Red Hat Process Automation Manager 7.7

Business Central でのビジネスプロセスの設計

Last Updated: 2020-03-31

Red Hat Process Automation Manager 7.7 Business Central でのビジネスプロセスの設計

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

法律上の通知

概要

このドキュメントでは、Red Hat Process Automation Manager 7.7 でビジネスプロセスを設計するための概念とオプションについて説明します。

目次

前書き

ビジネスプロセス開発者は、Red Hat Process Automation Manager の Business Central を使用して、特定のビジネス要件を満たすビジネスプロセスを設計できます。このドキュメントでは、ビジネスプロセスと、Red Hat Process Automation Manager のプロセスデザイナーを使用してビジネスプロセスを作成するための概念とオプションについて説明します。また、Red Hat Process Automation Manager の BPMN2 要素についても説明しています。BPMN2 の詳細については、Business Process Model and Notation Version 2.0 仕様を参照してください。

前提条件

- Red Hat JBoss Enterprise Application Platform 7.2 がインストールされている。詳細情報は『**Red Hat JBoss EAP 7.2 インストールガイド**』を参照してください。
- Red Hat Process Automation Manager がインストールされ、KIE Server で設定されている。詳細は、『**Red Hat JBoss EAP 7.2 への Red Hat Process Automation Manager のインストールおよび設定**』を参照してください。
- Red Hat Process Automation Manager が稼働し、**developer** ロールで Business Central にログインできる。詳細は、『**Red Hat Process Automation Manager インストールの計画**』を参照してください。

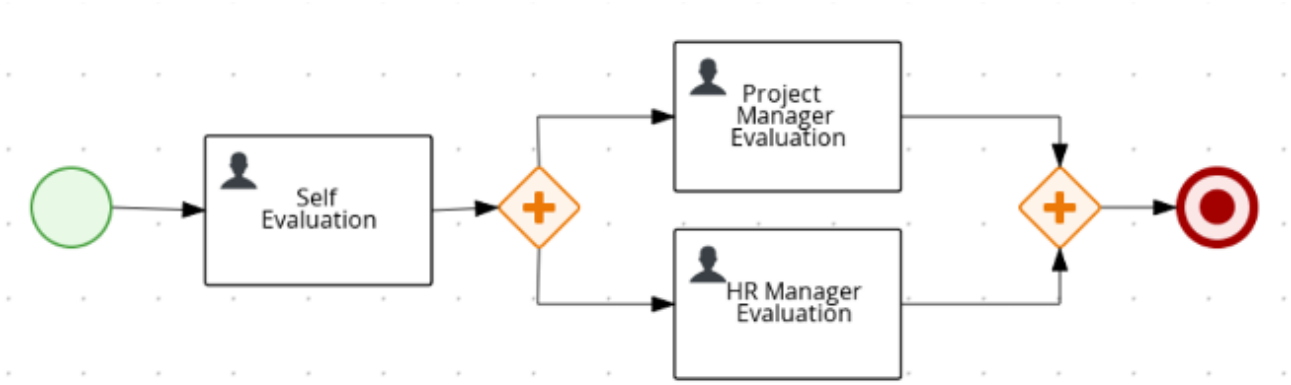
第1章 ビジネスプロセス

ビジネスプロセスとは、一連の手順の実行すべき順番を説明し、事前定義済みのノードや接続で構成されるダイアグラムのことです。各ノードは、プロセス内の手順 1 つを表し、接続はノード間の移動の方法を指定します。

典型的なビジネスプロセスは、以下のコンポーネントで構成されています。

- プロセスの名前、インポート、変数などのグローバル要素で構成されるヘッダーセクション
- プロセスの一部であるすべての異なるノードを含むノードセクション
- これらのノードを相互にリンクしてフローチャートを作成する接続セクション

図1.1 ビジネスプロセス



Red Hat Process Automation Manager には、レガシーのプロセスデザイナーと、ビジネスプロセスダイアグラムを作成するための新しいプロセスデザイナーが含まれています。新しいプロセスデザイナーのレイアウトと機能セットは改善されており、開発が続けられています。レガシーのプロセスデザイナーのすべての機能が新しいプロセスデザイナーに完全に実装されるまで、お客様は両方のデザイナーを Business Central で使用できます。

注記

Business Central のレガシーのプロセスデザイナーは、Red Hat Process Automation Manager 7.7.0 で非推奨になりました。これは、Red Hat Process Automation Manager の今後のリリースでは削除される予定です。そのため、レガシーのプロセスデザイナーには新しい機能拡張や機能は追加されません。新しいプロセスデザイナーを使用する場合は、お使いのプロセスを新しいデザイナーに移行し始め、新しいプロセスデザイナーですべての新規プロセスを作成してください。新規デザイナーへの移行に関する詳細は、『Business Central におけるプロジェクトの管理』を参照してください。

第2章 BUSINESS PROCESS MODELING AND NOTATION バージョン 2.0

Business Process Modeling and Notation バージョン 2.0 (BPMN2) 仕様は、ビジネスプロセスを描画表現するための標準や要素の実行セマンティクスの定義、XML 形式でのプロセス定義を行うときに使用する Object Management Group (OMG) 仕様です。

プロセスは、定義するか、プロセス定義を使用して判断します。また、プロセスはナレッジベースに存在しており、ID で識別し、モデル化した要素を格納するコンテナーの役割も果たします。プロセスには、フローオブジェクトやフローを使用してビジネスプロセスの実行ワークフローや、パーツを指定する要素が含まれます。プロセスごとに、独自の BPMN2 のダイアグラムが割り当てられます。Red Hat Process Automation Manager には、BPMN2 ダイアグラムを作成するためのレガシーおよび新規プロセスデザイナーが含まれます。新規プロセスデザイナーでは、レイアウトと機能セットが向上され、今後も開発が続けられる予定です。レガシーのプロセスデザイナーの全機能が完全に新規プロセスデザイナーに実装されるまで、Business Central でどちらのデザイナーもご利用いただけます。

2.1. BPMN2 向けの RED HAT PROCESS AUTOMATION MANAGER サポート

Red Hat Process Automation Manager では、BPMN 2.0 標準を使用して、ビジネスプロセスのモデル化が可能です。Red Hat Process Automation Manager を使用してこれらのビジネスプロセスを実行、管理、監視することができます。包括的な BPMN 2.0 仕様には、コレオグラフィーやコラボレーションなどの項目の表現方法に関する詳細が含まれます。ただし、Red Hat Process Automation Manager は、実行可能なプロセスの指定に使用可能な仕様の一部のみを使用します。これには、BPMN2 仕様の共通の実行可能なサブクラスに定義されているほぼすべての要素および属性だけでなく、追加の要素や属性も含まれます。

以下の表は、BPMN2 要素がレガシーのプロセスデザイナーでサポートされているか、レガシーおよび新規プロセスデザイナーでサポートされているか、またはサポートされていないかを示すアイコン一覧を示しています。

表2.1 サポートの状態を示すアイコン

キー	説明
	レガシーおよび新規プロセスデザイナーでのサポート
	レガシーのプロセスデザイナーでのみサポート
	サポートなし

アイコンのない要素は、BPMN2 仕様には存在しません。

表2.2 BPMN2 の Catch イベント

要素名	始端	中間
なし		
メッセージ		
タイマー		
エラー		
エスカレーション		
取り消し		
補正		
条件		
リンク		
シグナル		
多重		
並列多重		

表2.3 BPMN2 送出および中断なしイベント

要素名	送出		中断なし	
	終端	中間	始端	中間
なし	✓			
メッセージ	✓	✓	✓	✓
タイマー			✓	✓
エラー	✓			
エスカレーション	✓	✓	✓	✓
取り消し	✗	✗		✗
補正	✓	✓		
条件			✓	✓
リンク		✗		
シグナル	✓	✓	✓	✓
中断	✓			

要素名	送出		中断なし	
多重	✗	✗	✗	✗
並列多重			✗	✗

表2.4 BPMN2 要素

要素タイプ	要素	対応
タスク	ビジネスルール	✓
	スクリプト	✓
	ユーザータスク	✓
	サービスタスク	✓
複数のインスタンスサブプロセスを含むサブプロセス	埋め込み	✓
	アドホック	✓
	再利用可能	✓
	イベント	✓

要素タイプ	要素	対応
ゲートウェイ	包含的	
	排他的	
	並列	
	イベントベース	
	コンプレックス	
接続オブジェクト	シーケンスフロー	
	関連フロー	
スイムレーン	スイムレーン	
アーティファクト	グループ	
	テキストのアノテーション	

BPMN2 の背景およびアプリケーションに関する詳細は、OMG Business Process Model and Notation (BPMN) Version 2.0 を参照してください。

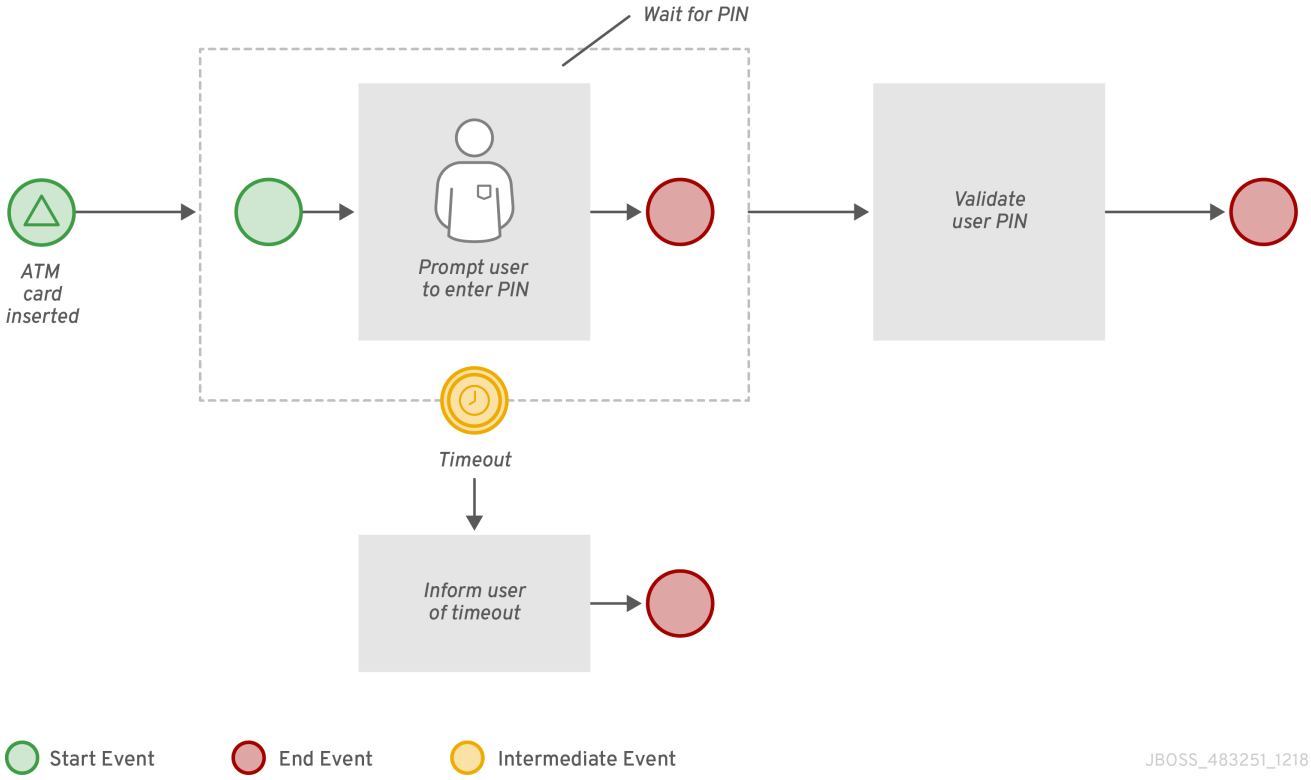
第3章 プロセスデザイナーでの BPMN2 イベント

イベントとは、ビジネスプロセスに発生する内容のことです。BPMN2 では、3 つのイベントカテゴリーをサポートします。

- 始端
- 終端
- 中間

開始イベントはイベントトリガーをキャッチし、終了イベントはイベントトリガーをスローします。中間イベントは、イベントトリガーをキャッチおよびスローできます。

以下のビジネスプロセスダイアグラムでは、イベントの例を紹介します。



この例では、以下のイベントが発生します。

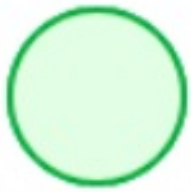
- シグナルを受信すると、ATM カード挿入のシグナル開始イベントがトリガーされます。
- タイムアウトの中間イベントは、タイマートリガーをもとにした割り込みイベントです。つまり、タイマーイベントがトリガーされると、暗証番号待ちのサブプロセスがキャンセルされるという意味です。
- プロセスへの入力に応じて、Validate User Pin タスクに関連付けられた終了イベント、または Inform User of Timeout タスクに関連付けられた終了イベントが、プロセスを終了します。

3.1. 開始イベント

開始イベントを使用して、ビジネスプロセスの始端を示します。開始イベントには、受信シーケンスフローを割り当てることができず、外向きシーケンスフローだけを割り当てる必要があります。「なし (none)」開始イベントは、トップレベルプロセス、埋め込みサブプロセス、呼び出し可能サブプロセス、イベントサブプロセスで使用できます。

「なし (none)」開始イベントの例外を除けば、開始イベントはすべてキャッチイベントです。たとえば、シグナルの開始イベントは、参照のシグナル (イベントトリガー) を受信した場合にのみプロセスを開始します。イベントサブプロセスの開始イベントを割り込みまたは割り込みなしイベントに設定できます。イベントサブプロセスに対する割り込みありの開始イベントでは、包含プロセスまたは親プロセスの実行を停止または中断します。割り込みなしの開始イベントは、包含プロセスまたは親プロセスの実行を停止したり、中断したりしません。

表3.1 開始イベント

開始イベントタイプ	トップレベル	サブプロセス	
		割り込み	割り込みなし
なし			
条件			
補正			
エラー			
エスカレーション			

開始イベントタイプ	トップレベル	サブプロセス	
メッセージ			
シグナル			
タイマー			

なし

「なし」開始イベントは、トリガー条件のない開始イベントです。プロセスまたはサブプロセスには、「なし」開始イベントを最大1つ含めることができます。このイベントは、デフォルトでプロセスまたはサブプロセスの開始によりトリガーされ、外向きフローがすぐに実行されます。

サブプロセスで「なし」開始イベントを使用すると、親プロセスからサブプロセスに、プロセスフローの実行が移動し、「なし」開始イベントがトリガーされます。これは、トークン (プロセスフローの内の現在の場所) が親プロセスからサブプロセスのアクティビティーに渡され、サブプロセスの「なし」開始イベントが独自のトークンを生成します。

条件

条件付きの開始イベントは、ブール型の条件定義を含む開始イベントです。条件が最初に **false** と評価され、次に **true** に評価された場合に実行がトリガーされます。プロセスの実行は、開始イベントがインスタンス化された後に条件が **true** と評価された場合にのみ開始されます。

プロセスには、複数の条件開始イベントを含めることができます。

補正

補正開始イベントは、補正中間イベントのターゲットアクティビティーとして、サブプロセスを使用した場合に、補正イベントのサブプロセスを開始するのに使用します。

エラー

プロセスまたはサブプロセスには、複数のエラー開始イベントを含めることができます。特定の **ErrorRef** プロパティーを含むエラーオブジェクトを受信した場合に、この開始イベントがトリガーされます。エラーオブジェクトは、エラーの終了イベントで生成可能です。これは、プロセスの終端が不正であることを示します。エラーの開始イベントが含まれるプロセスインスタンスは、該当するエラーオブジェクトの受信後に実行が開始されます。エラー開始イベントは、エラーオブジェクトの受信直後に実行されて、外向きフローが実行されます。

エスカレーション

エスカレーション開始イベントは、特定のエスカレーションコードを含むエスカレーションによりトリガーされる開始イベントです。プロセスには、複数のエスカレーション開始イベントを含めることができます。エスカレーション開始イベントが含まれるプロセスインスタンスは、定義されているエスカレーションオブジェクトを受信すると、実行を開始します。プロセスがインスタンス化され、直後にエスカレーション開始イベントが実行され、外向きフローが実行されます。

メッセージ

プロセスまたはイベントのサブプロセスには、複数の「メッセージ」開始イベントを含めることができます。これらのイベントは、通常特定のメッセージによりトリガーされます。「メッセージ」開始イベントが含まれるプロセスインスタンスの実行は、該当のメセージを受信した後に、このイベントからのみ開始されます。メッセージの受信後に、プロセスはインスタンス化され、「メッセージ」開始イベントが即座に実行されます (外向きフローが実行されます)。

メッセージは、要素なしなど、任意の数のプロセスおよびプロセス要素により消費可能であるため、1つのメッセージで複数の「メッセージ」開始イベントをトリガーできるので、複数のプロセスがインスタンス化されます。

シグナル

シグナル開始イベントは、特定のシグナルコードを含むシグナルによりトリガーされます。プロセスには、複数のシグナル開始イベントを含むことができます。シグナル開始イベントは、インスタンスが該当のシグナルを受信した後にのみ、プロセスインスタンス内で実行を開始します。その後に、シグナル開始イベントが実行され、外向きフローが実行されます。

