Appendix B: BPMN Element Attributes and Types

This appendix provides the complete set of BPMN Element Attributes and the definition of types that support the Attributes. All the tables in this appendix also appear in Chapters 3, 4, and 5.

Business Process Diagram Attributes

The following are attributes of a Business Process Diagram:

Attributes	Description
Id: ObjectId	This is a unique Id that distinguishes the Diagram from other Diagrams.
Name: String	Name is an attribute that is text description of the Diagram.
Version?: String	This defines the Version number of the Diagram.
Author?: String	This holds the name of the author of the Diagram.
Language?: String	This holds the name of the language in which text is written. The defeault is English.
ExpressionLanguage?: String	A Language MAY be provided so that the syntax of expressions used in the Diagram can be understood.
QueryLanguage?: String	A Language MAY be provided so that the syntax of queries used in the Diagram can be understood.
CreationDate?: Date	This defines the date on which the Diagram was create (for this Version).
ModificationDate?: Date	This defines the date on which the Diagram was last modified (for this Version).
Pool+: PoolId	A BDP SHALL contain one or more Pools. The boundary of one of the Pools MAY be invisible (especially if there is only one Pool in the Diagram).
Documentation?: String	The modeler MAY add optional text documentation about the Diagram.

Table 1 Business Process Diagram Attributes

Business Process Attributes

The following are attributes of a Process, which extends the set of common object elements (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 object from other objects within the Diagram.
Name: String	Name is an attribute that is text description of the object.

Attributes	Description
ProcessType: (None | Private | Abstract | Collaboration): None	ProcessType is an attribute that provides information about which lower-level language the Pool will be mapped.By default, the ProcessType is None (or undefined). A Private ProcessType MAY be mapped to an executable BPEL4WS *process*. An Abstract ProcessType is also called the public interface of a process (or other web services) and MAY be mapped to an abstract BPEL4WS *process*. A Collaboration ProcessType will have two Lanes that represent business roles (e.g., buyer or seller) and will show the interactions between these roles. These pools MAY be mapped to languages such as ebXML or WS Choreography. However, these mappings are not provided in this version of the specification. If the Process is to be used to create a BPEL4WS document, then the attrbute MUST be set to Executable or Abstract.
Status: (None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed) : None	The Status of a Process is determined when the Process is being executed by a process engine. The Status of a Process can be used within Assignment Expressions.
GraphicalElements*: ObjectID	The GraphicalElements attribute identifies all of the objects (e.g., Events, Activities, Gateways, and Artifiacts) that are contained within the Business Process.
Assign*: Assignment	One or more assignment expressions MAY be made for the object. The Assignment SHALL be performed as defined by the AssignTime attribute (see below). 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 282.
AssignTime: (Start | End): Start	Each Assignment Expression will have AssignTime. A value of Start means that the assignment SHALL occur at the start of the Process. A value of End means that the assignment SHALL occur at the end of the Process.
Properties*: Property	Modeler-defined Properties MAY be added to a Process. These Properties are “local” to the Process. All Tasks, Sub-Process objects, and Sub-Processes that are embedded SHALL have access to these Properties. The fully delineated name of these properties are “<process name>.<property name>” (e.g., “Add Customer.Customer Name”). If a process is embedded within another Process, then the fully delineated name SHALL also be preceded by the Parent Process name for as many Parents there are until the top level Process. Further details about the definition of a Property can be found in the section entitled “Property” on page 284.
AdHoc: Boolean: False	AdHoc is a Boolean attribute, which has a default of False. This specifies whether the Process is Ad Hoc or not. The activities within an Ad Hoc Process are not controlled or sequenced in a particular order, their performance is determined by the performers of the activities. If set to True, then the Ad Hoc marker SHALL be placed at the bottom center of the Process or the Sub-Process shape for Ad Hoc Processes.

