Red Hat Process Automation Manager 7.8

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

Last Updated: 2020-09-08

Red Hat Process Automation Manager 7.8 Process designer Business Process Model and Notation (BPMN2) reference guide

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

Legal Notice

Abstract

This document is a BPMN2 reference guide for the Red Hat Process Automation Manager 7.8 legacy and new process designers.

Table of Contents

PREFACE

This document provides an explanation of the BPMN2 elements in the Red Hat Process Automation Manager 7.8 legacy and new process designers. As a process manager, you can refer to this document for a detailed description of BPMN2 symbols and their uses when designing business or case management processes in Business Central. For more details about BPMN2, see the Business Process Model and Notation Version 2.0 specification.

NOTE

The legacy process designer is deprecated. It will not be updated and will be disabled in the next release.

If you import processes created with the legacy process designer into the new process designer, send, receive, and manual tasks are converted to empty tasks and group elements and data object elements are dropped.

CHAPTER 1. 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 1.1. 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.

CHAPTER 2. 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 2.1. 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 2.2. BPMN2 catching events

Element Name	Start	Intermediate
None	✓	
Message	✓	✓
Timer	✓	✓

Element Name	Start	Intermediate
Error	✓	✓
Escalation	✓	✓
Cancel		✗
Compensation	✓	✓
Conditional	✓	✓
Link		✗
Signal	✓	✓
Multiple	✗	✗
Parallel Multiple	✗	✗

Table 2.3. BPMN2 throwing and non-interrupting events

Element Name	Throwing		Non-interrupting	
	End	Intermediate	Start	Intermediate
None	✓			

Element Name	Throwing		Non-interrupting	
Message	✓	✓	✓	✓
Timer			✓	✓
Error	✓			
Escalation	✓	✓	✓	✓
Cancel	✗	✗		✗
Compensation	✓	✓		
Conditional			✓	✓
Link		✗		
Signal	✓	✓	✓	✓
Terminate	✓			
Multiple	✗	✗	✗	✗

Element Name	Throwing		Non-interrupting	
Parallel Multiple			✗	✗

Table 2.4. BPMN2 elements

Element type	Element	Supported
Task	Business rule	✓
	Script	✓
	User task	✓
	Service task	✓
Subprocesses, including multiple instance subprocesses	Embedded	✓
	Ad hoc	✓
	Reusable	✓
	Event	✓
Gateways	Inclusive	✓
	Exclusive	✓

Element type	Element	Supported
	Parallel	✓
	Event-based	✓
	Complex	✗
Connecting objects	Sequence flows	✓
	Association flows	✓
Swimlanes	Swimlanes	✓
Artifacts	Group	☆
	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.