タイマー

タイマー開始イベントは、タイミングのメカニズムを含む開始イベントです。プロセスには、複数のタイマー開始イベントを含めることができます。タイマー開始イベントは、タイミングのメカニズムが適用された後に、プロセスの開始時にトリガーされます。

サブプロセスでタイマー開始イベントを使用すると、プロセスフローの実行が親プロセスからサブプロセスに移動し、タイマー開始イベントがトリガーされます。親サブプロセスアクティビティーからトークンを取得し、サブプロセスのタイマー開始イベントがトリガーされ、タイマーがトリガーされるまで待機します。タイミングの定義で指定した時間が経過したら、外向きフローが実行されます。

3.2. 中間イベント

中間イベントは、ビジネスプロセスのフローを駆動します。中間イベントは、ビジネスプロセスの実行中にイベントをキャッチまたはスローするときに使用します。中間イベントは、開始イベントと終了イベントの間に配置され、サブプロセス、人間のタスクなどのアクティビティーの境界にキャッチイベントとして使用することもできます。境界キャッチイベントは、割り込みまたは割り込みなしとして設定できます。割り込みありの境界キャッチイベントは、バインドされているアクティビティーを取り消しますが、割り込みなしのイベントは取り消しません。

中間イベントは、プロセス実行時に発生する特定の状況を処理します。このような状況が中間イベントのトリガーになります。プロセスには、外向きフローを1つ含む中間イベントを、アクティビティーの境界に配置できます。

アクティビティーの実行時にイベントが発生した場合には、このイベントにより、外向きフローへの実行がトリガーされます。1つのアクティビティーに、複数の境界中間イベントが含まれる可能性があります。境界中間イベントで、アクティビティーから必要な動作によって、以下のいずれかの中間イベントタイプを使用できる点に注意してください。

- 割り込みあり: アクティビティーの実行は中断され、中間イベントの実行がトリガーされます。
- 割り込みなし: 中間イベントがトリガーされ、アクティビティーの実行が続行されます。

表3.2 中間イベント

中間イベントタイプ	キャッチ	境界		送出
		割り込み	割り込みなし	
メッセージ				
タイマー				
エラー				
シグナル				
条件				

中間イベントタイプ	キャッチ	境界		送出
補正				
エスカレーション				

メッセージ

メッセージの中間イベントは、メッセージオブジェクトを管理可能にする中間イベントです。以下のイベントのいずれかを使用します。

- スローメッセージの中間イベントでは、定義したプロパティーをもとにメッセージオブジェクトを作成します。
- キャッチメッセージの中間イベントは、定義したプロパティーを使用してメッセージオブジェクトがないかリッスンします。

タイマー

タイマー中間イベントでは、ワークフローの実行を遅延させたり、定期的にトリガーしたりできます。このイベントは、指定した期間が経過したら 1 回または複数回、トリガーできるタイマーを表します。タイマー中間イベントがトリガーされたら、タイマー条件 (定義した時間) がチェックされ、外向きフローが実行されます。タイマー中間イベントがプロセスワークフローに配置されている場合には、内向きフローが 1 つと、外向きフローが 1 つ含まれます。内向きフローがイベントに移動すると、これが実行されます。タイマー中間イベントがアクティビティー境界に配置されている場合には、アクティビティーの実行と同時に、この実行がトリガーされます。

包含のプロセスインスタンスを完了するか、中断するなど、タイマー要素がキャンセルされると、タイマーがキャンセルされます。

条件

条件の中間イベントは、ブール型の条件がトリガーとして含まれる中間イベントです。このイベントは、条件で **true** と判断され、外向きフローが実行された場合に、さらにワークフロー実行をトリガーします。

このイベントは、**Expression** プロパティーを定義する必要があります。条件の中間イベントがプロセスワークフローに配置されている場合は、内向きフロー 1 つ、外向きフロー 1 つが含まれ、内向きフローがイベントに移動したときに、実行が開始されます。条件の中間イベントがアクティビティー境界

に配置されている場合には、アクティビティーの実行時に、この実行がトリガーされます。イベントが割り込みなしの場合には、条件が **true** の場合は継続して、このイベントがトリガーされます。

シグナル

シグナルの中間イベントでは、シグナルオブジェクトを生成または消費できます。以下のオプションのいずれかを使用してください。

- スローシグナルの中間イベントは、定義したプロパティーをもとにシグナルオブジェクトを生成します。
- キャッチシグナルの中間イベントは、定義したプロパティーを使用してシグナルオブジェクトがないかリッスンします。

エラー

エラーの中間イベントは、アクティビティー境界でのみ使用可能な中間イベントです。このイベントでは、プロセスが、該当するアクティビティー内のエラー終了イベントに反応できるようになります。このアクティビティーは、アトミックであってはいけません。アクティビティーが、エラー終了イベントで完了し、対応の **ErrorCode** プロパティーでエラーオブジェクトを生成した場合には、エラーの中間イベントがこのエラーオブジェクトをキャッチして、実行が外向きフローに進みます。

補正

補正中間イベントは、トランザクションサブプロセスのアクティビティーに接続されている境界イベントです。補正終了イベントまたはキャンセル終了イベントで、このイベントを終了できます。補正中間イベントは、補正アクティビティーに接続されているフローと関連付ける必要があります。

境界補正の仲介イベントに関連付けられているアクティビティーは、トランザクションサブプロセスが補正終了イベントで終了した場合に実行されます。この実行は、対応のフローで続行されます。

エスカレーション

エスカレーション中間イベントは、エスカレーションオブジェクトを生成または消費できる中間イベントです。イベント要素が実行すべきアクションに合わせて、以下のオプションのいずれかを使用する必要があります。

- スローエスカレーションの中間イベントは、定義したプロパティーをもとにエスカレーションオブジェクトを生成します。
- キャッチエスカレーションの中間イベントは、定義したプロパティーを使用してエスカレーションオブジェクトがないかリッスンします。

3.3. 終了イベント

終了イベントは、ビジネスプロセスの終了に使用します。終了イベントには、外向きのシーケンスフローが何も含まれない場合があります。また、ビジネスプロセスに複数の終了イベントが存在する場合があります。「なし」終了イベントおよび中断終了イベント以外の終了イベントはすべてスローイベントです。

終了イベントは、ビジネスプロセスの完了を示します。終了イベントは、特定のワークフローを終了するノードです。このイベントには、1 つまたは複数の内向きシーケンスフローがあり、外向きフローはありません。

プロセスには最低でも 1 つの終了イベントが含まれている必要があります。

ランタイム中は、終了イベントでプロセスワークフローを終了します。終了イベントは、そのイベントに到達したワークフローのみ終了できます。終了イベントタイプによってはプロセスインスタンス内の全ワークフローを終了できます。

表3.3 終了イベント

終了イベント	アイコン
なし	
メッセージ	
シグナル	
エラー	
補正	
エスカレーション	
中断	

なし

「なし」終了イベントは、他に特別な動作がプロセスの終端に関連付けられていないことを示します。

メッセージ

フローがメッセージの終了イベントに入ると、このフローは終了し、終了イベントがプロパティーに定義されているようにメッセージを生成します。

シグナル

スローシグナルの終了イベントは、プロセスまたはサブプロセスフローの終了に使用します。実行フローがこの要素に入ると、実行フローが終了し、**SignalRef** プロパティーで特定されたシグナルを生成します。

エラー

スローエラーの終了イベントは、内向きワークフローを完了します。つまり、内向きのトークンを消費し、エラーオブジェクトを生成します。プロセスまたはサブプロセスで他に実行されているワークフローは、影響を受けません。

補正

補正終了イベントは、トランザクションのサブプロセスを終了し、サブプロセスアクティビティーに接続されている補正中間イベントで定義した補正をトリガーするのに使用します。

エスカレーション

エスカレーション終了イベントは、内向きワークフローを終了します。これは、内向きのトークンを消費して、プロパティーに定義されているようにエスカレーションシグナルを生成し、エスカレーションプロセスをトリガーします。

中断

中断終了イベントは、指定したプロセスインスタンス内の全実行フローを終了します。実行中のアクティビティーはキャンセルされます。サブプロセスで中断終了イベントに到達した場合には、プロセスインスタンス全体が中断されます。

第4章 BUSINESS CENTRAL でのビジネスプロセスの作成

プロセスデザイナーは、Red Hat Process Automation Manager のプロセスモデラーです。モデラーの出力は、BPMN 2.0 プロセス定義ファイルです。この定義は、定義に基づいてプロセスインスタンスを作成する Red Hat Process Automation Manager プロセスエンジンの入力として使用されます。

このセクションの手順では、簡単なビジネスプロセスを作成する方法の概要を説明します。より詳細なビジネスプロセスの例については、『ビジネスプロセスの使用ガイド』を参照してください。

前提条件

- Red Hat Process Automation Manager プロジェクトを作成済みまたはインポート済みである。プロジェクトの作成に関する詳細は、『Business Central におけるプロジェクトの管理』を参照してください。
- 必要なユーザーを作成済みである。ユーザーの特権と設定は、ユーザーに割り当てられたロールとユーザーが属するグループによって制御されます。ユーザーの作成に関する詳細は、『Red Hat JBoss EAP 7.2 への Red Hat Process Automation Manager のインストールおよび設定』を参照してください。

手順

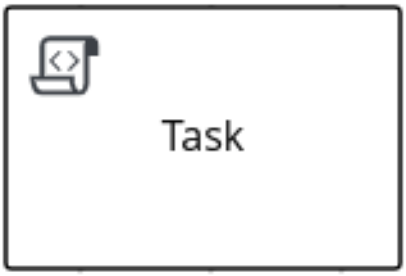
1. Business Central で、**Menu → Design → Projects** に移動します。
2. プロジェクト名をクリックして、プロジェクトのアセットリストを開きます。
3. **Add Asset → Business Process**の順にクリックします。
4. **Create new Business Process**ウィザードで、以下の値を入力します。
   - **Business Process**: 新しいビジネスプロセス名
   - **Package**: 新しいビジネスプロセスのパッケージの場所。例: **com.myspace.myProject**
5. **OK** をクリックしてプロセスデザイナーを開きます。
6. 右上隅の **Diagram properties** アイコンをクリックし、プロセスデータや変数などのビジネスプロセスプロパティー情報を追加します。
   a. スクロールダウンして、**Process Data** を展開します。
   b. **Process Variables** の横にある  をクリックして、ビジネスプロセスで使用するプロセス変数を定義します。
7. プロセスデザイナーキャンバスで、左側のツールバーを使用して BPMN コンポーネントをドラッグアンドドロップし、ビジネスプロセスロジック、接続、イベント、タスク、またはその他の要素を定義します。
8. ビジネスプロセスのすべてのコンポーネントを追加して定義した後に、**Save** をクリックし、完了したビジネスプロセスを保存します。

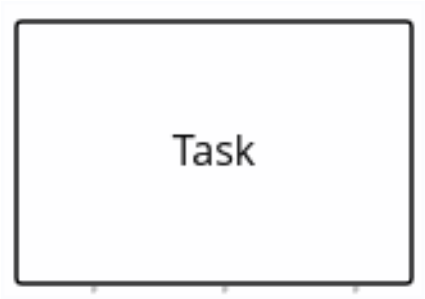
4.1. プロセスデザイナーでの BPMN2 タスク

タスクは、プロセスモデルに定義されている自動アクティビティーで、プロセスフロー内で最小の作業単位です。BPMN 2 仕様に定義されているタスクタイプで、Red Hat Process Automation Manager のプロセスデザイナーパレットで利用できるのは以下のとおりです。

- ビジネスルールタスク
- スクリプトタスク
- ユーザータスク
- サービスタスク
- タスクなし

表4.1 タスク

ビジネスルールタスク	
スクリプトタスク	
ユーザータスク	
サービスタスク	

タスクなし	

さらに、BPMN2 仕様では、カスタムタスクの作成が可能になります。Red Hat Process Automation Manager には、以下の事前定義済みのカスタムタスクが含まれます。

- REST サービスタスク: リモートの RESTful サービスの呼び出しに使用します。
- メールサービスタスク: メールの送信に使用します。
- ログサービスタスク: メッセージのログ記録に使用します。
- Java サービスタスク: Java コードの呼び出しに使用
- WebService サービスタスク: リモートの WebService コールの呼び出しに使用します。
- DecisionTask タスク: DMN ダイアグラム実行に使用します。

ビジネスルールタスク

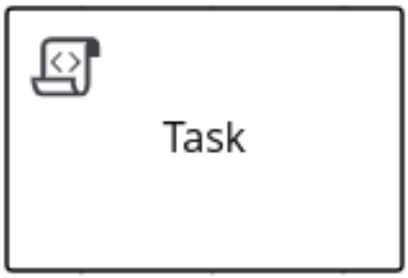
ビジネスルールタスクは、DMN モデルまたはルールフローグループを使用して、意思決定を行う方法を定義します。

プロセスが DMN モデルで定義したビジネスルールタスクに到達したら、プロセスエンジンが、入力された内容を使用して DMN モデルを実行します。

プロセスがルールフローグループで定義したビジネスルールタスクに到達したら、プロセスエンジンは、定義済みのルールフローグループでルールを実行開始します。ルールフローグループにアクティブなルールがない場合には、実行は次の要素に移動します。ルールフローグループの実行中は、アクティブなルールフローグループに所属するアクティベーションは、他のルールで変更されるので、新たにアジェンダに追加できます。

スクリプトタスク

スクリプトタスクは、プロセス実行中に実行されるスクリプトを表します。



関連付けられたスクリプトは、プロセス変数やグローバル変数にアクセスできます。スクリプトタスクを使用する前に以下の一覧をレビューしてください。

- プロセスでは詳細にわたる実装の内容は回避してください。スクリプトタスクは、変数の操作に使用できますが、より複雑な操作をモデル化する場合にサービスタスクの使用を検討してください。
- 即座にスクリプトを実行するようにしてください。すぐに実行しない場合には、非同期サービスタスクを使用してください。
- スクリプトタスクを使用して外部のサービスの問い合わせを回避してください。サービスタスクを使用して、外部サービスとの通信をモデル化します。
- スクリプトで例外がスローされないようにしてください。ランタイムの例外はスクリプト内などで、キャッチし、管理するか、プロセス内で処理できるシグナルまたはエラーに変換する必要があります。

実行中にスクリプトタスクに到達したら、スクリプトが実行され、外向きフローに移動します。

ユーザータスク

ユーザータスクは、システムで自動的に実行できないプロセスワークフローに含まれるタスクなので、ユーザー (人間)、つまり、アクターの介入が必要です。

実行時に、ユーザータスク要素は、1 つ以上のアクターのタスク一覧に表示されるタスクとしてインスタンス化されます。ユーザータスク要素で **Groups** 属性が定義されている場合に、このユーザータスク要素は、グループに所属する全ユーザー一覧に表示されます。このグループに所属するメンバーは誰でもタスクを要求できます。

タスクが要求されると、他のユーザーのタスク一覧からこのタスクは消失します。

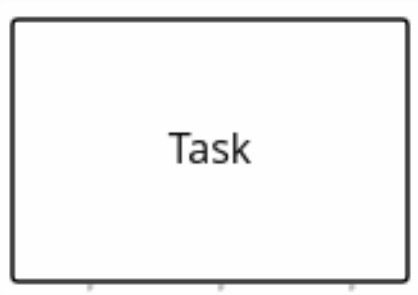
ユーザータスクは、ドメイン固有のタスクとして実装され、カスタムタスクのベースとして機能します。

サービスタスク

サービスタスクは、人間の介入を必要としないタスクです。これらは、外部のソフトウェアサービスによって自動的に完了します。

タスクなし

アクティベーション時に完了するタスクはありません。これは概念モデルのみです。「なし」タスクは、ITシステムによって実際に実行されることはありません。



4.1.1. ビジネスルールタスクの作成

ビジネスルールタスクは、Decision Model and Notation (DMN) モデルまたはルールフローグループを使用して意思決定を行うために使用されます。

手順

1. ビジネスプロセスを作成します。
2. プロセスデザイナーで、ツールパレットから **Activities** ツールを選択します。
3. **Business Rule** を選択します。
4. プロセスデザイナーキャンバスの空白エリアをクリックします。
5. 必要に応じて、画面の右上隅で、**Properties** アイコンをクリックします。
6. 必要に応じて、以下の表に一覧表示されているタスク情報を追加または定義します。

表4.2 ビジネスルールタスクのパラメーター

Label	説明
Name (文字列)	ビジネスルールタスクの名前。
Rule Language	タスクの出力言語。Decision Model and Notation (DMN) または Drools (DRL) を選択します。
Rule Flow Group	このビジネスタスクに関連付けられたルールフローグループ。一覧からルールフローグループを選択するか、新しいルールフローグループを指定します。

Label	説明
On Entry Action	タスクの開始時のアクションを指定する Java、JavaScript、または MVEL スクリプト。
On Exit Action	タスクの終了時にアクションを指定する Java、JavaScript、または MVEL スクリプト。
Is Async	このタスクを非同期で呼び出す必要がある場合に選択します。外部サービスによって実行されるタスクなど、タスクを瞬時に実行できない場合は、タスクを非同期にします。
Adhoc Autostart	これが自動的に開始される必要があるアドホックタスクである場合に選択します。**Adhoc Autostart** を使用すると、プロセスまたはケースインスタンスが作成されたときに、開始タスクによって開始されるのではなく、タスクが自動的に開始されます。多くの場合、ケース管理で使用されます。
SLA Due Date	サービスレベルアグリーメント (SLA) の有効期限が切れる日付。
Assignments	クリックしてローカル変数を追加します。

