルの PNG または GIF ファイルでなければなりません。この値を空白のままにして、デフォルトの「Service Task」アイコンを使用することができます。
category	このカスタムタスクが表示される Business Central パレット内のカテゴリー名
description	カスタムタスクの説明
defaultHandler	カスタムタスクにリンクされたワークアイテムハンドラーの Java クラス。このエントリーの形式は、**<language> : <class>** です。Red Hat Process Automation Manager は、この属性の言語の値として **mvel** を使用することを推奨しますが、**java** を使用することも可能です。mvel に関する詳細は、MVEL Documentation を参照してください。
documentation	カスタムタスクの説明が含まれる現在のプロジェクトの HTML ファイルへのパス
parameters	Business Central GUI で生成される値または、カスタムタスクのデータ入力として API 呼び出しが必要とする値を指定します。パラメーターは、**<key> : <DataType>** 形式を使用します。許容されるデータ型は、**StringDataType()**、**IntegerDataType()** および **ObjectDataType()** で、複数のパラメーターを指定できます。
results	Business Central GUI で生成される値または、カスタムタスクのデータ出力として API 呼び出しが必要とする値を指定します。結果は、**<key> : <DataType>** 形式を使用します。許容されるデータ型は、**StringDataType()**、**IntegerDataType()** および **ObjectDataType()** で、複数の結果を指定できます。
mavenDependencies	任意: ワークアイテムハンドラーを正しく機能させるのに必要な Maven 依存関係を指定します。依存関係は、ワークアイテムハンドラーの **pom.xml** ファイルで指定することもできます。依存関係は、**<group>:<artifact>:<version>** 形式で指定します。複数の依存関係を指定できます。

Red Hat Process Automation Manager はデフォルトで、2 つの場所で ***.wid** ファイルの場所を特定しようと試みます。

- Business Central 内にあるプロジェクトのトップレベルの **global/** ディレクトリー。これは、プロジェクトがビジネスプロセスアセットに初めて追加されると自動的に作成される、デフォルトの **WorkDefinitions.wid** ファイルです。

- Business Central 内にあるプロジェクトの **src/main/resources/** ディレクトリー。これは、Business Central で作成した WD ファイルの配置場所です。WID ファイルは、Java パッケージレベルで作成できるので、**<default>** のパッケージ場所で作成される WID ファイルは、直接 **src/main/resources/** 内に作成され、**com.redhat** のパッケージ場所で作成される WID ファイルは **src/main/resources/com/redhat/** に作成されます。

警告

Red Hat Process Automation Manager では、**defaultHandler** タグの値が実行可能化どうか、有効な Java クラス化どうかは検証されません。このタグに無効なクラスや間違ったクラスを指定するとエラーが返されます。

第5章 カスタムタスクのデプロイ

ワークアイテムハンドラーは、Red Hat Process Automation Manager 外にカスタムコードとして作成されます。カスタムタスクでこのコードを使用するには、このコードはサーバーにデプロイする必要があります。ワークアイテムハンドラープロジェクトは、Maven リポジトリーに配置可能な Java JAR ファイルでなければなりません。

Red Hat Process Automation Manager では、以下の 3 つの方法でカスタムタスクをデプロイできます。

- Business Central サービスタスクのリポジトリー内。詳細は、*1章Business Central でのサービスタスクの管理*を参照してください。
- Business Central 内。レガシーと現在のエディター両方を使用して、ワークアイテムハンドラー JAR を Business Central Maven リポジトリーにアーティファクトとしてアップロードできます。
- Business Central を使用せずに、JAR ファイルを Maven リポジトリーに手動でコピーできます。

5.1. BUSINESS CENTRAL のサービスタスクリポジトリーの使用

Business Central サービスタスクリポジトリーでサービスタスクの有効化、無効化、デプロイができます。詳細は、*1章Business Central でのサービスタスクの管理*を参照してください。

