Red Hat Process Automation Manager 7.8

Designing business processes in Business Central

Red Hat Process Automation Manager 7.8 Designing business processes in Business Central

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

Legal Notice

Abstract

This document describes concepts and options for designing business processes in Red Hat Process Automation Manager 7.8.

Table of Contents

PREFACE

As a business processes developer, you can use Business Central in Red Hat Process Automation Manager to design business processes to meet specific business requirements. This document describes business processes and the concepts and options for creating them using the process designer in Red Hat Process Automation Manager. This document also describes the BPMN2 elements in Red Hat Process Automation Manager. For more details about BPMN2, see the Business Process Model and Notation Version 2.0 specification.

Prerequisites

- Red Hat JBoss Enterprise Application Platform 7.3 is installed. For details, see *Red Hat JBoss Enterprise Application Platform 7.3 Installation Guide*.

- Red Hat Process Automation Manager is installed and configured with KIE Server. For more information, see *Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.3*.

- Red Hat Process Automation Manager is running and you can log in to Business Central with the **developer** role. For more information, see *Planning a Red Hat Process Automation Manager installation*.

CHAPTER 1. BUSINESS PROCESSES

A business process is a diagram that describes the order for a series of steps that must be executed and consists of predefined nodes and connections. Each node represents one step in the process while the connections specify how to transition from one node to another.

A typical business process consists of the following components:

- The header section that comprises global elements such as the name of the process, imports, and variables

- The nodes section that contains all the different nodes that are part of the process

- The connections section that links these nodes to each other to create a flow chart

Figure 1.1. Business process
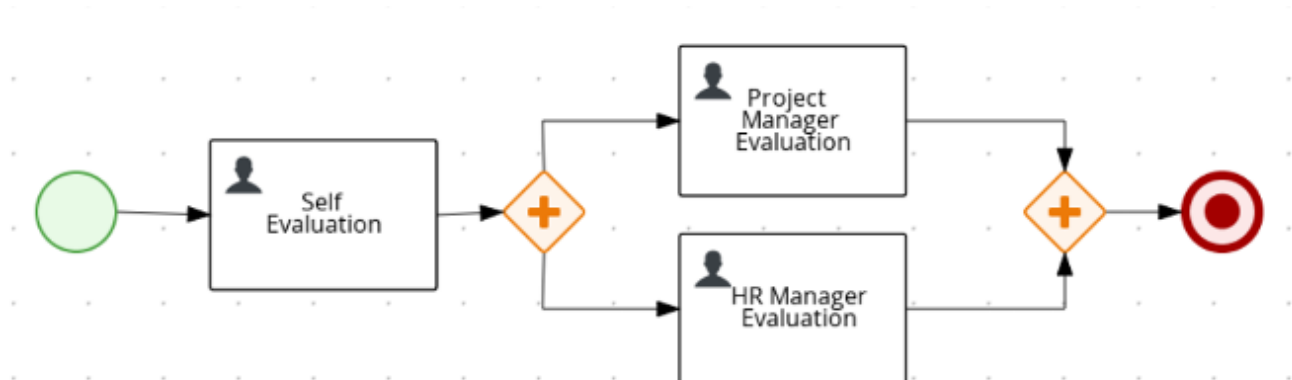
Red Hat Process Automation Manager contains the legacy process designer and the new process designer for creating business process diagrams. The new process designer has an improved layout and feature set and continues to be developed. Until all features of the legacy process designer are completely implemented in the new process designer, both designers are available in Business Central for you to use.

> **NOTE**
>
> The legacy process designer in Business Central is deprecated in Red Hat Process Automation Manager 7.8.0. It will be removed in a future Red Hat Process Automation Manager release. The legacy process designer will not receive any new enhancements or features. If you intend to use the new process designer, start migrating your processes to the new designer. Create all new processes in the new process designer. For information about migrating to the new designer, see *Managing projects in Business Central*.

CHAPTER 2. BUSINESS PROCESS MODELING AND NOTATION VERSION 2.0

The Business Process Modeling and Notation Version 2.0 (BPMN2) specification is an Object Management Group (OMG) specification that defines standards for graphically representing a business process, defines execution semantics for the elements, and provides process definitions in XML format.

A process is defined or determined by its process definition. It exists in a knowledge base and is identified by its ID.

Table 2.1. General process properties

Label	Description
Name	Enter the name of the process.
Documentation	Describes the process. The text in this field is included in the process documentation, if applicable.
ID	Enter an identifier for this process, for example **orderItems**.
Package	Enter the package location for this process in your Red Hat Process Automation Manager project, for example **org.acme**.
ProcessType	Specify whether the process is public or private. (Currently not supported.)
Version	Enter the artifact version for the process.
Ad hoc	Select this option if this process is an ad hoc subprocess.
Process Instance Description	Enter a description of the purpose of the process.
Imports	Click to open the **Imports** window and add any data type classes required for your process.
Executable	Select this option to make the process executable part of your Red Hat Process Automation Manager project.
SLA Due Date	Enter the service level agreement (SLA) expiration date.
Process Variables	Add any process variables for the process. Process variables are visible within the specific process instance. Process variables are initialized at process creation and destroyed on process completion. Variable tags provide greater control over variable behavior, for example whether the variable is tagged as **required** or **readonly**. For more information about variable tags, see *Designing business processes in Business Central*
Metadata Attributes	Add any custom metadata attribute name and value that you want to use for custom event listeners, such as a listener to implement some action when a metadata attribute is present.

Label	Description
Global Variables	Add any global variables for the process. Global variables are visible to all process instances and assets in a project. Global variables are typically used by business rules and constraints and are created dynamically by the rules or constraints.

A process is a container for a set of modeling elements. It contains elements that specify the execution workflow of a business process or its parts using flow objects and flows. Each process has its own BPMN2 diagram. Red Hat Process Automation Manager contains the new process designer for creating BPMN2 diagrams and the legacy process designer to open the old BPMN2 diagram with **.bpmn2** extension. The new process designer has an improved layout and feature set and continues to develop. By default, the new diagrams are created in the new process designer.

2.1. RED HAT PROCESS AUTOMATION MANAGER SUPPORT FOR BPMN2

With Red Hat Process Automation Manager, you can model your business processes using the BPMN 2.0 standard. You can then use Red Hat Process Automation Manager to run, manage, and monitor these business processes. The full BPMN 2.0 specification also includes details on how to represent items such as choreographies and collaboration. However, Red Hat Process Automation Manager uses only the parts of the specification that you can use to specify executable processes. This includes almost all elements and attributes as defined in the Common Executable subclass of the BPMN2 specification, extended with some additional elements and attributes.

The following table contains a list of icons used to indicate whether a BPMN2 element is supported in the legacy process designer, the legacy and new process designer, or not supported.

Table 2.2. Support status icons

Key	Description
	Supported in the legacy and new process designer
	Supported in the legacy process designer only
	Not supported

Elements that have no icon do not exist in the BPMN2 specification.

Table 2.3. BPMN2 catching events

Element Name	Start	Intermediate
None	✅	
Message	✅	✅
Timer	✅	✅
Error	✅	✅
Escalation	✅	✅
Cancel		❌
Compensation	✅	✅
Conditional	✅	✅
Link		❌
Signal	✅	✅
Multiple	❌	❌

Element Name	Start	Intermediate
Parallel Multiple	✗	✗

Table 2.4. BPMN2 throwing and non-interrupting events

Element Name	Throwing		Non-interrupting	
	End	Intermediate	Start	Intermediate
None	✓			
Message	✓	✓	✓	✓
Timer			✓	✓
Error	✓			
Escalation	✓	✓	✓	✓
Cancel	✗	✗		✗
Compensation	✓	✓		
Conditional			✓	✓
Link		✗		

Element Name	Throwing		Non-interrupting	
Signal	✅	✅	✅	✅
Terminate	✅			
Multiple	❌	❌	❌	❌
Parallel Multiple			❌	❌

Table 2.5. BPMN2 elements

Element type	Element	Supported
Task	Business rule	✅
	Script	✅
	User task	✅
	Service task	✅
Subprocesses, including multiple instance subprocesses	Embedded	✅
	Ad hoc	✅

Element type	Element	Supported
	Reusable	✅
	Event	✅
Gateways	Inclusive	✅
	Exclusive	✅
	Parallel	✅
	Event-based	✅
	Complex	❌
Connecting objects	Sequence flows	✅
	Association flows	✅
Swimlanes	Swimlanes	✅
Artifacts	Group	⭐

Element type	Element	Supported
	Text annotation	✅

For more information about the background and applications of BPMN2, see the OMG Business Process Model and Notation (BPMN) Version 2.0 specification.

CHAPTER 3. BPMN2 EVENTS IN PROCESS DESIGNER

An event is something that happens to a business process. BPMN2 supports three categories of events:

- Start

- End

- Intermediate

A start event catches an event trigger, an end event throws an event trigger, and an intermediate event can both catch and throw event triggers.

The following business process diagram shows examples of events:
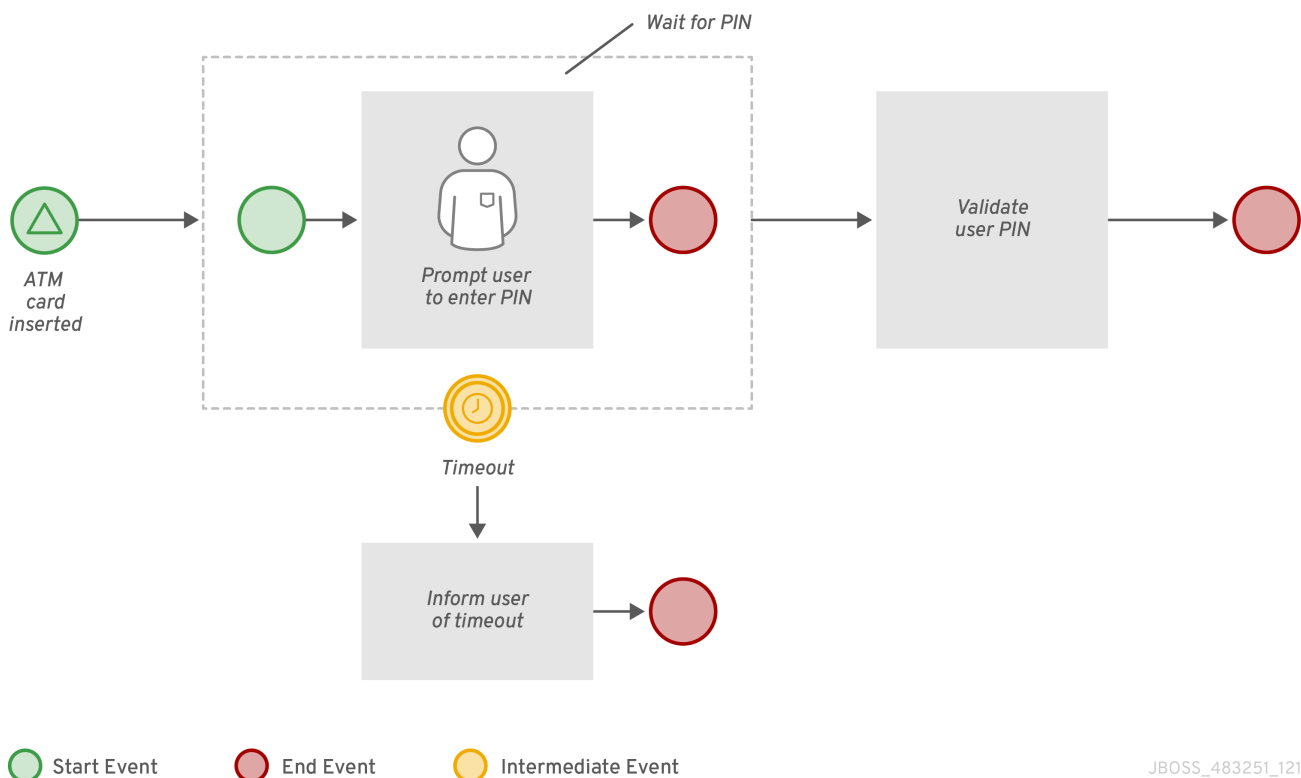
In this example, the following events occurred:

- The ATM Card Inserted signal start event is triggered when the signal is received.

- The timeout intermediate event is an interrupting event based on a timer trigger. This means that the Wait for PIN subprocess is canceled when the timer event is triggered.

- Depending on the inputs to the process, either end event associated with the Validate User Pin task or the end event associated with the Inform User of Timeout task ends the process.
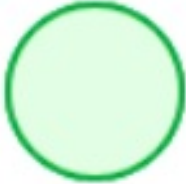
3.1. START EVENTS

Use start events to indicate the start of a business process. A start event cannot have an incoming sequence flow and must have only one outgoing sequence flow. You can use none start events in top-level processes, embedded subprocess, callable subprocesses, and event subprocesses.

All start events, with the exception of the none start event, are catch events. For example, a signal start event starts the process only when the referenced signal (event trigger) is received. You can configure

start events in event subprocesses to be interrupting or non-interrupting. An interrupting start event for an event subprocess stops or interrupts the execution of the containing or parent process. A non-interrupting start event does not stop or interrupt the execution of the containing or parent process.

Table 3.1. Start events

Start event type	Top-level	Subprocesses	
		Interrupt	Non-interrupt
None			
Conditional			
Compensation			
Error			
Escalation			
Message			

Start event type	Top-level	Subprocesses	
Signal			
Timer			

None

The none start event is a start event without a trigger condition. A process or a subprocess can contain at most one none start event, which is triggered on process or subprocess start by default, and the outgoing flow is taken immediately.

When you use a none start event in a subprocess, the execution of the process flow is transferred from the parent process into the subprocess and the none start event is triggered. This means that the token (the current location within the process flow) is passed from the parent process into the subprocess activity and the none start event of the subprocess generates a token of its own.