7. **Save** をクリックします。

4.1.2. スクリプトタスクの作成

スクリプトタスクは、Java、JavaScript、または MVEL で記述されたコードを実行するために使用されます。これらには、スクリプトタスクのアクションを指定するコードスニペットが含まれています。スクリプトにグローバル変数とプロセス変数を含めることができます。

Java、JavaScript、および MVEL でアクションスクリプトを作成できます。MVEL は有効な Java コードをすべて受け入れ、さらにパラメーターのネストされたアクセスをサポートすることに留意してください。たとえば、Java 呼び出し **person.getName()** に相当する MVEL は、 **person.name** です。MVEL は Java に対して他の改善も提供し、MVEL 式は一般にビジネスユーザーにとってより便利です。

手順

1. ビジネスプロセスを作成します。
2. プロセスデザイナーで、ツールパレットから **Activities** ツールを選択します。
3. **Script** を選択します。
4. プロセスデザイナーキャンバスの空白エリアをクリックします。
5. 必要に応じて、画面の右上隅で、**Properties** アイコンをクリックします。
6. 必要に応じて、以下の表に一覧表示されているタスク情報を追加または定義します。

表4.3 スクリプトタスクのパラメーター

Label	説明
Name (文字列)	ビジネスルールタスクの名前。
Documentation	タスクの説明を入力します。このフィールドのテキストは、プロセスのドキュメントに含まれています。プロセスデザイナーキャンバスの左上にある **Documentation** タブをクリックして、プロセスドキュメントを表示します。
Script	タスクによって実行されるスクリプトを Java、JavaScript、または MVEL で入力し、スクリプトタイプを選択します。
Is Async	このタスクを非同期で呼び出す必要がある場合に選択します。外部サービスによって実行されるタスクなど、タスクを瞬時に実行できない場合は、タスクを非同期にします。
Adhoc Autostart	これが自動的に開始される必要があるアドホックタスクである場合に選択します。**Adhoc Autostart** を使用すると、プロセスまたはケースインスタンスが作成されたときに、開始タスクによって開始されるのではなく、タスクが自動的に開始されます。多くの場合、ケース管理で使用されます。

7. **Save** をクリックします。

4.1.3. ユーザータスクの作成

ユーザータスクは、ビジネスプロセスに人間が行うアクションをインプットとして追加するために使用します。

手順

1. ビジネスプロセスを作成します。
2. プロセスデザイナーで、ツールパレットから **Activities** ツールを選択します。
3. **User** を選択します。
4. ビジネスルールをプロセスデザイナーキャンバスにドラッグアンドドロップするか、キャンバスの空白エリアをクリックします。
5. 必要に応じて、画面の右上隅で、**Properties** アイコンをクリックします。
6. 必要に応じて、以下の表に一覧表示されているタスク情報を追加または定義します。

表4.4 ユーザータスクパラメーター

Label	説明
Name (文字列)	ビジネスルールタスクの表示名。

Label	説明
Documentation	タスクの説明を入力します。このフィールドのテキストは、プロセスのドキュメントに含まれています。プロセスデザイナーキャンバスの左上にある **Documentation** タブをクリックして、プロセスドキュメントを表示します。
Task Name	ヒューマンタスクの名前。
Subject	タスクの件名を入力します。
Actors	ヒューマンタスクの実行を担当するアクター。**Add** をクリックして行を追加し、一覧からアクターを選択するか、**New** をクリックして新しいアクターを追加します。
Groups	ヒューマンタスクの実行を担当するグループ。**Add** をクリックして行を追加し、一覧からグループを選択するか、**New** をクリックして新しいグループを追加します。
Assignments	このタスクのローカル変数。**Task Data I/O** ウィンドウをクリックして開き、必要に応じてデータの入力と出力を追加します。
Reassignments	別のアクターを指定して、このタスクを完了します。
Notifications	クリックして、タスクに関連付けられた通知を指定します。
Is Async	このタスクを非同期で呼び出す必要がある場合に選択します。外部サービスによって実行されるタスクなど、タスクを瞬時に実行できない場合は、タスクを非同期にします。
Skippable	このタスクが必須ではない場合に選択します。
Priority	タスクの優先度を指定します。
Description	ヒューマンタスクの説明を入力します。
Created By	このタスクを作成したユーザー。
Adhoc Autostart	これが自動的に開始される必要があるアドホックタスクである場合に選択します。**Adhoc Autostart** を使用すると、プロセスまたはケースインスタンスが作成されたときに、開始タスクによって開始されるのではなく、タスクが自動的に開始されます。多くの場合、ケース管理で使用されます。
Multiple Instance	このタスクに複数のインスタンスがある場合に選択します。
On Entry Action	タスクの開始時のアクションを指定する Java、JavaScript、または MVEL スクリプト。

Label	説明
On Exit Action	タスクの終了時にアクションを指定する Java、JavaScript、または MVEL スクリプト。
Content	スクリプトのコンテンツ。
SLA Due Date	サービスレベルアグリーメント (SLA) の有効期限が切れる日付。

7. **Save** をクリックします。

4.1.4. サービスタスクの作成

サービスタスクは、プロセス外で実行されるプロセスの一部であるタスクですが、ヒューマンタスクではありません。サービスタスクの例には、これらのタスクがシステムによって実行されたときに電子メールを送信し、メッセージを記録することが含まれます。サービスタスクに関連付けられているパラメーター (入力) と結果 (出力) を定義できます。サービスタスクには、内向きの接続 1 つと外向きの接続 1 つが必要です。

手順

1. ビジネスプロセスを作成します。
2. プロセスデザイナーで、ツールパレットから **Activities** ツールを選択します。
3. **Service Task** を選択します。
4. プロセスデザイナーキャンバスの空白エリアをクリックします。
5. 必要に応じて、画面の右上隅で、**Properties** アイコンをクリックします。
6. 必要に応じて、以下の表に一覧表示されているタスク情報を追加または定義します。

表4.5 サービスタスクパラメーター

Label	説明
Name (文字列)	サービスタスクの名前。
Documentation	タスクの説明を入力します。このフィールドのテキストは、プロセスのドキュメントに含まれています。プロセスデザイナーキャンバスの左上にある **Documentation** タブをクリックして、プロセスドキュメントを表示します。
Implementation	タスクが Java で実装されているか、Web サービスであるかを指定します
Interface	**org.xyz.HelloWorld** など、スクリプトの実装に使用されるクラス。

Label	説明
Operation	**sayHello()** など、インターフェースによって呼び出されるメソッド。
Assignments	クリックしてローカル変数を追加します。
Adhoc Autostart	これが自動的に開始される必要があるアドホックタスクである場合に選択します。**Adhoc Autostart** を使用すると、プロセスまたはケースインスタンスが作成されたときに、開始タスクによって開始されるのではなく、タスクが自動的に開始されます。多くの場合、ケース管理で使用されます。
Is Async	このタスクを非同期で呼び出す必要がある場合に選択します。外部サービスによって実行されるタスクなど、タスクを瞬時に実行できない場合は、タスクを非同期にします。
Is Multiple Instance	このタスクに複数のインスタンスがある場合に選択します。
On Entry Action	タスクの開始時のアクションを指定する Java、JavaScript、または MVEL スクリプト。
On Exit Action	タスクの終了時にアクションを指定する Java、JavaScript、または MVEL スクリプト。
SLA Due Date	サービスレベルアグリーメント (SLA) の有効期限が切れる日付。

7. **Save** をクリックします。

4.2. ビジネスプロセスのコピーの作成

Business Central でビジネスプロセスのコピーを作成し、必要に応じてコピーしたプロセスを変更できます。

手順

1. ビジネスプロセスデザイナーで、右上のツールバーの **Copy** をクリックします。
2. **Make a Copy** ウィンドウで、コピーしたビジネスプロセスの新しい名前を入力し、ターゲットパッケージを選択して、オプションでコメントを追加します。
3. **Make a Copy** をクリックします。
4. 必要に応じてコピーしたビジネスプロセスを変更し、**Save** をクリックして、更新されたビジネスプロセスを保存します。

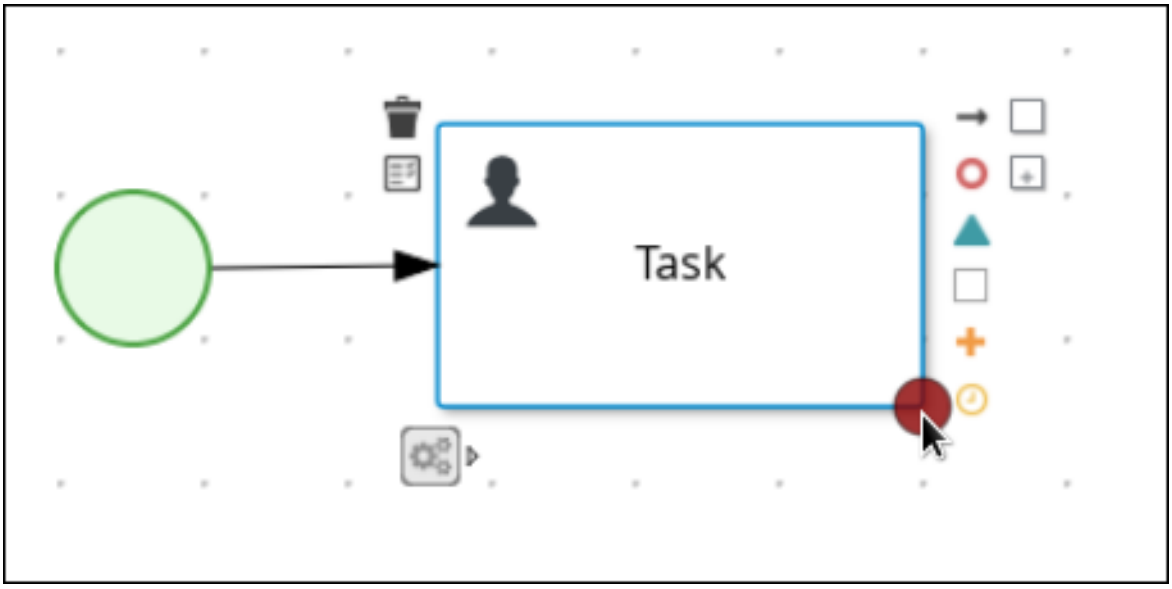
4.3. 要素のサイズを変更し、ズーム機能を使用したビジネスプロセスの表示

ビジネスプロセスの個々の要素のサイズを変更し、ズームインまたはズームアウトして、ビジネスプロセスの表示を変更できます。

手順

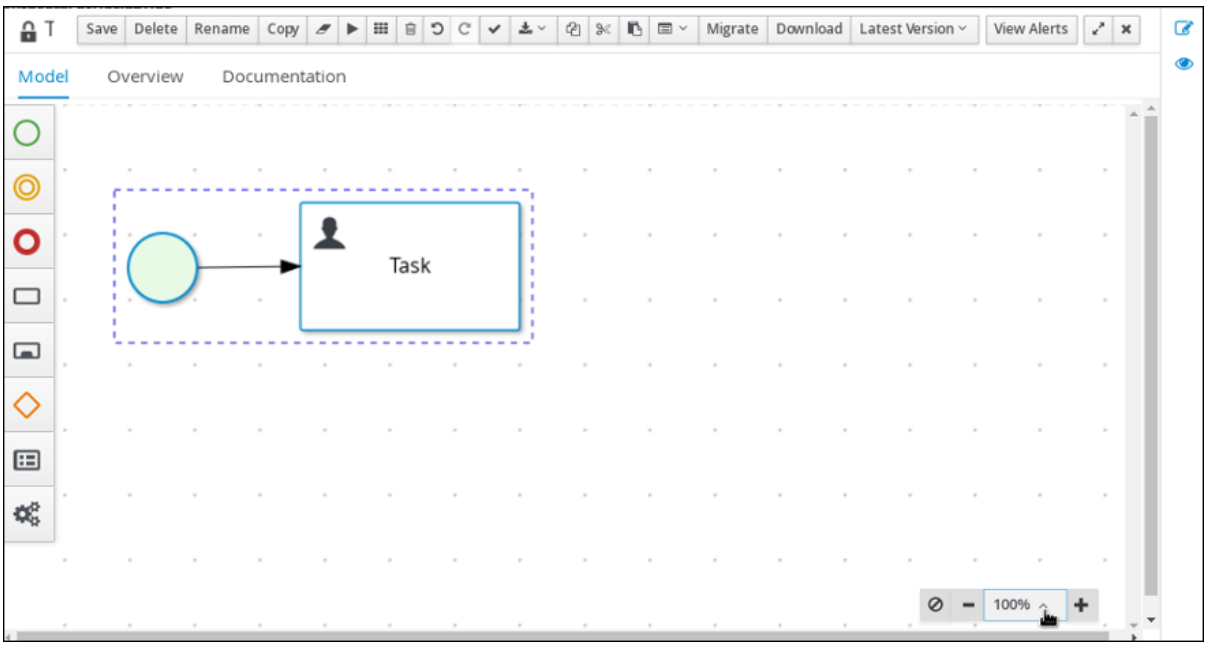
1. ビジネスプロセスデザイナーで、要素を選択し、要素の右下隅にある赤い点をクリックします。
2. 赤い点をドラッグして、要素のサイズを変更します。

図4.1 要素のサイズ変更



3. ズームインまたはズームアウトしてダイアグラム全体を表示するには、キャンバスの右下にあるプラス記号またはマイナス記号をクリックします。

図4.2 ビジネスプロセスの拡大または縮小



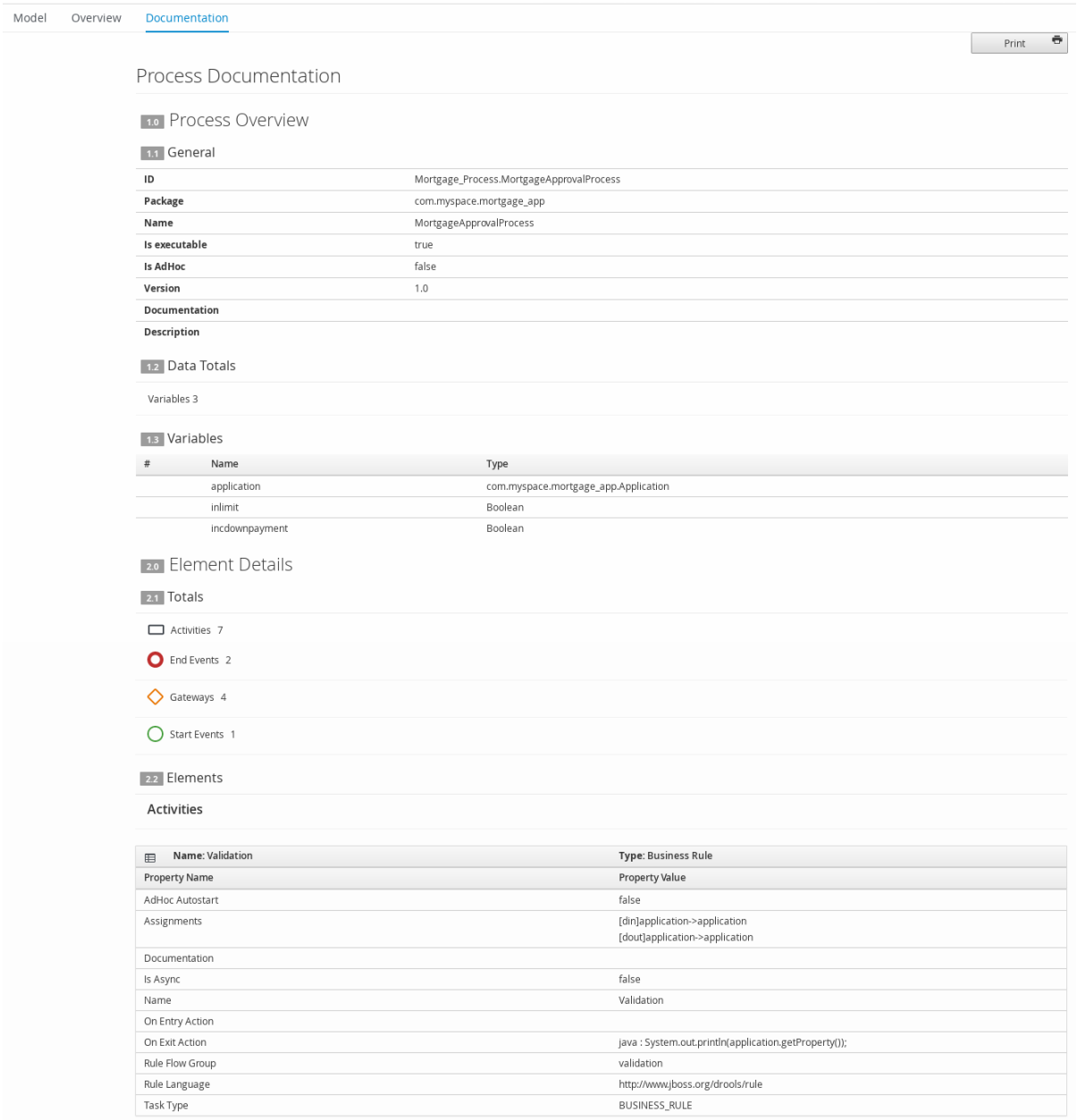
4.4. BUSINESS CENTRAL でのプロセスドキュメントの生成

Business Central のプロセスデザイナーでは、プロセス定義のレポートを表示し、出力できます。プロセスドキュメントには、コンポーネント、データ、プロセスの視覚的なフローが簡単に出力して共有できるように PDF 形式でまとめられています。

