BPMN:

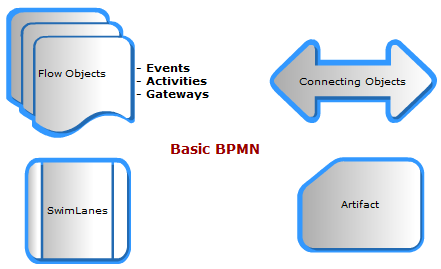
<http://www.mastertheboss.com/jboss-jbpm/activiti-bpmn/activiti-bpmn-on-jboss-as-7> bvn

BPMN tutorial for beginners.

**BPMN**stands for Business Process Modeling Notation. It describes a business-friendly, flow chart-like graphical notation that business process analysts and business users can use to model business processes and has support for process interactions, exception handling, compensation semantics, and so on.

The vision of BPMN is to have a single specification for notation, metamodel, and interchange. In addition, **BPMN 2.0**has been expanded to include orchestrations and choreography of process models.  
  
A Business Process Diagram is a simple diagram made up of a set of graphical elements that depicts a business process. There are four primary elements of BPD:

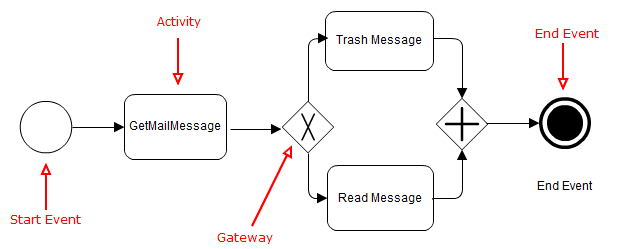
* **Flow Objects:** represent the core elemets of the business process diagram.
* **Connecting Objects:** are used to connect the BPMN core objects
* **Swimlanes:** are mechanism to organize activities and responsibilities on a process diagram.
* **Artifacts:** allow process designers to extend the basic BPMN notation to include additional information about the process in the process diagram



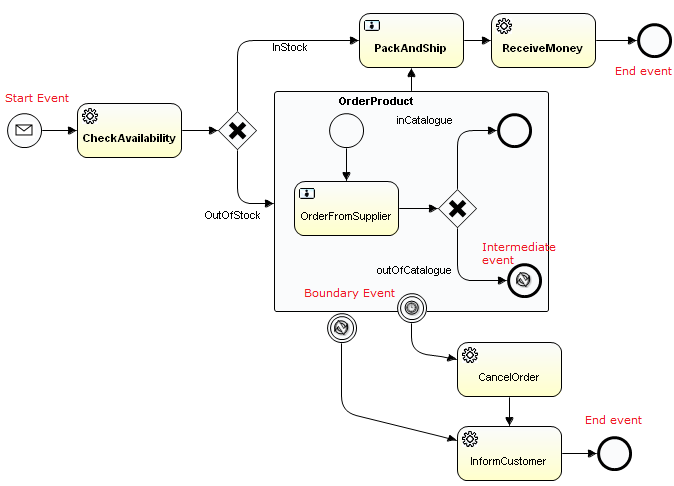
As we said,**Flow Objects** are shapes that represent the **core elements** of the Business Process Diagram (BPD), including:

* **Activity:** is any work that is being performed in a process.
* **Event:** are anything that "happens" during the course of a business process.
* **Gateway:** is used to control the flow of a process.

The following diagram depicts a sample process which includes a Start Event, an Activity (a Task), some Gateways and an End Event:



The biggest difference between **BPMN**and traditional flowcharting is the support for events.  An event is a signal that something happened, and BPMN lets you say how the process should respond.  
**Here Important definitions!**  
The Events can occur at the beginning ( **Start Events**) or at the end (**End Events**) or in the middle (**Intermediate**) of the process.  Additionally, some events can happen on the **Boundary**of an activity. This indicates that the event can interrupt the activity, and will divert the sequence flow from the “normal” flow to another flow.  
  
The following picture depicts a process of ordering some products which includes a **Start event** (a message event), of an **Intermediate error event**, a **Boundary Timer event**, and two **End events**.



