4. Business Process Diagram Graphical Objects

This section details the graphical representation and the semantics of the behavior of Business Process Diagram graphical elements. Refer to the section entitled “Mapping to BPEL4WS” on page 157 for more information about how these elements map to execution languages.

4.1 Common BPD Object Attributes

The following table displays a set of common attributes for BPMN Flow Objects (specifically Events, Activities, and Gateways):

Attributes	Description
Id: ObjectId	This is a unique Id that identifies the object from other objects within the Diagram.
Name: String	Name is an attribute that is text description of the object.
Assign*: Expression	Zero or more assignment expressions MAY be made for the object. The Assignment SHALL be performed as defined by the AssignTime attribute for activities or when the Token arrives at an Event or Gateway. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.
Pool: Pool	A Pool MUST be identified for the object to identify its location.The attributes of a Pool can be found section entitled “Pool” on page 103.
Lane*: Lane	If the Pool has more than one Lane, then the Id of at least one Lane MUST be added. There MAY be multiple Lanes listed if the Lanes are organized in matrix or overlap in a non-nested manner, The attributes of a Lane can be found section entitled “Lane” on page 216.
Category*: String	This modeler MAY add one or more defined Categories that can be used for purposes such as reporting and analysis.
Documentation?: String	The modeler MAY add text documentation about the object.

Table 7 Common Object Attributes

4.1.1 Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The Id attributes, within the set of Common Object attributes, was changed to be of type ObjectId.
- The Pool attributes, within the set of Common Object attributes, was changed to be of type Pool.
- The Lane attributes, within the set of Common Object attributes, was changed to be of

type Lane.

- The Category attribute was added to the set of Common Object attributes.
- Throughout the specification, attributes that were defined as being of type “(True | False)” were changed to “Boolean.”

4.2 Events

An Event is something that “happens” during the course of a business process. These Events affect the flow of the Process and usually have a cause or an impact. The term “event” is general enough to cover many things in a business process. The start of an activity, the end of an activity, the change of state of a document, a message that arrives, etc., all could be considered events. However, BPMN has restricted the use of events to include only those types of events that will affect the sequence or timing of activities of a process. BPMN further categorizes Events into three main types: Start, Intermediate, and End.

Start and Intermediate Events have “Triggers” that define the cause for the event. There are multiple ways that these events can be triggered (refer to the section entitled “Start Event Triggers” on page 52 and “Intermediate Event Triggers” on page 60). End Events may define a “Result” that is a consequence of a Sequence Flow ending. There are multiple types of Results that can be defined (refer to the section entitled “End Event Results” on page 57).

All Events share the same shape footprint, a small circle. Different line styles, as shown below, distinguish the three types of flow Events. All Events also have an open center so that BPMN-defined and modeler-defined icons can be included within the shape to help identify the Trigger or Result of the Event.

4.2.1 Common Event Attributes

The following are attributes common to the three types of Events, and which extends the set of common object attributes (see Table 7):

Attributes	Description
EventType: (Start | End | Intermediate)	The EventType MUST be of of type Start, End, or Intermediate.

Table 8 Common Event Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- A set of common set of Event attributes was added.
- The EventType attribute was added to the set of common Event attributes.

4.2.2 Start

As the name implies, the Start Event indicates where a particular Process will start. In terms of Sequence Flow, the Start Event starts the flow of the Process, and thus, will not have any incoming Sequence Flows—no Sequence Flows can connect to a Start Event.

The Start Event shares the same basic shape of the Intermediate Event and End Event, a circle with an open center so that markers can be placed within the circle to indicate variations of the Event.

- A Start Event is a circle that MUST be drawn with a single thin line (see Figure 2).
  - The use of text, color, size, and lines for a Start Event MUST follow the rules defined in section 3.3 on page 40 with the exception that:
    - The thickness of the line MUST remain thin so that the Start Event may be distinguished from the Intermediate and End Events.

Figure 2 A Start Event

Throughout this document, we will discuss how Sequence Flow proceeds within a Process. To facilitate this discussion, we will employ the concept of a "**Token**" that will traverse the Sequence Flows and pass through the flow objects in the Process. The behavior of the Process can be described by tracking the path(s) of the Token through the Process. A Token will have a unique identity, called a TokenId set, that can be used to distinguish multiple Tokens that may exist because of concurrent Process instances or the dividing of the Token for parallel processing within a single Process instance. The parallel dividing of a Token creates a lower level of the TokenId set. The set of all levels of TokenId will identify a Token. The TokenId set for a Token will be in the following format: "TokenId.TokenId. … TokenId," each level being separated by a dot.

A Start Event generates a Token that must eventually be consumed at an End Event (which may be implicit if not graphically displayed). The path of Tokens MUST be explicitly traced through the network of Sequence Flow with a Process. There CANNOT be any implicit flow during the course of normal Sequence Flow. Tokens can also be consumed through exception handling Intermediate Events, which act like a forced end to a Process level. Note: A Token does not traverse the Message Flows since it is a Message that is passed down those Flows (as the name implies).

Semantics of the Start Event include:

- A Start Event is OPTIONAL: a Process level—a top-level Process or an expanded Sub-Process—MAY (is not required to) have a Start Event:

Note: A BPD may have more than one Process level (i.e., it can include Expanded Sub-Processes). The use of Start and End Events is independent for each level of the Diagram.

  - If a Process is complex and/or the starting conditions are not obvious, then it is RECOMMENDED that a Start Event be used.
  - If there is an End Event, then there MUST be at least one Start Event.
  - If the Start Event is used, then there SHALL NOT be other flow elements that do not have incoming Sequence Flow—all other flow objects MUST be a target of at least one Sequence Flow.
    - An exception to this are activities that are defined as being Compensation activities (have the Compensation Marker). Compensation activities SHALL NOT have any incoming Sequence Flow, even if there is a Start Event in the Process level. Refer to the section entitled "Compensation Association" on page 152 for more informations on Compensation activities.

- An exception to this is the Intermediate Event, which MAY be without an incoming Sequence Flow (when attached to an activity boundary).

- If the Start Event is *not* used, then all flow objects that do not have an incoming Sequence Flow (i.e., are not a target of a Sequence Flow) SHALL be instantiated when the Process is instantiated. There is an assumption that there is only one implicit Start Event, meaning that all the starting flow objects will start at the same time.
  - An exception to this are activities that are defined as being Compensation activities (have the Compensation Marker). Compensation Activities are not considered a part of the normal flow and SHALL NOT be instantiated when the Process is instantiated.

- There MAY be multiple Start Events for a given Process level.
  - Each Start Event is an independent event. That is, a Process Instance SHALL be generated when the Start Event is triggered.

Note: The behavior of Process may be harder to understand if there are multiple Start Events. It is RECOMMENDED that this feature be used sparingly and that the modeler be aware that other readers of the Diagram may have difficulty understanding the intent of the Diagram.

When the trigger for a Start Event occurs, Tokens will be generated for each outgoing Sequence Flow from that event. The TokenId set for each of the Tokens will be established such that it can be identified that the Tokens are all from the same parallel Fork (AND-Split) and the number of Tokens in the group. These Tokens will begin their flow and not wait for any other Start Event to be triggered.

If there is a dependency for more than one Event to happen before a Process can start (e.g., two messages are required to start), then the Start Events must flow to the same activity within that Process. The attributes of the activity would specify when the activity could begin. If the attributes specify that the activity must wait for all inputs, then all Start Events will have to be triggered before the Process begins (refer to the section entitled “Attributes” on page 71 (for sub-processes) and “Attributes” on page 77 (for Tasks) for more information about activity attributes). In addition, a correlation mechanism will be required so that different triggered Start Events will apply to the same process instance. Correlation will likely be handled through Event attributes, but this an open issue will be addressed in a later version of the specification. Refer to the section entitled “Open Issues” on page 231 for a complete list of the issues open for BPMN.

Start Event Triggers

There are many ways that can business process can be started (instantiated). The Trigger for a Start Event is designed to show the general mechanism that will instantiate that particular Process. There are six types of Start Events in BPMN: None, Message, Timer, Rule, Link, and Multiple.

Table 9 displays the types of Triggers and the graphical marker that will be used for each:

Trigger	Description	Marker
None	The modeler does not display the type of Event. It is also used for a Sub-Process that starts when the flow is triggered by its Parent Process.	
Message	A message arrives from a participant and triggers the start of the Process.	
Timer	A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the start of the Process.	
Rule	This type of event is triggered when the conditions for a rule such as “S&P 500 changes by more than 10% since opening,” or “Temperature above 300C” become true.	
Link	A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process.	
Multiple	This means that there are multiple ways of triggering the Process. Only one of them will be required to start the Process. The attributes of the Start Event will define which of the other types of Triggers apply.	

Table 9 Start Event Types

Attributes

The following are attributes of a Start Event, which extends the set of common Event attributes (see Table 8):

Attributes	Description
Trigger (None | Message | Timer | Rule | Link | Multiple): None	Trigger is an attribute (default None) that defines the type of trigger expected for that Start. The next six rows define the attributes that are required for each of the Trigger types. The Trigger list MAY be extended to include new types. These new Triggers MAY have a new modeler- or tool-defined Marker to fit within the boundaries of the Event.
(Message Trigger only) **Message**: Message	If the Trigger is a Message, then the a Message MUST be supplied. The attributes of a Message can be found section entitled “Message” on page 260.
(Message Trigger only) **Message**: Message	If the Trigger is a Message, then the Implementation of the receipt of the Message MUST be supplied. By default, the Implementation is a Web Service, but other Implementations can be added to extend the list.
(Timer Trigger only) TimeDate?: Date	If the Trigger is a Timer, then a TimeDate MAY be entered. If a TimeDate is not entered, then a TimeCycle MUST be entered (see the attribute below).
(Timer Trigger only) TimeCycle?: String	If the Trigger is a Timer, then a TimeCycle MAY be entered. If a TimeCycle is not entered, then a TimeDate MUST be entered (see the attribute above).
(Rule Trigger only) RuleName: Rule	If the Trigger is a Rule, then a Rule MUST be entered. The attributes of a Rule can be found section entitled “Rule” on page 261.

Attributes	Description
(Linker only) **LinkId**: String	If the Trigger is a Link, then the LinkId MUST be entered.
(Multiple Trigger only): Trigger 2+: Trigger	If the Trigger is a Multiple, then a list of two or more Triggers MUST be provided. Each Trigger MUST have the appropriate data (as defined above). The Trigger MAY NOT be of type None or Multiple.

Table 10 Start Event Attributes

Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how they may be source or targets of Sequence Flows.

- A Start Event SHALL NOT be a target for a Sequence Flow; it MUST NOT have incoming Sequence Flows.
- A Start Event MUST be a source for a Sequence Flow.
- Multiple Sequence Flows MAY originate from a Start Event. For each Sequence Flow that has the Start Event as a source, a new parallel path SHALL be generated.
  - The Condition attribute for all outgoing Sequence Flow MUST be set to None.
  - When a Start Event is not used, then all flow objects that do not have an incoming Sequence Flow SHALL be the start of a separate parallel path.

Each path will have a separate unique Token that will traverse the Sequence Flow.

Message Flow Connections

Refer to the section entitled “Message Flow Rules” on page 42 for the entire set of objects and how they may be source or targets of Sequence Flows.

Note: All Message Flows must connect two separate Pools. They can connect to the Pool boundary or to flow objects within the Pool boundary. They cannot connect two objects within the same Pool.

- A Start Event MAY be the target for Message Flows; it can have 0 (zero) or more incoming Message Flows. Each Message Flow arriving at a Start Event represents an instantiation mechanism (a Trigger) for the process. Only one of the Triggers is required to start a new Process.
  - The Trigger attribute of the Start Event MUST be set to “Message” or “Multiple” if there are any incoming Message Flows.
    - The Trigger attribute of the Start Event MUST be set to “Multiple” if there are more than one incoming Message Flows.
- A Start Event SHALL NOT be a source for a Message Flow; it MUST NOT have outgoing Message Flows.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Event was removed.
- The attribute table was reorganized to make it more clear that some attributes applied only if the Trigger attribute was set to specific values.
- The Message attribute, within the set of the Start Event attributes, was changed to be of type Message.
- The RuleExpression attribute, within the set of the Start Event attributes, was renamed to RuleName and was changed to be of type Rule.
- The LinkName attribute, within the set of the Start Event attributes, was rename to LinkId and was changed to be of type String.
- The attribute for a Timer Event, within the set of the Start Event attributes, was divided into two separate attributes. One is TimeDate and the other is TimeCycle.
- The multiplicity of the list of Triggers for the Multiple Trigger was modified so that it is now 2 or more.
- Within the Message Flow Connections section, it was clarified that if there are multiple incoming Message Flow, then the Trigger must be Multiple.

4.2.3 End

As the name implies, the End Event indicates where a process will end. In terms of Sequence Flow, the End Event ends the flow of the Process, and thus, will not have any outgoing Sequence Flows—no Sequence Flows can connect from an End Event.

The End Event shares the same basic shape of the Start Event and Intermediate Event, a circle with an open center so that markers can be placed within the circle to indicate variations of the Event.

- An End Event is a circle that MUST be drawn with a single thick black line (see Figure 3).
  - The use of text, color, size, and lines for an End Event MUST follow the rules defined in section 3.3 on page 40 with the exception that:
    - The thickness of the line MUST remain thick so that the End Event may be distinguished from the Intermediate and Start Events.

Figure 3 End Event

To continue discussing how flow proceeds throughout the process, an End Event consumes a Token that had been generated from a Start Event within the same level of Process. If parallel Sequence Flows target the End Event, then the Tokens will be consumed as they arrive. All the Tokens that were generated within the Process must be consumed by an End Event before the Process has been completed.

Semantics of the End Event include:

- There MAY be multiple End Events within a single level of a process.
- This shape is OPTIONAL: a given Process level—a top-level Process or an expanded Sub-Process—MAY (is not required to) have this shape:
  - If there is a Start Event, then there MUST be at least one End Event.
  - If an End Event is used, then there SHALL NOT be other flow elements that do not have any outgoing Sequence Flows—all other flow objects MUST be a source of at least one Sequence Flow.
    - An exception to this are activities that are defined as being Compensation activities (have the Compensation Marker). Compensation Activities SHALL NOT have any outgoing Sequence Flow, even if there is an End Event in the Process level. Refer to the section entitled “Compensation Association” on page 152 for more information on Compensation activities.
  - If the End Event is not used, then all flow objects that do not have any outgoing Sequence Flows (i.e., are not a source of a Sequence Flow) mark the end of the Process. However, the process SHALL NOT end until all parallel paths have completed.
    - An exception to this are activities that are defined as being Compensation activities (have the Compensation Marker). Compensation Activities are not considered a part of the normal flow and SHALL NOT mark the end of the Process.

Note: A BPD may have more than one Process level (i.e., it can include Expanded Sub-Processes). The use of Start and End Events is independent for each level of the Diagram.

For Processes without an End Event, a Token entering a path-ending flow object will be consumed when the processing performed by those objects are completed (i.e., when the path has completed), as if the Token had then gone on to reach an End Event. When all Tokens for a given instance of the Process are consumed, then the Process will reach a state of being completed.

End Event Results

A BPMN modeler can define the consequence of reaching an End Event. This will be referred to as the End Event Result.

Table 11 displays the types of Results and the graphical marker that will be used for each:

Result	Description	Marker
None	The modeler does not display the type of Event. It is also used for a Sub-Process that end and the flow goes back to its Parent Process.	
Message	This type of End indicates that a message is sent to a participant at the conclusion of the Process.	
Exception	This type of End indicates that a named Error should be generated. This Error will be caught by an Intermediate Event within the Event Context.	
Cancel	This type of End is used within a Transaction Sub-Process. It will indicate that the Transaction should be cancelled and will trigger a Cancel Intermediate Event attached to the Sub-Process boundary. In addition, it will indicate that a Transaction Protocol Cancel message should be sent to any Entities involved in the Transaction.	
Compensation	This type of End will indicate that a Compensation is necessary. This Compensation identifier will be used by an Intermediate Event when the Process is rolling back.	
Link	A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process. A Token arriving at Link End Event will immediately jump to it corresponding target Start or Intermediate Event.	
Terminate	This type of End indicates that there is a fatal error and that all activities in the Process should be immediately ended. The Process is ended without compensation or event handling. Note that the marker for this Event is an Open Issue.	
Multiple	This means that there are multiple consequences of ending the Process. All of them will occur (e.g., there might be multiple messages sent). The attributes of the End Event will define which of the other types of Results apply.	

Table 11 End Event Types

Attributes

The following are attributes of a End Event, which extends the set of common Event attributes (see Table 8):

Attributes	Description
Result: (None | Message | Exception | Cancel | Compensation | Link | Terminate | Multiple): None	Result is an attribute (default None) that defines the type of result expected for that End. The Cancel Result MAY NOT be used unless the Event is used within a Process that is a Transaction. The Result list MAY be extended to include new types. These new Results MAY have a new modeler- or tool-defined Marker to fit within the boundaries of the Event.
(Message Result only) **Message**: Message	If the Result is a Message, then the Message MUST be supplied. The attributes of a Message can be found section entitled “Message” on page 260.
(Exception Result only) ExceptionCode: String	If the Result is an Exception, then the ExceptionCode MAY be supplied. If no ExceptionCode is provided, then
(Compensation Result only) Activity: ObjectId	If the Result is a Compensation, then the ObjectId of the Activity that needs to be compensated MUST be supplied.
(Link Result only) LinkId: String	If the Result is a Link, then the LinkId MUST be entered.
(Multiple Result only) Result 2+: Result	If the Result is a Multiple, then a list of two or more Results MUST be entered. Each Result on the list MUST have the appropriate data as specified for the above attributes. The Result MAY NOT be of type None, Terminate, or Multiple.