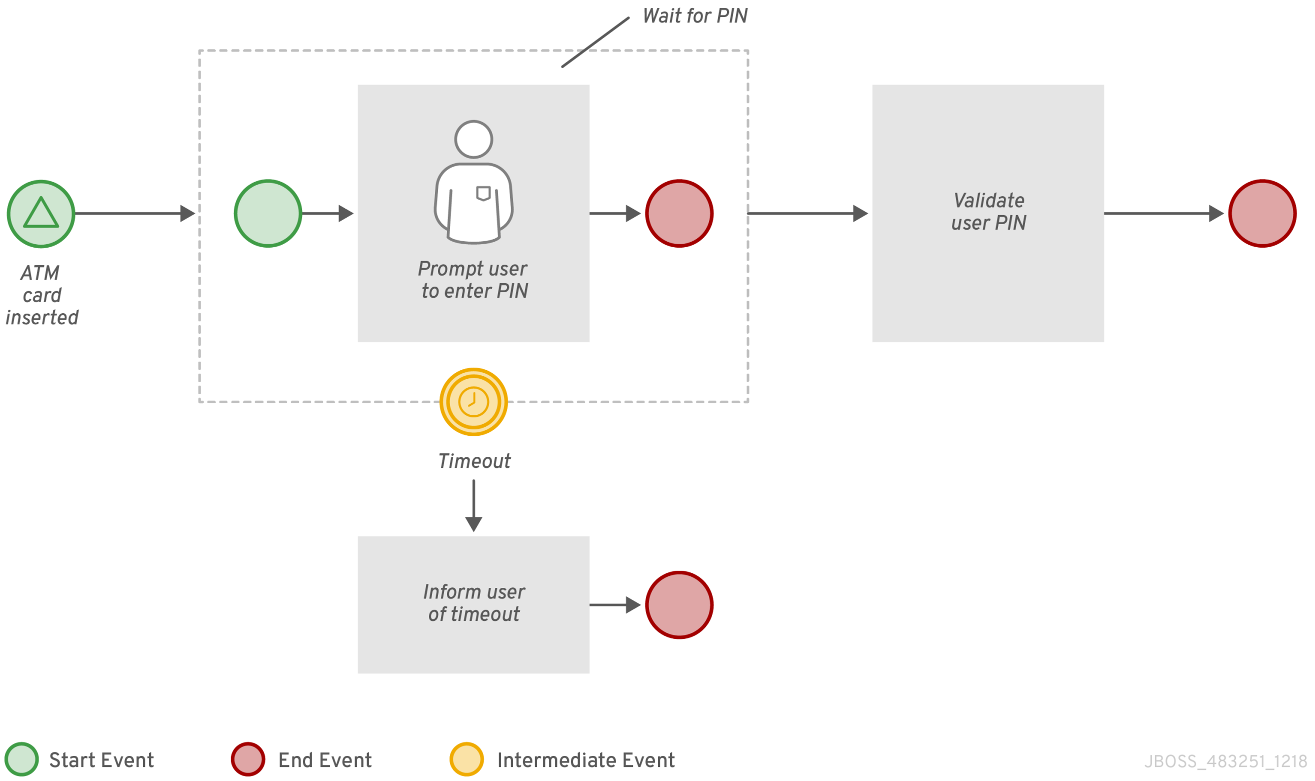
CHAPTER 3. BPMN2 EVENTS IN PROCESS DESIGNER

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:

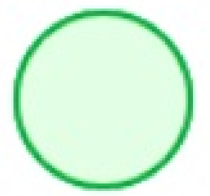
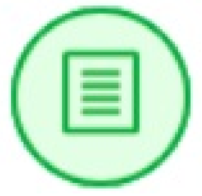
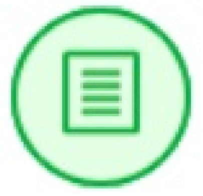
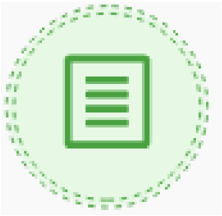
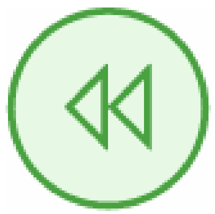
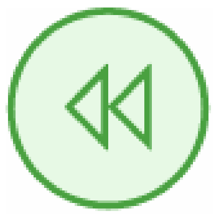
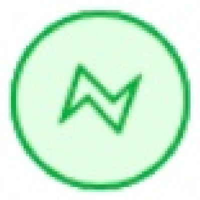
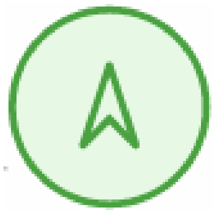
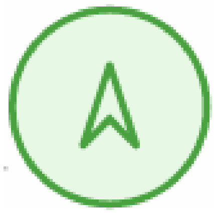
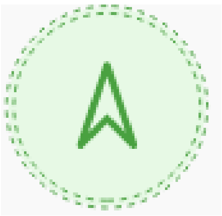
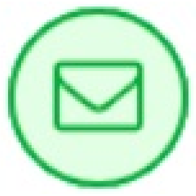
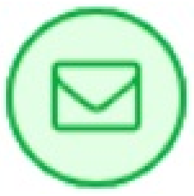
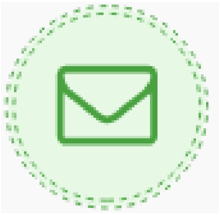
- The ATM Card Inserted signal start event is triggered when the signal is received.
- The timeout intermediate event is an interrupting event based on a timer trigger. This means that the Wait for PIN subprocess is canceled when the timer event is triggered.
- Depending on the inputs to the process, 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.

