Red Hat Process Automation Manager 7.8

ケース管理の設計およびビルド

Last Updated: 2020-09-18

Red Hat Process Automation Manager 7.8 ケース管理の設計およびビルド

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 のケース管理の概念について説明します。

目次

前書き

開発者は、Business Central を使用して、ケース管理できるように、Red Hat Process Automation Manager アセットを設定できます。

ケース管理はビジネスプロセス管理 (BPM) とは異なり、一連の目的達成の手順に焦点を当てるのではなく、ケース全体で扱う実際のデータに、より重点を置きます。ケースデータは、自動化されたケース処理の情報の中で最も重要な要素ですが、ビジネスの状況や意思決定はケース作業担当者の管理下に置かれます。

Red Hat Process Automation Manager では、Business Central に **IT_Orders** サンプルが含まれます。本書では、ケース管理の概念を説明して例を提示する場合にこのサンプルを参照します。

『**ケース管理の使用ガイド**』のチュートリアルでは、Business Central で IT_Orders プロジェクトを作成してテストする方法を説明します。本ガイドでコンセプトを確認した後に、チュートリアルの説明に沿って、ご自身のケースプロジェクトを正常に作成してデプロイし、テストできるように、進めてください。

前提条件

- Red Hat JBoss Enterprise Application Platform 7.3 がインストールされている。Red Hat JBoss Enterprise Application Platform 7.3 のインストールに関する情報は『**Red Hat JBoss Enterprise Application Platform 7.3 インストールガイド**』を参照してください。
- Red Hat Process Automation Manager をインストールしている。Red Hat Process Automation Manager のインストールに関する情報細は『**Red Hat Process Automation Manager インストールの計画**』を参照してください。
- Red Hat Process Automation Manager が稼働し、**user** ロールで Business Central にログインできる。ユーザーおよびパーミッションに関する情報は『**Red Hat Process Automation Manager インストールの計画**』を参照してください。
- Showcase アプリケーションがデプロイされている。Showcase アプリケーションをインストールしてログインする方法は、『**ケース管理への Showcase アプリケーションの使用**』

第1章 ケース管理

ケース管理は、Business Process Management (BPM) の拡張機能で、適用可能なビジネスプロセスを管理します。

BPM は、反復可能で共通のパターンを持つタスクの自動化に使用する管理プラクティスで、プロセスを完全化して最適化を図ることに焦点を当てます。ビジネスプロセスは通常、ビジネスの目標へのパスを明確に定義し、モデル化されています。これには、通常量産原理をもとに、多くを予測ができる必要があります。ただし、実際のアプリケーションの多くは、最初から最後まで完全に記述できません (考えられるパス、偏差、例外など)。特定のケースでプロセス指向のアプローチを使用すると、複雑なソリューションとなり、管理が困難になります。

推測可能なルーチンタスクを対象とする BPM の効率指向アプローチとは対照的に、ケース管理は反復なしで推測不可能なプロセスに対して問題解決を提供し、プロセスが事前に予測不可能な場合など、一回限りの状況を管理します。ケースの定義は通常、疎結合プロセスの断片で構成されており、このような断片を直接または間接的に結合して、ランタイム時に発生する変化に対応し、プロセスを動的に管理しつつ、特定のマイルストーンや最終的なビジネス目標に到達できるようになっています。

Red Hat Process Automation Manager のケース管理には、以下のコアプロセスエンジン機能が含まれます。

- ケースファイルのインスタンス
- ケースごとのランタイム戦略
- ケースコメント
- マイルストーン
- ステージ
- アドホックフラグメント
- 動的タスクおよびプロセス
- ケース識別子 (相関キー)
- ケースのライフサイクル (閉じる、再開、キャンセル、破棄)

ケース定義は常にアドホックのプロセス定義で、明示的な開始ノードは必要ありません。ケース定義は、ビジネスユースケースの主なエントリーポイントです。

プロセス定義は、ケースでサポートされる構成概念として導入され、ケース定義に指定された通り、または必要に応じて追加処理に動的に取り込むために呼び出すことができます。ケース定義は、以下の新しいオブジェクトを定義します。

- アクティビティー (必須)
- ケースファイル (必須)
- マイルストーン
- ロール
- ステージ

第2章 CASE MANAGEMENT MODEL AND NOTATION (CMMN)

Case Management Model and Notation (CMMN) ファイルのコンテンツをインポートして表示し、変更するには、Business Central を使用します。プロジェクトを作成するには、ケース管理モデルをインポートしてから、そのケース管理モデルをアセットリストから選択して標準の XML エディターで表示または変更できます。

以下の CMMN コンストラクトが現在利用できます。

- Tasks (human task、process task、decision task、case task)
- Discretionary tasks (上記と同じ)
- ステージ
- マイルストーン
- Case file items
- Sentries (entry および exit)

次のタスクはサポート対象外です。

- Required
- Repeat
- Manual activation

個別タスクの Sentry は、entry 基準に限定されていますが、entry と exit の基準は stages と milestones でサポートされています。Decision task はデフォルトで DMN decision にマッピングされます。イベントリスナーはサポートされていません。

Red Hat Process Automation Manager には CMMN のモデリング機能は含まれておらず、モデルの実行のみに焦点を当てています。

第3章 ケースファイル

ケースインスタンスは、ケース定義を含む1つのインスタンスで、ビジネスコンテキストをカプセル化します。ケースインスタンスデータはすべてケースファイルに保存され、特定のケースインスタンスに参加する可能性のあるすべてのプロセスインスタンスからアクセスできます。各ケースインスタンスと、ケースインスタンスのケースファイルは、他のケースから完全に分離されています。ケースインスタンスの参加者のみがケースファイルにアクセスできます。

ケースファイルは、ケースインスタンス全体のデータリポジトリーとして、ケースの管理に使用します。このファイルには、全ロール、データオブジェクト、データマップなどのデータが含まれます。ケースを完了してから、同じケースファイルを添付して、後日再度開くことができます。ケースインスタンスはいつでも終了でき、特別な解決策を提示して完了する必要はありません。

ケースファイルには、埋め込み型のドキュメント、参考資料、PDF 添付ファイル、Web リンクなどのオプションも含めることができます。

3.1. ケース ID 接頭辞の設定

caseId パラメーターは、ケースインスタンス ID の文字列値です。Red Hat Process Automation Manager デザイナーで **Case ID prefix** を設定して、異なるケースタイプを識別できます。

以下の手順では **IT_Orders** サンプルプロジェクトを使用し、特定のビジネスニーズに合わせて、一意のケース ID 接頭辞を作成します。

前提条件

- Business Central で **IT_Orders** サンプルプロジェクトを開いている。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。既存のプロジェクトがある場合には、**MySpace** のデフォルトのスペースをクリックして、**Add Project** プルダウンメニューから **Try Samples** を選択して、サンプルにアクセスできます。既存のプロジェクトがない場合には、**Try samples** をクリックします。
2. **IT_Orders** を選択し、**OK** をクリックします。
3. **Assets** ウィンドウで **orderhardware** ビジネスプロセスをクリックしてデザイナーを開きます。
4. キャンバスの空きスペースをクリックし、右上隅の **Properties** アイコンをクリックします。
5. 下方向にスクロールして、**Case Management** を展開します。
6. **ID-XXXXXXXXXX** 形式で **Case ID Prefix** を入力します。**XXXXXXXXXX** は、生成された番号で、ケースインスタンスの一意の ID を指定します。接頭辞が指定されていない場合には、デフォルトのセットうじとして、以下の ID に **CASE** が付けられます。

CASE-0000000001

CASE-0000000002

CASE-0000000003

接頭辞は、任意で指定できます。たとえば、**IT** の接頭辞を指定する場合には、次の識別子が生成されます。

IT-0000000001

IT-0000000002

IT-0000000003

図3.1 ケース ID 接頭辞のフィールド

3.2. ケース ID 式の設定

以下の手順は、**IT_Orders** サンプルプロジェクトを使用して、設定したメタデータ属性キーで **caseId** を生成する式をカスタマイズする方法を説明します。

前提条件

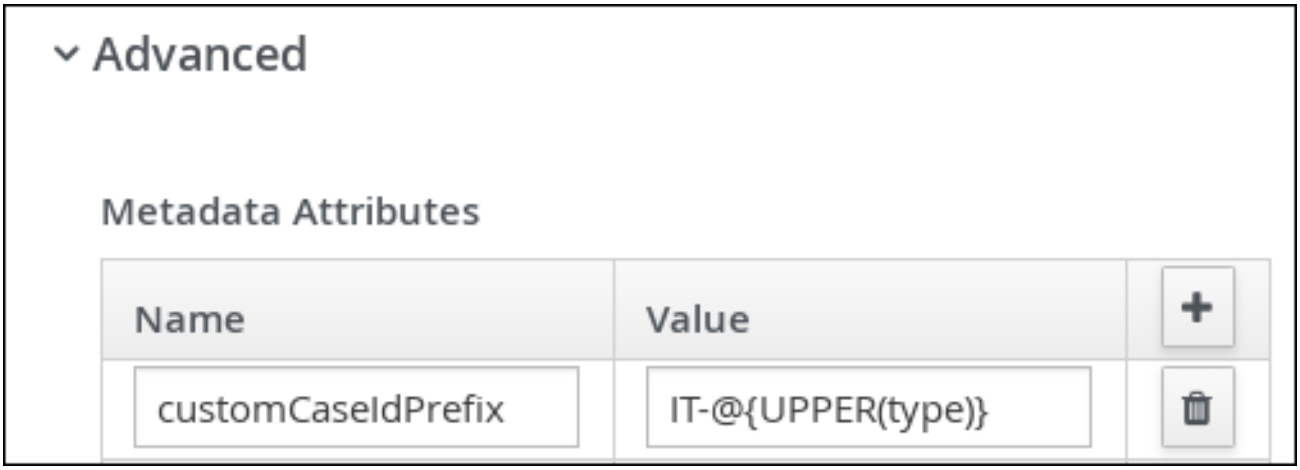
- Business Central で **IT_Orders** サンプルプロジェクトを開いている。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。既存のプロジェクトがある場合には、**MySpace** のデフォルトのスペースをクリックして、**Add Project** プルダウンメニューから **Try Samples** を選択して、サンプルにアクセスできます。既存のプロジェクトがない場合には、**Try samples** をクリックします。
2. **IT_Orders** を選択し、**OK** をクリックします。
3. **Assets** ウィンドウで **orderhardware** ビジネスプロセスをクリックしてデザイナーを開きます。
4. キャンバスの空きスペースをクリックし、右上隅の **Properties** アイコンをクリックします。
5. **Advanced** メニューを展開して **Metadata Attributes** フィールドにアクセスします。
6. **customCaseIdPrefix** メタデータ属性の以下の関数を 1 つ指定します。
   - **LPAD** - 左パディング
   - **RPAD** - 右パディング

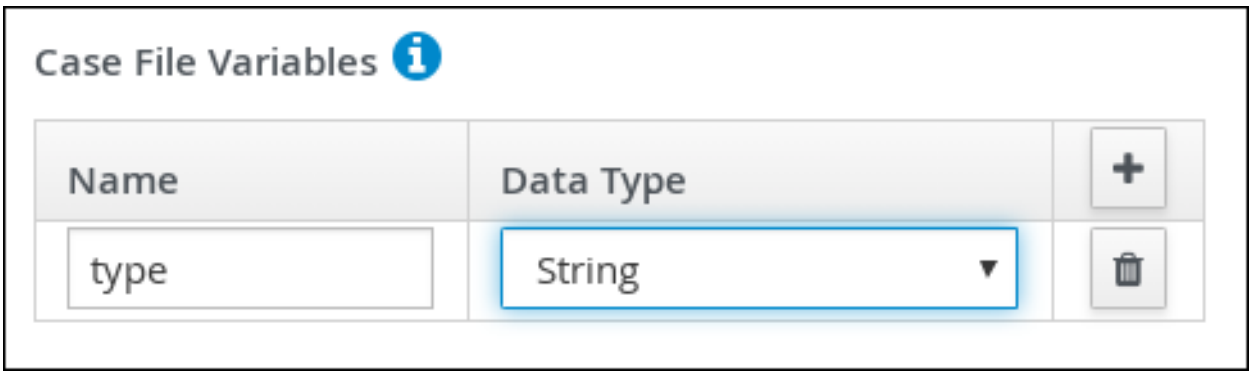
- TRUNCATE - 省略
- UPPER - 大文字

図3.2 customCaseIdPrefix メタデータ属性の UPPER 関数



この例では、**type** は、Case File Variables フィールドに設定された変数です。この変数は、ランタイム時に **type1** の値に変更できます。**UPPER** は事前に組み込まれた関数で変数を大文字に変換します。ただし、"`IT`-" は静的な接頭辞となっています。そのため、動的なケース ID は **IT-TYPE1-0000000001**、**IT-TYPE1-0000000002** および **IT-TYPE1-0000000003** のようになります。

図3.3 ケースファイル変数



customCaseIdPrefixIsSequence ケースメタデータ属性が **false** (デフォルト値 **true**) に設定されている場合には、ケースインスタンスではシーケンスが作成されず、**caseIdPrefix** 式がケース ID になりますたとえば、社会保障番号をもとにケース ID を生成した場合には、特定のシーケンスまたはインスタンス ID は必要ありません。

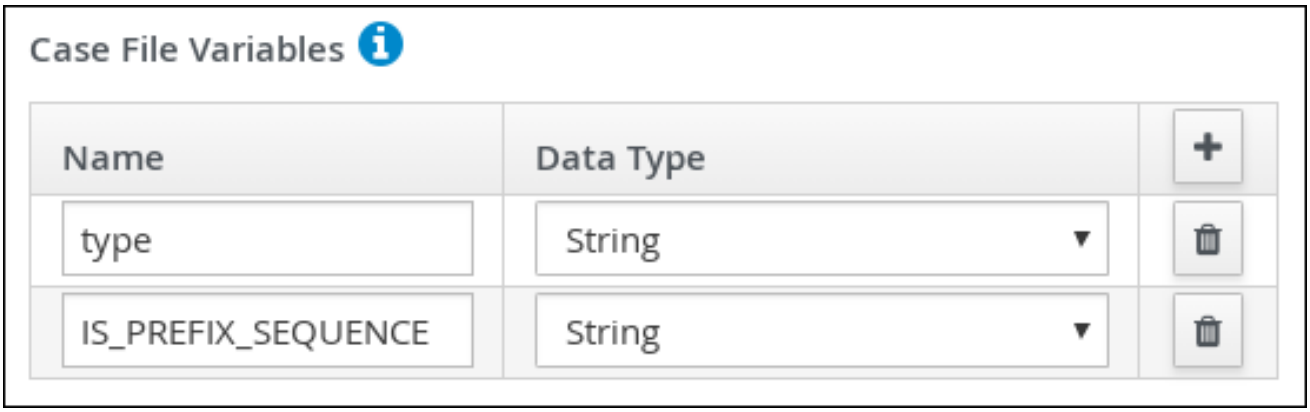
customCaseIdPrefixIsSequence メタデータ属性はオプションで追加し、**false** (デフォルト値 **true**) に設定してケース ID の番号順を無効にします。カスタムのケース ID に使用する式にケースファイル変数が含まれており、一般的なシーケンス値ではなく、一意のビジネス ID で表現されている場合に便利です。たとえば、社会保障番号をもとにケース ID を生成する場合に、特定の順番やインスタンス ID は必要ありません。以下の例では、**SOCIAL_SECURITY_NUMBER** は、ケースファイル変数として宣言された変数でもあります。