Table 12 End Event Attributes

Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how they may be source or targets of Sequence Flows.

- An End Event MUST be a target for a Sequence Flow.
- An End Event MAY have multiple incoming Sequence Flows.

The Flows MAY come from either alternative or parallel paths. For modeling convenience, each path MAY connect to a separate End Event object. The End Event is used as a Sink for all Tokens that arrive at the Event. All Tokens that are generated at the Start Event for that Process must eventually arrive at an End Event. The Process will be in a *running* state until all Tokens are consumed.

- An End Event SHALL NOT be a source for a Sequence Flow; that is, there SHALL NOT be outgoing Sequence Flows.

Message Flow Connections

Refer to the section entitled “Message Flow Rules” on page 42 for the entire set of objects and how they may be source or targets of Sequence Flows.

Note: All Message Flows must connect two separate Pools. They can connect to the Pool boundary or to flow objects within the Pool boundary. They cannot connect two objects within the same Pool.

- An End Event MUST NOT be the target for Message Flows; it can have no incoming Message Flows.
- An End Event MAY be a source for a Message Flow; it can have one or more outgoing Message Flow.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Event was removed.
- The definition of the Link End Event was updated to include a description of Token flow.
- The attribute table was reorganized to make it more clear that some attributes applied only if the Result attribute was set to specific values.
- For the Result attribute, within the set of End Event attributes, the type Rule was removed from the list of types of Results. It should not have been listed there initially.
- The Message attribute, within the set of the End Event attributes, was changed to be of type Message.
- For a Result of type Compensation. the attributes were rather vaguely defined and have been consolidated to be a single attribute named Activity of type ObjectId.
- The LinkName attribute, within the set of the End Event attributes, renamed to LinkId was changed to be of type String.
- The multiplicity of the list of Results for the Multiple Result was modified so that it is now 2 or more.

4.2.4 Intermediate

Intermediate Events occur between a Start Event and an End Event. This is an event that occurs after a Process has been started. It will affect the flow of the process, but will not start or (directly) terminate the process. Intermediate Events can be used to:

- Show where messages or delays are expected within the Process,
- Disrupt the normal flow through exception handling, or
- Show the extra work required for compensation.

The Intermediate Event shares the same basic shape of the Start Event and End Event, a circle with an open center so that markers can be placed within the circle to indicate variations of the Event.

- An Intermediate Event is a circle that MUST drawn with a double thin black line. (see Figure 3).
  - The use of text, color, size, and lines for an Intermediate Event MUST follow the rules defined in section 3.3 on page 40 with the exception that:
    - The thickness of the line MUST remain double so that the Intermediate Event may be distinguished from the Start and End Events.

Figure 4 Intermediate Event

One use of Intermediate Events is to represent exception or compensation handling. This will be shown by placing the Intermediate Event on the boundary of a Task or Sub-Process (either collapsed or expanded). Figure 5 displays an example of an Intermediate Event attached to a Task. The Intermediate Event can be attached to any location of the activity boundary and the outgoing Sequence Flow can flow in any direction. However, in the interest of clarity of the Diagram, we recommend that the modeler choose a consistent location on the boundary. For example, if the Diagram orientation is horizontal, then the Intermediate Events can be attached to the bottom of the activity and the Sequence Flow directed down and then to the right. If the Diagram orientation is vertical, then the Intermediate Events can be attached to the left or right side of the activity and the Sequence Flow directed to the left or right and then down.

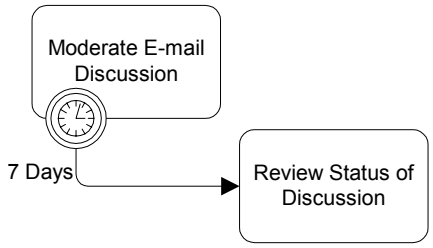

Figure 5 Task with an Intermediate Event attached to its boundary

Intermediate Event Triggers

There are eight types of Intermediate Events in BPMN: Message, Timer, Exception, Compensation, Cancel, Rule, Link, and Multiple. These Event types indicate the different ways that a Process may be interrupted or delayed after it has started. Each type of Intermediate Event will have a different icon placed in the center of the Intermediate Event shape to distinguish one from another.

Table 13 displays the types of Triggers and the graphical marker that will be used for each:

Trigger	Description	Marker
None	This is valid for only Intermediate Events that are in the main flow of the Process. The modeler does not display the type of Event. It is used for modeling methodologies that use Events to indicate some change of state in the Process.	
Message	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.	
Timer	A specific time-date or a specific cycle (e.g., every Monday 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.	
Exception	This is used for exception handling--both to set (throw) and to react to (catch) exceptions. It sets an exception if the Event is part of a normal flow. It reacts to a named exception, or to any exception if a name is not specified, when attached to the boundary of an activity.	
Cancel	This type of Intermediate Event is used within a Transaction Sub-Process. This type of Event MUST be attached to the boundary of a Sub-Process. It SHALL be triggered if a Cancel End Event is reached within the Transaction Sub-Process. It also SHALL be triggered if a Transaction Protocol “Cancel” message has been received while the Transaction is being performed.	
Compensation	This is used for compensation handling--both setting and performing compensation. It call for compensation if the Event is part of a normal flow. It reacts to a named compensation call when attached to the boundary of an activity.	
Rule	This is only used for exception handling. This type of event is triggered when a named Rule becomes true. A Rule is an expression that evaluates some Process data.	
Link	A Link is a mechanism for connecting an End Event (Result) of one Process to an Intermediate Event (Trigger) in another Process. Paired Intermediate Events can also be used as “Go To” objects within a Process.	
Multiple	This means that there are multiple ways of triggering the Event. Only one of them will be required. The attributes of the Intermediate Event will define which of the other types of Triggers apply.	

Table 13 Intermediate Event Types

Attributes

The following are attributes of an Intermediate Event, which extends the set of common Event attributes (see Table 8):

Attributes	Description
Trigger: (None | Message | Timer | Exception | Cancel | Compensation | Rule | Multiple): Message	Trigger is an attribute (default Message) that defines the type of trigger expected for that Intermediate Event. The None and Link Trigger MAY NOT be used when the Event is attached to the boundary of an Activity. The Multiple, Rule, and Cancel Triggers MAY NOT be used when the Event is part of the normal flow of the Process. The Cancel Trigger MAY NOT be used when the Event is attached to the boundary of an Activity that is not a Transaction or if the Event is not contained within a Process that is a Transaction. The Trigger list MAY be extended to include new types. These new Triggers MAY have a new modeler- or tool-defined Marker to fit within the boundaries of the Event.
Target*: ObjectId	A Target MAY be included for the Intermediate Event. The Target MUST be an activity (Sub-Process or Task). This means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity.
(Message Trigger only) **Message**: Message	If the Trigger is a Message, then the Message MUST be supplied. The attributes of a Message can be found section entitled “Message” on page 260.
(Timer Trigger only) Timedate?: Date	If the Trigger is a Timer, then a TimeDate MAY be entered. If a TimeDate is not entered, then a TimeCycle MUST be entered (see the attribute below).
(Timer **Trigger only**) TimeCycle?: String	If the Trigger is a Timer, then a TimeCycle MAY be entered. If a TimeCycle is not entered, then a TimeDate MUST be entered (see the attribute above).
(Exception Trigger only) ExceptionCode: String	*For an Intermediate Event within normal flow:* If the Trigger is an Exception, then the error code MUST be entered. This “throws” the exception. *For an Intermediate Event attached to the boundary of an Activity:* If the Trigger is an Exception, then the error code MAY be entered. This “catches” the exception. If there is no error code, then any Exception SHALL trigger the Event. If there is an error code, then only an Error that matches the error code SHALL trigger the Event.
(Compensation Trigger only) Activity: ObjectId	*For an Intermediate Event within normal flow:* If the Trigger is a Compensation, then the ObjectId of the Activity that needs to be compensated MUST be supplied. This “throws” the compensation. *For an Intermediate Event attached to the boundary of an Activity:* The “catches” the compensation. No further information is required. The ObjectId of the activity the Event is attached to will provide the Id necessary to match the compensation event with the event that “threw” the compensation.

Attributes	Description
(Rule Trigger only) RuleName: Rule	If the Trigger is a Rule, then a Rule MUST be entered. The attributes of a Rule can be found section entitled “Rule” on page 261.
(Link Trigger only) LinkId: String	If the Trigger is a Link, then the LinkId MUST be supplied.
(Multiple Trigger only) Trigger 2+: Trigger	If the Trigger is a Multiple, then each Trigger on the list MUST have the appropriate data as specified for the above attributes. The Trigger MAY NOT be of type None or Multiple.

Table 14 Intermediate Event Attributes

Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

- Intermediate Events MAY be attached directly to the boundary of an Activity.
  - To be attached to the boundary of an Activity, an Intermediate Event MUST be one of the following Triggers: Message, Timer, Exception, Cancel, Compensation, Rule, and Multiple.
    - An Intermediate Event with a Cancel Trigger MAY be attached to an Activity boundary only if the Transaction attribute of the Activity is set to TRUE.
- If the Intermediate Event is attached to the boundary of an activity, then it MAY NOT be a target for a Sequence Flow; it cannot have an incoming Flow.
- If the Intermediate Event is not attached to the boundary of an activity; that is, it is within normal flow, then it MAY be a target for a Sequence Flow. It MAY have one (and only one) incoming Flow.
  - Intermediate Event of the following types MAY be a target of a Sequence Flow: None, Message, Timer, Exception, Link, and Compensation.
    - An Intermediate Event with a Link Trigger MAY NOT be both a target and a source of a Sequence Flow unless it is part of an Event-Based Exclusive Gateway.
- An Intermediate Event MUST be a source for a Sequence Flow; it can have one (and only one) outgoing Sequence Flow.
  - An exception to this: an Intermediate Event with a Compensation Trigger MUST NOT have an outgoing Sequence Flow (it MAY have an outgoing Association).
  - An exception to this: an Source Link Intermediate Event (as defined below).

To define the use of a Link Intermediate Event as an “Off-Page Connector” or a “Go To” object:

- A Link Intermediate Event MAY be the target (Target Link) or a source (Source Link) of a Sequence Flow, but MUST NOT be both a target and a source.
  - If there is a Source Link, there MUST be a matching Target Link (they have the same LinkId). Note: A Source Link (Intermediate Event) should not be used for linking with another Process within the same Pool; an End Event should be used for this purpose.

- There MAY be multiple Source Links for a single Target Link (??--This would reduce clutter, but could not happen if the Sequence Flow were connected directly--??).

- A Target Link MAY be used without a corresponding Source Link. This indicates that the Source Link (an End Event) exists in another Process within the same Pool.

Message Flow Connections

Refer to the section entitled “Message Flow Rules” on page 42 for the entire set of objects and how the may be source or targets of Sequence Flows.

Note: All Message Flows must connect two separate Pools. They can connect to the Pool boundary or to flow objects within the Pool boundary. They cannot connect two objects within the same Pool.

- An Intermediate Event of type Message MAY be the target for Message Flows; it can have one incoming Message Flows.
- An Intermediate Event MAY NOT be a source for a Message Flow; it can have no outgoing Message Flows.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Event was removed.
- The definition of Link Intermediate Events was expanded to describe behavior as “Go To” objects within a Process.
- The attribute table was reorganized to make it more clear that some attributes applied only if the Trigger attribute was set to specific values.
- The Target attribute was added to the set of Intermediate Event attributes.
- The Message attribute, within the set of the Intermediate Event attributes, was changed to be of type Message.
- The attribute for a Timer Event, within the set of the Intermediate Event attributes, was divided into two separate attributes. One is TimeDate and the other is TimeCycle.
- The definition of Exception setting of the Trigger attribute of the set of Intermediate attributes was updated. The update provided a separation for the uses of the Event either within normal flow or attached to the boundary of an activity. Also, the attribute ErrorCode was renamed ExceptionCode and its type was changed to String.
- The definition of Compensation setting of the Trigger attribute of the set of Intermediate attributes was updated. The update provided a separation for the uses of the Event either within normal flow or attached to the boundary of an activity. Also, the term ActivityId was changed to ObjectId.
- The RuleExpression attribute, within the set of the Intermediate Event attributes, was renamed to RuleName and was changed to be of type Rule.
- The LinkName attribute, within the set of the Intermediate Event attributes, was renamed

to LinkId and was changed to be of type String.

- The multiplicity of the list of Triggers for the Multiple Trigger was modified so that it is now 2 or more.

4.3 Activities

An activity is work that is performed within a business process. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a Business Process Diagram are: Process, Sub-Process, and Task. However, a Process is not a specific graphical object. Instead, it is a set of graphical objects. The following sections will focus on the graphical objects Sub-Process and Task. More information about Processes can be found in the section entitled “Business Process” on page 43.

4.3.1 Common Activity Attributes

The following are attributes common to both a Sub-Process and a Task, and which extends the set of common object attributes (see Table 7) -- Note that Table 16 and Table 17 contain additional attributes that must be included within this set if extended by any other attribute table:

Attributes	Description
ActivityType: (Task | Sub-Process)	The ActivityType MUST be of of type Task or Sub-Process.
Status: (None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed) : None	The Status of an activity is determined when the activity is being executed by a process engine. The Status of an activity can be used within Assignment Expressions.
Property*	Modeler-defined Properties MAY be added to an activity. These Properties are “local” to the activity. These Properties are only for use within the processing of the activity. The fully delineated name of these properties are “<process name>.<activity name>.<property name>” (e.g., “Add Customer.Review Credit.Status”). Further details about the definition of a Property can be found in the section entitled “Property” on page 259.
InputSet*: Input	The InputSet attribute defines the data requirements for input to the activity. Zero or more InputSets MAY be defined. Each Input set is sufficient to allow the activity to be performed (if it has first been instantiated by the appropriate signal arriving from an incoming Sequence Flow).
(for InputSet only) Input+: Artifact	An Input MUST be defined for each InputSet. An Input is one or more Artifacts, usually Document Objects. Note that the Artifacts MAY also be displayed on the diagram and MAY be connected to the activity through an Association--however, it is not required for them to be displayed.
OutputSet*: Output	The OutputSet attribute defines the data requirements for output from the activity. Zero or more OutputSets MAY be defined. At the completion of the activity, only one of the OutputSets may be produced--It is up to the implementation of the activity to determine which set will be produced. However, the IORule attribute MAY indicate a relationship between an OutputSet and an InputSet that started the activity.

Attributes	Description
(for OutputSet only) Output+: Artifact	An Output MUST be defined for each OutputSet. An Output is one or more Artifacts, usually Document Objects. Note that the Artifacts MAY also be displayed on the diagram and MAY be connected to the activity through an Association--however, it is not required for them to be displayed.
IORule*: Expression	The IORule attribute is an expression that defines the relationship between one InputSet and one OutputSet. That is, if the activity is instantiated with a specified InputSet, then the output of the activity MUST produce the specified OutputSet. Zero or more IORules may be entered.
Start Quantity: Integer: 1	The default value is 1. The value MAY NOT be less than 1. This attribute defines the number of Tokens that must arrive from a single Sequence Flow before the activity can begin.
LoopType: (None | Standard | MultiInstance): None	LoopType is an attribute and is by default None, but MAY be set to Standard or MultiInstance. If so, the Loop marker SHALL be placed at the bottom center of the activity shape (see Figure 9 and Figure 12). A Task of type Receive that has its Instantiate attribute set to True MAY NOT have a Standard or MultiInstance LoopType.
AssignTime*: (Start | End): Start	Each Assignment MUST have a separate AssignTime setting. A value of Start means that the assignment SHALL occur at the start of the activity. This can be used to assign the higher-level (global) Properties of the Process to the (local) Properties of the activity as an input to the activity. A value of End means that the assignment SHALL occur at the end of the activity. This can be used to assign the (local) Properties of the activity to the higher-level (global) Properties of the Process as an output to the activity.

Table 15 Common Activity Attributes

Standard Loop Attributes

A Standard Loop activity will have a boolean expression that is evaluated after each cycle of the loop. If the expression is still True, then the loop will continue. There are two variations of the loop, which reflect the programming constructs of while and until. That is, a while loop will evaluate the expression before the activity is performed, which means that the activity may not actually be performed. The until loop will evaluate the expression after the activity has been performed, which means that the activity will be performed at least once.

The following are additional attributes of a Standard Loop Activity (where the LoopType attribute is set to “Standard”), which extends the set of common activity attributes (see Table 14):

Attributes	Description
LoopCondition: Expression	Standard Loops MUST have a boolean Expression to be evaluated, plus the timing when the expression SHALL be evaluated. The attributes of an Expression can be found section entitled “Expression” on page 261.
LoopCounter: Integer	**The LoopCounter attribute is used at runtime to count the number of loops and is automatically updated by the process engine.** The LoopCounter attribute MUST be incremented at the start of a loop. The modeler may use the attribute in the LoopCondition Expression.

