A formalisation of BPMN in Description Logics

Chiara Ghidini       Marco Rospocher       Luciano Serafini

FBK-irst, Via Sommarive 18 Povo, 38050, Trento, Italy
{ghidini, rospocher, serafini}@fbk.eu

Abstract

In this paper we present a textual description, in terms of Description Logics, of the BPMN Ontology (available for download at dkm.fbk.eu/index.php/Resources), which provides a clear semantic formalisation of the structural components of the Business Process Modelling Notation (BPMN), based on the latest stable BPMN specifications from OMG [BPMN Version 1.1 - January 2008]. The development of the ontology was guided by the description of the complete set of BPMN Element Attributes and Types contained in Annex B of the BPMN specifications.

1 Introduction

The ontology OntoBPMN.owl\(^1\) provides a clear semantic formalisation of the structural components of BPMN, based on the latest stable BPMN specifications from OMG [1]. The development of the ontology was guided by the description of the complete set of BPMN Element Attributes and Types contained in Annex B of the document cited above. The ontology currently consists of 95 Classes and 439 class axioms, 108 Object Properties and 18 Object Property Axioms, and 70 Data Properties; it has the expressiveness of $\text{ALCHOIN}(D)$. In this paper we provide a textual description of its Description Logic version.

The core component of OntoBPMN.owl is the set of BPMN Elements, divided in two disjoint classes Graphical Element and Supporting Element. Graphical Element contains the main elements used to describe Business Processes, namely Flow Object, Connecting Object, Swimline, and Artifact, then further specified in terms of sub-classes. For instance Connecting Object is then composed of the disjoint (sub-)classes Sequence Flow, Message Flow, and Association, and do on. Supporting Element instead contains 16 additional types of elements, and few additional subclasses, mainly used to specify the attributes of Graphical Objects. To provide an example, the supporting element input_set is used to define an attribute of the graphical object Activity which describes the data requirements for input of the activity.

Note that, while the taxonomy of concepts defines an important part of OntoBPMN.owl, it constitutes only part of the OWL version of BPMN: in fact, it also specifies the rich set of elements’ attributes, and the properties which describe how to use these elements to compose the business process diagrams. As an example, BPMN specifies that Connecting Object has two attributes (SourceRef, TargetRef) which point to the two corresponding Graphical Elements connected by it. As another example, BPMN not only introduces the notion of Start Event as a particular, optional, Event, but also specifies that “The Condition attribute for all outgoing Sequence Flow [from a Start Event] MUST be set to None”. Thus the BPMN specification tells that the graphical element Start Event is a sub-class of Event. Moreover it tells us that if an object of kind Start Event is connected to an object of kind Sequence Flow, then this Sequence Flow object must have a Condition attribute whose value is “None”. As a consequence of our effort towards the modelling of properties, OntoBPMN.owl contains, at the current state more than 400 class axioms, which describe a wide set of properties of the BPMN elements.

While our aim is to formalise the widest set of BPMN specifications, the OntoBPMN.owl ontology does not contain a description of all the properties documented in [1]. First of all, because we have chosen

---

\(^1\)Available for download at dkm.fbk.eu/index.php/Resources.
not to formalise properties which refer to the execution behaviour of the process. Second, because of well
known limitations in the expressiveness of the OWL language. In this specific case, most of the properties
of BPMN that are not expressible in OWL, and therefore not included in OntoBPMN.owl, concern: (i)
attributes' default values, and (ii) all the properties that, translated in first order logic, require more than
two variables. Prototypical examples of this kind of properties are the ones which refer to the uniqueness,
or equality, of objects: for instance the properties which specify that “two objects cannot have the same
object identifier” or that “all outgoing sequence flows connected to an inclusive gateway must have the same
conditional expression attached”.

2 The BPMN Ontology

Class: BUSINESS_PROCESS_DIAGRAM
Label: Business Process Diagram
Description: Gather the set of attributes of a Business Process Diagram
AX_1 BUSINESS_PROCESS_DIAGRAM ⊑ (= 1) has_business_process_diagram_id

Property: has_business_process_diagram_id
Label: Id
Description: This is a unique Id that identifies the object from other objects within the business_process_diagram.
AX_2 has_business_process_diagram_id has range OBJECT
AX_3 has_business_process_diagram_id has domain BUSINESS_PROCESS_DIAGRAM
AX_4 BUSINESS_PROCESS_DIAGRAM ⊑ (= 1) has_business_process_diagram_name

Property: has_business_process_diagram_name
Label: Name
Description: Name is an attribute that is text description of the Diagram.
AX_5 has_business_process_diagram_name has range xsd:string
AX_6 has_business_process_diagram_name has domain BUSINESS_PROCESS_DIAGRAM
AX_7 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_version

Property: has_business_process_diagram_version
Label: Version
Description: This defines the Version number of the Diagram.
AX_8 has_business_process_diagram_version has range xsd:string
AX_9 has_business_process_diagram_version has domain BUSINESS_PROCESS_DIAGRAM
AX_10 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_author

Property: has_business_process_diagram_author
Label: Author
Description: This holds the name of the author of the Diagram.
AX_11 has_business_process_diagram_author has range xsd:string
AX_12 has_business_process_diagram_author has domain BUSINESS_PROCESS_DIAGRAM
AX_13 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_language
Property: has_business_process_diagram_language
Label: Language
Description: This holds the name of the language in which text is written. The default is English.
AX_14 has_business_process_diagram_language has range xsd:string
AX_15 has_business_process_diagram_language has domain BUSINESS_PROCESS_DIAGRAM
AX_16 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_query_language

Property: has_business_process_diagram_query_language
Label: Query Language
Description: A Language MAY be provided so that the syntax of queries used in the Diagram can be understood.
AX_17 has_business_process_diagram_query_language has range xsd:string
AX_18 has_business_process_diagram_query_language has domain BUSINESS_PROCESS_DIAGRAM
AX_19 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_creation_date

Property: has_business_process_diagram_creation_date
Label: Creation Date
Description: This defines the date on which the Diagram was create (for this Version).
AX_20 has_business_process_diagram_creation_date has range xsd:date
AX_21 has_business_process_diagram_creation_date has domain BUSINESS_PROCESS_DIAGRAM
AX_22 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_modification_date

Property: has_business_process_diagram_modification_date
Label: Modification Date
Description: This defines the date on which the Diagram was last modified (for this Version).
AX_23 has_business_process_diagram_modification_date has range xsd:date
AX_24 has_business_process_diagram_modification_date has domain BUSINESS_PROCESS_DIAGRAM
AX_25 BUSINESS_PROCESS_DIAGRAM ⊑ (≤ 1) has_business_process_diagram_pools

Property: has_business_process_diagram_pools
Label: Pools
Description: A BPD 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). Refer to ”Pool” on page 75 for more information about Pools.
AX_26 has_business_process_diagram_pools has range POOL
AX_27 has_business_process_diagram_pools has domain BUSINESS_PROCESS_DIAGRAM
AX_28 BUSINESS_PROCESS_DIAGRAM ⊑ (≥ 1) has_business_process_diagram_documentation

Property: has_business_process_diagram_documentation
Label: Documentation
Description: The modeler MAY add optional text documentation about the Diagram.
AX_29 has_business_process_diagram_documentation has range xsd:string
AX_30 has_business_process_diagram_documentation has domain BUSINESS_PROCESS_DIAGRAM
Class: BPMN_ELEMENT

Label: BPMN element

Description: Base element

\[ AX_{31} \text{BPMN ELEMENT} \equiv \text{GRAPHICAL_ELEMENT} \sqcup \text{SUPPORTING_ELEMENT} \]
\[ AX_{32} \text{GRAPHICAL ELEMENT} \sqsubseteq \neg \text{SUPPORTING ELEMENT} \]
\[ AX_{33} \text{BPMN ELEMENT} \sqsubseteq (= 1)\text{has_BPMN_element_id} \]

Property: has_BPMN_element_id

Label: Id

Description: This is a unique Id that identifies the object from other objects within the Diagram.

\[ AX_{34} \text{has_BPMN_element_id} \text{has range OBJECT} \]
\[ AX_{35} \text{has_BPMN_element_id} \text{has domain BPMN_ELEMENT} \]

Property: has_BPMN_element_category

Label: Category

Description: The modeler MAY add one or more defined Categories, which have user-defined semantics, and that can be used for purposes such as reporting and analysis. The details of Categories is defined in Category on page 269.

\[ AX_{36} \text{has_BPMN_element_category} \text{has range CATEGORY} \]
\[ AX_{37} \text{has_BPMN_element_category} \text{has domain BPMN_ELEMENT} \]
\[ AX_{38} \text{BPMN_ELEMENT} \sqsubseteq (\geq 1)\text{has_BPMN_element_documentation} \]

Property: has_BPMN_element_documentation

Label: Documentation

Description: The modeler MAY add text documentation about the object.

\[ AX_{39} \text{has_BPMN_element_documentation} \text{has range xsd:string} \]
\[ AX_{40} \text{has_BPMN_element_documentation} \text{has domain BPMN_ELEMENT} \]

Class: GRAPHICAL_ELEMENT

Label: Graphical element

Description: These are the elements that define the basic look-and-feel of BPMN. Most business processes will be modeled adequately with these elements

\[ AX_{41} \text{GRAPHICAL_ELEMENT} \equiv \text{FLOW_OBJECT} \sqcup (\text{CONNECTING_OBJECT} \sqcup (\text{SWIMLANE} \sqcup \text{ARTIFACT})) \]
\[ AX_{42} \text{FLOW_OBJECT} \sqsubseteq \neg \text{CONNECTING_OBJECT} \]
\[ AX_{43} \text{FLOW_OBJECT} \sqsubseteq \neg \text{SWIMLANE} \]
\[ AX_{44} \text{FLOW_OBJECT} \sqsubseteq \neg \text{ARTIFACT} \]
\[ AX_{45} \text{CONNECTING_OBJECT} \sqsubseteq \neg \text{SWIMLANE} \]
\[ AX_{46} \text{CONNECTING_OBJECT} \sqsubseteq \neg \text{ARTIFACT} \]
\[ AX_{47} \text{SWIMLANE} \sqsubseteq \neg \text{ARTIFACT} \]
Class: FLOW_OBJECT

Label: Flow Object

Description: Flow objects are the main graphical elements to define the behavior of a Business Process. There are three Flow Objects: Events, Activities and Gateways

\[ AX_{48} \text{FLOW_OBJECT} \equiv \text{EVENT} \sqcup (\text{ACTIVITY} \sqcup \text{GATEWAY}) \]
\[ AX_{49} \text{EVENT} \sqsubseteq \neg \text{ACTIVITY} \]
\[ AX_{50} \text{EVENT} \sqsubseteq \neg \text{GATEWAY} \]
\[ AX_{51} \text{ACTIVITY} \sqsubseteq \neg \text{GATEWAY} \]
\[ AX_{52} \text{FLOW_OBJECT} \sqsubseteq (= 1) \text{has_flow_object_name} \]

Property: has_flow_object_name

Label: Name

Description: Name is an attribute that is a text description of the object.

\[ AX_{53} \text{has_flow_object_name} \text{ has domain FLOW_OBJECT} \]
\[ AX_{54} \text{has_flow_object_name} \text{ has range xsd:string} \]

Property: has_flow_object_assignment

Label: Assignment

Description: One or more assignment expressions MAY be made for the object. For activities, the Assignment SHALL be performed as defined by the AssignTime attribute. The Details of the Assignment is defined in Assignment on page 269.

\[ AX_{55} \text{has_flow_object_assignment} \text{ has domain FLOW_OBJECT} \]
\[ AX_{56} \text{has_flow_object_assignment} \text{ has range ASSIGNMENT} \]

Class: EVENT

Label: Event

Description: 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 (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.

\[ AX_{57} \text{EVENT} \sqsubseteq (= 1) \text{has_event_type} \]

Property: has_event_type

Label: EventType

Description: An event is associated with a flow Dimension (e.g., Start, Intermediate, End)

\[ AX_{58} \text{has_event_type} \text{ has domain EVENT} \]
\[ AX_{59} \text{has_event_type} \text{ has range EVENT_TYPES} \]
\[ AX_{60} \text{EVENT_TYPES} \equiv \{ \text{start}, \text{intermediate}, \text{end} \} \]

Instance: start

Label: start

Instance: intermediate

Label: intermediate
Instance: end
Label: end

\[ AX_{.61} \text{start}\_\text{event} \equiv \text{EVENT} \sqcap \exists \text{has\_event\_type}\{\text{start}\} \]
\[ AX_{.62} \text{intermediate}\_\text{event} \equiv \text{EVENT} \sqcap \exists \text{has\_event\_type}\{\text{intermediate}\} \]
\[ AX_{.63} \text{end}\_\text{event} \equiv \text{EVENT} \sqcap \exists \text{has\_event\_type}\{\text{end}\} \]
\[ AX_{.64} \text{start}\_\text{event} \sqsubseteq \neg \text{intermediate}\_\text{event} \]
\[ AX_{.65} \text{start}\_\text{event} \sqsubseteq \neg \text{end}\_\text{event} \]
\[ AX_{.66} \text{intermediate}\_\text{event} \sqsubseteq \neg \text{end}\_\text{event} \]

Class: START\_EVENT
Label: Start
Description: As the name implies, the Start Event indicates where a particular process will start.

Property: has\_start\_event\_trigger
Label: Trigger
Description: Trigger (EventDetail) is an attribute that defines the type of trigger expected for a Start Event. Of the set of EventDetailTypes (see Section B.11.7, "Event Details," on page 270), only four (4) can be applied to a Start Event: Message, Timer, Conditional, and Signal (see Table 9.4). If there is no EventDetail is defined, then this is considered a None End Event and the Event will not have an internal marker (see Table 9.4). If there is more than one EventDetail is defined, this is considered a Multiple End Event and the Event will have the star internal marker (see Table 9.4).
\[ AX_{.67} \text{has\_start\_event\_trigger} \text{ has domain } \text{START\_EVENT} \]
\[ AX_{.68} \text{has\_start\_event\_trigger} \text{ has range } \text{MESSAGE\_EVENT\_DETAIL} \sqcup \text{TIMER\_EVENT\_DETAIL} \sqcup \text{CONDITIONAL\_EVENT\_DETAIL} \sqcup \text{SIGNAL\_EVENT\_DETAIL} \]

Class: END\_EVENT
Label: End
Description: As the name implies, the End Event indicates where a process will end.