図3.4 customCaseIdPrefixIsSequence metadata attribute

IS_PREFIX_SEQUENCE ケースファイル変数はランタイム時にオプションで追加され、ケース ID をシーケンスで生成するのを無効または有効にします。たとえば、個人医療保険の補償範囲にシーケンスの接尾辞を作成する必要はありません。マルチホーム保険契約の場合は、企業は **IS_PREFIX_SEQUENCE** のケース変数を **true** に設定し、家族の一人ずつ、順番に番号を付与します。

customCaseIdPrefixIsSequence メタデータ属性を静的に **false** として使用するか、または **IS_PREFIX_SEQUENCE** ケースファイル変数を使用して、ランタイム時に値を **false** に設定すると、どちらも同じ結果が得られます。

図3.5 IS_PREFIX_SEQUENCE ケース変数



第4章 サブケース

サブケースは、他のケースを含めて複雑なケースを柔軟に構成できます。つまり、大規模で複雑なケースを複数の抽象階層、さらには複数のケースプロジェクトに分割できるようになります。これは、プロセスを複数のサブプロセスに分割するのと類似します。

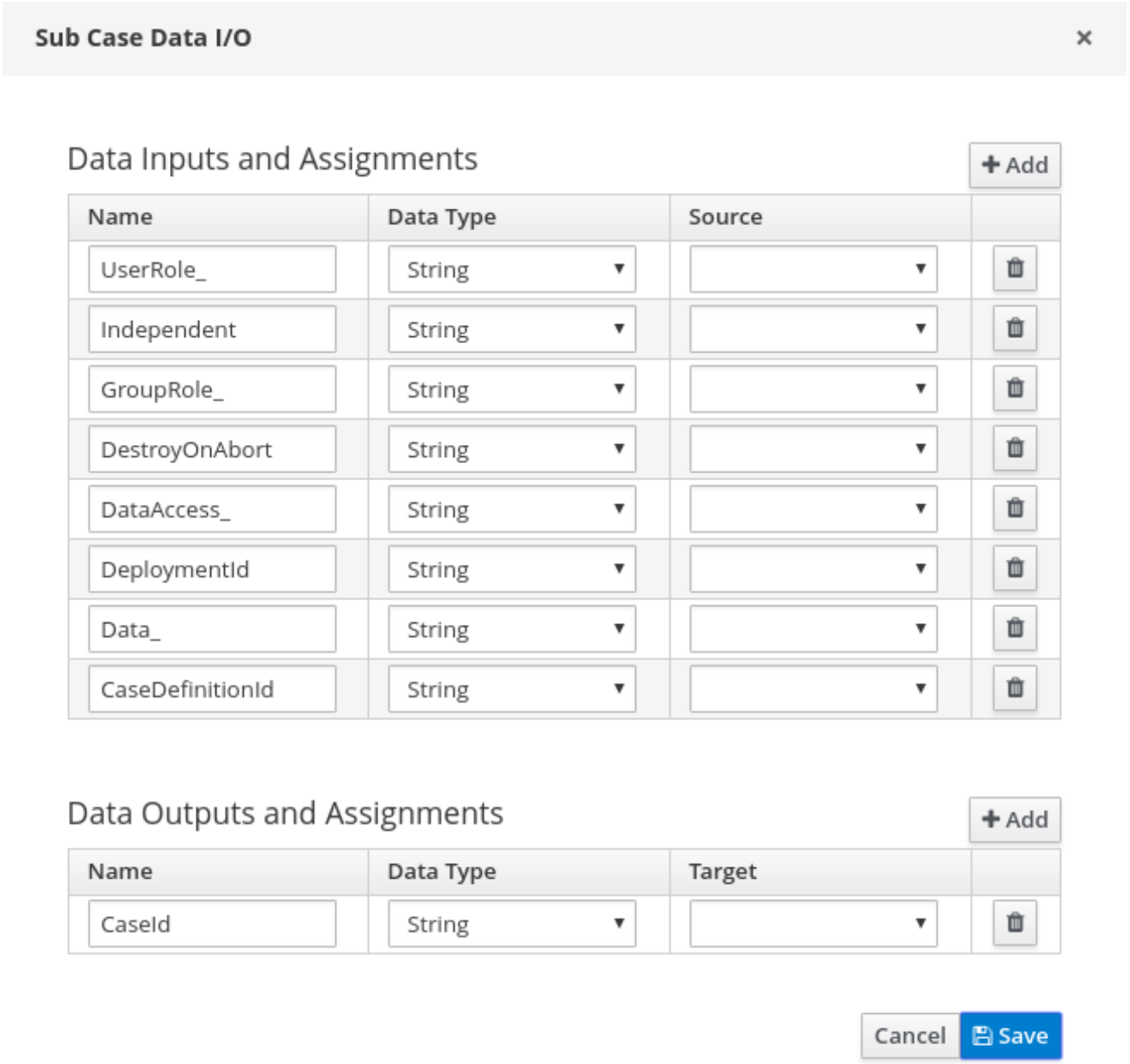
サブケースは、別のケースインスタンス、または通常のプロセスインスタンスから呼び出された別のケース定義です。これには、通常のケースインスタンスの機能がすべて含まれます。

- 専用のケースファイルがある。
- 別のケースインスタンスから分離している。
- ケースロールのセットを所有する。
- 独自のケース接頭辞がある。

ケースの定義にサブクラスを追加するには、プロセスデザイナーを使用できます。サブクラスは、ケースプロジェクト内のケースのことで、プロセスに含まれるサブプロセスに似ています。サブクラスは、通常のビジネスプロセスに追加することもできます。こうすることで、プロセスインスタンス内からケースを起動できます。

ケースへのサブケースの追加に関する詳細は、『**ケース管理の使用ガイド**』を参照してください。

Sub Case Data I/O ウィンドウでは、サブケースを設定して起動できるように、以下の入力パラメーターをサポートします。



Independent

ケースインスタンスが独立しているかどうかを、プロセスデザイナーに通知する任意のインジケーター。独立している場合は、メインのケースインスタンスは、完了するまで待機しません。デフォルトでは、このプロパティーの値は、**false** です。

GroupRole_XXX

ケースロールマッピングの任意のグループ。このケースインスタンスに所属するロール名はここで参照できるので、メインのケースの参加者を、サブケースの参加者にマッピングできます。つまり、メインケースに割り当てられたグループは、サブケースに自動的に割り当てられます。**XXX** はロール名に、プロパティーの値は、グループのロール割り当ての値に置き換えてください。

DataAccess_XXX

任意のデータアクセス制限。**XXX** は、データ項目の名前に、プロパティーの値はアクセス制限に置き換えてください。

DestroyOnAbort

サブケースのアクティビティーが中断された場合に、サブケースを取り消して、破棄するかどうかをプロセスエンジンに指示する任意のインジケーター。デフォルト値は、**true** です。

UserRole_XXX

ケースロールマッピングの任意のユーザー。ここで、ケースインスタンスのロール名を参照できる

ので、メインのケースの所有者を、サブケースの所有者にマッピングできます。つまり、メインケースに割り当てられたユーザーは、サブケースに自動的に割り当てられます。**XXX** はロール名に、プロパティーの値は、ユーザーのロール割り当ての値に置き換えてください。

Data_XXX

ケースインスタンスまたはビジネスプロセスからサブケースへの任意のデータマッピング。**XXX** は、対象のサブクラスに含まれるデータ名に置き換えます。このパラメーターは、必要に応じていくつでも指定できます。

DeploymentId

任意のデプロイメント ID (または KIE Server の場合はコンテキスト ID)。この ID で、対象のケース定義がどこにあるかを指定します。

CaseDefinitionId

開始するのに必要なケース定義 ID

CaseId

起動後のサブケースのケースインスタンスID

第5章 アドホックおよび動的タスク

エンドツーエンドプロセスに厳密に従う代わりに、ケース管理を使用して、アドホックにタスクを実行できます。タスクは、ランタイム時にケースに動的に追加することもできます。

アドホックタスクは、ケースモデリングフェーズに定義されます。**Adhoc autostart** として設定されていないアドホックタスクはオプションであるため、ケース実行時に使用されない場合もあります。したがって、そのタスクは、1つのイベントまたは1つの Java API からトリガーするる必要があります。

動的タスクはケース実行時に定義され、ケース定義モデルには表示されません。動的タスクは、ケース時に発生する特定の要求に対応します。Red Hat Process Automation Manager Showcase のデモにあるように、タスクはケースに追加され、ケースアプリケーションを使用していつでも作業できます。動的タスクは、Java および Remote API コールから追加することもできます。

動的タスクは、ユーザーまたはサービスアクティビティーのいずれかで、アドホックタスクは任意のタイプのタスクに指定できます。タスクタイプに関する詳細は、『Business Central でのビジネスプロセスの設計』の「プロセスデザイナーの BPMN2 タスク」を参照してください。

動的プロセスは、ケースプロジェクトから再利用できるサブプロセスです。

内向き接続がないアドホックノードは、ノードの **Adhoc autostart** プロパティーで設定でき、ケースインスタンスの起動時に自動的に開始します。

アドホックタスクは、ケース定義に設定される任意のタスクです。このタスクはアドホックであるため、通常はシグナルイベントまたは Java API コールによって発生します。

第6章 API を使用してケースへの動的タスクおよびプロセスの追加

ランタイム時に動的タスクとプロセスをケースに追加して、ケースのライフサイクル時に発生する可能性がある予定外の変更を処理できます。動的アクティビティーはケース定義に定義されているわけでないため、定義したアドホックタスクまたはプロセスが可能な方法をシグナル化することはできません。

以下の動的アクティビティーをケースに追加できます。

- ユーザータスク
- サービスタスク (作業項目として実装されるすべてのタイプ)
- 再利用可能なサブプロセス

動的ユーザーおよびサービスタスクがケースインスタンスに追加され、直ちに実行されます。動的タスクの特性に従って、開始して完了を待つ (ユーザータスクの) か、実行後に直接完了 (サービスタスク) します。動的サブプロセスの場合、プロセスエンジンは、この動的プロセスに対するプロセス定義を含む KJAR を要求して、ID でプロセスを探して実行します。このサブプロセスはケースに属し、ケースファイルのすべてのデータにアクセスします。

Swagger REST API アプリケーションを使用して、動的タスクおよびサブプロセスを作成します。

前提条件

- Business Central にログインしており、Showcase アプリケーションを使用してケースインスタンスを起動している。Showcase の使用に関する情報は、『**ケース管理への Showcase アプリケーションの使用**』を参照してください。

手順

1. Web ブラウザーで、以下の URL を開きます。
   /http://localhost:8080/kie-server/docs
2. **Case instances :: Case Management** で利用可能なエンドポイントの一覧を開きます。
3. **POST** メソッドのエンドポイントを探し、動的アクティビティーを作成します。
   POST /server/containers/{id}/cases/instances/{caseId}/tasks

   ケースインスタンスに動的タスク (ペイロードに合わせてユーザーまたはサービスを選択) を追加します。

   POST /server/containers/{id}/cases/instances/{caseId}/stages/{caseStageId}/tasks

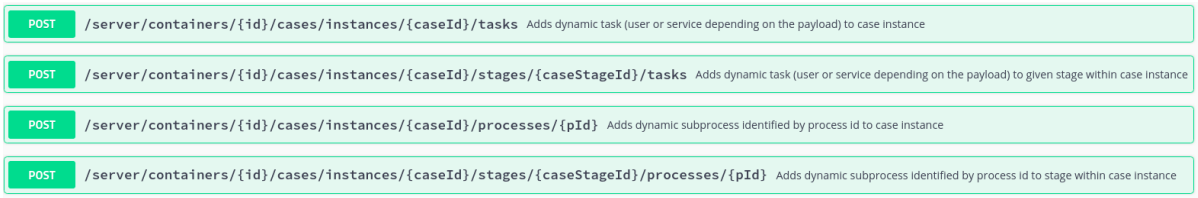
   ケースインスタンス内の特定のステージに動的タスク (ペイロードに合わせてユーザーまたはサービスを選択) を追加します。

   POST /server/containers/{id}/cases/instances/{caseId}/processes/{pId}

   プロセス ID で識別される動的サブプロセスをケースインスタンスに追加します。

   POST
   /server/containers/{id}/cases/instances/{caseId}/stages/{caseStageId}/processes/{pId}

   ケースインスタンス内のステージに、プロセス ID で識別される動的サブプロセスを追加します。



4. ドキュメントを開くには、動的タスクまたはプロセスの作成に必要な REST エンドポイントをクリックします。
5. **Try it out** をクリックして、動的アクティビティーの作成に必要なパラメーターとボディーを入力します。
6. **Execute** をクリックして、REST API を使用する動的タスクまたはサブプロセスを作成します。

6.1. REST API を使用する動的ユーザータスクの作成

ケースの実行時には、REST API を使用して、動的ユーザータスクを作成できます。動的ユーザータスクを作成するには、次の情報を指定する必要があります。

- タスク名
- タスクの件名 (オプション、推奨)
- アクターまたはグループ (または両方)
- 入力データ

以下の手順に沿って、Swagger REST API ツールを使用して、Business Central で利用可能な **IT_Orders** サンプルプロジェクトの動的ユーザータスクを作成します。Swagger のない REST API でも、同じエンドポイントを利用できます。

前提条件

- Business Central にログインしており、Showcase アプリケーションを使用して IT Orders ケースインスタンスを起動している。Showcase の使用に関する情報は、『ケース管理への Showcase アプリケーション』を参照してください。

手順

1. Web ブラウザーで、以下の URL を開きます。
/http://localhost:8080/kie-server/docs
2. **Case instances :: Case Management** で利用可能なエンドポイントの一覧を開きます。
3. 以下の **POST** メソッドのエンドポイントをクリックし、詳細を開きます。
/server/containers/{id}/cases/instances/{caseId}/tasks
4. **Try it out** をクリックしてから、以下のパラメーターを入力します。

表6.1 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**

本文

```
{
 "name" : "RequestManagerApproval",
 "data" : {
   "reason" : "Fixed hardware spec",
   "caseFile_hwSpec" : "#{caseFile_hwSpec}"
  },
 "subject" : "Ask for manager approval again",
 "actors" : "manager",
 "groups" : ""
}
```

5. Swagger アプリケーションで、**Execute** をクリックして動的タスクを作成します。

この手順は、ケース **IT-000000001** に関連付けられている新しいユーザータスクを作成します。このタスクは、**manager** ケースロールに割り当てるユーザーに割り当てられます。このタスクには、2 つの入力変数があります。

- **reason**
- **caseFile_hwSpec**: defined as an expression to allow run time capturing of a process or case data.