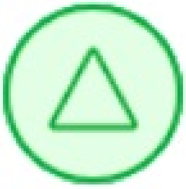
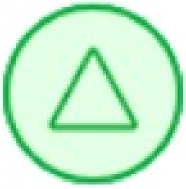
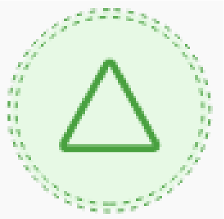
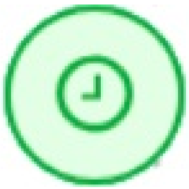
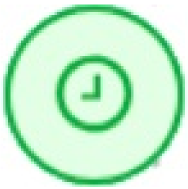
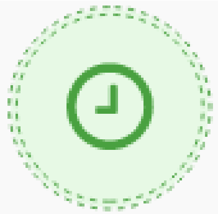
3.1. START EVENTS

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.

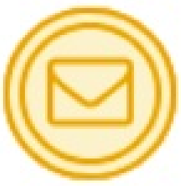
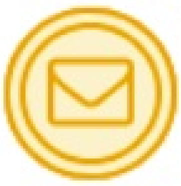
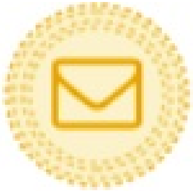
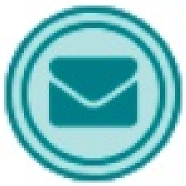
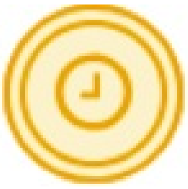
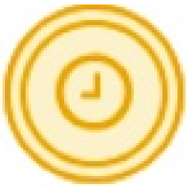
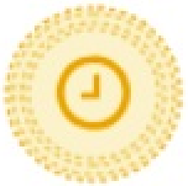
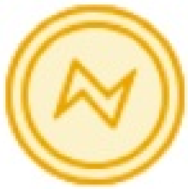
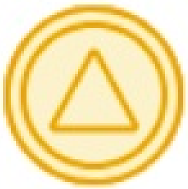
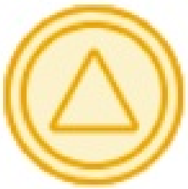
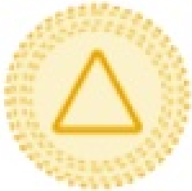
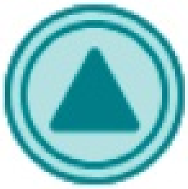
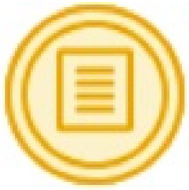
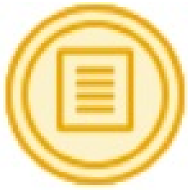
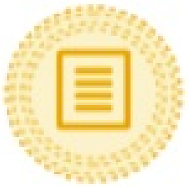
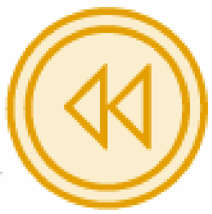
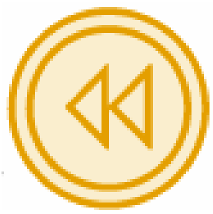
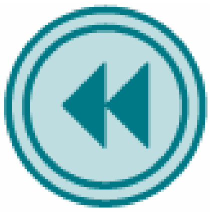
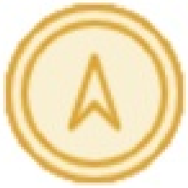
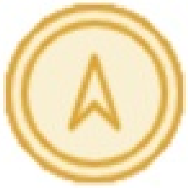
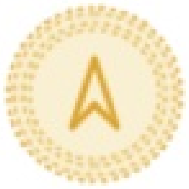
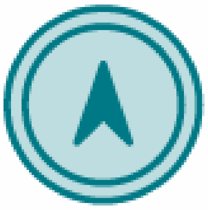
An intermediate event handles a particular situation that occurs during process execution. The situation is a trigger for an intermediate event. In a process, intermediate events with one outgoing flow can be placed on an activity boundary.