5.2. JAR アーティファクトの BUSINESS CENTRAL へのアップロード

レガシーおよび現在のエディターを使用して、ワークアイテムハンドラーの JAR を Business Central Maven リポジトリーにアーティファクトとして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** アイコンを選択して、**Artifacts** を選びます。
2. **アップロード** をクリックします。
3. **Artifact Upload** ウインドウで、**Choose File** アイコンをクリックします。
4. ワークアイテムハンドラーの JAR の場所に移動し、ファイルを選択して **Open** をクリックします。
5. ポップアップダイアログで **Upload** アイコンをクリックします。
   アーティファクトがアップロードされ、**Artifacts** ページで表示して参照できるようになります。

5.3. BUSINESS CENTRAL MAVEN リポジトリーへのワークアイテム定義の手動コピー

Business Central は、Maven リポジトリーフォルダーを自動的に作成するか、再利用します。デフォルトでは、Red Hat JBoss EAP を起動したユーザーの場所をもとに、場所が決定されます。たとえば、デフォルトのパスは、**<startup location>/repositories/kie/global** です。標準の Maven リポジトリーフォルダーのレイアウト (**<groupId>/<artifactId>/<versionId>/**) を複製し、この場所にワークアイテムハンドラーの JAR ファイルをコピーできます。以下に例を示します。

```
<startup location>/repositories/kie/global/com/redhat/myworkitem/1.0.0-SNAPSHOT/myworkitems-1.0.0-SNAPSHOT.jar
```

この形式でコピーしたアーティファクトは、サーバーを再起動しなくてもRed Hat Process Automation Manager で利用できます。Business Central の **Artifacts** ページでアーティファクトを表示するには、**Refresh** をクリックする必要があります。

第6章 カスタムタスクの登録

Red Hat Process Automation Manager は、カスタムタスクのワークアイテムと、ワークアイテムハンドラーが実行するコードと関連付ける方法を知っておく必要があります。ワークアイテム定義ファイルは、名前と Java クラスで、カスタムタスクと、ワークアイテムハンドラーをリンクします。ワークアイテムハンドラーの Java クラスは、Red Hat Process Automation Manager で利用できるように登録しておく必要があります。

注記

サービスリポジトリーには、各種システムとプロセスを統合できるように、ドメイン固有のサービスが含まれています。サービスリポジトリーを使用する場合には、インポートプロセスでカスタムタスクが登録されるので、カスタムタスクの登録は、必要ありません。

Red Hat Process Automation Manager では、ビジネスプロセスが最低 1 つ含まれるプロジェクトには、デフォルトで WID ファイルが作成されます。ワークアイテムハンドラーの登録時に WID ファイルを作成したり、デフォルトの WID ファイルを編集したりできます。WID ファイルの場所や形式に関する情報は、*4章 ワークアイテム定義*を参照してください。

サービスリポジトリーを使用しないデプロイメントの場合には、ワークアイテムハンドラーは 2 種類の方法で登録できます。

- デプロイメント記述子を使用した登録
- Spring コンポーネント登録を使用した登録

6.1. BUSINESS CENTRAL でデプロイメント記述子を使用したカスタムタスクの登録

Business Central でデプロイメント記述子を使用してワークアイテムハンドラーで、カスタムタスクのワークアイテムを登録できます。

手順

1. Business Central で、**Menu → Design → Projects** に移動して、プロジェクト名を選択します。
2. プロジェクトペインで **Settings → Deployments → Work Item Handlers** の順に選択します。
3. **Add Work Item Handler** をクリックします。
4. **Name** フィールドで、カスタムタスクの表示名を入力します。
5. **Resolver** リストから **MVEL**、**Reflection** または **Spring** を選択します。
6. **Value** フィールドに、リゾルバ-タイプをもとに値を入力します。
   - MVEL の場合には、**new <full Java package>.<Java work item handler class name>()** の形式を使用します。
     例: **new com.redhat.MyWorkItemWorkItemHandler()**
   - Reflection の場合には、**<full Java package>.<Java work item handler class name>** の形式を使用します。
     例: **com.redhat.MyWorkItemWorkItemHandler**