タスクによっては、タスク名で検索できる、ユーザーフレンドリーな UI を提供するフォームが提供されている場合があります。IT Orders のケースでは、**RequestManagerApproval** タスクには、KJAR に **RequestManagerApproval-taskform.form** フォームが含まれます。

タスクを作成すると、Business Central で、タスクが割り当てられたユーザーの **Task Inbox** にタスクが表示されます。

6.2. REST API を使用した動的サービスタスクの作成

サービスタスクは通常、ユーザータスクより複雑ではありませんが、正常に実行するにはさらにデータが必要となる可能性があります。サービスタスクには以下の情報が必要です。

- **name**: アクティビティー名
- **nodeType**: 作業アイテムハンドラーの検索に使用するノードタイプ
- **data**: 正しく実行を処理するためのデータのマッピング

ケースの実行時に、ユーザータスクと同じエンドポイントを使用して動的サービスタスクを作成できますが、ボディーペイロードは異なります。

以下の手順に沿って、Swagger REST API ツールを使用して、Business Central で利用可能な **IT_Orders** サンプルプロジェクトの動的サービスタスクを作成します。Swagger のない REST API でも、同じエンドポイントを利用できます。

前提条件

- Business Central にログインしており、Showcase アプリケーションを使用して IT Orders ケースインスタンスを起動している。Showcase の使用に関する情報は、『ケース管理への Showcase アプリケーション』を参照してください。

手順

1. Web ブラウザーで、以下の URL を開きます。
   /http://localhost:8080/kie-server/docs
2. **Case instances :: Case Management** で利用可能なエンドポイントの一覧を開きます。
3. 以下の **POST** メソッドのエンドポイントをクリックし、詳細を開きます。
   /server/containers/{id}/cases/instances/{caseId}/stages/{caseStageId}/tasks
4. **Try it out** をクリックしてから、以下のパラメーターを入力します。

   表6.2 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**

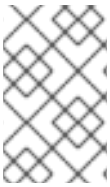
   本文

```
{
 "name" : "InvokeService",
 "data" : {
   "Parameter" : "Fixed hardware spec",
   "Interface" : "org.jbpm.demo.itorders.services.ITOrderService",
   "Operation" : "printMessage",
   "ParameterType" : "java.lang.String"
  },
 "nodeType" : "Service Task"
}
```

5. Swagger アプリケーションで、**Execute** をクリックして動的タスクを作成します。

この例では、Java ベースのサービスが実行されます。この例には、**org.jbpm.demo.itorders.services.ITOrderService** のパブリッククラスと、**String** が 1 つ指定された **printMessage** パブリックメソッド、インターフェースで構成されます。このサービスを実行すると、パラメーターの値がメソッドに渡されて実行されます。

サービスタスク作成に指定する数字、名前、他のタイプのデータは、サービスタスクのハンドラーの実装により異なります。提供されている例では、**org.jbpm.process.workitem.bpmn2.ServiceTaskHandler** ハンドラーが使用されています。



注記

カスタムサービスタスクの場合は、**Work Item Handlers** セクションのデプロイメント記述子にハンドラーが登録されていることを確認します。名前は、動的サービスタスクの作成に使用される **nodeType** と同じです。

6.3. REST API を使用した動的サブプロセスの作成

動的サブクラスを作成すると、任意のデータのみが提示されます。動的タスクの作成時には、特別なパラメーターはありません。

以下の手順では、Swagger REST API ツールを使用して、Business Central で利用可能な **IT_Orders** サンプルプロジェクトの動的なサブプロセスタスクを作成する方法を説明します。Swagger のない REST API でも、同じエンドポイントを利用できます。

前提条件

- Business Central にログインしており、Showcase アプリケーションを使用して IT Orders ケースインスタンスを起動している。Showcase の使用に関する情報は、『ケース管理への Showcase アプリケーション』を参照してください。

手順

1. Web ブラウザーで、以下の URL を開きます。
/http://localhost:8080/kie-server/docs
2. **Case instances :: Case Management** で利用可能なエンドポイントの一覧を開きます。
3. 以下の **POST** メソッドのエンドポイントをクリックし、詳細を開きます。
/server/containers/{id}/cases/instances/{caseId}/processes/{pId}
4. **Try it out** をクリックして以下のパラメーターを入力します。

   表6.3 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**
pId	**itorders-data.place-order**

   pId は、作成するサブプロセスのプロセス ID です。

   本文

```
{
 "placedOrder" : "Manually"
}
```

5. Swagger アプリケーションで、**Execute** をクリックして動的サブプロセスを開始します。

この例では、ケース ID **IT-0000000001** の **place-order** サブプロセスが IT 発注ケースで開始しています。Business Central の **Menu → Manage → Process Instances** の下で、このプロセスを確認できます。

説明に使用されている例を正しく実行したら、**place-order** プロセスがプロセスインスタンスの一覧に表示されます。プロセスの詳細を開き、プロセスの相関キーに IT 発注ケースインスタンス ID が含まれていることに注意してください。**Process Variables** 一覧には、REST API 本文に配信されているように、**Manually** 値を持つ **placedOrder** 変数が含まれます。

第7章 コメント

ケース管理では、コメントを使用すればケースインスタンス内での協業作業が容易になり、簡単にケース作業者が互いに情報を交換できるようになります。

コメントはケースインスタンスにバインドされます。ケースインスタンスはケースファイルの一部であるため、コメントを使用してインスタンスに対してアクションを実行できます。基本的なテキストベースのコメントには、CRUD (作成、読み取り、更新、削除) と同様の完全な操作セットを含めることができます。

第8章 ケースロール

ケースロールは、ユーザーがケース処理に参加する追加の抽象層を提供します。ロール、ユーザー、グループは、ケース管理の別の目的に使用されます。

ロール

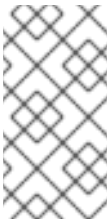
ロールは、ケースインスタンスの認証や、ユーザーアクティビティーの割り当てを可能にします。所有者ロールには、ユーザーまたは1つ以上のグループを割り当てることができます。ケースが所属するユーザーが所有者です。ケースの定義では、ロールの割り当てはユーザーまたはグループ1つだけに制限されません。特定のユーザーまたはグループにタスクを割り当てる代わりに、ロールを使用してタスクの割り当てを指定することで、ケースを動的に保ちます。

Groups

グループとは、特定のタスクを実行できるユーザー、または指定の責任が割り当てられたユーザーの集合です。グループには何人でも割り当てることができ、ロールにはどのグループでも割り当てることができます。グループのメンバーは、いつでも追加または変更できるので、特定のタスクにグループをハードコード化しないでください。

Users

ユーザーとは、ロールに割り当てたり、グループに追加したりして、特定のタスクを割り当てることができる個人を指します。



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。

以下の例では、前述のケース管理の概念をホテル予約にどのように適用するかを説明します。

- ロール = **Guest**
- グループ = **Receptionist**、**Maid**
- ユーザー = **Marilyn**

Guest のロールを割り当てると、関連ケースの特定の作業に影響があり、ケースインスタンスごとに固有です。ロールに割り当てることができるユーザーまたはグループの数は **Case Cardinality** で制限されています。Case Cardinality は、プロセスデザイナーのロール作成時に設定されます (例: ホテル予約ケースではゲストロールが1つ、**IT_Orders** サンプルプロジェクトでは IT ハードウェア業者ロールが2つです)。

ロールが定義されている場合には、ロールがケース定義の一部としてユーザー1人またはグループ1つにハードコードされておらず、ケースインスタンスごとに違うものを指定できるようにする必要があります。ケースのロール割り当てが重要なのは、このような理由からです。

ロールは、ケースの開始時や、ケースがアクテイブになった時点で割り当てまたは割り当ての解除ができます。ロールはオプションですが、ケース定義でロールを使用して、整理されたワークフローを維持します。

重要

タスク割り当てに実際のユーザーまたはグループ名を使用する代わりに、ロールを使用します。これにより、必要に応じて、ユーザーまたはグループを実際に動的に割り当てるタイミングを遅らせることができます。

ロールはユーザーまたはグループに割り当てられ、ケースインスタンスの起動時にタスクを実行する権限があります。

8.1. ケースロールの作成

プロセスデザイナーでケースの設計時に、ケース定義でケースのロールを作成して、定義できます。ケースのロールは、ケースの定義レベルで設定して、ケースインスタンスを処理するアクターと分離させることができます。また、ロールは、ユーザータスクに割り当てるか、ケースのライフサイクル全体で問い合わせの参照として使用することができますが。固有のユーザーまたはユーザーのグループとして、ケースには定義されていません。

ケースインスタンスには、ケースの作業を実際に処理する個人が含まれます。新規ケースインスタンスを開始する場合にはロールを割り当ててください。ケースのランタイム中に、ケースのロール割り当てを変更して、ケースの柔軟性を保つことができますが、以前のロール割り当てをもとにすでに作成されているタスクには効果がありません。ロールに割り当てられたアクターには柔軟性がありますが、ロール自体はどのケースも同じままです。

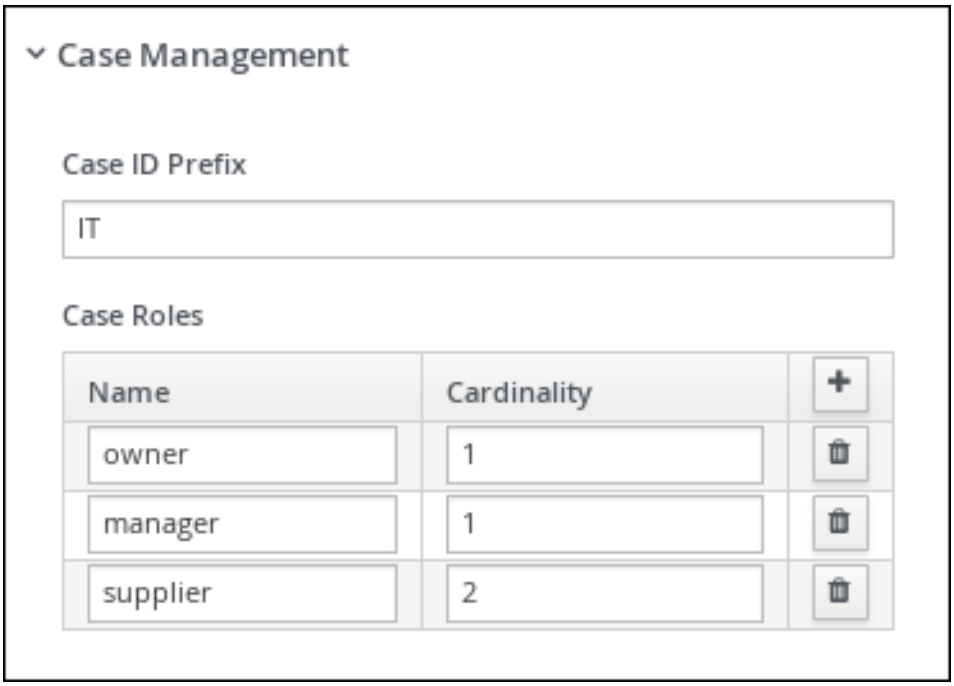
前提条件

- ケース定義が含まれるケースプロジェクトが Business Central に存在する。
- プロセスデザイナーでケース定義アセットが開いている。

手順

1. ケースに関連するロールを定義するには、エディターのキャンバスの空いているスペースをクリックします。 をクリックして **Properties** メニューをクリックします。
2. **Case Management** を展開してケースロールを追加します。
   ケースのロールには、ロールの名前とケースカーディナリティーが必要です。ケースカーディナリティーとは、ケースインスタンスでロールに割り当てられたアクターの数です。たとえば、**IT_Orders** サンプルのケース管理プロジェクトには、以下のロールが含まれます。

図8.1 ITOrders ケースロール



この例では、ケースの **owner** にアクター (ユーザーまたはグループ) 1 つのみを、**manager** ロールには 1 つのアクターのみを割り当てることができます。**supplier** ロールには、アクターを 2 つ割り当てることができます。ケースによっては、ロールに設定済みのケースカーディナリティーをもとに、特定のロールにいくつでもアクターを割り当てることができます。

8.2. ロールの認証

ロールには、Showcase アプリケーションまたは REST API を使用して新しいケースインスタンスを開始するときに、特定のケース管理タスクを実行する権限があります。

以下の手順では、REST API を使用して新しい IT 発注ケースを開始します。

前提条件

- **IT_Orders** サンプルプロジェクトが Business Central にインポートされており、KIE Server にデプロイされている。

手順

1. 以下のエンドポイントで **POST** REST API コールを作成します。
   http://host:port/kie-server/services/rest/server/containers/itorders/cases/itorders.orderhardware/instances
   - **itorders**: KIE Server でデプロイしているコンテナーエイリアス。
   - **itorders.orderhardware**: ケース定義の名前。
2. 要求の本文に以下のロール設定を追加します。

```
{
  "case-data" : {  },
  "case-user-assignments" : {
    "owner" : "cami",
    "manager" : "cami"
```

```
  },
  "case-group-assignments" : {
    "supplier" : "IT"
  }
}
```

これにより、定義したロールと、自動的に開始するアクティビティーを持つ新しいケースが開始します。これは開始して、処理できるようになります。ロールのうち 2 つがユーザーへの割り当て (**owner** および **manager**) で、3 つ目はグループへの割り当て (**supplier**) です。