Conditional

The conditional start event is a start event with a Boolean condition definition. The execution is triggered when the condition is first evaluated to **false** and then to **true**. The process execution starts only if the condition is evaluated to **true** after the start event has been instantiated.

A process can contain multiple conditional start events.

Compensation

A compensation start event is used to start a compensation event subprocess when using a subprocess as the target activity of a compensation intermediate event.

Error

A process or subprocess can contain multiple error start events, which are triggered when an error object with a particular **ErrorRef** property is received. The error object can be produced by an error end event. It indicates an incorrect process ending. The process instance with the error start event starts execution after it has received the respective error object. The error start event is executed immediately upon receiving the error object and its outgoing flow is taken.

Escalation

The escalation start event is a start event that is triggered by an escalation with a particular escalation code. Processes can contain multiple escalation start events. The process instance with an escalation start event starts its execution when it receives the defined escalation object. The process is instantiated and the escalation start event is executed immediately and its outgoing flow is taken.

Message

A process or an event subprocess can contain multiple message start events, which are triggered by a particular message. The process instance with a message start event only starts its execution from this event after it has received the respective message. After the message is received, the process is instantiated and its message start event is executed immediately (its outgoing flow is taken).

Because a message can be consumed by an arbitrary number of processes and process elements, including no elements, one message can trigger multiple message start events and therefore instantiate multiple processes.

Signal

The signal start event is triggered by a signal with a particular signal code. A process can contain multiple signal start events. The signal start event only starts its execution within the process instance after the instance has received the respective signal. Then, the signal start event is executed and its outgoing flow is taken.

Timer

The timer start event is a start event with a timing mechanism. A process can contain multiple timer start events, which are triggered at the start of the process, after which the timing mechanism is applied.

When you use a timer start event in a subprocess, execution of the process flow is transferred from the parent process into the subprocess and the timer start event is triggered. The token is taken from the parent subprocess activity and the timer start event of the subprocess is triggered and waits for the timer to trigger. After the time defined by the timing definition has been reached, the outgoing flow is taken.

3.2. INTERMEDIATE EVENTS

Intermediate events drive the flow of a business process. Intermediate events are used to either catch or throw an event during the execution of the business process. These events are placed between the start and end events and can also be used on the boundary of an activity, like a subprocess or a human task, as a catch event. The boundary catch events can be configured as interrupting or non-interrupting. An interrupting boundary catch event cancels the bound activity whereas a non-interrupting event does not.

An intermediate event handles a particular situation that occurs during process execution. The situation is a trigger for an intermediate event. In a process, intermediate events with one outgoing flow can be placed on an activity boundary.

If the event occurs while the activity is being executed, the event triggers its execution to the outgoing flow. One activity may have multiple boundary intermediate events. Note that depending on the behavior you require from the activity with the boundary intermediate event, you can use either of the following intermediate event types:

- Interrupting: The activity execution is interrupted and the execution of the intermediate event is triggered.

- Non-interrupting: The intermediate event is triggered and the activity execution continues.

Table 3.2. Intermediate events

Intermediate event type	Catching	Boundary		Throwing
		Interrupt	Non-interrupt	

Intermediate event type	Catching	Boundary		Throwing
Message				
Timer				
Error				
Signal				
Conditional				
Compensation				
Escalation				

Message

A message intermediate event is an intermediate event that enables you to manage a message object. Use one of the following events:

- A throwing message intermediate event produces a message object based on the defined properties.

- A catching message intermediate event listens for a message object with the defined properties.

Timer

A timer intermediate event enables you to delay workflow execution or to trigger the workflow execution periodically. It represents a timer that can trigger one or multiple times after a specified period of time. When the timer intermediate event is triggered, the timer condition, which is the defined time, is checked and the outgoing flow is taken. When the timer intermediate event is placed in the process workflow, it has one incoming flow and one outgoing flow. Its execution starts when the incoming flow transfers to the event. When a timer intermediate event is placed on an activity boundary, the execution is triggered at the same time as the activity execution.

The timer is canceled if the timer element is canceled, for example by completing or aborting the enclosing process instance.

Conditional

A conditional intermediate event is an intermediate event with a boolean condition as its trigger. The event triggers further workflow execution when the condition evaluates to **true** and its outgoing flow is taken.

The event must define the **Expression** property. When a conditional intermediate event is placed in the process workflow, it has one incoming flow, one outgoing flow, and its execution starts when the incoming flow transfers to the event. When a conditional intermediate event is placed on an activity boundary, the execution is triggered at the same time as the activity execution. Note that if the event is non-interrupting, the event triggers continuously while the condition is **true**.

Signal

A signal intermediate event enables you to produce or consume a signal object. Use either of the following options:

- A throwing signal intermediate event produces a signal object based on the defined properties.

- A catching signal intermediate event listens for a signal object with the defined properties.

Error

An error intermediate event is an intermediate event that can be used only on an activity boundary. It enables the process to react to an error end event in the respective activity. The activity must not be atomic. When the activity finishes with an error end event that produces an error object with the respective **ErrorCode** property, the error intermediate event catches the error object and execution continues to its outgoing flow.

Compensation

A compensation intermediate event is a boundary event attached to an activity in a transaction subprocess. It can finish with a compensation end event or a cancel end event. The compensation intermediate event must be associated with a flow, which is connected to the compensation activity.

The activity associated with the boundary compensation intermediate event is executed if the transaction subprocess finishes with the compensation end event. The execution continues with the respective flow.

Escalation

An escalation intermediate event is an intermediate event that enables you to produce or consume an escalation object. Depending on the action the event element should perform, you need to use either of the following options:

- A throwing escalation intermediate event produces an escalation object based on the defined properties.

- A catching escalation intermediate event listens for an escalation object with the defined properties.

3.3. END EVENTS

End events are used to end a business process and may not have any outgoing sequence flows. There may be multiple end events in a business process. All end events, with the exception of the none and terminate end events, are throw events.

End events indicate the completion of a business process. An end event is a node that ends a particular workflow. It has one or more incoming sequence flows and no outgoing flow.

A process must contain at least one end event.

During run time, an end event finishes the process workflow. The end event can finish only the workflow that reached it, or all workflows in the process instance, depending on the end event type.

Table 3.3. End events

End event	Icon
None	
Message	
Signal	

End event	Icon
Error	
Compensation	
Escalation	
Terminate	

None

The none end event specifies that no other special behavior is associated with the end of the process.

Message

When a flow enters a message end event, the flow finishes and the end event produces a message as defined in its properties.

Signal

A throwing signal end event is used to finish a process or subprocess flow. When the execution flow enters the element, the execution flow finishes and produces a signal identified by its **SignalRef** property.

Error

The throwing error end event finishes the incoming workflow, which means consumes the incoming token, and produces an error object. Any other running workflows in the process or subprocess remain uninfluenced.

Compensation

A compensation end event is used to finish a transaction subprocess and trigger the compensation defined by the compensation intermediate event attached to the boundary of the subprocess activities.

Escalation

The escalation end event finishes the incoming workflow, which means consumes the incoming token, and produces an escalation signal as defined in its properties, triggering the escalation process.

Terminate

The terminate end event finishes all execution flows in the specified process instance. Activities being executed are canceled. The subprocess instance terminates if it reaches a terminate end event.

CHAPTER 4. CREATING A BUSINESS PROCESS IN BUSINESS CENTRAL

The process designer is the Red Hat Process Automation Manager process modeler. The output of the modeler is a BPMN 2.0 process definition file. The definition is used as input for the Red Hat Process Automation Manager process engine, which creates a process instance based on the definition.

The procedures in this section provide a general overview of how to create a simple business process. For a more detailed business process example, see *Getting started with business processes*.

Prerequisites

- You have created or imported a Red Hat Process Automation Manager project. For more information about creating projects, see *Managing projects in Business Central*.

- You have created the required users. User privileges and settings are controlled by the roles assigned to a user and the groups that a user belongs to. For more information about creating users, see *Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.3*.

Procedure

1. In Business Central, go to **Menu → Design → Projects**.

2. Click the project name to open the project's asset list.

3. Click **Add Asset → Business Process**

4. In the **Create new Business Process** wizard, enter the following values:

   - **Business Process**: New business process name

   - **Package**: Package location for your new business process, for example **com.myspace.myProject**

5. Click **Ok** to open the process designer.

6. In the upper-right corner, click the **Diagram properties** icon and add your business process property information, such as process data and variables:

   a. Scroll down and expand **Process Data**.

   b. Click ⊞ next to **Process Variables** and define the process variables that you want to use in your business process.

   Table 4.1. General process properties

Label	Description
Name	Enter the name of the process.

Label	Description
Documentation	Describes the process. The text in this field is included in the process documentation, if applicable.
ID	Enter an identifier for this process, such as **orderItems**.
Package	Enter the package location for this process in your Red Hat Process Automation Manager project, such as **org.acme**.
ProcessType	Specify whether the process is public or private (or null, if not applicable).
Version	Enter the artifact version for the process.
Ad hoc	Select this option if this process is an ad hoc subprocess.
Process Instance Description	Enter a description of the purpose of the process.
Imports	Click to open the **Imports** window and add any data object classes required for your process.
Executable	Select this option to make the process executable part of your Red Hat Process Automation Manager project.
SLA Due Date	Enter the service level agreement (SLA) expiration date.
Process Variables	Add any process variables for the process. Process variables are visible within the specific process instance. Process variables are initialized at process creation and destroyed on process completion. Variable **Tags** provide greater control over variable behavior, for example whether the variable is **required** or **readonly**. For more information about variable tags, see Chapter 5, *Variables*.
Metadata Attributes	Add any custom metadata attribute name and value that you want to use for custom event listeners, such as a listener to implement some action when a metadata attribute is present.
Global Variables	Add any global variables for the process. Global variables are visible to all process instances and assets in a project. Global variables are typically used by business rules and constraints, and are created dynamically by the rules or constraints.

The **Metadata Attributes** entries are similar to **Process Variables** tags in that they enable new **metaData** extensions to BPMN diagrams. However, process variable tags modify the behavior of specific process variables, such as whether a certain variable is **required** or **readonly**, whereas metadata attributes are key–value definitions that modify the behavior of the overall process.

For example, the following custom metadata attribute **riskLevel** and value **low** in a BPMN process correspond to a custom event listener for starting the process:

Figure 4.1. Example metadata attribute and value in the BPMN modeler

Metadata Attributes

Name	Value	+
riskLevel	low	🗑

Example metadata attribute and value in the BPMN file

```xml
<bpmn2:process id="approvals" name="approvals" isExecutable="true"
processType="Public">
  <bpmn2:extensionElements>
    <tns:metaData name="riskLevel">
      <tns:metaValue><![CDATA[low]]></tns:metaValue>
    </tns:metaData>
  </bpmn2:extensionElements>
```

Example event listener with metadata value

```java
public class MyListener implements ProcessEventListener {
    ...
    @Override
    public void beforeProcessStarted(ProcessStartedEvent event) {
        Map < String, Object > metadata =
event.getProcessInstance().getProcess().getMetaData();
        if (metadata.containsKey("low")) {
            // Implement some action for that metadata attribute
        }
    }
}
```

7. In the process designer canvas, use the left toolbar to drag and drop BPMN components to define your business process logic, connections, events, tasks, or other elements.

8. After you add and define all components of the business process, click **Save** to save the completed business process.
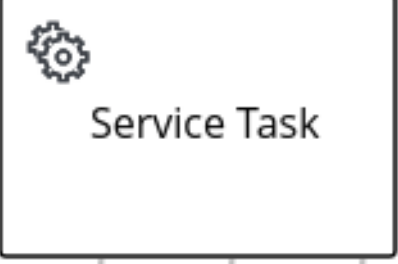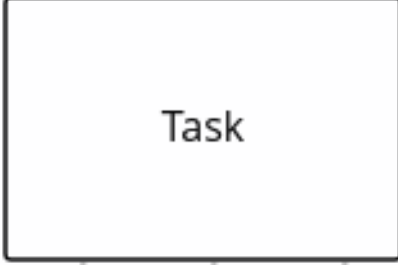
4.1. BPMN2 TASKS IN PROCESS DESIGNER

A task is an automatic activity that is defined in the process model and the smallest unit of work in a process flow. The following task types defined in the BPMN2 specification are available in the Red Hat Process Automation Manager process designer palette:

- Business rule task

- Script task

- User task

- Service task

- None task

Table 4.2. Task

Business rule task	 Task
Script task	 Task
User task	 Task
Service task	 Service Task
None task	 Task

In addition, the BPMN2 specification provides the ability to create custom tasks. The following predefined custom tasks are included with Red Hat Process Automation Manager:

- Rest service tasks: Used to invoke a remote RESTful service

- Email service tasks: Used to send an email

- Log service tasks: Used to log a message

- Java service tasks: Used to call Java code

- WebService service tasks: Used to invoke a remote WebService call

- DecisionTask tasks: Used to execute a DMN diagram

Business rule task

A business rule task defines a way to make a decision either through a DMN model or a rule flow group.

When a process reaches a business rule task defined by a DMN model, the process engine executes the DMN model decision with the inputs provided.

When a process reaches a business rule task defined by a rule flow group, the process engine begins executing the rules in the defined rule flow group. When there are no more active rules in the rule flow group, the execution continues to the next element. During the rule flow group execution, new activations belonging to the active rule flow group can be added to the agenda because these activations are changed by other rules.