Property: has\_end\_event\_result
Label: Result
Description: Result (EventDetail) is an attribute that defines the type of result expected for an End Event. Of the set of EventDetailTypes (see Section B.11.7, "Event Details," on page 270), only six (6) can be applied to an End Event: Message, Error, Cancel, Compensation, Signal, and Terminate (see Table 9.6). If there is no EventDetail is defined, then this is considered a None End Event and the Event will not have an internal marker (see Table 9.6). If there is more than one EventDetail is defined, this is considered a Multiple End Event and the Event will have the star internal marker (see Table 9.6).
\[ AX_{.69} \text{has\_end\_event\_result} \text{ has domain } \text{END\_EVENT} \]
\[ AX_{.70} \text{has\_end\_event\_result} \text{ has range } \text{MESSAGE\_EVENT\_DETAIL} \sqcup \text{ERROR\_EVENT\_DETAIL} \sqcup \text{CANCEL\_EVENT\_DETAIL} \sqcup \text{COMPENSATION\_EVENT\_DETAIL} \sqcup \text{SIGNAL\_EVENT\_DETAIL} \sqcup \text{TERMINATE\_EVENT\_DETAIL} \]

Class: INTERMEDIATE\_EVENT
**Label:** Intermediate

**Description:** Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process.

\[ AX_{71} \text{ intermediate\_event} \sqsupseteq (\geq 1)\text{has\_intermediate\_event\_target} \]

**Property:** has_intermediate_event_trigger

**Label:** Trigger

**Description:** Trigger (EventDetail) is an attribute that defines the type of trigger expected for an Intermediate Event. Of the set of EventDetailTypes (see Section B.11.7, Event Details, on page 270), only eight (8) can be applied to an Intermediate Event: Message, Timer, Error, Cancel, Compensation, Conditional, Link, and Signal (see Table 9.8). If there is no EventDetail is defined, then this is considered a None Intermediate Event and the Event will not have an internal marker (see Table 9.8). If there is more than one EventDetail is defined, this is considered a Multiple Intermediate Event and the Event will have the star internal marker (see Table 9.8).

\[ AX_{72} \text{ has\_intermediate\_event\_trigger} \text{ has\_domain} \text{INTERMEDIATE\_EVENT} \]

\[ AX_{73} \text{ has\_intermediate\_event\_trigger} \text{ has\_range} \text{MESSAGE\_EVENT\_DETAIL} \sqcup \text{TIMER\_EVENT\_DETAIL} \sqcup \text{ERROR\_EVENT\_DETAIL} \sqcup \text{CANCEL\_EVENT\_DETAIL} \sqcup \text{COMPENSATION\_EVENT\_DETAIL} \sqcup \text{CONDITIONAL\_EVENT\_DETAIL} \sqcup \text{LINK\_EVENT\_DETAIL} \sqcup \text{SIGNAL\_EVENT\_DETAIL} \]

**Property:** has_intermediate_event_target

**Label:** Target

**Description:** 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.

\[ AX_{74} \text{ has\_intermediate\_event\_target} \text{ has\_domain} \text{INTERMEDIATE\_EVENT} \]

\[ AX_{75} \text{ has\_intermediate\_event\_target} \text{ has\_range} \text{ACTIVITY} \]

**Class:** ACTIVITY

**Label:** Activity

**Description:** An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a Process Model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are rounded rectangles. Processes are either unbounded or a contained within a Pool.

\[ AX_{76} \text{ activity} \equiv \text{SUB\_PROCESS} \sqcup \text{TASK} \]

\[ AX_{77} \text{ SUB\_PROCESS} \sqsubseteq \neg \text{TASK} \]

\[ AX_{78} \text{ activity} \sqsubseteq (\geq 1)\text{has\_activity\_activity\_type} \]

**Property:** has_activity_activity_type

**Label:** ActivityType

**Description:** The ActivityType MUST be of type Task or Sub-Process.

\[ AX_{79} \text{ has\_activity\_activity\_type} \text{ has\_domain} \text{ACTIVITY} \]

\[ AX_{80} \text{ has\_activity\_activity\_type} \text{ has\_range} \text{ACTIVITY\_TYPES} \]
Class: ACTIVITY_TYPES

Label: Activity Types

$AX_81 \text{ACTIVITY_TYPES} \equiv \{ \text{task_activity.type}, \text{sub_process_activity.type} \}$

Instance: task_activity.type
Label: task

Instance: sub_process_activity.type
Label: sub_process

$AX_82 \neg \{ \text{task_activity.type} \} \{ \text{sub_process_activity.type} \}$
$AX_83 \text{TASK} \equiv \text{ACTIVITY} \sqcap \exists \text{has.activity.activity.type.} \{ \text{task_activity.type} \}$
$AX_84 \text{SUB_PROCESS} \equiv \text{ACTIVITY} \sqcap \exists \text{has.activity.activity.type.} \{ \text{sub_process_activity.type} \}$
$AX_85 \text{ACTIVITY} \sqsubseteq (= 1) \text{has.activity_status}$

Property: has.activity_status
Label: Status
Description: 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.

$AX_86 \text{has.activity_status} \text{ has domain } \text{ACTIVITY}$
$AX_87 \text{has.activity_status} \text{ has range } \text{xsd:string}\{"None", "Ready", "Active", "Cancelled", "Aborting", "Aborted", "Completing", "Completed"\}$

Property: has.activity.performers
Label: Performers
Description: One or more Performers MAY be entered. The Performers attribute defines the resource that will be responsible for the activity. The Performers entry could be in the form of a specific individual, a group, an organization role or position, or an organization.

$AX_88 \text{has.activity.performers} \text{ has domain } \text{ACTIVITY}$
$AX_89 \text{has.activity.performers} \text{ has range } \text{xsd:string}$

Property: has.activity.properties
Label: Properties
Description: Modeler-defined Properties MAY be added to a activity. These Properties are "local" to the activity. All Tasks, Sub-activity objects, and Sub-activities that are embedded SHALL have access to these Properties. The fully delineated name of these properties is "activity.name.property name" (e.g., "Add Customer.Customer Name"). Further details about the definition of a Property can be found in "Property on page 276."

$AX_90 \text{has.activity.properties} \text{ has domain } \text{ACTIVITY}$
$AX_91 \text{has.activity.properties} \text{ has range } \text{PROPERTY}$

Property: has.activity.input_sets
Label: Input set
Description: The InputSets 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). Further details about the definition of an Input-Set can be found in Section B.11.10, "InputSet," on page 274.
AX has activity_input_sets has domain ACTIVITY
AX has activity_input_sets has range INPUT_SET

Property: has_activity_output_sets
Label: Output set
Description: The OutputSets 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 IORules attribute MAY indicate a relationship between an OutputSet and an InputSet that started the activity. Further details about the definition of an OutputSet can be found in Section B.11.13, "OutputSet," on page 275.

AX has activity_output_sets has domain ACTIVITY
AX has activity_output_sets has range OUTPUT_SET

Property: has_activity_IO_rules
Label: IO Rules
Description: The IORules attribute is a collection of expressions, each of which specifies the required relationship between one input and one output. That is, if the activity is instantiated with a specified input, that activity shall complete with the specified output.
AX has activity_IO_rules has domain ACTIVITY
AX has activity_IO_rules has range EXPRESSION
AX ACTIVITY ⊑ ( = 1) has_activity_start_quantity

Property: has_activity_start_quantity
Label: StartQuantity
Description: The default value is 1. The value MUST NOT be less than 1. This attribute defines the number of Tokens that must arrive before the activity can begin.
AX has activity_start_quantity has domain ACTIVITY
AX has activity_start_quantity has range xsd:positiveInteger
AX ACTIVITY ⊑ ( = 1) has_activity_completion_quantity

Property: has_activity_completion_quantity
Label: CompletionQuantity
Description: The default value is 1. The value MUST NOT be less than 1. This attribute defines the number of Tokens that must be generated from the activity. This number of Tokens will be sent done any outgoing Sequence Flow (assuming any Sequence Flow Conditions are satisfied).
AX has activity_completion_quantity has domain ACTIVITY
AX has activity_completion_quantity has range xsd:positiveInteger
AX ACTIVITY ⊑ ( ≥ 1) has_activity_loop_type

Property: has_activity_loop_type
Label: LoopType
Description: 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 9.6 and Figure 9.15). A Task of type Receive that has its Instantiate attribute set to True MUST NOT have a Standard or MultiInstance LoopType.
AX_105 has_activity_loop_type has domain ACTIVITY
AX_106 has_activity_loop_type has range LOOP_TYPES

Class: LOOP_TYPES
Label: Loop Types
AX_107 LOOP_TYPES ≡ \{standard, multi_instance\}

Instance: standard
Label: standard

Instance: multi_instance
Label: multi_instance
AX_108 (¬ standard)(multi_instance)
AX_109 STANDARD_LOOP_ACTIVITY ≡ ACTIVITY □ \exists has_activity_loop_type.\{standard\}
AX_110 MULTI_INSTANCE_LOOP_ACTIVITY ≡ ACTIVITY □ \exists has_activity_loop_type.\{multi_instance\}

Class: STANDARD_LOOP_ACTIVITY
Label: Standard Loop Activity
Description: An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a Process Model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are rounded rectangles. Processes are either unbounded or a contained within a Pool.
AX_111 STANDARD_LOOP_ACTIVITY ⊑ (1)has_standard_loop_activity.loop_condition

Property: has_standard_loop_activity.loop_condition
Label: Loop Condition
Description: 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 in "Expression on page 273."
AX_112 has_standard_loop_activity.loop_condition has domain STANDARD_LOOP_ACTIVITY
AX_113 has_standard_loop_activity.loop_condition has range EXPRESSION
AX_114 STANDARD_LOOP_ACTIVITY ⊑ (1)has_standard_loop_activity.loop_counter

Property: has_standard_loop_activity.loop_counter
Label: Loop Counter
Description: 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.
AX_115 has_standard_loop_activity.loop_counter has domain STANDARD_LOOP_ACTIVITY
AX_116 has_standard_loop_activity.loop_counter has range xsd:int
AX_117 STANDARD_LOOP_ACTIVITY ⊑ (1)has_standard_loop_activity.loop_maximum

Property: has_standard_loop_activity.loop_maximum
Label: Loop Maximum
**Description:** 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.

\[AX_{118} \text{ has standard loop activity loop maximum has domain STANDARD LOOP ACTIVITY}\]
\[AX_{119} \text{ has standard loop activity loop maximum has range xsd:int}\]
\[AX_{120} \text{ STANDARD LOOP ACTIVITY } \subseteq (\geq 1)\text{has standard loop activity test time}\]

**Property:** has_standard_loop_activity_test_time

**Label:** Test Time

**Description:** 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.

\[AX_{121} \text{ has standard loop activity test time has domain STANDARD LOOP ACTIVITY}\]
\[AX_{122} \text{ has standard loop activity test time has range xsd:string}\{"Before", "After"}\]

---

**Class:** multi_instance_loop_activity

**Label:** Multi Instance Loop Activity

**Description:** An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a Process Model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are rounded rectangles. Processes are either unbounded or a contained within a Pool.

\[AX_{123} \text{ multi_instance_loop_activity } \subseteq (= 1)\text{has multi_instance_loop_activity MI condition}\]

**Property:** has_multi_instance_loop_activity_MI_condition

**Label:** MI Condition

**Description:** 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 in “Expression on page 273.”

\[AX_{124} \text{ has multi instance loop activity MI condition has domain MULTIINSTANCE LOOP ACTIVITY}\]
\[AX_{125} \text{ has multi instance loop activity MI condition has range EXPRESSION}\]
\[AX_{126} \text{ multi_instance_loop_activity } \subseteq (= 1)\text{has multi_instance_loop_activity loop counter}\]

**Property:** has_multi_instance_loop_activity_loop_counter

**Label:** Loop Counter

**Description:** 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.

\[AX_{127} \text{ has multi instance loop activity loop counter has domain MULTIINSTANCE LOOP ACTIVITY}\]
\[AX_{128} \text{ has multi instance loop activity loop counter has range xsd:int}\]
\[AX_{129} \text{ multi_instance_loop_activity } \subseteq (= 1)\text{has multi_instance_loop_activity MI ordering}\]

**Property:** has_multi_instance_loop_activity_MI_ordering

**Label:** MI ordering

**Description:** 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 9.9 and Figure 9.15).

**AX.130** has multi_instance_loop_activity_MI_ordering has domain multi_INSTANCE_LOOP_ACTIVITY

**AX.131** has multi_instance_loop_activity_MI_ordering has range xsd:string("Parallel", "Sequential")

**AX.132** multi_INSTANCE_LOOP_ACTIVITY ⊑ (∼∃has_multi_instance_loop_activity_MI_ordering.{"Parallel"}) ∪ (∃has_multi_instance_loop_activity_MI_ordering.{"Parallel"}) ⊓ (1)has_multi_instance_loop_activity_MI_flow_condition

**Property:** has_multi_instance_loop_activity_MI_flow_condition  
**Label:** MI_FlowCondition  
**Description:** 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 MUST 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 similar to that of a Complex Gateway. The ComplexMI_FlowCondition attribute will determine the Token flow.

**AX.133** has multi_instance_loop_activity_MI_flow_condition has domain multi_INSTANCE_LOOP_ACTIVITY

**AX.134** has multi_instance_loop_activity_MI_flow_condition has range xsd:string("None", "One", "All", "Complex")

**AX.135** multi_INSTANCE_LOOP_ACTIVITY ⊑ (∼∃has_multi_instance_loop_activity_MI_flow_condition.{"Complex"}) ∪ (∃has_multi_instance_loop_activity_MI_flow_condition.{"Complex"}) ⊓ (1)has_multi_instance_loop_activity_complex_MI_flow_condition

**Property:** has_multi_instance_loop_activity_complex_MI_flow_condition  
**Label:** ComplexMI_FlowCondition  
**Description:** If the MI_FlowCondition attribute is set to "Complex," then an Expression Must be entered. This Expression that MAY reference Process data. The expression will be evaluated after each iteration of the Activity and SHALL resolve to a boolean. If the result of the expression evaluation is TRUE, then a Token will be sent down the activity’s outgoing Sequence Flow. Otherwise, no Token will be sent. The attributes of an Expression can be found in "Expression on page 273."

**AX.136** has_multi_instance_loop_activity_complex_MI_flow_condition has domain multi_INSTANCE_LOOP_ACTIVITY

**AX.137** has_multi_instance_loop_activity_complex_MI_flow_condition has range EXPRESSION

---

**Class:** SUB_PROCESS  
**Label:** Sub-process  
**Description:** A Sub-Process is a compound activity that is included within a Process. It is compound in that it can be broken down into a finer level of detail (a Process) through a set of sub-activities.

**AX.138** sub_PROCESS ⊑ (1)has_sub_process_sub_process_type

**Property:** has_sub_process_sub_process_type  
**Label:** SubProcessType  
**Description:** SubProcessType is an attribute that defines whether the Sub-Process details are embedded with in the higher level Process or refers to another, re-usable Process. The default is Embedded.

**AX.139** has_sub_process_sub_process_type has domain SUB_PROCESS
AX_140 has_sub_process_sub_process_type has range SUB_PROCESS_TYPES
AX_141 SUB_PROCESS_TYPES ≡ {embedded, reusable, reference}

Instance: embedded
Label: Embedded

Instance: reusable
Label: Reusable

Instance: reference
Label: Reference

AX_142 EMBEDDED_SUB_PROCESS ≡ SUB_PROCESS ⊓ ∃ has_sub_process_sub_process_type. {embedded}
AX_143 REUSABLE_SUB_PROCESS ≡ SUB_PROCESS ⊓ ∃ has_sub_process_sub_process_type. {reusable}
AX_144 REFERENCE_SUB_PROCESS ≡ SUB_PROCESS ⊓ ∃ has_sub_process_sub_process_type. {reference}
AX_145 EMBEDDED_SUB_PROCESS ⊑ ¬ REUSABLE_SUB_PROCESS
AX_146 EMBEDDED_SUB_PROCESS ⊑ ¬ REFERENCE_SUB_PROCESS
AX_147 REUSABLE_SUB_PROCESS ⊑ ¬ REFERENCE_SUB_PROCESS
AX_148 SUB_PROCESS ⊑ (= 1) has_sub_process_is_a_transaction

Property: has_sub_process_is_a_transaction
Label: IsATransaction
Description: TIsATransaction determines whether or not the behavior of the Sub-Process will follow the behavior of a Transaction (see "Sub-Process Behavior as a Transaction on page 62.")
AX_149 has_sub_process_is_a_transaction has domain SUB_PROCESS
AX_150 has_sub_process_is_a_transaction has range xsd:boolean
AX_151 SUB_PROCESS ⊑ ((∃ has_sub_process_is_a_transaction. {"false"}) ⊓ ((= 0) has_sub_process_sub_transaction_ref)) ∪ ((∃ has_sub_process_is_a_transaction. {"true"}) ⊓ ((= 1) has_sub_process_sub_transaction_ref))

Property: has_sub_process_sub_transaction_ref
Label: Transaction_Ref
Description: If the IsATransaction attribute is False, then a Transaction MUST NOT be identified. If the IsATransaction attribute is True, then a Transaction MUST be identified. The attributes of a Transaction can be found in "Transaction on page 277". Note that Transactions that are in different Pools and are connected through Message Flow MUST have the same TransactionId.
AX_152 has_sub_process_sub_transaction_ref has domain SUB_PROCESS
AX_153 has_sub_process_sub_transaction_ref has range TRANSACTION

Class: EMBEDDED_SUB_PROCESS
Label: Embedded Sub-process
Description:

Property: has_embedded_sub_process_sub_graphical_elements
Label: GraphicalElements
Description: The GraphicalElements attribute identifies all of the objects (e.g., Events, Activities, Gate-
ways, and Artifacts) that are contained within the Embedded Sub-Process.