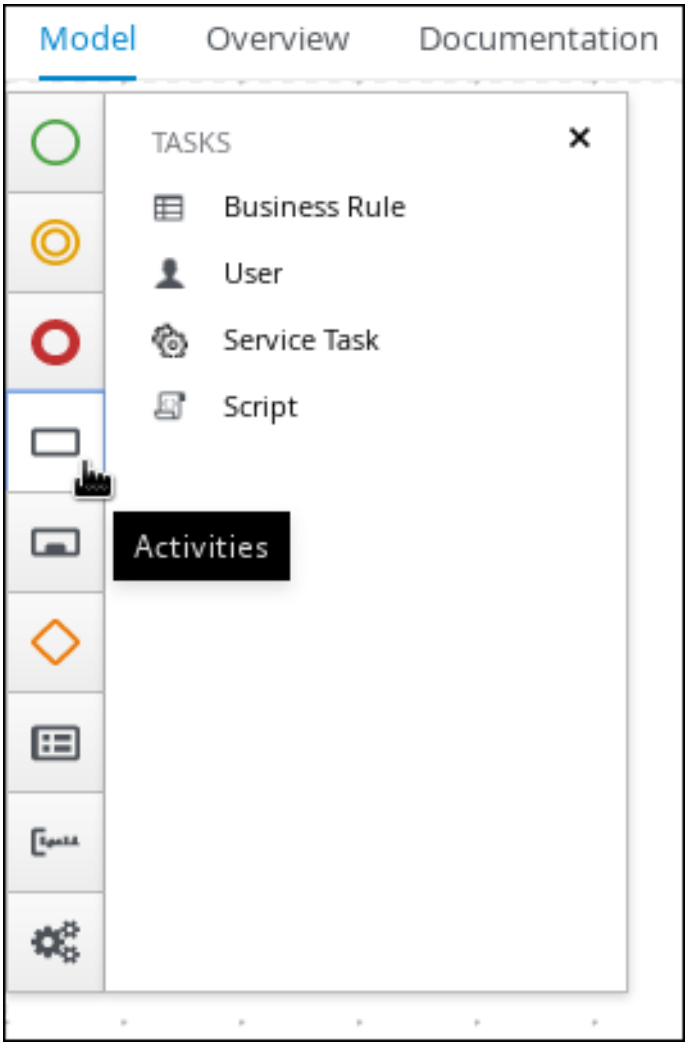
ケースインスタンスが正常に開始されると、**IT-0000000001** ケース ID を返します。

Showcase アプリケーションを使用して新規ケースインスタンスを開始する方法は、『**ケース管理への Showcase アプリケーションの使用**』を参照してください。

8.3. ロールへのタスクの割り当て

ケース管理プロセスは、新しいケースインスタンス、またはアクティブなケースのユーザー割り当てを変更するなど、ランタイム時に動的に発生する変更に対応するために、できるだけ柔軟である必要があります。このため、ケース定義にあるユーザーまたはグループの単一セットにハードコードしないようにしてください。代わりに、ケース作成時にロールにユーザーまたはグループが割り当てられ、ロール割り当てをケース定義のタスクノードに定義できます。

Red Hat Process Automation Manager には、ビジネスプロセスの作成を簡略化する、事前定義済みのノードタイプが各種含まれます。事前定義済みのノードパネルは、ダイアグラムエディターの左側に置かれます。

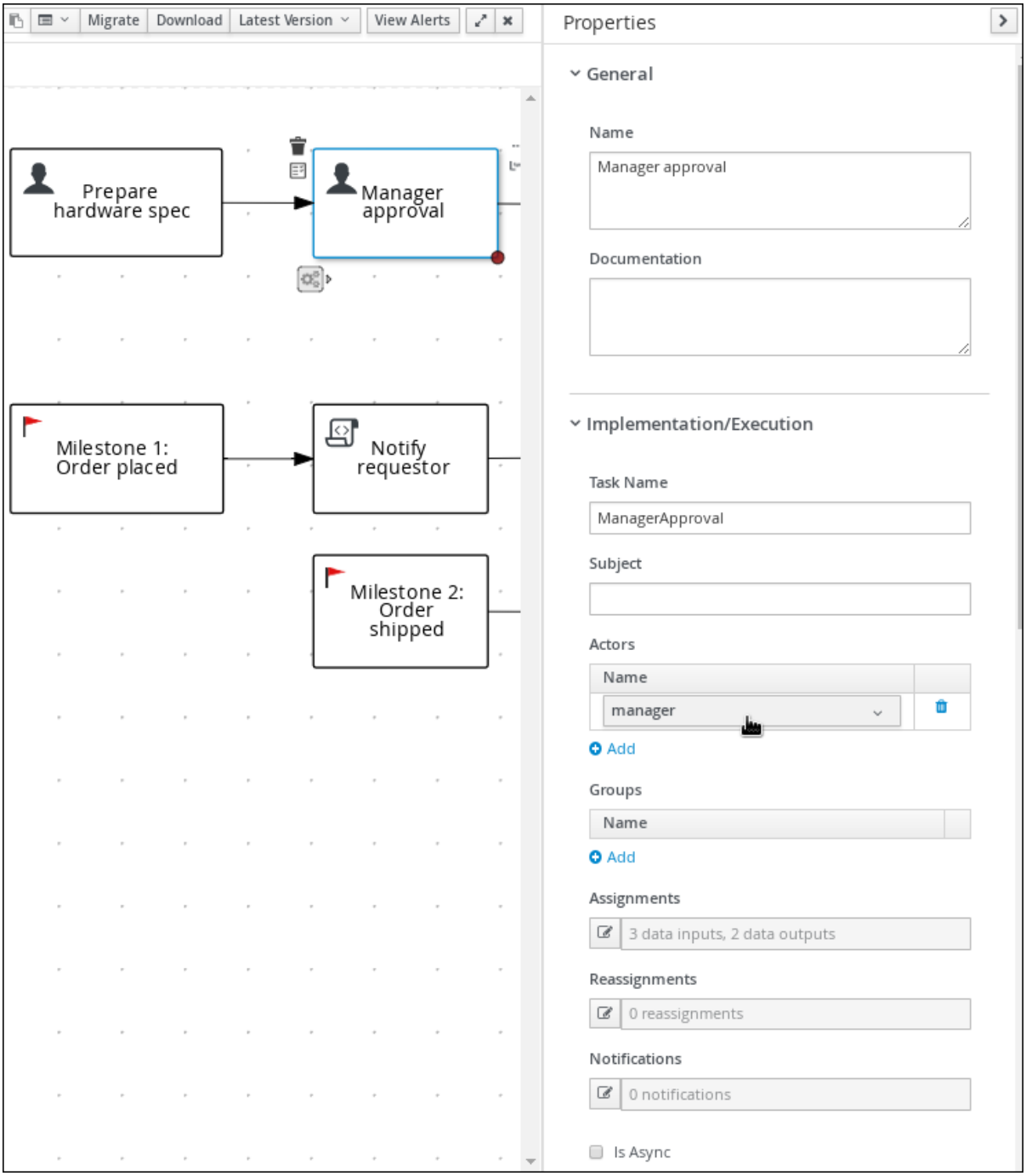


前提条件

- ケース定義が、ケース設定レベルに設定したケースロールで作成されている。ケースロールの作成方法は「ケースロールの作成」を参照してください。

手順

1. Tasks 一覧を開き、プロセス設定パレットにケース定義を追加するユーザーまたはサービスタスクをドラッグします。
2. タスクノードを選択して、 をクリックし、デザイナーの右側の Properties パネルを開きます。
3. Implementation/Execution を展開し、Actors プロパティーの下にある Add をクリックして、タスクを割り当てるロール名を選択するか、入力します。グループの割当も同じように、Groups プロパティーを使用します。
   たとえば、IT_Orders のサンプルプロジェクトでは、**Manager approval** ユーザータスクが **manager** ロールに割り当てられています。



この例では、**Prepare hardware spec** ユーザータスクが完了すると、**manager** ロールに割り当てられているユーザーは、Business Central の **Task Inbox** で **Manager approval** を受け取ります。

ロールに割り当てられているユーザーはケースのランタイム時に変更できますが、タスクそのものには引き続き同じロールが割り当てられます。たとえば、**manager** ロールに最初に割り当てられたユーザーが (病気などで) 時間休をとる場合、または予定外に退職する場合などが考えられます。そのような状況でこの変更に応えるには、**manager** ロールの割り当てを編集して、そのロールに関連付けられているタスクに他のユーザーを割り当てることができます。

ランタイム時にロール割り当てを変更する方法は「Showcase を使用してランタイム時にケースのロール割り当ての修正」または「REST API を使用してランタイム時にケースのロール割り当ての修正」を参照してください。

8.4. SHOWCASE を使用してランタイム時にケースのロール割り当ての修正

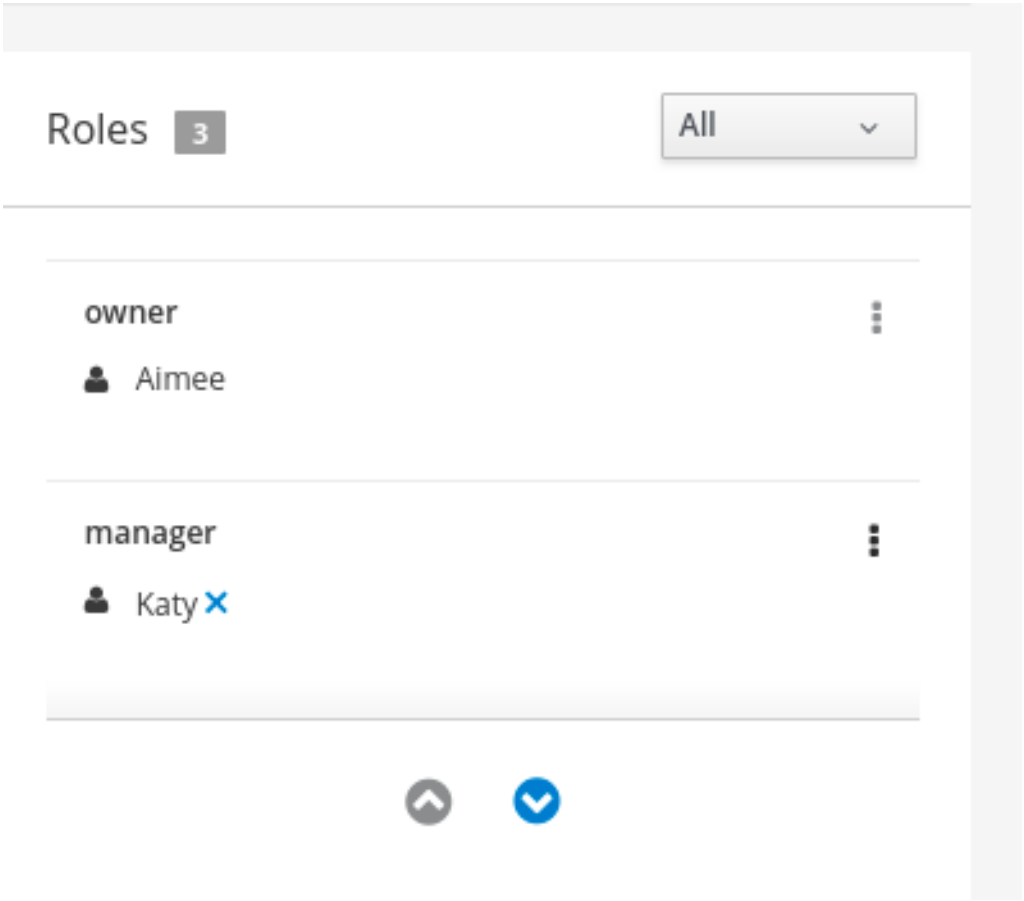
Showcase アプリケーションを使用して、ランタイム時にケースインスタンスのロール割り当てを変更できます。ロールはケース定義に定義され、ケースのライフサイクルでタスクに割り当てます。ロールは事前に定義されるためランタイム時に変更できませんが、ロールに割り当てたアクターを、ケースタスクを実行するユーザーに変更できます。

前提条件

- アクティブなケースインスタンスがあり、その中ですでにユーザーまたはグループが最低でも 1 つのケースロールに割り当てられている。

手順

1. Showcase アプリケーションで、**Case list** から作業するケースをクリックし、ケースの概要を開きます。
2. ページの右下の **Roles** ボックスで、変更するロール割り当てを探します。



3. ロールの割り当てからユーザまたはグループを削除するには、対象のロール割り当ての横にある ✕ をクリックします。確認ウィンドウで、**Remove** をクリックして、ロールからユーザーまたはグループを削除します。
4. ロールからロール割り当てすべてを削除するには、ロールの横にある ⋮ をクリックして、**Remove all assignments** オプションを選択します。確認ウィンドウで、**Remove** をクリックして、ロールからユーザーとグループ割り当てすべてを削除します。
5. ロール割り当てを別のユーザーやグループに変更するには、ロールの横にある ⋮ をクリックして、**Edit** オプションを選択します。

6. **Edit role assignment** ウィンドウで、ロール割り当てから削除する割り当て先の名前を削除します。ロールに割り当てるユーザーの名前を **User** フィールドに入力するか、割り当てるグループを **Group** フィールドに追加します。
ロール割り当ての編集時に、1 つ以上のユーザーまたはグループが割り当てられている必要があります。

7. **Assign** をクリックしてルールの割り当てを完了します。

8.5. REST API を使用してランタイム時にケースのロール割り当ての修正

REST API または Swagger アプリケーションを使用して、ランタイム時にケースインスタンスのロール割り当てを変更できます。ロールはケース定義に定義され、ケースのライフサイクルでタスクに割り当てます。ロールは事前に定義されるためランタイム時に変更できませんが、ロールに割り当てたアクターを、ケースタスクを実行するユーザーに変更できます。

以下の手順には、**IT_Orders** サンプルプロジェクトをもとにした例が含まれます。Swagger アプリケーション、または他の REST API クライアントと同じ REST API エンドポイントを使用するか、Curl を使用します。

前提条件

- IT 発注ケースインスタンスを、**owner**、**manager**、および **supplier** のロールがアクターにすでに割り当てられている状態で開始している。

手順

1. 以下のエンドポイントで **GET** リクエストを使用して現在のロール割り当ての一覧を取得します。
/http://localhost:8080/kie-server/services/rest/server/containers/{id}/cases/instances/{caseId}/roles

   表8.1 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**

   これにより、以下の応答が返されます。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<case-role-assignment-list>
    <role-assignments>
        <name>owner</name>
        <users>Aimee</users>
    </role-assignments>
    <role-assignments>
        <name>manager</name>
        <users>Katy</users>
    </role-assignments>
    <role-assignments>
        <name>supplier</name>
```

```
            <groups>Lenovo</groups>
        </role-assignments>
    </case-role-assignment-list>
```

2. **manager** ロールに割り当てられているユーザーを変更する場合は、最初に **DELETE** を使用して、ユーザー **Katy** からロール割り当てを削除する必要があります。
/server/containers/{id}/cases/instances/{caseId}/roles/{caseRoleName}

   Swagger クライアントリクエストに以下の情報を追加します。

   表8.2 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**
caseRoleName	**manager**
user	**Katy**

   Execute をクリックします。

3. 最初の手順の **GET** リクエストを再実行し、**manager** ロールがユーザーに割り当てられなくなったことを確認します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<case-role-assignment-list>
    <role-assignments>
        <name>owner</name>
        <users>Aimee</users>
    </role-assignments>
    <role-assignments>
        <name>manager</name>
    </role-assignments>
    <role-assignments>
        <name>supplier</name>
        <groups>Lenovo</groups>
    </role-assignments>
</case-role-assignment-list>
```

4. 以下のエンドポイントで **PUT** リクエストを使用して、**Cami** ユーザーを **manager** ロールに割り当てます。
/server/containers/{id}/cases/instances/{caseId}/roles/{caseRoleName}