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.

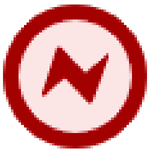
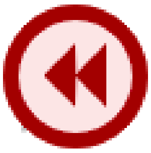
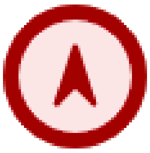
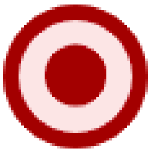
End events indicate the completion of a business process. An end event is a node that ends a particular workflow. It has one or more incoming sequence flows and no outgoing flow.

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

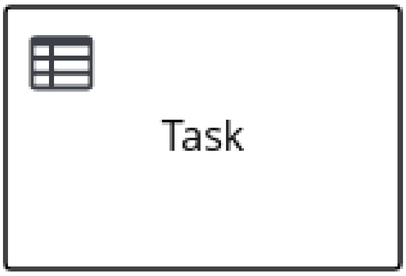
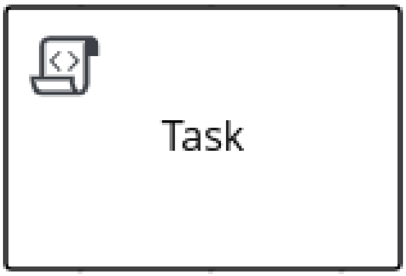
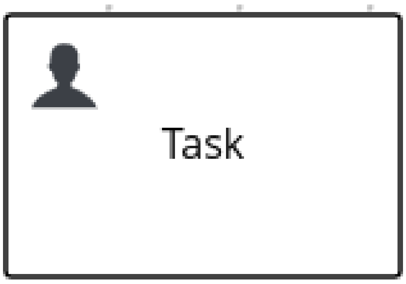
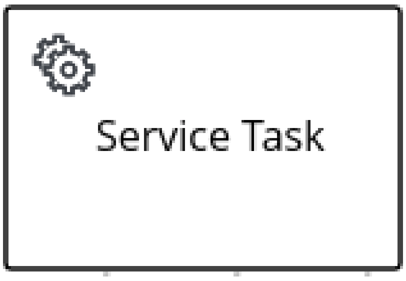
The terminate end event finishes all execution flows in the specified process instance. Activities being executed are canceled. The subprocess instance terminates if it reaches a terminate end event.

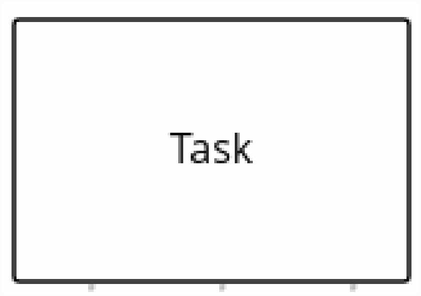
CHAPTER 4. BPMN2 TASKS IN PROCESS DESIGNER

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 4.1. Task

Business rule task	
Script task	
User task	
Service task	

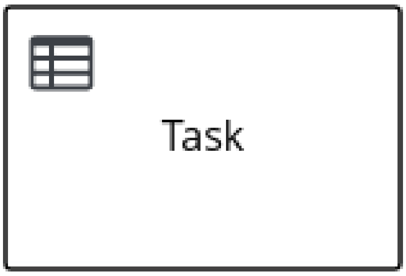
None task	

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.

