BPMN 2.0 Method & Style

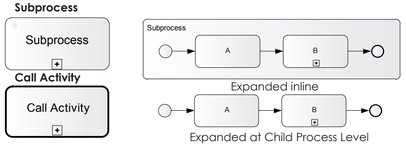
BPMN-Level 1 Palette

Activity: Work performed in a process

Task

A **Task** is an atomic Activity, having no subparts defined in the model. Level 1 palette distinguishes human **User Task** from automated **Service Task**. Generic **None Task** means task type undefined. A **Global Task** (of each type) is a call to a reusable Task definition.

Subprocess



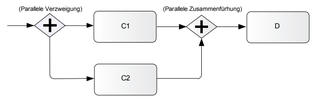
Call Activity

A **Subprocess** is a compound Activity, having subparts defined in the model. May be displayed either **collapsed**, **expanded inline**, **or expanded** at a **Child Process Level** in hierarchical style. Expanded Subprocess inherits Process (Pool) of collapsed Subprocess at Parent Level. It must have a None Start Event.
A **Call Activity** is a call to a reusable Subprocess or Global Task definition.

Gateway: Routing logic

Exclusive (XOR) Gateway

Parallel (AND) Gateway



A **Gateway** controls process flow. Without a Gateway, all Sequence Flows out of an Activity are taken in parallel. **Exclusive (XOR) Gateway** is exclusive decision.
Parallel (AND) Gateway means **split** into Parallel Paths, or **join** Parallel Paths.

Event: A Signal that „something happened"

Start Event

A **Start Event** indicates the start of a Process or Subprocess. A Top-Level Process may have a **Trigger** representing the type of signal that starts the Process: **Message** (external request), **Timer** (scheduled start), or **None** (manual start).
A Subprocess always has a None trigger.

End Event

An **End Event** indicates the end of a path in a Process or Subprocess. Best to use a separate End Event for each distinct end state. End Event may throw a **result** signal: **Message** (to external entity), **Terminate** (abort Subprocess), or **None** (no signal thrown). All Parallel Paths in a Process or Subprocess must reach an End Event to complete normally.

Pools, Lanes and Connectors

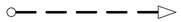
Pool, Lane

A **Pool** represents a participant in a **Collaboration**, an interaction between a Process and the external environment. A Pool can contain a single Process. An empty black-box Pool represents an external participant.
A **Lane** is a subdivision of a Process, typically representing a performer role or organizational unit.

Sequence Flow

Sequence Flow represents orchestration, or flow of control within a Process or Subprocess. When the node at the tail is complete, the node at the arrowhead is started.

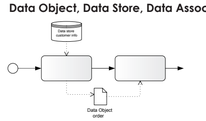
Message Flow



Message Flow represents Collaboration, or interaction between Pools in the form of Messages.

Data

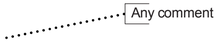
Data Object, Data Store, Data Association



A **Data Object** represents information stored within a process level.
A **Data Store** represents external information accessible to the Process.
The **Data Association** connector represents Data Flow.

Miscellaneous

Text Annotation and Association



A **Text Annotation** is a comment visible in the diagram, linked to a diagram node via an Association connector.

Group

Group is a drawing aid that visually links enclosed elements.

Documentation

Documentation is purely an XML element. It has no graphical representation.

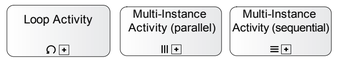
BPMN-Level 2 Palette

Activity: Additional Types and Properties

Task

A **Send Task** sends a Message.
A **Receive Task** receives (waits for) a Message.

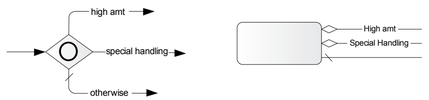
Repeating Activities



A **Loop Activity** evaluates a true/false condition after each iteration; if true, the Activity is performed again.
A **Multi-Instance Activity** is performed, typically in parallel, for each item in a list. Loop and Multi-Instance (MI) Activities may be either Tasks or Subprocesses.

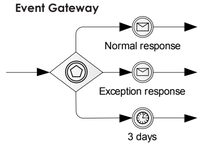
Gateway: Additional Flow Control Pattern

Inclusive (OR) Gateway and Conditional Sequence Flow



An **Inclusive (OR) Gateway** represents independent conditions; all Sequence Flows with a true condition are enabled in parallel.
Conditional Sequence Flow (right) is an alternative representation without a Gateway.
Default Flow (tickmark) means "otherwise," i.e., no other conditions are true.
Use **OR Gateway** to join conditionally parallel paths.

Event Gateway



An **Event Gateway** represents exclusive choice based on the Event that occurs first. Each Gate must contain a catching Intermediate Event, typically Message or Timer.

Event: Additional Event Types

Additional Start Events

Conditional Start signifies triggering by a monitored data condition, such as "low inventory."
Signal Start signifies triggering by a broadcast signal (publish-subscribe integration).

Additional End Events

An **Error End Event** in a Subprocess throws a signal caught by an interrupting Error Boundary Event on the same Subprocess.
An **Escalation End Event** in a Subprocess throws a signal caught by a non-interrupting Escalation Boundary Event on the same Subprocess.
A **Signal End Event** broadcasts a signal catchable by any Signal Event.

