A proposla of merging axioms between BPMN and DOLCE ontologies

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 [?]. 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 $\mathcal{ALCHOIN}(\mathcal{D})$. 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.

[1]Available for download at `dkm.fbk.eu/index.php/Resources`.

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 [?]. First of all, because we have chosen 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 Merging Axioms

Class: BUSINESS_PROCESS_DIAGRAM

Label: Business Process Diagram

Description: Gather the set of attributes of a Business Process Diagram

AX_1 BUSINESS_PROCESS_DIAGRAM $\sqsubseteq (=1)$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 $\sqsubseteq (=1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\leq 1)$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 $\sqsubseteq (\geq 1)$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 BPMN_ELEMENT $\equiv$ GRAPHICAL_ELEMENT $\sqcup$ SUPPORTING_ELEMENT

AX_32 GRAPHICAL_ELEMENT $\sqsubseteq \neg$SUPPORTING_ELEMENT

AX_33 BPMN_ELEMENT $\sqsubseteq (= 1)$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 has_BPMN_element_id has range OBJECT

AX_35 has_BPMN_element_id 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 Catogories is defined in Category on page 269.

AX_36 has_BPMN_element_category has range CATEGORY

AX_37 has_BPMN_element_category has domain BPMN_ELEMENT

AX_38 BPMN_ELEMENT $\sqsubseteq (\geq 1)$has_BPMN_element_documentation

Property: has_BPMN_element_documentation

Label: Documentation

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

AX_39 has_BPMN_element_documentation has range *xsd:string*

AX_40 has_BPMN_element_documentation 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 GRAPHICAL_ELEMENT $\equiv$ FLOW_OBJECT $\sqcup$ (CONNECTING_OBJECT $\sqcup$ (SWIMLANE $\sqcup$ ARTIFACT))

AX_42 FLOW_OBJECT $\sqsubseteq \neg$CONNECTING_OBJECT

AX_43 FLOW_OBJECT $\sqsubseteq \neg$SWIMLANE

AX_44 FLOW_OBJECT $\sqsubseteq \neg$ARTIFACT

AX_45 CONNECTING_OBJECT $\sqsubseteq \neg$SWIMLANE

AX_46 CONNECTING_OBJECT $\sqsubseteq \neg$ARTIFACT

AX_47 SWIMLANE $\sqsubseteq \neg$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 FLOW_OBJECT $\equiv$ EVENT $\sqcup$ (ACTIVITY $\sqcup$ GATEWAY)

AX_49 EVENT $\sqsubseteq \neg$ACTIVITY

AX_50 EVENT $\sqsubseteq \neg$GATEWAY

AX_51 ACTIVITY $\sqsubseteq \neg$GATEWAY

AX_52 FLOW_OBJECT $\sqsubseteq (= 1)$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 has_flow_object_name has domain FLOW_OBJECT

AX_54 has_flow_object_name 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 has_flow_object_assignment has domain FLOW_OBJECT

AX_56 has_flow_object_assignment 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 EVENT $\sqsubseteq (= 1)$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 has_event_type has domain EVENT

AX_59 has_event_type has range EVENT_TYPES

AX_60 EVENT_TYPES $\equiv \{start, intermediate, end\}$

Instance: *start*

Label: start

Instance: *intermediate*

Label: intermediate

Instance: *end*

Label: end

AX_61 $\textsc{start\_event} \equiv \textsc{event} \sqcap \exists \mathsf{has\_event\_type}.\{start\}$

AX_62 $\textsc{intermediate\_event} \equiv \textsc{event} \sqcap \exists \mathsf{has\_event\_type}.\{intermediate\}$

AX_63 $\textsc{end\_event} \equiv \textsc{event} \sqcap \exists \mathsf{has\_event\_type}.\{end\}$

AX_64 $\textsc{start\_event} \sqsubseteq \neg\textsc{intermediate\_event}$

AX_65 $\textsc{start\_event} \sqsubseteq \neg\textsc{end\_event}$

AX_66 $\textsc{intermediate\_event} \sqsubseteq \neg\textsc{end\_event}$

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 has_start_event_trigger has domain START_EVENT