Attributes	Description
(AdHoc = True only) **AdHocOrdering ?**: (Sequential | Parallel): Parallel	If the Process is Ad Hoc (the AdHoc attribute is True), then the AdHocOrdering attribute MUST be included. This attribute defines if the activities within the Process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.
(AdHoc = True only) **AdHocCompletionCondition ?**: Expression	If the Process is Ad Hoc (the AdHoc attribute is True), then the AdHocCompletionCondition attribute MUST be included. This attribute defines the conditions when the Process will end.
SuppressJoinFailure: Boolean: False	This attribute is included for mapping to BPEL4WS. This specifies whether or not a BPEL4WS joinFailure fault will be suppressed for all activities in the BPEL4WS process.
EnableInstanceCompensation: Boolean: False	This attribute is included for mapping to BPEL4WS. It specifies whether or not a compensation can be performed after the Process has completed normally.
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 Process.

Table 2 Process Attributes

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*: Assignment	One 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 282.
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 277.
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 278.
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 3 Common Object Attributes

Events

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 3):

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

Table 4 Common Event Attributes

Start Event

The following are attributes of a Start Event, which extends the set of common Event elements (see Table 4):

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 283.
(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 284.
(Link Trigger only) **LinkId**: String	If the Trigger is a Link, then the the LinkId MUST be entered.
(Link Trigger only) ProcessRef: Process	If the Trigger is a Link, then the ProcessRef MUST be entered. The identified Process MAY be the same Process as that of the Link Event.
(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 5 Start Event Attributes

End Event

The following are attributes of a End Event, which extends the set of common Event elements (see Table 4):

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 283.
(Exception Result only) ExceptionCode: String	If the Result is an Exception, then the ExceptionCode MUST be supplied.
(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.
(Link Result only) ProcessRef: Process	If the Result is a Link, then the ProcessRef MUST be entered. The identified Process MAY be the same Process as that of the Link Event.
(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 6 End Event Attributes

Intermediate Event

The following are attributes of an Intermediate Event, which extends the set of common Event elements (see Table 4):

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.

Attributes	Description
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 283.
(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 componsation. *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.
(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 284.
(Link Trigger only) LinkId: String	If the Trigger is a Link, then the LinkId MUST be supplied.

Table 7 Intermediate Event Attributes

Activities

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 3) -- Note that Table 9 and Table 10 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 object. These Properties are only for use within the processing of the activity. The fully delineated name of these properties are "<process name>.<sub-process 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 284.
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 requried 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.
(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 requried for them to be displayed.
IORule*: Expression	The IORule attribute is an expression that defines the relationship between one InputSet and one OuputSet. That is, if the activity is instantiated with a specified InputSet, then the ouput 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 11 and Figure 14). A Task of type Receive that has its Instantiate attribute set to True MAY NOT have a Standard or MultiInstance LoopType.

Attributes	Description
AssignTime*: (Start | End): Start	Each Assignment Expression 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 8 Common Activity Attributes

Standard Loop Attributes

The following are additional attributes of a Standard Loop Activity (where the LoopType attribute is set to “Standard”), which extends the set of commaon activity attributes (see Table 8):

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 283.
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.
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 9 Standard Loop Activity Attributes

Mutli-Instance Loop Attributes

The following are additional attributes of a Multi-Instance Loop Activity (where the LoopType attribute is set to “MultiInsatnce”), which extends the set of commaon activity attributes (see Table 8):

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 283.

Attributes	Description
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 11 and Figure 14).
(Parallel MI_Ordering only) MI_FlowCondition: (None | One | All | Complex): All	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 283.

Table 10 Multi-Instance Loop Activity Attributes

Sub-Process

The following are attributes of a Sub-Process, which extends the set of common activity attributes (see Table 8):

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 Indepent SubProcessType can be found in Table 13.
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 72).

Attributes	Description
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 284. Note that Transactions that are in different Pools and are connected through Message Flow MUST have the same TransactionId.

Table 11 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 11):

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.
(AdHoc = True only) **AdHocOrdering?**: (Sequential | Parallel): Parallel	If the Embedded Sub-Process is Ad Hoc (the AdHoc attribute is True), then the AdHocOrdering attribute MUST be included. This attribute defines if the activities within the Process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.
(AdHoc = True only) **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 12 Embedded Sub-Process Attributes

Independent Sub-Process Attributes

The following are additional attributes of a Embedded Sub-Process, which extends the set of Sub-Process attributes (see Table 11):

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 Attributes” on page 257.
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).