Throwing and Catching Intermediate Events

Catching and throwing Intermediate Events have Sequence Flow in and out.
Throwing Intermediate Events (black icon) send a signal and continue.
Catching Intermediate Events (white icon) wait for a signal, then resume.
Message Event sends or receives a Message to/ from another Pool.
Timer Event is a specif ed time delay.
Signal Event broadcasts a signal or subscribes to a broadcast Signal.

Link Event Pair

A **Link Event Pair** stands for a Sequence Flow "go-to," typically used as an off-page connector. It is only allowed where a Sequence Flow would be allowed.

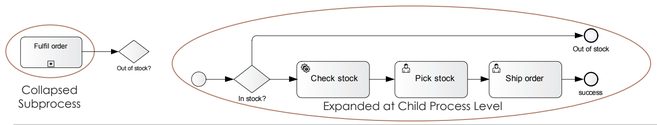
Interrupting and Non-Interrupting Boundary Events

Boundary Events
(attached to an Activity boundary) listen for a signal while the Activity is running. If the signal occurs, **Interrupting Events** (top) abort the Activity and exit on the Exception Flow.
Non-Interrupting Events (bottom) trigger the Exception Flow in parallel with normal Activity completion and exit. **Message Event** is a signal from outside the Process. **Timer Event** is a timeout.
Error and Escalation Boundary Event on Subprocess catch exception signal thrown from the child level of the Subprocess.
Conditional Boundary Event responds to a continuously monitored data condition.

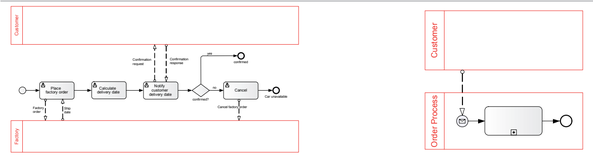
Message

A **Message** represents the content of a communication between two Participants. In BPMN 2.0, a Message is a graphical object.

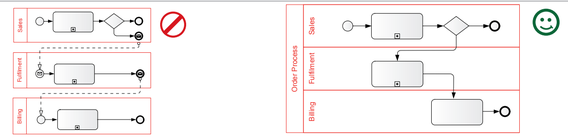
Elements of BPMN Style



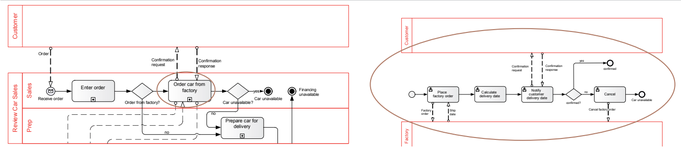
- Make models hierarchical, using Subprocesses to represent Process Levels.
- User labels to make Process logic obvious from the diagram alone.
- Label Activities Verb-Noun
- If possible, label Exclusive Decision Gateways with a yes/no question, and label the outgoing Sequence Flows yes and no.



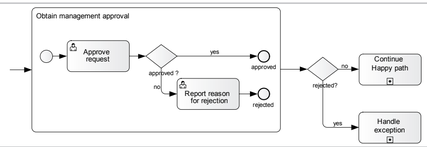
- Use empty (Black-box) Pools to represent external participants.
- Begin customer-facing Processes with a Message Start Event receiving a Message Flow from the Customer Pool.
- Label White-box Pools with the name of a Process; label Black-box Pools with a participant role or business entity.



- Model internal process participants (activity performers) as Lanes within a single Process Pool, not as separate Pools.

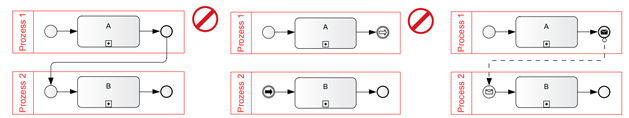


- Show Message Flows between Process and all external Pools in Top-level Diagram, and show Message Flows consistently in Parent- and Child-level-Diagrams.

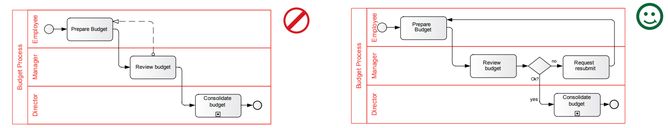


- Indicate success and failed end states of a Process with separate End Events, and label them to indicate the end state.
- If Subprocess is followed by yes/no Gateway, match one End Event of the Subprocess with Gateway Label.

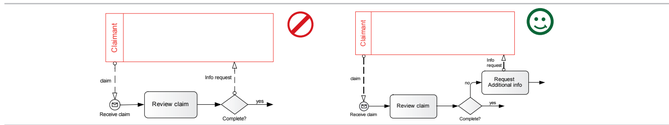
- All Activities, Gateways, and Events must be connected via a continuous chain of Sequence Flows leading from a Start Event to End Event.
- Do not leave Flow Objects "floating" in the Diagram.



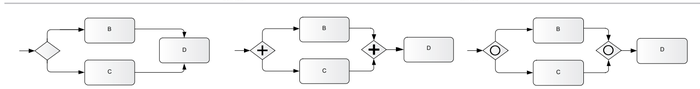
- Sequence Flow (or equivalent link event pair) must not cross a Pool boundary.
- Use Message Flow to link Pools.



- Message Flow cannot connect points in the same pool



- Message Flow cannot connect to a Gateway.



- Do not use gateway to merge exclusive alternative paths into an activity
- Use AND-gateway to join unconditionally parallel paths
- Use OR-gateway to join conditionally parallel paths