Attributes	Description
LoopMaximum?: Integer	The Maximum an optional attribute that provides is a simple way to add a cap to the number of loops. This SHALL be added to the Expression defined in the LoopCondition.
TestTime: (Before | After): After	The expressions that are evaluated Before the activity begins are equivalent to a programming while function. The expression that are evaluated After the activity finishes are equivalent to a programming until function.

Table 16 Standard Loop Activity Attributes

Multi-Instance Loop Attributes

Multi-Instance loops reflect the programming construct foreach. The loop expression for a Multi-Instance loop is a numeric expression evaluated only once before the activity is performed. The result of the expression evaluation will be an integer that will specify the number of times that the activity will be repeated.

There are also two variations of the Multi-Instance loop where the instances are either performed sequentially or in parallel.

The following are additional attributes of a Multi-Instance Loop Activity (where the LoopType attribute is set to “MultiInstance”), which extends the set of common activity attributes (see Table 14):

Attributes	Description
MI_Condition: Expression	MultiInstance Loops MUST have a numeric Expression to be evaluated--the Expression MUST resolve to an integer. The attributes of an Expression can be found section entitled “Expression” on page 261.
LoopCounter: Integer	The LoopCounter attribute is only applied for Sequential MultiInstance Loops and for processes that are being executed by a process engine. The attribute is updated at runtime by a process engine to count the number of loops as they occur. The LoopCounter attribute MUST be incremented at the start of a loop. Unlike a Standard loop, the modeler does not use this attribute in the MI_Condition Expression, but it can be used for tracking the status of a loop.
MI_Ordering: (Sequential | Parallel): Sequential	This applies to only MultiInstance Loops. The MI_Ordering attribute defines whether the loop instances will be performed sequentially or in parallel. Sequential **MI_Ordering** is a more traditional loop. Parallel **MI_Ordering** is equivalent to multi-instance specifications that other notations, such as UML Activity Diagrams use. If set to Parallel, the Parallel marker SHALL replace the Loop Marker at the bottom center of the activity shape (see Figure 9 and Figure 12).

Attributes	Description
(Parallel MI_Ordering only) MI_FlowCondition: (None | One | All | Complex): One	This attribute is equivalent to using a Gateway to control the flow past a set of parallel paths. An MI_FlowCondition of “None” is the same as uncontrolled flow (no Gateway) and means that all activity instances SHALL generate a token that will continue when that instance is completed.. An MI_FlowCondition of “One” is the same as an Exclusive Gateway and means that the Token SHALL continue past the activity after only one of the activity instances has completed. The activity will continue its other instances, but additional Tokens SHALL NOT be passed from the activity. An MI_FlowCondition of “All” is the same as a Parallel Gateway and means that the Token SHALL continue past the activity after all of the activity instances have completed. An MI_FlowCondition of “Complex” is the same as a Complex Gateway. The **Complex**MI_FlowCondition attribute will determine the Token flow.
(Complex MI_FlowCondition only) **Complex**MI_FlowCondition?: Expression	If the MI_FlowCondition attribute is set to “Complex,” then an Expression Must be entered. This Expression that MAY reference Process data. The expression SHALL determine when and how many Tokens will continue past the activity. The attributes of an Expression can be found section entitled “Expression” on page 261. The

Table 17 Multi-Instance Loop Activity Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- This set of attributes was removed from each of the set of Sub-Process attributes (Table 18) and Task attributes (Table 21) and moved to the above table (Table 14).
  - Within the set of common activity attributes, the attributes for standard activity loops and for multi-instance activity loops were each placed into separate tables (Table 16 and Table 17, respectively).
- The ActivityType attribute was added to the set of common activity attributes.
- The Name and Type attributes were removed from the set of common activity attributes. These attributes can be found in the definition of a Property, which can be found in the section entitled “Property” on page 259.
- The InputSet attribute was added to the set of common activity attributes.
- The Input attribute was included in the set of common activity attributes, which means that it also applies to Sub-Processes. This attribute was redefined to be a list of Artifacts and to be dependent on the InputSet attribute.
- The OutputSet attribute was added to the set of common activity attributes.
- The Output attribute was included in the set of common activity attributes, which means that it also applies to Sub-Processes. This attribute was redefined to be a list of Artifacts

and to be dependent on the OutputSet attribute.

- The IORule attribute was added to the set of common activity attributes.
- The StartQuantity attribute was added to the set of common activity attributes.
- The multiplicity of the AssignTime attribute, within the set of common activity attributes, was changed to 0 to many (*).
- The specification of the LoopType attribute, within the set of common activity attributes, was updated to note that Receive Tasks that instantiate the Process cannot be a looping activity.
- The definition of the AssignTime attribute, within the set of common activity attribute, was updated to show how assignment can be used for defining Property values as inputs and outputs of an activity.
- The Counter attribute, within the set of looping activity attributes, was renamed to LoopCounter and its type was changed to Integer. The description of the attribute was updated to show the difference between its use for Standard and Multi-Instance loops.
- The Maximum attribute, within the set of standard looping activity attributes, was renamed to LoopMaximum and its type was changed to Integer.
- The InstanceGeneration attribute, within the set of Multi-Instance looping activity attributes, was renamed to MI_Ordering.
  - The Serial type for this attribute was renamed to Sequential.
- The LoopFlowCondition attribute, within the set of Multi-Instance looping activity attributes, was renamed to MI_FlowCondition.
  - A None type was added to the types for the MI_FlowCondition attribute. The definitions of the One type and the All type were updated.
- The Complex attribute, within the set of Multi-Instance looping activity attributes, was renamed to ComplexMI_FlowCondition.
- The description of the TestTime attribute, within the set of standard looping activity attributes, was updated to show that a TestTime of After is the same as a programming until function.

4.3.2 Sub-Process

A **Sub-Process** is a compound activity in that it has detail that is defined as a flow of other activities. A Sub-Process is a graphical object within a Process Flow, but it also references another Process (either embedded or independent). A Sub-Process shares the same shape as the Task, which is a rectangle.

- A Sub-Process is a rounded corner rectangle that MUST be drawn with a single thin black line.
  - The use of text, color, size, and lines for a Sub-Process MUST follow the rules defined in section 3.3 on page 40 with the exception that.
    - The boundary drawn with a double line SHALL be reserved for Sub-Process that have its IsATransaction attribute set to True.

The Sub-Process can be in a collapsed view that hides its details (see Figure 6) or a Sub-Process can be in an expanded view that shows its details within the view of the Process in

which it is contained (see Figure 7). In the collapsed form, the Sub-Process object uses a marker to distinguish it as a Sub-Process, rather than a Task.

- The Sub-Process marker MUST be a small square with a plus sign (+) inside. The square MUST be positioned at the bottom center of the shape.

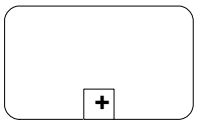

Figure 6 Collapsed Sub-Process

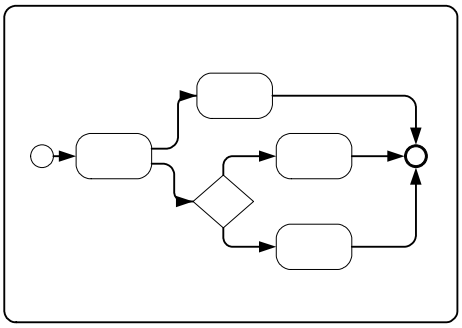

Figure 7 Expanded Sub-Process

Expanded Sub-Process may be used for multiple purposes. They can be used to “flatten” a hierarchical process so that all detail can be shown at the same time. They are used to create a context for exception handling that applies to a group of activities (Refer to the section entitled “Exception Flow” on page 149 for more details). Compensations can be handled the similarly (Refer to the section entitled “Compensation Association” on page 152 for more details).

Expanded Sub-Process may be used as a mechanism for showing a group of parallel activities in a less-cluttered, more compact way. In Figure 8, activities “C” and “D” are enclosed in an unlabeled Expanded Sub-Process. These two activities will be performed in parallel. Notice that the Expanded Sub-Process does not include a Start Event or an End Event and the Sequence Flow to/from these Events. This usage of Expanded Sub-Processes for “parallel boxes” is the motivation for having Start and End Events being optional objects.

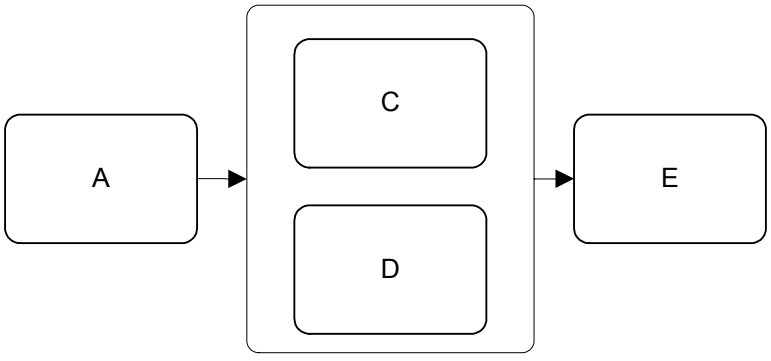

Figure 8 Expanded Sub-Process used as a “parallel box”

BPMN specifies five types of standard markers for Sub-Processes. The (Collapsed) Sub-Process Marker, seen in Figure 6, can be combined with four other markers: a Loop Marker or a Parallel Marker, a Compensation Marker, and an Ad Hoc Marker. A Sub-Process may have

one to three of these other markers, in all combinations except that Loop and Multiple Instance cannot be shown at the same time (see Figure 9).

- The marker for a Sub-Process that loops MUST be a small line with an arrowhead that curls back upon itself.
  - The Loop Marker MAY be used in combination with any of the other markers except the Multiple Instance Marker.
- The marker for a Sub-Process that has multiple instances MUST be a pair of vertical lines in parallel.
  - The Multiple Instance Marker MAY be used in combination with any of the other markers except the Loop Marker.
- The marker for a Sub-Process that is Ad Hoc MUST be a “tilde” symbol.
  - The Ad-Hoc Marker MAY be used in combination with any of the other markers.
- The marker for a Sub-Process that is used for compensation MUST be a pair of left facing triangles (like a tape player “rewind” button).
  - The Compensation Marker MAY be used in combination with any of the other markers.
- All the markers that are present MUST be grouped and the whole group centered at the bottom of the shape.

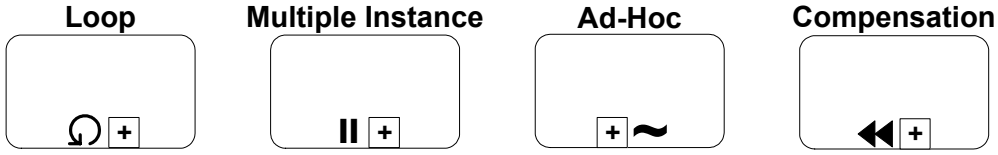

Figure 9 Collapsed Sub-Process Markers

Attributes

The following are attributes of a Sub-Process, which extends the set of common object attributes (see Table 7):

Attributes	Description
SubProcessType: (Embedded | Independent): Embedded	SubProcessType is an attribute that defines whether the Sub-Process details are embedded within the higher level Process or refers to another, re-usable Process. The default is Embedded. Attributes specific to an Independent SubProcessType can be found in Table 20.
Expanded: Boolean: False	This attribute is used to determine whether or not the Sub-Process is expanded. If the Sub-Process is not expanded, then the Sub-Process marker is shown in the bottom center of the shape (see Figure 6).
IsATransaction: Boolean: False	IsATransaction determines whether or not the behavior of the Sub-Process will follow the behavior of a Transaction (see refer to the section entitled “Sub-Process Behavior as a Transaction” on page 73).
Transaction: Transaction	If the Transaction attribute is True, then the Transaction MUST be identified. The attributes of a Transaction can be found section entitled “Transaction” on page 260. Note that Transactions that are in different Pools and are connected through Message Flow MUST have the same TransactionId.

Attributes	Description
TransactionProtocol: String	This identifies the Protocol (e.g., WS-Transaction or BTP) that will be used to control the transactional behavior of the Sub-Process.
TransactionMethod (Compensate | Store | Image): Compensate	**TransactionMethod** is an attribute that defines the technique that will be used to undo a Transaction that has been cancelled. The default is Compensate, but the attribute MAY be set to Store or Image.

Table 18 Sub-Process Attributes

Embedded Sub-Process

The following are additional attributes of a Embedded Sub-Process (where the SubProcessType attribute is set to “Embedded”), which extends the set of Sub-Process attributes (see Table 18):

Attributes	Description
GraphicalElements*: ObjectID	The GraphicalElements attribute identifies all of the objects (e.g., Events, Activities, Gateways, and Artifiacts) that are contained within the Embedded Sub-Process.
AdHoc: Boolean: False	AdHoc is a Boolean attribute, which has a default of False. This specifies whether the Embedded Sub-Process is Ad Hoc or not. The activities within an Ad Hoc Embedded Sub-Process are not controlled or sequenced in a particular order, there performance is determined by the performers of the activities.
AdHocCompletionCondition? : Expression	If the Embedded Sub-Process is Ad Hoc (the AdHoc attribute is True), then a Completion Condition MUST be included, which defines the conditions when the Process will end. The Ad Hoc marker SHALL be placed at the bottom center of the Process or the Sub-Process shape for Ad Hoc Processes.

Table 19 Embedded Sub-Process Attributes

Independent Sub-Process

The following are additional attributes of a Embedded Sub-Process (where the SubProcessType attribute is set to “Embedded”), which extends the set of Sub-Process attributes (see Table 18):

Attributes	Description
ProcessRef: Process	If the SubProcessType is Independent, then the Process MUST be identified. The attributes of a Process can be found section entitled “Business Process” on page 43.
InputPropertyMap*: Expression	For Independent, multiple input mappings MAY be made between properties of the Independent Sub-Process and the properties of the Process referenced by this object. These mappings are in the form of an expression (although a modeling tool can present this to a modeler in any number of ways).
OutputPropertyMap*: Expression	For Independent, multiple output mappings MAY be made between properties of the Independent Sub-Process and the properties of the Process referenced by this object. These mappings are in the form of an expression (although a modeling tool can present this to a modeler in any number of ways).

Table 20 Independent Sub-Process Attributes

Sub-Process Behavior as a Transaction

A Sub-Process, either collapse or expanded, can be set as being a Transaction, which will have a special behavior that is controlled through a transaction protocol (such as BTP or WS-Transaction). The boundary of the activity will be double-lined to indicate that it is a Transaction (see Figure 10).

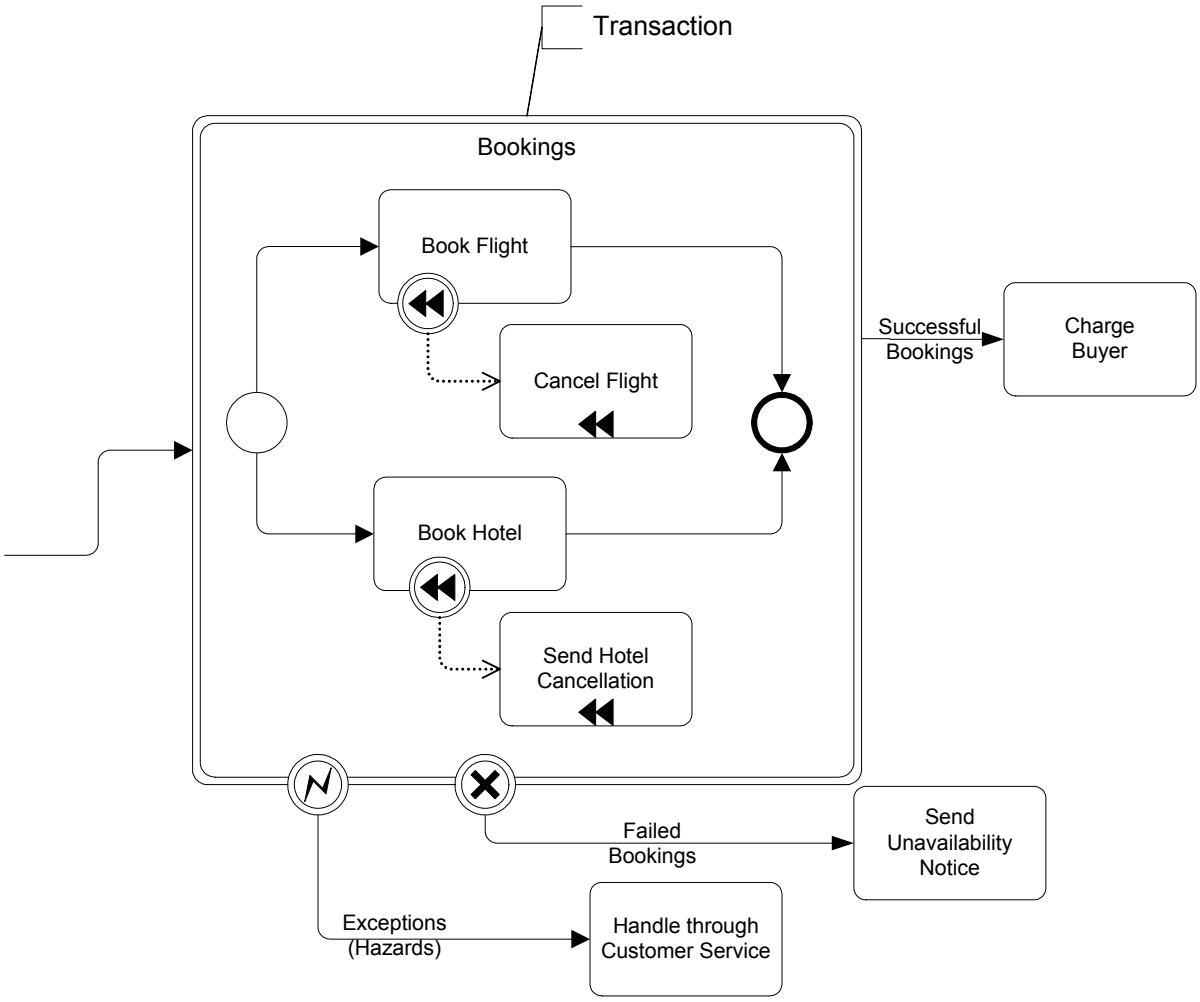

Figure 10 An Example of a Transaction Expanded Sub-Process

There are three basic outcomes of a Transaction:

- Successful completion: this will be shown as a normal Sequence Flow that leaves the Sub-Process.
- Failed completion (Cancel): When a Transaction is cancelled, then the activities inside the Transaction will be subjected to the cancellation actions, which could include rolling back the process and compensation for specific activities. Note that other mechanisms for interrupting a Sub-Process will not cause Compensation (e.g., Exception, Timer, and anything for a non-Transaction activity). A Cancel Intermediate Event, attached to the boundary of the activity, will direct the flow after the Transaction has been rolled back and all compensation has been completed. The Cancel Intermediate Event can only be used

when attached to the boundary of a Transaction activity. It cannot be used in any normal flow and cannot be attached to a non-Transaction activity. There are two mechanisms that can signal the cancellation of a Transaction:

- A Cancel End Event is reached within the Transaction Sub-Process. A Cancel End Event can only be used within a Sub-Process that is set to a Transaction.
- A Cancel Message can be received via the Transaction Protocol that is supporting the execution of the Sub-Process.