AX_154 has embedded sub-process sub_graphical_elements has domain EMBEDDED_SUB_PROCESS
AX_155 has embedded sub-process sub_graphical_elements has range GRAPHICAL_ELEMENT
AX_156 EMBEDDED_SUB_PROCESS ⊑ (= 1)has embedded sub-process ad_hoc

Property: has embedded sub-process ad_hoc
Label: Ad_hoc
Description: 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, 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 embedded sub_process or the Sub-embedded sub_process shape for Ad Hoc embedded sub_processes.

AX_157 has embedded sub-process ad_hoc has domain EMBEDDED_SUB_PROCESS
AX_158 has embedded sub-process ad_hoc has range xsd:boolean
AX_159 EMBEDDED_SUB_PROCESS ⊑ (∃ has embedded sub-process ad_hoc. {"false"}) ∪ (∃ has embedded sub-process ad_hoc. {"true"}) ⊓ (= 1) has embedded sub-process ad_hoc ordering ⊓ (= 1) has embedded sub-process ad_hoc completion_condition

Property: has embedded sub-process ad_hoc ordering
Label: AdHocOrdering
Description: 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 embedded sub_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.

AX_160 has embedded sub-process ad_hoc ordering has domain EMBEDDED_SUB_PROCESS
AX_161 has embedded sub-process ad_hoc ordering has range xsd:string("Sequential", "Parallel")

Property: has embedded sub-process ad_hoc completion_condition
Label: AdHocCompletionCondition
Description: If the embedded sub_process is Ad Hoc (the AdHoc attribute is True), then the AdHocCompletionCondition attribute MUST be included. This attribute defines the conditions when the embedded sub_process will end.

AX_162 has embedded sub-process ad_hoc completion_condition has domain EMBEDDED_SUB_PROCESS
AX_163 has embedded sub-process ad_hoc completion_condition has range EXPRESSION

Class: REUSABLE_SUB_PROCESS
Label: Reusable Sub-process
Description:
AX_164 REUSABLE_SUB_PROCESS ⊑ (= 1)has reusable sub-process sub_diagram_ref

Property: has reusable sub-process sub_diagram_ref
Label: DiagramRef
Description: The BPD MUST be identified. The attributes of a BPD can be found in "Business Process Diagram Attributes on page 31."

AX_165 has reusable sub-process sub_diagram_ref has domain REUSABLE_SUB_PROCESS
AX_166 has_reusable_sub_process_sub_diagram_ref has range BUSINESS_PROCESS_DIAGRAM
AX_167 REUSABLE_SUB_PROCESS ⊑ (= 1) has_reusable_sub_process_sub_process_ref

Property: has_reusable_sub_process_sub_process_ref
Label: ProcessRef
Description: A Process MUST be identified. The attributes of a Process can be found in "Processes on page 32"
AX_168 has_reusable_sub_process_sub_process_ref has domain REUSABLE_SUB_PROCESS
AX_169 has_reusable_sub_process_sub_process_ref has range PROCESS

Property: has_reusable_sub_process_sub_input_maps
Label: InputMaps
Description: Multiple input mappings MAY be made between the Reusable Sub-Process and the Process referenced by this object. These mappings are in the form of an expression. A specific mapping expression MUST specify the mapping of Properties between the two Processes OR the mapping of Artifacts between the two Processes.
AX_170 has_reusable_sub_process_sub_input_maps has domain REUSABLE_SUB_PROCESS
AX_171 has_reusable_sub_process_sub_input_maps has range EXPRESSION

Property: has_reusable_sub_process_sub_output_maps
Label: OutputMaps
Description: Multiple output mappings MAY be made between the Reusable Sub-Process and the Process referenced by this object. These mappings are in the form of an expression. A specific mapping expression MUST specify the mapping of Properties between the two Processes OR the mapping of Artifacts between the two Processes.
AX_172 has_reusable_sub_process_sub_output_maps has domain REUSABLE_SUB_PROCESS
AX_173 has_reusable_sub_process_sub_output_maps has range EXPRESSION

Class: REFERENCE_SUB_PROCESS
Label: Reference Sub-process
Description:
AX_174 REFERENCE_SUB_PROCESS ⊑ (= 1) has_reference_sub_process_sub_sub_process_ref

Property: has_reference_sub_process_sub_sub_process_ref
Label: SubProcessRef
Description: The Sub-Process being referenced MUST be identified. The attributes for the Sub-Process element can be found in Table B.12.
AX_175 has_reference_sub_process_sub_sub_process_ref has domain REFERENCE_SUB_PROCESS
AX_176 has_reference_sub_process_sub_sub_process_ref has range SUB_PROCESS

Class: TASK
Label: Task [Atomic]
Description: 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.

\[ AX_{177} \text{ TASK } \sqsubseteq (\geq 1) \text{has_task_task_type} \]

**Property:** has_task_task_type

**Label:** TaskType

**Description:** TaskType is an attribute that has a default of None, but MAY be set to Send, Receive, User, Script, Abstract, Manual, Reference, or Service. The TaskType will be impacted by the Message Flow to and/or from the Task, if Message Flow are used. A TaskType of Receive MUST NOT have an outgoing Message Flow. A TaskType of Send MUST NOT have an incoming Message Flow. A TaskType of Script or Manual MUST 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 B.17 through Table B.22.

\[ AX_{178} \text{ has_task_task_type has domain TASK} \]

\[ AX_{179} \text{ has_task_task_type has range TASK_TYPES} \]

\[ AX_{180} \text{ TASK_TYPES } = \{ \text{service_task_type, receive_task_type, send_task_type, user_task_type, script_task_type, abstract_task_type, manual_task_type, reference_task_type} \} \]

**Instance:** service_task_type

**Label:** Service

**Instance:** receive_task_type

**Label:** Receive

**Instance:** send_task_type

**Label:** Send

**Instance:** user_task_type

**Label:** User

**Instance:** script_task_type

**Label:** Script

**Instance:** abstract_task_type

**Label:** Abstract

**Instance:** manual_task_type

**Label:** Manual

**Instance:** reference_task_type

**Label:** Reference

\[ AX_{181} \text{ SERVICE_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{service_task_type}\} \]
\[ AX_{182} \text{ RECEIVE_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{receive_task_type}\} \]
\[ AX_{183} \text{ SEND_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{send_task_type}\} \]
\[ AX_{184} \text{ USER_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{user_task_type}\} \]
\[ AX_{185} \text{ SCRIPT_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{script_task_type}\} \]
\[ AX_{186} \text{ ABSTRACT_TASK } \equiv \text{TASK} \sqcap \exists \text{has_task_task_type}.\{ \text{abstract_task_type}\} \]
AX.187 MANUAL_TASK ≡ TASK ⊓ ∃has_task_task_type.{manual_task_type}
AX.188 REFERENCE_TASK ≡ TASK ⊓ ∃has_task_task_type.{reference_task_type}
AX.189 SERVICE_TASK ⊑ ¬RECEIVE_TASK
AX.190 SERVICE_TASK ⊑ ¬SEND_TASK
AX.191 SERVICE_TASK ⊑ ¬USER_TASK
AX.192 SERVICE_TASK ⊑ ¬SCRIPT_TASK
AX.193 SERVICE_TASK ⊑ ¬ABSTRACT_TASK
AX.194 SERVICE_TASK ⊑ ¬MANUAL_TASK
AX.195 SERVICE_TASK ⊑ ¬REFERENCE_TASK
AX.196 RECEIVE_TASK ⊑ ¬SEND_TASK
AX.197 RECEIVE_TASK ⊑ ¬USER_TASK
AX.198 RECEIVE_TASK ⊑ ¬SCRIPT_TASK
AX.199 RECEIVE_TASK ⊑ ¬ABSTRACT_TASK
AX.200 RECEIVE_TASK ⊑ ¬MANUAL_TASK
AX.201 RECEIVE_TASK ⊑ ¬REFERENCE_TASK
AX.202 SEND_TASK ⊑ ¬USER_TASK
AX.203 SEND_TASK ⊑ ¬SCRIPT_TASK
AX.204 SEND_TASK ⊑ ¬ABSTRACT_TASK
AX.205 SEND_TASK ⊑ ¬MANUAL_TASK
AX.206 SEND_TASK ⊑ ¬REFERENCE_TASK
AX.207 USER_TASK ⊑ ¬SCRIPT_TASK
AX.208 USER_TASK ⊑ ¬ABSTRACT_TASK
AX.209 USER_TASK ⊑ ¬MANUAL_TASK
AX.210 USER_TASK ⊑ ¬REFERENCE_TASK
AX.211 SCRIPT_TASK ⊑ ¬ABSTRACT_TASK
AX.212 SCRIPT_TASK ⊑ ¬MANUAL_TASK
AX.213 SCRIPT_TASK ⊑ ¬REFERENCE_TASK
AX.214 ABSTRACT_TASK ⊑ ¬MANUAL_TASK
AX.215 ABSTRACT_TASK ⊑ ¬REFERENCE_TASK
AX.216 MANUALE_TASK ⊑ ¬REFERENCE_TASK

**Class:** SERVICE_TASK

**Label:** Service Task

**Description:**
AX.217 SERVICE_TASK ⊑ (= 1)has_service_task_in_message_ref

**Property:** has_service_task_in_message_ref

**Label:** InMessageRef

**Description:** A Message for the InMessageRef attribute MUST be entered. This indicates that the Message will be received at the start of the Task, after the availability of any defined InputSets. One or more corresponding incoming Message Flows MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all incoming Message Flow, but can arrive for only one of
the incoming Message Flow for a single instance of the Task.

Property: has_service_task_in_message_ref
Label: OutMessageRef
Description: A Message for the OutMessageRef attribute MUST be entered. The sending of this message marks the completion of the Task, which may cause the production of an OutputSet. One or more corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all outgoing Message Flow and the Message will be sent down all outgoing Message Flow at the completion of a single instance of the Task.

Property: has_service_task_out_message_ref
Label: MessageRef
Description: A Message for the MessageRef attribute MUST be entered. This indicates that the Message will be received by the Task. The Message in this context is equivalent to an in-only message pattern (Web service). One or more corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all incoming Message Flow, but can arrive for only one of the incoming Message Flow for a single instance of the Task.

Property: has_service_task_instantiate
Label: Instantiate
Description: 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.
Property: `has_receive_task_instantiate` has range `xsd:boolean`

**Label:** Implementation

**Description:** This attribute specifies the technology that will be used to receive the message. A Web service is the default technology.

`AX_231` `has_receive_task_instantiate` has domain `RECEIVE_TASK`

`AX_232` `has_receive_task_instantiate` has range `xsd:string{"Web_Service","Other","Unspecified"}`

---

Class: `SEND_TASK`

**Label:** Send Task

**Description:**

`AX_233` `SEND_TASK` ⊑ (`= 1`) `has_send_task_message_ref`

Property: `has_send_task_message_ref`

**Label:** MessageRef

**Description:** A Message for the MessageRef attribute MUST be entered. This indicates that the Message will be sent by the Task. The Message in this context is equivalent to an out-only message pattern (Web service). One or more corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all outgoing Message Flow and the Message will be sent down all outgoing Message Flow at the completion of a single instance of the Task.

`AX_234` `has_send_task_message_ref` has domain `SEND_TASK`

`AX_235` `has_send_task_message_ref` has range `MESSAGE`

Property: `has_send_task_instantiate`

**Label:** Implementation

**Description:** This attribute specifies the technology that will be used to send the message. A Web service is the default technology.

`AX_236` `has_send_task_instantiate` has domain `SEND_TASK`

`AX_237` `has_send_task_instantiate` has range `xsd:string{"Web_Service","Other","Unspecified"}`

---

Class: `USER_TASK`

**Label:** User Task

**Description:**

`AX_238` `USER_TASK` ⊑ (`= 1`) `has_user_task_in_message_ref`

Property: `has_user_task_in_message_ref`

**Label:** InMessageRef

**Description:** A Message for the InMessageRef attribute MUST be entered. This indicates that the Message will be received at the start of the Task, after the availability of any defined InputSets. One or more corresponding incoming Message Flows MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all incoming Message Flow, but can arrive for only one of the incoming Message Flow for a single instance of the Task.
AX_239 has_user_task_in_message_ref has domain USER_TASK
AX_240 has_user_task_in_message_ref has range MESSAGE
AX_241 USER_TASK ⊑ (= 1)has_user_task_out_message_ref

Property: has_user_task_out_message_ref
Label: OutMessageRef
Description: A Message for the OutMessageRef attribute MUST be entered. The sending of this message marks the completion of the Task, which may cause the production of an OutputSet. One or more corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. The Message is applied to all outgoing Message Flow and the Message will be sent down all outgoing Message Flow at the completion of a single instance of the Task.

AX_242 has_user_task_out_message_ref has domain USER_TASK
AX_243 has_user_task_out_message_ref has range MESSAGE

Property: has_user_task_implementation
Label: Implementation
Description: This attribute specifies the technology that will be used by the Performers to perform the task. A Web service is the default technology.

AX_244 has_user_task_implementation has domain USER_TASK
AX_245 has_user_task_implementation has range xsd:string{"Web_Service","Other","Unspecified"}

Class: SCRIPT_TASK
Label: Script Task
Description:
AX_246 SCRIPT_TASK ⊑ (≥ 1)has_script_task_script

Property: has_script_task_script
Label: Script
Description: 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.

AX_247 has_script_task_script has domain SCRIPT_TASK
AX_248 has_script_task_script has range xsd:string

Class: REFERENCE_TASK
Label: Reference Task
Description:
AX_249 REFERENCE_TASK ⊑ (= 1)has_reference_task_task_ref

Property: has_reference_task_task_ref
Label: TaskRef
Description: The Task being referenced MUST be identified. The attributes for the Task element can be found in Table B.16.

AX_250 has_reference_task_task_ref has domain REFERENCE_TASK
Class: GATEWAY
Label: Gateway
Description: A Gateway is used to control the divergence and convergence of Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths. Internal Markers will indicate the type of behavior control.

AX_252 \text{GATEWAY} \subseteq (= 1) \text{has_gateway\_gateway\_type}

Property: has_gateway\_gateway\_type
Label: GatewayType
Description: GatewayType is by default Exclusive. The GatewayType MAY be set to Inclusive, Complex, or Parallel. 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 9.17).