   Swagger クライアントリクエストに以下の情報を追加します。

   表8.3 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**
caseRoleName	**manager**
user	**Cami**

Execute をクリックします。

5. 最初の手順の **GET** リクエストを再実行し、**manager** ロールが **Cami** に割り当てられていることを確認します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<case-role-assignment-list>
    <role-assignments>
        <name>owner</name>
        <users>Aimee</users>
    </role-assignments>
    <role-assignments>
        <name>manager</name>
        <users>Cami</users>
    </role-assignments>
    <role-assignments>
        <name>supplier</name>
        <groups>Lenovo</groups>
    </role-assignments>
</case-role-assignment-list>
```

第9章 ステージ

ケース管理ステージはタスクの集まりです。ステージは、プロセスデザイナーを使用して定義できるアドホックサブプロセスで、マイルストーンなどの、別のケース管理ノードに含まれる可能性もあります。マイルストーンは、1つのステージまたは複数のステージが完了した場合に完了するように設定されます。したがって、マイルストーンはステージの完了によりアクティベートまたは達成することができ、ステージにはマイルストーンを1つまたは複数追加できます。

たとえば、患者のトリアージケースでは、最初のステージでは、明らかな身体症状を観察して書き留めたり、その症状が何かを患者に説明してもらい、2番目のステージでテストを行い、3番目のステージで診断および治療を行います。

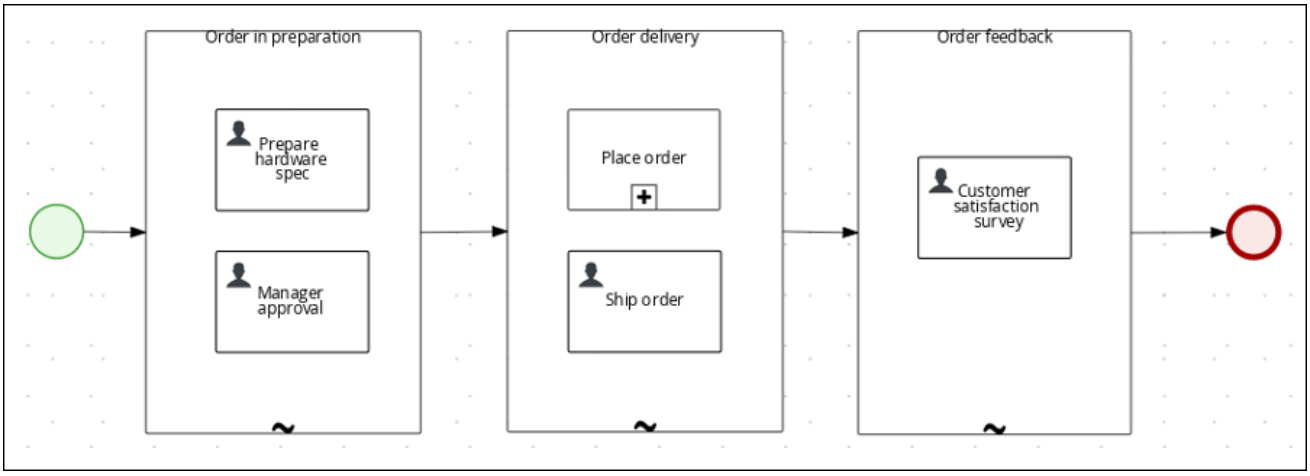
ステージを完了する3つの方法があります。

- 完了条件
- 終端の終了イベント
- **Completion Condition** を **autocomplete** に設定して、ステージにアクティブなタスクがなくなったら、自動的にそのステージを完了します。

9.1. ステージの定義

ステージは、プロセスデザイナーを使用して BPMN2 でモデル化できます。ステージとは、ステージがアクティベートされている場合に、次のステージの開始前に、完了する必要のあるアクティビティーが明確に定義されるように、関連のタスクをグループ化する手段のことを指します。たとえば、ステージを使用して、以下の方法で **IT_Orders** ケース定義を設定することもできます。

図9.1IT_Orders プロジェクトステージの例



手順

1. ダイアグラムエディターの左側にある事前定義済みのノードパネルから、**Ad-Hoc** ノードをデザインキャンバスにドラッグアンドドロップして、ステージノードの名前を指定します。
2. ステージをアクティベートする方法を定義します。
   - 受信ノードがステージをアクティベートした場合は、ステージを、受信ノードのシーケンスフローラインに接続します。
   - シグナルイベントで、このステージが代わりにアクティベートされている場合は、シグナルイベントに、最初の手順で設定したステージの名前で **SignalRef** を設定します。

- または、条件が満たされると、ステージをアクティベートするように **AdHocActivationCondition** プロパティーを設定します。

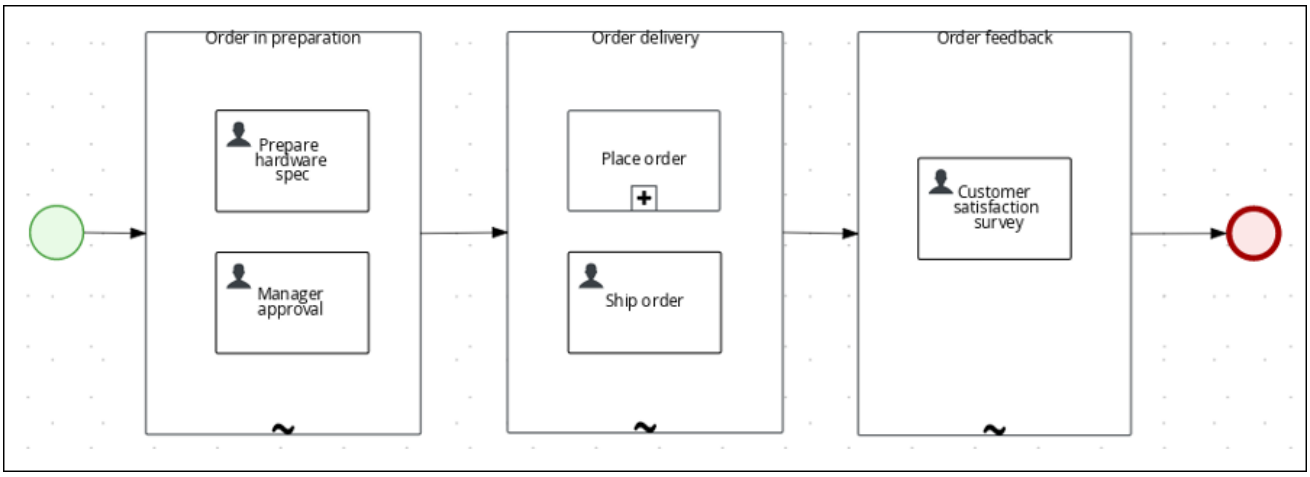
3. ステージにタスクノードを追加する余裕を持たせるために、必要に応じてノードのサイズを変更します。
4. 関連タスクをステージに追加して、必要に応じて設定します。
5. (任意) ステージの完了条件を設定します。アドホックサブプロセスとして、ステージはデフォルトで **autocomplete** として設定されます。これは、ステージが自動的に完了し、ステージ内のすべてのインスタンスがアクティブでなくなると、ケース定義の次のアクティビティーが発生することを示しています。
   完了条件を変更するには、ステージノードを選択して、右側の **Properties** パネルを選択し、**Implementation/Execution** を展開して、 必要な完了条件になるように free-form Drools 式を使用して **AdHocCompletionCondition** を変更します。ステージ完了条件に関する情報は、「ステージのアクティベーションおよび完了条件の設定」を参照してください。
6. ステージを設定したら、シーケンスフローラインを使用して、ケース定義内の次のアクティビティーに接続します。

9.2. ステージのアクティベーションおよび完了条件の設定

開始ノード、中間ノード、または手動の API コールを使用してステージを発生できます。

free-form Drools ルールを使用して、マイルストーンの完了条件を設定するのと同じ方法で、アクティベーションと完了条件の両方を含めてステージを設定できます。たとえば、**IT_Orders** サンプルプロジェクトでは、**Milestone 2: Order shipped** の完了条件 (**org.kie.api.runtime.process.CaseData(data.get("shipped") == true)**) を、ここで使用されている **Order delivery** の完了条件として使用することも可能です。

図9.2 IT_Orders プロジェクトのステージ例



ステージをアクティベートする **AdHocActivationCondition** プロパティーを設定するアクティベーション条件は、Free Form Drools ルールを使用しても設定できます。

前提条件

- Business Central プロセスデザイナーでケース定義を作成している。
- アドホックサブプロセスを、ステージとして使用されるケース定義に追加している。

手順

1. ステージを選択して、 をクリックし、デザイナーの右側の **Properties** パネルを開きます。

2. **Implementation/Execution** を展開して、**AdHocActivationCondition** プロパティーエディターを開き、開始ノードのアクティベーション条件を定義します。たとえば、**autostart: true** と設定して、新規ケースインスタンスが開始されたら、ステージが自動的にアクティベートされるようにします。

3. **AdHocCompletionCondition** はデフォルトでは、**autocomplete** に設定されています。これを変更するには、free-form Drools 式を使用して完了条件を入力します。たとえば、**org.kie.api.runtime.process.CaseData(data.get("ordered") == true)** と設定して、以前の例の 2 つ目のステージをアクティベートします。

IT_Orders サンプルプロジェクトで使用する条件に関する例や情報は『ケース管理の使用ガイド』を参照してください。

9.3. ステージへの動的タスクの追加

動的タスクは、REST API 要求を使用してランタイム時にケースステージに追加できます。これは、ケースインスタンスに動的タスクを追加することに似ていますが、タスクが追加されるステージの **caseStageId** を定義する必要もあります。

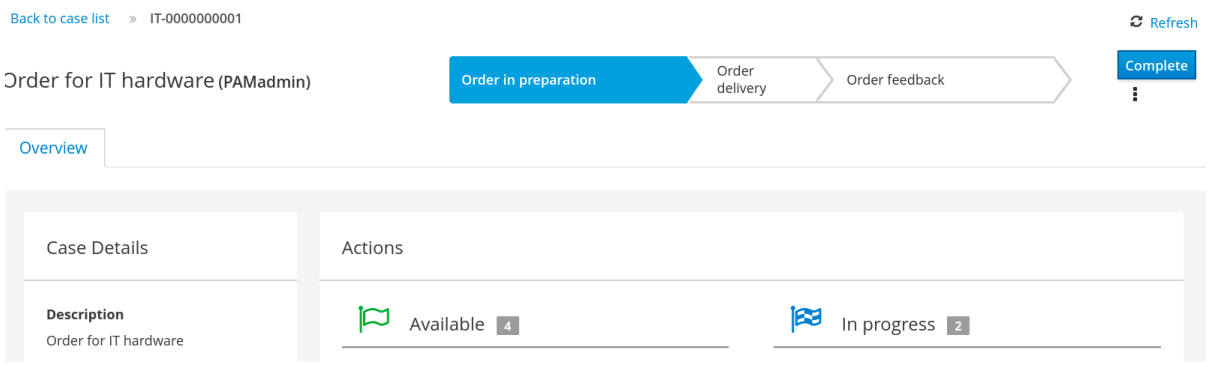
以下の手順に沿って、Swagger REST API ツールを使用して、Business Central で利用可能な IT_Orders サンプルプロジェクトの動的タスクをステージに追加します。Swagger のない REST API でも、同じエンドポイントを利用できます。

前提条件

- 以前の例に示されているように、IT_Orders サンプルプロジェクトの BPMN2 ケース定義は、マイルストーンではなくステージを使用して再設定できます。 ケース管理向けにステージを設定する方法については、「ステージの定義」を参照してください。

手順

1. Showcase アプリケーションを使用して新規ケースインスタンスを開始します。Showcase の使用に関する情報は、『ケース管理への Showcase アプリケーションの使用』を参照してください。
このケースはステージを使用して作成されているため、ケース詳細ページにはステージの追跡が表示されます。



最初のステージは、ケースインスタンスの開始時に自動的に開始します。

2. **manager** ユーザーとして、Business Central の **Menu → Track → Task Inbox** の下で、ハードウェア明細書を承認し、ケースの進捗を確認します。

   a. Business Central で **Menu → Manage → Process Instances** の順にクリックし、アクティブケースインスタンス **IT-0000000001** を開きます。

   b. **Diagram** をクリックして、ケースの進捗を表示します。

3. Web ブラウザーで、以下の URL を開きます。
   /http://localhost:8080/kie-server/docs

4. **Case instances :: Case Management** で利用可能なエンドポイントの一覧を開きます。

5. 以下の **POST** メソッドのエンドポイントをクリックし、詳細を開きます。
   /server/containers/{id}/cases/instances/{caseId}/stages/{caseStageId}/tasks

6. **Try it out** をクリックして、以下のパラメーターを完了します。

   表9.1 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**
caseStageId	**Order delivery**

   caseStageId は、ケース定義に含まれるステージ名です。このケース定義で、動的タスクが作成されます。これには、動的またはサービスタスクペイロードを指定できます。サンプルについては、「REST API を使用した動的サブプロセスの作成」または「REST API を使用した動的サービスタスクの作成」を参照してください。

動的タスクをステージに追加したら、ステージを完了して、ケースフローの次の項目にプロセスを進ませるために、その動的タスクを完了する必要があります。

第10章 マイルストーン

マイルストーンとは、プロセスデザイナーパレットにマイルストーンノードを追加して、ケース定義デザイナーで設定できる、特別なサービスタスクのことです。新規ケース定義の作成時に、 **Adhoc autostart** として設定されたマイルストーンは、デフォルトでデザインパレットに含まれます。新規作成したマイルストーンはデフォルトで **Adhoc autostart** には設定されません。

ケース管理のマイルストーンは、ステージの最後に発生するのが通常ですが、他のマイルストーンを達成した結果として発生する場合もあります。マイルストーンには、進捗を追跡するために、条件を定義する必要があります。マイルストーンは、ケースにデータを追加すると、ケースファイルのデータに反応します。また、マイルストーンは、ケースインスタンス内の達成地点を表し、特定のイベントにフラグを立てるために使用できます。これは、重要業績評価指標 (KPI) の追跡や、完了前のタスクの特定に有用な場合があります。

マイルストーンには、ケース実行中の以下のいずれのか状態を指定できます。

- **Active**: 条件はマイルストーンで定義されているが、条件がまだ満たされていない。
- **Completed**: マイルストーンの条件が満たされ、達成されたので、このケースは次のタスクに進むことができる。
- **Terminated**: マイルストーンがケースプロセスから除外され、必要なくなっている。

Adhoc autostart が設定されていて、マイルストーンが使用可能な場合や、完了している場合には、自動的に、またはシグナルを使用して手動で、ケースインスタンスの開始時に、マイルストーンをトリガーできます。マイルストーンは何回でもトリガーできますが、条件が満たされている場合には、直接マイルストーンが達成されます。

10.1. マイルストーンの設定およびトリガー

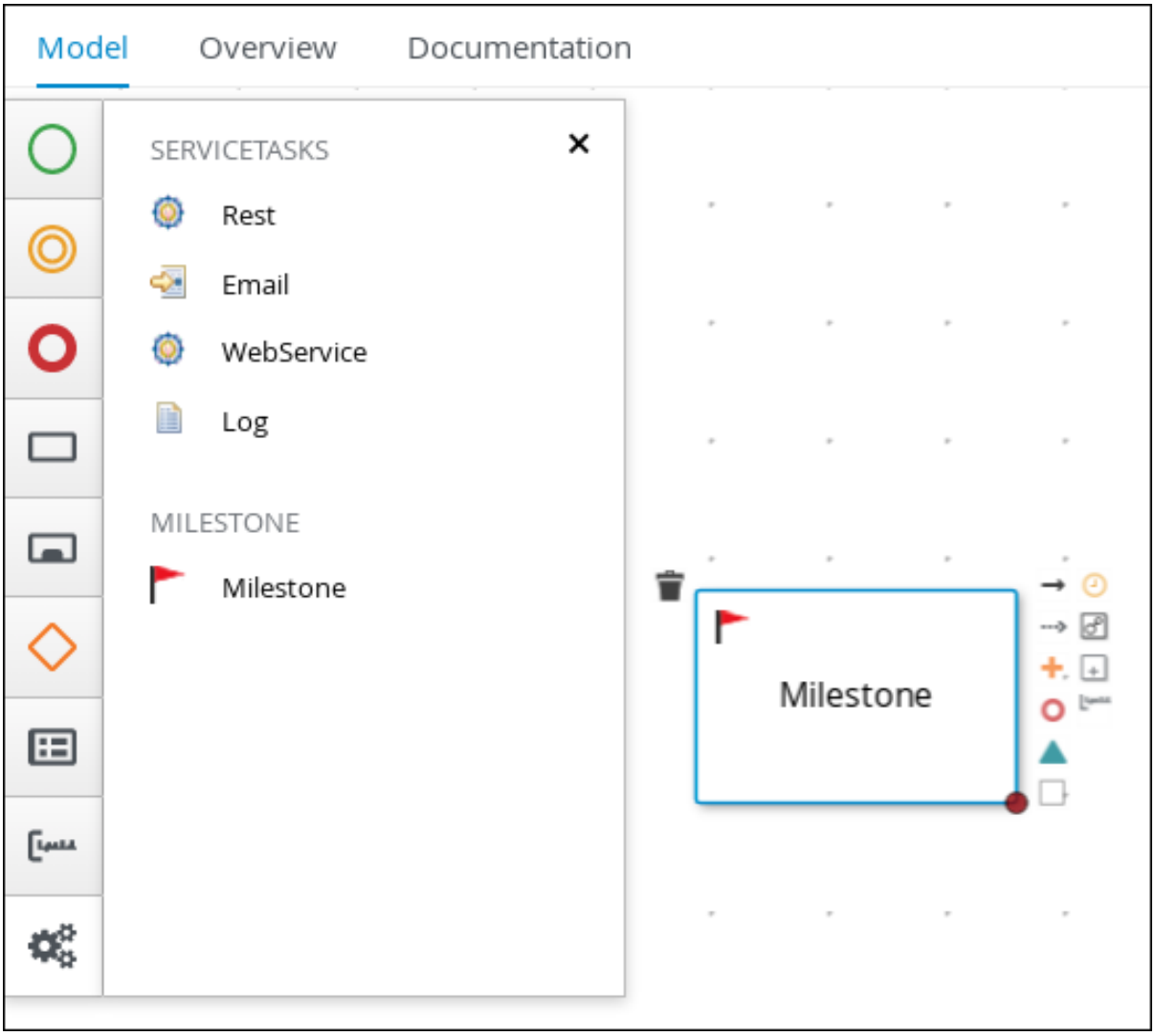
マイルストーンは、ケースインスタンスの開始時に自動的に開始するように設定できます。または、ケース設計時に手動で設定したシグナルを使用して発生させることもできます。

前提条件

- Business Central でケースプロジェクトが作成されている。
- ケース定義が作成されている。

手順

1. ダイアグラムエディターの左側にある事前定義済みのノードパネルから、**Milestone** オブジェクトをパレットにドラッグアンドドロップします。



2. マイルストーンを選択して、 をクリックし、デザイナーの右側の **Properties** パネルを開きます。

3. **Data Assignments** を展開して完了条件を追加します。マイルストーンには、デフォルトで **condition** パラメーターが含まれます。

4. マイルストーンに完了条件を定義するには、**Source** 一覧から **Constant** を選択します。条件は Drools 構文で定義する必要があります。

5. **Implementation/Execution** を展開して、**Adhoc Autostart** プロパティーを設定します。

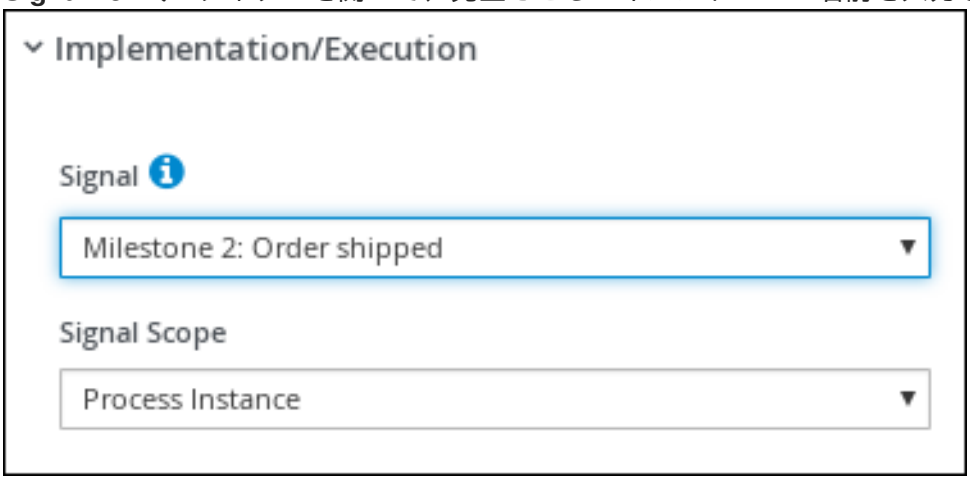
   - ケースインスタンスの開始時に自動的に開始する必要があるマイルストーンの場合は、チェックボックスを選択して、このプロパティーを **true** に設定します。
   - シグナルイベントで発生させるマイルストーンの場合は、チェックボックスを選択せずに、このプロパティーを **false** に設定します。

6. **(任意)** ケースゴールが達成した場合にマイルストーンを発生させるシグナルイベントを設定します。

   a. ケース設計パレットでシグナルイベントを選択した状態で、右側に **Properties** パネルを開きます。

   b. **Signal Scope** プロパティーを **Process Instance** に設定します。

c. **SignalRef** 式エディターを開いて、発生させるマイルストーンの名前を入力します。



7. **Save** をクリックします。

第11章 変数タグ

ランタイム時に使用するデータを格納する変数。変数の動作をより細かく制御するには、BPMN ケースファイルでケース変数とローカル変数にタグ付けできます。タグは、特定の変数にメタデータとして追加する単純な文字列値です。

Red Hat Process Automation Manager は、ケースとローカル変数の以下のタグをサポートします。

- **required**: ケースを開始するための要件として変数を設定します。要件である変数なしでケースインスタンスを起動すると、Red Hat Process Automation Manager は **VariableViolationException** エラーを生成します。
- **readonly**: 変数が情報提供のみを目的としており、設定できるのはケースの実行中に 1 回のみであることを示します。readonly 変数の値がいずれかの時点で変更されると、Red Hat Process Automation Manager は **VariableViolationException** エラーを生成します。
- **restricted**: **VariableGuardProcessEventListener** で使用するタグで、既存のロールをもとに変数を変更できるパーミッションが付与されていることを示します。2 つ目のコンストラクターを使用して、新規タグ名を渡す場合には、**restricted** タグは他のタグ名に置き換えることができます。

VariableGuardProcessEventListener クラスは、**DefaultProcessEventListener** クラスから拡張されたもので、2 つの異なるコンストラクターをサポートします。

- **VariableGuardProcessEventListener**