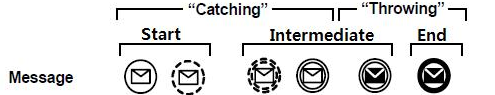
The Events can be of **catch**(when a receive trigger is fired) or **throw**(which sends a trigger to the process) type.

The Start Events are *always*of catch type and the End Events are *always*of throw type. The Intermediate Events can be either of throw or catch type.There are various flavors of Events in BPMN 2.0. The Event Types are listed as follows:

|  |  |  |
| --- | --- | --- |
| bpmn 2.0 tutorial introduction quickstart events | **Message Events** | Send or receive messages. |
| bpmn 2.0 tutorial introduction quickstart events | **Timer Events** | Are always of catch type and used to signify waiting for a specific time condition to evaluate to true. |
| bpmn 2.0 tutorial introduction quickstart events | **Signal Events** | Are used for publish and subscribe of signals. |
| bpmn 2.0 tutorial introduction quickstart events | **Error Events** | Are used for exception handling and they can occur only at the end of the process. |
| bpmn 2.0 tutorial introduction example events quickstart | **Termination Event** | Are used to terminate the process and can occur only at the end of the process. |
| bpmn 2.0 tutorial introduction example events quickstart | **Conditional Event** | Are used for rule-based triggers. |
| bpmn 2.0 tutorial introduction example events quickstart | **Escalation Event** | Has been newly introduced in BPMN 2.0 to handle escalation conditions. |
| bpmn 2.0 tutorial introduction example events quickstart | **Compensation Event** | Introduced to to handle compensation in the process. |

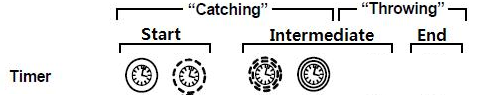
Depending on the combination of events, phase in which the event is triggered (Start, Intermediate, Boundary) and if the event interrupts an Activity, the event can be represented with a different combination of graphic notations.

# Message Events

Message events are events which reference a named message. A message has a name and a payload and is always directed at a single receiver.  
The graphical representation of the message can vary depending on the phase in which the message is sent/received  
  
**Start:** A Message arrives from a participant and triggers the start of the Process.  
  
**Intermediate Event:** A message arrives from a participant and triggers the Event. This causes the process to continue if it was waiting for the message, or changes the flow for exception handling. When used to “catch” the message, then the Event marker will be unfilled. When used to “throw” the message, the Event marker will be filled.  
  
**Intermediate Boundary event:** A Message arrives from a participant and triggers the Event. If a Message Event is attached to the boundary of an Activity, it will change the Normal Flow into an Exception Flow upon being triggered.  
For a Message Event that interrupts the Activity to which it is attached, the boundary of the Event is solid.  
For a Message Event that does not interrupt the Activity to which it is attached, the boundary of the Event is dashed.  
  
**End:** This type of End indicates that a message is sent to a participant at the conclusion of the Process.

# Timer Events

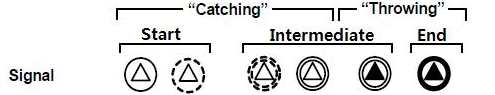
Timer events are events which are triggered by defined timer. They can be used as start event, intermediate event or boundary event.



**Start:** A specific time-date or a specific cycle (e.g., every day at 9am) can be set that will trigger the start of the Process.  
  
**Intermediate Event:** A specific time-date or a specific cycle (e.g., every Tuesday at 9am) can be set that will trigger the Event. If used within the main flow it acts as a delay mechanism. If used for exception handling it will change the Normal Flow into an Exception Flow.  
  
**Intermediate boundary:** A specific time-date or a specific cycle (e.g., every day at 9am) can be set that will trigger the Event. If a Timer Event is attached to the boundary of an Activity, it will change the Normal Flow into an Exception Flowupon being triggered.

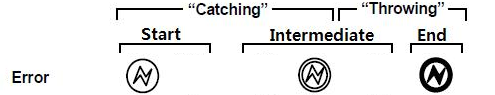
For a Timer Event that interrupts the Activity to which it is attached, the boundary of the Event is solid. For a Timer Event that does not interrupt the Activity to which it is attached, the boundary of the Event is dashed.