Script task

A script task represents a script to be executed during the process execution.
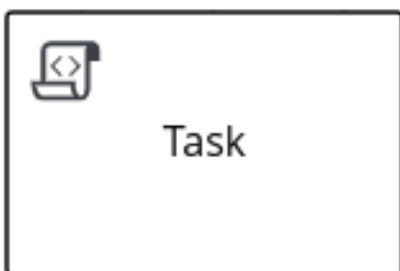
The associated script can access process variables and global variables. Review the following list before using a script task:

- Avoid low-level implementation details in the process. A script task can be used to manipulate variables, but consider using a service task when modelling more complex operations.

- Ensure that the script is executed immediately, otherwise use an asynchronous service task.

- Avoid contacting external services through a script task. Use a service task to model communication with an external service.

- Ensure scripts do not throw exceptions. Runtime exceptions should be caught and managed, for example, inside the script or transformed into signals or errors that can then be handled inside the process.

When a script task is reached during execution, the script is executed and the outgoing flow is taken.

User task

User tasks are tasks in the process workflow that cannot be performed automatically by the system and therefore require the intervention of a human user, the actor.
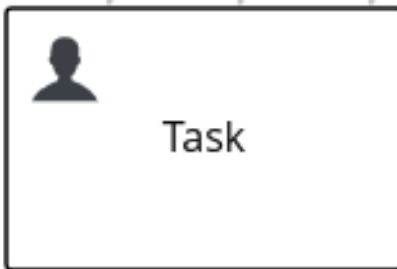
On execution, the User task element is instantiated as a task that appears in the list of tasks of one or more actors. If a User task element defines the **Groups** attribute, it is displayed in task lists of all users that are members of the group. Any user who is a member of the group can claim the task.

After it is claimed, the task disappears from the task list of the other users.

User tasks are implemented as domain-specific tasks and serve as a base for custom tasks.

Service task

Service tasks are tasks that do not require human interaction. They are completed automatically by an external software service.
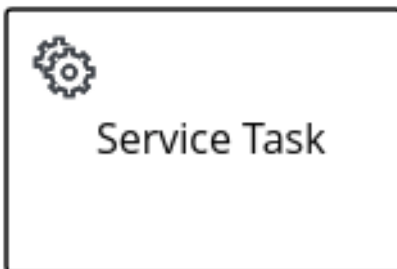
None task

None tasks are completed on activation. This is a conceptual model only. A none task is never actually executed by an IT system.

4.1.1. Creating business rules tasks

Business rules tasks are used to make decisions through a Decision Model and Notation (DMN) model or rule flow group.

Procedure

1. Create a business process.

2. In the process designer, select the **Activities** tool from the tool palette.

3. Select **Business Rule**.

4. Click a blank area of the process designer canvas.

5. If necessary, in the upper-right corner of the screen, click the **Properties** icon.

6. Add or define the task information listed in the following table as required.

   Table 4.3. Business rule task parameters

Label	Description
Name	The name of the business rule task.
Rule Language	The output language for the task. Select Decision Model and Notation (DMN) or Drools (DRL).
Rule Flow Group	The rule flow group associated with this business task. Select a rule flow group from the list or specify a new rule flow group.
On Entry Action	A Java, JavaScript, or MVEL script that specifies an action at the start of the task.
On Exit Action	A Java, JavaScript, or MVEL script that specifies an action at the end of the task.
Is Async	Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example a task performed by an outside service.
Adhoc Autostart	Select if this is an ad hoc task that should be started automatically. **Adhoc Autostart** enables the task to automatically start when the process or case instance is created instead of being starting by a start task. It is often used in case management.
SLA Due Date	The date that the service level agreement (SLA) expires.
Assignments	Click to add local variables.

7. Click **Save**.

4.1.2. Creating script tasks

Script tasks are used to execute a piece of code written in Java, JavaScript, or MVEL. They contain code snippets that specify the action of the script task. You can include global and process variables in your scripts.

You can write action scripts in Java, JavaScript, and MVEL. Note that MVEL accepts any valid Java code and additionally provides support for nested access of parameters. For example, the MVEL equivalent of the Java call **person.getName()** is **person.name**. MVEL also provides other improvements over Java and MVEL expressions are generally more convenient for business users.

Procedure

1. Create a business process.

2. In the process designer, select the **Activities** tool from the tool palette.

3. Select **Script**.

4. Click a blank area of the process designer canvas.

5. If necessary, in the upper-right corner of the screen, click the **Properties** icon.

6. Add or define the task information listed in the following table as required.

Table 4.4. Script task parameters

Label	Description
Name	The name of the business rule task.
Documentation	Enter a description of the task. The text in this field is included in the process documentation. Click the **Documentation** tab in the upper-left side of the process designer canvas to view the process documentation.
Script	Enter a script in Java, JavaScript, or MVEL to be excuted by the task, and select the script type.
Is Async	Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example a task performed by an outside service.
Adhoc Autostart	Select if this is an ad hoc task that should be started automatically. **Adhoc Autostart** enables the task to automatically start when the process or case instance is created instead of being starting by a start task. It is often used in case management.

7. Click **Save**.

4.1.3. Creating user tasks

User tasks are used to include human actions as input to the business process.

Procedure

1. Create a business process.

2. In the process designer, select the **Activities** tool from the tool palette.

3. Select **User**.

4. . Either drag and drop a business rule onto the process designer canvas or click a blank area of the canvas.

5. If necessary, in the upper-right corner of the screen, click the **Properties** icon.

6. Add or define the task information listed in the following table as required.

Table 4.5. User task parameters

Label	Description
Name	The display name of the business rule task.
Documentation	Enter a description of the task. The text in this field is included in the process documentation. Click the **Documentation** tab in the upper-left side of the process designer canvas to view the process documentation.
Task Name	The name of the human task.
Subject	Enter a subject for the task.
Actors	The actors responsible for executing the human task. Click **Add** to add a row then select an actor from the list or click **New** to add a new actor.
Groups	The groups responsible for executing the human task. Click **Add** to add a row then select a group from the list or click **New** to add a new group.
Assignments	Local variables for this task. Click to open the **Task Data I/O** window then add data inputs and outputs as required.
Reassignments	Specify a different actor to complete this task.
Notifications	Click to specify notifications associated with the task.
Is Async	Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example a task performed by an outside service.
Skippable	Select if this task is not mandatory.
Priority	Specify a priority for the task.

Label	Description
Description	Enter a description for the human task.
Created By	The user that created this task.
Adhoc Autostart	Select if this is an ad hoc task that should be started automatically. **Adhoc Autostart** enables the task to automatically start when the process or case instance is created instead of being starting by a start task. It is often used in case management.
Multiple Instance	Select if this task has multiple instances.
On Entry Action	A Java, JavaScript, or MVEL script that specifies an action at the start of the task.
On Exit Action	A Java, JavaScript, or MVEL script that specifies an action at the end of the task.
Content	The content of the script.
SLA Due Date	The date that the service level agreement (SLA) expires.

7. Click **Save**.

4.1.4. Creating service tasks

A service task is a task that is part of the process that is executed outside of the process, but is not a human task. Examples of service tasks include sending an email and logging a message when these tasks are performed by systems. You can define the parameters (input) and results (output) that are associated with a service task. A Service Task should have one incoming connection and one outgoing connection.

Procedure

1. Create a business process.

2. In the process designer, select the **Activities** tool from the tool palette.

3. Select **Service Task**.

4. Click a blank area of the process designer canvas.

5. If necessary, in the upper-right corner of the screen, click the **Properties** icon.

6. Add or define the task information listed in the following table as required.

   Table 4.6. Service task parameters

Label	Description
Name	The name of the service task.
Documentation	Enter a description of the task. The text in this field is included in the process documentation. Click the **Documentation** tab in the upper-left side of the process designer canvas to view the process documentation.
Implementation	Specify whether the task is implemented in Java or is a web service
Interface	The class used to implement the script, for example **org.xyz.HelloWorld**.
Operation	The method that will be called by the interface, for example **sayHello()**.
Assignments	Click to add local variables.
Adhoc Autostart	Select if this is an ad hoc task that should be started automatically. **Adhoc Autostart** enables the task to automatically start when the process or case instance is created instead of being starting by a start task. It is often used in case management.
Is Async	Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example a task performed by an outside service.
Is Multiple Instance	Select if this task has multiple instances.
On Entry Action	A Java, JavaScript, or MVEL script that specifies an action at the start of the task.
On Exit Action	A Java, JavaScript, or MVEL script that specifies an action at the end of the task.
SLA Due Date	The date that the service level agreement (SLA) expires.

7. Click **Save**.

4.2. MAKING A COPY OF A BUSINESS PROCESS

You can make a copy of a business process in Business Central and modify the copied process as needed.

Procedure

1. In the business process designer, click **Copy** in the upper-right toolbar.

2. In the **Make a Copy** window, enter a new name for the copied business process, select the target package, and optionally add a comment.

3. Click **Make a Copy**.

4. Modify the copied business process as needed and click **Save** to save the updated business process.

4.3. RESIZING ELEMENTS AND USING THE ZOOM FUNCTION TO VIEW BUSINESS PROCESSES

You can resize individual elements in a business process and zoom in or out to modify the view of your business process.

Procedure

1. In the business process designer, select the element and click the red dot in the lower-right corner of the element.

2. Drag the red dot to resize the element.

Figure 4.2. Resize an element
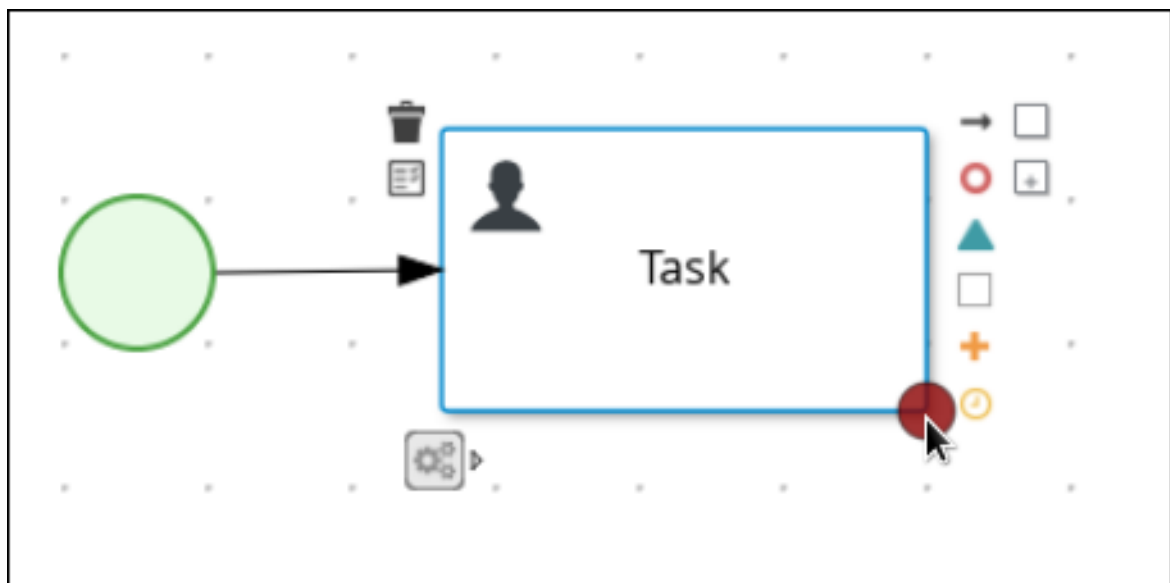
3. To zoom in or out to view the entire diagram, click the plus or minus sign on the lower-right side of the canvas.

Figure 4.3. Enlarge or shrink a business process
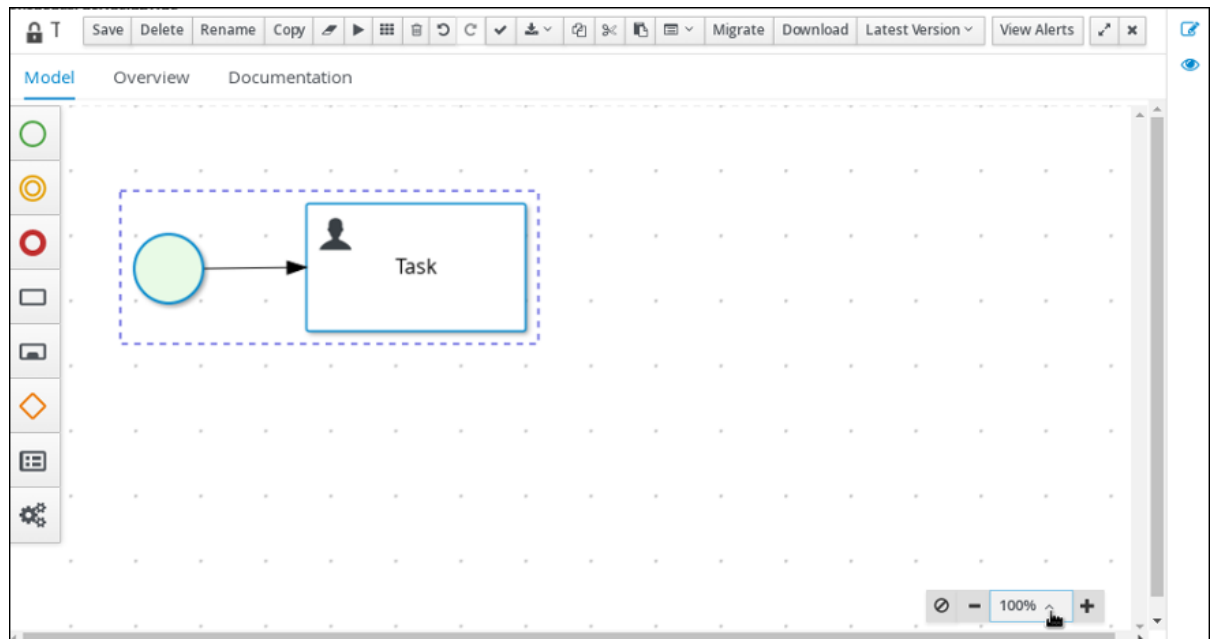
4.4. GENERATING PROCESS DOCUMENTATION IN BUSINESS CENTRAL

In the process designer in Business Central, you can view and print a report of the process definition. The process documentation summarizes the components, data, and visual flow of the process in a format (PDF) that you can print and share more easily.