```
public VariableGuardProcessEventListener(String requiredRole, IdentityProvider
identityProvider) {
    this("restricted", requiredRole, identityProvider);
}
```

- **VariableGuardProcessEventListener**

```
public VariableGuardProcessEventListener(String tag, String requiredRole, IdentityProvider
identityProvider) {
    this.tag = tag;
    this.requiredRole = requiredRole;
    this.identityProvider = identityProvider;
}
```

したがって、以下の例に示すように、許可されたロール名とユーザーロールを返す ID プロバイダーを使用して、イベントリスナーをセッションに追加する必要があります。

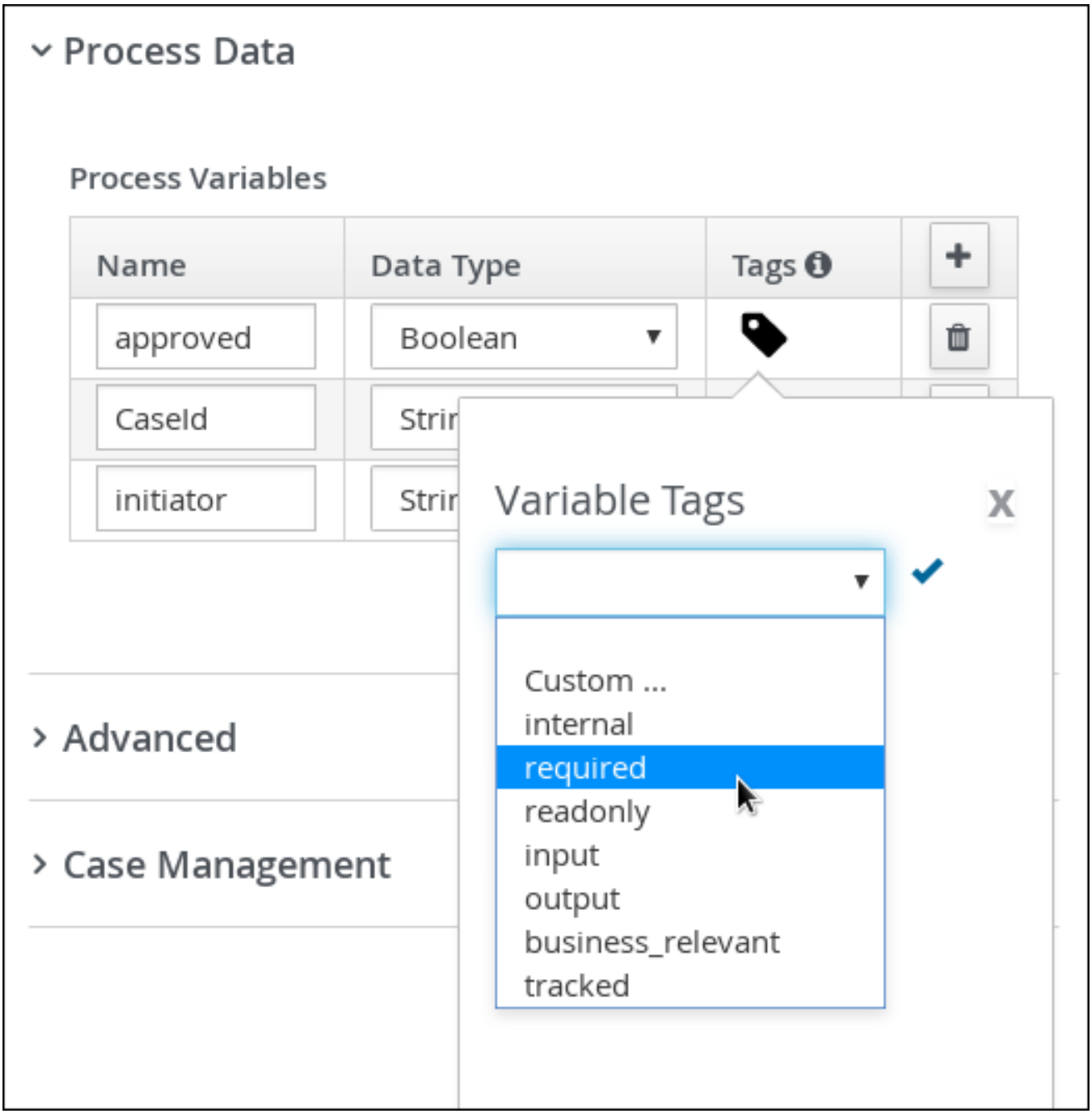
```
ksession.addEventListener(new VariableGuardProcessEventListener("AdminRole",
myIdentityProvider));
```

上記の例では、**VariableGuardProcessEventListener** メソッドで、変数にセキュリティー制約タグ (**restricted**) が付いているかどうかを確認します。必要なロールがユーザーに割り当てられていない場合 (例: **AdminRole**) には、Red Hat Process Automation Manager は **VariableViolationException** エラーを生成します。注記: Business Central UI (例: **internal**、**input**、**output**、**business-relevant** および **tracked**) に表示される変数タグは、Red Hat Process Automation Manager ではサポートされません。

タグは、**![CDATA[TAG_NAME]]** 形式で定義されたタグ値を使用し、**customTags** メタデータプロパティーとして BPMN プロセスソースファイルに直接追加できます。

たとえば、以下の BPMN プロセスは、**required** タグを **approved** プロセス変数に適用します。

図11.1 BPMN モデラーでタグ付けされた変数の例



BPMN ファイルでタグ付けされた変数の例

```
<bpmn2:property id="approved" itemSubjectRef="ItemDefinition_9" name="approved">
  <bpmn2:extensionElements>
    <tns:metaData name="customTags">
      <tns:metaValue><![CDATA[required]]></tns:metaValue>
    </tns:metaData>
  </bpmn2:extensionElements>
</bpmn2:property>
```

必要に応じて、変数に複数のタグを使用できます。BPMN ファイルでカスタム変数タグを定義して、Red Hat Process Automation Manager プロセスイベントリスナーが変数データを利用できるようにすることも可能です。カスタムタグは、標準の変数タグのように Red Hat Process Automation Manager

のランタイムに影響を与えることはせず、情報提供のみを目的としています。カスタム変数タグは、標準の Red Hat Process Automation Manager 変数タグに使用する場合と同じ **customTags** メタデータプロパティー形式で定義します。

第12章 ケースイベントリスナー

CaseEventListener は、ケースインスタンスで呼び出される、ケース関連のイベントやオペレーションの通知を開始するのに使用します。ケースイベントリスナーは、特定のユースケースに対して必要に応じてメソッドを上書きして、実装します。

Menu → Design → $PROJECT_NAME → Settings → Deployments で、Business Central にあるデプロイメント記述子を使用して、リスナーを設定できます。

新規プロジェクトが作成されると、**kie-deployment-descriptor.xml** ファイルがデフォルト値で生成されます。

== CaseEventListener メソッド