- Hazard: This means that something went terribly wrong and that a normal success or cancel is not possible. We are using an Exception to show Hazards. When a Hazard happens, the activity is interrupted (without Compensation) and the flow will continue from the Exception Intermediate Event.

The behavior at the end of a successful Transaction Sub-Process is slightly different than that of a normal Sub-Process. When each path of the Transaction Sub-Process reaches a non-Cancel End Event(s), the flow does not immediately move back up to the higher-level Parent Process, as does a normal Sub-Process. First, the transaction protocol must verify that all the participants have successfully completed their end of the Transaction. Most of the time this will be true and the flow will then move up to the higher-level Process. But it is possible that one of the participants may end up with a problem that causes a Cancel or a Hazard. In this case, the flow will then move to the appropriate Intermediate Event, even though it had apparently finished successfully.

Note: The exact behavior and notation for defining Transactions is still an Open Issue. Refer to the section entitled “Open Issues” on page 231 for a complete list of the issues open for BPMN.

Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

- A Sub-Process MAY be a target for a Sequence Flow; it can have multiple incoming Flows. Incoming Flow MAY be from an alternative path and/or a parallel paths.

Note: If the Sub-Process has multiple incoming Sequence Flows, then this is considered uncontrolled flow. This means that when a Token arrives from one of the Paths, the Sub-Process will be instantiated. It will not wait for the arrival of Tokens from the other paths. If another Token arrives from the same path or another path, then a separate instance of the Sub-Process will be created. If the flow needs to be controlled, then the flow should converge on a Gateway that precedes the Sub-Process (Refer to the section entitled “Gateways” on page 82 for more information on Gateways).

  - If the Sub-Process does not have an incoming Sequence Flow, and there is no Start Event for the Process, then the Sub-Process MUST be instantiated when the process is instantiated.
    - An exception to this are Sub-Processes that are defined as being Compensation activities (have the Compensation Marker). Compensation Sub-Processes are not

considered a part of the normal flow and SHALL NOT be instantiated when the Process is instantiated.

- A Sub-Process MAY be a source for a Sequence Flow; it can have multiple outgoing Flows. If there are multiple outgoing Sequence Flows, then this means that a separate parallel path is being created for each Flow.

Tokens will be generated for each outgoing Sequence Flow from Sub-Process. The TokenIds for each of the Tokens will be set such that it can be identified that the Tokens are all from the same parallel Fork (AND-Split) and the number of Tokens in the group

- If the Sub-Process does not have an outgoing Sequence Flow, and there is no End Event for the Process, then the Sub-Process marks the end of one or more paths in the Process. When the Sub-Process ends and there are no other parallel paths active, then the Process MUST be completed.
  - An exception to this are Sub-Processes that are defined as being Compensation activities (have the Compensation Marker). Compensation Sub-Processes are not considered a part of the normal flow and SHALL NOT mark the end of the Process.

Message Flow Connections

Refer to the section entitled “Message Flow Rules” on page 42 for the entire set of objects and how the may be source or targets of Sequence Flows.

Note: All Message Flows must connect two separate Pools. They can connect to the Pool boundary or to flow objects within the Pool boundary. They cannot connect two objects within the same Pool.

- A Sub-Process MAY be the target for Message Flows; it can have zero or more incoming Message Flows.
- A Sub-Process MAY be a source for a Message Flow; it can have zero or more outgoing Message Flows.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Sub-Process was removed.
- Attributes that are specific to a Sub-Process of type Embedded a were added and presented in a new table (Table 19).
- The set of attributes that are specific to a Sub-Process of type Independent were removed from the main table (Table 18) and placed in a separate table (Table 20).
- The Expanded attribute was added to the set of Sub-Process attributes.
- The TransactionProtocol and TransactionMethod attributes were removed from the set of Sub-Process attributes. These attributes can be found in the definition of a Transaction, which can be found in the section entitled “Transaction” on page 260.
- The ProcessRef attribute, within the set of Independent Sub-Process attributes, was

changed to be of type Process.

- The Process attribute was removed from the set of Sub-Process attributes.
- The InputMap attribute, within the set of Independent Sub-Process attributes, was renamed to InputPropertyMap and its multiplicity was change to 0 to many (*). In addition, the description was updated to clarify that the mapping was between the properties of the object and the properties of the referenced Process.
- The OutputMap attribute, within the set of Independent Sub-Process attributes, was renamed to OutputPropertyMap and its multiplicity was change to 0 to many (*). In addition, the description was updated to clarify that the mapping was between the properties of the object and the properties of the referenced Process.

4.3.3 Task

A Task is an atomic activity that is included within a Process. A Task is used when the work in the Process is not broken down to a finer level of Process Model detail. Generally, an end-user and/or an application are used to perform the Task when it is executed.

A Task object shares the same shape as the Sub-Process, which is a rectangle that has rounded corners (see Figure 11).

- A Task is a rounded corner rectangle that MUST be drawn with a single thin black line.
  - The use of text, color, size, and lines for a Task MUST follow the rules defined in section 3.3 on page 40.

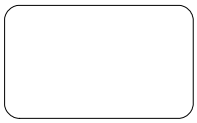

Figure 11 A Task Object

BPMN specifies three types of markers for Task: a Loop Marker or a Multiple Instance Marker and an Ad Hoc Marker. A Task may have one or two of these markers (see Figure 12).

- The marker for a Task that loops MUST be a small line with an arrowhead that curls back upon itself.
  - The Loop Marker MAY be used in combination with the Compensation Marker.
- The marker for a Task that has multiple instances MUST be a pair of vertical lines in parallel.
  - The Multiple Instance Marker MAY be used in combination with the Compensation Marker.
- The marker for a Task that is used for compensation MUST be a pair of left facing triangles (like a tape player “rewind” button).
  - The Compensation Marker MAY be used in combination with the Loop Marker or the Multiple Instance Marker.
- All the markers that are present MUST be grouped and the whole group centered at the bottom of the shape.

All the markers that are present will be grouped and the whole group will be centered at the bottom of the shape.

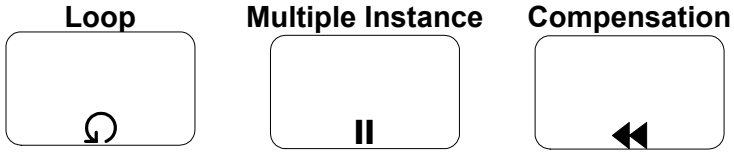

Figure 12 Task Markers

In addition to categories of Task shown above, there are different types of Tasks identifiied within BPMN to separate the types of inherent behavior that Tasks might represent (see Table 7). However, BPMN does not specify any graphical indicators for the different types of Tasks. It is expected that modelers or modeling tools will create their own indicators or markers to show the readers of the diagram the type of Task. This is permitted by BPMN as long as the basic shape of the Task (a rounded rectangle) is not modified. The list of Task types may be extended along with any corresponding indicators.

Attributes

The following are attributes of a Task, which extends the set of common object attributes (see Table 7):

Attributes	Description
TaskType (Service | Receive | Send | User | Script | Manual | Reference | None): Service	TaskType is an attribute that has a default of Service, but MAY be set to Send, Receive, User, Script, Manual, Reference, or None. The TaskType will be impacted by the Message Flows to and/or from the Task, if Message Flows are used. A TaskType of Receive SHALL NOT have an outgoing Message Flow. A TaskType of Send SHALL NOT have an incoming Message Flow. A TaskType of Script, Manual, or None SHALL NOT have an incoming or an outgoing Message Flow. The TaskType list MAY be extended to include new types. The attributes for specific values of TaskType can be found in Table 20 through Table 28.

Table 21 Task Attributes

Service Task

A Service Task is a Task that does not involve a human participation

The following are attributes of a Service Task (where the TaskType attribute is set to “Service”), which extends the set of Task attributes (see Table 21):

Attributes	Description
OutMessage: Message	A Message for the OutMessage attribute MUST be entered. This indicates that the Message will be sent at the start of the Task, after the availability of any defined InputSets. The combination of OutMessage and InMessage (see row below) is equivalent to a *out-in* message pattern (Web service). A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.

Attributes	Description
InMessage: Message	A Message for the InMessage attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet. The combination of InMessage and OutMessage (see row above) is equivalent to a *out-in* message pattern (Web service). A corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
Implementation: (Web Service | Other | Unspecified): Web Service	This attribute specifies the technology that will be used to send and receieve the messages. A Web service is the default technology.

Table 22 Service Task Attributes

Receive Task

A Recieve Task is a simple Task that is designed to wait for a message to arrive from an external entitiy (relative to the the Business Process). Once the message has been received, the Task is completed.

A Recieve Task is often used to start a Process. In a sense, the Process is bootstrapped by the receipt of the message. In order for the Task to Instantiate the Process it must meet one of the following conditions:

- The Process does not have a Start Event and the Receive Task has no incoming Sequence Flow.
- The Incoming Sequence Flow for the Receive Task has a source of a Start Event.
  - Note that no other incoming Sequence Flow are allowed for the Recieve Task (in particular, a loop connection from a downstream object).

The following are attributes of a Receive Task (where the TaskType attribute is set to “Receive”), which extends the set of Task attributes (see Table 21):

Attributes	Description
Message: Message	A Message for the Message attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet. The Message in this context is equivalent to a *in-only* message pattern (Web service). A corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
Instantiate: Boolean: False	Receive Tasks can be defined as the instantiation mechanism for the Process with the Instantiate attribute. This attribute MAY be set to true if the Task is the first activity after the Start Event or a starting Task if there is no Start Event (i.e., there are no incoming Sequence Flow). [here] Multiple Tasks MAY have this attribute set to True.
Implementation: (Web Service | Other | Unspecified): Web Service	This attribute specifies the technology that will be used to receieve the message. A Web service is the default technology.

Table 23 Receive Task Attributes

Send Task

A Send Task is a simple Task that is designed to send a message to an external entitiy (relative to the the Business Process). Once the message has been sent, the Task is completed.

The following are attributes of a Send Task (where the TaskType attribute is set to “Send”), which extends the set of Task attributes (see Table 21):

Attributes	Description
Message: Message	A Message for the Message attribute MUST be entered. This indicates that the Message will be sent at the start of the Task, after the availability of any defined InputSets. The Message in this context is equivalent to a *out-only* message pattern (Web service). A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
Implementation: (Web Service | Other | Unspecified): Web Service	This attribute specifies the technology that will be used to send the message. A Web service is the default technology.

Table 24 Send Task Attributes

User Task

A User Task is a typical “workflow” task where a human performer performs the Task with the assistance of a software application and is scheduled through a task list manager of some sort.

The following are attributes of a User Task (where the TaskType attribute is set to “User”), which extends the set of Task attributes (see Table 21):

Attributes	Description
Performer: String	One or more Performers MAY be entered. The Performer attribute defines the human resource that will be performing the Manual Task. The Performer entry could be in the form of a specific individual, a group, or an organization. Additional parameters that help define the Performer assignment can be added by a modeling tool.
Implementation: (Web Service | Other | Unspecified): Web Service	This attribute specifies the technology that will be used by the Performer to perform the Task. A Web service is the default technology.

Table 25 User Task Attributes

Script Task

A Script Task is executed by a business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the Task is ready to start, the engine will execute the script. When the script is completed, the Task will also be completed.

The following are attributes of a Script Task (where the TaskType attribute is set to “Script”), which extends the set of Task attributes (see Table 21):

Attributes	Description
Script?: String	The modeler MAY include a script that can be run when the Task is performed. If a script is not included, then the Task will act equivalent to a TaskType of None.

Table 26 Script Task Attributes

Manual Task

The following are attributes of a Manual Task (where the TaskType attribute is set to “Manual”), which extends the set of Task attributes (see Table 21):

Attributes	Description
Performer*: String	One or more Performers MAY be entered. The Performer attribute defines the human resource that will be performing the Manual Task. The Performer entry could be in the form of a specific individual, a group, or an organization.

Table 27 Manual Task Attributes

Reference Task

There may be times where a modeler, with or without a modeling tool, may want to refence another activity that has been defined. If the two (or more) activities share the exact same behavior, then by one referencing the other, the attributes that define the behavior only have to be created once and maintained in only one location.

The following are attributes of a Reference Task (where the TaskType attribute is set to “Reference”), which extends the set of Task attributes (see Table 21):

Attributes	Description
TaskRef: Task	The Task being referenced MUST be identified. The attributes for the Task element can be found in Table 21.

Table 28 Reference Task Attributes

Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

- A Task MAY be a target for a Sequence Flow; it can have multiple incoming Flows. Incoming Flow MAY be from an alternative path and/or a parallel paths.

Note: If the Task has multiple incoming Sequence Flows, then this is considered uncontrolled flow. This means that when a Token arrives from one of the Paths, the Task will be instantiated. It will not wait for the arrival of Tokens from the other paths. If another Token arrives from the same path or another path, then a separate instance of the Task will be created. If the flow needs to be controlled, then the flow should

converge with a Gateway that precedes the Task (Refer to the section entitled "Gateways" on page 82 for more information on Gateways).

- If the Task does not have an incoming Sequence Flow, and there is no Start Event for the Process, then the Task MUST be instantiated when the process is instantiated.
  - An exception to this are Tasks that are defined as being Compensation activities (have the Compensation Marker). Compensation Tasks are not considered a part of the normal flow and SHALL NOT be instantiated when the Process is instantiated.
- A Task MAY be a source for a Sequence Flow; it can have multiple outgoing Flows. If there are multiple outgoing Sequence Flows, then this means that a separate parallel path is being created for each Flow.

Tokens will be generated for each outgoing Sequence Flow from the Task. The TokenIds for each of the Tokens will be set such that it can be identified that the Tokens are all from the same parallel Fork (AND-Split) and the number of Tokens in the group

- If the Task does not have an outgoing Sequence Flow, and there is no End Event for the Process, then the Task marks the end of one or more paths in the Process. When the Task ends and there are no other parallel paths active, then the Process MUST be completed.
  - An exception to this are Tasks that are defined as being Compensation activities (have the Compensation Marker). Compensation Tasks are not considered a part of the normal flow and SHALL NOT mark the end of the Process.

Message Flow Connections

Refer to the section entitled "Message Flow Rules" on page 42 for the entire set of objects and how the may be source or targets of Sequence Flows.

Note: All Message Flows must connect two separate Pools. They can connect to the Pool boundary or to flow objects within the Pool boundary. They cannot connect two objects within the same Pool.

- A Task MAY be the target for Message Flows; it can have zero or one incoming Message Flows.
- A Task MAY be a source for a Message Flow; it can have zero or more outgoing Message Flows.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Task was removed.
- The TaskType attribute, within the set of Task attributes, was updated to include a new type named Reference. Also, the Abstract TaskType was removed.
- The definition of the Input attribute, within the set of Task attributes, was modified to

indicate that Inputs can be either be Process Properties or Artifacts such as Data Objects.

- The RequiredForStart attribute was added as a sub-attribute for the Input attribute within the set of Task attributes.
- The definition of the Output attribute, within the set of Task attributes, was modified to indicate that Inputs can be either be Process Properties or Artifacts such as Data Objects.
- The ProducedAtCompletion attribute was added as a sub-attribute for the Output attribute within the set of Task attributes.
- The IORule attribute was added to the set of Task attributes.
- The Counter attribute, within the set of Task attributes, was renamed to LoopCounter and its type was changed to Integer.
- The Maximum attribute, within the set of Task attributes, was renamed to LoopMaximum and its type was changed to Integer.
- [The StartQuantity attribute was added to the set of Task attributes.]