AX_68 has_start_event_trigger has range $\textsc{message\_event\_detail} \sqcup \textsc{timer\_event\_detail} \sqcup \textsc{conditional\_event\_detail} \sqcup \textsc{signal\_event\_detail}$

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 has_end_event_result has domain END_EVENT

AX_70 has_end_event_result has range $\textsc{message\_event\_detail} \sqcup \textsc{error\_event\_detail} \sqcup \textsc{cancel\_event\_detail} \sqcup \textsc{compensation\_event\_detail} \sqcup \textsc{signal\_event\_detail} \sqcup \textsc{terminate\_event\_detail}$

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 INTERMEDIATE_EVENT $\sqsubseteq (\geq 1)$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 has_intermediate_event_trigger has domain INTERMEDIATE_EVENT

AX_73 has_intermediate_event_trigger has range MESSAGE_EVENT_DETAIL $\sqcup$ TIMER_EVENT_DETAIL $\sqcup$ ERROR_EVENT_DETAIL$\sqcup$CANCEL_EVENT_DETAIL$\sqcup$COMPENSATION_EVENT_DETAIL$\sqcup$CONDITIONAL_EVENT_DETAIL$\sqcup$ LINK_EVENT_DETAIL $\sqcup$ 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 has_intermediate_event_target has domain INTERMEDIATE_EVENT

AX_75 has_intermediate_event_target has range 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 ACTIVITY $\equiv$ SUB_PROCESS $\sqcup$ TASK

AX_77 SUB_PROCESS $\sqsubseteq \neg$TASK

AX_78 ACTIVITY $\sqsubseteq (= 1)$has_activity_activity_type

Property: has_activity_activity_type

Label: ActivityType

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

AX_79 has_activity_activity_type has domain ACTIVITY

AX_80 has_activity_activity_type has range ACTIVITY_TYPES

Class: ACTIVITY_TYPES

Label: Activity Types

AX_81 ACTIVITY_TYPES $\equiv \{task\_activity\_type, sub\_process\_activity\_type\}$

Instance: *task_activity_type*

Label: task

Instance: *sub_process_activity_type*

Label: sub_process

AX_82 $(\neg\{task\_activity\_type\})(sub\_process\_activity\_type)$

AX_83 TASK $\equiv$ ACTIVITY $\sqcap \exists$has_activity_activity_type.$\{task\_activity\_type\}$

AX_84 SUB_PROCESS $\equiv$ ACTIVITY $\sqcap \exists$has_activity_activity_type.$\{sub\_process\_activity\_type\}$

AX_85 ACTIVITY $\sqsubseteq (= 1)$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 has_activity_status has domain ACTIVITY

AX_87 has_activity_status has range *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 has_activity_performers has domain ACTIVITY

AX_89 has_activity_performers has range *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-activityes 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 has_activity_properties has domain ACTIVITY

AX_91 has_activity_properties has range 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_92 has_activity_input_sets has domain ACTIVITY

AX_93 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_94 has_activity_output_sets has domain ACTIVITY

AX_95 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_96 has_activity_IO_rules has domain ACTIVITY

AX_97 has_activity_IO_rules has range EXPRESSION

AX_98 ACTIVITY $\sqsubseteq (= 1)$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_99 has_activity_start_quantity has domain ACTIVITY

AX_100 has_activity_start_quantity has range *xsd:positiveInteger*

AX_101 ACTIVITY $\sqsubseteq (= 1)$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_102 has_activity_completion_quantity has domain ACTIVITY

AX_103 has_activity_completion_quantity has range *xsd:positiveInteger*

AX_104 ACTIVITY $\sqsubseteq (\geq 1)$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 $\equiv \{standard, multi\_instance\}$