```
public interface CaseEventListener extends EventListener {

    default void beforeCaseStarted(CaseStartEvent event) {
    };

    default void afterCaseStarted(CaseStartEvent event) {
    };

    default void beforeCaseClosed(CaseCloseEvent event) {
    };

    default void afterCaseClosed(CaseCloseEvent event) {
    };

    default void beforeCaseCancelled(CaseCancelEvent event) {
    };

    default void afterCaseCancelled(CaseCancelEvent event) {
    };

    default void beforeCaseDestroyed(CaseDestroyEvent event) {
    };

    default void afterCaseDestroyed(CaseDestroyEvent event) {
    };

    default void beforeCaseReopen(CaseReopenEvent event) {
    };

    default void afterCaseReopen(CaseReopenEvent event) {
    };

    default void beforeCaseCommentAdded(CaseCommentEvent event) {
    };

    default void afterCaseCommentAdded(CaseCommentEvent event) {
    };

    default void beforeCaseCommentUpdated(CaseCommentEvent event) {
    };
```

```
    default void afterCaseCommentUpdated(CaseCommentEvent event) {
    };

    default void beforeCaseCommentRemoved(CaseCommentEvent event) {
    };

    default void afterCaseCommentRemoved(CaseCommentEvent event) {
    };

    default void beforeCaseRoleAssignmentAdded(CaseRoleAssignmentEvent event) {
    };

    default void afterCaseRoleAssignmentAdded(CaseRoleAssignmentEvent event) {
    };

    default void beforeCaseRoleAssignmentRemoved(CaseRoleAssignmentEvent event) {
    };

    default void afterCaseRoleAssignmentRemoved(CaseRoleAssignmentEvent event) {
    };

    default void beforeCaseDataAdded(CaseDataEvent event) {
    };

    default void afterCaseDataAdded(CaseDataEvent event) {
    };

    default void beforeCaseDataRemoved(CaseDataEvent event) {
    };

    default void afterCaseDataRemoved(CaseDataEvent event) {
    };

    default void beforeDynamicTaskAdded(CaseDynamicTaskEvent event) {
    };

    default void afterDynamicTaskAdded(CaseDynamicTaskEvent event) {
    };

    default void beforeDynamicProcessAdded(CaseDynamicSubprocessEvent event) {
    };

    default void afterDynamicProcessAdded(CaseDynamicSubprocessEvent event) {
    };
}
```

第13章 ケース管理のルール

ケースは、シーケンスフローには従わず、データ駆動型です。ケースを解決するのに必要な手順は、ケースに関与するユーザーによって提供されます。または、利用可能なデータに基づいてその後のアクションを発生させるようにシステムを設定できます。後者の場合は、ビジネスルールを使用して、ケースを継続させる後続のアクションを決定したり、解決に到達したりできます。

データは、ケース内のどの時点でも、ケースファイルに挿入できます。デシジョンエンジンは常に、ケースファイルデータをモニタリングしているので、ケースファイルに含まれるデータに、ルールが反応します。ルールを使用して、ケースファイルデータでの変化をモニタリングして対応し、ある程度自動的に、ケースが前に進むようにします。

13.1. ルールを使用したケースの前進

Business Central のケース管理に関する **IT_Orders** サンプルプロジェクトを参照します。

業者が入力した特定のハードウェア明細書に誤りがあるか無効だったとしましょう。ケースを続行させるためには、業者が、有効な注文を新たに行う必要があります。マネージャーが無効な明細書を却下して、業者に対して新しい要求を作成する代わりに、入力した明細書が無効であることをケースデータが示したらすぐに応答するビジネスルールを作成できます。このルールは、業者に新しいハードウェア明細書の要求を作成できます。

以下の手順は、このシナリオを実行するビジネスルールを作成して使用する方法を実演します。

前提条件

- **IT_Orders** サンプルプロジェクトが Business Central にインポートされているが、KIE Server にデプロイされていない。
- **ServiceRegistry** は **jbpm-services-api** モジュールに含まれており、クラスパスで利用できるようにする。

注記

Business Central 外でプロジェクトをビルドする場合は、以下の依存関係をプロジェクトに追加する必要があります。

- **org.jbpm:jbpm-services-api**
- **org.jbpm:jbpm-case-mgmt-api**

手順

1. **validate-document.drl** という名前で、以下のビジネスルールファイルを作成します。

```
package defaultPackage;

import java.util.Map;
import java.util.HashMap;
import org.jbpm.casemgmt.api.CaseService;
import org.jbpm.casemgmt.api.model.instance.CaseFileInstance;
import org.jbpm.document.Document;
import org.jbpm.services.api.service.ServiceRegistry;

rule "Invalid document name - reupload"
```

```
when
   $caseData : CaseFileInstance()
   Document(name == "invalid.pdf") from $caseData.getData("hwSpec")

then

   System.out.println("Hardware specification is invalid");
   $caseData.remove("hwSpec");
   update($caseData);
   CaseService caseService = (CaseService)
ServiceRegistry.get().service(ServiceRegistry.CASE_SERVICE);
   caseService.triggerAdHocFragment($caseData.getCaseId(), "Prepare hardware spec",
null);
end
```

このビジネスルールは、**invalid.pdf** ファイルをケースファイルにアップロードしたタイミングを検出します。これは **invalid.pdf** ドキュメントを削除し、**Prepare hardware spec** ユーザータスクの新しいインスタンスを作成します。

2. **Deploy** をクリックして、**IT_Orders** プロジェクトをビルドし、KIE Server にデプロイします。

注記

Build & Install オプションを選択してプロジェクトをビルドし、KJAR ファイルを KIE Server にデプロイせずに設定済みの Maven リポジトリーに公開することもできます。開発環境では、**Deploy** をクリックすると、ビルドされた KJAR ファイルを KIE Server に、(該当する場合) 実行中のインスタンスを停止せずにデプロイできます。または **Redeploy** をクリックして、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もできます。ビルドされた KJAR ファイルを次回に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 コンテナー) 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 **Redeploy** オプションは無効になっており、**Deploy** をクリックして、ビルドされた KJAR ファイルを KIE Server 上の新規デプロイメントユニット (KIE コンテナー) にデプロイすることのみが可能です。

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ーを **org.kie.server.mode=development** または **org.kie.server.mode=production** に設定します。Business Central でそれぞれのプロジェクトのデプロイメント動作を設定するには、プロジェクトの **Settings → General Settings → Version** に移動し、**Development Mode** オプションを選択します。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。**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。

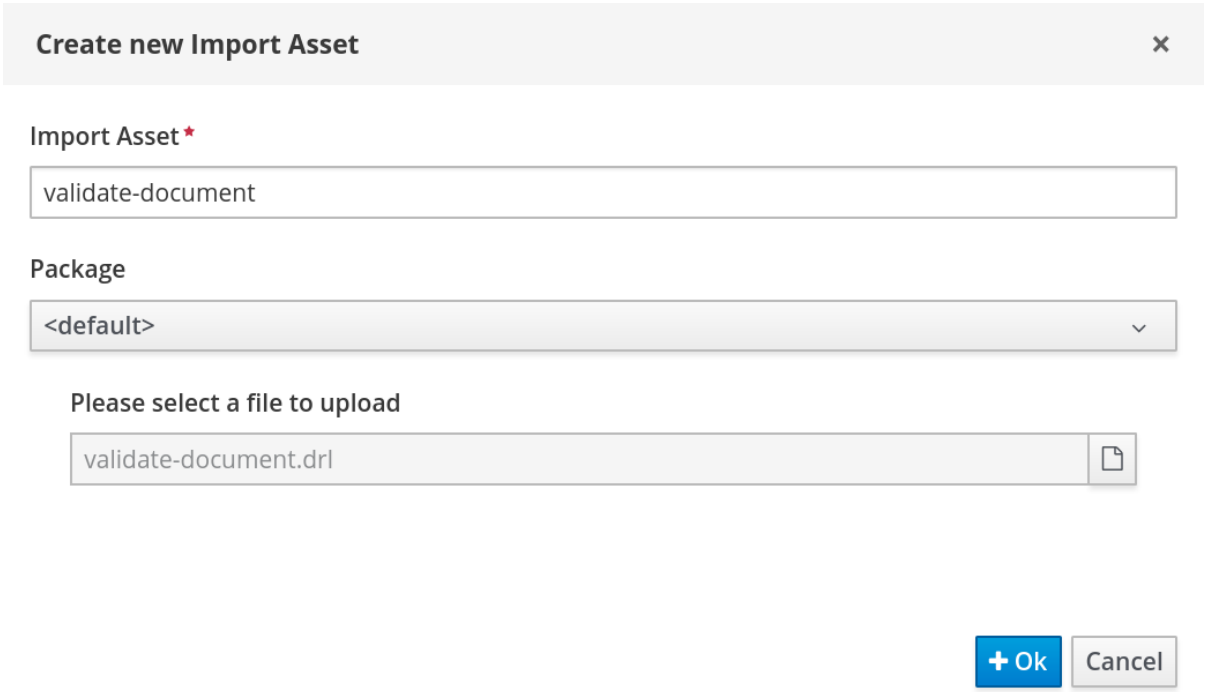
3. **invalid.pdf** ファイルを作成し、ローカルで保存します。

4. **valid-spec.pdf** ファイルを作成し、ローカルで保存します。

5. Business Central で、**Menu → Projects → IT_Orders** に移動して、**IT_Orders** プロジェクトを開きます。

6. ページの右上で **Import Asset** をクリックします。

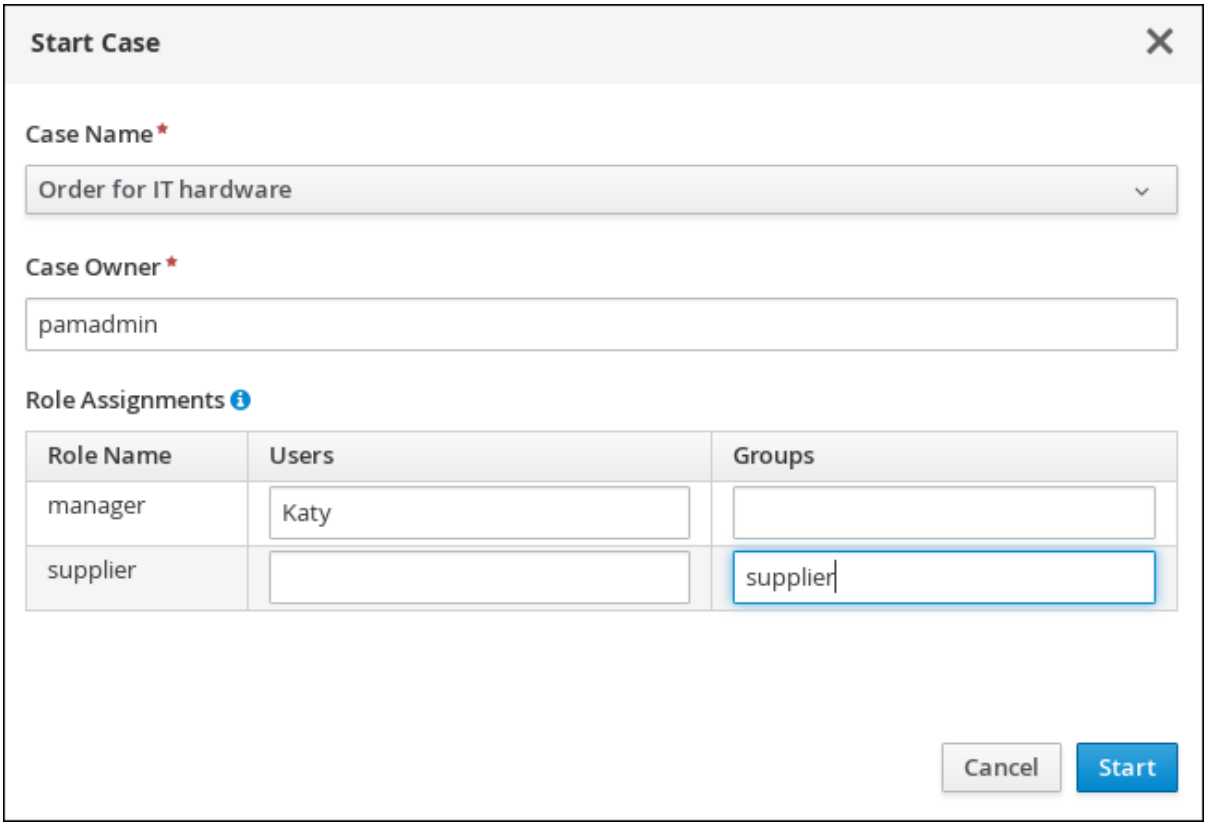
7. **validate-document.drl** ファイルを **default** パッケージ (**src/main/resources**) にアップロードし、**Ok** をクリックします。



validate-document.drl ルールがルールエディターに表示されます。**Save** をクリックするか、ルールエディターを閉じて終了します。

8. **Apps launcher** (インストールされている場合) をクリックするか、**/http://localhost:8080/rhpam-case-mgmt-showcase/jbpm-cm.html** に移動して、Showcase アプリケーションを開きます。

9. **IT_Orders** の **Start Case** をクリックします。
この例では、Aimee がケースの **owner**、Katy が **manager**、および業者グループが **supplier** になります。

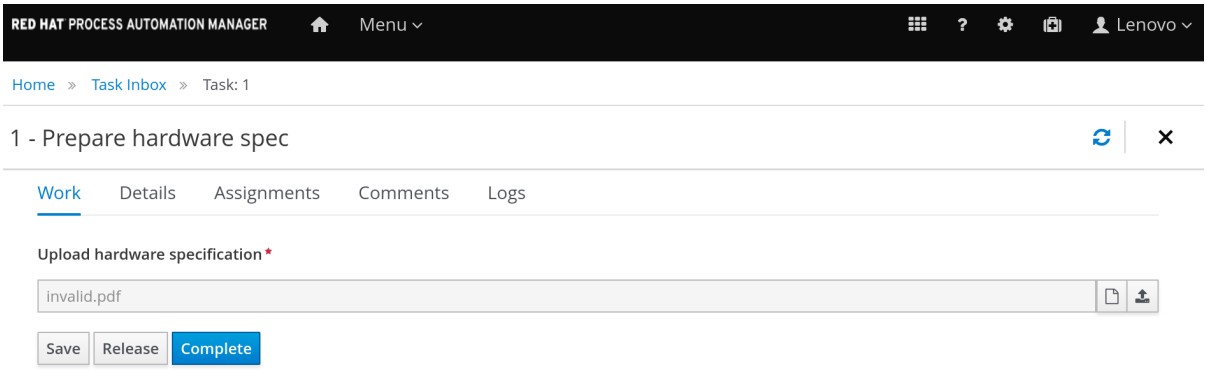


10. Business Central からログアウトし、**supplier** グループに属するユーザーでログインし直します。

11. **Menu → Track → Task Inbox** に移動します。

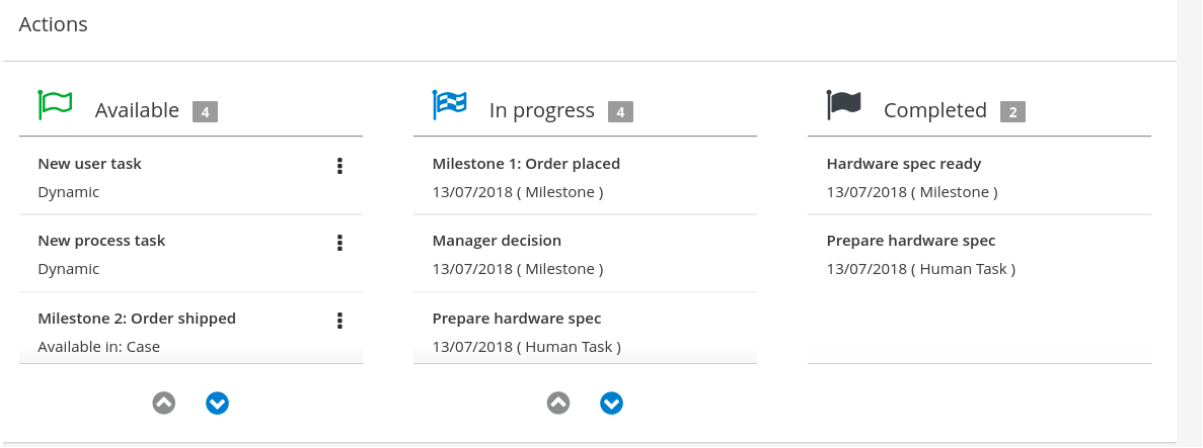
12. **Prepare hardware spec** タスクを開いて、**Claim** をクリックします。これにより、タスクがログインユーザーに割り当てられます。

13. **Start** をクリックして  をクリックし、 **invalid.pdf** ハードウェア仕様ファイルの場所を特定します。  をクリックしてファイルをアップロードします。



14. **Complete** をクリックします。
Prepare hardware spec の Task Inbox の値が **Ready** になります。

15. Showcase で、右上隅の **Refresh** をクリックします。**Prepare hardware task** メッセージが **Completed** の列と、**In Progress** の列に表示されていることを確認します。



これは、最初の **Prepare hardware spec** タスクが明細書ファイル **invalid.pdf** を使用して完了されているためです。その結果、ビジネスルールによりタスクおよびファイルが破棄され、新しいユーザータスクが作成されます。

16. Business Central の **Task Inbox** で手順 12 および 13 を繰り返し、 **invalid.pdf** の代わりに **valid-spec.pdf** ファイルをアップロードします。

第14章 ケース管理のセキュリティー

ケースは、ケースロールを使用して、ケース定義レベルで設定されます。これは、ケース処理に関与する一般的な参加者です。このようなロールはユーザータスクに割り当てられるか、連絡先参照として使用されます。ロールは、特定のユーザーまたはグループにはハードコードされず、ケース定義を、指定したケースインスタンスに関与する実際のアクターとは独立させます。ケースインスタンスがアクティブである限り、いつでもケースロール割り当てを修正できます。ただし、ロール割り当ては、以前のロール割り当てに基づいて作成されているタスクには影響を及ぼしません。

ケースインスタンスのセキュリティーはデフォルトで有効になります。ケース定義を使用して、対象のケースに属していないユーザーがケースデータにアクセスできないようにします。ユーザーにケースロール割り当て (ユーザーまたはグループメンバーとしての割り当て) がない場合は、ケースインスタンスにアクセスできません。

ケースのセキュリティーは、ケースインスタンスを開始する場合に、ケースロールを割り当てることが推奨される理由の 1 つです。これにより、ケースにアクセスできないユーザーにタスクが割り当てられないようにします。

14.1. ケース管理のセキュリティーの設定

以下のシステムプロパティーを **false** に設定すると、ケースインスタンスの認証を無効にできます。

org.jbpm.cases.auth.enabled

システムプロパティーは、ケースインスタンスに対するセキュリティーコンポーネントの 1 つでしかありません。さらに、**case-authorization.properties** ファイルを使用して実行サーバーレベルでケース操作を設定できます。これは、実行サーバーアプリケーションのクラスパスの root (**kie-server.war/WEB-INF/classes**) で利用できます。

可能なすべてのケース定義に簡易な設定ファイルを使用すると、ケース管理はドメイン固有のものとして見なされます。ケースセキュリティーの **AuthorizationManager** はプラグ可能で、特定のセキュリティー処理にカスタムコードを追加できます。

以下のケースインスタンス操作をケースロールに制限できます。

- **CANCEL_CASE**
- **DESTROY_CASE**
- **REOPEN_CASE**
- **ADD_TASK_TO_CASE**
- **ADD_PROCESS_TO_CASE**
- **ADD_DATA**
- **REMOVE_DATA**
- **MODIFY_ROLE_ASSIGNMENT**
- **MODIFY_COMMENT**

前提条件

- Red Hat Process Automation Manager の KIE Server が実行されていない。

手順

1. 任意のエディターで、**JBOSS_HOME/standalone/deployments/kie-server.war/WEB-INF/classes/case-authorization.properties** ファイルを開きます。
   デフォルトでは、ファイルには以下の操作制限が含まれます。