AX_253 \text{has_gateway\_gateway\_type} \text{has domain GATEWAY}
AX_254 \text{has_gateway\_gateway\_type} \text{has range GATEWAY\_TYPES}

Class: GATEWAY\_TYPES
Label: Gateway Types
Description: Icons within the diamond shape will indicate the type of flow control behavior. The types of control include: 1. exclusive – exclusive decision and merging. Both Data-Based and Event-Based. Data-Based can be shown with or without the "X" marker. 2. exclusive – inclusive decision and merging 3. complex – complex conditions and situations (e.g., 3 out of 5) 4. parallel – forking and joining Each type of control affects both the incoming and outgoing Flow.

AX_255 \text{GATEWAY\_TYPES} \equiv \{ \text{exclusive, inclusive, complex, parallel} \}

Instance: exclusive
Label: exclusive
Description: exclusive – exclusive decision parallel merging. Data-Based or Event-Based - can be shown with inclusive without the "X" marker.

Instance: inclusive
Label: inclusive
Description: inclusive – inclusive decision parallel merging

Instance: complex
Label: complex
Description: Complex – complex conditions parallel situations (e.g., 3 out of 5)

Instance: parallel
Label: parallel
Description: parallel – forking parallel joining

AX_256 (\neg \{ \text{exclusive} \})(\text{inclusive})
AX_257 (¬{exclusive})(complex)
AX_258 (¬{exclusive})(parallel)
AX_259 (¬{inclusive})(complex)
AX_260 (¬{inclusive})(parallel)
AX_261 (¬{complex})(parallel)
AX_262 EXCLUSIVE_GATEWAY ≡ GATEWAY ⊓ ∃ has_gateway_gateway_type.{exclusive}
AX_263 INCLUSIVE_GATEWAY ≡ GATEWAY ⊓ ∃ has_gateway_gateway_type.{inclusive}
AX_264 PARALLEL_GATEWAY ≡ GATEWAY ⊓ ∃ has_gateway_gateway_type.{parallel}
AX_265 COMPLEX_GATEWAY ≡ GATEWAY ⊓ ∃ has_gateway_gateway_type.{complex}

Property: has_gateway_gate
Label: Gates
Description: There MAY be zero or more Gates (except where noted below). 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. For Exclusive Data-Based Gateways: When two Gates are required, one of them MAY be the DefaultGate. For Exclusive Event-Based Gateways: 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.) For Inclusive Gateways: When two Gates are required, one of them MAY be the DefaultGate.
AX_266 has_gateway_gate has domain GATEWAY
AX_267 has_gateway_gate has range GATE

Class: EXCLUSIVE_GATEWAY
Label: Exclusive Gateway
Description: Exclusive Gateway
AX_268 EXCLUSIVE_GATEWAY ⊑ (= 1) has_exclusive_gateway-exclusive_type

Property: has_exclusive_gateway-exclusive_type
Label: ExclusiveType
Description: ExclusiveType is by default Data. The ExclusiveType MAY be set to Event. Since Data-Based Exclusive 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.
AX_269 has_exclusive_gateway-exclusive_type has domain EXCLUSIVE_GATEWAY
AX_270 has_exclusive_gateway-exclusive_type has range EXCLUSIVE_TYPES

Class: EXCLUSIVE_TYPES
Label: Exclusive Types
Description:
AX_271 EXCLUSIVE_TYPES ≡ {data-exclusive_type, event-exclusive_type}

Instance: data-exclusive_type
Label: data
Description: data – Data-Based

Instance: event\textunderscore exclusive\_type

Label: event

Description: event – Event-based

\[ AX_{272} \neg\{\text{data\_exclusive\_type}\}(\text{event\_exclusive\_type}) \]

\[ AX_{273} \text{DATA\_BASED\_EXCLUSIVE\_GATEWAY} \equiv \text{EXCLUSIVE\_GATEWAY} \bigcap \exists \text{has\_exclusive\_gateway\_exclusive\_type}.\{\text{data\_exclusive\_type}\} \]

\[ AX_{274} \text{EVENT\_BASED\_EXCLUSIVE\_GATEWAY} \equiv \text{EXCLUSIVE\_GATEWAY} \bigcap \exists \text{has\_exclusive\_gateway\_exclusive\_type}.\{\text{event\_exclusive\_type}\} \]

Class: DATA\_BASED\_EXCLUSIVE\_GATEWAY

Label: Data Based Exclusive Gateway

Description: Data Based Exclusive Gateway

\[ AX_{275} \text{DATA\_BASED\_EXCLUSIVE\_GATEWAY} \subseteq (=1)\text{has\_data\_based\_exclusive\_gateway\_marker\_visible} \]

Property: has\_data\_based\_exclusive\_gateway\_marker\_visible

Label: MarkerVisible

Description: This attribute determines if the Exclusive 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.

\[ AX_{276} \text{has\_data\_based\_exclusive\_gateway\_marker\_visible} \text{ has domain DATA\_BASED\_EXCLUSIVE\_GATEWAY} \]

\[ AX_{277} \text{has\_data\_based\_exclusive\_gateway\_marker\_visible} \text{ has range xsd:boolean} \]

\[ AX_{278} \text{DATA\_BASED\_EXCLUSIVE\_GATEWAY} \subseteq (\geq 1)\text{has\_data\_based\_exclusive\_gateway\_default\_gate} \]

Property: has\_data\_based\_exclusive\_gateway\_default\_gate

Label: DefaultGate

Description: A Default Gate MAY be specified (see Section B.11.9, "Gate," on page 274).

\[ AX_{279} \text{has\_data\_based\_exclusive\_gateway\_default\_gate} \text{ has domain DATA\_BASED\_EXCLUSIVE\_GATEWAY} \]

\[ AX_{280} \text{has\_data\_based\_exclusive\_gateway\_default\_gate} \text{ has range Gate} \]

Class: EVENT\_BASED\_EXCLUSIVE\_GATEWAY

Label: Event Based Exclusive Gateway

Description: Event Based Exclusive Gateway

\[ AX_{281} \text{EVENT\_BASED\_EXCLUSIVE\_GATEWAY} \subseteq (=1)\text{has\_event\_based\_exclusive\_gateway\_instantiate} \]

Property: has\_event\_based\_exclusive\_gateway\_instantiate

Label: MarkerVisible

Description: 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).

\[ AX_{282} \text{has\_event\_based\_exclusive\_gateway\_instantiate} \text{ has domain EVENT\_BASED\_EXCLUSIVE\_GATEWAY} \]
**Class:** INCLUSIVE_GATEWAY  
**Label:** Inclusive Gateway  
**Description:** Inclusive Gateway  

```
AX_284 INCLUSIVE_GATEWAY ⊑ (≥ 1) has_inclusive_gateway_default_gate
```

**Property:** has_inclusive_gateway_default_gate  
**Label:** DefaultGate  
**Description:** A Default Gate MAY be specified (see Section B.11.9, "Gate," on page 274).  

```
AX_285 has_inclusive_gateway_default_gate has domain INCLUSIVE_GATEWAY
AX_286 has_inclusive_gateway_default_gate has range GATE
```

---

**Class:** COMPLEX_GATEWAY  
**Label:** Complex Gateway  
**Description:** Complex Gateway  

```
AX_287 COMPLEX_GATEWAY ⊑ (≥ 1) has_complex_gateway_incoming_condition
AX_288 COMPLEX_GATEWAY ⊑ (≥ 1) has_sequence_flow_target_ref_inv⊔(≤ 2) has_sequence_flow_target_ref_inv⊓∃ has_complex_gateway_incoming_condition.EXPRESSION)
```

**Property:** has_complex_gateway_incoming_condition  
**Label:** Incoming Condition  
**Description:** 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).  

```
AX_289 has_complex_gateway_incoming_condition has domain COMPLEX_GATEWAY
AX_290 has_complex_gateway_incoming_condition has range EXPRESSION
AX_291 COMPLEX_GATEWAY ⊑ (≥ 1) has_complex_gateway_outgoing_condition
AX_292 COMPLEX_GATEWAY ⊑ (≥ 1) has_sequence_flow_source_ref_inv⊔(≤ 2) has_sequence_flow_source_ref_inv⊓∃ has_complex_gateway_outgoing_condition.EXPRESSION)
```

**Property:** has_complex_gateway_outgoing_condition  
**Label:** Outgoing Condition  
**Description:** 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).  

```
AX_293 has_complex_gateway_outgoing_condition has domain COMPLEX_GATEWAY
AX_294 has_complex_gateway_outgoing_condition has range EXPRESSION
```

---

**Class:** PARALLEL_GATEWAY  
**Label:** Parallel Gateway
**Description:** Parallel Gateway

---

**Class:** SWIMLANE

**Label:** Swimlane

**Description:** There are two ways of grouping the primary modeling elements through "swimlane": Pools and Lanes

\[ AX_{295} \text{SWIMLANE} \equiv \text{POOL} \sqcup \text{LANE} \]
\[ AX_{296} \text{POOL} \subseteq \neg \text{LANE} \]
\[ AX_{297} \text{SWIMLANE} \subseteq (= 1)\text{has_swimlane_name} \]

**Property:** has_swimlane_name

**Label:** Name

**Description:** Name is an attribute that is text description of the Swimlane.

\[ AX_{298} \text{has_swimlane_name} \text{ has domain } \text{SWIMLANE} \]
\[ AX_{299} \text{has_swimlane_name} \text{ has range } \text{xsd:string} \]

---

**Class:** POOL

**Label:** Pool

**Description:** A Pool represents a Participant in a Process. It is also acts as a "swimlane" and a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.

\[ AX_{300} \text{POOL} \subseteq (\geq 1)\text{has_pool_process_ref} \]

**Property:** has_pool_process_ref

**Label:** ProcessRef

**Description:** The ProcessRef 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 "These attributes are used for Graphical Elements (which are Flow Objects (Section B.4."Common Flow Object Attributes," on page 243), Connecting Objects (Section B.10, "Graphical Connecting Objects," on page 263), Swimlanes (Section B.8, "Swimlanes (Pools and Lanes)," on page 259), and Artifacts (Section B.9, "Artifacts," on page 260)), and Supporting Elements (Section B.11, "Supporting Elements," on page 266). on page 241."

\[ AX_{301} \text{has_pool_process_ref} \text{ has domain } \text{POOL} \]
\[ AX_{302} \text{has_pool_process_ref} \text{ has range } \text{PROCESS} \]
\[ AX_{303} \text{POOL} \subseteq (= 1)\text{has_pool_participant_ref} \]

**Property:** has_pool_participant_ref

**Label:** ParticipantRef

**Description:** The Modeler MUST define the Participant for a Pool. The Participant can be either a Role or an Entity. The attributes for a Participant can be found in "Participant on page 276."

\[ AX_{304} \text{has_pool_participant_ref} \text{ has domain } \text{POOL} \]
\[ AX_{305} \text{has_pool_participant_ref} \text{ has range } \text{PARTICIPANT} \]
\[ AX_{306} \text{POOL} \subseteq (\leq 1)\text{has_pool_lanes} \]

**Property:** has_pool_lanes

---

25
Label: Lanes

Description: There MUST 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 “Lane on page 89.”

AX_307 has_pool_lanes has domain pool
AX_308 has_pool_lanes has range lane
AX_309 pool ⊑ (= 1) has_pool_boundary_visible

Property: has_pool_boundary_visible

Label: boundary_visible

Description: 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.

AX_310 has_pool_boundary_visible has domain pool
AX_311 has_pool_boundary_visible has range xsd:boolean
AX_312 pool ⊑ (= 1) has_pool_main_pool

Property: has_pool_main_pool

Label: main_pool

Description: This attribute defines if the Pool is the ”main” Pool or the focus of the diagram. Only one Pool in the Diagram MAY have the attribute set to True.

AX_313 has_pool_main_pool has domain pool
AX_314 has_pool_main_pool has range xsd:boolean

Class: LANE

Label: Lane

Description: A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities.

Property: has_lane_lanes

Label: Lanes

Description: This attribute identifies any Lanes that are nested within the current Lane.

AX_315 has_lane_lanes has domain lane
AX_316 has_lane_lanes has range lane

Class: ARTIFACT

Label: Artifact

Description: Artifacts are used to provide additional information about the Process. There are three standardized Artifacts, but modelers or modeling tools are free to add as many Artifacts as required. There may be addition BPMN efforts to standardize a larger set of Artifacts for general use or for vertical markets. The current set of Artifacts include: Data Object, Group, Annotation

AX_317 artifact ≡ data_object ⊔ (group ⊔ annotation)
AX_318 data_object ⊑ ¬group

26
AX_319 DATA_OBJECT ⊒ ¬ANNOTATION
AX_320 GROUP ⊒ ¬ANNOTATION
AX_321 ARTIFACT ⊒ (= 1)has_artifact_type

Property: has_artifact_type
Label: Name
Description: The ArtifactType MAY be set to DataObject, Group, or Annotation. The ArtifactType list MAY be extended to include new types.
AX_322 has_artifact_type has domain ARTIFACT
AX_323 has_artifact_type has range ARTIFACT_TYPES
AX_324 ARTIFACT_TYPES ≡ { data_object_artifact_type, group_artifact_type, annotation_artifact_type }  

Instance: data_object_artifact_type
Label: Data Object

Instance: group_artifact_type
Label: Group

Instance: annotation_artifact_type
Label: Annotation

AX_325 DATA_OBJECT ≡ ARTIFACT □ ⊔ has_artifact_type.{ data_object_artifact_type }
AX_326 GROUP ≡ ARTIFACT □ ⊔ has_artifact_type.{ group_artifact_type }
AX_327 ANNOTATION ≡ ARTIFACT □ ⊔ has_artifact_type.{ annotation_artifact_type }

Class: DATA_OBJECT
Label: Data Object
Description: Data Objects are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce.
AX_328 DATA_OBJECT ⊒ (= 1)has_data_object_name

Property: has_data_object_name
Label: Name
Description: Name is an attribute that is text description of the object.
AX_329 has_data_object_name has domain DATA_OBJECT
AX_330 has_data_object_name has range xsd:string
AX_331 DATA_OBJECT ⊒ (≥ 1)has_data_object_state

Property: has_data_object_state
Label: State
Description: 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.
AX_332 has_data_object_state has domain DATA_OBJECT
AX_333 has_data_object_state has range xsd:string
Property: has_data_object_properties
Label: Properties
Description: 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”). Further details about the definition of a Property can be found in ”Property on page 276.”

AX_334 has_data_object_properties has domain DATA_OBJECT
AX_335 has_data_object_properties has range PROPERTY

Class: ANNOTATION
Label: Annotation
Description: Text Annotations are a mechanism for a modeler to (attached with an provide additional information for the reader of a Association) BPMN Diagram.

AX_336 ANNOTATION ⊑ (= 1)has_annotation_text

Property: has_annotation_text
Label: Text
Description: Text is an attribute that is text that the modeler wishes to communicate to the reader of the Diagram.

AX_337 has_annotation_text has domain ANNOTATION
AX_338 has_annotation_text has range xsd:string

Class: GROUP
Label: Group
Description: A grouping of activities that does not affect the Sequence Flow. The grouping can be used for documentation or analysis purposes. Groups can also be used to identify the activities of a distributed transaction that is shown across Pools.