4.4 Gateways

Gateways are modeling elements that are used to control how Sequence Flows interact as they converge and diverge within a Process. If the flow does not need to be controlled, then a Gateway is not needed. The term “Gateway” implies that there is a gating mechanism that either allows or disallows passage through the Gateway--that is, as Tokens arrive at a Gateway, they can be merged together on input and/or split apart on output as the Gateway mechanisms are invoked. To be more descriptive, the control of the output flow a Gateway is actually a collection of “Gates” and the behavior a particular Gateway will determine how many of the Gates will be available for the continuation of flow. There will be one Gate for each outgoing Sequence Flow of the Gateway.

A Gateway is a diamond (see Figure 13), which has been used in many flow chart notations for exclusive branching and is familiar to most modelers.

- A Gateway is a diamond that MUST be drawn with a single thin black line.
  - The use of text, color, size, and lines for a Gateway MUST follow the rules defined in section 3.3 on page 40.

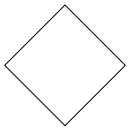

Figure 13 A Gateway

Note: Although the shape of a Gateway is a diamond, it is not a requirement that incoming and outgoing Sequence Flow must connect to the corners of the diamond. Sequence Flow can connect to any position on the boundary of the Gateway shape.

Gateways can define all the types of business process Sequence Flow behavior: Decisions/ branching (OR-Split; exclusive--XOR, inclusive--OR, and complex), merging (OR-Join), forking (AND-Split), and joining (AND-Join). Thus, while the diamond has been used

traditionally for exclusive decisions, BPMN 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 (see Figure 14).

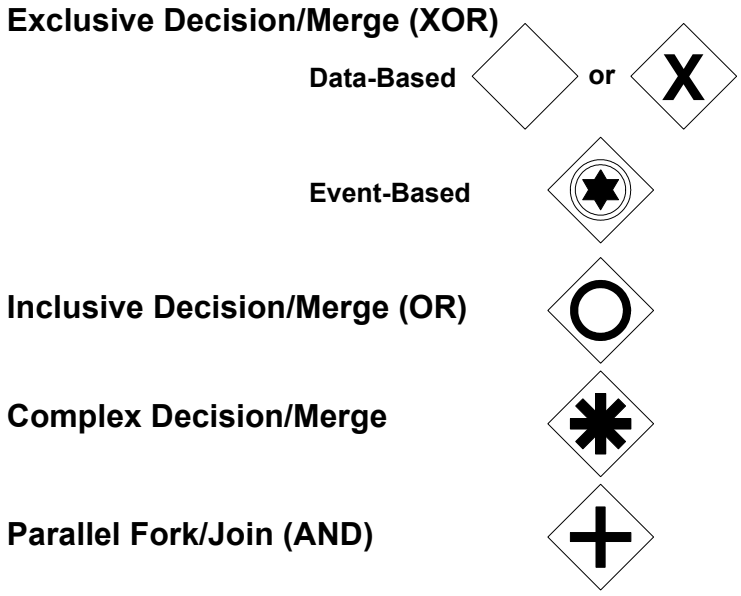

Figure 14 The Different types of Gateways

- The internal marker associated with the Gateway MUST be placed inside the shape, in any size or location, depending on the preference of the modeler or modeling tool vendor, with the exception that the marker for the Data-Based Exclusive Gateway is not required.

The Gateways will control the flow of both diverging and/or converging Sequence Flow. That is, a particular Gateway could have multiple incoming Sequence Flow and multiple outgoing Sequence Flow at the same time. The type of Gateway will determine the same type of behavior for both the diverging and converging Sequence Flow. Modelers and Modeling tools may want to enforce a best practice of a Gateway only performing one of these functions. Thus, it would take two sequential Gateways to first converge and then diverge the Sequence Flow.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Gateway was removed.
- The constraint for placing the internal marker inside the Gateway was changed from MAY to MUST.

4.4.1 Common Gateway Features

Common Gateway Attributes

The following table displays the attributes common for all types of Gateways, and which extends the set of common object attributes (see Table 7):

Attributes	Description
GatewayType: (XOR | OR | Complex | AND): XOR	GatewayType is by default XOR. The GatewayType MAY be set to OR, Complex, or AND. The GatewayType will determine the behavior of the Gateway, both for incoming and outgoing Sequence Flow, and will determine the internal indicator (as shown in Figure 14).

Table 29 Common Gateway Attributes

Common Gateway Sequence Flow Connections

This section applies to all Gateways. Additional Sequence Flow Connection rules will be specified for each type of Gateway in the sections below. Refer to the section entitled "Sequence Flow Rules" on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

- A Gateway MAY be a target for a Sequence Flow; it can have zero or more incoming Sequence Flows. An incoming Flow MAY be from an alternative path or a parallel path.
  - If the Gateway does not have an incoming Sequence Flow, and there is no Start Event for the Process, then the Gateway's divergence behavior, depending on the GatewayType attribute (see below), SHALL be performed when the Process is instantiated.
- A Gateway MAY be a source of Sequence Flow; it can have zero or more outgoing Flows.
- A Gateway MAY have both multiple incoming and outgoing Sequence Flow.

Message Flow Connections

This section applies to both Data-Based and Event-Based Exclusive Gateways. Refer to the section entitled "Message Flow Rules" on page 42 for the entire set of objects and how the may be source or targets of Sequence Flows.

- An Gateway MAY NOT be a target for a Message Flow.
- An Gateway MAY NOT be a source for a Message Flow.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

-

4.4.2 Exclusive Gateways (XOR)

Exclusive Gateways (Decisions) are locations within a business process where the Sequence Flow can take two or more alternative paths. This is basically the "fork in the road" for a process. For a given performance (or instance) of the process, only one of the paths can be taken (this should not be confused with forking of paths—refer to the section entitled "Forking

Flow" on page 131). A Decision is not an activity from the business process perspective, but is a type of Gateway that control the Sequence Flow between activities. It can be thought of as a question that is asked at that point in the Process. The question has a defined set of alternative answers (Gates). Each Decision Gate is associated with a condition expression found within an outgoing Sequence Flow. When an Gate is chosen during the performance of the Process, the corresponding Sequence Flow is then chosen. A Token arriving at the Decision would be directed down the appropriate path, based on the chosen Gate.

The Exclusive Decision has two or more outgoing Sequence Flows, but only one of them may be taken during the performance of the Process. Thus, the Exclusive Decision defines a set of alternative paths for the Token to take as it traverses the Flows. There are two types of Exclusive Decisions: Data-Based and Event-Based.

Data-Based

The Data-Based Exclusive Gateways are the most commonly used type of Gateways. The set of Gates for Data-Based Exclusive Decisions are based on the boolean expression contained ConditionExpression attribute of the outgoing Sequence Flow of the Gateway. These expressions use the values of process data to determine which path should be taken (hence the name Data-Based).

Note: BPMN does not specify the format of the expressions used in Gateways or any other BPMN element that uses expressions.

- The Data-Based Exclusive Gateway MAY use a marker that is shaped like an "X" and is placed within the Gateway diamond (see Figure 16) to distinguish it from other Gateways. This marker is not required (see Figure 15).
  - A Diagram SHOULD be consistent in the use of the "X" internal indicator. That is, a Diagram SHOULD NOT have some Gateways with an indicator and some Gateways without an indicator.

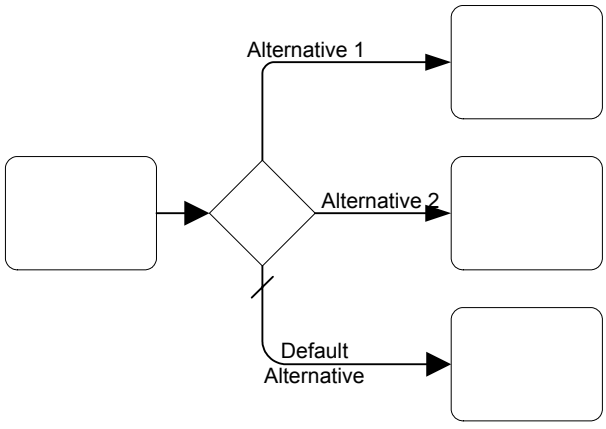

Figure 15 An Exclusive Data-Based Decision (Gateway) Example without the Internal Indicator

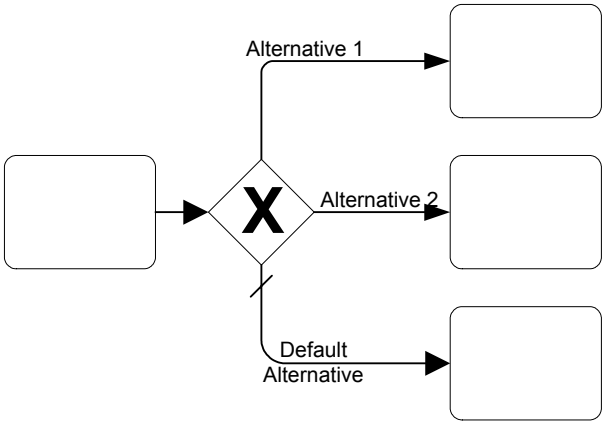

Figure 16 A Data-Based Exclusive Decision (Gateway) Example with the Internal Indicator

The conditions for the alternative Gates should be evaluated in a specific order. The first one that evaluates as TRUE will determine the Sequence Flow that will be taken. Since the behavior of this Gateway is exclusive, any other conditions that may actually be TRUE will be ignored--only one Gate can be chosen. One of the Gates may be “default” (or otherwise), and is the last Gate considered. This means that if none of the other Gates are chosen, then the default Gate will be chosen—along with its associated Sequence Flow.

The default Gate is not mandatory for a Gateway. This means that if it is not used, then it is up to the modeler to insure that at least one Gate be valid at runtime. BPMN does not specify what will happen if there are no valid Gates. However, BPMN does specify that there SHALL NOT be implicit flow and that all normal flow of a Process must be expressed through Sequence Flow. This would mean that a Process Model that has a Gateway that potentially does not have a valid Gate at runtime is an invalid model.

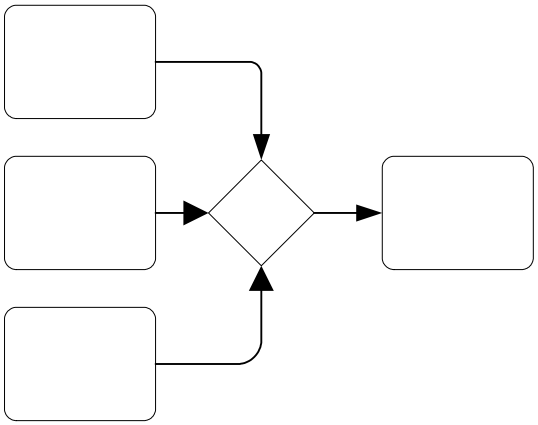

Figure 17 An Exclusive Merge (Gateway) (without the Internal Indicator)

Exclusive Gateways can also be used as a merge (see Figure 17), although it is rarely required for the modeler to use them this way. The merging behavior of the Gateway can also be modeled as seen in Figure 18. The behavior of Figure 17 and Figure 18 are the same if all the incoming flow are alternative. This is true because when a Token arrives at an activity, that activity will be instantiated. The Exclusive Gateway merely merges the Sequence Flow into a single Sequence Flow, but it does not restrict the flow of Tokens through the Gateway. That is, if there happens to be some parallel incoming Sequence Flow for the Gateway, each Token that traverses the Sequence Flow into the Gateway will immediate pass through without

waiting for any other potential Token that may come along. If another Token happens through the Gateway, it will also continue through without being restricted by any previous or future Tokens that may also pass through. Thus, it is not necessary to have the Sequence Flow merge through the Gateway prior to the activity.

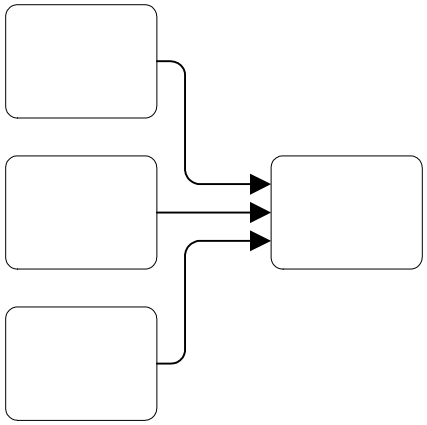

Figure 18 Uncontrolled Merging of Sequence Flow

There are certain situations where an Exclusive Gateway is required to act as a merging object. In Figure 20 an Exclusive Gateway (labeled “Merge”) merges two alternative Sequence Flow that were generated by an upstream Decision. The alternative Sequence Flow are merged in preparation for an Parallel Gateway that synchronizes a set of parallel Sequence Flow that were generated even further upstream. If the merging Gateway was not used, then there would have been four incoming Sequence Flow into the Parallel Gateway. However, only three of the four Sequence Flow would ever pass a Token at one time. Thus, the Gateway would be waiting for a fourth Token that would never arrive. Thus, the Process would be stuck at the point of the Parallel Gateway.

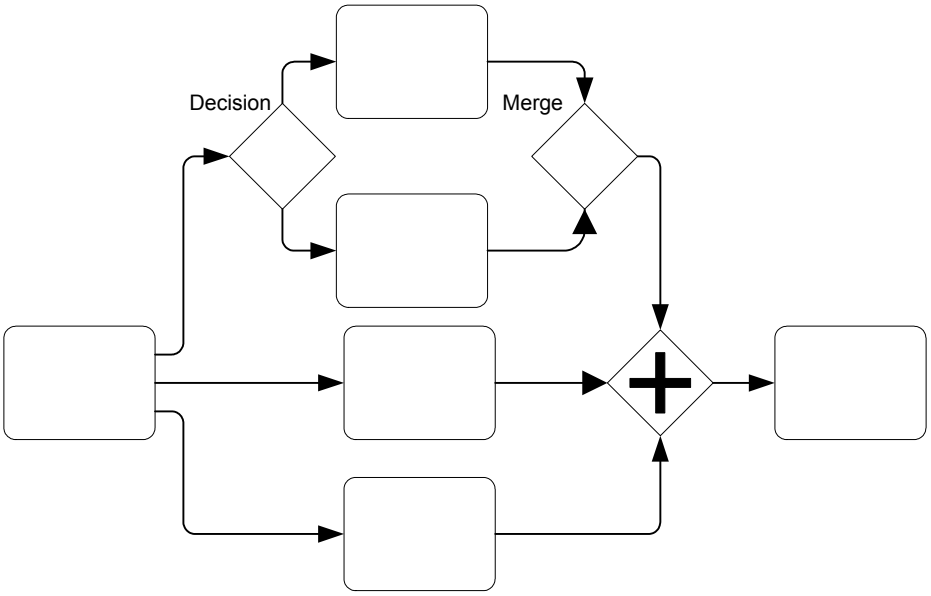

Figure 19 Exclusive Gateway that merges Sequence Flow prior to an Parallel Gateway

In simple situations, Exclusive Gateways need not be used for merging Sequence Flow, but there are more complex situations where they are required. Thus, a modeler should always be aware of the behavior of a situation where Sequence Flow are uncontrolled. Some modelers or modeling tools may, in fact, require that Exclusive Gateways be used in all situations as a matter of Best Practice.

Attributes

The following table displays the attributes for an Data-Based Exclusive Gateway. These attributes only apply if the GatewayType attribute is set to XOR. The following attributes extend the set of common Gateway attributes (see Table 29):

Attributes	Description
XORType: (Data | Event): Data	XORType is by default Data. The XORType MAY be set to Event. Since Data-Based XOR Gateways is the subject of this section, the attribute MUST be set to Data for the attributes and behavior defined in this section to apply to the Gateway.
MarkerVisible: Boolean: False	This attribute determines if the XOR Marker in the center of the Gateway diamond (an "X"). The marker is displayed if the attribute is True and it is not displayed if the attribute is False (by default).
Gate*: ObjectId	There MAY be zero or more Gates. Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow (i.e, the Gateway is acting as a Decision), then there MUST be at least one Gate. In this case, if there is no DefaultGate, then there MUST be at least two Gates.
OutgoingSequenceFlow: SequenceFlowId	Each Gate MUST have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to Expression and MUST have a valid ConditionExpression. If there is only one Gate (i.e., the Gateway is acting only as a Merge), then Sequence Flow MUST have its Condition set to None.
Assign*: Expression	Zero or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled "Assignment" on page 261. The type for each side of the Assignment MUST match.
DefaultGate?: ObjectId	A Default Gate MAY be specified.
OutgoingSequenceFlow: SequenceFlowId	If there is a DefaultGate, the it MUST have an associated Sequence Flow. The Sequence Flow SHALL have the Default Indicator (see Figure 15). The Sequence Flow MUST have its Condition attribute set to Default.
Assign*: Expression	Zero or more assignment expressions MAY be made for the DefaultGate. The Assignment SHALL be performed when the DefaultGate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled "Assignment" on page 261. The type for each side of the Assignment MUST match.