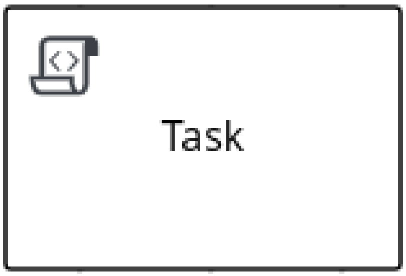


When a process reaches a business rule task defined by a DMN model, the process engine executes the DMN model decision with the inputs provided.

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.



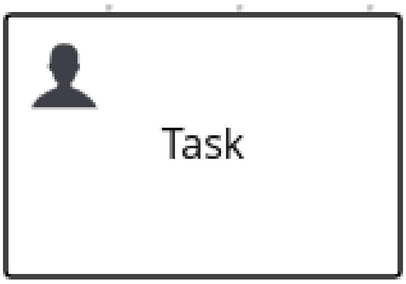
The associated script can access process variables and global variables. Review the following list before using a script task:

- 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.



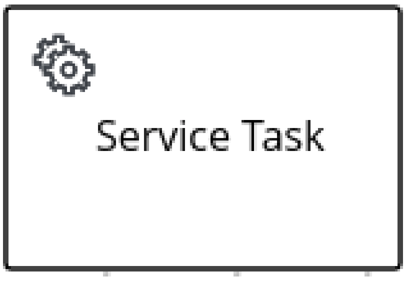
On execution, the User task element is instantiated as a task that appears in the list of tasks of one or more actors. If a User task element defines the **Groups** attribute, it is displayed in task lists of all users that are members of the group. Any user who is a member of the group can claim the 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.

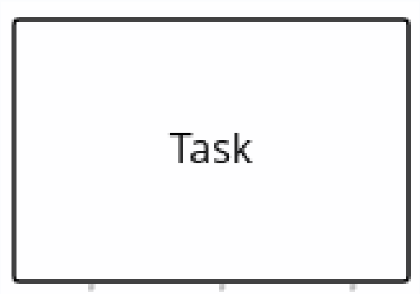
Service task

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.



CHAPTER 5. BPMN2 SUBPROCESSES IN PROCESS DESIGNER

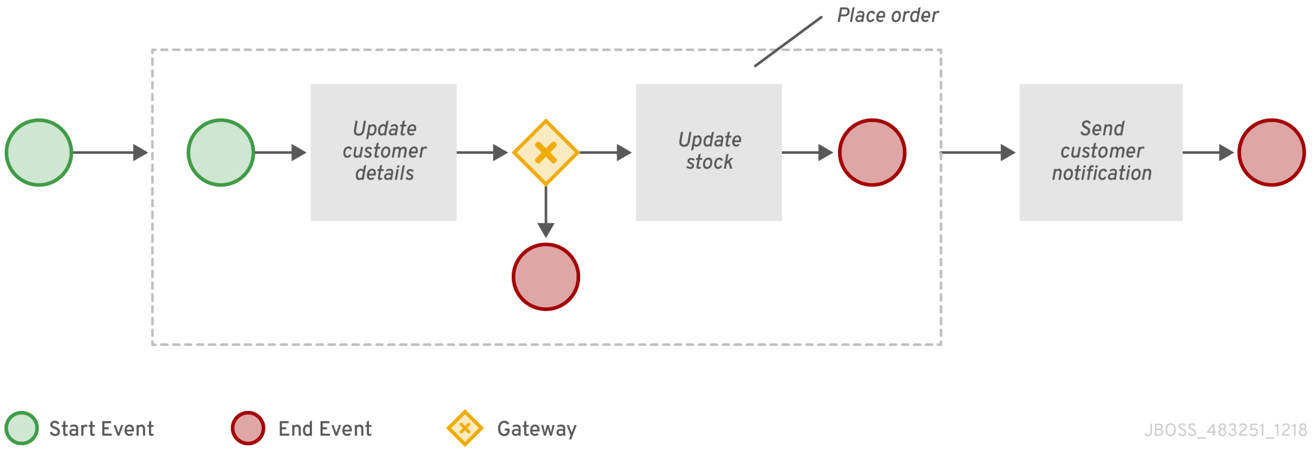
A subprocess is an activity that contains nodes. You can embed part of the main process within a subprocess. You can also include variable definitions within the subprocess. These variables are accessible to all nodes inside the subprocess.