# Signal Events

This type of event is used for sending or receiving Signals. A Signal is for general communication within the process components. A BPMN Signal is similar to a signal flare that shot into the sky for anyone who might be interested to notice and then react. Thus, there is a source of the Signal, but no specific intended target. **This is different than a BPMN Message, which has a specific Source and a specific Target** (which can be an Entity or an abstract Role).   
  
**Start:**A signal arrives that has been broadcast from another Process and triggers the start of the Process. Note that the Signal is not a Message, which has a specific target for the Message.

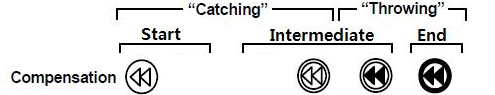
**Intermediate Event:** This type of Intermediate Event can send or receive a Signal if the Event is part of a Normal Flow. A signal event can be caught by an intermediate catch signal event.  
  
**Intermediate boundary:** When attached to the boundary of an activity, the signal will change the Normal Flow into an Exception Flow upon being triggered. The Signal Event differs from an Error Event in that the Signal defines a more general, non-error condition for interrupting activities (such as the successful completion of another activity) as well as having a larger scope than Error Events.  
For a Signal Event that interrupts the Activity to which it is attached, the boundary of the Event is solid. For a Signal Event that does not interrupt the Activity to which it is attached, the boundary of the Event is dashed.  
  
**Error:**This type of End indicates that a Signal will be broadcasted when the End has been reached

# Error Events

  
**Start:** the Error Start Event is only allowed for triggering an in-line Event Sub-Process. Given the nature of Errors, an Event Sub-Process with an Error trigger will always interrupt its containing Process.  
  
**Intermediate boundary:** An Intermediate Error Catch Event can only be attached to the boundary of an activity.Note that an Error Event always interrupts the Activity to which it is attached, i.e., there is not a non-interrupting version of this Event. The boundary of the Event is thus always solid.  
  
**End:** This type of End indicates that a named Error should be generated. The Error will be caught by the Error intermediate event with the same ErrorCode or no ErrorCode which is on the boundary of the nearest enclosing parent activity (hierarchically).

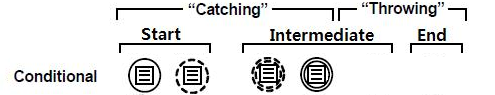
# Compensation Events

Compensation is a means for “undoing” the effects of an action.

For example, supposing you have booked a ticket to a show at the beginning of your process, then a compensation could be cancelling the reservation.  
  
**Start:** The Compensation Start Event is only allowed for triggering an in-line Compensation Event Sub-Process. This type of Event is triggered when compensation occurs. This Event does not interrupt the Process since the Process has to be completed before this Event can be triggered.  
  
**Intermediate:** This is used for compensation handling - both activating and performing compensation.  
When used in Normal flow, this Intermediate Event indicates that a Compensation is necessary. Thus, it is used to “throw” the Compensation event, and the Event marker MUST be filled. If the Event identifies an activity, then that is the activity (and no other) that will be compensated. Otherwise, the compensation is broadcast to all activities that have completed within the Process Instance, including the top-level Process and including all Sub-Processes. Each completed activity that is subject to compensation will be compensated, in the reverse order of the completion of the activities. To be compensated, an activity MUST have a Compensation Intermediate Event attached to its boundary.  
  
**Boundary:** A compensation boundary event has a different activation policy from other boundary events. Other boundary events like for instance the signal boundary event are activated when the activity they are attached to is started. The compensation boundary is activated when the activity is attached to completes successfully. At this point, the corresponding subscription to compensation events is created.  
  
When attached to the boundary of an Activity, this Event is used to "catch" the Compensation Event, thus the Event marker MUST be unfilled. The Event will be triggered by a thrown compensation targeting that Activity. When the Event is triggered, the Compensation Activity that is Associated to the Event will be performed.  
Note that the interrupting and non-interrupting aspect of other Events does not apply in the case of a Compensation Event. Compensations can only be triggered after completion of the activity to which they are attached.  
Thus they cannot interrupt the Activity. The boundary of the Event is always solid.  
  