Procedure

1. In Business Central, navigate to a project that contains a business process and select the process.

2. In the process designer, click the **Documentation** tab to view the summary of the process file, and click **Print** in the top-right corner of the window to print the PDF report.

Figure 4.4. Generate process documentation

4.5. BPMN2 SUBPROCESSES IN PROCESS DESIGNER

A subprocess is an activity that contains nodes. You can embed part of the main process within a subprocess. You can also include variable definitions within the subprocess. These variables are accessible to all nodes inside the subprocess.

A subprocess must have one incoming connection and one outgoing connection. A terminate end event inside a subprocess ends the subprocess instance but does not automatically end the parent process instance. A subprocess ends when there are no more active elements in it.

The following subprocess types are supported in Red Hat Process Automation Manager:

- Embedded subprocess, which is a part of the parent process execution and shares its data

- Ad hoc subprocess, which has no strict element execution order

- Reusable subprocess, which is independent from its parent process

- Event subprocess, which is only triggered on a start event or a timer

- Multi-instance subprocess

In the following example, the Place Order subprocess checks whether sufficient stock is available to place the order and updates the stock information if the order can be placed. The customer is then notified through the main process based on whether or not the order was placed.

Place order

| ○ Start Event | ● End Event | ◇ Gateway | JBOSS_483251_1218 |
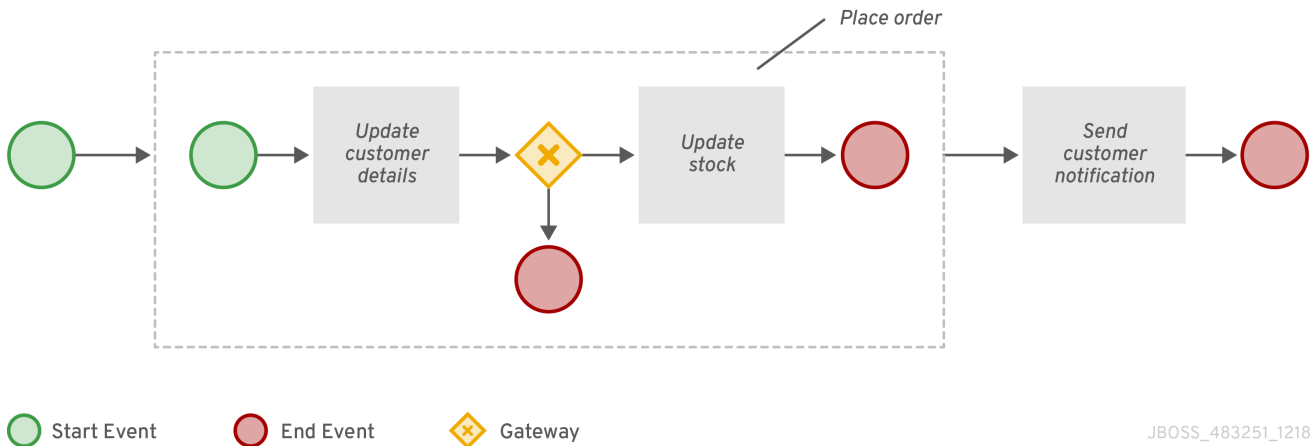
Embedded subprocess

An embedded subprocess encapsulates a part of the process. It must contain a start event and at least one end event. Note that the element enables you to define local subprocess variables that are accessible to all elements inside this container.

AdHoc subprocess

An ad hoc subprocess or process contains a number of embedded inner activities and is intended to be executed with a more flexible ordering compared to the typical routing of processes. Unlike regular processes, an ad hoc subprocess does not contain a complete, structured BPMN2 diagram description, for example, from start event to end event. Instead, the ad hoc subprocess contains only activities, sequence flows, gateways, and intermediate events. An ad hoc subprocess can also contain data objects and data associations. The activities within the ad hoc subprocesses are not required to have incoming and outgoing sequence flows. However, you can specify sequence flows between some of the contained activities. When used, sequence flows provide the same ordering constraints as in a regular process. To have any meaning, intermediate events must have outgoing sequence flows and they can be triggered multiple times while the ad hoc subprocess is active.
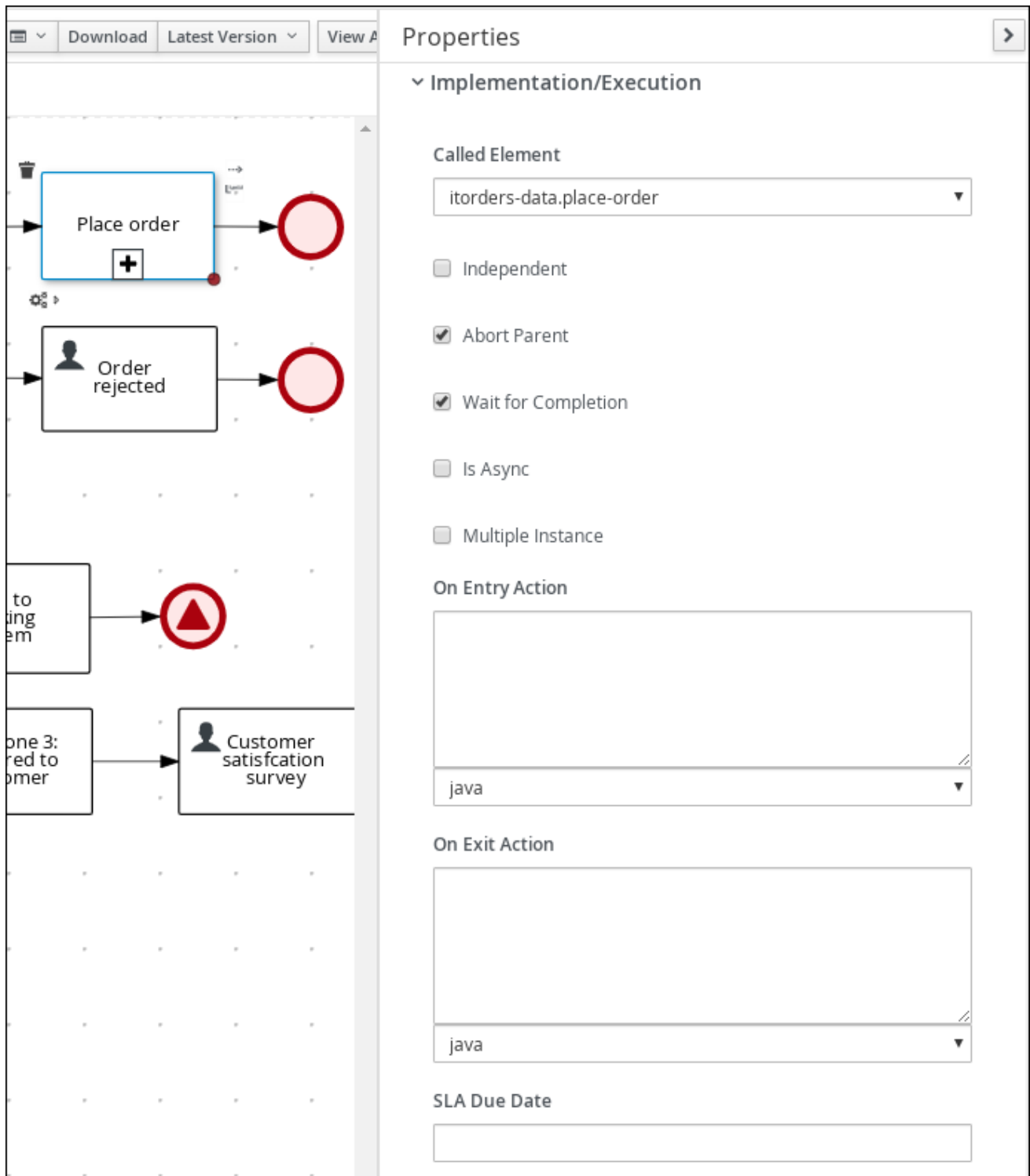
Reusable subprocess

Reusable subprocesses appear collapsed within the parent process. To configure a reusable subprocess, select the reusable subprocess, click $\boxed{✎}$, and expand **Implementation/Execution**. Set the following properties:

- **Called Element**: The ID of the subprocess that the activity calls and instantiates.

- **Independent**: If selected, the subprocess is started as an independent process. If not selected, the active subprocess is canceled when the parent process is terminated.

- **Abort Parent**: If selected, non-independent reusable subprocesses can abort the parent process when there is an error during the execution of the called process instance. For example, when there's an error when trying to invoke the subprocess or when the subprocess instance is aborted. This property is visible only when the **Independent** property is not selected. The following rules apply:

    - If the reusable subprocess is independent, **Abort parent** is not available.

- If the reusable subprocess is independent, **Abort parent** is not available.

- If the reusable subprocess is not independent, **Abort parent** is available.

- **Wait for completion**: If selected, the specified **On Exit Action** is not performed until the called subprocess instance is terminated. The parent process execution continues when the **On Exit Action** completes. This property is selected (set to **true**) by default.

- **Is Async**: Select if the task should be invoked asynchronously and cannot be executed instantly.

- **Multiple Instance**: Select to execute the subprocess elements a specified number of times. If selected, the following options are available:

  - **MI Execution mode**: Indicates if the multiple instances execute in parallel or sequentially. If set to **Sequential**, new instances are not created until the previous instance completes.

  - **MI Collection input**: Select a variable that represents a collection of elements for which new instances are created. The subprocess is instantiated as many times as the size of the collection.

  - **MI Data Input**: Specifies the name of the variable containing the selected element in the collection. The variable is used to access elements in the collection.

  - **MI Collection output** Optional variable that represents the collection of elements that will gather the output of the multi-instance node.

  - **MI Data Output**: Specifies the name of the variable that is added to the output collection that you selected in the **MI Collection output** property.

  - **MI Completion Condition (mvel)**: MVEL expression that is evaluated on each completed instance to check if the specified multiple instance node can complete. If it evaluates to **true**, all remaining instances are canceled.

- **On Entry Action**: A Java or MVEL script that specifies an action at the start of the task.

- **On Exit Action**: A Java or MVEL script that specifies an action at the end of the task.

- **SLA Due Date**: The date that the service level agreement (SLA) expires.

Figure 4.5. Reusable subprocess properties

Event subprocess

An event subprocess becomes active when its start event is triggered. It can interrupt the parent process context or run in parallel with it.

With no outgoing or incoming connections, only an event or a timer can trigger the subprocess. The subprocess is not part of the regular control flow. Although self-contained, it is executed in the context of the bounding process.

Use an event subprocess within a process flow to handle events that happen outside of the main process flow. For example, while booking a flight, two events may occur:

- Cancel booking (interrupting)

- Check booking status (non-interrupting)

You can model both of these events using the event subprocess.

Multiple instance subprocess

A multiple instances subprocess is instantiated multiple times when its execution is triggered. The instances are created sequentially. A new subprocess instance is created only after the previous instance has finished.

A multiple instances subprocess has one incoming connection and one outgoing connection.

4.6. BPMN2 GATEWAYS IN PROCESS DESIGNER

Gateways are used to create or synchronize branches in the workflow using a set of conditions called the gating mechanism. BPMN2 supports two types of gateways:

- Converging gateways, merging multiple flows into one flow

- Diverging gateways, splitting one flow into multiple flows

One gateway cannot have multiple incoming and multiple outgoing flows.

In the following business process diagram, the XOR gateway evaluates only the incoming flow whose condition evaluates to true:

JBOSS_483251_1218
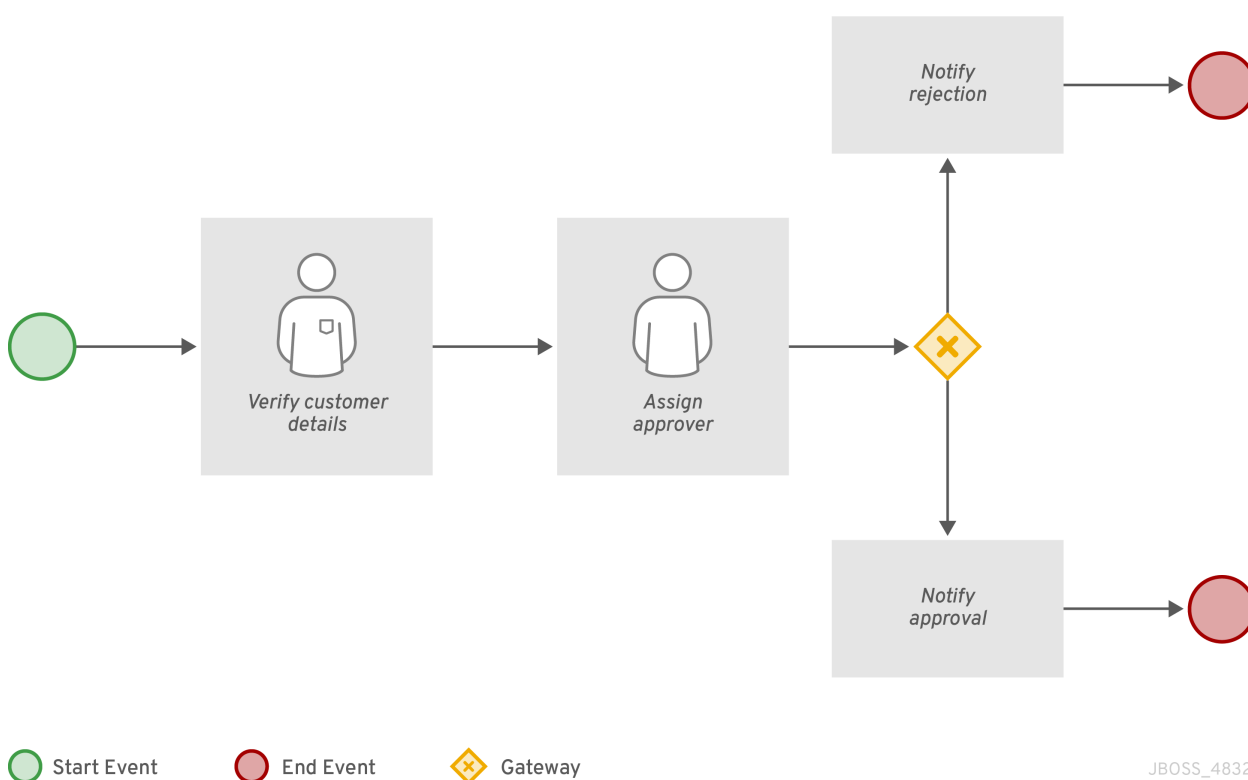
In this example, the customer details are verified by a user and the process is assigned to a user for approval. If approved, an approval notification is sent to the user. If the event of the request is rejected, a rejection notification is sent to the user.