Attributes	Description
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 13 Independent Sub-Process Attributes

Task

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

Attributes	Description
TaskType (Service | Receive | Send | User | Script | Abstract | Manual | Reference | None): Service	TaskType is an attribute that has a default of Service, but MAY be set to Send, Receive, User, Script, Abstract, 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 settings of TaskType can be found in Table 15 through Table 21.

Table 14 Task Attributes

Service Task Attributes

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 14):

Attributes	Description
InMessage: Message	A Message for the InMessage 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. A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
OutMessage: Message	A Message for the OutMessage attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet. 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 receive the messages. A Web service is the default technology.

Table 15 Service Task Attributes

Receive Task Attributes

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 14):

Attributes	Description
Message: Message	A Message for the Message attribute MUST be entered. This indicates that the Message will be recieved by the Task. 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. Multiple Tasks MAY have this attribute set to True.
Implementation: (Web Service | Otherr | Unspecified): Web Service	This attribute specifies the technology that will be used to receive the message. A Web service is the default technology.

Table 16 Receive Task Attributes

Send Task Attributes

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 14):

Attributes	Description
Message: Message	A Message for the Message attribute MUST be entered. This indicates that the Message will be sent by the Task. 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 17 Send Task Attributes

User Task Attributes

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 14):

Attributes	Description
Performer: String	One or more Performers MAY be entered. The Performer attribute defines the human resource that will be performing the 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.

Attributes	Description
InMessage: Message	A Message for the InMessage 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. A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
OutMessage: Message	A Message for the OutMessage attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet. A corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.
Implementation: (Web Service | Otherr | 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 18 User Task Attributes

Script Task Attributes

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 14):

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 19 Script Task Attributes

Manual Task Attributes

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 14):

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 20 Manual Task Attributes

Reference Task Attributes

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 14):

Attributes	Description
TaskRef: Task	The Task being referenced MUST be identified. The attributes for the Task element can be found in Table 14.

Table 21 Reference Task Attributes

Gateways

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 3):

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 22 Common Gateway Attributes

Exclusive Gateways (XOR)

Data-Based

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 22):

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 is displayed 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, the marker is not displayed.
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*: Assignment	One 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 282.
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. The Sequence Flow MUST have its Condition attribute set to Default.
Assign*: Assignment	One 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 282.

Table 23 Data-Based Exclusive Gateway Attributes

Event-Based

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 22):

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.
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*: Assignment	One 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 282.

Table 24 Event-Based Exclusive Gateway Attributes

Inclusive Gateways (OR)

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 22):

Attributes	Description
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 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*: Assignment	One 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 282.
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. The Sequence Flow MUST have its Condition attribute set to Default.
Assign*: Assignment	One 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 282.

Table 25 Inclusive Gateway Attributes

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

Complex Gateways

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 22):

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*: Assignment	One 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 282.
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 26 Complex Gateway Attributes

Parallel Gateways (AND)

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 22):

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*: Assignment	One 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 282.

Table 27 Parallel Gateway Attributes

Pool

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 “Process Attributes” on page 259.
Role?: Role	This defines the role that a particular Entity or Process will play in a Diagram that includes collaboration.
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 283.
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 100.
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 28 Pool Attributes

Lane

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 277.
Lane*: Lane	If the Pool 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.
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 29 Common Artifact Attributes

Data Object

The following table displays the attributes for Data Objects, and which extends the set of common Artifact attributes (see Table 29). 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 actvity 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 actvity MAY produce the output (more than once) before it has completed.

Table 30 Data Object Attributes

Text Annotation

The following table displays the attributes for Annotations, and which extends the set of common Artifact attributes (see Table 29). 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 31 Text Annotation Attributes

Group

The following table displays the attributes for Groups, and which extends the set of common Artifact attributes (see Table 29). 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 32 Group Attributes