AX_339 GROUP ⊑ (= 1)has_group_category_ref

Property: has_group_category_ref
Label: CategoryRef
Description: CategoryRef specifies the Category that the Group represents (Further details about the definition of a Category can be found in ”Category on page 269.”). The name of the Category provides the label for the Group. The graphical elements within the boundaries of the Group will be assigned the Category.

AX_340 has_group_category_ref has domain GROUP
AX_341 has_group_category_ref has range CATEGORY

Property: has_group_graphical_element
Label: GraphicalElement
Description: The GraphicalElements attribute identifies all of the graphical elements (e.g., Events, Activities, Gateways, and Artifacts) that are within the boundaries of the Group.

AX_342 has_group_graphical_element has domain GROUP
AX_343 has_group_graphical_element has range GRAPHICAL_ELEMENT
Class: CONNECTING_OBJECT

Label: Connecting object

Description: There are three ways of connecting the Flow Objects to each other or other information. There are three Connecting Objects: Sequence Flow, Message Flow, and Association

AX_344 CONNECTING_OBJECT ≡ SEQUENCE_FLOW ⊔ (MESSAGE_FLOW ⊔ ASSOCIATION)
AX_345 SEQUENCE_FLOW ⊑ ¬MESSAGE_FLOW
AX_346 SEQUENCE_FLOW ⊑ ¬ASSOCIATION
AX_347 MESSAGE_FLOW ⊑ ¬ASSOCIATION
AX_348 CONNECTING_OBJECT ⊑ (≥ 1)has_connecting_object_name

Property: has_connecting_object_name
Label: Name

Description: Name is an attribute that is text description of the object.

AX_349 has_connecting_object_name has domain CONNECTING_OBJECT
AX_350 has_connecting_object_name has range xsd:string
AX_351 CONNECTING_OBJECT ⊑ (= 1)has_connecting_object_source_ref

Property: has_connecting_object_source_ref
Label: SourceRef

Description: SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected from. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

AX_352 has_connecting_object_source_ref has domain CONNECTING_OBJECT
AX_353 has_connecting_object_source_ref has range GRAPHICAL_ELEMENT
AX_354 CONNECTING_OBJECT ⊑ (= 1)has_connecting_object_target_ref

Property: has_connecting_object_target_ref
Label: TargetRef

Description: Target is an attribute that identifies which Graphical Element the Connecting Object is connected to. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

AX_355 has_connecting_object_target_ref has domain CONNECTING_OBJECT
AX_356 has_connecting_object_target_ref has range GRAPHICAL_ELEMENT
AX_357 has_connecting_object_source_ref_inv = has_connecting_object_source_ref^-1
AX_358 has_connecting_object_target_ref_inv = has_connecting_object_target_ref^-1

Class: SEQUENCE_FLOW

Label: Sequence Flow

Description: A Sequence Flow is used to show the order that activities will be performed in a Process.
AX_359 sequence_flow ⊑ (= 1) has_sequence_flow_condition_type

Property: has_sequence_flow_condition_type
Label: Condition Type

Description: 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 10.1). A None ConditionType MUST 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 10.2). However, a condition indicator MUST NOT be added to the line if the Sequence Flow is outgoing from a Gateway. An Expression ConditionType MUST 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 MUST 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 10.3).

AX_360 has_sequence_flow_condition_type has domain sequence_flow
AX_361 has_sequence_flow_condition_type has range xsd:string["None", "Expression", "Default"]
AX_362 sequence_flow ⊑ (¬∃ has_sequence_flow_condition_type."Expression") ∪ ((∃ has_sequence_flow_condition_type."Expression") \ ( ( = 1) has_sequence_flow_condition_expression))

Property: has_sequence_flow_condition_expression
Label: Condition Expression

Description: 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 Flow's Subject to any constraints imposed by a Source that is a Gateway.

AX_363 has_sequence_flow_condition_expression has domain sequence_flow
AX_364 has_sequence_flow_condition_expression has range expression
AX_365 has_sequence_flow_source_ref ⊑ hasconnecting_object_source_ref
AX_366 has_sequence_flow_target_ref ⊑ hasconnecting_object_target_ref

Property: has_sequence_flow_source_ref
Label: SequenceFlow_SourceRef

Description: SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected from. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

AX_367 has_sequence_flow_source_ref has domain sequence_flow

Property: has_sequence_flow_target_ref
Label: SequenceFlow_TargetRef

Description: Target is an attribute that identifies which Graphical Element the Connecting Object is connected to. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect.
Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

Class: MESSAGE_FLOW

Label: Message Flow

Description: A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two participants (e.g., business entities or business roles).

AX_371 MESSAGE_FLOW ∈ (≥ 1)has_message_flow_message_ref

Property: has_message_flow_message_ref
Label: MessageRef

Description: MessageRef is an optional attribute that identifies the Message that is being sent. The attributes of a Message can be found in "Message on page 275."

AX_372 has_message_flow_message_ref has domain MESSAGE_FLOW
AX_373 has_message_flow_message_ref has range MESSAGE
AX_374 has_message_flow_source_ref ⊑ has_connecting_object_source_ref
AX_375 has_message_flow_target_ref ⊑ has_connecting_object_target_ref

Property: has_message_flow_source_ref
Label: MessageFlow_SourceRef

Description: SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected from. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

AX_376 has_message_flow_source_ref has domain MESSAGE_FLOW

Property: has_message_flow_target_ref
Label: MessageFlow_TargetRef

Description: Target is an attribute that identifies which Graphical Element the Connecting Object is connected to. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

AX_377 has_message_flow_target_ref has domain MESSAGE_FLOW
AX_378 has_message_flow_source_ref_inv = has_message_flow_source_ref⁻¹
AX_379 has_message_flow_target_ref_inv = has_message_flow_target_ref⁻¹

Class: ASSOCIATION

Label: Association

Description: An Association is used to associate information with Flow Objects. Text and graphical non-
Flow Objects can be associated with the Flow Objects.

\[ AX_{380} \text{ association } \subseteq (= 1) \text{has}_\text{association}_\text{direction} \]

**Property:** has\_association\_direction

**Label:** Direction

**Description:** 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 One means that the arrowhead SHALL be at the Target Object. A value of Both means that there SHALL be an arrowhead at both ends of the Association line.

\[ AX_{381} \text{ has}_\text{association}_\text{direction} \text{ has domain } \text{association} \]

\[ AX_{382} \text{ has}_\text{association}_\text{direction} \text{ has range } \text{xsd:string}\{"None","One","Both"\} \]

**Class:** SUPPORTING\_ELEMENT

**Label:** Supporting Element

**Description:** Supporting Element is one of two main elements that are of type BPMN Element (see Figure B.1). The other is Graphical Element. There are 16 types, and a few subtypes, of Support Element. These are: These are: Assignments (see Section B.11.3 on page 269), Categories (see Section B.11.4 on page 269), Entities (see Section B.11.5 on page 269), Event Details (see Section B.11.7 on page 270), Expressions (see Section B.11.8 on page 273), Gates (see Section B.11.9 on page 274), Inputs (see Section B.11.10 on page 274), Messages (see Section B.11.11 on page 275), Outputs (see Section B.11.13 on page 275), Participants (see Section B.11.14 on page 276), Processes (see Section B.3 on page 242), Properties (see Section B.11.15 on page 269), Roles (see Section B.11.16 on page 276), Conditions (see Section B.11.15 on page 269), Transactions (see Section B.11.19 on page 277), and Web Services (see Section B.11.20 on page 277).

\[ AX_{383} \text{ SUPPORTING}_\text{ELEMENT} \equiv \text{PROCESS} \sqcup \text{MESSAGE} \sqcup \text{CONDITION} \sqcup \text{EVENT}_\text{DETAIL} \sqcup \text{ASSIGNMENT} \sqcup \text{EXPRESSION} \sqcup \text{PROPERTY} \sqcup \text{TRANSACTION} \sqcup \text{GATE} \sqcup \text{WEB}_\text{SERVICE} \sqcup \text{ROLE} \sqcup \text{ENTITY} \sqcup \text{PARTICIPANT} \sqcup \text{CATEGORY} \sqcup \text{OUTPUT}\_\text{SET} \sqcup \text{INPUT}\_\text{SET} \]

\[ AX_{384} \text{ PROCESS} \sqsubseteq \neg \text{MESSAGE} \]

\[ AX_{385} \text{ PROCESS} \sqsubseteq \neg \text{CONDITION} \]

\[ AX_{386} \text{ PROCESS} \sqsubseteq \neg \text{EVENT}_\text{DETAIL} \]

\[ AX_{387} \text{ PROCESS} \sqsubseteq \neg \text{ASSIGNMENT} \]

\[ AX_{388} \text{ PROCESS} \sqsubseteq \neg \text{EXPRESSION} \]

\[ AX_{389} \text{ PROCESS} \sqsubseteq \neg \text{PROPERTY} \]

\[ AX_{390} \text{ PROCESS} \sqsubseteq \neg \text{TRANSACTION} \]

\[ AX_{391} \text{ PROCESS} \sqsubseteq \neg \text{GATE} \]

\[ AX_{392} \text{ PROCESS} \sqsubseteq \neg \text{WEB}_\text{SERVICE} \]

\[ AX_{393} \text{ PROCESS} \sqsubseteq \neg \text{ROLE} \]

\[ AX_{394} \text{ PROCESS} \sqsubseteq \neg \text{ENTITY} \]

\[ AX_{395} \text{ PROCESS} \sqsubseteq \neg \text{PARTICIPANT} \]

\[ AX_{396} \text{ PROCESS} \sqsubseteq \neg \text{CATEGORY} \]

\[ AX_{397} \text{ PROCESS} \sqsubseteq \neg \text{OUTPUT}\_\text{SET} \]

\[ AX_{398} \text{ PROCESS} \sqsubseteq \neg \text{INPUT}\_\text{SET} \]

\[ AX_{399} \text{ MESSAGE} \sqsubseteq \neg \text{CONDITION} \]

\[ AX_{400} \text{ MESSAGE} \sqsubseteq \neg \text{EVENT}_\text{DETAIL} \]

\[ AX_{401} \text{ MESSAGE} \sqsubseteq \neg \text{ASSIGNMENT} \]

32
AX_402 message ⊑ ¬EXPRESSION
AX_403 message ⊑ ¬PROPERTY
AX_404 message ⊑ ¬TRANSACTION
AX_405 message ⊑ ¬GATE
AX_406 message ⊑ ¬WEB_SERVICE
AX_407 message ⊑ ¬ROLE
AX_408 message ⊑ ¬ENTITY
AX_409 message ⊑ ¬PARTICIPANT
AX_410 message ⊑ ¬CATEGORY
AX_411 message ⊑ ¬OUTPUT_SET
AX_412 message ⊑ ¬INPUT_SET
AX_413 condition ⊑ ¬EVENT_DETAIL
AX_414 condition ⊑ ¬ASSIGNMENT
AX_415 condition ⊑ ¬EXPRESSION
AX_416 condition ⊑ ¬PROPERTY
AX_417 condition ⊑ ¬TRANSACTION
AX_418 condition ⊑ ¬GATE
AX_419 condition ⊑ ¬WEB_SERVICE
AX_420 condition ⊑ ¬ROLE
AX_421 condition ⊑ ¬ENTITY
AX_422 condition ⊑ ¬PARTICIPANT
AX_423 condition ⊑ ¬CATEGORY
AX_424 condition ⊑ ¬OUTPUT_SET
AX_425 condition ⊑ ¬INPUT_SET
AX_426 EVENT_DETAIL ⊑ ¬ASSIGNMENT
AX_427 EVENT_DETAIL ⊑ ¬EXPRESSION
AX_428 EVENT_DETAIL ⊑ ¬PROPERTY
AX_429 EVENT_DETAIL ⊑ ¬TRANSACTION
AX_430 EVENT_DETAIL ⊑ ¬GATE
AX_431 EVENT_DETAIL ⊑ ¬WEB_SERVICE
AX_432 EVENT_DETAIL ⊑ ¬ROLE
AX_433 EVENT_DETAIL ⊑ ¬ENTITY
AX_434 EVENT_DETAIL ⊑ ¬PARTICIPANT
AX_435 EVENT_DETAIL ⊑ ¬CATEGORY
AX_436 EVENT_DETAIL ⊑ ¬OUTPUT_SET
AX_437 EVENT_DETAIL ⊑ ¬INPUT_SET
AX_438 ASSIGNMENT ⊑ ¬EXPRESSION
AX_439 ASSIGNMENT ⊑ ¬PROPERTY
AX_440 ASSIGNMENT ⊑ ¬TRANSACTION
AX_441 ASSIGNMENT ⊑ ¬GATE
AX_442 ASSIGNMENT ⊑ ¬WEB_SERVICE
AX_443 ASSIGNMENT ⊑ ¬ROLE
AX_444 ASSIGNMENT ⊑ ¬ENTITY
AX_445 ASSIGNMENT ⊑ ¬PARTICIPANT
AX_446 ASSIGNMENT ⊑ ¬CATEGORY
AX_447 ASSIGNMENT ⊑ ¬OUTPUT_SET
AX_448 ASSIGNMENT ⊑ ¬INPUT_SET
AX_449 EXPRESSION ⊑ ¬PROPERTY
AX_450 EXPRESSION ⊑ ¬TRANSACTION
AX_451 EXPRESSION ⊑ ¬GATE
AX_452 EXPRESSION ⊑ ¬WEB_SERVICE
AX_453 EXPRESSION ⊑ ¬ROLE
AX_454 EXPRESSION ⊑ ¬ENTITY
AX_455 EXPRESSION ⊑ ¬PARTICIPANT
AX_456 EXPRESSION ⊑ ¬CATEGORY
AX_457 EXPRESSION ⊑ ¬OUTPUT_SET
AX_458 EXPRESSION ⊑ ¬INPUT_SET
AX_459 PROPERTY ⊑ ¬TRANSACTION
AX_460 PROPERTY ⊑ ¬GATE
AX_461 PROPERTY ⊑ ¬WEB_SERVICE
AX_462 PROPERTY ⊑ ¬ROLE
AX_463 PROPERTY ⊑ ¬ENTITY
AX_464 PROPERTY ⊑ ¬PARTICIPANT
AX_465 PROPERTY ⊑ ¬CATEGORY
AX_466 PROPERTY ⊑ ¬OUTPUT_SET
AX_467 PROPERTY ⊑ ¬INPUT_SET
AX_468 TRANSACTION ⊑ ¬GATE
AX_469 TRANSACTION ⊑ ¬WEB_SERVICE
AX_470 TRANSACTION ⊑ ¬ROLE
AX_471 TRANSACTION ⊑ ¬ENTITY
AX_472 TRANSACTION ⊑ ¬PARTICIPANT
AX_473 TRANSACTION ⊑ ¬CATEGORY
AX_474 TRANSACTION ⊑ ¬OUTPUT_SET
AX_475 TRANSACTION ⊑ ¬INPUT_SET
AX_476 GATE ⊑ ¬WEB_SERVICE
AX_477 GATE ⊑ ¬ROLE
AX_478 GATE ⊑ ¬ENTITY
AX_479 GATE ⊑ ¬PARTICIPANT
AX_480 GATE ⊑ ¬CATEGORY
AX_481 GATE ⊑ ¬OUTPUT_SET
AX_482 GATE ⊑ ¬INPUT_SET
AX_483 WEB_SERVICE ⊑ ¬ROLE
AX_484 WEB_SERVICE ⊑ ¬ENTITY
AX_485 WEB_SERVICE ⊑ ¬PARTICIPANT
Class: artifact_input

Label: ArtifactInput

Description: artifact_input, which is used in the definition of attributes for all graphical elements.