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.



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.

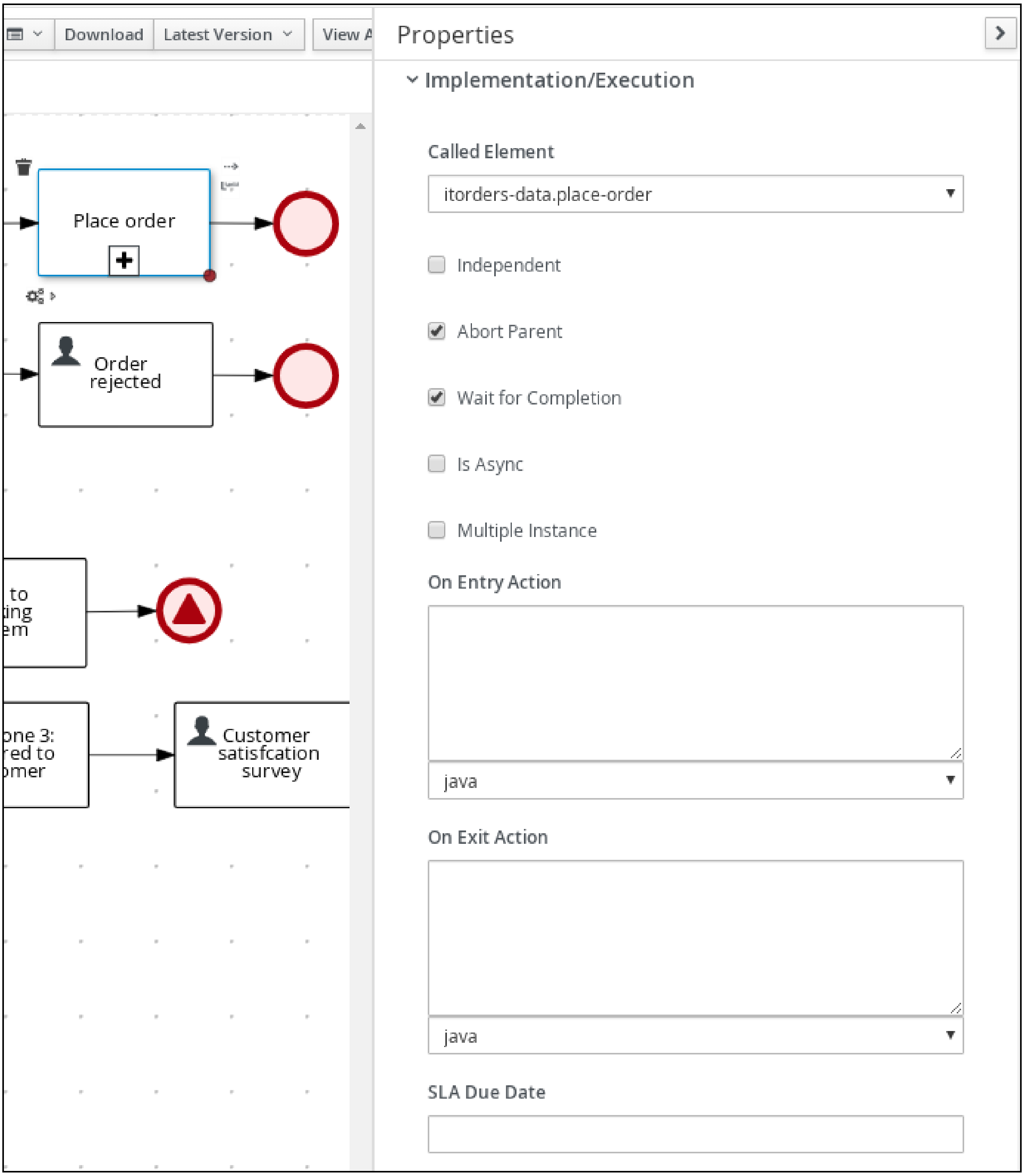
Reusable subprocess

Reusable subprocesses appear collapsed within the parent process. To configure a reusable subprocess, select the reusable subprocess, click ✎ , 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 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 5.1. Reusable subprocess properties



Properties

Implementation/Execution