Table 4.7. Gateway elements

Element type	Icon
exclusive (XOR)	
Inclusive	
Parallel	
Event	

Exclusive

In an exclusive diverging gateway, only the first incoming flow whose condition evaluates to true is chosen. In a converging gateway, the next node is triggered for each triggered incoming flow.

The gateway triggers exactly one outgoing flow. The flow with the constraint evaluated to true and the *lowest* priority number is taken.

> **IMPORTANT**
>
> Ensure that at least one of the outgoing flows evaluates to true at run time. Otherwise, the process instance terminates with a runtime exception.

The converging gateway enables a workflow branch to continue to its outgoing flow as soon as it reaches the gateway. When one of the incoming flows triggers the gateway, the workflow continues to the outgoing flow of the gateway. If it is triggered from more than one incoming flow, it triggers the next node for each trigger.

Inclusive

With an inclusive diverging gateway, the incoming flow is taken and all outgoing flows that evaluate to true are taken. Connections with lower priority numbers are triggered before triggering higher priority connections. Priorities are evaluated but the BPMN2 specification does not guarantee the priority order. Avoid depending on the **priority** attribute in your workflow.

IMPORTANT

Ensure that at least one of the outgoing flows evaluates to true at run time. Otherwise, the process instance terminates with a runtime exception.

A converging inclusive gateway merges all incoming flows previously created by an inclusive diverging gateway. It acts as a synchronizing entry point for the inclusive gateway branches.

Parallel

Use a parallel gateway to synchronize and create parallel flows. With a parallel diverging gateway, the incoming flow is taken, all outgoing flows are taken simultaneously. With a converging parallel gateway, the gateway waits until all incoming flows have entered and only then triggers the outgoing flow.

Event

An event-based gateway is only diverging and enables you to react to possible events as opposed to the data-based exclusive gateway, which reacts to the process data. The outgoing flow is taken based on the event that occurs. Only one outgoing flow is taken at a time. The gateway might act as a start event, where the process is instantiated only if one of the intermediate events connected to the event-based gateway occurs.

4.7. BPMN2 CONNECTING OBJECTS IN PROCESS DESIGNER

Connecting objects create an association between two BPMN2 elements. When a connecting object is directed, the association is sequential and indicates that one of the elements is executed immediately before the other, within an instance of the process. Connecting objects can start and end at the top, bottom, right, or left of the process elements being associated. The OMG BPMN2 specification allows you to use your discretion, placing connecting objects in a way that makes the process behavior easy to understand and follow.

BPMN2 supports two main types of connecting objects:

- Sequence flows: Connect elements of a process and define the order in which those elements are executed within an instance.

- Association flows: Connect the elements of a process without execution semantics. Association flows can be undirected or unidirectional.
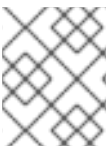
NOTE

The new process designer supports only undirected association flows. The legacy designer supports one direction and Unidirection flows.

4.8. BPMN2 USER TASK LIFE CYCLE IN PROCESS DESIGNER

You can trigger a user task element during the process instance execution to create a user task. The user task service of the task execution engine executes the user task instance. The process instance continues the execution only when the associated user task is completed or aborted. A user task life cycle is as follows:

- When a process instance enters a user task element, the user task is in the **Created** stage.

- **Created** stage is a transient stage and the user task enters the **Ready** stage immediately. The task appears in the task list of all the actors who are allowed to execute the task.

- When an actor claims the user task, the task becomes **Reserved**.
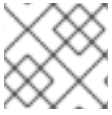
> **NOTE**
>
> If a user task has a single potential actor, the task is assigned to that actor upon creation.

- When an actor who claimed the user task starts the execution, the status of the user task changes to **InProgress**.

- Once an actor completes the user task, the status changes to **Completed** or **Failed** depending on the execution outcome.

There are also several other life cycle methods, including:

- Delegating or forwarding a user task so the user task is assigned to another actor.

- Revoking a user task, then the user task is no longer claimed by a single actor but is available to all actors who are allowed to take it.

- Suspending and resuming a user task.

- Stopping a user task that is in progress.

- Skipping a user task, in which the execution of the task is suspended.

For more information about the user task life cycle, refer Web Services Human Task.

4.9. BPMN2 TASK PERMISSION MATRIX IN PROCESS DESIGNER

The user task permission matrix summarizes the actions that are allowed for specific user roles. The user roles are as follows:

- Potential owner: User who can claim the task, which was claimed earlier and is released and forwarded. The tasks with **Ready** status can be claimed, and the potential owner becomes the actual owner of the task.

- Actual owner: User who claims the task and progresses the task to completion or failure.

- Business administrator: Super user who can modify the status or progress with the task at any point of the task life cycle.

The following permission matrix represents the authorizations for all operations that modify a task.

- **+** indicates that the user role is allowed to do the specified operation.

- **-** indicates that the user role is not allowed to do the specified operation, or the operation does not match with the user's role.

Table 4.8. Main operations permissions matrix

Operation	Potential owner	Actual owner	Business administrator
activate	-	-	+

Operation	Potential owner	Actual owner	Business administrator
claim	+	-	+
complete	-	+	+
delegate	+	+	+
fail	-	+	+
forward	+	+	+
nominate	-	-	+
release	-	+	+
remove	-	-	+
resume	+	+	+
skip	+	+	+
start	+	+	+
stop	-	+	+
suspend	+	+	+

4.10. BPMN2 SWIMLANES IN PROCESS DESIGNER

Swimlanes are process elements that visually group tasks related to one group or user. You can use user tasks in combination with swimlanes to assign multiple human tasks to the same actor. At run time, swimlanes auto-claim or assign tasks to users who have completed another task in that lane, within the same process instance. When the first task in a swimlane is created, and that task has an actor ID specified, that actor ID is assigned to all other tasks of that swimlane as well. A lane is a sub-partition within a process that enables you to group some process elements and define their common parameters.

In the following example, the Analyst lane has two user tasks:
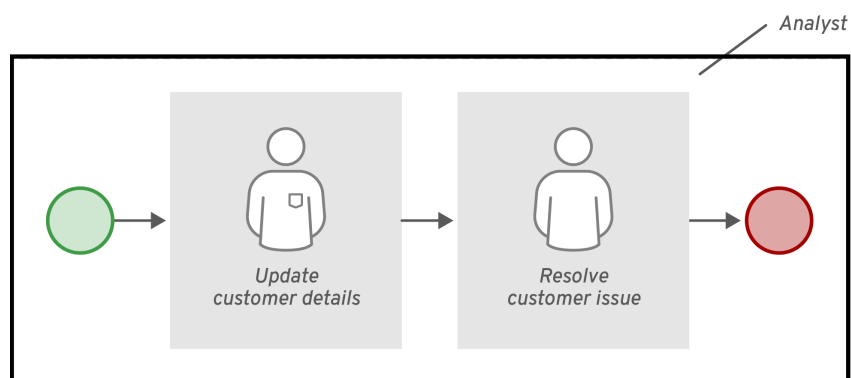
The **Group** field in the Update Customer Details and Resolve Customer Issue tasks has the value **analyst**. When the process is started, and the Update Customer Details task is claimed, started, or completed by an analyst user, the Resolve Customer Issue task is claimed and assigned to the user who completed the first task. However, if only the Update Customer Details task has the analyst group assigned, and the second task had no user or group assignments, the process stops after the first task completes.

CHAPTER 5. VARIABLES

Variables store data that is used during runtime. Process designer uses three types of variables:

Global variables

Global variables are visible to all process instances and assets in a particular session. They are intended to be used primarily by business rules and by constraints and are created dynamically by rules or constraints.

Process variables

Process variables are defined as properties in the BPMN2 definition file and are visible within the process instance. They are initialized at process creation and destroyed on process completion.

Local variables

Local variables are associated with and available within specific process elements, such as activities. They are initialized when the element context is initialized, that is, when the execution workflow enters the node and execution of the **onEntry** action has finished, if applicable. They are destroyed when the element context is destroyed, that is, when the execution workflow leaves the element.

An element, such as a process, sub-process, or task can only access variables in its own and parent contexts. An element cannot access a variable defined in the element's child element. Therefore, when an elements requires access to a variable during runtime, its own context is searched first.

If the variable cannot be found directly in the element's context, the immediate parent context is searched. The search continues until the process context is reached. In case of global variables, the search is performed directly on the session container.

If the variable cannot be found, a read access request returns **null** and a write access produces an error message, and the process continues its execution. Variables are searched for based on their ID.

5.1. VARIABLE TAGS

For greater control over variable behavior, you can tag process variables and local variables in the BPMN process file. Tags are simple string values that you add as metadata to a specific variable.

Red Hat Process Automation Manager supports the following tags for process variables and local variables:

- **required**: Sets the variable as a requirement in order to start a process instance. If a process instance starts without the required variable, Red Hat Process Automation Manager generates a **VariableViolationException** error.

- **readonly**: Indicates that the variable is for informational purposes only and can be set only once during process instance execution. If the value of a read-only variable is modified at any time, Red Hat Process Automation Manager generates a **VariableViolationException** error.

- **restricted**: A special tag that is used with the **VariableGuardProcessEventListener** to indicate that permission is granted to modify the variable based on the required and the existing role. **VariableGuardProcessEventListener** is extended from **DefaultProcessEventListener** and supports two different constructors:

  - **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;
}
```

Therefore, you must add an event listener to the session with the allowed role name and identity provider that returns the user role as shown in the following example:

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

In the above example, the **VariableGuardProcessEventListener** method verifies if a variable is tagged with a security constraint tag (**restricted**). If the user does not have the required role, then Red Hat Process Automation Manager generates a **VariableViolationException** error.

> **NOTE**
>
> The variable tags that appear in the Business Central UI, for example **internal**, **input**, **output**, **business-relevant**, and **tracked** are not supported in Red Hat Process Automation Manager.

You can add the tag directly to the BPMN process source file as a **customTags** metadata property with the tag value defined in the format **![CDATA[*TAG_NAME*]]**.

For example, the following BPMN process applies the **required** tag to an **approver** process variable:

Figure 5.1. Example variable tagged in the BPMN modeler
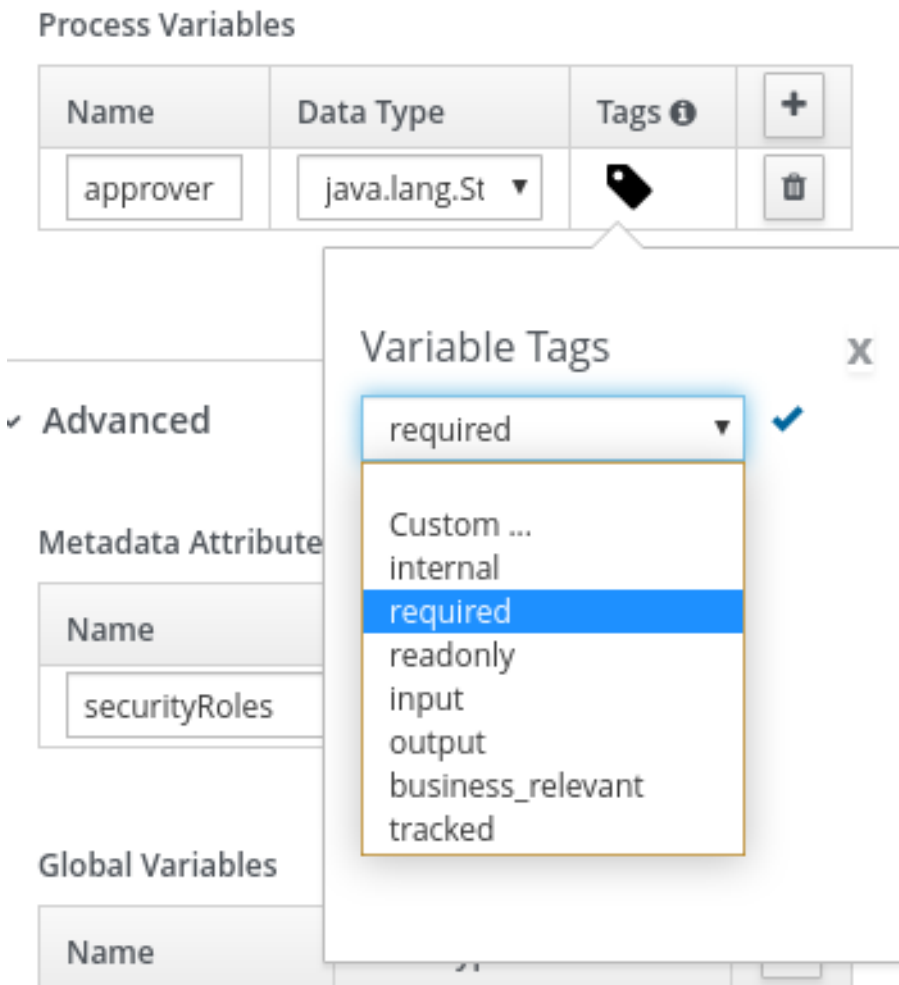
Example variable tagged in a BPMN file

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

You can use more than one tag for a variable where applicable. You can also define custom variable tags in your BPMN files to make variable data available to Red Hat Process Automation Manager process event listeners. Custom tags do not influence the Red Hat Process Automation Manager runtime as the standard variable tags do and are for informational purposes only. You define custom variable tags in the same **customTags** metadata property format that you use for standard Red Hat Process Automation Manager variable tags.

5.2. DEFINING GLOBAL VARIABLES

Global variables exist in a knowledge session and can be accessed and are shared by all assets in that session. They belong to the particular session of the Knowledge Base and they are used to pass information to the engine. Every global variable defines its ID and item subject reference. The ID serves as the variable name and must be unique within the process definition. The item subject reference defines the data type the variable stores.

IMPORTANT

The rules are evaluated at the moment the fact is inserted. Therefore, if you are using a global variable to constrain a fact pattern and the global is not set, the system returns a **NullPointerException**.

Global variables are initialized either when the process with the variable definition is added to the session or when the session is initialized with globals as its parameters.

Values of global variables can typically be changed during the assignment, which is a mapping between a process variable and an activity variable. The global variable is then associated with the local activity context, local activity variable, or by a direct call to the variable from a child context.

Prerequisites

- You have created a project in Business Central and it contains at least one business process asset.

Procedure

1. Open a business process asset.

2. Click a blank area of the process designer canvas.

3. Click the **Properties** icon on the upper-right side of the screen to open the **Properties** panel.

4. If necessary, expand the **Process** section.

5. In the **Global Variables** sub-section, click the plus icon.

6. Enter a name for the variable in the **Name** box.

7. Select a data type from the **Data Type** menu.

5.3. DEFINING PROCESS VARIABLES

Process variables are defined as properties in the BPMN2 definition file and are visible within the process instance. They are initialized at process creation and destroyed on process completion.

A process variable is a variable that exists in a process context and can be accessed by its process or its child elements. Process variables belong to a particular process instance and cannot be accessed by other process instances. Every process variable defines its ID and item subject reference: the ID serves as the variable name and must be unique within the process definition. The item subject reference defines the data type the variable stores.

Process variables are initialized when the process instance is created. Their value can be changed by the process activities using the Assignment, when the global variable is associated with the local Activity context, local Activity variable, or by a direct call to the variable from a child context.

Note that process variables should be mapped to local variables.

Prerequisites

- You have created a project in Business Central and it contains at least one business process asset.

Procedure

1. Open a business process asset.

2. Click a blank area of the process designer canvas.

3. Click the **Properties** icon on the upper-right side of the screen to open the **Properties** panel.

4. If necessary, expand the **Process Data** section.

5. In the **Process Variables** sub-section, click the plus icon.

6. Enter a name for the variable in the **Name** box.

7. Select a data type from the **Data Type** menu.

5.4. DEFINING LOCAL VARIABLES

Local variables are available within their process element, such as an activity. They are initialized when the element context is initialized, that is, when the execution workflow enters the node and execution of the **onEntry** action has finished, if applicable. They are destroyed when the element context is destroyed, that is, when the execution workflow leaves the element.

Values of local variables can be mapped to global or process variables. This enables you to maintain relative independence of the parent element that accommodates the local variable. Such isolation might help prevent technical exceptions.

A local variable is a variable that exists in a child element context of a process and can be accessed only from within this context. Local variables belong to the particular element of a process.

For tasks, with the exception of the Script task, you can define **Data Input Assignments** and **Data Output Assignments** in the **Assignments** property. Data Input Assignment defines variables that enter the Task and therefore provide the entry data needed for the task execution. The Data Output Assignments can refer to the context of the Task after execution to acquire output data.

User Tasks present data related to the actor that is executing the User Task. Additionally, User Tasks also request the actor to provide result data related to the execution.

To request and provide the data, use task forms and map the data in the Data Input Assignment parameter to a variable. Map the data provided by the user in the Data Output Assignment parameter if you want to preserve the data as output.

Prerequisites

- You have created a project in Business Central and it contains at least one business process asset that has at least one task that is not a script task.

Procedure

1. Open a business process asset.

2. Select a task that is not a script task.

3. Click the **Properties** icon on the upper-right side of the screen to open the **Properties** panel.

4. Click the box under the **Assignments** sub-section. The **Task Data I/O** dialog box opens.

5. Click **Add** next to **Data Inputs and Assignments** or **Data Outputs and Assignments**.

6. Enter a name for the local variable in the **Name** box.

7. Select a data type from the **Data Type** menu.

8. Select a source or target then click **Save**.

CHAPTER 6. DEPLOYING A BUSINESS PROCESS IN BUSINESS CENTRAL

After you design your business process in Business Central, you can build and deploy your project in Business Central to make the process available to KIE Server.

Prerequisites

- KIE Server is deployed and connected to Business Central. For more information about KIE Server configuration, see *Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.3*.

Procedure

1. In Business Central, go to **Menu → Design → Projects**.

2. Click the project that you want to deploy.

3. Click **Deploy**.

   > **NOTE**
   >
   > You can also select the **Build & Install** option to build the project and publish the KJAR file to the configured Maven repository without deploying to a KIE Server. In a development environment, you can click **Deploy** to deploy the built KJAR file to a KIE Server without stopping any running instances (if applicable), or click **Redeploy** to deploy the built KJAR file and replace all instances. The next time you deploy or redeploy the built KJAR, the previous deployment unit (KIE container) is automatically updated in the same target KIE Server. In a production environment, the **Redeploy** option is disabled and you can click **Deploy** only to deploy the built KJAR file to a new deployment unit (KIE container) on a KIE Server.
   >
   > To configure the KIE Server environment mode, set the **org.kie.server.mode** system property to **org.kie.server.mode=development** or **org.kie.server.mode=production**. To configure the deployment behavior for a corresponding project in Business Central, go to project **Settings → General Settings → Version** and toggle the **Development Mode** option. By default, KIE Server and all new projects in Business Central are in development mode. You cannot deploy a project with **Development Mode** turned on or with a manually added **SNAPSHOT** version suffix to a KIE Server that is in production mode.

   To review project deployment details, click **View deployment details** in the deployment banner at the top of the screen or in the **Deploy** drop-down menu. This option directs you to the **Menu → Deploy → Execution Servers** page.

CHAPTER 7. EXECUTING A BUSINESS PROCESS IN BUSINESS CENTRAL

After you build and deploy the project that contains your business process, you can execute the defined functionality for the business process.

As an example, this procedure uses the **Mortgage_Process** sample project in Business Central. In this scenario, you input data into a mortgage application form acting as the mortgage broker. The **MortgageApprovalProcess** business process runs and determines whether or not the applicant has offered an acceptable down payment based on the decision rules defined in the project. The business process either ends the rule testing or requests that the applicant increase the down payment to proceed. If the application passes the business rule testing, the bank approver reviews the application and either approves or denies the loan.

Prerequisites

- KIE Server is deployed and connected to Business Central. For more information about KIE Server configuration, see *Installing and configuring Red Hat Process Automation Manager on Red Hat JBoss EAP 7.3*.

Procedure

1. In Business Central, go to **Menu → Projects** and select a space. The default space is MySpace.

2. In the upper-right corner of the window, click the arrow next to **Add Project** and select **Try Samples**.

3. Select the **Mortgage_Process** sample and click **Ok**.

4. On the project page, select **Mortgage_Process**.

5. On the **Mortgage_Process** page, click **Build**.

6. After the project has built, click **Deploy**.

7. Go to **Menu → Manage → Process Definitions**.

8. Click anywhere in the **MortgageApprovalProcess** row to view the process details.

9. Click the **Diagram** tab to view the business process diagram in the editor.

10. Click **New Process Instance** to open the **Application** form and input the following values into the form fields:

    - 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. Click **Submit** to start a new process instance. After starting the process instance, the **Instance Details** view opens.

12. Click the **Diagram** tab to view the process flow within the process diagram. The state of the process is highlighted as it moves through each task.

13. Click **Menu → Manage → Tasks**.
    For this example, the user or users working on the corresponding tasks are members of the following groups:

    - **approver**: For the **Qualify** task

    - **broker**: For the **Correct Data** and **Increase Down Payment** tasks

    - **manager**: For the **Final Approval** task

14. As the approver, review the **Qualify** task information, click **Claim** and then **Start** to start the task, and then select **Is mortgage application in limit?** and click **Complete** to complete the task flow.

15. In the **Tasks** page, click anywhere in the **Final Approval** row to open the **Final Approval** task.

16. Click **Claim** to claim responsibility for the task, and click **Complete** to finalize the loan approval process.
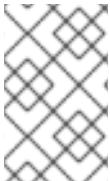
> **NOTE**
>
> The **Save** and **Release** buttons are only used to either pause the approval process and save the instance if you are waiting on a field value, or to release the task for another user to modify.

CHAPTER 8. MANAGING LOG FILES

Red Hat Process Automation Manager manages the required maintenance, runtime data that is removed, including:

- **Process instance data**, which is removed upon process instance completion.

- **Work item data**, which is removed upon work item completion.

- **Task instance data**, which is removed upon completion of a process to which the given task belongs.

Runtime data, which is not cleaned automatically includes session information data that is based on the selected runtime strategy.

- **Singleton strategy** ensures that runtime data of session information is not automatically removed.

- **Per request strategy** allows automatic removal when a request is terminated.

- **Per process instances** are automatically removed when a process instance is mapped to a session that is completed or aborted.

In order to keep the track of process instances, Red Hat Process Automation Manager provides audit data tables. There are two ways to manage and maintain the audit data tables, including cleaning up the jobs automatically and manually.

8.1. SETTING UP AUTOMATIC CLEANUP JOB

You can set up an automatic cleanup job in Business Central.

Procedure

1. In Business Central, go to **Deploy** > **Jobs**.

2. Click **New Job**.

3. Enter a name, due date, time, and the following command into the **Type** text field.

   > org.jbpm.executor.commands.LogCleanupCommand

4. Click **Add Parameter** if you want to use the parameters, enter a parameter in the key section and enter a parameter value in the value section.

5. Click **Create** to finalize the job creation wizard.

The automatic cleanup job is created successfully.

8.2. MANUAL CLEANUP

To perform manual cleanup, you can use the audit API. The audit API is divided into the following areas:

Table 8.1. Audit API areas

Name	Description
Process audit	It is used to clean up process, node and variable logs that are accessible in the **jbpm-audit** module. For example, you can access the module as follows: **org.jbpm.process.audit.JPAAuditLogService**
Task audit	It is used to clean up tasks and events that are accessible in the **jbpm-human-task-audit** module. For example, you can access the module as follows: **org.jbpm.services.task.audit.service.TaskJPAAuditService**
Executor jobs	It is used to clean up executor jobs and errors that are accessible in the **jbpm-executor** module. For example, you can access the module as follows: **org.jbpm.executor.impl.jpa.ExecutorJPAAuditService**

8.3. REMOVING LOGS FROM THE DATABASE

Use **LogCleanupCommand** executor command to clean up the data, which is using the database space. The **LogCleanupCommand** consists of logic to automatically clean up all or selected data.

There are several configuration options that you can use with the **LogCleanupCommand**:

Table 8.2. LogCleanupCommand parameters table

Name	Description	Is Exclusive
SkipProcessLog	Indicates whether process and node instances, and process variables log cleanup is skipped when the command runs. The default value is **false**.	No, it is used with other parameters.
SkipTaskLog	Indicates if the task audit and event log cleanup are skipped. The default value is **false**.	No, it is used with other parameters.
SkipExecutorLog	Indicates if Red Hat Process Automation Manager executor entries cleanup is skipped. The default value is **false**.	No, it is used with other parameters.
SingleRun	Indicates if a job routine runs only once. The default value is **false**.	No, it is used with other parameters.

Name	Description	Is Exclusive
NextRun	Schedules the next job execution. The default value is **24h**. For example, set to **12h** for jobs to be executed every 12 hours. The schedule is ignored if you set SingleRun to **true**, unless you set both SingleRun and **NextRun**. If both are set, the NextRun schedule takes priority. The ISO format can be used to set the precise date.	No, it is used with other parameters.
OlderThan	Logs that are older than the specified date are removed. The date format is **YYYY-MM-DD**. Usually, this parameter is used for single run jobs.	Yes, it is not used with **OlderThanPeriod** parameter.
OlderThanPeriod	Logs that are older than the specified timer expression are removed. For example, set **30d** to remove logs, which are older than 30 days.	Yes, it is not used with **OlderThan** parameter.
ForProcess	Specifies process definition ID for logs that are removed.	No, it is used with other parameters.
ForDeployment	Specifies deployment ID of the logs that are removed.	No, it is used with other parameters.
EmfName	Persistence unit name that is used to perform delete operation.	Not applicable

NOTE

LogCleanupCommand does not remove any active instances, such as running process instances, task instances, or executor jobs.

CHAPTER 9. PROCESS DEFINITIONS AND PROCESS INSTANCES IN BUSINESS CENTRAL

A process definition is a Business Process Model and Notation (BPMN) 2.0 file that serves as a container for a process and its BPMN diagram. The process definition shows all of the available information about the business process, such as any associated subprocesses or the number of users and groups that are participating in the selected definition.

A process definition also defines the **import** entry for imported processes that the process definition uses, and the **relationship** entries.

BPMN2 source of a process definition