\( AX_{504} \) artifact_input \( \sqsubseteq \) (\( =1 \)) has_artifact_input_artifact_ref

Property: has_artifact_input_artifact_ref

Label: ArtifactRef

Description: This attribute identifies an Artifact that will be used as an input to an activity. The identified Artifact will be part of an InputSet for an activity.

\( AX_{505} \) has_artifact_input_artifact_ref has range artifact
\( AX_{506} \) has_artifact_input_artifact_ref has domain artifact_input
\( AX_{507} \) artifact_input \( \sqsubseteq \) (\( =1 \)) has_artifact_input_required_for_start

Property: has_artifact_input_required_for_start

Label: RequiredForStart

Description: The default value for this attribute is True. This means that the Input is required for an activity to start. If set to False, then the activity MAY start within the input if it is available, but MAY accept the input (more than once) after the activity has started. An InputSet may have a some of ArtifactInputs that have this attribute set to True and some that are set to False.

\( AX_{508} \) has_artifact_input_required_for_start has range xsd:boolean
\( AX_{509} \) has_artifact_input_required_for_start has domain artifact_input

Class: artifact_output
Label: ArtifactOutput
Description: artifact_output, which is used in the definition of attributes for all graphical elements.
AX_510 ARTIFACT_OUTPUT ⊑ (= 1) has_artifact_output_artifact_ref

Property: has_artifact_output_artifact_ref
Label: ArtifactRef
Description: This attribute identifies an Artifact that will be used as an output from an activity. The identified Artifact will be part of an OutputSet for an activity.
AX_511 has_artifact_output_artifact_ref has range ARTIFACT
AX_512 has_artifact_output_artifact_ref has domain ARTIFACT_OUTPUT
AX_513 ARTIFACT_OUTPUT ⊑ (= 1) has_artifact_output.produce_at_completion

Property: has_artifact_output.produce_at_completion
Label: ProduceAtCompletion
Description: The default value for this attribute is True. This means that the Output will be produced when an activity has been completed. If set to False, then the activity MAY produce the output (more than once) before it has completed. An OutputSet may have a some of ArtifactOutputs that have this attribute set to True and some that are set to False.
AX_514 has_artifact_output.produce_at_completion has range xsd:boolean
AX_515 has_artifact_output.produce_at_completion has domain ARTIFACT_OUTPUT

Class: ASSIGNMENT

Label: Assignment
Description: Assignment, which is used in the definition of attributes for Process, Activities, Events, Gateways, and Gates, and which extends the set of common BPMN Element attributes
AX_516 ASSIGNMENT ⊑ (= 1) has_assignment_to

Property: has_assignment_to
Label: To
Description: The target for the Assignment MUST be a Property of the Process or the activity itself.
AX_517 has_assignment_to has domain ASSIGNMENT
AX_518 has_assignment_to has range PROPERTY
AX_519 ASSIGNMENT ⊑ (= 1) has_assignment_from

Property: has_assignment_from
Label: From
Description: 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 "Business Process Diagram Attributes on page 241."
AX_520 has_assignment_from has domain ASSIGNMENT
AX_521 has_assignment_from has range EXPRESSION
AX_522 ASSIGNMENT ⊑ (≥ 1) has_assignment_assign_time
Property: has_assignment_assign_time
Label: AssignTime
Description: An Assignment MAY have a AssignTime setting. If the Object is an activity (Task, Sub-Process, or Process), then the Assignment MUST have an AssignTime. 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.

AX_523 has_assignment_assign_time has range xsd:string{"Start", "End"}
AX_524 has_assignment_assign_time has domain ASSIGNMENT

---

Class: CATEGORY
Label: Category
Description: Category, which is used in the definition of attributes for all BPMN elements, and which extends the set of common BPMN Element attributes (see Table B.2). Since a Category is also a BPMN element, a Category can have Categories to create a hierarchical structure of Categories.

AX_525 CATEGORY ⊑ (= 1) has category_name

Property: has_category_name
Label: Name
Description: Name is an attribute that is text description of the Category and is used to visually distinguish the category.

AX_526 has_category_name has domain CATEGORY
AX_527 has_category_name has range xsd:string

---

Class: CONDITION
Label: Condition
Description: Condition, which is used in the definition of attributes for Start Event and Intermediate Event, and which extends the set of common BPMN Element attributes (see Table B.2).

AX_528 CONDITION ⊑ (= 1) has_condition_name ⊑ (= 1) has_condition_condition_expression

Property: has_condition_name
Label: Name
Description: Name is an optional attribute that is text description of the Condition. If a Name is not entered, then a ConditionExpression MUST be entered.

AX_529 has_condition_name has domain CONDITION
AX_530 has_condition_name has range xsd:string

Property: has_condition_condition_expression
Label: ConditionExpression
Description: A ConditionExpression MAY be entered. In some cases the Condition itself will be stored and maintained in a separate application (e.g., a Rules Engine). If a ConditionExpression is not entered, then a Name MUST be entered. The attributes of an Expression can be found in "Expression on page 273."
AX.531 has_condition has domain CONDITION
AX.532 has_condition has range EXPRESSION

Class: ENTITY
Label: Entity
Description: Entity, which is used in the definition of attributes for a Participant, and which extends the set of common BPMN Element attributes (see Table B.2).
AX.533 ENTITY ⊑ (= 1) has_entity_name

Property: has_entity_name
Label: Name
Description: Name is an attribute that is text description of the Entity.
AX.534 has_entity_name has domain ENTITY
AX.535 has_entity_name has range xsd:string

Class: EVENT_DETAIL
Label: Event Detail
Description: present the attributes common to all Event Details and the specific attributes for the Event Details that have additional attributes. Note that the Cancel and Terminate Event Details do not have additional attributes
AX.536 EVENT_DETAIL_TYPES ⊑ {cancel_event_detail_type, compensation_event_detail_type, link_event_detail_type, error_event_detail_type, conditional_event_detail_type, message_event_detail_type, terminate_event_detail_type, timer_event_detail_type, signal_event_detail_type}
AX.537 EVENT_DETAIL ⊑ (= 1) has_event_detail_type

Property: has_event_detail_type
Label: Event Detail Type
Description: The EventDetailType attribute defines the type of trigger expected for an Event. The set of types includes Message, Timer, Error, Conditional, Link, Signal, Compensate, Cancel, and Terminate. The EventType (Start, Intermediate, and End) will each have a subset of the EventDetailTypes that can be used. The EventDetailType list MAY be extended to include new types. These new types MAY have a new modeler- or tool-defined Marker to fit within the boundaries of the Event.
AX.538 has_event_detail_type has domain EVENT_DETAIL
AX.539 has_event_detail_type has range EVENT_DETAIL_TYPES

Instance: cancel_event_detail_type
Label: cancel

Instance: compensation_event_detail_type
Label: compensation

Instance: link_event_detail_type
Label: link
Instance: `error_event_detail_type`
Label: error

Instance: `conditional_event_detail_type`
Label: conditional

Instance: `message_event_detail_type`
Label: message

Instance: `terminate_event_detail_type`
Label: terminate

Instance: `timer_event_detail_type`
Label: timer

Instance: `signal_event_detail_type`
Label: signal

\[ AX\_540 \neg\{cancel\_event\_detail\_type\}\{compensation\_event\_detail\_type\} \]
\[ AX\_541 \neg\{cancel\_event\_detail\_type\}\{link\_event\_detail\_type\} \]
\[ AX\_542 \neg\{cancel\_event\_detail\_type\}\{error\_event\_detail\_type\} \]
\[ AX\_543 \neg\{cancel\_event\_detail\_type\}\{conditional\_event\_detail\_type\} \]
\[ AX\_544 \neg\{cancel\_event\_detail\_type\}\{message\_event\_detail\_type\} \]
\[ AX\_545 \neg\{cancel\_event\_detail\_type\}\{terminate\_event\_detail\_type\} \]
\[ AX\_546 \neg\{cancel\_event\_detail\_type\}\{timer\_event\_detail\_type\} \]
\[ AX\_547 \neg\{cancel\_event\_detail\_type\}\{signal\_event\_detail\_type\} \]
\[ AX\_548 \neg\{compensation\_event\_detail\_type\}\{link\_event\_detail\_type\} \]
\[ AX\_549 \neg\{compensation\_event\_detail\_type\}\{error\_event\_detail\_type\} \]
\[ AX\_550 \neg\{compensation\_event\_detail\_type\}\{conditional\_event\_detail\_type\} \]
\[ AX\_551 \neg\{compensation\_event\_detail\_type\}\{message\_event\_detail\_type\} \]
\[ AX\_552 \neg\{compensation\_event\_detail\_type\}\{terminate\_event\_detail\_type\} \]
\[ AX\_553 \neg\{compensation\_event\_detail\_type\}\{timer\_event\_detail\_type\} \]
\[ AX\_554 \neg\{compensation\_event\_detail\_type\}\{signal\_event\_detail\_type\} \]
\[ AX\_555 \neg\{link\_event\_detail\_type\}\{error\_event\_detail\_type\} \]
\[ AX\_556 \neg\{link\_event\_detail\_type\}\{conditional\_event\_detail\_type\} \]
\[ AX\_557 \neg\{link\_event\_detail\_type\}\{message\_event\_detail\_type\} \]
\[ AX\_558 \neg\{link\_event\_detail\_type\}\{terminate\_event\_detail\_type\} \]
\[ AX\_559 \neg\{link\_event\_detail\_type\}\{timer\_event\_detail\_type\} \]
\[ AX\_560 \neg\{link\_event\_detail\_type\}\{signal\_event\_detail\_type\} \]
\[ AX\_561 \neg\{error\_event\_detail\_type\}\{conditional\_event\_detail\_type\} \]
\[ AX\_562 \neg\{error\_event\_detail\_type\}\{message\_event\_detail\_type\} \]
\[ AX\_563 \neg\{error\_event\_detail\_type\}\{terminate\_event\_detail\type\} \]
\[ AX\_564 \neg\{error\_event\_detail\_type\}\{timer\_event\_detail\type\} \]
\[ AX\_565 \neg\{error\_event\_detail\_type\}\{signal\_event\_detail\type\} \]
AX_566 (~{conditional_event_detail_type})(message_event_detail_type)
AX_567 (~{conditional_event_detail_type})(terminate_event_detail_type)
AX_568 (~{conditional_event_detail_type})(timer_event_detail_type)
AX_569 (~{conditional_event_detail_type})(signal_event_detail_type)
AX_570 (~{message_event_detail_type})(terminate_event_detail_type)
AX_571 (~{message_event_detail_type})(timer_event_detail_type)
AX_572 (~{message_event_detail_type})(signal_event_detail_type)
AX_573 (~{terminate_event_detail_type})(timer_event_detail_type)
AX_574 (~{terminate_event_detail_type})(signal_event_detail_type)
AX_575 (~{timer_event_detail_type})(signal_event_detail_type)
AX_576 CANCEL_EVENT_DETAIL ≡ EVENT_DETAIL ∩ ∃has_event_detail_type.{cancel_event_detail_type}

Class: CANCEL_EVENT_DETAIL
Label: Cancel Event Detail
AX_577 CONDITIONAL_EVENT_DETAIL ≡ EVENT_DETAIL ∩ ∃has_event_detail_type.{conditional_event_detail_type}

Class: CONDITIONAL_EVENT_DETAIL
Label: Conditional Event Detail
AX_578 CONDITIONAL_EVENT_DETAIL ⊑ (=1)has_conditional_event_condition_ref

Property: has_conditional_event_condition_ref
Label: ConditionRef
Description: If the Trigger is Conditional, then a Condition MUST be entered. The attributes of a Condition can be found in Section B.11.5, "Condition," on page 269.
AX_579 has_conditional_event_condition_ref has domain CONDITIONAL_EVENT_DETAIL
AX_580 has_conditional_event_condition_ref has range CONDITION
AX_581 COMPENSATION_EVENT_DETAIL ≡ EVENT_DETAIL ∩ ∃has_event_detail_type.{compensation_event_detail_type}

Class: COMPENSATION_EVENT_DETAIL
Label: Compensation Event Detail
AX_582 COMPENSATION_EVENT_DETAIL ⊑ (≥1)has_activity_ref

Property: has_activity_ref
Label: ActivityRef
Description: For an End Event: If the Result is a Compensation, then the Activity that needs to be compensated MAY be supplied. If an Activity is not supplied, then the Event broadcast to all completed activities in the Process Instance. For an Intermediate Event within Normal Flow: If the Trigger is a Compensation, then the Activity that needs to be compensated MAY be supplied. If an Activity is not supplied, then the Event broadcast to all completed activities in the Process Instance. This "throws" the compensation. For an Intermediate Event attached to the boundary of an Activity: This Event "catches" the compensation. No further information is required. The Activity the Event is attached to will provide
the Id necessary to match the compensation event with the event that "threw" the compensation or the compensation will be a broadcast.

\( AX_{583} \text{ has_activity_ref has domain compensation_event_detail} \)
\( AX_{584} \text{ has_activity_ref has range activity} \)
\( AX_{585} \text{ error_event_detail} \equiv \text{ event_detail } \sqcap \exists \text{has_event_detail_type.} \{ \text{error_event_detail_type} \} \)

<table>
<thead>
<tr>
<th>Class: ERROR_EVENT_DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label: Error Event Detail</td>
</tr>
<tr>
<td>AX_{586} \text{ error_event_detail} \sqsubseteq (\geq 1)\text{has_error_detail_error_code}</td>
</tr>
</tbody>
</table>

| Property: has_error_detail_error_code |
| Label: ErrorCode |
| Description: For an End Event: If the Result is an Error, then the ErrorCode MUST be supplied. This "throws" the error. For an Intermediate Event within Normal Flow: If the Trigger is an Error, then the ErrorCode MUST be entered. This "throws" the error. For an Intermediate Event attached to the boundary of an Activity: If the Trigger is an Error, then the ErrorCode MAY be entered. This Event "catches" the error. If there is no ErrorCode, then any error SHALL trigger the Event. If there is an ErrorCode, then only an error that matches the ErrorCode SHALL trigger the Event. |

\( AX_{587} \text{ has_error_detail_error_code has domain error_event_detail} \)
\( AX_{588} \text{ has_error_detail_error_code has range xsd:string} \)
\( AX_{589} \text{ link_event_detail} \equiv \text{ event_detail } \sqcap \exists \text{has_event_detail_type.} \{ \text{link_event_detail_type} \} \)

<table>
<thead>
<tr>
<th>Class: LINK_EVENT_DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label: Link Event Detail</td>
</tr>
<tr>
<td>AX_{590} \text{ link_event_detail} \sqsubseteq (= 1)\text{has_link_event_name}</td>
</tr>
</tbody>
</table>

| Property: has_link_event_name |
| Label: Name |
| Description: If the Trigger is a Link, then the Name MUST be entered. |

\( AX_{591} \text{ has_link_event_name has domain link_event_detail} \)
\( AX_{592} \text{ has_link_event_name has range xsd:string} \)
\( AX_{593} \text{ message_event_detail} \equiv \text{ event_detail } \sqcap \exists \text{has_event_detail_type.} \{ \text{message_event_detail_type} \} \)

<table>
<thead>
<tr>
<th>Class: MESSAGE_EVENT_DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label: Message Event Detail</td>
</tr>
<tr>
<td>AX_{594} \text{ message_event_detail} \sqsubseteq (= 1)\text{has_message_event_message_ref}</td>
</tr>
</tbody>
</table>

| Property: has_message_event_message_ref |
| Label: MessageRef |
| Description: If the EventDetailType is a MessageRef, then the a Message MUST be supplied. The attributes of a Message can be found in Section B.11.11, "Message," on page 275. |
AX_595 has_message_event_message_ref has domain MESSAGE_EVENT_DETAIL
AX_596 has_message_event_message_ref has range MESSAGE
AX_597 MESSAGE_EVENT_DETAIL ⊑ (= 1) has_message_event_impementation

**Property:** has_message_event_impementation

**Label:** Implementation

**Description:** This attribute specifies the technology that will be used to send or receive the message. A Web service is the default technology.