Graphical Connecting Objects

Sequence Flow

The following are attributes of a Sequence Flow (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 object from other objects within the Diagram.
Name: String	Name is an attribute that is text description of the object.
Source: ObjectId	Source is an attribute that identifies which flow object the Sequence Flow is connected *from*; i.e., the Sequence Flow is an outgoing flow from that object. The Source MUST be from the set of the following flow objects: Start Event, Intermediate Event, End Event, Task, Sub-Process, and Decision.
Target: ObjectId	Target is an attribute that identifies which flow object the Sequence Flow is connected *to*; i.e., the Sequence Flow is an incoming flow to that object. The Target MUST be from the set of the following flow objects: Start Event, Intermediate Event, End Event, Task, Sub-Process, and Decision.
ConditionType: (None | Expression | Default): None	By default, the **ConditionType** of a Sequence Flow is None. This means that there is no evaluation at runtime to determine whether or not the Sequence Flow will be used. Once a Token is ready to traverse the Sequence Flow (i.e., the Source is an activity that has completed), then the Token will do so. The normal, uncontrolled use of Sequence Flow, in a sequence of activities, will have a None **ConditionType** (see Figure 51). A None **ConditionType** SHALL NOT be used if the Source of the Sequence Flow is an Exclusive Data-Based or Inclusive Gateway. The **ConditionType** attribute MAY be set to Expression if the Source of the Sequence Flow is a Task, a Sub-Process, or a Gateway of type Exclusive-Data-Based or Inclusive. If the **ConditionType** attribute is set to Expression, then a condition marker SHALL be added to the line if the Sequence Flow is outgoing from an activity (see Figure 41). However, a condition indicator SHALL NOT be added to the line if the Sequence Flow is outgoing from a Gateway. An Expression **ConditionType** SHALL NOT be used if the Source of the Sequence Flow is an Event-Based Exclusive Gateway, a Complex Gateway, a Parallel Gateway, a Start Event, or an Intermediate Event. In addition, an Expression **ConditionType** SHALL NOT be used if the Sequence Flow is associated with the Default Gate of a Gateway. The **ConditionType** attribute MAY be set to Default only if the Source of the Sequence Flow is an activity or an Exclusive Data-Based Gateway. If the **ConditionType** is Default, then the Default marker SHALL be displayed (see Figure 42).

Attributes	Description
(ConditionType is set to Expression only) ConditionExpression: Expression	If the **ConditionType** attribute is set to Expression, then the ConditionExpression attribute MUST be defined as a valid expression. The expression will be evaluated at runtime. If the result of the evaluation is TRUE, then a Token will be generated and will traverse the Sequence--Subject to any constraints imposed by a Source that is a Gateway.
Quantity: Integer: 1	The default value is 1. The value MAY NOT be less than 1. This attribute defines the number of Tokens that will be generated down the Sequence Flow.
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 Sequence Flow.

Table 33 Sequence Flow Attributes

Message Flow

The following table displays the identified attributes of a Message Flow (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 Message Flow from other objects within the Diagram.
Name ?: String	Name is an optional attribute that is text description of the Message Flow.
Message ?: Message	Message is an optional attribute that identifies the Message that is being sent. The attributes of a Message can be found section entitled “Message” on page 283.
Source: ObjectId	Source is an attribute that identifies the object the Message Flow is connected *from*; i.e., the Message Flow is an outgoing flow from that object. The Message Flow MAY originate from the boundary of the Pool or an object within the Pool. If the source is an object within the Pool, then the ObjectName MUST identify the Pool and the Object.
Target: ObjectId	Target is an attribute that identifies the object the Message Flow is connected *to*; i.e., the Message Flow is an incoming flow to that object. The Message Flow MAY target the boundary of the Pool or an object within the Pool. If the target is an object within the Pool, then the ObjectName MUST identify the Pool and the Object.
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 Message Flow.

Table 34 Message Flow Attributes

Association

The following table displays the identified attributes of a Association (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 Association from other objects within the Diagram.
Name ?: String	Name is an optional attribute that is text description of the Association.
Source: ObjectId	Source is an attribute that identifies which object the Association is connected *from*. The set of objects that an Association MAY connect to are: Pool, Lane, all Events, Task, Sub-Process, Gateway, Sequence Flow, and Message Flow.
Target: ObjectId	Target is an attribute that identifies which object the Association is connected *to*. Associations MUST only connect to Artifacts or Compensation Activities.
Direction (None | To | From | Both): None	Direction is an attribute that defines whether or not the Association shows any directionality with an arrowhead. The default is None (no arrowhead). A value of To means that the arrowhead SHALL be at the Source object. A value of From means that the arrowhead SHALL be at the Target artifact. A value of Both means that there SHALL be an arrowhead at both ends of the Association line.
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 Association.