手順

1. Business Central で、ビジネスプロセスが含まれるプロジェクトに移動し、プロセスを選択します。
2. プロセスデザイナーの **Documentation** タブでプロセスファイルの概要を表示し、画面の右上隅の **Print** をクリックして PDF レポートを出力します。

図4.3 プロセスドキュメントの生成



4.5. プロセスデザイナー内の BPMN2 サブプロセス

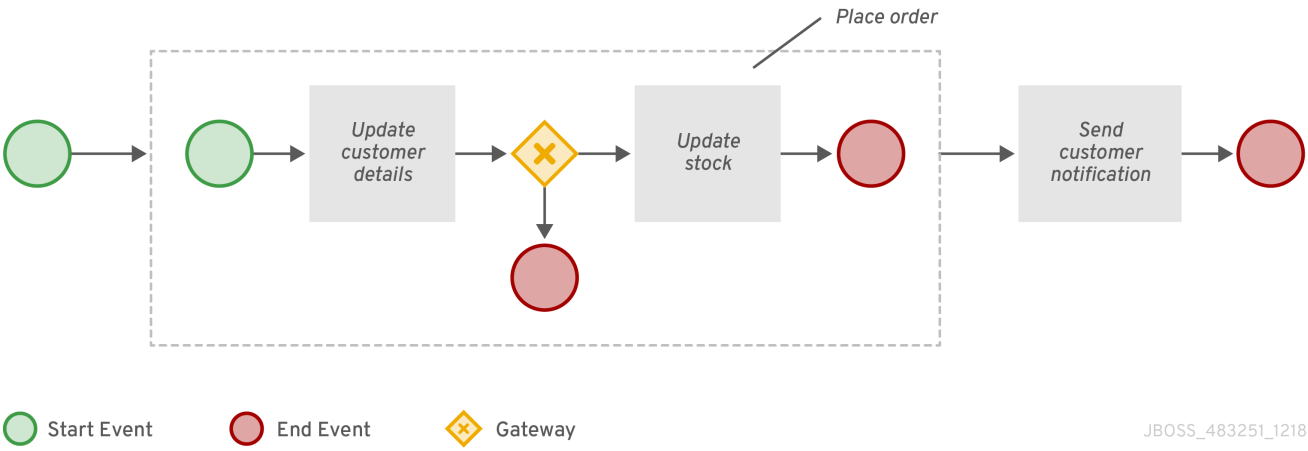
サブプロセスは、複数のノードが含まれるアクティビティーです。サブプロセスにメインのプロセスの一部を埋め込むことができます。また、サブプロセスには変数定義を追加できます。これらの変数は、サブプロセス内の全ノードにアクセスできます。

サブプロセスには、内向きの接続と外向きの接続1つずつ含める必要があります。サブプロセスに中断終了イベントを使用する場合には、サブプロセスだけでなく、サブプロセスが含まれる全プロセスインスタンスが中断されます。サブプロセスからアクティブな要素がなくなると、サブプロセスが終了します。

Red Hat Process Automation Manager では、以下のサブプロセスのタイプがサポートされます。

- 埋め込みのサブプロセス。親プロセス実行の一部で、親プロセスのデータを共有します。
- アドホックサブプロセス。厳密な要素実行の順番はありません。
- 再利用可能なサブプロセス。親プロセスから独立しています。
- イベントサブプロセス。開始イベントまたはタイマーでのみトリガーされます。
- マルチインスタンスのサブプロセス

以下の例では、発注のサブプロセスは、その注文を受けるのに十分な在庫があるかを確認し、注文できた場合に在庫情報を更新します。注文の可否により、メインのプロセス経由で、顧客に通知が行きます。



埋め込みサブプロセス

埋め込みサブプロセスは、プロセスの一部をカプセル化します。このサブプロセスには、開始イベントと、最低でも 1 つの終了イベントが含まれている必要があります。この要素を使用して、このコンテナー内の全要素にアクセスできるローカルのサブプロセス変数を定義できます。

アドホックサブプロセス

アドホックサブプロセスまたはプロセスには、埋め込みの内部アクティビティーが複数含まれ、通常のプロセスルーティングに比べて、より柔軟な順番で実行することを目的としています。通常のプロセスとは違い、アドホックサブプロセスには、開始イベントから終了イベントまでといった、完全な体系化された BPMN2 ダイアグラムの説明は含まれません。代わりに、アクティビティー、シーケンスフロー、ゲートウェイ、中間イベントのみが含まれます。また、アドホックサブプロセスには、データオブジェクトやデータの関連付けも含めることができます。アドホックサブプロセス内のアクティビティーでは、内向きおよび外向きのシーケンスフローを含める必要はありませんが、その中に含まれているアクティビティー間のシーケンスフローを指定できます。このサブプロセスを使用する場合には、シーケンスフローで、通常のプロセスと同じように順序の制約が課されます。意味をもたせるには、中間イベントに外向きのシーケンスフローを設定して、アドホックプロセスがアクティブな間は複数回トリガーできるようにします。

再利用可能なサブプロセス

再利用可能なサブプロセスは、このサブプロセスが含まれるプロセス内に折りたたまれて表示されます。

イベントサブプロセス

イベントサブプロセスは、開始イベントがトリガーされるとアクティブになります。親プロセスのコンテキストを中断するか、並行して実行できます。

外向きまたは内向きの接続では、イベントまたはタイマーがサブプロセスをトリガーできます。サブプロセスは、通常のコントロールフローの一部ではありません。自己完結型ではありますが、バインドされているプロセスのコンテキストで実行されます。

プロセスフロー内のイベントサブプロセスを使用して、主なプロセスフロー外で発生するイベントを処理します。たとえば、飛行機の予約時には、以下の 2 つのイベントが発生する可能性があります。

- 予約の取り消し (割り込み)
- 予約ステータスの確認 (割り込みなし)

イベントのサブプロセスを使用して、これらのイベント両方をモデル化します。

マルチインスタンスサブプロセス

マルチインスタンスサブプロセスは、実行がトリガーされると、複数回インスタンス化されます。インスタンスが順次作成されます。新しいサブプロセスインスタンスは、前のインスタンスが完了した後にのみ作成されます。

マルチインスタンスサブプロセスには、内向きの接続1つと、外向きの接続1つが含まれます。

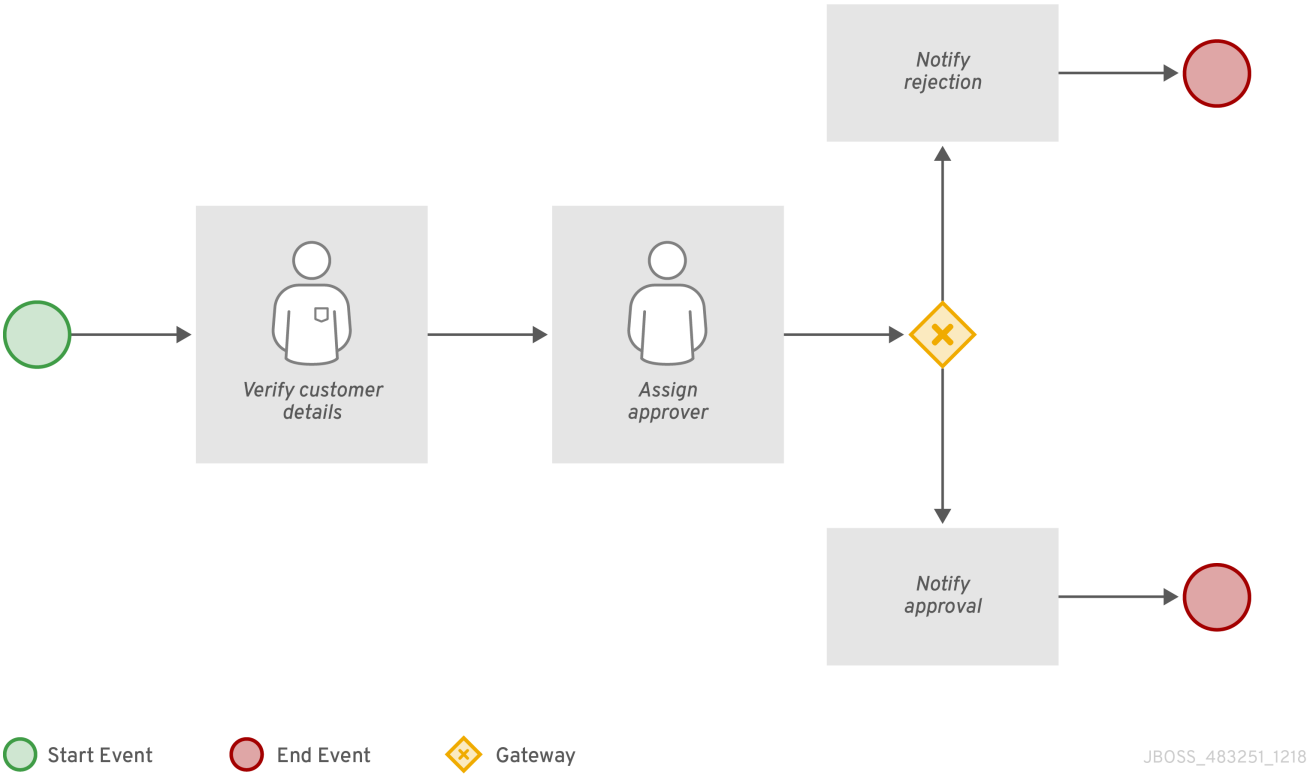
4.6. プロセスデザイナーでの BPMN2 ゲートウェイ

ゲートウェイは、ゲートメカニズムと呼ばれる条件セットを使用して、ワークフロー内にブランチを作成するか、ワークフロー内のブランチを同期するのに使用します。BPMN2 は、2 種類のゲートウェイをサポートします。

- 収束ゲートウェイ。複数のフローを1つにマージします。
- 分岐ゲートウェイ。1つのフローを複数のフローに分割します。

1つのゲートウェイに、複数の内向きと外向きフローを割り当てることはできません。

以下のビジネスプロセスダイアグラムで、XOR ゲートウェイは、条件が True と評価された内向きフローのみを評価します。



この例では、顧客の詳細がユーザーにより検証され、ユーザーが承認できるようにプロセスが割り当てられます。承認されると、承認通知がユーザーに送信されます。要求イベントが却下された場合には、却下通知がユーザーに送信されます。

表4.6 ゲートウェイ要素

要素タイプ	アイコン
排他的論理和 (XOR)	
包含的	
並列	

要素タイプ	アイコン
イベント	

排他的

排他的な分岐ゲートウェイでは、最初の内向きフローで条件が True と評価されたもののみが選択されます。収束ゲートウェイでは、トリガーされた内向きのフローごとに、次のノードがトリガーされます。

このゲートウェイは、1つだけ外向きフローをトリガーします。フローの条件が True と評価されタナカで、優先順位が **最も低い** 数字が選択されます。

重要

実行時に、最低でも1つの外向きフローが True と評価されるようにしてください。そうでないと、プロセスインスタンスは、ランタイムの例外で中断されます。

収束ゲートウェイでは、ワークフローブランチが、ゲートウェイに到達すると同時に外向きフローに進むことができます。内向きフローの1つがゲートウェイをトリガーすると、ワークフローはゲートウェイの外向きフローに進みます。複数の内向きフローからトリガーされた場合には、トリガーごとに次のノードをトリガーします。

包含的

包含的な分岐ゲートウェイでは、内向きフローが選択され、さらに True と評価された外向きフローすべてが選択されます。優先順位の数値が低い接続は、数値が高い接続よりも先にトリガーされます。優先順位は評価されますが、BPMN2 仕様では優先順位の順番は保証されません。ワークフローで **priority** 属性に依存しないようにしてください。

重要

実行時に、最低でも1つの外向きフローが True と評価されるようにしてください。そうでないと、プロセスインスタンスは、ランタイムの例外で中断されます。

包含的な収束ゲートウェイでは、包含的な分岐ゲートウェイでこれまでに作成された内向きフローすべてがマージされます。これは、包含ゲートウェイブランチの同期エントリーポイントとして機能します。

並列

並列ゲートウェイを使用して、並列フローを同期し、作成します。並列の分岐ゲートウェイでは、内向きフローが選択されると同時に、外向きフローもすべて選択されます。収束並列ゲートウェイでは、ゲートウェイは、内向きのフローがすべて到達するまで待機してからでないと、外向きフローをトリガーしません。

イベント

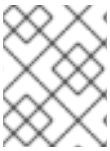
イベントベースのゲートウェイは分岐のみで、データをもとにした排他的ゲートウェイ (プロセスデータに反応) とは対照的に、発生する可能性のあるイベントに反応できます。発生したイベントをもとに、外向きフローに移動します。1度に実行できる外向きフローは1つとなっています。ゲートウェイは、イベントベースのゲートウェイに接続されている中間イベントが発生した場合にのみ、プロセスがインスタンス化される、開始イベントとして機能する可能性があります。

4.7. プロセスデザイナーでの BPMN2 接続オブジェクト

接続オブジェクトは、BPMN2 要素 2 つの間の関連性を作成します。接続オブジェクトが転送された場合には、関連付けは順番に行われ、プロセスのインスタンス内で、要素の1つが他の要素よりも先に即座に実行されることを指定します。接続オブジェクトは、関連付けられているプロセス要素の上、下、右、左側で開始、終了できます。OMG BPMN2 仕様では、プロセスの動作を簡単に理解して従うことができるように、接続オブジェクトを独断で配置できます。

BPMN2 は主に、2 種類の接続オブジェクトをサポートします。

- シーケンスフロー: プロセスの要素を接続し、インスタンス内でこれらの要素を実行する順番を定義します。
- 関連付けフロー: 実行セマンティクスなしでプロセスの要素を接続します。関連付けフローは転送できません。できる場合は、単方向となっています。



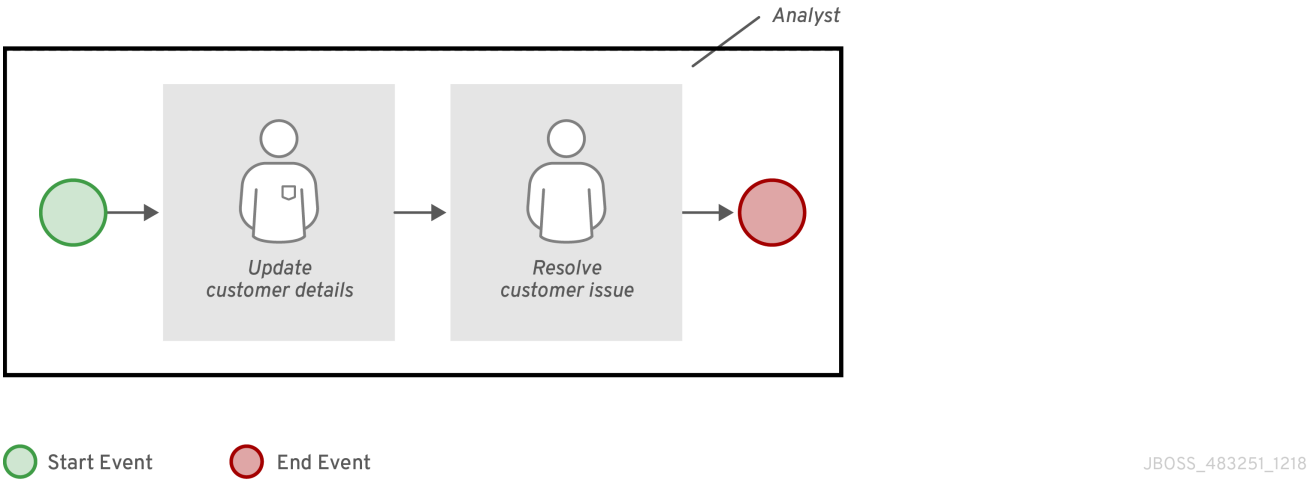
注記

新しいプロセスデザイナーは、転送されない関連付けフローのみをサポートします。レガシーのデザイナーは、一方向と単方向のフローをサポートしています。

4.8. プロセスデザイナーでの BPMN2 スイムレーン

スイムレーンは、1つのグループまたはユーザーに関連のあるタスクを視覚的にグループ化するプロセス要素です。スイムレーンとユーザータスクを組み合わせて使用し、人間による複数のタスクを同じアクターに割り当てます。ランタイム時に、スイムレーンは、同じプロセスインスタンス内で、タスクを自動的に要求するか、そのレーンで別のタスクを完了したユーザーに割り当てます。スイムレーンで最初のタスクが作成され、タスクにアクター ID が指定されている場合には、アクター ID がそのスイムレーンの他の全タスクにも割り当てられます。レーンとは、プロセス内のサブパーティションのことで、プロセス要素をグループ化し、共通のパラメーターを定義できます。

以下の例では、アナリストには 2 つのユーザータスクが割り当てられています。



Update Customer Details と Resolve Customer Issue のタスクの **Group** フィールドには **analyst** の値が