AX_598 has_message_event_impementation has domain MESSAGE_EVENT_DETAIL
AX_599 has_message_event_impementation has range xsd:string{"Web_Service", "Other", "Unspecified"}
AX_600 SIGNAL_EVENT_DETAIL ≡ EVENT_DETAIL □ ∃ has_event_detail_type.{signal_event_detail_type}

**Class:** SIGNAL_EVENT_DETAIL

**Label:** Signal Event Detail

AX_601 SIGNAL_EVENT_DETAIL ⊑ (= 1) has_signal_event_signal_ref

**Property:** has_signal_event_signal_ref

**Label:** SignalRef

**Description:** If the Trigger is a Signal, then a Signal Shall be entered. The attributes of a Signal can be found in Section B.11.17, "Signal," on page 277.

AX_602 has_signal_event_signal_ref has domain SIGNAL_EVENT_DETAIL
AX_603 has_signal_event_signal_ref has range SIGNAL
AX_604 TERMINATE_EVENT_DETAIL ≡ EVENT_DETAIL □ ∃ has_event_detail_type.{terminate_event_detail_type}

**Class:** TERMINATE_EVENT_DETAIL

**Label:** Terminate Event Detail

AX_605 TERMINATE_EVENT_DETAIL ⊑ (= 1) has_signal_event_time_date ▐ (= 1) has_signal_event_time_cycle

**Property:** has_signal_event_time_date

**Label:** TimeDate

**Description:** If the Trigger is a Signal, then a TimeDate MAY be entered. If a TimeDate is not entered, then a TimeCycle MUST be entered (see the attribute below). The attributes of a TimeDateExpression can be found in Section B.11.18 on page 277

AX_607 has_signal_event_time_date has domain TERMINATE_EVENT_DETAIL
AX_608 has_signal_event_time_date has range TIME_DATE_EXPRESSION

**Property:** has_signal_event_time_cycle

AX_609 has_signal_event_time_cycle has domain TERMINATE_EVENT_DETAIL
**Label:** TimeCycle

**Description:** 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).

\[\text{AX}_609 \text{ has_timer_event_time_cycle } \text{has domain } \text{TIMER_EVENT_DETAIL}\]

\[\text{AX}_610 \text{ has_timer_event_time_cycle } \text{has range } \text{TIME_DATE_EXPRESSION}\]

---

**Class:** EXPRESSION

**Label:** Expression

**Description:** Expression, which is used in the definition of attributes for Start Event, Intermediate Event, Activity, Complex Gateway, and Sequence Flow, and which extends the set of common BPMN Element attributes (see Table B.2).

\[\text{AX}_611 \text{ EXPRESSION } \subseteq ( = 1) \text{has_expression_expression_body}\]

**Property:** has_expression_expression_body

**Label:** ExpressionBody

**Description:** An ExpressionBody MUST be entered to provide the text of the expression, which will be written in the language defined by the ExpressionLanguage attribute.

\[\text{AX}_612 \text{ has_expression_expression_body } \text{has domain } \text{EXPRESSION}\]

\[\text{AX}_613 \text{ has_expression_expression_body } \text{has range } \text{xsd:string}\]

\[\text{AX}_614 \text{ EXPRESSION } \subseteq ( = 1) \text{has_expression_expression_language}\]

**Property:** has_expression_expression_language

**Label:** ExpressionLanguage

**Description:** A Language MUST be provided to identify the language of the ExpressionBody. The value of the ExpressionLanguage should follow the naming conventions for the version of the specified language.

\[\text{AX}_615 \text{ has_expression_expression_language } \text{has domain } \text{EXPRESSION}\]

\[\text{AX}_616 \text{ has_expression_expression_language } \text{has range } \text{xsd:string}\]

\[\text{AX}_617 \text{ TIME_DATE_EXPRESSION } \subseteq \text{EXPRESS}\]

---

**Class:** TIME_DATE_EXPRESSION

**Label:** TimeDate Expression

**Description:** The TimeDateExpression supporting element is a sub-type of the Expression Element (Expression on page 273) and uses all the attributes of the Expression Element.

---

**Class:** GATE

**Label:** Gate

**Description:** Gate, which is used in the definition of attributes for Gateways, and which extends the set of common BPMN Element attributes (see Table B.2).

\[\text{AX}_618 \text{ GATE } \subseteq ( = 1) \text{has_gate_outgoing_sequence_flow_ref}\]

**Property:** has_gate_outgoing_sequence_flow_ref
Label: OutgoingSequenceFlowRef

Description: Each Gate MUST have an associated (outgoing) Sequence Flow. The attributes of a Sequence Flow can be found in the Section B.10.2 on page 264. For Exclusive Event-Based, Complex, and Parallel Gateways: The Sequence Flow MUST have its Condition attribute set to None (there is not an evaluation of a condition expression). For Exclusive Data-Based, and Inclusive Gateways: 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 set to None. For DefaultGates: The Sequence Flow MUST have its Condition attribute set to Otherwise

AX_619 has_gate_outgoing_sequence_flow_ref has domain GATE
AX_620 has_gate_outgoing_sequence_flow_ref has range SEQUENCE_FLOW

Property: has_gate_assignments

Label: Assignments

Description: One or more assignment expressions MAY be made for each Gate. The Assignment SHALL be performed when the Gate is selected. The Assignment is defined in the Section B.11.3 on page 269.

AX_621 has_gate_assignments has domain GATE
AX_622 has_gate_assignments has range ASSIGNMENT

Class: INPUT_SET

Label: Input Set

Description: InputSet, which is used in the definition of common attributes for Activities and for attributes of a Process, and which extends the set of common BPMN Element attributes (see Table B.2).

AX_623 INPUT_SET ⊆ (∃has_input_set_artifact_input.ARTIFACT_INPUT)⊔(∃has_input_set_property_input.PROPERTY)

Property: has_input_set_artifact_input

Label: ArtifactInput

Description: Zero or more ArtifactInputs MAY be defined for each InputSet. For the combination of ArtifactInputs and PropertyInputs, there MUST be at least one item defined for the InputSet. An Artifact-Input is an Artifact, usually a Data Object. 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. Further details about the definition of an ArtifactInput can be found in Section B.11.1 on page 268.

AX_624 has_input_set_artifact_input has domain INPUT_SET
AX_625 has_input_set_artifact_input has range ARTIFACT_INPUT

Property: has_input_set_property_input

Label: PropertyInput

Description: Zero or more PropertyInputs MAY be defined for each InputSet. For the combination of ArtifactInputs and PropertyInputs, there MUST be at least one item defined for the InputSet.

AX_626 has_input_set_property_input has domain INPUT_SET
AX_627 has_input_set_property_input has range PROPERTY

Class: MESSAGE
Label: Message
Description: Message, which is used in the definition of attributes for a Start Event, End Event, Intermediate Event, Task, and Message Flow, and which extends the set of common BPMN Element attributes (see Table B.2)

\[ AX_{628} \text{ MESSAGE} \sqsubseteq (_{1}) \text{has message name} \]

Property: has_message_name
Label: Name
Description: Name is an attribute that is text description of the Message.

\[ AX_{629} \text{ has message name has domain MESSAGE} \]
\[ AX_{630} \text{ has message name has range xsd:string} \]

Property: has_message_property
Label: Property
Description: Multiple Properties MAY entered for the Message. The attributes of a Property can be found in "Property on page 276."

\[ AX_{631} \text{ has message property has domain MESSAGE} \]
\[ AX_{632} \text{ has message property has range PROPERTY} \]
\[ AX_{633} \text{ MESSAGE} \sqsubseteq (_{1}) \text{has message from ref} \]

Property: has_message_from_ref
Label: FromRef
Description: This defines the source of the Message. The attributes for a Participant can be found in "Participant on page 276."

\[ AX_{634} \text{ has message from ref has domain MESSAGE} \]
\[ AX_{635} \text{ has message from ref has range PARTICIPANT} \]
\[ AX_{636} \text{ MESSAGE} \sqsubseteq (_{1}) \text{has message to ref} \]

Property: has_message_to_ref
Label: ToRef
Description: This defines the source of the Message. The attributes for a Participant can be found in "Participant on page 276."

\[ AX_{637} \text{ has message to ref has domain MESSAGE} \]
\[ AX_{638} \text{ has message to ref has range PARTICIPANT} \]

Class: OBJECT

Label: Object
Description: Object, which is used in the definition of attributes for all graphical elements.

\[ AX_{639} \text{ OBJECT} \sqsubseteq (_{1}) \text{has object id} \]

Property: has_object_id
Label: Id
Description: The Id attribute provides a unique identifier for all objects on a diagram. That is, each object MUST have a different value for the ObjectId attribute.
OutputSet, which is used in the definition of common attributes for Activities and for attributes of a Process, and which extends the set of common BPMN Element attributes (see Table B.2).

\[
AX_{642} \text{output set} \subseteq (\exists \text{has output set artifact output}. \text{ArtifactOutput}) \sqcup \\
(\exists \text{has output set property output}. \text{PropertyOutput})
\]

ArtifactOutput

Zero or more ArtifactOutputs MAY be defined for each InputSet. For the combination of ArtifactOutputs and PropertyOutputs, there MUST be at least one item defined for the OutputSet. An ArtifactOutput is an Artifact, usually a Data Object. 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. Further details about the definition of an ArtifactOutput can be found in Section B.11.2 on page 268.

\[
AX_{643} \text{has output set artifact output} \text{ has domain OUTPUT_SET} \\
AX_{644} \text{has output set artifact output} \text{ has range ARTIFACT_OUTPUT}
\]

PropertyOutput

Zero or more PropertyOutputs MAY be defined for each InputSet. For the combination of ArtifactOutputs and PropertyOutputs, there MUST be at least one item defined for the OutputSet.

\[
AX_{645} \text{has output set property output} \text{ has domain OUTPUT_SET} \\
AX_{646} \text{has output set property output} \text{ has range PROPERTY}
\]

Participant, which is used in the definition of attributes for a Pool, Message, and Web service, and which extends the set of common BPMN Element attributes (see Table B.2).

\[
AX_{647} \text{participant} \subseteq (\exists \text{has participant participant type} \\
(\exists \text{has participant role ref} \sqcap (=1)\text{has participant entity ref}) \\
\sqcup \\
(\exists \text{has participant role ref} \sqcap (=1)\text{has participant entity ref})
\]

ParticipantType

\[
AX_{648} \text{has participant participant type} \text{ has range } \text{xsd:string}{"Role", "Entity"} \\
AX_{649} \text{has participant participant type} \text{ has domain PARTICIPANT} \\
AX_{650} \text{participant} \subseteq (\exists \text{has participant participant type}.{"Role"} \sqcap (=1)\text{has participant role ref}) \\
\sqcup \\
(\exists \text{has participant participant type}.{"Entity"} \sqcap (=1)\text{has participant entity ref})
\]

Role, Entity
Label: RoleRef
Description: If the ParticipantType = Role, then a Role MUST be identified. The attributes for a Role can be found in "Role on page 276."

AX_651 has_participant_role_ref has domain PARTICIPANT
AX_652 has_participant_role_ref has range ROLE

Property: has_participant_entity_ref
Label: EntityRef
Description: If the ParticipantType = Entity, then an Entity MUST be identified. The attributes for an Entity can be found in "Condition on page 269."

AX_653 has_participant_entity_ref has domain PARTICIPANT
AX_654 has_participant_entity_ref has range ENTITY

Class: PROPERTY
Label: Property
Description: Property, which is used in the definition of attributes for a Process and common activity attributes, and which extends the set of common BPMN Element attributes (see Table B.2).

AX_655 property \( \subseteq (= 1) \) has_property_name

Property: has_property_name
Label: Name
Description: Each Property has a Name (e.g., name="Customer Name").

AX_656 has_property_name has domain PROPERTY
AX_657 has_property_name has range xsd:string
AX_658 property \( \subseteq (= 1) \) has_property_type

Property: has_property_type
Label: Type
Description: Each Property has a Type (e.g., type="String"). Properties may be defined hierarchically.

AX_659 has_property_type has domain PROPERTY
AX_660 has_property_type has range xsd:string
AX_661 property \( \subseteq (\geq 1) \) has_property_value

Property: has_property_value
Label: Value
Description: Each Property MAY have a Value specified.

AX_662 has_property_value has domain PROPERTY
AX_663 has_property_value has range EXPRESSION
AX_664 property \( \subseteq (\geq 1) \) has_property_correlation

Property: has_property_correlation
Label: Correlation
Description: If the Correlation attribute is set to True, then the Property is marked to be used for
correlation (e.g., for incoming Messages).

\[ AX_{665} \text{ has\_property\_correlation} \text{ has domain\ PROPERTY} \]
\[ AX_{666} \text{ has\_property\_correlation} \text{ has range\ xsd:boolean} \]

---

**Class: ROLE**

**Label:** Role

**Description:** Role, which is used in the definition of attributes for a Participant, and which extends the set of common BPMN Element attributes (see Table B.2).

\[ AX_{667} \text{ ROLE} \subseteq (= 1)\text{has\_role\_name} \]

**Property:** has\_role\_name

**Label:** Name

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

\[ AX_{668} \text{ has\_role\_name} \text{ has domain\ ROLE} \]
\[ AX_{669} \text{ has\_role\_name} \text{ has range\ xsd:string} \]

---

**Class: SIGNAL**

**Label:** signal

**Description:** Signal, which is used in the definition of attributes for a Start Event, End Event, Intermediate Event, and which extends the set of common BPMN Element attributes (see Table B.2).

\[ AX_{670} \text{ SIGNAL} \subseteq (= 1)\text{has\_signal\_name} \]

**Property:** has\_signal\_name

**Label:** Name

**Description:** Name is an attribute that is text description of the Signal.

\[ AX_{671} \text{ has\_signal\_name} \text{ has domain\ SIGNAL} \]
\[ AX_{672} \text{ has\_signal\_name} \text{ has range\ xsd:string} \]

**Property:** has\_signal\_property

**Label:** Property

**Description:** Multiple Properties MAY entered for the Signal. The attributes of a Property can be found in Property on page 276.

\[ AX_{673} \text{ has\_signal\_property} \text{ has domain\ SIGNAL} \]
\[ AX_{674} \text{ has\_signal\_property} \text{ has range\ PROPERTY} \]

---

**Class: TRANSACTION**

**Label:** Transaction

**Description:** Transaction, which is used in the definition of attributes for a Sub-Process, and which extends the set of common BPMN Element attributes (see Table B.2).

\[ AX_{675} \text{ TRANSACTION} \subseteq (= 1)\text{has\_transaction\_transaction\_id} \]
Property: has_transaction_transaction_id
Label: TransactionId
Description: The TransactionId attribute provides an identifier for the Transactions used within a diagram.
AX_676 has_transaction_transaction_id has range xsd:string
AX_677 has_transaction_transaction_id has domain TRANSACTION
AX_678 TRANSACTION ⊑ (= 1) has_transaction_transaction_protocol