```
<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>
```

After you have created, configured, and deployed your project that includes your business processes, you can view the list of all the process definitions in Business Central **Menu → Manage → Process Definitions**. You can refresh the list of deployed process definitions at any time by clicking the refresh button in the upper-right corner.

The process definition list shows all the available process definitions that are deployed into the platform. Click any of the process definitions listed to show the corresponding process definition details. This displays information about the process definition, such as if there is a sub-process associated with it, or how many users and groups exist in the process definition. The **Diagram** tab in the process definition details page contains the BPMN2-based diagram of the process definition.

Within each selected process definition, you can start a new process instance for the process definition by clicking the **New Process Instance** button in the upper-right corner. Process instances that you start from the available process definitions are listed in **Menu → Manage → Process Instances**.

You can also define the default pagination option for all users under the **Manage** drop-down menu (**Process Definition**, **Process Instances**, **Tasks**, **Jobs**, and **Execution Errors**) and in **Menu → Track → Task Inbox**.

For more information about process and task administration in Business Central, see *Managing and monitoring business processes in Business Central*.

9.1. STARTING A PROCESS INSTANCE FROM THE PROCESS DEFINITIONS PAGE

You can start a process instance in **Menu → Manage → Process Definitions**. This is useful for environments where you are working with several projects or process definitions at the same time.

Prerequisites

- A project with a process definition has been deployed in Business Central.

Procedure

1. In Business Central, go to **Menu → Manage → Process Definitions**.

2. Select the process definition for which you want to start a new process instance from the list. The details page of the definition opens.

3. Click **New Process Instance** in the upper-right corner to start a new process instance.

4. Provide any required information for the process instance.

5. Click **Submit** to create the process instance.

6. View the new process instance in **Menu → Manage → Process Instances**.

9.2. STARTING A PROCESS INSTANCE FROM THE PROCESS INSTANCES PAGE

You can create new process instances or view the list of all the running process instances in **Menu → Manage → Process Instances**.

Prerequisites

- A project with a process definition has been deployed in Business Central.

Procedure

1. In Business Central, go to **Menu → Manage → Process Instances**.

2. Click **New Process Instance** in the upper-right corner and select the process definition for which you want to start a new process instance from the drop-down list.

3. Provide any information required to start a new process instance.

4. Click **Start** to create the process instance.
   The new process instance appears in the **Manage Process Instances** list.

9.3. PROCESS DEFINITIONS IN XML

You can create processes directly in XML format using the BPMN 2.0 specifications. The syntax of these XML processes is defined using the BPMN 2.0 XML Schema Definition.

A process XML file consists of the following core sections:

- **process**: This is the top part of the process XML that contains the definition of the different nodes and their properties. The process XML file consists of exactly one **<process>** element. This element contains parameters related to the process (its type, name, ID, and package name), and consists of three subsections: a header section where process-level information such as variables, globals, imports, and lanes are defined, a nodes section that defines each of the nodes in the process, and a connections section that contains the connections between all the nodes in the process.

- **BPMNDiagram**: This is the lower part of the process XML file that contains all graphical information, such as the location of the nodes. The nodes section contains a specific element for each node and defines the various parameters and any sub-elements for that node type.

The following process XML file fragment shows a simple process that contains a sequence of a start event, a script task that prints **"Hello World"** to the console, and an end event:

```xml
<?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>
```

CHAPTER 10. FORMS IN BUSINESS CENTRAL

A form is a layout definition for a page, defined as HTML, that is displayed as a dialog window to the user during process and task instantiation. Task forms acquire data from a user for both the process and task instance execution, whereas process forms take input and output from process variables.

The input is then mapped to the task using the data input assignment, which you can use inside of a task. When the task is completed, the data is mapped as a data output assignment to provide the data to the parent process instance.

10.1. FORM MODELER

Red Hat Process Automation Manager provides a custom editor for defining forms called Form Modeler. With Form Modeler, you can generate forms for data objects, task forms, and process start forms without writing code. Form Modeler includes a widget library for binding multiple data types and a callback mechanism to send notifications when form values change. Form Modeler uses bean-based validation and supports binding form fields to static or dynamic models.

Form Modeler includes the following features:

- Form modeling user interface for forms

- Form auto-generation from the data model or Java objects

- Data binding for Java objects

- Formula and expressions

- Customized forms layouts

- Forms embedding

Form Modeler comes with predefined field types that you place onto the canvas to create a form.

Figure 10.1. Example mortgage loan application form
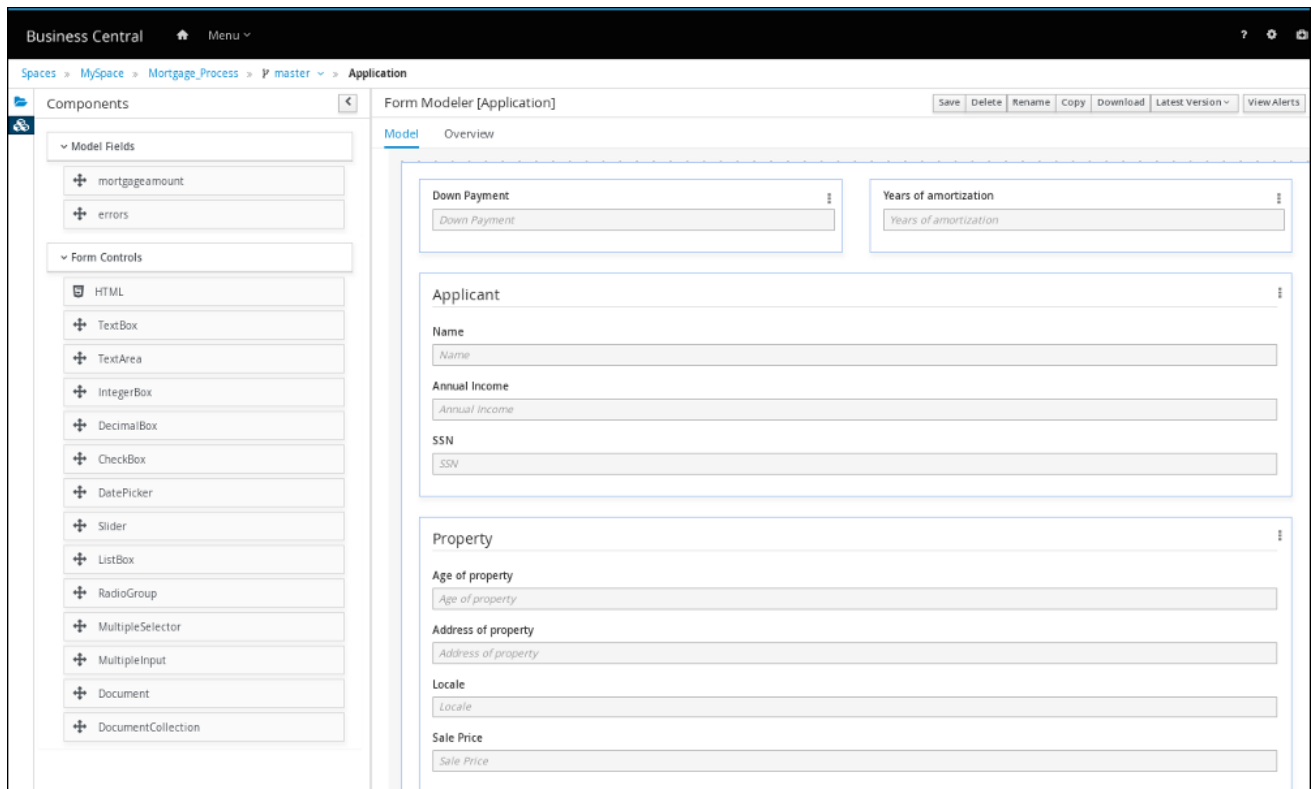
10.2. GENERATING PROCESS AND TASK FORMS IN BUSINESS CENTRAL

You can generate a process form from your business process that is displayed at process instantiation to the user who instantiated the process. You can also generate a task form from your business process that is displayed at user task instantiation, when the execution flow reaches the task, to the actor of the user task.

Procedure

1. In Business Central, go to **Menu → Design → Projects**.

2. Click the project name to open the asset view and then click the business process name.

3. In the process designer, click the process task that you want to create a form for (if applicable).

4. In the upper-right toolbar, click the **Form Generation** icon and select the forms that you want to generate:

   - **Generate process form**: Generates the form for the entire process. This is the initial form that a user must complete when the process instance is started.

   - **Generate all forms**: Generates the form for the entire process and for all user tasks.

   - **Generate forms for selection**: Generates the forms for the selected user task nodes.

Figure 10.2. Form generation menu
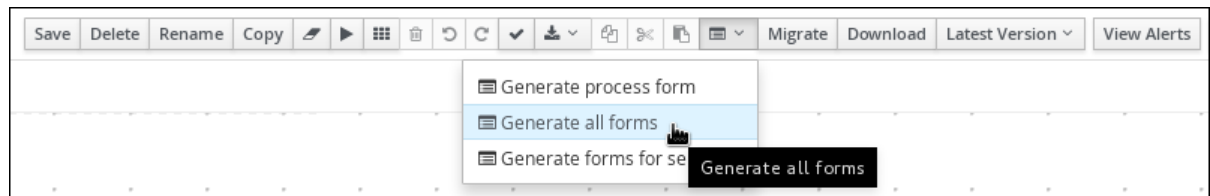
The forms are created in the root directory of your project.

5. Go to the root directory of your project in Business Central, click the new form name, and use the Form Modeler to customize the form to meet your requirements.

10.3. MANUALLY CREATING FORMS IN BUSINESS CENTRAL

You can create task and process forms manually from your project asset view. This is another way to generate a form without selecting to generate forms from your business process. For example, the Form Modeler now supports creating forms from external data objects.

Procedure

1. In Business Central, go to **Menu → Design → Projects** and click the project name.

2. Click **Add Asset → Form**.

3. Provide the following information in the **Create new Form** window:

   - Form name (must be unique)

   - Package name

   - Model type: Select either **Business Process** or **Data Object**.

     - For the **Business Process** model type, select your business process from the **Select Process** drop-down menu, and then select the form that you want to create from the **Select Form** drop-down menu.

     - For the **Data Object** model type, select one of your project data objects from the **Select Data Object from Project** drop-down menu.

4. Click **Ok** to open the Form Modeler.

5. In the **Components** view on the left side of the Form Modeler, expand the **Model Fields** and **Form Controls** menus and create a new form by dragging your required fields and form controls to the canvas.

6. Click **Save** to save your changes.

10.4. DOCUMENT ATTACHMENTS IN A FORM OR PROCESS

Red Hat Process Automation Manager supports document attachments in forms using the **Document** form field. With the **Document** form field, you can upload documents that are required as part of a form or process.

To enable document attachments in forms and processes, complete the following procedures:

1. Set the document marshalling strategy.

2. Create a document variable in the business process.

3. Map the task inputs and outputs to the document variable.

10.4.1. Setting the document marshalling strategy

The document marshalling strategy for your project determines where documents are stored for use with forms and processes. The default document marshalling strategy in Red Hat Process Automation Manager is **org.jbpm.document.marshalling.DocumentMarshallingStrategy**. This strategy uses a **DocumentStorageServiceImpl** class that stores documents locally in your *PROJECT_HOME*/**docs** folder. You can set this document marshalling strategy or a custom document marshalling strategy for your project in Business Central or in the **kie-deployment-descriptor.xml** file.

Procedure

1. In Business Central, go to **Menu** → **Design** → **Projects**.

2. Select a project. The project **Assets** window opens.

3. Click the **Settings** tab.

Figure 10.3. Settings tab
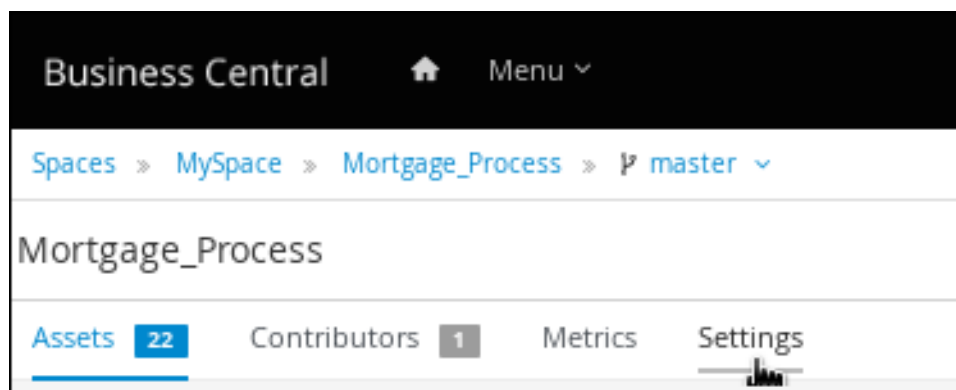
4. Click **Deployments** → **Marshalling Strategies**→ **Add Marshalling Strategy**.

5. In the **Name** field, enter the identifier of a document marshalling strategy, and in the **Resolver** drop-down menu, select the corresponding resolver type:

   - For single documents: Enter **org.jbpm.document.marshalling.DocumentMarshallingStrategy** as the document marshalling strategy and set the resolver type to **Reflection**.

   - For multiple documents: Enter **new org.jbpm.document.marshalling.DocumentCollectionImplMarshallingStrategy(new org.jbpm.document.marshalling.DocumentMarshallingStrategy())** as the document marshalling strategy and set the resolver type to **MVEL**.

   - For custom document support: Enter the identifier of the custom document marshalling strategy and select the relevant resolver type.

6. Click **Test** to validate your deployment descriptor file.

7. Click **Deploy** to build and deploy the updated project.

Alternatively, if you are not using Business Central, you can navigate to
*PROJECT_HOME***/src/main/resources/META_INF/kie-deployment-descriptor.xml** (if
applicable) and edit the deployment descriptor file with the required **<marshalling-strategies>**
elements.

8. Click **Save**.

Example deployment descriptor file with document marshalling strategy for multiple documents