Instance: *standard*

Label: standard

Instance: *multi_instance*

Label: multi_instance

AX_108 $(\neg\{standard\})(multi\_instance)$

AX_109 STANDARD_LOOP_ACTIVITY $\equiv$ ACTIVITY $\sqcap \exists$has_activity_loop_type.$\{standard\}$

AX_110 MULTI_INSTANCE_LOOP_ACTIVITY $\equiv$ ACTIVITY $\sqcap \exists$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (\geq 1)$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 has_standard_loop_activity_loop_maximum has domain STANDARD_LOOP_ACTIVITY

*AX*_119 has_standard_loop_activity_loop_maximum has range *xsd:int*

*AX*_120 STANDARD_LOOP_ACTIVITY $\sqsubseteq$ (≥ 1)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 has_standard_loop_activity_test_time has domain STANDARD_LOOP_ACTIVITY

*AX*_122 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 MULTI_INSTANCE_LOOP_ACTIVITY $\sqsubseteq$ $(= 1)$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 has_multi_instance_loop_activity_MI_condition has domain MULTI_INSTANCE_LOOP_ACTIVITY

*AX*_125 has_multi_instance_loop_activity_MI_condition has range EXPRESSION

*AX*_126 MULTI_INSTANCE_LOOP_ACTIVITY $\sqsubseteq$ $(= 1)$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 has_multi_instance_loop_activity_loop_counter has domain MULTI_INSTANCE_LOOP_ACTIVITY

*AX*_128 has_multi_instance_loop_activity_loop_counter has range *xsd:int*

*AX*_129 MULTI_INSTANCE_LOOP_ACTIVITY $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ ($\neg\exists$has_multi_instance_loop_activity_MI_ordering.{"Parallel"}) $\sqcup$ (($\exists$has_multi_instance_loop_activity_MI_ordering.{"Parallel"})$\sqcap$($=1$)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 $\sqsubseteq$ ($\neg\exists$has_multi_instance_loop_activity_MI_flow_condition.{"Complex"})$\sqcup$ (($\exists$has_multi_instance_loop_activity_MI_flow_condition.{"Complex"}) $\sqcap$ ($=1$)has_multi_instance_loop_activity_complex_MI_flow_condition)

Property: has_multi_instance_loop_activity_complex_MI_flow_condition

Label: ComplexMI_FlowCcondition

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 $\sqsubseteq$ ($=1$)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 $\equiv \{embedded, reusable, reference\}$

Instance: *embedded*
Label: Embedded

Instance: *reusable*
Label: Reusable

Instance: *reference*
Label: Reference

AX_142 EMBEDDED_SUB_PROCESS $\equiv$ SUB_PROCESS $\sqcap \exists$has_sub_process_sub_process_type.$\{embedded\}$

AX_143 REUSABLE_SUB_PROCESS $\equiv$ SUB_PROCESS $\sqcap \exists$has_sub_process_sub_process_type.$\{reusable\}$

AX_144 REFERENCE_SUB_PROCESS $\equiv$ SUB_PROCESS $\sqcap \exists$has_sub_process_sub_process_type.$\{reference\}$

AX_145 EMBEDDED_SUB_PROCESS $\sqsubseteq \neg$REUSABLE_SUB_PROCESS

AX_146 EMBEDDED_SUB_PROCESS $\sqsubseteq \neg$REFERENCE_SUB_PROCESS

AX_147 REUSABLE_SUB_PROCESS $\sqsubseteq \neg$REFERENCE_SUB_PROCESS

AX_148 SUB_PROCESS $\sqsubseteq (= 1)$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 $\sqsubseteq ((\exists$has_sub_process_is_a_transaction.$\{$"false"$\}) \sqcap ((= 0)$has_sub_process_sub_transaction_ref$)) \sqcup ((\exists$has_sub_process_is_a_transaction.$\{$"true"$\}) \sqcap ((= 1)$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, Gateways, 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 $\sqsubseteq (=1)$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 $\sqsubseteq (\exists$has_embedded_sub_process_ad_hoc.$\{$"false"$\}) \sqcup$
$(\exists$has_embedded_sub_process_ad_hoc.$\{$"true"$\} \sqcap (=1)$has_embedded_sub_process_ad_hoc_ordering $\sqcap$
$(=1)$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 $\sqsubseteq (=1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 TASK $\sqsubseteq (\geq 1)$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 has_task_task_type has domain TASK