Called Element: itorders-data.place-order

□ Independent

☑ Abort Parent

☑ Wait for Completion

□ Is Async

□ Multiple Instance

On Entry Action: java

On Exit Action: java

SLA Due Date

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.

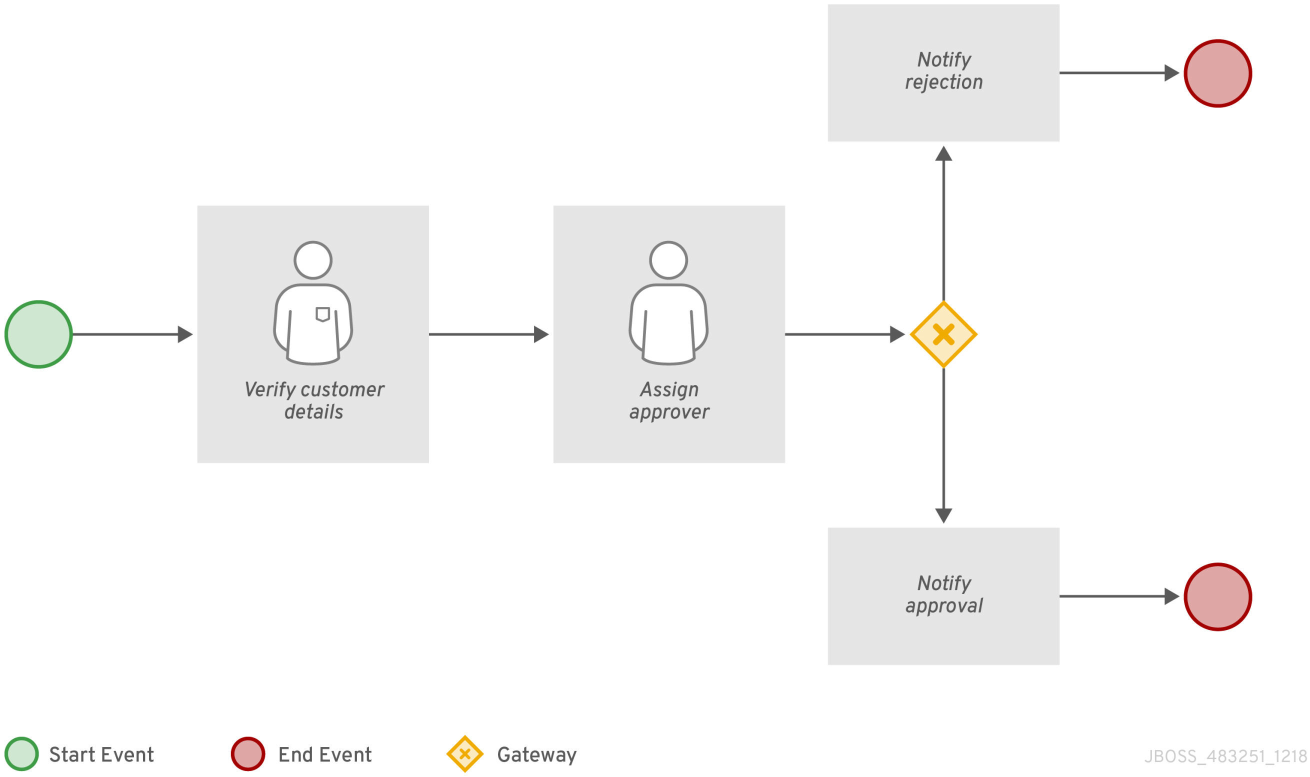
CHAPTER 6. BPMN2 GATEWAYS IN PROCESS DESIGNER

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

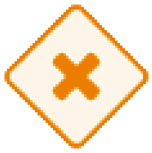
One gateway cannot have multiple incoming and multiple outgoing flows.