**End:**This type of End indicates that a Compensation is necessary. If an activity is identified, then that is the activity that will be compensated. Otherwise, all activities that have completed within the Process, starting with the top-level Process and including all Sub-Processes, are subject to compensation, proceeding in reverse order. To be compensated, an activity MUST have a Compensation Intermediate Event attached to its boundary.

# Conditional Events

Real world business processes often embody complex decision making. Conditional events can thus be used for rule-based triggers which are included in the process.



**Start:** This type of event is triggered when a Rule Condition such as "S&P 500 changes by more than 10% since opening", or "Temperature above 300°C" become true. The ConditionExpression for the Event must become false and then true before the Event can be triggered again.  
  
**Intermediate:** This type of event is triggered when a Rule Condition becomes true.  
  
**Boundary:** This type of event is triggered when a Rule Condition becomes true. A Condition is a type of Expression. If a Conditional Event is attached to the boundary of an Activity, it will change the normal flow into an exception flow upon being triggered.  
For a Conditional Event that interrupts the Activity to which it is attached, the boundary of the Event is solid. For a Conditional Event that does not interrupt the Activity to which it is attached, the boundary of the Event is dashed.

--

Have you completed the [BPMN introduction tutorial](http://www.mastertheboss.com/jbpm/279-bpmn-tutorial-for-beginners.html)?  In this second part tutorial we will discuss about the two other core **BPMN**elements: **Activities** and **Gateways**.

An activity, represented with a rounded-corner rectangle, describes a kind of work which must be done. It commonly includes **Tasks**and **Subprocesses**.

BPMN task tutorial introduction

# BPMN Tasks

A **BPMN Task** is essentially an activity which needs to be performed either by a human actor or automatically. Prior to BPMN 2.0 the only way to understand what type of task is was look at in the lanes in the pool, or using a text annotation to clarify the intent of the digram.  BPMN 2.0 adds new icon annotations to the tasks. These icons might differ a bit depending on the BPMN software you are using, however I encourage you to use them because it brings a lot of clarity to a diagram.

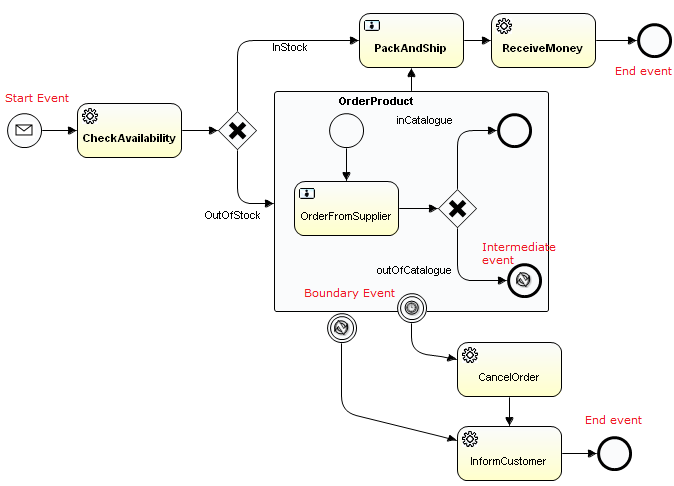
Let's see all type of tasks available:

|  |  |  |
| --- | --- | --- |
| BPMN tutorial introduction howto business process management | Human task | A human task is a task that requires intervention by a human (actor). For example the user fills up a form and submit his work to the manager. |
| BPMN tutorial introduction howto business process management | Manual task | A human task which is to be performed without the aid of any business process execution or any application. Example: phone call |
| BPMN tutorial introduction howto business process management | Service task | Represents an automated unit of work that should be executed in this process. All work that is executed outside the process engine should be represented (in a declarative way) using a Service Task. |
| BPMN tutorial introduction howto business process management | Call Activity | Represents a reusable global task or process. Call activity references a process that is external to the process definition, whereas the subprocess is embedded within the original process definition. |
| BPMN tutorial introduction howto business process management | Script Task | A script task is an automatic activity. When a process execution arrives at the script task, the corresponding script is executed. |
| BPMN tutorial introduction howto business process management | Send Task | A simple task that is designed to send a message to an external participant (relative to the process). Once the message has been sent, the task is completed. |
| BPMN tutorial introduction howto business process management | Receive Task | A simple task that is designed to wait for a message from an external participant (relative to the process). Once the message has been received, the task is completed. |
| BPMN tutorial introduction howto business process management | Business rule task | Provides a mechanism for the process to provide input to a business rule engine and to get the output of calculations that the business rules engine might provide. In the process model it does not take any action.  It just returns the rule result, which can be used by subsequent process logic in a variety of ways. |

# Sub-Processes

Sub-Processes are a very handy concept in BPMN.  Besides providing a natural way to draw a condensed top-down view with drill-down to any level of detail, BPMN subprocesses also determine the boundaries of when an event can be received – a change to an order in process, for example.

In terms of graphical aspect, a Sub-Process object shares the same shape as the Task object, which is a rounded rectangle. The Sub-Process can be in a collapsed view that hides its details (see picture)  
  
A Sub-Process can be in an expanded view that shows its details within the view of the Process in which it is contained. Here is an example of Sub-Process ("OrderProduct") which is part of an Inventory system. The subprocess is used to model and hide the complexity of the refurnish subprocess and to issue an Exception if goods cannot be ordered in time.



# Gateways

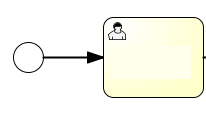
A **gateway**is used to control the flow of execution (or as the BPMN 2.0 describes, the tokens of execution). A gateway is capable of consuming or generating tokens.  
  
Gateways can define all the types of business process Sequence Flow behavior: Decisions/branching (exclusive, inclusive, and complex), merging, forking, and joining. A gateway is graphically visualized as a diamond shape, with an icon inside. BPMN, however, extends the behavior of the diamonds to reflect any type of Sequence Flow control. Each type of Gateway will have an internal indicator or marker to show the type of Gateway that is being used.

The following table summarizes the most common gateways used in a business process:

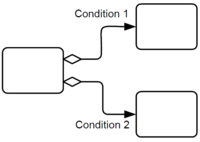
|  |  |  |
| --- | --- | --- |
| business process management model tutorial | Exclusive Gateway | Used to model a decision in the process. Only **one sequence flow is selected** when using the exclusive gateway. In case multiple sequence flow have a condition that evaluates to true, the first one defined in the XML is selected for continuing the process. |
| business process management model tutorial | Parallel Gateway | Allows to **fork**into multiple paths of execution or **join**multiple incoming paths of execution. When used to fork the sequence flow, all outgoing branches are activated simultaneously. When joining parallel branches it waits for all incoming branches to complete before moving to the outgoing flow |
| business process management model tutorial | Inclusive Gateway | Inclusive gateway is also a division point of the business process. Unlike the exclusive gateway, inclusive gateway may trigger **more than 1 out-going paths.** Thus, all out-going conditions will be evaluated no matter has fulfilled out-going flow or not. |

# Sequence Flows

A sequence flow is the connector between two elements of a process. After an element is visited during process execution, **all outgoing sequence flow will be followed**. A sequence flow is visualized as an arrow going from the source element towards the target element. The arrow always points towards the target.



A Sequence Flow MAY have a **conditional expression attribute**, depending on its source object. This means that the condition expression must be evaluated before a Token can be generated and then leave the source object to traverse the Flow. The conditions are usually associated with Decision Gateways, but can also be used with activities.



All BPMN 2.0 tasks and gateways can have a **default sequence flow**. This sequence flow is only selected as the outgoing sequence flow for that activity if and only if none of the other sequence flow could be selected. Conditions on a default sequence flow are always ignored.

A default sequence flow is visualized as a regular sequence flow, with a 'slash' marker at the beginning.