割り当てられています。プロセスが開始されて、Update Customer Details タスクがアナリストユーザーにより要求されるか、開始されるか、完了されると、Resolve Customer Issue タスクが要求されて、最初のタスクを完了したユーザーに割り当てられます。ただし、Update Customer Details タスクにアナリストグループが割り当てられていて、2 番目のタスクにユーザーやグループが割り当てられていない場合に、プロセスは最初のタスク完了後に停止します。

第5章 変数

変数には、実行時に使用されるデータが格納されます。プロセスデザイナーは、3 種類の変数を使用します。

グローバル変数

グローバル変数は、特定のセッションのすべてのプロセスインスタンスとアセットに表示されます。これらは、主にビジネスルールおよび制約によって使用されることを目的としており、ルールまたは制約によって動的に作成されます。

プロセス変数

プロセス変数は、BPMN2 定義ファイルのプロパティーとして定義され、プロセスインスタンス内で表示されます。プロセスの作成時に初期化され、プロセスの完了時に破棄されます。

ローカル変数

ローカル変数は、アクティビティーなどの特定のプロセス要素に関連付けられ、その中で使用できます。要素コンテキストが初期化されると、ローカル変数は初期化されます。つまり、実行ワークフローがノードに入り、**onEntry** アクションの実行が終了した場合に初期化されます (該当する場合)。要素コンテキストが破棄されると、ローカル変数は破棄されます。つまり、実行ワークフローが要素を離れる場合に破棄されます。

プロセス、サブプロセス、またはタスクなどの要素は、独自のコンテキストと親コンテキストの変数にのみアクセスできます。要素は、要素の子要素で定義された変数にアクセスできません。したがって、実行時に要素が変数へのアクセスを必要とする場合、独自のコンテキストが最初に検索されます。

変数が要素のコンテキストで直接見つからない場合、直接の親コンテキストが検索されます。検索は、プロセスコンテキストに到達するまで続行されます。グローバル変数の場合、検索はセッションコンテナーで直接実行されます。

変数が見つからない場合、読み取りアクセス要求は **null** を返し、書き込みアクセスはエラーメッセージを生成して、プロセスは実行を継続します。変数は ID に基づいて検索されます。

5.1. グローバル変数の定義

グローバル変数はナレッジセッションに存在し、アクセスが可能で、そのセッションのすべてのアセットで共有されます。これらはナレッジベースの特定のセッションに属し、エンジンに情報を渡すために使用されます。すべてのグローバル変数は、その ID とアイテムサブジェクト参照を定義します。ID は変数名として機能し、プロセス定義内で一意である必要があります。アイテムサブジェクト参照は、変数が保存するデータタイプを定義します。

重要

ルールは、ファクトが挿入されたときに評価されます。したがって、ファクトパターンを制約するためにグローバル変数を使用していて、グローバルが設定されていない場合、システムは **NullPointerException** を返します。

グローバル変数は、変数定義を持つプロセスがセッションに追加されるとき、またはセッションがパラメーターとしてグローバルで初期化されるときに初期化されます。

グローバル変数の値は通常、割り当て中に変更できます。これは、プロセス変数とアクティビティー変数間のマッピングです。続いてグローバル変数は、ローカルアクティビティーコンテキスト、ローカルアクティビティー変数、または子コンテキストから変数への直接呼び出しによって関連付けられます。

前提条件

- Business Central でプロジェクトを作成済みである。これにはビジネスプロセスアセットが 1 つ以上含まれます。

手順

1. ビジネスプロセスアセットを開きます。
2. プロセスデザイナーキャンバスの空白エリアをクリックします。
3. 画面の右上にある **Properties** アイコンをクリックして、**Properties** パネルを開きます。
4. 必要に応じて、**Process** セクションを展開します。
5. **Global Variables** のサブセクションで、プラスアイコンをクリックします。
6. **Name** ボックスに変数の名前を入力します。
7. **Data Type** メニューからデータタイプを選択します。

5.2. プロセス変数の定義

プロセス変数は、BPMN2 定義ファイルのプロパティーとして定義され、プロセスインスタンス内で表示されます。プロセスの作成時に初期化され、プロセスの完了時に破棄されます。

プロセス変数は、プロセスコンテキストに存在する変数であり、そのプロセスまたはその子要素からアクセスできます。プロセス変数は特定のプロセスインスタンスに属し、他のプロセスインスタンスからアクセスすることはできません。すべてのプロセス変数は、その ID とアイテムサブジェクト参照を定義します。ID は変数名として機能し、プロセス定義内で一意である必要があります。アイテムサブジェクト参照は、変数が保存するデータタイプを定義します。

プロセス変数は、プロセスインスタンスの作成時に初期化されます。それらの値は、グローバル変数がローカルアクティビティーコンテキスト、ローカルアクティビティー変数に関連付けられている場合に、割り当てを使用するプロセスアクティビティーによって、または子コンテキストから変数への直接呼び出しによって変更できます。

プロセス変数は、ローカル変数にマッピングする必要があることに注意してください。

前提条件

- Business Central でプロジェクトを作成済みである。これにはビジネスプロセスアセットが 1 つ以上含まれます。

手順

1. ビジネスプロセスアセットを開きます。
2. プロセスデザイナーキャンバスの空白エリアをクリックします。
3. 画面の右上にある **Properties** アイコンをクリックして、**Properties** パネルを開きます。
4. 必要に応じて、**Process Data** セクションを展開します。
5. **Process Variables** サブセクションで、プラスアイコンをクリックします。

6. **Name** ボックスに変数の名前を入力します。
7. **Data Type** メニューからデータタイプを選択します。

5.3. ローカル変数の定義

ローカル変数は、アクティビティーなどのプロセス要素内で使用できます。要素コンテキストが初期化されると、ローカル変数は初期化されます。つまり、実行ワークフローがノードに入り、**onEntry** アクションの実行が終了した場合に初期化されます (該当する場合)。要素コンテキストが破棄されると、ローカル変数は破棄されます。つまり、実行ワークフローが要素を離れる場合に破棄されます。

ローカル変数の値は、グローバル変数またはプロセス変数にマッピングできます。これにより、ローカル変数を収容する親要素の相対的な独立性を維持できます。このような分離は、技術的な例外の防止に役立つ場合があります。

ローカル変数は、プロセスの子要素コンテキストに存在する変数であり、このコンテキスト内からのみアクセスできます。ローカル変数は、プロセスの特定の要素に属します。

スクリプトタスクを除くタスクの場合、**Assignments** プロパティーの **Data Input Assignments** および **Data Output Assignments** を定義できます。Data Input Assignments は、Task に入る変数を定義し、タスクの実行に必要な入力データを提供します。Data Output Assignments は、実行後の Task のコンテキストを参照して、出力データを取得できます。

ユーザータスクは、ユーザータスクを実行しているアクターに関連するデータを提示します。さらに、ユーザータスクは、実行に関連する結果データを提供するようにアクターに要求します。

データを要求および提供するには、タスクフォームを使用し、Data Input Assignment パラメーターのデータを変数にマッピングします。データを出力として保持する場合は、Data Output Assignment パラメーターでユーザーが提供したデータをマッピングします。

前提条件

- Business Central でプロジェクトを作成済みである。プロジェクトには、スクリプトタスクではないタスクが 1 つ以上あるビジネスプロセスアセットが 1 つ以上含まれます。

手順

1. ビジネスプロセスアセットを開きます。
2. スクリプトタスクではないタスクを選択します。
3. 画面の右上にある **Properties** アイコンをクリックして、**Properties** パネルを開きます。
4. 必要に応じて、**Data Assignments** セクションを展開します。
5. **Assignments** サブセクションの下のボックスをクリックします。**Task Data I/O** ダイアログボックスが開きます。
6. **Data Inputs and Assignments** または **Data Inputs and Assignments** の横にある **Add** をクリックします。
7. **Name** ボックスにローカル変数の名前を入力します。
8. **Data Type** メニューからデータタイプを選択します。
9. ソースまたはターゲットを選択してから、**Save** をクリックします。

第6章 BUSINESS CENTRAL でのビジネスプロセスのデプロイ

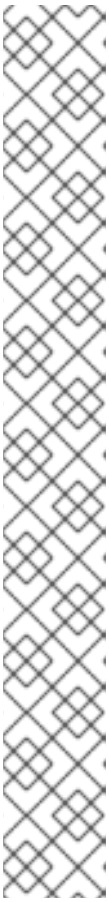
Business Central でビジネスプロセスを設計したら、Business Central でプロジェクトをビルドおよびデプロイして、KIE Server でプロセスを使用できるようにします。

前提条件

- KIE Server がデプロイされて Business Central に接続されている。KIE Server の設定に関する詳細は、『Red Hat JBoss EAP 7.2 への Red Hat Process Automation Manager のインストールおよび設定』を参照してください。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。
2. デプロイするプロジェクトをクリックします。
3. **Deploy** をクリックします。



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。

プロジェクトのデプロイメントに関する詳細を確認するには、画面の上部にあるデプロイメントバナーの **View deployment details** か、**Deploy** のドロップダウンメニューをクリックします。このオプションを使用すると、**Menu → Deploy → Execution Servers** ページに移動します。

第7章 BUSINESS CENTRAL でのビジネスプロセスの実行

ビジネスプロセスを含むプロジェクトをビルドおよびデプロイした後、ビジネスプロセスに定義された機能を実行できます。

例として、この手順では Business Central の **Mortgage_Process** のサンプル例を使用します。このシナリオでは、住宅ローンブローカーとして、住宅ローン申請書にデータを入力します。**MortgageApprovalProcess** ビジネスプロセスが実行し、プロジェクトで定義しておいたデシジョンルールに基づいて、申請者が条件に合った頭金を提示したかどうかを判断します。ビジネスプロセスは、ルールのテストを終了するか、続行するために申請者が頭金を増額することを依頼します。申請書がビジネスルールのテストをパスしたら、銀行の承認者が申請書を見直し、ローンを承認または却下します。

前提条件

- KIE Server がデプロイされて Business Central に接続されている。KIE Server の設定に関する詳細は、『Red Hat JBoss EAP 7.2 への Red Hat Process Automation Manager のインストールおよび設定』を参照してください。

手順

1. Business Central で、**Menu → Projects** に移動して、スペースを選択します。デフォルトのスペースは MySpace です。
2. ウィンドウの右上隅にある **Add Project** の横の矢印をクリックし、**Try Samples** を選択します。
3. **Mortgage_Process** サンプルを選択し、**OK** をクリックします。
4. プロジェクトページで、**Mortgage_Process** を選択します。
5. **Mortgage_Process** ページで、**Build** をクリックします。
6. プロジェクトがビルドされたら、**Deploy** をクリックします。
7. **Menu → Manage → Process Definitions** の順にクリックします。
8. **MortgageApprovalProcess** 行の任意の場所をクリックし、プロセスの詳細を表示します。
9. **Diagram** タブをクリックし、エディターでビジネスプロセスダイアグラムを表示します。
10. **New Process Instance** をクリックすると **Application** フォームが開き、以下の値をフォームフィールドに入力します。
    - Down Payment: **30000**
    - Years of amortization: **10**
    - Name: **Ivo**
    - Annual Income: **60000**
    - SSN: **123456789**
    - Age of property: **8**
    - Address of property: **Brno**

- Locale: **Rural**
- Property Sale Price: **50000**

11. **Submit** をクリックして、新しいプロセスインスタンスを開始します。プロセスインスタンスを開始すると、**Instance Details** ビューが開きます。
12. **Diagram** タブをクリックして、プロセスダイアグラムのプロセスフローを表示します。各タスクを通過した時のプロセスの状態が強調表示されます。
13. **Menu → Manage → Tasks** をクリックします。
この例では、対応するタスクで作業しているユーザーは、以下のグループのメンバーです。
    - approver: **Qualify** タスクの場合
    - broker: **Correct Data** タスクおよび **Increase Down Payment**タスクの場合
    - manager: **Final Approval** タスクの場合
14. 承認者として、**Qualify** タスク情報を確認し、**Claim** をクリックしてから **Start** をクリックしてタスクを開始します。続いて、**Is mortgage application in limit?**を選択し、**Complete** をクリックしてタスクフローを完了します。
15. **Tasks** ページで、**Final Approval** 行の任意の場所をクリックし、**Final Approval** タスクを開きます。
16. **Claim** をクリックして、タスクの担当を要求し、**Complete** をクリックして、ローンの承認プロセスを終了します。

注記

Save ボタンおよび **Release** ボタンは、承認プロセスを中断したり、(フィールド値を待っている場合は) インスタンスを保存したり、別のユーザーが修正するタスクを解除したりするために使用します。

第8章 BUSINESS CENTRAL でのプロセス定義とプロセスインスタンス

プロセス定義は、Business Process Model and Notation (BPMN) 2.0 ファイルであり、プロセスとその BPMN ダイアグラムのコンテナーとして機能します。プロセス定義には、関連するサブプロセスや、選択した定義に参加しているユーザーとグループの数など、ビジネスプロセスに関する利用可能な情報がすべて表示されます。

プロセス定義は、プロセス定義が使用するインポートされたプロセスの **import** エントリー、および **relationship** エントリーも定義します。

プロセス定義の BPMN2 ソース

```
<definitions id="Definition"
             targetNamespace="http://www.jboss.org/drools"
             typeLanguage="http://www.java.com/javaTypes"
             expressionLanguage="http://www.mvel.org/2.0"
             xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"Rule Task
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd"
             xmlns:g="http://www.jboss.org/drools/flow/gpd"
             xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
             xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
             xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
             xmlns:tns="http://www.jboss.org/drools">

    <process>
      PROCESS
    </process>

    <bpmndi:BPMNDiagram>
     BPMN DIAGRAM DEFINITION
    </bpmndi:BPMNDiagram>

    </definitions>
```

ビジネスプロセスを含むプロジェクトを作成し、設定してデプロイした後に、Business Central の **Menu → Manage → Process Definitions** ですべてのプロセス定義の一覧を確認できます。右上隅のリフレッシュボタンをクリックすれば、デプロイしたプロセス定義の一覧をいつでもリフレッシュできます。

プロセス定義の一覧には、プラットフォームにデプロイした利用可能なすべてのプロセス定義が表示されます。各プロセス定義をクリックすると、対応するプロセス定義の詳細が表示されます。そのプロセスに関連するサブプロセスが存在するかどうか、プロセス定義にユーザーおよびグループがいくつ存在するかなど、プロセス定義の情報が表示されます。プロセス定義に関する詳細ページの **Diagram** タブには、プロセス定義の BPMN2 ベースのダイアグラムが含まれます。

選択したプロセス定義内からそれぞれ、右上隅の **New Process Instance** ボタンをクリックして、プロセス定義用の新規プロセスインスタンスを起動できます。利用可能なプロセスから起動したプロセスインスタンスは、**Menu → Manage → Process Instances** に一覧表示されます。

Manage ドロップダウンメニュー (**Process Definition**、**Process Instances**、**Tasks**、**Execution Errors** および **Jobs**) と **Menu → Track → Task Inbox** で、全ユーザーのデフォルトページネーションオプションを定義することも可能です。

Business Central でのプロセスおよびタスクの管理に関する詳細は、『Business Central でのビジネスプロセスの管理とモニタリング』を参照してください。

8.1. プロセス定義ページからのプロセスインスタンスの開始

Menu → Manage → Process Definitions からプロセスインスタンスを起動できます。これは、複数のプロジェクトやプロセス定義を同時に使用する環境で、有用です。

前提条件

- Business Central に、プロセス定義が設定されたプロジェクトがデプロイされている。

手順

1. Business Central で **Menu → Manage → Process Definitions** に移動します。
2. 一覧から、新しいプロセスインスタンスを開始するプロセス定義を選択します。定義の詳細ページが開きます。
3. 右上にある **New Process Instance** ボタンをクリックし、新しいプロセスインスタンスを開始します。
4. プロセスインスタンスに必要な情報を提供します。
5. **Submit** をクリックして、プロセスインスタンスを作成します。
6. **Menu → Manage → Process Instances** で新規プロセスインスタンスを表示します。

8.2. プロセスインスタンスページからプロセスインスタンスの開始

Menu → Manage → Process Instances で、新規プロセスインスタンスを作成するか、実行中のプロセスインスタンスの全リストを表示することができます。

前提条件

- Business Central に、プロセス定義が設定されたプロジェクトがデプロイされている。

手順

1. Business Central で、**Menu → Manage → Process Instances** に移動します。
2. 右上の **New Process Instance** ボタンをクリックし、ドロップダウンリストから新しいプロセスインスタンスを開始するプロセス定義を選択します。
3. 新しいプロセスインスタンスを開始するのに必要な情報を入力します。
4. **Start** をクリックして、プロセスインスタンスを作成します。
   Manage Process Instances 一覧に新しいプロセスインスタンスが表示されます。

8.3. XML でのプロセス定義

BPMN 2.0 仕様を使用して、XML 形式でプロセスを直接作成できます。これらの XML プロセスの構文は、BPMN 2.0 XML スキーマ定義を使用して定義されます。