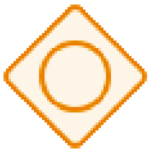
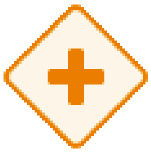
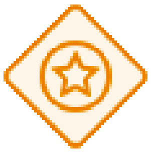
In the following business process diagram, the XOR gateway evaluates only the incoming flow whose condition evaluates to true:



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 6.1. Gateway elements

Element type	Icon
exclusive (XOR)	

Element type	Icon
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.

CHAPTER 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.

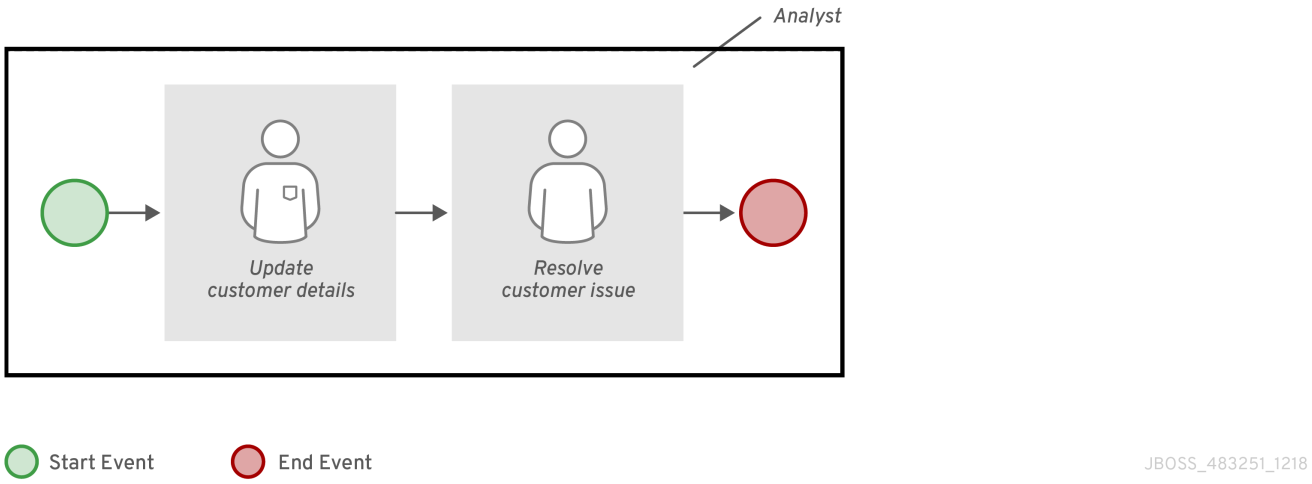
NOTE

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

CHAPTER 8. 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 9. BPMN2 ARTIFACTS IN PROCESS DESIGNER

Artifacts are used to provide additional information about a process. An artifact is any object depicted in the BPMN2 diagram that is not part of the process workflow. Artifacts have no incoming or outgoing flow objects.The purpose of artifacts is to provide additional information required to understand the diagram. The artifacts table lists the artifacts supported in the legacy process designer.

Table 9.1. Artifacts

Artifact type	Description
Group	Organizes tasks or processes that have significance in the overall process. Group artifacts are not supported in the new process designer.
Text annotation	Provides additional textual information for the BPMN2 diagram.

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Thursday, September 08, 2020.