```
CLOSE_CASE=owner,admin
CANCEL_CASE=owner,admin
DESTROY_CASE=owner,admin
REOPEN_CASE=owner,admin
```

2. この操作に対するロールパーミッションを追加または削除します。

   a. ロールが操作を実行するためのパーミッションを削除するには、**case-authorization.properties** ファイルでその操作に対して認証されているロールの一覧から削除します。たとえば、**admin** ロールを **CLOSE_CASE** 操作から削除すると、すべてのケースで、そのケース所有者に対してケースを閉じるパーミッションに制限がかかります。

   b. ケース操作を実行するロールパーミッションを付与するには、**case-authorization.properties** ファイルでその操作に対して承認されているロールの一覧に追加します。たとえば、**manager** ロールがあれば **CLOSE_CASE** 操作を実行できるように許可する場合は、ロールの一覧に、コンマ区切りのロールを追加できます。
   CLOSE_CASE=owner,admin,manager

3. ファイルに挙げられているその他のケース操作にロール制限を追加するには、行から **#** を削除し、以下の形式でロール名を一覧表示します。
   OPERATION=role1,role2,roleN

   ファイル内で **#** で始まる操作の制限は無視され、ケースに関与するユーザーは誰でも実行できます。

4. ロールパーミッションの割り当てを終了したら、**case-authorization.properties** ファイルを保存して閉じます。

5. 実行サーバーを開始します。
   ケース認証設定は、実行サーバーのすべてのケースに適用します。

第15章 ケースの終了

ケースインスタンスは、実行するアクティビティーがなくなったかビジネスゴールに達成した場合に完了、または完全に閉じることができます。通常は、すべての作業が完了してケースゴールを満たした場合に、ケースの所有者がケースを閉じますが、ケースを閉じる時に、ケースインスタンスを閉じる理由をコメントとして追加することもできます。

必要に応じて、後から閉じたケースを同じケース ID を使用して再開することができます。ケースの再開時には、ケースを閉じた時にアクティブなステージが有効になります。

KIE Server REST API 要求を使用してケースインスタンスをリモートで、または Showcase アプリケーションで直接閉じることができます。

15.1. KIE SERVER REST API を使用したケースの終了

REST API 要求を使用してケースインスタンスを閉じることができます。Red Hat Process Automation Manager には Swagger クライアントが含まれており、REST API 要求のエンドポイントやドキュメントが提供されます。もしくは、サンプルエンドポイントで、任意のクライアントや Curl を使用して API コールを作成できます。

前提条件

- Showcase を使用して、ケースインスタンスを開始している。
- **admin** ロールを持つユーザーとして、API 要求を認証できる。

手順

1. Web ブラウザーで Swagger REST API クライアントを開きます。
   /http://localhost:8080/kie-server/docs
2. **Case Instances :: Case Management** の下で、以下のエンドポイントで **POST** リクエストを開きます。
   /server/containers/{id}/cases/instances/{caseId}
3. **Try it out** をクリックし、必要なパラメーターを入力します。

   表15.1 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**

4. 任意で、ケースファイルに含まれるコメントを追加できます。コメントを残す場合は、**body** テキストフィールドに **文字列** で入力します。
5. **Execute** をクリックして、ケースを閉じます。
6. ケースが閉じたことを確定するには、Showcase アプリケーションを開いて、ケースリストのステータスを **Closed** に変更します。

15.2. SHOWCASE アプリケーションを使用したケースの終了

ケースインスタンスは、実行するアクティビティーがなくなり、ビジネス目標が達成されたら終了します。ケースが完了したら、ケースを終了して、ケースの完了と、作業の必要性がないことを示します。ケースを終了したら、ケースの終了理由について具体的なコメントを追加することを検討してみてください。必要な場合には、同じケース ID を使用して、対象のケースをもう一度、開くことができます。

Showcase アプリケーションを使用していつでもケースを閉じることができます。Showcase から、簡単にケースの詳細を表示したり、閉じる前にコメントを残したりできます。

前提条件

- Showcase アプリケーションにログインしていること。また、終了するケースインスタンスの所有者か、管理者であること。

手順

1. Showcase アプリケーションで、ケースインスタンスの一覧から閉じるケースインスタンスを見つけます。
2. 詳細を確認せずに先にケースを終了するには、**Close** をクリックします。
3. ケース詳細ページからケースを閉じるには、リストでケースをクリックして、開きます。ケースの概要ページで、ケースにコメントを追加して、ケース情報に基づいて正しいケースを閉じていることを確認します。
4. **Close** をクリックして、ケースを閉じます。
5. ページ右上で **Back to Case List**をクリックし、Showcase ケース一覧ビューに戻ります。
6. **Status** の横にあるドロップダウンリストをクリックし、**Canceled** をクリックすると、閉じたケースおよびキャンセルしたケースの一覧が表示されます。

第16章 ケースのキャンセルまたは破棄

ケースが必要なくなったり、ケース作業を実行する必要がなくなった場合は、ケースをキャンセルできます。キャンセルしたケースは、その後も同じケースインスタンス ID とケースファイルデータを使用して再開できます。場合によっては、ケースを再開できないようにケースを永続的に破棄することもできます。

ケースのキャンセルまたは破棄は、API 要求からしかできません。Red Hat Process Automation Manager には Swagger クライアントが含まれており、REST API 要求のエンドポイントやドキュメントが提供されています。もしくは、サンプルエンドポイントで、任意のクライアントや Curl を使用して API コールを作成できます。

前提条件

- Showcase を使用して、ケースインスタンスを開始している。
- **admin** ロールを持つユーザーとして、API 要求を認証できる。

手順

1. Web ブラウザーで Swagger REST API クライアントを開きます。
   /http://localhost:8080/kie-server/docs
2. **Case Instances :: Case Management** の下で、以下のエンドポイントで **DELETE** リクエストを開きます。
   /server/containers/{id}/cases/instances/{caseId}

   DELETE リクエストを使用してキャンセルできます。任意で、**destroy** パラメーターを使用してケースを破棄することもできます。
3. **Try it out** をクリックし、必要なパラメーターを入力します。

   表16.1 パラメーター

名前	説明
id	**itorders**
caseId	**IT-0000000001**
destroy	**true** (任意。永続的にケースを破壊します。このパラメーターはデフォルトで **false** です)。

4. **Execute** をクリックして、ケースをキャンセル (または破棄) します。
5. ケースのキャンセルを確定するには、Showcase アプリケーションを開いて、ケースリストのステータスを **Canceled** にします。ケースが破棄されていると、ケースリストには表示されません。

16.1. データベースからのケースログの削除

CaseLogCleanupCommand を使用して、データベースのスペースを占有しているキャンセル済みのケースなど、ケースを消去します。**CaseLogCleanupCommand** コマンドには、選択したケースまたは、全ケースを自動的に消去するロジックが含まれています。

CaseLogCleanupCommand コマンドと合わせて使用可能な設定オプションが複数あります。

表16.2 CaseLogCleanupCommand パラメーターテーブル

名前	説明	排他的
SkipProcessLog	コマンドの実行時に、プロセスやノードインスタンス、プロセス変数ログの消去を省略するかどうかを指定します。デフォルト値: **false**	いいえ、他のパラメーターと併用できる
SkipTaskLog	コマンドの実行時に、タスク監査、タスクイベント、タスク変数ログの消去を省略するかどうかを指定します。デフォルト値: **false**	いいえ、他のパラメーターと併用できる
SkipExecutorLog	コマンドの実行時に、Red Hat Process Automation Manager エグゼキューターのエントリー消去を省略するかどうかを指定します。 デフォルト値: **false**	いいえ、他のパラメーターと併用できる
SingleRun	ジョブルーチンを 1 回だけ実行するかどうかを指定します。デフォルト値: **false**	いいえ、他のパラメーターと併用できる
NextRun	次のジョブ実行をスケジュールします。たとえば、12時間ごとにジョブを実行するには、**12h** と指定します。**SingleRun** が **true** にされており、**SingleRun** と **NextRun** 両方が設定されていない場合は、このスケジュールは無視されます。両方が設定されている場合には、**NextRun** のスケジュールが優先されます。正確な日付を設定するには、ISO 形式を使用できます。デフォルト値: **24h**	いいえ、他のパラメーターと併用できる
OlderThan	指定の日付より古いログを削除します。日付の形式は、**YYYY-MM-DD** です。通常、このパラメーターは単一のジョブ実行に使用します。	はい。**OlderThanPeriod** パラメーターを使用する場合にはこのパラメーターは使用できません。

名前	説明	排他的
OlderThanPeriod	指定のタイマー式より古いログを削除します。たとえば、30 日が経過したログを削除するには、30d を設定します。	はい。**OlderThan** パラメーターを使用する場合には使用できません。
ForCaseDefId	削除するログのケース定義 ID を指定します。	いいえ、他のパラメーターと併用できる
ForDeployment	削除するログのデプロイメント ID を指定します。	いいえ、他のパラメーターと併用できる
EmfName	削除操作の実行に使用する永続ユニット名。デフォルト値: **org.jbpm.domain**	該当なし
DateFormat	時間関連のパラメーターの日付形式を指定します。デフォルト値: **yyyy-MM-dd**	いいえ、他のパラメーターと併用できる
ステータス	削除されたログのケースインスタンスのステータス	いいえ、他のパラメーターと併用できる

第17章 関連資料

- 『ケース管理の使用ガイド』
- 『ケース管理への Showcase アプリケーションの使用』

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)