Table 35 Association Attributes

Supporting Types

Assignment

The following are attributes of an Assignment, which is used in the definition of attributes for Process, Activites, Events, Gateways, and Gates:

Attributes	Description
To: Property	The target for the Assignment MUST be a Property of the Process or the activity itself.
From: Expression	The Expression MUST be made up of a combination of Values, Properties, and Attributes, which are separated by operators such as add or multiply. The expression language is defined in the ExpressionLanguage attribute of the Business Process Diagram (see the section entitled “Business Process Diagram Attributes” on page 257)

Table 36 Assignment Attributes

Entity

The following are attributes of an Entity, which is used in the definition of attributes for a Pool and a Message:

Attributes	Description
Name: String	Name is an attribute that is text description of the Entity.
Role*: Role	An Entity MAY have multiple Roles within a Business Process.

Table 37 Entity Attributes

Expression

The following are attributes of an Expression, which is used in the definition of attributes for Start Event, Intermediate Event, Activity, Compelx Gateway, and Sequence Flow:

Attributes	Description
Expression: String	An Expression MUST be entered to provide a mathematical expression to be either tested as True or False or to be evaluated to update the value of Properties (e.g., assignment).

Table 38 Expression Attributes

Message

The following are attributes of a Message, which is used in the definition of attributes for a Start Event, End Event, Intermediate Event, Task, and Message Flow:

Attributes	Description
Name: String	Name is an attribute that is text description of the Message.
Properties*: Property	Mutliple Properties MAY entered for the Message. The attributes of a Property can be found section entitled “Property” on page 284.
From: Entity	This defines the source of the Message
To: Entity	This defines the target of the Message.

Table 39 Message Attributes

ObjectId

The following are attributes of an ObjectId, which is used in the definition of attributes for all graphical elemens:

Attributes	Description
ObjectId: String	The ObjectId attribute provides a unique identifier for all objects on a diagram. That is, each object MUST have a different value for the ObjectId attribute.

Table 40 Property Attributes

Property

The following are attributes of a Property, which is used in the definition of attributes for a Process and common activity attributes:

Attributes	Description
Name: String	Each Property has a Name (e.g., name="Customer Name").
Type: String	Each Property has a Type (e.g., type="String"). A Property may be of type Set, which allows child Properties.
(Type = "Set" only) Correlation?: Boolean: False	If the **ConditionType** attribute is set to Expression, then the ConditionExpression attribute MUST be defined. Otherwise, it is not valid. This attribute is included for mapping to BPEL4WS. The Property will map to a *correlationSet* and the child Properties will be *properties* of that *correlationSet*.

Table 41 Property Attributes

Role

The following are attributes of a Role, which is used in the definition of attributes for Message, Entity, and Pool:

Attributes	Description
Name: String	Name is an attribute that is text description of the Role.

Table 42 Rule Attributes

Rule

The following are attributes of a Rule, which is used in the definition of attributes for Start Event and Intermediate Event:

Attributes	Description
Name: String	Name is an attribute that is text description of the Entity.
RuleExpression ?: Expression	A RuleExpression May be entered. In some cases the Rule itself will be stored and maintained in a separate application (e.g., a Rules Engine). The attributes of an Expression can be found section entitled "Expression" on page 283.

Table 43 Rule Attributes

Transaction

The following are attributes of a Transaction, which is used in the definition of attributes for a Sub-Process:

Attributes	Description
TransactionID: String	The TransactionID attribute provides an identifier for the Transactions used within a diagram.
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.

Attributes	Description
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 44 Transaction Attributes

Types

The following are attributes of the set of Types that are used to define attributes for Start Event, End Event, Intermediate Event, Activity, Compelx Gateway, and Sequence Flow:

Attributes	Description

Table 45 Types Attributes

Web Service

The following are attributes of an Web Service, which is used in the definition of attributes for Message Start Event, Message Intermediate Event, Message End Event, Receive Task, Send Task, Service Task, and User Task:

Attributes	Description
Entity: Entity	An Entity for the Web Service MUST be entered. Note, this will map to the BPEL4WS *partnerLink*.
Interface: String	(aka portType) An Interface for the Web Service MUST be entered.
Operation: String	One or more Operations for the Web Service MUST be entered.

Table 46 Web Service Attributes