```
<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>
```

10.4.1.1. Using a custom document marshalling strategy for a content management system (CMS)

The document marshalling strategy for your project determines where documents are stored for use
with forms and processes. The default document marshalling strategy in Red Hat Process Automation
Manager is **org.jbpm.document.marshalling.DocumentMarshallingStrategy**. This strategy uses a
DocumentStorageServiceImpl class that stores documents locally in your *PROJECT_HOME***/docs**
folder. If you want to store form and process documents in a custom location, such as in a centralized
content management system (CMS), add a custom document marshalling strategy to your project. You
can set this document marshalling strategy in Business Central or in the **kie-deployment-descriptor.xml** file directly.

Procedure

1. Create a custom marshalling strategy **.java** file that includes an implementation of the
   org.kie.api.marshalling.ObjectMarshallingStrategy interface. This interface enables you to
   implement the variable persistence required for your custom document marshalling strategy.
   The following methods in this interface help you create your strategy:

   - **boolean accept(Object object)**: Determines if the specified object can be marshalled by
     the strategy

   - **byte[] marshal(Context context, ObjectOutputStream os, Object object)**: Marshals the
     specified object and returns the marshalled object as **byte[]**

   - **Object unmarshal(Context context, ObjectInputStream is, byte[] object, ClassLoader
     classloader)**: Reads the object received as **byte[]** and returns the unmarshalled object

- **void write(ObjectOutputStream os, Object object)**: Same as the **marshal** method, provided for backward compatibility

- **Object read(ObjectInputStream os)**: Same as the **unmarshal** method, provided for backward compatibility

The following code sample is an example **ObjectMarshallingStrategy** implementation for storing and retrieving data from a Content Management Interoperability Services (CMIS) system:

Example implementation for storing and retrieving data from a CMIS system

```java
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;
```

```java
    }

    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");
    }
   }
  }
```

```java
    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 {
```

```java
  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 {
```
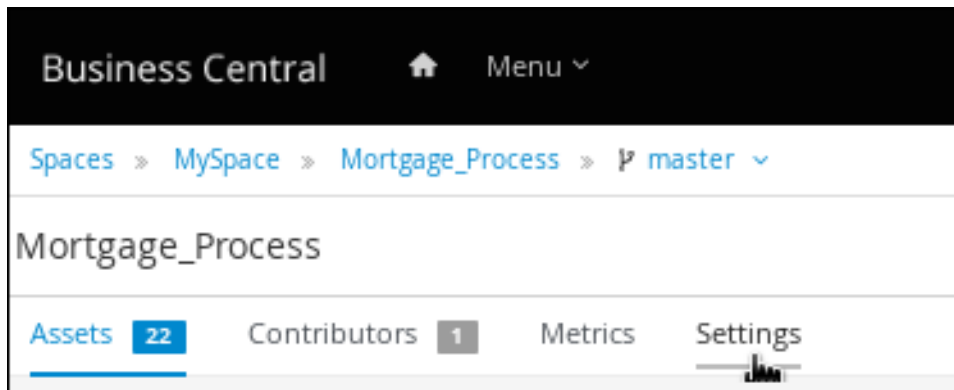
```
      session.clear();
    }
   }


   }
```

2. In Business Central, go to **Menu → Design → Projects**.

3. Click the project name and click **Settings**.

   Figure 10.4. Settings tab

4. Click **Deployments → Marshalling Strategies→ Add Marshalling Strategy**.

5. In the **Name** field, enter the identifier of the custom document marshalling strategy, such as **org.jbpm.integration.cmis.impl.OpenCMISPlaceholderResolverStrategy** in this example.

6. Select the relevant option from the **Resolver** drop-down menu, such as **Reflection** in this example.

7. Click **Test** to validate your deployment descriptor file.

8. Click **Deploy** to build and deploy the updated project.
   Alternatively, if you are not using Business Central, you can navigate to
   PROJECT_HOME/src/main/resources/META_INF/kie-deployment-descriptor.xml (if
   applicable) and edit the deployment descriptor file with the required **<marshalling-strategies>**
   elements.

   Example deployment descriptor file with custom document marshalling strategy

   ```xml
   <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. To enable documents stored in a custom location to be attached to forms and processes, create a document variable in the relevant processes and map task inputs and outputs to that document variable in Business Central.

10.4.2. Creating a document variable in a business process

After you set a document marshalling strategy, create a document variable in the related process to upload documents to a human task and for the document or documents to be visible in the **Process Instances** view in Business Central.

Prerequisites

- You have set a document marshalling strategy as described in Section 10.4.1, "Setting the document marshalling strategy".

Procedure

1. In Business Central, go to **Menu → Design → Projects**.

2. Click the project name to open the asset view and click the business process name.

3. Click the canvas and click on the right side of the window to open the **Diagram properties** panel.

4. Expand **Process Data** and click and enter the following values:

   - Name: **document**

   - Custom Type: **org.jbpm.document.Document** for a single document or **org.jbpm.document.DocumentCollection** for multiple documents

10.4.3. Mapping task inputs and outputs to the document variable

If you want to view or modify the attachments inside of task forms, create assignments inside of the task inputs and outputs.

Prerequisties

- You have a project that contains a business process asset that has at least one user task.

Procedure

1. In Business Central, go to **Menu → Design → Projects**.

2. Click the project name to open the asset view and click the business process name.

3. Click a user task and click on the right side of the window to open the **Diagram properties** panel.

4. Expand **Implementation/Execution** and next to **Assignments**, click to open the **Data I/O** window.

5. Next to **Data Inputs and Assignments**, click **Add** and enter the following values:

   - Name: **taskdoc_in**

   - Data Type: **org.jbpm.document.Document** for a single document or **org.jbpm.document.DocumentCollection** for multiple documents

   - Source: **document**

6. Next to **Data Outputs and Assignments**, click **Add** and enter the following values:

   - Name: **taskdoc_out**

   - Data Type: **org.jbpm.document.Document** for a single document or **org.jbpm.document.DocumentCollection** for multiple documents

   - Target: **document**

   The **Source** and **Target** fields contain the name of the process variable you created earlier.

7. Click **Save**.

CHAPTER 11. ADVANCED PROCESS CONCEPTS AND TASKS

11.1. INVOKING A DECISION MODEL AND NOTATION (DMN) SERVICE IN A BUSINESS PROCESS

You can use Decision Model and Notation (DMN) to model a decision service graphically in a decision requirements diagram (DRD) in Business Central and then invoke that DMN service as part of a business process in Business Central. Business processes interact with DMN services by identifying the DMN service and mapping business data between DMN inputs and the business process properties.

As an illustration, this procedure uses an example **TrainStation** project that defines train routing logic. This example project contains the following data object and DMN components designed in Business Central for the routing decision logic:

Example Train object

```
public class Train {

    private String departureStation;

    private String destinationStation;

    private BigDecimal railNumber;

    // Getters and setters
}
```

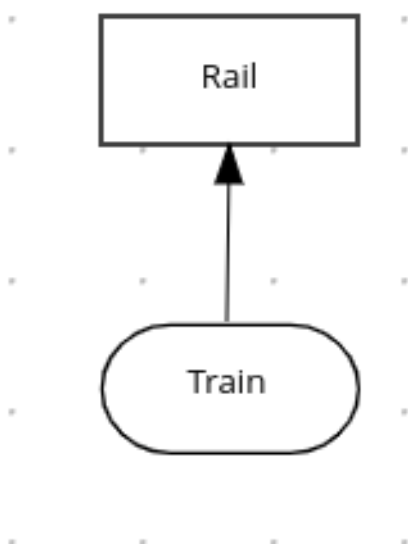**Figure 11.1. Example Compute Rail DMN model**

Figure 11.2. Example **Rail** DMN decision table

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	

Figure 11.3. Example **tTrain** DMN data type

tTrain (Structure)

departureStation (string)

destinationStation (string)

For more information about creating DMN models in Business Central, see *Designing a decision service using DMN models*.

Prerequisites

- All required data objects and DMN model components are defined in the project.

Procedure

1. In Business Central, go to **Menu → Design → Projects** and click the project name.

2. Select or create the business process asset in which you want to invoke the DMN service.

3. In the process designer, use the left toolbar to drag and drop BPMN components as usual to define your overall business process logic, connections, events, tasks, or other elements.

4. To incorporate a DMN service in the business process, add a **Business Rule** task from the left toolbar or from the start-node options and insert the task in the relevant location in the process flow.

For this example, the following **Accept Train** business process incorporates the DMN service in the **Route To Rail** node:

Figure 11.4. Example Accept Train business process with a DMN service
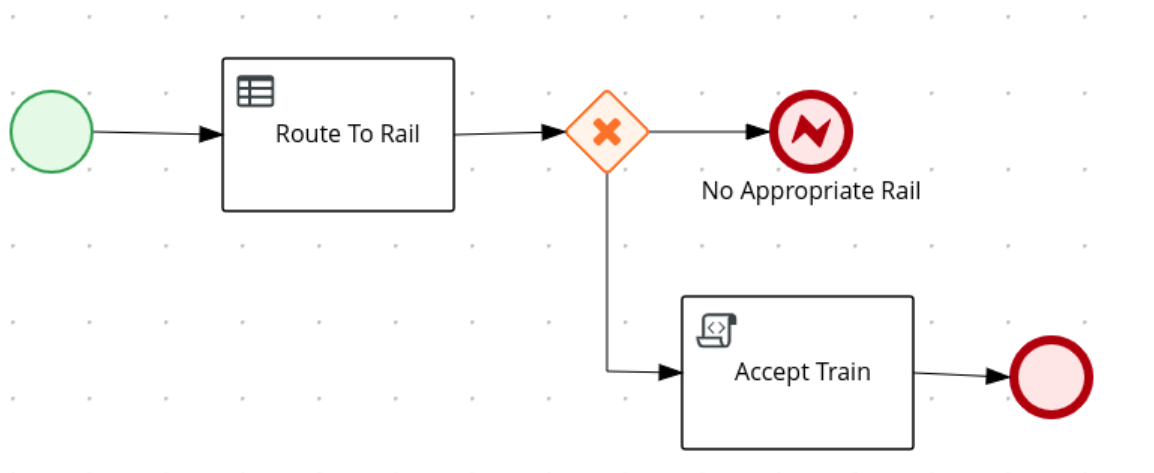
5. Select the business rule task node that you want to use for the DMN service, click **Diagram properties** in the upper-right corner of the process designer, and under **Implementation/Execution**, define the following fields:

   - **Rule Language**: Select **DMN**.

   - **Namespace**: Enter the unique namespace from the DMN model file. Example: **https://www.drools.org/kie-dmn**

   - **Decision Name**: Enter the name of the DMN decision node that you want to invoke in the selected process node. Example: **Rail**

   - **DMN Model Name**: Enter the DMN model name. Example: **Compute Rail**

   IMPORTANT

   When you explore the root node, ensure that the **Namespace** and **DMN Model Name** fields consist of the same value in BPMN as DMN diagram.

6. Under **Data Assignments → Assignments**, click the **Edit** icon and add the DMN input and output data to define the mapping between the DMN service and the process data. For the **Route To Rail** DMN service node in this example, you add an input assignment for **Train** that corresponds to the input node in the DMN model, and add an output assignment for **Rail** that corresponds to the decision node in the DMN model. The **Data Type** must match the type that you set for that node in the DMN model, and the **Source** and **Target** definition is the relevant variable or field for the specified object.

Figure 11.5. Example input and output mapping for the **Route To Rail** DMN service node

7. Click **Save** to save the data input and output data.

8. Define the remainder of your business process according to how you want the completed DMN service to be handled.
   For this example, the **Diagram properties → Implementation/Execution → On Exit Action** value is set to the following code to store the rail number after the **Route To Rail** DMN service is complete:

Example code for On Exit Action

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

If the rail number is not computed, the process reaches a **No Appropriate Rail** end error node that is defined with the following condition expression:

Figure 11.6. Example condition for**No Appropriate Rail** end error node

Implementation/Execution

Priority

Condition Expression

◉ Condition   ○ Expression

Process Variable ⓘ

train.railNumber

Condition ⓘ

Is null

If the rail number is computed, the process reaches an **Accept Train** script task that is defined with the following condition expression:

Figure 11.7. Example condition for**Accept Train** script task node

Implementation/Execution

Priority

Condition Expression

◉ Condition   ○ Expression

Process Variable ⓘ

train.railNumber

Condition ⓘ

Greater than

Min Value ⓘ

0

The **Accept Train** script task also uses the following script in **Diagram properties →
Implementation/Execution → Script** to print a message about the train route and current rail:

```
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. After you define your business process with the incorporated DMN service, save your process in the process designer, deploy the project, and run the corresponding process definition to invoke the DMN service.

   For this example, when you deploy the **TrainStation** project and run the corresponding process definition, you open the process instance form for the **Accept Train** process definition and set the **departure station** and **destination station** fields to test the execution:

   Figure 11.8. Example process instance form for the **Accept Train** process definition

   After the process is executed, a message appears in the server log with the train route that you specified:

   Example server log output for the **Accept Train** process

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

CHAPTER 12. ADDITIONAL RESOURCES

- *Getting started with business processes*

- *Process designer Business Process Model and Notation (BPMN2) reference guide*

- *Managing and monitoring business processes in Business Central*

- *Interacting with processes and tasks*

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Thursday, September 03, 2020.