Table 30 Data-Based Exclusive Gateway Attributes

Sequence Flow Connections

This section extends the basic Gateway Sequence Flow connection rules as defined in the section entitled “Common Gateway Sequence Flow Connections” on page 84. Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

To define the exclusive nature of this Gateway’s behavior for converging Sequence Flow:

- If there are multiple incoming Sequence Flows, all of them will be used to continue the flow of the Process (as if there were no Gateway). That is,
  - Process flow SHALL continue when a signal (a Token) arrives from any of a set of Sequence Flows.
    - Signals from other Sequence Flow within that set may arrive at other times and the flow will continue when they arrive as well, without consideration or synchronization of signals that have arrived from other Sequence Flow.

To define the exclusive nature of this Gateway’s behavior for diverging Sequence Flow:

- If there are multiple outgoing Sequence Flow, then only one Gate (or the DefaultGate) SHALL be selected during performance of the Process.
  - The Gate SHALL be chosen based on the result of evaluating the ConditionExpression that is defined for the Sequence Flow associated with the Gate.
    - The Conditions associated with the Gates SHALL be evaluated in the order in which the Gates appear on the list for the Gateway.
    - If a ConditionExpression is evaluated as “TRUE,” then that Gate SHALL be chosen and any Gates remaining on the list SHALL NOT be evaluated.
    - If none of the ConditionExpressions for the Gates are evaluated as “TRUE,” then the DefaultGate SHALL be chosen.

Note: If the Gateway does not have a DefaultGate and none of the Gate ConditionExpressions are evaluated as “TRUE,” then the Process is considered to have an invalid model.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The Gate and DefaultGate attributes, within the set of Data-Based Exclusive Gateway attributes, were changed to type ObjectId.

Event-Based

The inclusion of Event-Based Exclusive Gateways is the result of recent developments in the handling of distributed systems (e.g., with pi-calculus) and will map to the BPEL4WS *pick*. On the input side, their behavior is the same as a Data-Based Exclusive Gateway (refer to the section entitled “Data-Based” on page 85 above). On the output side, the basic idea is that this Decision represents a branching point in the process where the alternatives are based on an events that occurs at that point in the Process, rather than the evaluation of expressions using

process data. A specific event, usually the receipt of a message, determines which of the paths will be taken. For example, if a company is waiting for a response from a customer, they will perform one set of activities if the customer responds “Yes” and another set of activities if the customer responds “No.” The customer’s response determines which path is taken. The identity of the Message determines which path is taken. That is, the “Yes” Message and the “No” message are different messages—they are not the same message with different values within a property of the Message. The receipt of the message can be modeled with a Task of TaskType Receive or an Intermediate Event with a Message Trigger. In addition to Messages, other Triggers for Intermediate Events can be used, such as Timers and Exceptions.

- The Event-Based Exclusive Gateway MUST use a marker that is the same as the Multiple Intermediate Event and is placed within the Gateway diamond (see Figure 20 and Figure 21) to distinguish it from other Gateways.
- The Event-Based Exclusive Decisions are configured by having outgoing Sequence Flows target a Task of TaskType Receive or an Intermediate Event (see Figure 20 and Figure 21).
  - All of the outgoing Sequence Flows must have this type of target; there cannot be a mixing of condition expressions and Intermediate Events for a given Decision.

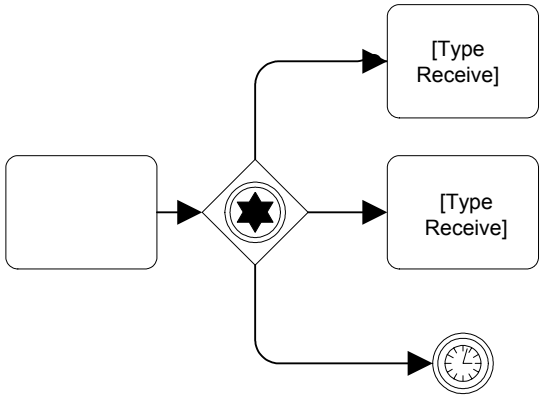

Figure 20 An Event-Based Decision (Gateway) Example Using Receive Tasks

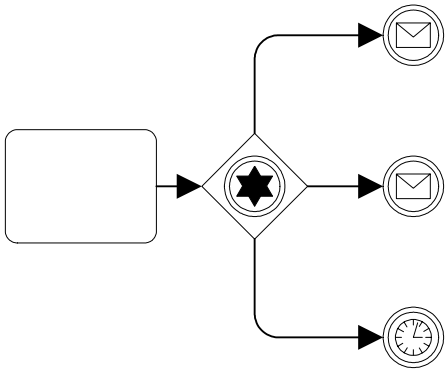

Figure 21 An Event-Based Decision (Gateway) Example Using Message Events

To relate the Event-Based Exclusive Gateway to BPEL4WS, the Gateway diamond marks the location of a BPEL4WS *pick* and the Intermediate Events that follow the Decision become the event handlers of the *pick* or *choice*. The activities that follow the Intermediate Events become

the contents of the *activity sets* for the event handlers. The boundaries of the activity sets is actually determined by the configuration of the process; that is, the boundaries extend to where all the alternative paths are finally joined together (which could be the end of the Process).

Because this Gateway is an Exclusive Gateway, the merging functionality for the Event-Based Exclusive Gateway is the same as the Data-Based Exclusive Gateway described in the previous section.

A Gateway can be used to start a Process. In a sense, the Process is bootstrapped by the receipt of a message. The reciept of any of the message defined by the Gateway configuration will instantiate the Process. Thus, the Gateway provides a set of alternative ways for the Process to begin.

In order for the Gateway to Instantiate the Process it must meet one of the following conditions:

- The Process does not have a Start Event and the Gateway has no incoming Sequence Flow.
- The Incoming Sequence Flow for the Gateway has a source of a Start Event.
  - Note that no other incoming Sequence Flow are allowed for the Gateway (in particular, a loop connection from a downstream object).
- The Targets for the Gateway's outgoing Sequence Flow MAY NOT be a Timer Intermediate Event.

Attributes

The following table displays the attributes for an Event-Based Exclusive Gateway. These attributes only apply if the GatewayType attribute is set to XOR. The following attributes extend the set of common Gateway attributes (see Table 29):

Attributes	Description
XORType: (Data | Event): Event	XORType is by default Data. The XORType MAY be set to Event. Since Event-Based XOR Gateways is the subject of this section, the attribute MUST be set to Event for the attributes and behavior defined in this section to apply to the Gateway.
Instantiate: Boolean: False	Event-Based Gateways can be defined as the instantiation mechanism for the Process with the Instantiate attribute. This attribute MAY be set to true if the Gateway is the first element after the Start Event or a starting Gateway if there is no Start Event (i.e., there are no incoming Sequence Flow).
Gate 2+: GateId	There MUST be two or more Gates. (Note that this type of Gateway does not act *only* as a Merge--it is always a Decision, at least.)
OutgoingSequenceFlow: SequenceFlowId	Each Gate MUST have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to None (there is not an evaluation of a condition expression).
Target: ObjectId	The targets of the Sequence flow MUST be an Intermediate Event or a Task of TaskType Receive. Intermediate Events with Trigger of Exception, Compensation, Multiple, or Branching SHALL NOT be allowed as a Target. If a Receive Task is the Target for one Alterative, then a Message Intermediate Event SHALL NOT be allowed for Targets of other Gates.
Assign*: Expression	Zero or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.

Table 31 Event-Based Exclusive Gateway Attributes

Sequence Flow Connections

This section extends the basic Gateway Sequence Flow connection rules as defined in the section entitled “Common Gateway Sequence Flow Connections” on page 84. Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

To define the exclusive nature of this Gateway’s behavior for converging Sequence Flow:

- If there are multiple incoming Sequence Flows, all of them will be used to continue the flow of the Process (as if there were no Gateway). That is,
  - Process flow SHALL continue when a signal (a Token) arrives from any of a set of Sequence Flows.

- Signals from other Sequence Flow within that set may arrive at other times and the flow will continue when they arrive as well, without consideration or synchronization of signals that have arrived from other Sequence Flow.

To define the exclusive nature of this Gateway's behavior for diverging Sequence Flow:

- Only one Gate SHALL be selected during performance of the Process.
  - The Gate SHALL be chosen based on the Target of the Gate's Sequence Flow.
    - If a Target is instantiated (e.g., a message is received or a time is exceeded), then that Gate SHALL be chosen and the remaining Gates SHALL NOT be evaluated (i.e., their Targets will be disabled).
- The outgoing Sequence Flow Condition attribute MUST be set to None.
- The Target of the Gateway's outgoing Sequence Flows MUST be one of the following objects:
  - Task with the TaskType attribute set to Receive.
  - Intermediate Event with the Trigger attribute set to Message, Timer, Rule, Exception, or Link.
    - If one Gate Target is a Task, then an Intermediate Event with a Trigger Message MAY NOT be used as a Target for another Gate. That is, messages MUST be received by only Receive Tasks or only Message Events, but not a mixture of both for a given Gateway.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- A description of how Event-Based Gateways can instantiate a Process was added.
- The Instantiate attribute was added to the set of Event-Based Exclusive Gateway attributes.
- The Exception Intermediate Event was removed as a possible target for the Sequence Flow that exit the Gateway.

4.4.3 Inclusive Gateways (OR)

This Decision represents a branching point where Alternatives are based on conditional expressions contained within outgoing Sequence Flow. However, in this case, the True evaluation of one condition expression does not exclude the evaluation of other condition expressions. All Sequence Flow with a True evaluation will be traversed by a Token. In some sense it like is a grouping of related independent Binary (Yes/No) Decisions--and can be modeled that way. Since each path is independent, all combinations of the paths may be taken, from zero to all. However, it should be designed so that at least one path is taken.

Note: If none of the Inclusive Decision Gate ConditionExpressions are evaluated as "TRUE," then the Process is considered to have an invalid model.

There are two mechanism for modeling this type of Decision:

The first method for modeling Inclusive Decision situations does not actually use an Inclusive Gateway, but instead uses a collection of conditional Sequence Flow, marked with mini-diamonds--the Gates without the Gateway (see Figure 22). Conditional Sequence Flow have their Condition attribute set to Expression and the ConditionExpression attribute set to a boolean mathematical expression based on information available to the Process. These Sequence Flow are indicated by a “mini-diamond” marker at the start of the Sequence Flow line.

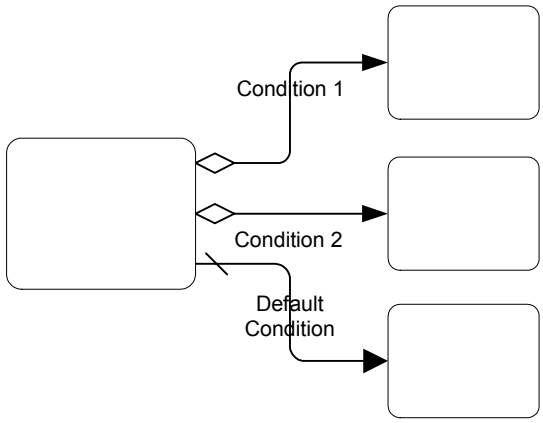

Figure 22 An Inclusive Decision using Conditional Sequence Flow

There are some restrictions in using the conditional Sequence Flow (with mini-diamonds):

- The source object MUST NOT be an Event. The source object MAY a Gateway, but the mini-diamond SHALL NOT be displayed in this case. The source object MAY be an activity (Task or Sub-Process) and the mini-diamond SHALL be displayed in this case.
  - A source Gateway MUST NOT be of type AND (Parallel).
- If a conditional Sequence Flow is used from a source activity, then there MUST be at least one other outgoing Sequence Flow from that activity
  - The additional Sequence Flow(s) MAY also be conditional, but it is not required that are conditional.

The second method for modeling Inclusive Decision situations uses an OR Gateway (see Figure 23), sometimes in combination with other Gateways. A marker will be placed in the center of the Gateway to indicate that the behavior of the Gateway is inclusive.

❖ The Inclusive Gateway MUST use a marker that is in the shape of a circle or an “O” and is placed within the Gateway diamond (see Figure 23) to distinguish it from other Gateways.

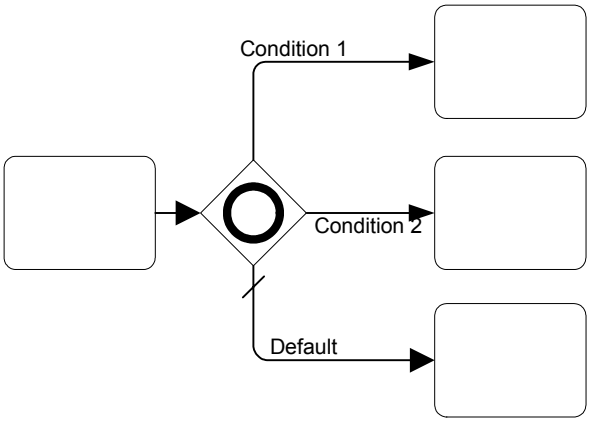

Figure 23 An Inclusive Decision using an OR Gateway

The behavior of the model depicted in Figure 22 is equivalent to the behavior of the model depicted in Figure 23. Again, it is up to the modeler to insure that at least one of the conditions will be TRUE when the Process is performed.

When the Inclusive Gateway is used as a Merge, it will wait for (synchronize) all Tokens that have been produced upstream. It does not require that all incoming Sequence Flow produce a Token (as the Parallel Gateway does). It requires that all Sequence Flow that were actually produced by an upstream (by an Inclusive OR situation, for example). If an upstream Inclusive OR produces two out of a possible three Tokens, then a downstream Inclusive OR will synchronize those two Tokens and not wait for another Token, even though there are three incoming Sequence Flow (see Figure 24).

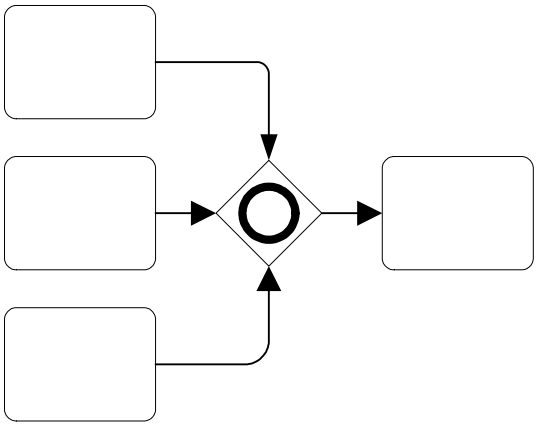

Figure 24 An Inclusive Gateway Merging Sequence Flow

Attributes

The following table displays the attributes for an Inclusive Gateway[1]. These attributes only apply if the GatewayType attribute is set to OR. The following attributes extend the set of common Gateway attributes (see Table 29):

Attributes	Description
Gate*: GateId	There MAY be zero or more Gates. Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow (i.e, the Gateway is acting as a Decision), then there MUST be at least two Gates.
OutgoingSequenceFlow: SequenceFlowId	Each Gate MUST have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to Expression and MUST have a valid ConditionExpression. The ConditionExpression MUST be unique for all the Gates within the Gateway. If there is only one Gate (i.e., the Gateway is acting only as a Merge), then Sequence Flow MUST have its Condition attribute set to None.
Assign*: Expression	Zero or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.
DefaultGate?: ObjectId	A Default Gate MAY be specified.
OutgoingSequenceFlow: SequenceFlowId	If there is a DefaultGate, the it MUST have an associated Sequence Flow. The Sequence Flow SHALL have the Default Indicator (see Figure 23). The Sequence Flow MUST have its Condition attribute set to Default.
Assign*: Expression	Zero or more assignments MAY be made for the DefaultGate. The Assignment SHALL be performed when the DefaultGate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.

Table 32 Inclusive Gateway Attributes

Sequence Flow Connections

This section extends the basic Gateway Sequence Flow connection rules as defined in the section entitled “Common Gateway Sequence Flow Connections” on page 84. Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

To define the inclusive nature of this Gateway’s behavior for converging Sequence Flow:

1.Inclusive Gateways may be updated to include a DefaultGate attribute. This is currently an Open Issue.

- If there are multiple incoming Sequence Flows, one or more of them will be used to continue the flow of the Process. That is,
  - Process flow SHALL continue when the signals (Tokens) arrive from all of the incoming Sequence Flow that are expecting a signal based on the upstream structure of the Process (e.g., an upstream Inclusive Decision).
    - Some of the incoming Sequence Flow will not have signals and the pattern of which Sequence Flow will have signals may change for different instantiations of the Process.

Note: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source. They will be considered as part of a different set of Sequence Flow from those Sequence Flow that have a source that is an upstream activity.

To define the inclusive nature of this Gateway's behavior for diverging Sequence Flow:

- One or more Gates SHALL be selected during performance of the Process.
  - The Gates SHALL be chosen based on the Condition expression that is defined for the Sequence Flow associated with the Gates.
    - The Condition associated with all Gates SHALL be evaluated.
    - If a Condition is evaluated as "TRUE," then that Gate SHALL be chosen, independent of what other Gates have or have not been chosen.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- Figure 22 and Figure 23 were updated to show three conditional Sequence Flow, one of which has a default conition.
- The DefaultGate attribute, with supporting attributes, was added to the set of Inclusive Gateway Attributes.

4.4.4 Complex Gateways

BPMN includes a Complex Gateway to handle situations that are not easily handled through the other types of Gateways. Complex Gateways can also be used to combine a set of linked simple Gateways into a single, more compact situation. Modelers can provide complex expressions that determine the merging and/or splitting behavior of the Gateway.

