What you will learn?

- What are BPMN Events?
- Types of BPMN Events
- Sample process demonstrating how Events work

What are BPMN Events?

BPMN events are elements that represent occurrences during the execution of a process. They are essential in BPMN, enabling processes to start, interrupt, delay, or end based on both internal and external triggers. Events can influence the flow of a process in various ways, such as temporarily halting the movement of a token or completely disrupting its progression.

Types of Events

Start Events

These events trigger the initiation of a process instance.

Message Start Event
Starts a process when a specific message is received.

Timer Start Event
Starts the process based on a scheduled date or time.

Conditional Start Event
Starts the process when a condition (like an expression) evaluates to true.

Signal Start Event
Initiates the process when a signal is triggered (from another process or external source).

Intermediate Events

These events occur during the execution of a process, after the start event and before the end event. They can be used to pause the process, wait for certain conditions, or trigger specific actions.

Intermediate ThrowEvent

Intermediate Throw Event
A BPMN event that executes an action or throws an event while the process is in progress, without terminating the process.

Message Intermediate Catch Event

Message Intermediate Catch Event
When this event is triggered, a message subscription is created. The process stops and waits until the message is received.

Message Intermediate Throw Event

Message Intermediate Throw Event
Sends a message to an external service or process. It creates a job and waits for completion before continuing.

Timer Intermediate Catch Event

Timer Intermediate Catch Event
Pauses the process until a specific time or date is reached, then resumes.

Escalation Intermediate Throw Event

Escalation Intermediate Throw Event
Triggers an escalation without stopping the process. Outgoing sequence flows are taken if present.

Conditional Intermediate Catch Event

Conditional Intermediate Catch Event
Waits until a defined condition becomes true. If not true at first, the process waits until it is.

Link Intermediate Catch Event

Link Intermediate Catch Event
A re-entry point in the process that connects different parts and enables loops or jumps.

Link Intermediate Throw Event

Link Intermediate Throw Event
Triggers a link and signals the process to move to the corresponding Link Catch Event.

Compensation Intermediate Throw Event

Compensation Intermediate Throw Event
Triggers compensation logic to reverse prior actions.

Signal Intermediate Catch Event

Signal Intermediate Catch Event
Waits for a signal to be broadcasted before proceeding.

Signal Intermediate Throw Event

Signal Intermediate Throw Event
Broadcasts a signal to be received by other processes or events.

End Events

End events represent the completion or termination of a process.

Message
End Event

Message End Event
Ends the process and sends a message to another system or process.

Escalation
End Event

Escalation End Event
Triggers an escalation; other threads continue execution.

Error
End Event

Error End Event
Terminates the process due to an error.

Compensation End Event
Triggers compensation handlers to undo completed tasks.

Signal End Event
Sends a signal when the process ends.

Terminate End Event
Instantly ends the entire process instance and all subprocesses.

Boundary Events

Boundary events in BPMN define how a process should respond if a specific event occurs during an activity. For example, a timer boundary event can trigger a reminder if a task takes too long. These events are always intermediate catch events and can be either interrupting which stops the activity or non-interrupting, which allows the activity to continue while launching a parallel path.

Message Boundary Event
These events are linked to a unique message name and can be attached to an activity. When the activity starts, a subscription is created for each event, allowing the process to handle incoming messages while the activity is still in progress.

Timer Boundary Event
This event is based on a predefined time duration or date. When the timer condition is met, the associated activity is terminated.

Escalation Boundary Event
A specialized intermediate catch event that handles escalations within the process, allowing you to manage exceptions or deviations during the activity's execution.

Conditional Boundary Event
These events monitor for specific conditions during the execution of an activity. When the defined condition becomes true, the event is triggered.

Error Boundary Event
This event catches errors that occur within the scope of the activity to which it is attached. If no matching error is defined, the process follows the default behavior of the "none end event."

Signal Boundary Event
This event listens for a specific signal while an activity is active. Signals are globally scoped, meaning they can be captured from any process or instance, not just the local process.

Compensation Boundary Event
This type of event is used to handle compensation actions. It monitors for a specific condition to be met, and when the condition is true, it triggers an alternative path to undo or compensate for previous actions in the process.

Non-Interrupting Boundary Events

Non-interrupting boundary events enable parallel execution paths, allowing the original activity to continue while a new path is triggered. Below are examples of such events:

Message Boundary Event (Non-Interrupting)
In this event, the original activity (token) continues to run, while an additional token is created to follow the sequence flow triggered by the boundary event. This allows both the primary activity and the triggered path to run concurrently.

Timer Boundary Event (Non-Interrupting)
This event allows the activity to proceed while also triggering a separate execution path when the timer condition is met. It creates a parallel flow without interrupting or canceling the original task.

Escalation Boundary Event (Non-Interrupting)
An escalation can be triggered while the main activity remains active. This creates a parallel execution path and is helpful in situations where a problem or deviation needs attention, but the primary process should not be halted.

Conditional Boundary Event (Non-Interrupting)
This event triggers when a defined condition is met, but it does not interrupt the ongoing activity. Instead, it spawns a new parallel execution path, allowing both the original activity and the triggered event to continue simultaneously.

Signal Boundary Event (Non-Interrupting)
This event allows a new execution path to be triggered when a signal is received, without halting the main activity. The original process continues while the signal event's flow runs in parallel.