プロセス XML ファイルは、以下のコアセクションで構成されます。

- **process**: これは、さまざまなノードとそのプロパティーの定義を含むプロセス XML の最上部になります。プロセス XML ファイルは、1 つの **<process>** 要素で構成されます。この要素には、プロセスに関連するパラメーター (そのタイプ、名前、ID、およびパッケージ名) が含まれ、3 つのサブセクションで構成されます。つまり、変数、グローバル、インポート、レーンなどのプロセスレベル情報が定義されるヘッダーセクション、プロセス内の各ノードを定義するノードセクション、およびプロセス内のすべてのノード間の接続を含む接続セクションの 3 つです。

- **BPMNDiagram**: これは、ノードの場所など、すべての描画情報を含むプロセス XML ファイルの下の部分になります。ノードセクションには、各ノードの特定の要素が含まれ、そのノードタイプのさまざまなパラメーターとサブ要素を定義します。

以下のプロセス XML ファイルのフラグメントは、開始イベント、**"Hello World"** をコンソールに出力するスクリプトタスク、および終了イベントを含むシンプルなプロセスを示しています。

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions
  id="Definition"
  targetNamespace="http://www.jboss.org/drools"
  typeLanguage="http://www.java.com/javaTypes"
  expressionLanguage="http://www.mvel.org/2.0"
  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd"
  xmlns:g="http://www.jboss.org/drools/flow/gpd"
  xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
  xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
  xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
  xmlns:tns="http://www.jboss.org/drools">

  <process processType="Private" isExecutable="true" id="com.sample.hello" name="Hello Process">
    <!-- nodes -->
    <startEvent id="_1" name="Start" />

    <scriptTask id="_2" name="Hello">
      <script>System.out.println("Hello World");</script>
    </scriptTask>

    <endEvent id="_3" name="End" >
      <terminateEventDefinition/>
    </endEvent>

    <!-- connections -->

    <sequenceFlow id="_1-_2" sourceRef="_1" targetRef="_2" />
    <sequenceFlow id="_2-_3" sourceRef="_2" targetRef="_3" />
  </process>

  <bpmndi:BPMNDiagram>
    <bpmndi:BPMNPlane bpmnElement="com.sample.hello" >

      <bpmndi:BPMNShape bpmnElement="_1" >
        <dc:Bounds x="16" y="16" width="48" height="48" />
```

```
    </bpmndi:BPMNShape>

    <bpmndi:BPMNShape bpmnElement="_2" >
      <dc:Bounds x="96" y="16" width="80" height="48" />
    </bpmndi:BPMNShape>

    <bpmndi:BPMNShape bpmnElement="_3" >
      <dc:Bounds x="208" y="16" width="48" height="48" />
    </bpmndi:BPMNShape>

    <bpmndi:BPMNEdge bpmnElement="_1-_2" >
      <di:waypoint x="40" y="40" />
      <di:waypoint x="136" y="40" />
    </bpmndi:BPMNEdge>

    <bpmndi:BPMNEdge bpmnElement="_2-_3" >
      <di:waypoint x="136" y="40" />
      <di:waypoint x="232" y="40" />
    </bpmndi:BPMNEdge>

  </bpmndi:BPMNPlane>
 </bpmndi:BPMNDiagram>

</definitions>
```

第9章 BUSINESS CENTRAL のフォーム

フォームは、HTML として定義されたページのレイアウト定義であり、プロセスおよびタスクのインスタンス化の間にユーザーにダイアログウィンドウとして表示されます。タスクフォームはプロセスとタスクインスタンスの両方の実行のためにユーザーからデータを取得しますが、プロセスフォームはプロセス変数から入力と出力を受け取ります。

入力は、Data Input Assignment を使用してタスクにマッピングされ、タスク内で使用できます。タスクが完了すると、データは Data Output Assignment としてマッピングされ、データを親プロセスインスタンスに提供します。

9.1. FORM MODELER

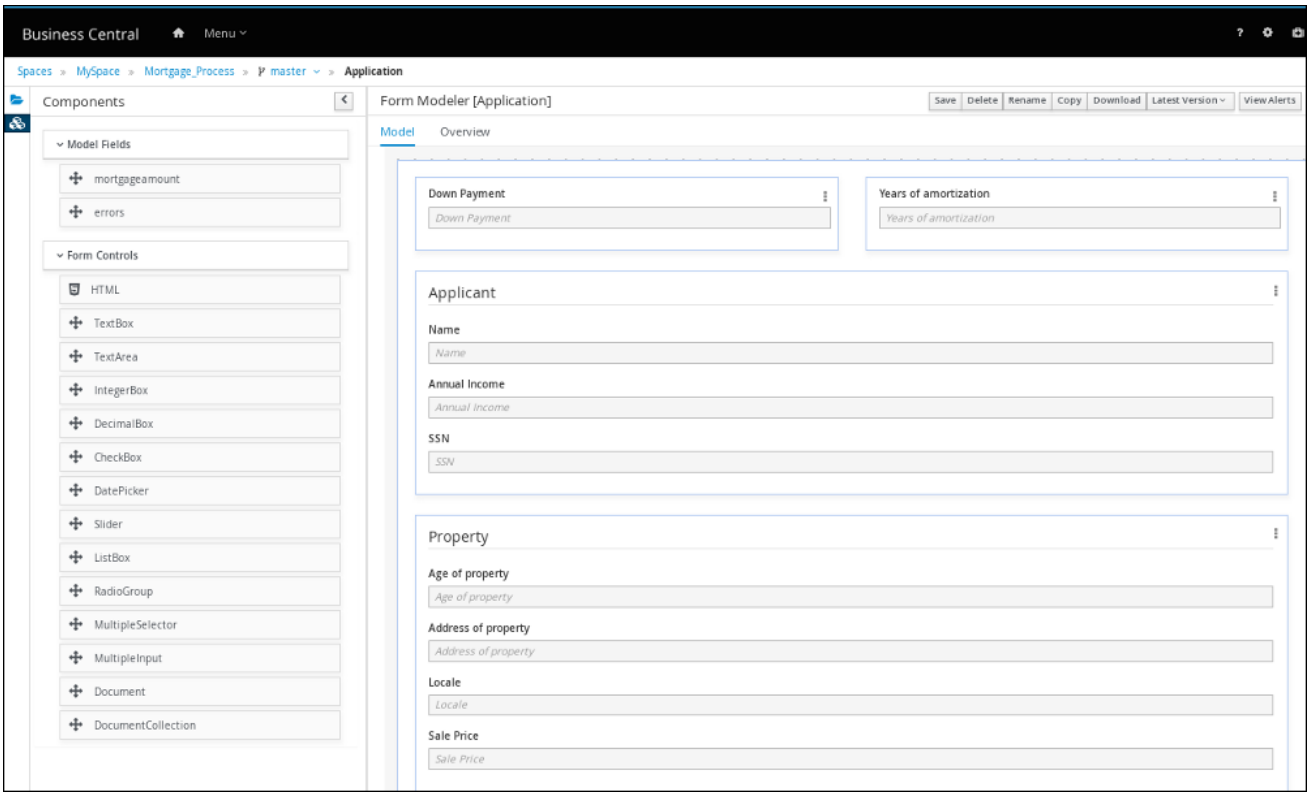
Red Hat Process Automation Manager は、Form Modeler と呼ばれるフォームを定義するためのカスタムエディターを提供します。Form Modeler を使用すると、コードを記述せずに、データオブジェクトのフォーム、タスクフォーム、およびプロセス開始フォームを生成できます。Form Modeler には、複数のデータタイプをバインドするためのウィジェットライブラリーと、フォームの値が変更されたときに通知を送信するコールバックメカニズムが含まれています。Form Modeler は、Bean ベースの検証を使用し、フォームフィールドの静的または動的モデルへのバインドをサポートします。

Form Modeler には以下の機能が含まれています。

- フォームのフォームモデリングユーザーインターフェース
- データモデルまたは Java オブジェクトからのフォーム自動生成
- Java オブジェクトのデータバインディング
- 公式と式
- カスタマイズされたフォームレイアウト
- 埋め込みフォーム

Form Modeler には、フォームを作成するためにキャンバスに配置する定義済みのフィールドタイプが付属します。

図9.1 住宅ローンの申し込みフォームの例



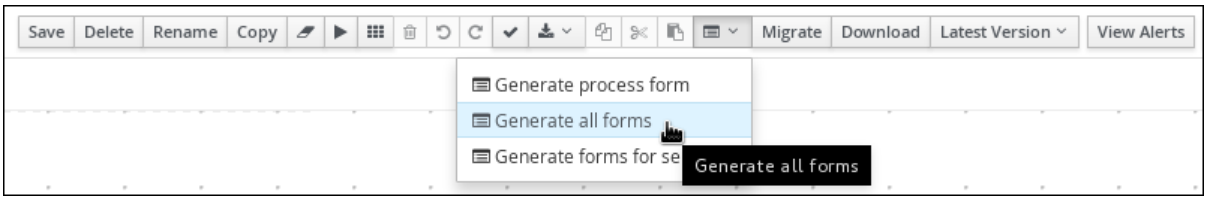
9.2. BUSINESS CENTRAL でのプロセスフォームおよびタスクフォームの生成

プロセスをインスタンス化したときに、プロセスをインスタンス化したユーザーに表示されるビジネスプロセスからプロセスフォームを生成できます。また、ビジネスプロセスからタスクフォームを生成することもできます。このタスクフォームは、実行フローがタスクに到達すると、ユーザータスクのインスタンス化時にユーザータスクのアクターに表示されます。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。
2. プロジェクト名をクリックしてアセットビューを開いてから、ビジネスプロセス名をクリックします。
3. プロセスデザイナーで、フォームを作成するプロセスタスクをクリックします (該当する場合)。
4. 右上のツールバーで、**Form Generation** アイコンをクリックして、生成するフォームを選択します。
   - **Generate process form**: プロセス全体のフォームを生成します。これは、プロセスインスタンスの起動時にユーザーが入力する必要がある初期フォームです。
   - **Generate all forms**: プロセス全体およびすべてのユーザータスクのフォームを生成します。
   - **Generate forms for selection**: 選択したユーザータスクノードのフォームを生成します。

図9.2 フォーム生成メニュー



フォームは、プロジェクトのルートディレクトリーに作成されます。

5. Business Central で、プロジェクトのルートディレクトリーに移動して新しいフォーム名をクリックし、Form Modeler を使用して要件に合わせてフォームをカスタマイズします。

9.3. BUSINESS CENTRAL での手動によるフォームの作成

プロジェクトアセットビューから、タスクおよびプロセスフォームを手動で作成できます。これは、ビジネスプロセスからフォームを生成することを選択せずにフォームを生成する別の方法です。たとえば、Form Modeler は外部データオブジェクトからのフォームの作成をサポートするようになりました。

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. **Add Asset → Form** をクリックします。
3. **Create new Form** ウィンドウで、以下の情報を入力します。
   - フォーム名 (一意である必要があります)
   - パッケージ名
   - モデルタイプ: **Business Process** または **Data Object** のいずれかを選択します。
     - **Business Process** モデルタイプの場合、**Select Process** ドロップダウンメニューからビジネスプロセスを選択し、**Select Form** ドロップダウンメニューから作成するフォームを選択します。
     - **Data Object** モデルタイプの場合、**Select Data Object from Project** のドロップダウンメニューから、プロジェクトデータオブジェクトの 1 つを選択します。
4. **OK** をクリックして、Form Modeler を開きます。
5. Form Modeler の左側の **Components** ビューで、**Model Fields** および **Form Controls** メニューを展開し、必要なフィールドとフォームコントロールをキャンバスにドラッグして新しいフォームを作成します。
6. **Save** をクリックして変更を保存します。

9.4. フォームまたはプロセスでのドキュメントの添付

Red Hat Process Automation Manager は、**Document** フォームフィールドを使用したフォームでのドキュメントの添付をサポートしています。**Document** フォームフィールドを使用すると、フォームまたはプロセスの一部として必要なドキュメントをアップロードできます。

フォームおよびプロセスでドキュメントの添付を有効にするには、以下の手順を実行します。

1. ドキュメントマーシャリング戦略を設定します。
2. ビジネスプロセスでドキュメント変数を作成します。
3. タスクの入力と出力をドキュメント変数にマッピングします。

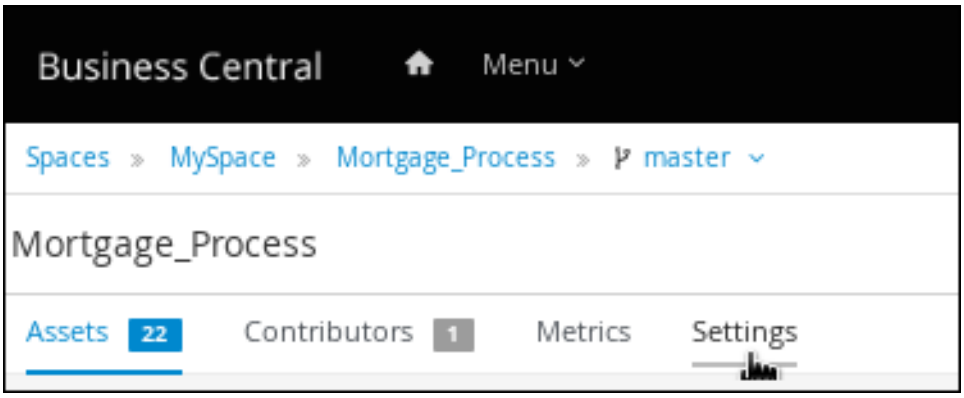
9.4.1. ドキュメントマーシャリング戦略の設定

プロジェクトのドキュメントマーシャリング戦略により、フォームおよびプロセスで使用するドキュメントの保存場所が決まります。Red Hat Process Automation Manager のデフォルトのドキュメントマーシャリング戦略は **org.jbpm.document.marshalling.DocumentMarshallingStrategy** です。この戦略では、**PROJECT_HOME/docs** フォルダーにドキュメントをローカルに保存する **DocumentStorageServiceImpl** クラスを使用します。Business Central または **kie-deployment-descriptor.xml** ファイルのプロジェクトに、このドキュメントマーシャリング戦略またはカスタムドキュメントマーシャリング戦略を設定できます。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。
2. プロジェクトを選択します。プロジェクトの **Assets** ウィンドウが開きます。
3. プロジェクトの **Settings** タブをクリックします。

図9.3 設定タブ



4. **Deployments → Marshalling Strategies→ Add Marshalling Strategy** をクリックします。
5. **Name** フィールドにドキュメントマーシャリング戦略の識別子を入力し、**Resolver** のドロップダウンメニューで、対応するリゾルバータイプを選択します。
   - 1 つのドキュメントの場合: ドキュメントマーシャリング戦略として **org.jbpm.document.marshalling.DocumentMarshallingStrategy** を入力し、リゾルバータイプを **Reflection** に設定します。
   - 複数のドキュメントの場合: ドキュメントマーシャリング戦略として **new org.jbpm.document.marshalling.DocumentCollectionImplMarshallingStrategy(new org.jbpm.document.marshalling.DocumentMarshallingStrategy())** を入力し、リゾルバータイプを **MVEL** に設定します。
   - カスタムドキュメントサポートの場合: カスタムドキュメントマーシャリング戦略の識別子を入力し、関連するリゾルバータイプを選択します。

6. **Test** をクリックして、デプロイメント記述子ファイルを検証します。

7. **Deploy** をクリックして、更新されたプロジェクトをビルドおよびデプロイします。
   または、Business Central を使用していない場合は、**PROJECT_HOME/src/main/resources/META_INF/kie-deployment-descriptor.xml** (該当する場合) に移動し、必要な **<marshalling-strategies>** 要素を使用してデプロイメント記述子ファイルを編集します。

8. **Save** をクリックします。

複数のドキュメントのドキュメントマーシャリング戦略を使用したデプロイメント記述子ファイルの例

```
<deployment-descriptor
    xsi:schemaLocation="http://www.jboss.org/jbpm deployment-descriptor.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <persistence-unit>org.jbpm.domain</persistence-unit>
  <audit-persistence-unit>org.jbpm.domain</audit-persistence-unit>
  <audit-mode>JPA</audit-mode>
  <persistence-mode>JPA</persistence-mode>
  <runtime-strategy>SINGLETON</runtime-strategy>
  <marshalling-strategies>
    <marshalling-strategy>
      <resolver>mvel</resolver>
      <identifier>new org.jbpm.document.marshalling.DocumentCollectionImplMarshallingStrategy(new
org.jbpm.document.marshalling.DocumentMarshallingStrategy());</identifier>
    </marshalling-strategy>
  </marshalling-strategies>
```

9.4.1.1. コンテンツ管理システム (CMS) にカスタムドキュメントマーシャリング戦略を使用する

プロジェクトのドキュメントマーシャリング戦略により、フォームおよびプロセスで使用するドキュメントの保存場所が決まります。Red Hat Process Automation Manager のデフォルトのドキュメントマーシャリング戦略は **org.jbpm.document.marshalling.DocumentMarshallingStrategy** です。この戦略では、**PROJECT_HOME/docs** フォルダーにドキュメントをローカルに保存する **DocumentStorageServiceImpl** クラスを使用します。集中型のコンテンツ管理システム (CMS) などのカスタムの場所にフォームおよびプロセスドキュメントを保存する場合は、カスタムドキュメントマーシャリング戦略をプロジェクトに追加します。このドキュメントマーシャリング戦略は、Business Central または **kie-deployment-descriptor.xml** ファイルで直接設定できます。

手順

1. **org.kie.api.marshalling.ObjectMarshallingStrategy** インターフェースの実装を含むカスタムマーシャリング戦略 **.java** ファイルを作成します。このインターフェースを使用すると、カスタムドキュメントマーシャリング戦略に必要な変数の永続性を実装できます。
   このインターフェースの以下のメソッドは、戦略の作成に役立ちます。

   - **boolean accept(Object object)**: 指定されたオブジェクトを戦略でマーシャリングできるかどうかを決定します

   - **byte[] marshal(Context context, ObjectOutputStream os, Object object)**: 指定されたオブジェクトをマーシャリングし、マーシャリングされたオブジェクトを **byte[]** として返します

- **Object unmarshal(Context context, ObjectInputStream is, byte[] object, ClassLoader classloader)**: 受信したオブジェクトを **byte[]** として読み取り、マーシャリングされていないオブジェクトを返します
- **void write(ObjectOutputStream os, Object object)**: 下位互換性のために提供されている **marshal** メソッドと同じです
- **Object read(ObjectInputStream os)**: 下位互換性のために提供されている **unmarshal** メソッドと同じです

以下のコードサンプルは、コンテンツ管理相互運用サービス (CMIS) システムからデータを保存および取得するための **ObjectMarshallingStrategy** 実装例です。

CMIS システムからデータを保存および取得するための実装例