- Spring の場合には **<Spring bean identifier>** の形式を使用します。
  例: **workItemSpringBean**

7. **Save** をクリックして変更を保存します。

6.2. BUSINESS CENTRAL 外でのデプロイメント記述子を使用したカスタムタスクの登録

Business Central 外でデプロイメント記述子を使用して、ワークアイテムハンドラーでカスタムタスクのワークアイテムを登録できます。

手順

1. **src/main/resources/META-INF/kie-deployment-descriptor.xml** のファイルを開きます。
2. **<work-item-handlers>** のリゾルバータイプをもとに、以下の内容を追加します。
   - MVEL の場合には、以下を追加します。

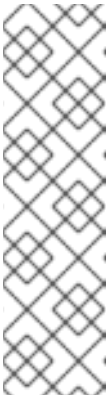
```
<work-item-handler>
  <resolver>mvel</resolver>
  <identifier>new com.redhat.MyWorkItemWorkItemHandler()</identifier>
  <parameters/>
  <name>MyWorkItem</name>
</work-item-handler>
```

   - Reflections の場合には、以下を追加します。

```
<work-item-handler>
  <resolver>reflection</resolver>
  <identifier>com.redhat.MyWorkItemWorkItemHandler</identifier>
  <parameters/>
  <name>MyWorkItem</name>
</work-item-handler>
```

   - Springの場合には、以下を追加し、識別子が Spring Bean の識別子であることを確認します。

```
<work-item-handler>
  <resolver>spring</resolver>
  <identifier>beanIdentifier</identifier>
  <parameters/>
  <name>MyWorkItem</name>
</work-item-handler>
```



注記

Spring を使用して、Spring bean を検出して設定する場合には、**org.springframework.stereotype.Component** クラスのアノテーションを使用して、ワークアイテムハンドラーを自動的に登録できます。

ワークアイテムハンドラー内で、ワークアイテムハンドラークラスの宣言の前に **@Component("<Name>")** のアノテーションを追加します。例:
@Component("MyWorkItem") public class MyWorkItemWorkItemHandler extends AbstractLogOrThrowWorkItemHandler {

第7章 カスタムタスクの配置

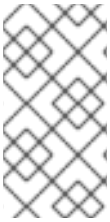
Red Hat Process Automation Manager でカスタムタスクを登録すると、カスタムタスクはプロセスデザイナーのパレットに表示されます。カスタムタスクに名前がつけられ、該当する WID ファイルのエントリーに基づいて分類されます。

前提条件

- カスタムタスクが Red Hat Process Automation Manager に登録されます。カスタムタスクの登録に関する情報は、6章 *カスタムタスクの登録*を参照してください。
- カスタムタスクに名前がつけられ、該当する WID ファイルのエントリーに基づいて分類されます。WID ファイルの場所とフォーマットに関する詳細は、4章 *ワークアイテム定義*を参照してください。

手順

1. Business Central で、**Menu → Design → Projects** に移動して、プロジェクトをクリックします。
2. カスタムタスクを追加するビジネスプロセスを選択します。
3. パレットからカスタムタスクを選択し、BPMN2 ダイアグラムにドラッグします。
4. オプション: カスタムタスク属性を変更します。たとえば、該当する WID ファイルから、データの出入力を変更します。



注記

WID ファイルがプロジェクトに表示されず、プロジェクトの **Others** カテゴリーに **Work Item Definition** オブジェクトが表示されない場合には、カスタムタスクを登録する必要があります。カスタムタスクの登録に関する詳細は、6章 *カスタムタスクの登録*を参照してください。

付録A バージョン情報

本書の最終更新日: 2020 年 9 月 8 日 (木)