- The Complex Gateway MUST use a marker that is in the shape of an asterisk and is placed within the Gateway diamond (see Figure 25) to distinguish it from other Gateways.

When the Gateway is used as a Decision (see Figure 25), then there will be an expression that will determine which of the outgoing Sequence Flow will be chosen for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, an expression may evaluate Process data and then select different sets of outgoing Sequence Flow, based on the results of the evaluation. However, The expression should be designed so that at least one of the outgoing Sequence Flow will be chosen.

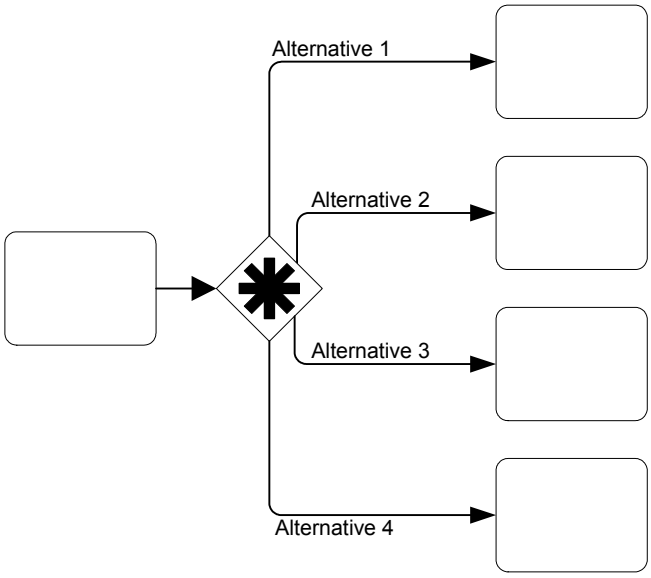

Figure 25 A Complex Decision (Gateway)

When the Gateway is used as a Merge (see Figure 26), then there will be an expression that will determine which of the incoming Sequence Flow will be required for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, an expression may specify that any 3 out of 5 incoming Tokens will continue the Process. Another example would be an expression that specifies that a Token is required from Sequence Flow “a” and that a Token from either Sequence Flow “b” or “c” is acceptable. However, the expression should be designed so that the Process is not stalled at that location.

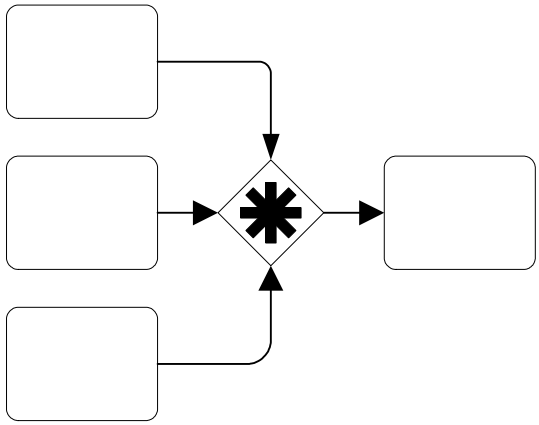

Figure 26 A Complex Merge (Gateway)

Attributes

The following table displays the attributes for a Complex Gateway. These attributes only apply if the GatewayType attribute is set to Complex. The following attributes extend the set of common Gateway attributes (see Table 29):

Attributes	Description
Gate*: GateId	There MAY be zero or more Gates. Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates.
OutgoingSequenceFlow: SequenceFlowId	Each Gate MUST have an associated Sequence Flow. Each Gate MUST have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to None. If there is only one Gate (i.e., the Gateway is acting only as a Merge), then Sequence Flow MUST have its Condition attribute set to None.
Assign*: Expression	Zero or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.
IncomingCondition?: Expression	If there are Multiple incoming Sequence Flow, an IncomingCondition expression MUST be set by the modeler. This will consist of an expression that can reference Sequence Flow names and or Process Properties (Data).
OutgoingCondition?: Expression	If there are Multiple outgoing Sequence Flow, an OutgoingCondition expression MUST be set by the modeler. This will consist of an expression that can reference (outgoing) Sequence Flow Ids and or Process Properties (Data).

Table 33 Complex Gateway Attributes

Sequence Flow Connections

This section extends the basic Gateway Sequence Flow connection rules as defined in the section entitled “Common Gateway Sequence Flow Connections” on page 84. Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

To define the complex nature of this Gateway’s behavior for converging Sequence Flow:

- If there are multiple incoming Sequence Flows, one or more of them will be used to continue the flow of the Process. The exact combination of incoming Sequence Flows will be determined by the Gateway’s IncomingCondition expression.
  - Process flow SHALL continue when the appropriate number of signals (Tokens) arrives from appropriate incoming Sequence Flows.

- Signals from other Sequence Flow within that set MAY arrive, but they SHALL NOT be used to continue the flow of the Process.

Note: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source. They will be considered as part of a different set of Sequence Flow from those Sequence Flow that have a source that is an upstream activity.

To define the inclusive nature of this Gateway's behavior for diverging Sequence Flow:

- One or more Gates SHALL be selected during performance of the Process.
  - The Gates SHALL be chosen based on the Gateway's OutgoingCondition expression.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

-

4.4.5 Parallel Gateways (AND)

Parallel Gateways provide a mechanism to synchronize parallel flow and to create parallel flow. These Gateways are not required to create parallel flow, but they can be used to clarify the behavior of complex situations where a string of Gateways are used and parallel flow is required. In addition, some modelers may wish to create a "best practice" where Parallel Gateways are always used for creating parallel paths. This practice will create an extra modeling element where one is not required, but will provide a balanced approach where forking and joining elements can be paired up.

- The Parallel Gateway MUST use a marker that is in the shape of an plus sign and is placed within the Gateway diamond (see Figure 27) to distinguish it from other Gateways.

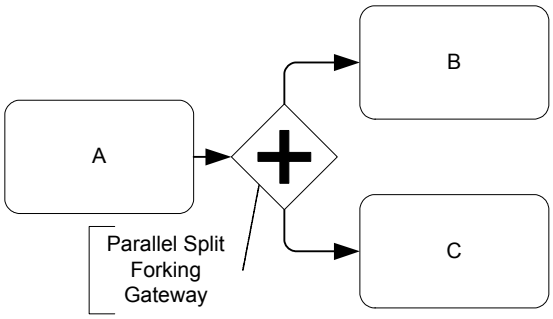

Figure 27 A Parallel Gateway

Parallel Gateways are required for synchronizing parallel flow. Synchronization

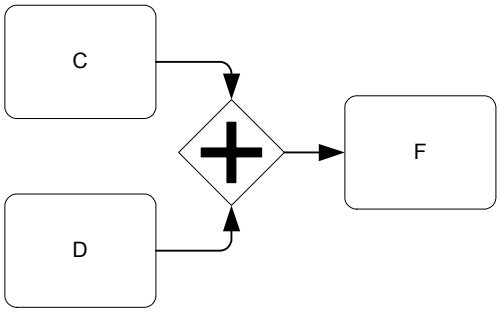

Figure 28 Joining – the joining of parallel paths

Attributes

The following table displays the attributes for a Parallel Gateway. These attributes only apply if the GatewayType attribute is set to AND (Parallel). The following attributes extend the set of common Gateway attributes (see Table 29):

Attributes	Description
Gate*: GateId	There MAY be zero or more Gates. Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow (i.e, the Gateway is acting as a fork), then there MUST be at least two Gates.
OutgoingSequenceFlow: SequenceFlowId	Each Gate MUST have an associated Sequence Flow. The Sequence Flow MUST have its Condition attribute set to None.
Assign*: Expression	Zero or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment will be in the following format: To = From. Both sides of the Assignment are defined separately as defined in the section entitled “Assignment” on page 261. The type for each side of the Assignment MUST match.

Table 34 Parallel Gateway Attributes

Sequence Flow Connections

This section extends the basic Gateway Sequence Flow connection rules as defined in the section entitled “Common Gateway Sequence Flow Connections” on page 84. Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

To define the parallel nature of this Gateway’s behavior for converging Sequence Flow:

- If there are multiple incoming Sequence Flows, all of them will be used to continue the flow of the Process--the flow will be synchronized. That is,
  - Process flow SHALL continue when a signal (a Token) has arrived from all of a set of Sequence Flows (i.e., the process will wait for all signals to arrive before it can continue).

Note: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source. They will be considered as part of a different set of Sequence Flow from those Sequence Flow that have a source that is an upstream activity.

To define the parallel nature of this Gateway's behavior for diverging Sequence Flow:

- All Gates SHALL be selected during performance of the Process.

4.5 Pools and Lanes

BPMN has a larger scope than BPEL4WS, and this scope is expressed in different dimensions. The dimension discussed in this section has to with defining business processes in a collaborative B2B environment. BPMN uses the concept known as "swimlanes" to help partition and/organize activities.

BPEL4WS is focused on a specific private process that is internal to a given Participant (i.e., a company or organization). BPEL4WS also can define an abstract process, but from the point of view of a single participant. It is possible that a BPMN Diagram may depict more than one private process, as well as the processes that show the collaboration between private processes or Participants. If so, then each private business process will be considered as being performed by different Participants. Graphically, each Participant will be partitioned; that is, will be contained within a rectangular box call a "Pool." Pools can have sub-swimlanes that are called, simply, "Lanes."

The section entitled "Uses of BPMN" on page 24 describes the uses of BPMN for modeling private processes and the interactions of processes in B2B scenarios. Pools and Lanes are designed to support these uses of BPMN.

4.5.1 Pool

A Pool (also referred to as a "swimlane") is a graphical container for partitioning a set of activities from other Pools, when modeling business-to-business situations.

- A Pool is a square-cornered rectangle that MUST be drawn with a solid single black line (as seen in Figure 29).
  - The use of text, color, size, and lines for a Pool MUST follow the rules defined in section 3.3 on page 40.

Figure 29 A Pool

To help with the clarity of the Diagram, A Pool will extend the entire length of the Diagram, either horizontally or vertically. However, there is no specific restriction to the size and/or positioning of a Pool. Modelers and modeling tools can use Pools (and Lanes) in a flexible

manner in the interest of conserving the “real estate” of a Diagram on a screen or a printed page.

A Pool acts as the container for the Sequence Flow between activities. The Sequence Flow can cross the boundaries between Lanes of a Pool, but cannot cross the boundaries of a Pool. The interaction between Pools, e.g., in a B2B context, is shown through Message Flows.

Another aspect of Pools is whether or not there is any activity detailed within the Pool. Thus, a given Pool may be shown as a “White Box,” with all details exposed, or as a “Black Box,” with all details hidden. No Sequence Flow is associated with a “Black Box” Pool, but Message Flows can attach to its boundaries (see Figure 30).

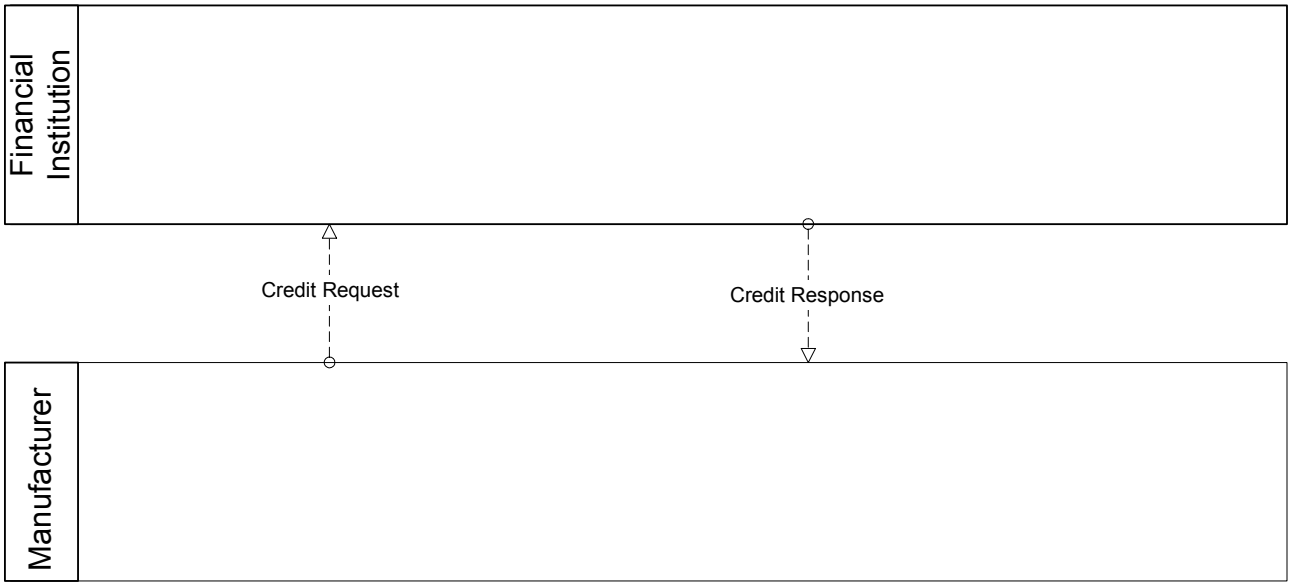

Figure 30 Message Flow connecting to the boundaries of two Pools

For a “White Box” Pool, the activities within are organized by Sequence Flows. Message Flows can cross the Pool boundary to attach to the appropriate activity (see Figure 31).

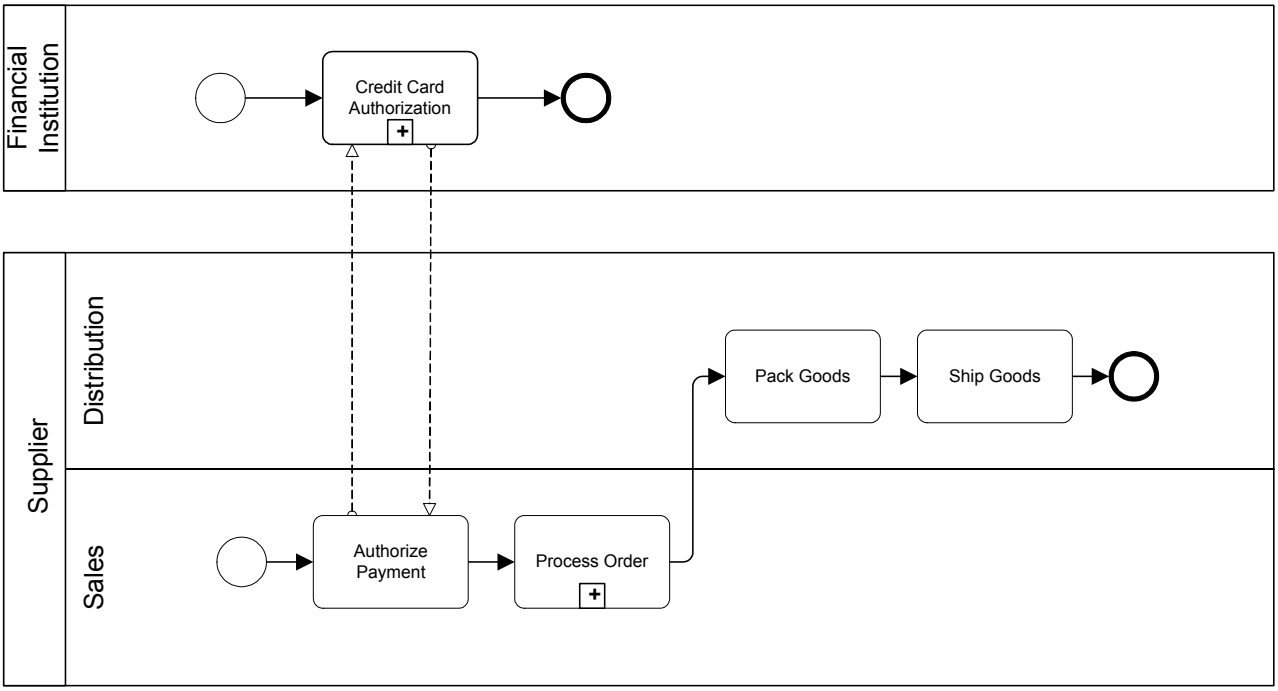

Figure 31 Message Flow connecting to flow objects within two Pools

All BPDs contain at least one Pool. In most cases, a BPD that consists of a single Pool will only display the activities of the Process and not display the boundaries of the Pool. Furthermore, many BPDs may show the “main” Pool without boundaries. That is, the activities that represent the work performed from the point of view of the modeler or the modeler’s organization are considered “internal” activities and may not be surrounded by the boundaries of a Pool, while the other Pools in the Diagram will have their boundary. (see Figure 32)