```
package org.jbpm.integration.cmis.impl;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.HashMap;

import org.apache.chemistry.opencmis.client.api.Folder;
import org.apache.chemistry.opencmis.client.api.Session;
import org.apache.chemistry.opencmis.commons.data.ContentStream;
import org.apache.commons.io.IOUtils;
import org.drools.core.common.DroolsObjectInputStream;
import org.jbpm.document.Document;
import org.jbpm.integration.cmis.UpdateMode;

import org.kie.api.marshalling.ObjectMarshallingStrategy;

public class OpenCMISPlaceholderResolverStrategy extends OpenCMISSupport implements
ObjectMarshallingStrategy {

 private String user;
 private String password;
 private String url;
 private String repository;
 private String contentUrl;
 private UpdateMode mode = UpdateMode.OVERRIDE;

 public OpenCMISPlaceholderResolverStrategy(String user, String password, String url,
String repository) {
  this.user = user;
  this.password = password;
  this.url = url;
  this.repository = repository;
 }

 public OpenCMISPlaceholderResolverStrategy(String user, String password, String url,
String repository, UpdateMode mode) {
  this.user = user;
  this.password = password;
```

```
  this.url = url;
  this.repository = repository;
  this.mode = mode;
 }

    public OpenCMISPlaceholderResolverStrategy(String user, String password, String url,
String repository, String contentUrl) {
        this.user = user;
        this.password = password;
        this.url = url;
        this.repository = repository;
        this.contentUrl = contentUrl;
    }

    public OpenCMISPlaceholderResolverStrategy(String user, String password, String url,
String repository, String contentUrl, UpdateMode mode) {
        this.user = user;
        this.password = password;
        this.url = url;
        this.repository = repository;
        this.contentUrl = contentUrl;
        this.mode = mode;
    }

 public boolean accept(Object object) {
  if (object instanceof Document) {
   return true;
  }
  return false;
 }

 public byte[] marshal(Context context, ObjectOutputStream os, Object object) throws
IOException {
  Document document = (Document) object;
  Session session = getRepositorySession(user, password, url, repository);
  try {
   if (document.getContent() != null) {
    String type = getType(document);
    if (document.getIdentifier() == null || document.getIdentifier().isEmpty()) {
     String location = getLocation(document);

     Folder parent = findFolderForPath(session, location);
     if (parent == null) {
      parent = createFolder(session, null, location);
     }
     org.apache.chemistry.opencmis.client.api.Document doc = createDocument(session,
parent, document.getName(), type, document.getContent());
     document.setIdentifier(doc.getId());
     document.addAttribute("updated", "true");
    } else {
     if (document.getContent() != null && "true".equals(document.getAttribute("updated"))) {
      org.apache.chemistry.opencmis.client.api.Document doc = updateDocument(session,
document.getIdentifier(), type, document.getContent(), mode);

      document.setIdentifier(doc.getId());
      document.addAttribute("updated", "false");
```

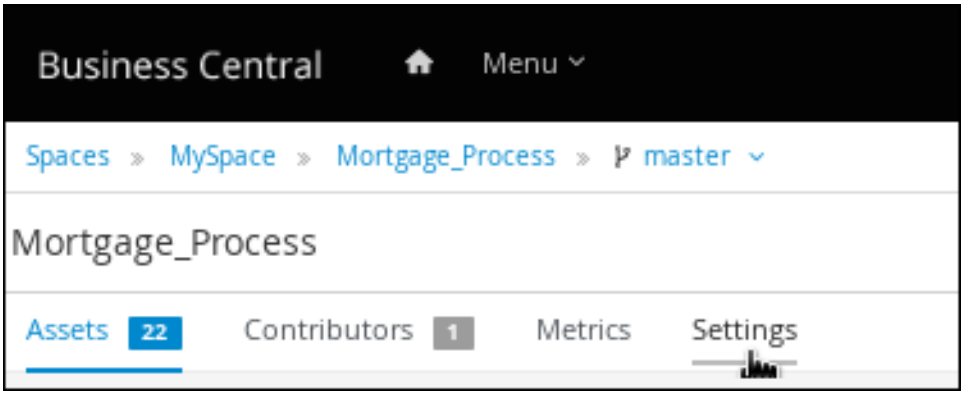
```
    }
   }
  }
  ByteArrayOutputStream buff = new ByteArrayOutputStream();
       ObjectOutputStream oos = new ObjectOutputStream( buff );
       oos.writeUTF(document.getIdentifier());
       oos.writeUTF(object.getClass().getCanonicalName());
       oos.close();
       return buff.toByteArray();
 } finally {
  session.clear();
 }
}

 public Object unmarshal(Context context, ObjectInputStream ois, byte[] object, ClassLoader
classloader) throws IOException, ClassNotFoundException {
  DroolsObjectInputStream is = new DroolsObjectInputStream( new ByteArrayInputStream(
object ), classloader );
  String objectId = is.readUTF();
  String canonicalName = is.readUTF();
  Session session = getRepositorySession(user, password, url, repository);
  try {
   org.apache.chemistry.opencmis.client.api.Document doc =
(org.apache.chemistry.opencmis.client.api.Document) findObjectForId(session, objectId);
   Document document = (Document) Class.forName(canonicalName).newInstance();
   document.setAttributes(new HashMap<String, String>());

   document.setIdentifier(objectId);
   document.setName(doc.getName());
   document.setLastModified(doc.getLastModificationDate().getTime());
   document.setSize(doc.getContentStreamLength());
   document.addAttribute("location", getFolderName(doc.getParents()) +
getPathAsString(doc.getPaths()));
   if (doc.getContentStream() != null && contentUrl == null) {
    ContentStream stream = doc.getContentStream();
    document.setContent(IOUtils.toByteArray(stream.getStream()));
    document.addAttribute("updated", "false");
    document.addAttribute("type", stream.getMimeType());
   } else {
      document.setLink(contentUrl + document.getIdentifier());
   }
   return document;
  } catch(Exception e) {
   throw new RuntimeException("Cannot read document from CMIS", e);
  } finally {
   is.close();
   session.clear();
  }
 }

 public Context createContext() {
  return null;
 }

 // For backward compatibility with previous serialization mechanism
 public void write(ObjectOutputStream os, Object object) throws IOException {
```

```
  Document document = (Document) object;
  Session session = getRepositorySession(user, password, url, repository);
  try {
   if (document.getContent() != null) {
    String type = document.getAttribute("type");
    if (document.getIdentifier() == null) {
     String location = document.getAttribute("location");

     Folder parent = findFolderForPath(session, location);
     if (parent == null) {
      parent = createFolder(session, null, location);
     }
     org.apache.chemistry.opencmis.client.api.Document doc = createDocument(session,
parent, document.getName(), type, document.getContent());
     document.setIdentifier(doc.getId());
     document.addAttribute("updated", "false");
    } else {
     if (document.getContent() != null && "true".equals(document.getAttribute("updated"))) {
      org.apache.chemistry.opencmis.client.api.Document doc = updateDocument(session,
document.getIdentifier(), type, document.getContent(), mode);

      document.setIdentifier(doc.getId());
      document.addAttribute("updated", "false");
     }
    }
   }
   ByteArrayOutputStream buff = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream( buff );
        oos.writeUTF(document.getIdentifier());
        oos.writeUTF(object.getClass().getCanonicalName());
        oos.close();
  } finally {
   session.clear();
  }
 }

 public Object read(ObjectInputStream os) throws IOException, ClassNotFoundException {
  String objectId = os.readUTF();
  String canonicalName = os.readUTF();
  Session session = getRepositorySession(user, password, url, repository);
  try {
   org.apache.chemistry.opencmis.client.api.Document doc =
(org.apache.chemistry.opencmis.client.api.Document) findObjectForId(session, objectId);
   Document document = (Document) Class.forName(canonicalName).newInstance();

   document.setIdentifier(objectId);
   document.setName(doc.getName());
   document.addAttribute("location", getFolderName(doc.getParents()) +
getPathAsString(doc.getPaths()));
   if (doc.getContentStream() != null) {
    ContentStream stream = doc.getContentStream();
    document.setContent(IOUtils.toByteArray(stream.getStream()));
    document.addAttribute("updated", "false");
    document.addAttribute("type", stream.getMimeType());
   }
   return document;
```

```
  } catch(Exception e) {
   throw new RuntimeException("Cannot read document from CMIS", e);
  } finally {
   session.clear();
  }
 }

}
```

2. Business Central で、**Menu → Design → Projects** に移動します。
3. プロジェクトの名前をクリックしてから、**Settings** をクリックします。

図9.4 設定タブ



4. **Deployments → Marshalling Strategies→ Add Marshalling Strategy** をクリックします。
5. **Name** フィールドに、この例の **org.jbpm.integration.cmis.impl.OpenCMISPlaceholderResolverStrategy** のように、カスタムドキュメントマーシャリング戦略の識別子を入力します。
6. この例の **Reflection** のように、**Resolver** ドロップダウンメニューから関連オプションを選択します。
7. **Test** をクリックして、デプロイメント記述子ファイルを検証します。
8. **Deploy** をクリックして、更新されたプロジェクトをビルドおよびデプロイします。
または、Business Central を使用していない場合は、**PROJECT_HOME/src/main/resources/META_INF/kie-deployment-descriptor.xml** (該当する場合) に移動し、必要な **<marshalling-strategies>** 要素を使用してデプロイメント記述子ファイルを編集します。

カスタムドキュメントマーシャリング戦略を使用したデプロイメント記述子ファイルの例

```
<deployment-descriptor
   xsi:schemaLocation="http://www.jboss.org/jbpm deployment-descriptor.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <persistence-unit>org.jbpm.domain</persistence-unit>
  <audit-persistence-unit>org.jbpm.domain</audit-persistence-unit>
  <audit-mode>JPA</audit-mode>
  <persistence-mode>JPA</persistence-mode>
  <runtime-strategy>SINGLETON</runtime-strategy>
  <marshalling-strategies>
    <marshalling-strategy>
```

```
    <resolver>reflection</resolver>
    <identifier>
      org.jbpm.integration.cmis.impl.OpenCMISPlaceholderResolverStrategy
    </identifier>
  </marshalling-strategy>
</marshalling-strategies>
```

9. カスタムの場所に保存されたドキュメントをフォームおよびプロセスに添付できるようにするには、関連するプロセスでドキュメント変数を作成し、Business Central でそのドキュメント変数にタスクの入力と出力をマッピングします。

9.4.2. ビジネスプロセスでのドキュメント変数の作成

ドキュメントマーシャリング戦略を設定したら、関連プロセスでドキュメント変数を作成して、ドキュメントをヒューマンタスクにアップロードし、Business Central の **Process Instances** ビューにドキュメントを表示できるようにします。

前提条件

- 「ドキュメントマーシャリング戦略の設定」 の説明に従って、ドキュメントマーシャリング戦略を設定済みである。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。
2. プロジェクト名をクリックしてアセットビューを開き、ビジネスプロセス名をクリックします。
3. キャンバスをクリックし、ウィンドウの右側にある をクリックして、**Diagram properties** パネルを開きます。
4. **Process Data** を展開し、 をクリックして、以下の値を入力します。
   - Name: **document**
   - Custom Type: 1 つのドキュメントの場合は **org.jbpm.document.Document**、複数のドキュメントの場合は **org.jbpm.document.DocumentCollection**。

9.4.3. ドキュメント変数へのタスクの入力と出力のマッピング

タスクフォーム内の添付ファイルを表示または変更する場合は、タスクの入力および出力内に割り当てを作成します。

前提条件

- 1 つ以上のユーザータスクを持つビジネスプロセスアセットを含むプロジェクトがある。

手順

1. Business Central で、**Menu → Design → Projects** に移動します。
2. プロジェクト名をクリックしてアセットビューを開き、ビジネスプロセス名をクリックします。

3. ユーザータスクをクリックし、ウィンドウの右側にある をクリックして **Diagram properties** パネルを開きます。

4. **Implementation/Execution** を展開し、**Assignments** の横の  をクリックして、**Data I/O** ウィンドウを開きます。

5. **Data Inputs and Assignments**の横の **Add** をクリックして、以下の値を入力します。

   - Name: **taskdoc_in**
   - Data Type: 1つのドキュメントの場合は **org.jbpm.document.Document**、複数のドキュメントの場合は **org.jbpm.document.DocumentCollection**。
   - Source: **document**

6. **Data Outputs and Assignments** の横の **Add** をクリックし、以下の値を入力します。

   - Name: **taskdoc_out**
   - Data Type: 1つのドキュメントの場合は **org.jbpm.document.Document**、複数のドキュメントの場合は **org.jbpm.document.DocumentCollection**。
   - Target: **document**

   Source および **Target** フィールドには、前に作成したプロセス変数の名前が含まれています。

7. **Save** をクリックします。

第10章 詳細にわたるプロセスの概念およびタスク

10.1. ビジネスプロセスで DECISION MODEL AND NOTATION (DMN) サービスを呼び出す

Decision Model and Notation (DMN) を使用して、Business Central の意思決定要件ダイアグラム (DRD) でデシジョンサービスを描画を使ってモデル化し、Business Central のビジネスプロセスの一環としてそのDMN サービスを呼び出すことができます。ビジネスプロセスは、DMN サービスを識別し、DMN 入力とビジネスプロセスプロパティー間でビジネスデータをマッピングすることにより、DMN サービスと対話します。

例として、この手順では、列車の経路ロジックを定義する **TrainStation** プロジェクトの例を使用します。このサンプルプロジェクトには、経路決定ロジック用に Business Central で設計された以下のデータオブジェクトと DMN コンポーネントが含まれています。

Train オブジェクトの例

```
public class Train {

    private String departureStation;

    private String destinationStation;

    private BigDecimal railNumber;