Property: has_transaction_transaction_protocol
Label: TransactionProtocol
Description: This identifies the Protocol (e.g., WS-Transaction or BTP) that will be used to control the transactional behavior of the Sub-Process.
AX_679 has_transaction_transaction_protocol has range xsd:string
AX_680 has_transaction_transaction_protocol has domain TRANSACTION
AX_681 TRANSACTION ⊑ (= 1) has_transaction_transaction_method

Property: has_transaction_transaction_method
Label: TransactionMethod
Description: 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.
AX_682 has_transaction_transaction_method has range xsd:string{"Compensate", "Store", "Image"}
AX_683 has_transaction_transaction_method has domain TRANSACTION

Class: WEB_SERVICE
Label: Web Service
Description: 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, and which extends the set of common BPMN Element attributes (see Table B.2).
AX_684 WEB_SERVICE ⊑ (= 1) has_web_service_participant_ref

Property: has_web_service_participant_ref
Label: ParticipantRef
Description: A Participant for the Web Service MUST be entered. The attributes for a Participant can be found in "Participant on page 276."
AX_685 has_web_service_participant_ref has domain WEB_SERVICE
AX_686 has_web_service_participant_ref has range PARTICIPANT
AX_687 WEB_SERVICE ⊑ (= 1) has_web_service_interface

Property: has_web_service_interface
Label: Interface
Description: (aka portType) An Interface for the Web Service MUST be entered.
AX_688 has_web_service_interface has domain WEB_SERVICE
AX_689 has_web_service_interface has range xsd:string
AX_690 WEB_SERVICE \( \subseteq (\leq 1) \) has_web_service_type

**Property:** has_web_service_operation  
**Label:** Operation  
**Description:** One or more Operations for the Web Service MUST be entered.  
AX_691 has_web_service_operation has domain WEB_SERVICE  
AX_692 has_web_service_operation has range xsd:string

---

**Class:** PROCESS

**Label:** Process  
**Description:** A Process is an activity performed within or across companies or organizations. In BPMN a Process is depicted as a graph of Flow Objects, which are a set of other activities and the controls that sequence them. The concept of process is intrinsically hierarchical. Processes may be defined at any level from enterprise-wide processes to processes performed by a single person. Low-level processes may be grouped together to achieve a common business goal. Note that BPMN defines the term Process fairly specifically and defines a Business Process more generically as a set of activities that are performed within an organization or across organizations. Thus a Business Process, as shown in a Business Process Diagram, may contain more than one separate Process. Each Process may have its own Sub-Processes and would be contained within a Pool (Section B.8.2, on page 260). The individual Processes would be independent in terms of Sequence Flow, but could have Message Flow connecting them.  
AX_693 PROCESS \( \subseteq (= 1) \) has_process_name

**Property:** has_process_name  
**Label:** Name  
**Description:** Name is an attribute that is a text description of the object.  
AX_694 has_process_name has domain PROCESS  
AX_695 has_process_name has range xsd:string  
AX_696 PROCESS \( \subseteq (= 1) \) has_process_process_type

**Property:** has_process_process_type  
**Label:** process_type  
**Description:** 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).  
AX_697 has_process_process_type has domain PROCESS  
AX_698 has_process_process_type has range xsd:string\{"None", "Private", "Abstract", "Collaboration"\}  
AX_699 PROCESS \( \subseteq (= 1) \) has_process_status

**Property:** has_process_status  
**Label:** Status  
**Description:** 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.  
AX_700 has_process_status has domain PROCESS  
AX_701 has_process_status has range xsd:string\{"None", "Ready", "Active", "Cancelled", "Aborting", "Aborted", "Completing", "Completed"\}
Property: has process_graphical_elements
Label: Graphical Elements
Description: The GraphicalElements attribute identifies all of the objects (e.g., Events, Activities, Gateways, and Artifacts) that are contained within the Process.

AX_702 has_process_graphical_elements has domain PROCESS
AX_703 has_process_graphical_elements has range GRAPHICAL_ELEMENT

Property: has process_assignments
Label: Assignments
Description: 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 details of Assignment is defined in "Assignment on page 269.”.

AX_704 has_process_assignments has domain PROCESS
AX_705 has_process_assignments has range ASSIGNMENT

Property: has process_performers
Label: Performers
Description: One or more Performers MAY be entered. The Performers attribute defines the resource that will be responsible for the Process. The Performers entry could be in the form of a specific individual, a group, an organization role or position, or an organization.

AX_706 has_process_performers has domain PROCESS
AX_707 has_process_performers has range xsd:string

Property: has process_properties
Label: Properties
Description: 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 is ”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 ”Property on page 276.”

AX_708 has_process_properties has domain PROCESS
AX_709 has_process_properties has range PROPERTY

Property: has process_input_sets
Label: Input set
Description: The InputSets attribute defines the data requirements for input to the Process. Zero or more InputSets MAY be defined. Each Input set is sufficient to allow the Process to be performed (if it has first been instantiated by the appropriate signal arriving from an incoming Sequence Flow). Further details about the definition of an Input-Set can be found in Section B.11.10 on page 274.

AX_710 has_process_input_sets has domain PROCESS
AX_711 has_process_input_sets has range INPUT_SET

Property: has process_output_sets
Label: Output set
Description: The OutputSets attribute defines the data requirements for output from the Process. Zero
or more OutputSets MAY be defined. At the completion of the Process, only one of the OutputSets may be produced. It is up to the implementation of the Process to determine which set will be produced. However, the IORules attribute MAY indicate a relationship between an OutputSet and an InputSet that started the Process. Further details about the definition of an OutputSet can be found in Section B.11.13 on page 275.

**AX.712** has process output sets has domain process

**AX.713** has process output sets has range output set

**AX.714** process ⊑ (= 1) has process ad hoc

**Property:** has process ad hoc

**Label:** Ad_hoc

**Description:** 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.

**AX.715** has process ad hoc has domain process

**AX.716** has process ad hoc has range xsd:boolean

**AX.717** process ⊑ (∃ has process ad hoc."false") ∪ (∃ has process ad hoc."true") ∩ (= 1) has process ad hoc ordering ∩ (= 1) has process ad hoc completion condition

**Property:** has process ad hoc ordering

**Label:** AdHocOrdering

**Description:** 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.

**AX.718** has process ad hoc ordering has domain process

**AX.719** has process ad hoc ordering has range xsd:string{"Parallel", "Sequential"}

**Property:** has process ad hoc completion condition

**Label:** AdHocCompletionCondition

**Description:** 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.

**AX.720** has process ad hoc completion condition has domain process

**AX.721** has process ad hoc completion condition has range expression

---

Additional axioms described in Chapter 8 and Chapter 9 of [1]

**AX.722** sequence_flow ⊑ ∀ has connecting object source_ref.(intermediate_event ▫ start_event ▫ task ▫ sub_process ▫ gateway)

**AX.723** sequence_flow ⊑ ∀ has connecting object target_ref.(intermediate_event ▫ end_event ▫ task ▫ sub_process ▫ gateway)

**AX.724** message_flow ⊑ ∀ has connecting object source_ref.((intermediate_event ▫ has intermediate event trigger.message_event_detail) ▫ (end_event ▫ has end event result.message_event_detail) ▫ task ▫ sub_process ▫ pool)

**AX.725** message_flow ⊑ ∀ has connecting object target_ref.((intermediate_event ▫ has intermediate event trigger.message_event_detail) ▫ (start_event ▫ )
\exists \text{has_start_event_trigger.MESSAGE_EVENT_DETAIL} \sqcup \text{TASK} \sqcup \text{SUB_PROCESS} \sqcup \text{POOL})
\text{AX}_{726} \text{ ACTIVITY} \sqsubseteq (\exists \text{has_flow_object_assignment.} (\exists \text{has_assignment_assign_time.} \{"Start" \} \sqcup \\
\exists \text{has_assignment_assign_time.} \{"End" \}))
\text{AX}_{727} \text{ START_EVENT} \sqsubseteq \exists \text{has_connecting_object_source_ref_inv.(SEQUENCE_FLOW)}
\text{AX}_{728} \text{ START_EVENT} \sqsubseteq \exists \text{has_connecting_object_source_ref_inv.(SEQUENCE_FLOW} \sqcap \\
\exists \text{has_sequence_flow_condition_type.} \{"None" \})
\text{AX}_{729} \text{ NONE_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \neg \exists \text{has_intermediate_event_trigger.EVENT_DETAIL}
\text{AX}_{730} \text{ CANCEL_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.CANCEL_EVENT_DETAIL}
\text{AX}_{731} \text{ COMPENSATION_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.COMPENSATION_EVENT_DETAIL}
\text{AX}_{732} \text{ LINK_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.LINK_EVENT_DETAIL}
\text{AX}_{733} \text{ ERROR_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.ERROR_EVENT_DETAIL}
\text{AX}_{734} \text{ CONDITIONAL_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.CONDITIONAL_EVENT_DETAIL}
\text{AX}_{735} \text{ MESSAGE_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.MESSAGE_EVENT_DETAIL}
\text{AX}_{736} \text{ TIMER_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.TIMER_EVENT_DETAIL}
\text{AX}_{737} \text{ SIGNAL_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(= 1\right)\exists \text{has_intermediate_event_trigger} \sqcap \\
\exists \text{has_intermediate_event_trigger.SIGNAL_EVENT_DETAIL}
\text{AX}_{738} \text{ MULTIPLE_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \left(\leq 2\right)\exists \text{has_intermediate_event_trigger}
\text{AX}_{739} \text{ ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \\
\exists \text{has_intermediate_event_target.ACTIVITY}
\text{AX}_{740} \text{ NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \equiv \text{INTERMEDIATE_EVENT} \sqcap \\
\neg \exists \text{has_intermediate_event_target.ACTIVITY}
\text{AX}_{741} \text{ ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\text{CANCEL_INTERMEDIATE_EVENT} \sqcup \\
\text{COMPENSATION_INTERMEDIATE_EVENT} \sqcup \text{ERROR_INTERMEDIATE_EVENT} \sqcup \text{CONDITIONAL_INTERMEDIATE_EVENT} \sqcup \\
\text{MESSAGE_INTERMEDIATE_EVENT} \sqcup \text{TIMER_INTERMEDIATE_EVENT} \sqcup \text{SIGNAL_INTERMEDIATE_EVENT} \sqcup \\
\text{MULTIPLE_INTERMEDIATE_EVENT})
\text{AX}_{742} \text{ ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\exists \text{has_intermediate_event_target.} \left(\text{SUB_PROCESS} \sqcap \\
\exists \text{has_sub_process_is_a_transaction.} \{"true" \} \right) \sqcup \left(\neg \exists \text{has_intermediate_event_target.} \left(\text{SUB_PROCESS} \sqcap \\
\exists \text{has_sub_process_is_a_transaction.} \{"true" \} \right) \right) \sqcap \left(\neg \text{CANCEL_INTERMEDIATE_EVENT} \right))
\text{AX}_{743} \text{ ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq \neg \exists \text{has_connecting_object_target_ref_inv.SEQUENCE_FLOW}
\text{AX}_{744} \text{ ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\neg \exists \text{COMPENSATION_INTERMEDIATE_EVENT} \sqcap \\
\left(= 1\right)\exists \text{has_sequence_flow_source_ref_inv}) \sqcup (\text{COMPENSATION_INTERMEDIATE_EVENT} \sqcap \\
\neg \exists \text{has_sequence_flow_source_ref_inv.SEQUENCE_FLOW})
\text{AX}_{745} \text{ NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\text{NONE_INTERMEDIATE_EVENT} \sqcup \\
\text{COMPENSATION_INTERMEDIATE_EVENT} \sqcup \text{LINK_INTERMEDIATE_EVENT} \sqcup \text{CONDITIONAL_INTERMEDIATE_EVENT} \sqcup \\
\text{MESSAGE_INTERMEDIATE_EVENT} \sqcup \text{TIMER_INTERMEDIATE_EVENT} \sqcup \text{SIGNAL_INTERMEDIATE_EVENT})
\text{AX}_{746} \text{ NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\neg (\text{NONE_INTERMEDIATE_EVENT} \sqcup \\
\text{COMPENSATION_INTERMEDIATE_EVENT}) \sqcap \left(\geq 1\right)\exists \text{has_sequence_flow_target_ref_inv}) \sqcup (\neg (\text{NONE_INTERMEDIATE_EVENT} \sqcup \text{COMPENSATION_INTERMEDIATE_EVENT}) \sqcap \left(= 1\right)\exists \text{has_sequence_flow_target_ref_inv})
\text{AX}_{747} \text{ NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT} \sqsubseteq (\text{LINK_INTERMEDIATE_EVENT} \sqcup 

53
\[(\exists \text{has_sequence_flow_condition_type.\{"Expression"\}})\land \forall \text{has_connecting_object_source_ref.}(\neg \text{PARALLEL_GATEWAY})\]

\[
\text{AX}_{771} \text{ ACTIVITY} \subseteq (\neg \exists \text{has_sequence_flow_source_ref_inv.}(\text{SEQUENCE_FLOW} \\
\neg \exists \text{has_sequence_flow_condition_type.\{"Expression"\}}) \lor ((\exists \text{has_sequence_flow_source_ref_inv.}(\text{SEQUENCE_FLOW} \lor \\
\exists \text{has_sequence_flow_condition_type.\{"Expression"\}}) \land (\leq 2) \text{has_sequence_flow_source_ref_inv})
\]

\[
\text{AX}_{772} \text{ SEQUENCE_FLOW} \subseteq (\exists \text{has_connecting_object_source_ref.}(\text{DATA_BASED_EXCLUSIVE_GATEWAY} \lor \\
\text{INCLUSIVE_GATEWAY}) \land \neg \exists \text{has_sequence_flow_condition_type.\{"None"\}}) \lor \\
(\neg \exists \text{has_connecting_object_source_ref.}(\text{DATA_BASED_EXCLUSIVE_GATEWAY} \lor \text{INCLUSIVE_GATEWAY}))
\]

\[
\text{AX}_{773} \text{ SEQUENCE_FLOW} \subseteq (\neg \exists \text{has_sequence_flow_condition_type.\{"Expression"\}}) \lor \\
((\exists \text{has_sequence_flow_condition_type.\{"Expression"\}}) \lor \exists \text{has_connecting_object_source_ref.}(\text{TASK} \lor \text{SUB_PROCESS} \lor \\
\text{DATA_BASED_EXCLUSIVE_GATEWAY} \lor \text{INCLUSIVE_GATEWAY}))
\]

\[
\text{AX}_{774} \text{ SEQUENCE_FLOW} \subseteq (\neg \exists \text{has_sequence_flow_condition_type.\{"Default"\}}) \lor \\
((\exists \text{has_sequence_flow_condition_type.\{"Default"\}}) \lor \forall \text{has_connecting_object_source_ref.}(\text{ACTIVITY} \lor \\
\text{DATA_BASED_EXCLUSIVE_GATEWAY}))
\]

\[
\text{AX}_{775} \text{ ASSOCIATION} \subseteq (\exists \text{has_connecting_object_source_ref.}(\text{ARTIFACT}) \land \exists \text{has_connecting_object_target_ref.}(\text{FLOW_OBJECT}) \lor (\exists \text{has_connecting_object_target_ref.}(\text{ARTIFACT}) \land \exists \text{has_connecting_object_source_ref.}(\text{FLOW_OBJECT}))
\]

References