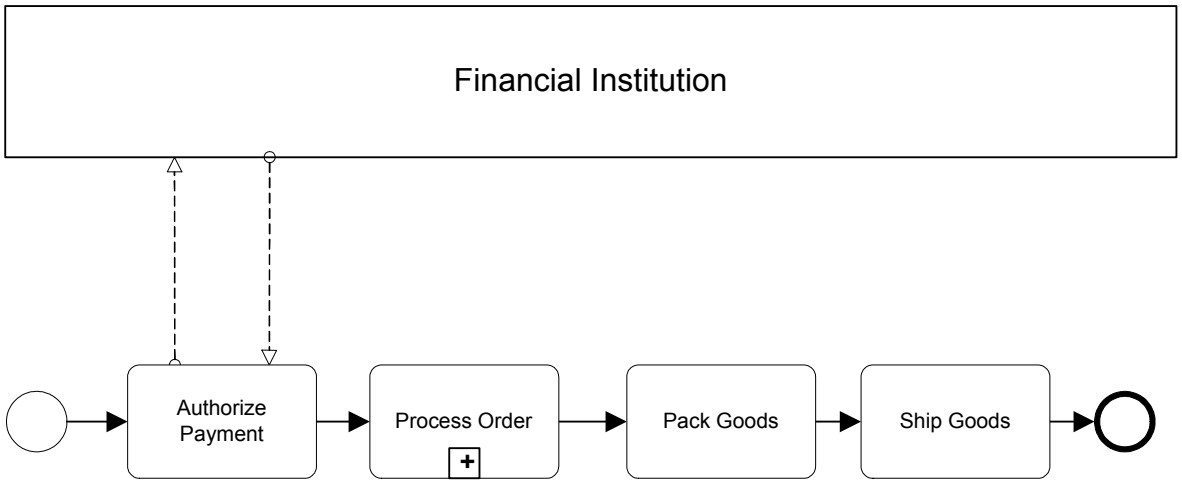

Figure 32 Main (Internal) Pool without boundaries

Attributes

The following table displays the identified attributes of a Pool (Note that this is the complete set and it does not extend the set of common object attributes):

Attributes	Description
Id: ObjectId	This is a unique Id that identifies the Pool from other objects within the Diagram.
Name: String	Name is an attribute that is text description of the Pool. If the Pool is the only one in the Diagram, it will share the name of the Diagram.
Process?: Process	The Process attribute defines the Process that is contained within the Pool. Each Pool MAY have a Process. The attributes for a Process can be found in the section entitled “Business Process” on page 43.
Role?: String	A modeler MAY define a Role for the Pool. A Role is a descriptive label for the business relationships between Entities in a Diagram. Examples of Roles are “Buyer,” “Supplier,” etc. This attribute is optional. However, if an Entity is not specified (see row below), then a Role MUST be entered.
Entity?: Entity	The modeler MAY define an Entity. identifies the point-of-view of the Diagram. If the PoolType is Collaboration, then the Entity MAY be defined as mixed. This attribute is optional. However, if a Role is not specified (see row above), then a Role MUST be entered. The attributes for an Entity can be found in the section entitled “Entity” on page 260.
Lane+: Lane	There can be one or more Lanes within a Pool. If there is only one Lane, then that Lane shares the name of the Pool and only the Pool name is displayed. If there is more than one Lane, then each Lane has to have its own name and all names are displayed.The attributes for a Lane can be found in the section entitled “Lane” on page 107.
BoundaryVisible: Boolean: True	This attribute defines if the rectangular boundary for the Pool is visible. Only one Pool in the Diagram MAY have the attribute set to False.
Category*: String	A modeler MAY add one or more defined Categories that can be used for purposes such as reporting and analysis.
Documentation?: String	The modeler can add optional text documentation about the Pool.

Table 35 Pool Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Pool was removed.
- The Process, Role, Category, and Documentation attributes were added to the set of Pool attributes.
- The Id attribute, within the set of Pool attributes, was change to be of type ObjectId.
- The Owner attributes, within the set of Pool attributes, was renamed to be Entity and of type Entity. Also, the attribute was changed from optional to mandatory.
- The Lane attribute, within the set of Pool attributes, was change to be of type Lane.

4.5.2 Lane

A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally (see Figure 33). Text associated with the Lane (e.g., its name and/or any attribute) can be placed inside the shape, in any direction or location, depending on the preference of the modeler or modeling tool vendor. Our examples place the name as a banner on the left side (for horizontal Pools) or at the top (for vertical Pools) on the other side of the line that separates the Pool name, however, this is not a requirement.

Figure 33 Two Lanes in a Pool

Lanes are used to organize and categorize activities within a Pool. The meaning of the Lanes is up to the modeler. BPMN does not specify the usage of Lanes. Lanes are often used for such things as internal roles (e.g., Manager, Associate), systems (e.g., an enterprise application), an internal department (e.g., shipping, finance), etc. In addition, Lanes can be nested or defined in a matrix. For example, there could be an outer set of Lanes for company departments and then an inner set of Lanes for roles within each department.

Attributes

The following table displays the identified attributes of a Lane ((Note that this is the complete set and it does not extend the set of common object attributes):

Attributes	Description
Id: ObjectId	This is a unique Id that identifies the Lane from other objects within the Diagram.
Name: String	Name is an attribute that is text description of the Lane. If the Lane is the only one in the Pool, it will share the name of the Pool.
ParentPool: Pool	The Parent Pool MUST be specified. There can be only one Parent. The attributes for a Pool can be found in the section entitled “Pool” on page 103.
ParentLane?: Lane	ParentLane is an optional attribute that is used if the Lane is nested within another Lane. Nesting can be multi-level, but only the immediate parent is specified.
Category*: String	This modeler MAY add one or more defined Categories that can be used for purposes such as reporting and analysis.
Documentation?: String	The modeler can add optional text documentation about the Lane.

Table 36 Lane Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The Id attribute, within the set of Pool attributes, were change to be of type ObjectId.
- The ParentPool attribute, within the set of Pool attributes, were change to be of type Pool.
- The ParentLane attribute, within the set of Pool attributes, were change to be of type Lane.
- The Category attribute was added to the set of Lane attributes.

4.6 Artifacts

BPMN provides modelers with the capability of showing additional information about a Process that is not directly related to the Sequence Flow or Message Flow of the Process.

At this point, BPMN provides three standard artifacts: A Data Object, a Group, and an Annotation. Additional standard Artifacts may be added to the BPMN specification in later versions. A modeler or modeling tool may extend a BDP and add new types of Artifacts to a Diagram. Any new Artifact must follow the Sequence Flow and Message Flow connection rules (listed below). Associations can be used to link Artifacts to flow objects (refer to the section entitled “Association” on page 125).

4.6.1 Common Artifact Definitions

The following sections provide definitions that a common to all artifacts.

Common Artifact Attributes

The following table displays the identified attributes of a Data Object (Note that this is the complete set and it does not extend the set of common object attributes):

Attributes	Description
ArtifactType: (DataObject | Group | Annotation)	The ArtifactType MAY be set to DataObject, Group, or Annotation. The ArtifactType list MAY be extended to include new types.
Id: ObjectId	This is a unique Id that identifies the object from other objects within the Diagram.
Pool?: Pool	A PoolName MAY be added to the object to identify its location. Artifacts, such as Annotations, can be placed outside of any of the Diagrams Pools. The attributes for a Pool can be found in the section entitled “Pool” on page 103.
Lane*: Lane	If the Pool has been specified and it has more than one Lane, then a LaneName MUST be added. There MAY be multiple Lanes listed if the Lanes are organized in matrix or overlap in a non-nested manner. The attributes for a Lane can be found in the section entitled “Lane” on page 107.
Category*: String	This modeler MAY add one or more defined Categories that can be used for purposes such as reporting and analysis.
Documentation?: String	The modeler MAY add optional text documentation about the Artifact.

Table 37 Common Artifact Attributes

Artifact Sequence Flow Connections

Refer to the section entitled “Sequence Flow Rules” on page 41 for the entire set of objects and how the may be source or targets of Sequence Flows.

- An Artifact cannot be a target for a Sequence Flow.
- An Artifact cannot be a source for a Sequence Flow.

Artifact Message Flow Connections

Refer to the section entitled “Message Flow Rules” on page 42 for the entire set of objects and how the may be source or targets of Sequence Flows.

- An Artifact cannot be a target for a Message Flow.
- An Artifact cannot be a source for a Message Flow.

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The Id attribute, within the set of Pool attributes, was change to be of type ObjectId
- The Pool, Lane, and Category attributes were added to the set of common Artifact attributes.

4.6.2 Data Object

In BPMN, a Data Object is considered an Artifact and not a flow object. They are considered an artifact because they do not have any direct affect on the Sequence Flow or Message Flow of the Process, but they do provide information about what the Process does. That is, how documents, data, and other objects are used and updated during the Process. While the name “Data Object” may imply an electronic document, they can be used to represent many different types of objects, both electronic and physical.

In general, BPMN will not standardize many modeling artifacts. These will mainly be up to modelers and modeling tool vendors to create for their own purposes. However, equivalents of the BPMN Data Object are used by Document Management oriented workflow systems and many other process modeling methodologies. Thus, this object is used enough that it is important to standardize its shape and behavior.

- A Data Object is a portrait-oriented rectangle that has its upper-right corner folded over that MUST be drawn with a solid single black line (as seen in Figure 34).
  - The use of text, color, size, and lines for a Data Object MUST follow the rules defined in section 3.3 on page 40.

Figure 34 A Data Object

As an artifact, Data Objects generally will be associated with flow objects. An Association will be used to make the connection between the Data Object and the flow object. This means that the behavior of the Process can be modeled without Data Objects for modelers who want to reduce clutter. The same Process can be modeled with Data Objects for modelers who want to include more information without changing the basic behavior of the Process.

In some cases, the Data Object will be shown being sent from one Process to another, via a Sequence Flow (see Figure 35). Data Objects will also be associated with Message Flows. They are not to be confused with the message itself, but could be though of as the “payload” or content of some messages.

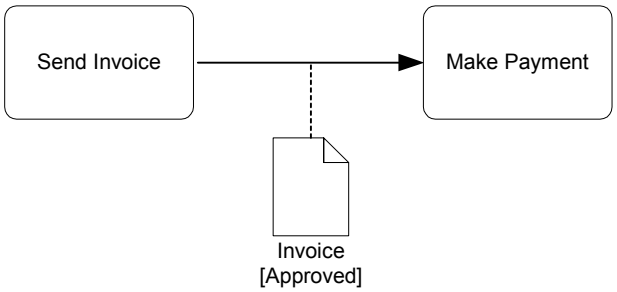

Figure 35 A Data Object associated with a Sequence Flow

In other cases, the same Data Object will be shown as being an input, then an output of a Process (see Figure 36). Directionality added to the Association will show whether the Data Object is an input or an output. Also, the state attribute of the Data Object can change to show the impact of the Process on the Data Object.

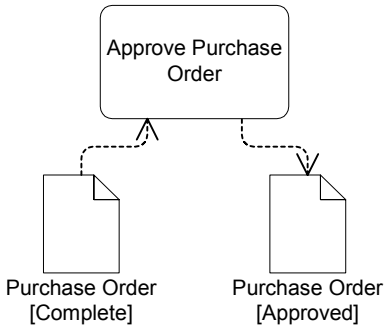

Figure 36 Data Objects shown as inputs and outputs

Attributes

The following table displays the attributes for Data Objects, and which extends the set of common Artifact attributes (see Table 37). These attributes only apply if the ArtifactType attribute is set to DataObject:

Attributes	Description
Name: String	Name is an attribute that is text description of the object.
State?: String	State is an optional attribute that indicates the impact the Process has had on the Data Object. Multiple Data Objects with the same name MAY share the same state within one Process.
Property*	Modeler-defined Properties MAY be added to a Data Object. The fully delineated name of these properties are “<process name>.<task name>.<property name>” (e.g., “Add Customer.Review Credit Report.Score”).
RequiredForStart: Boolean: True	The default value for this attribute is True. This means that the Input is required for the activity to start. If set to False, then the activity MAY start within the input, but MAY accept the input (more than once) after the activity has started.
ProducedAtCompletion: Boolean: True	The default value for this attribute is True. This means that the Output will be produced when the activity has been completed. If set to False, then the activity MAY produce the output (more than once) before it has completed.

Table 38 Data Object Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The Pool and Lane attributes were removed from the set of Data Object attributes. These two attributes were added to the set of Common Artifact attributes.
- The Name and Type attributes were removed from the set of common activity attributes. These attributes can be found in the definition of a Property, which can be found in the section entitled “Property” on page 259.
- The RequiredForStart and ProducedAtCompletion attributes were added to the set of Data Object attributes.
- The constraint about the fill of the Data Object was removed.

4.6.3 Text Annotation

Text Annotations are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.

- A Text Annotation is an open rectangle that MUST be drawn with a solid single black line (as seen in Figure 37).
  - The use of text, color, size, and lines for a Text Annotation MUST follow the rules defined in section 3.3 on page 40.

The Text Annotation object can be connected to a specific object on the Diagram with an Association (see Figure 37). Text associated with the Annotation can be placed within the bounds of the open rectangle.

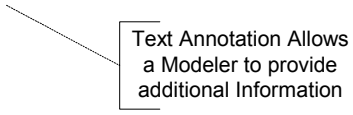

Figure 37 A Text Annotation

Text Annotations do not affect the flow of the Process and do not map to any BPEL4WS elements.

Attributes

The following table displays the attributes for Annotations, and which extends the set of common Artifact attributes (see Table 37). These attributes only apply if the ArtifactType attribute is set to Annotation:

Attributes	Description
Text: String	Text is an attribute that is text that the modeler wishes to communicate to the reader of the Diagram.

Table 39 Text Annotation Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Text Annotation was removed.

4.6.4 Group

The Group object is an artifact that provides a visual mechanism to group elements of a Process informally.

- A Group is a rounded corner rectangle that MUST be drawn with a solid dashed black line (as seen in Figure 38).
  - The use of text, color, size, and lines for a Group MUST follow the rules defined in section 3.3 on page 40.

Figure 38 A Group Artifact

As an Artifact, a Group is not an activity or any flow object, and, therefore, cannot connect to Sequence Flow or Message Flow. In addition, Groups are not constrained by restrictions of Pools and Lanes. This means that a Group can stretch across the boundaries of a Pool to

surround Diagram elements (see Figure 39), often to identify activities that exist within a distribute business-to-business transaction.

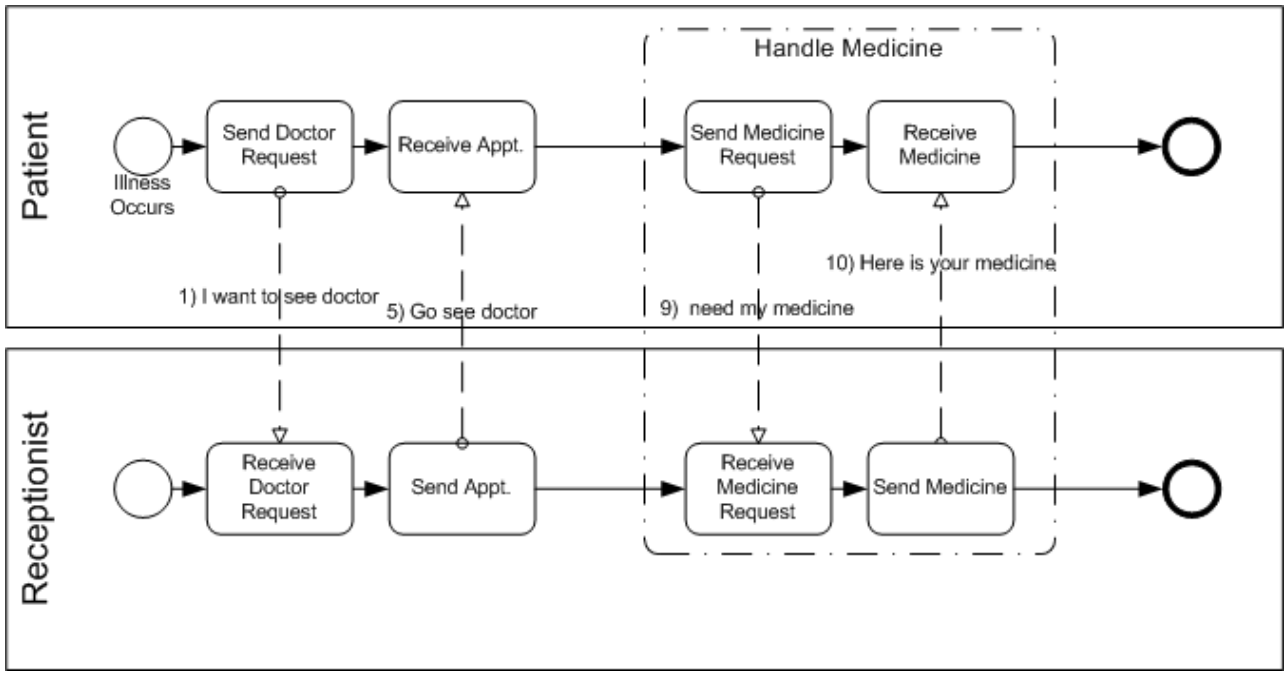

Figure 39 A Group around activities in different Pools

Groups are often used to highlight certain sections of a Diagram without adding additional constraints for performance--as a Sub-Process would. The highlighted (grouped) section of the Diagram can be separated for reporting and analysis purposes. Groups do not affect the flow of the Process and do not map to any BPEL4WS elements.

The following table displays the attributes for Groups, and which extends the set of common Artifact attributes (see Table 37). These attributes only apply if the ArtifactType attribute is set to Group:

Attributes	Description
Name?: String	Name is an optional attribute that is text description of the Group.

Table 40 Group Attributes

Changes Since 1.0 Draft Version

These are the changes since the last publicly release version:

- The constraint about the fill of the Group was removed.