    // Getters and setters
}
```

図10.1 Compute Rail DMN モデルの例

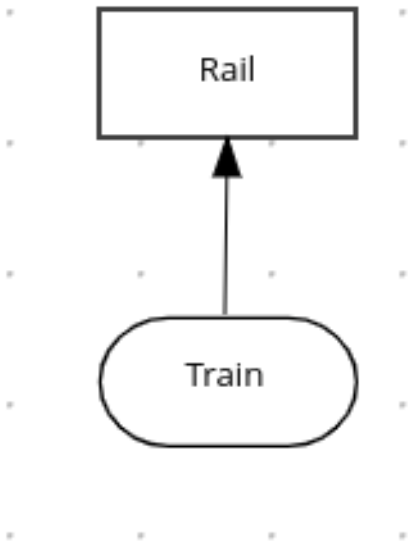
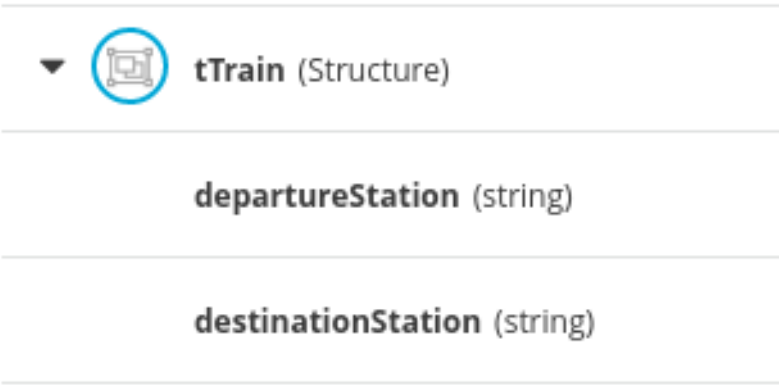


図10.2 Rail DMN デシジョンテーブルの例

Rail *(Decision Table)*

F	Train.departureStation *(string)*	Train.destinationStation *(string)*	Rail *(number)*	Description
1	"Prague"	"Hamburg"	5	
2	"Prague"	"Krakow"	2	
3	-	"Belgrade"	1	Just one option to Belgrade
4	"Zagreb"	-	3	Just one option from Zagreb
5	"Graz"	"Vienna"	1	

図10.3 tTrain DMN データタイプの例



Business Central での DMN モデルの作成方法に関する詳細は、『DMN モデ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。

前提条件

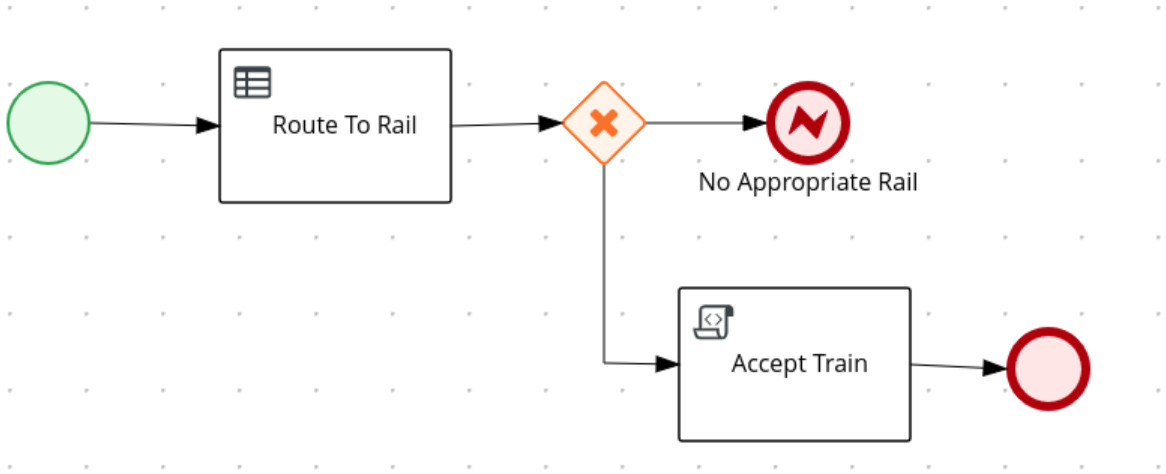
- 必要なすべてのデータオブジェクトと DMN モデルコンポーネントは、プロジェクトで定義済みである。

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. DMN サービスを呼び出すビジネスプロセスアセットを選択または作成します。
3. プロセスデザイナーで、左側のツールバーを使用して通常どおりに BPMN コンポーネントをドラッグアンドドロップし、ビジネスプロセスロジック、接続、イベント、タスク、またはその他の要素全体を定義します。

4. ビジネスプロセスに DMN サービスを組み込むには、左側のツールバーまたは開始ノードオプションから **Business Rule** タスクを追加し、プロセスフローの関連する場所にタスクを挿入します。
この例では、以下の **Accept Train** ビジネスプロセスは、DMN サービスを **Route To Rail** ノードに組み込みます。

図10.4 DMN サービスを使用した Accept Train ビジネスプロセスの例

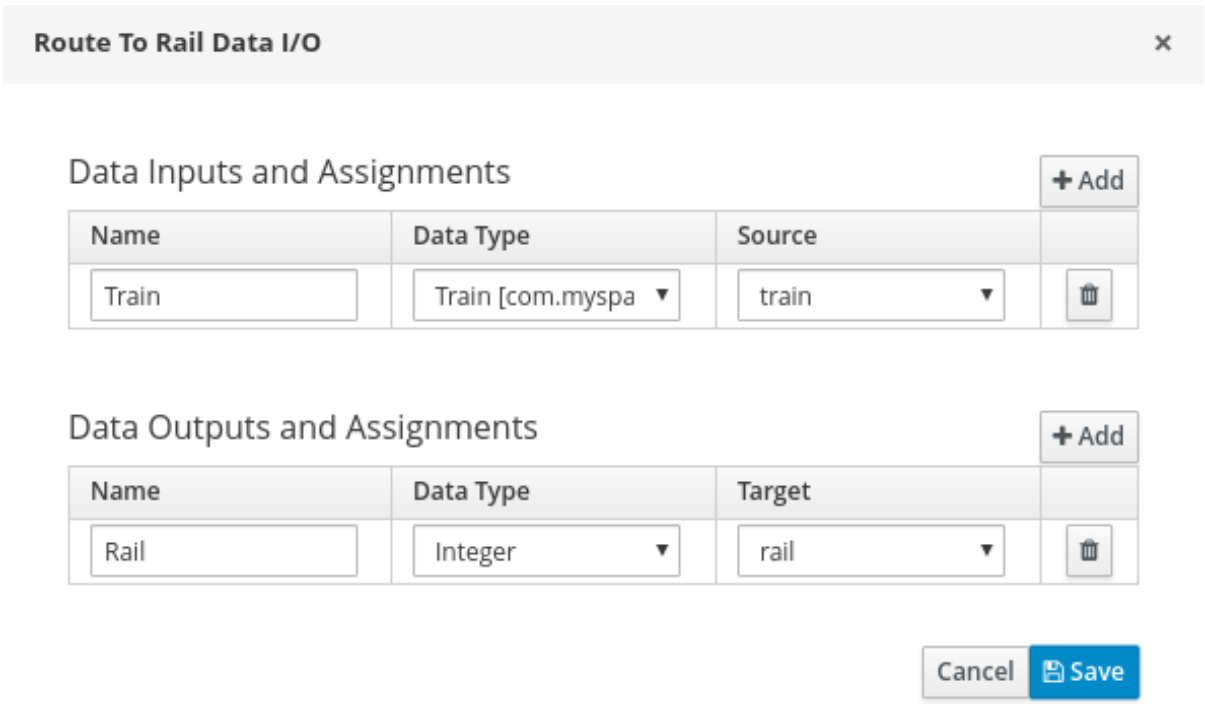


5. DMN サービスに使用するビジネスルールタスクノードを選択し、プロセスデザイナーの右上隅にある **Diagram properties** をクリックしてから、**Implementation/Execution** で、以下のフィールドを定義します。

- Rule Language: **DMN** を選択します。
- **名前空間**: DMN モデルファイルから一意の名前空間を入力します。例: **https://www.drools.org/kie-dmn**
- Decision Name: 選択したプロセスノードで呼び出す DMN デシジョンノードの名前を入力します。例: **Rail**
- DMN Model Name: DMN モデル名を入力します。例: **Compute Rail**

6. **Data Assignments → Assignments** 配下で、**Edit** アイコンをクリックし、DMN の入力および出力データを追加して、DMN サービスとプロセスデータ間のマッピングを定義します。
この例の **Route To Rail** DMN サービスノードの場合、DMN モデルの入力ノードに対応する **Train** の入力割り当てを追加し、DMN モデルのデシジョンノードに対応する **Rail** の出力割り当てを追加します。**Data Type** は、DMN モデルでそのノードに設定したタイプに一致する必要があり、**Source** および **Target** の定義は、指定されたオブジェクトに関連する変数またはフィールドです。

図10.5 Route To Rail DMN サービスノードの入力および出力マッピングの例



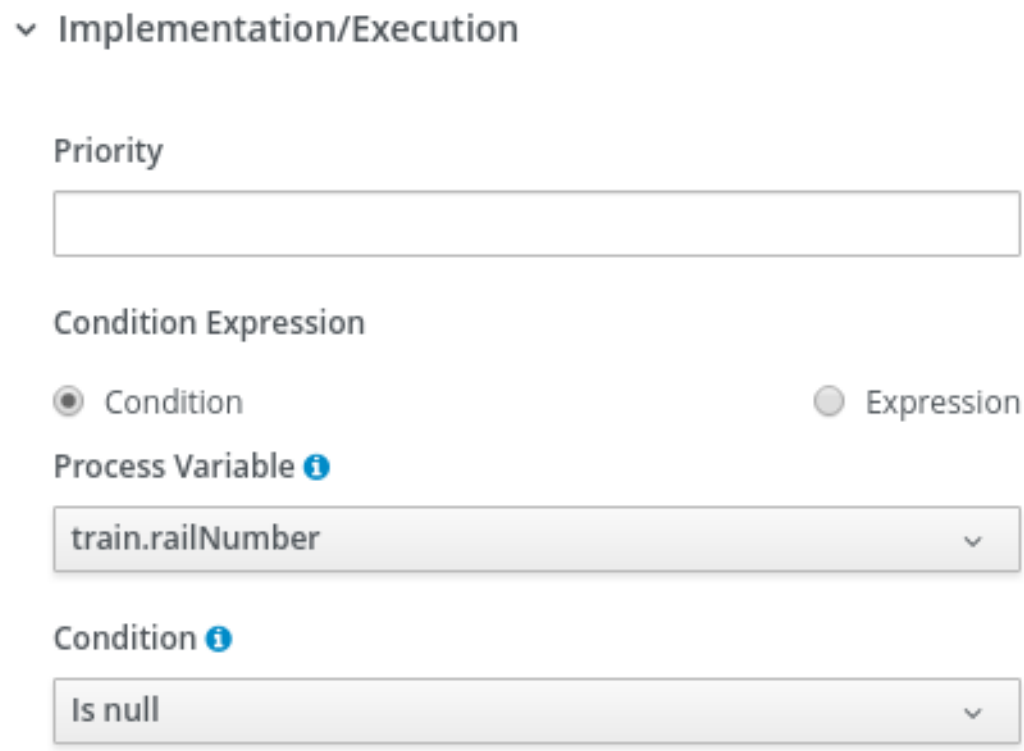
7. **Save** をクリックして、入力および出力データを保存します。
8. 完了した DMN サービスの処理方法に応じて、ビジネスプロセスの残りを定義します。
   この例では、**Diagram properties → Implementation/Execution → On Exit Action** の値が以下のコードに設定され、**Route To Rail** DMN サービスの完了後にレール番号を保存します。

   On Exit Action のサンプルコード

   ```
   train.setRailNumber(rail);
   ```

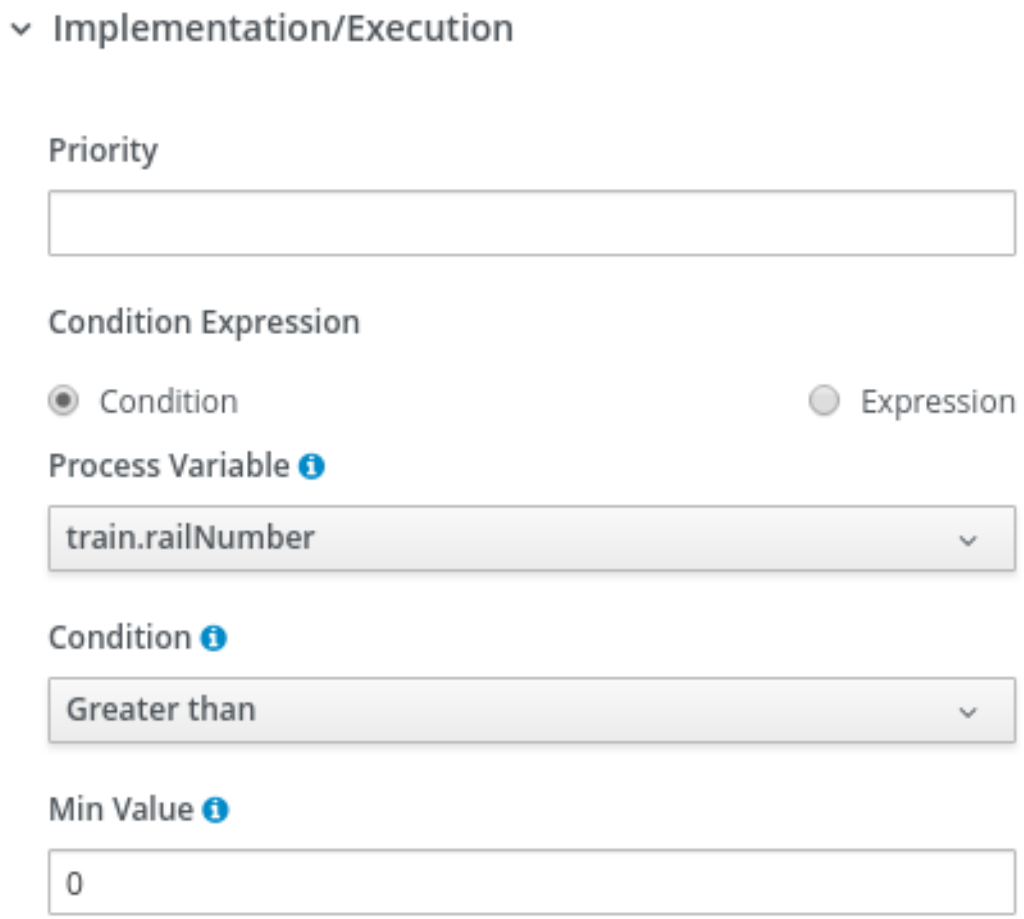
   レール番号が計算されない場合、プロセスは、以下の条件式で定義された **No Appropriate Rail** 終了エラーノードに到達します。

図10.6 No Appropriate Rail 終了エラーノードの条件の例



レール番号が計算されると、プロセスは、以下の条件式で定義された **Accept Train** スクリプトタスクに到達します。

図10.7 Accept Train スクリプトタスクノードの条件の例

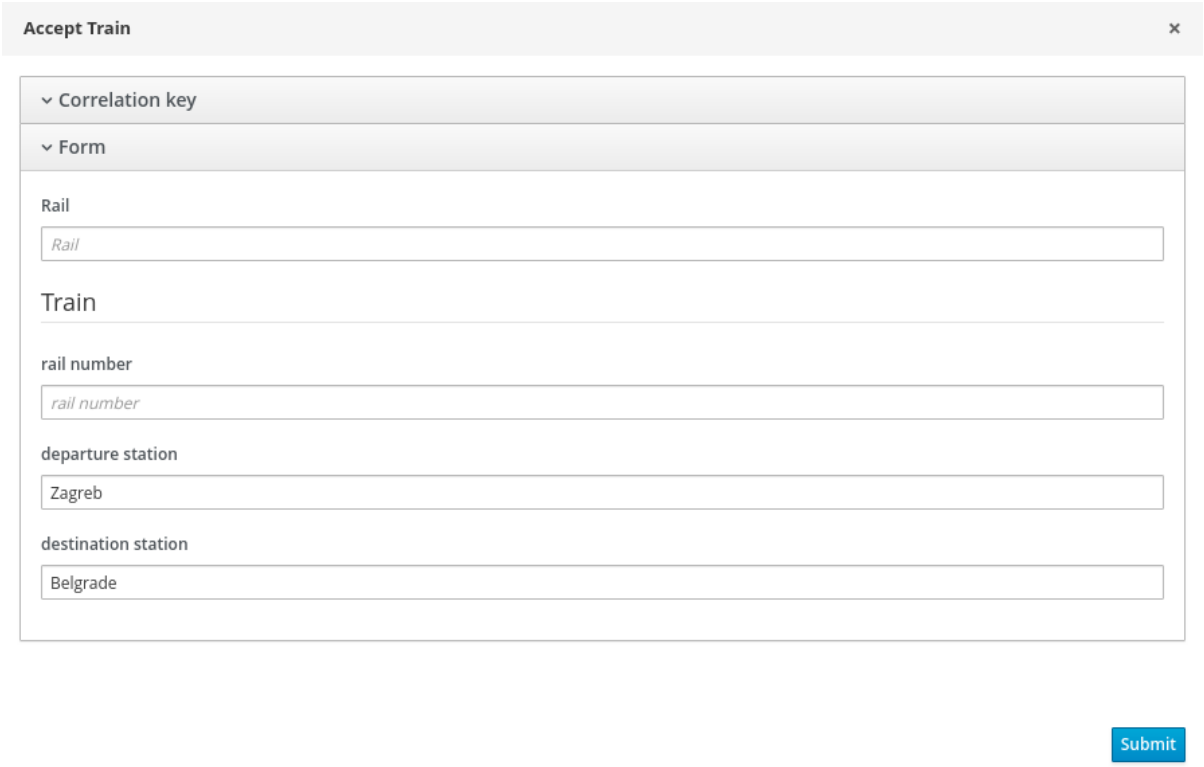


Accept Train スクリプトタスクは、**Diagram properties → Implementation/Execution → Script** で以下のスクリプトも使用して、列車のルートと現在のレールに関するメッセージを出力します。

```
com.myspace.trainstation.Train t =
    (com.myspace.trainstation.Train) kcontext.getVariable("train");
System.out.println("Train from: " + t.getDepartureStation() +
                   ", to: " + t.getDestinationStation() +
                   ",  is on rail: " + t.getRailNumber());
```

9. 組み込まれた DMN サービスでビジネスプロセスを定義した後、プロセスデザイナーでプロセスを保存してプロジェクトをデプロイし、対応するプロセス定義を実行して DMN サービスを呼び出します。
この例では、 **TrainStation** プロジェクトをデプロイし、対応するプロセス定義を実行するときに、**Accept Train** プロセス定義のプロセスインスタンスフォームを開いて **departure station** および **destination station** フィールドを設定して実行をテストします。

図10.8 Accept Train プロセス定義のプロセスインスタンスフォームの例



プロセスが実行されると、サーバーログに指定した列車のルートを示すメッセージが表示されます。

Accept Train プロセスのサーバーログ出力の例

```
Train from: Zagreb, to: Belgrade,  is on rail: 1
```

第11章 関連資料

- 『ビジネスプロセスの使用ガイド』
- 『Process designer Business Process Model and Notation (BPMN2) リファレンスガイド』
- 『Business Central でのビジネスプロセスの管理とモニタリング』
- 『プロセスおよびタスクとの対話』

付録A バージョン情報

本ドキュメントの最終更新日: 2020 年 3 月 18 日 (水)