AX_179 has_task_task_type has range TASK_TYPES

AX_180 TASK_TYPES $\equiv \{$*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 SERVICE_TASK $\equiv$ TASK $\sqcap \exists$has_task_task_type.$\{$*service_task_type*$\}$

AX_182 RECEIVE_TASK $\equiv$ TASK $\sqcap \exists$has_task_task_type.$\{$*receive_task_type*$\}$

AX_183 SEND_TASK $\equiv$ TASK $\sqcap \exists$has_task_task_type.$\{$*send_task_type*$\}$

AX_184 $\textsc{user\_task} \equiv \textsc{task} \sqcap \exists \mathsf{has\_task\_task\_type}.\{\mathit{user\_task\_type}\}$
AX_185 $\textsc{script\_task} \equiv \textsc{task} \sqcap \exists \mathsf{has\_task\_task\_type}.\{\mathit{script\_task\_type}\}$
AX_186 $\textsc{abstract\_task} \equiv \textsc{task} \sqcap \exists \mathsf{has\_task\_task\_type}.\{\mathit{abstract\_task\_type}\}$
AX_187 $\textsc{manual\_task} \equiv \textsc{task} \sqcap \exists \mathsf{has\_task\_task\_type}.\{\mathit{manual\_task\_type}\}$
AX_188 $\textsc{reference\_task} \equiv \textsc{task} \sqcap \exists \mathsf{has\_task\_task\_type}.\{\mathit{reference\_task\_type}\}$
AX_189 $\textsc{service\_task} \sqsubseteq \neg\textsc{receive\_task}$
AX_190 $\textsc{service\_task} \sqsubseteq \neg\textsc{send\_task}$
AX_191 $\textsc{service\_task} \sqsubseteq \neg\textsc{user\_task}$
AX_192 $\textsc{service\_task} \sqsubseteq \neg\textsc{script\_task}$
AX_193 $\textsc{service\_task} \sqsubseteq \neg\textsc{abstract\_task}$
AX_194 $\textsc{service\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_195 $\textsc{service\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_196 $\textsc{receive\_task} \sqsubseteq \neg\textsc{send\_task}$
AX_197 $\textsc{receive\_task} \sqsubseteq \neg\textsc{user\_task}$
AX_198 $\textsc{receive\_task} \sqsubseteq \neg\textsc{script\_task}$
AX_199 $\textsc{receive\_task} \sqsubseteq \neg\textsc{abstract\_task}$
AX_200 $\textsc{receive\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_201 $\textsc{receive\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_202 $\textsc{send\_task} \sqsubseteq \neg\textsc{user\_task}$
AX_203 $\textsc{send\_task} \sqsubseteq \neg\textsc{script\_task}$
AX_204 $\textsc{send\_task} \sqsubseteq \neg\textsc{abstract\_task}$
AX_205 $\textsc{send\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_206 $\textsc{send\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_207 $\textsc{user\_task} \sqsubseteq \neg\textsc{script\_task}$
AX_208 $\textsc{user\_task} \sqsubseteq \neg\textsc{abstract\_task}$
AX_209 $\textsc{user\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_210 $\textsc{user\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_211 $\textsc{script\_task} \sqsubseteq \neg\textsc{abstract\_task}$
AX_212 $\textsc{script\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_213 $\textsc{script\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_214 $\textsc{abstract\_task} \sqsubseteq \neg\textsc{manual\_task}$
AX_215 $\textsc{abstract\_task} \sqsubseteq \neg\textsc{reference\_task}$
AX_216 $\textsc{manual\_task} \sqsubseteq \neg\textsc{reference\_task}$

Class: SERVICE_TASK

Label: Service Task
Description:
AX_217 $\textsc{service\_task} \sqsubseteq (=1)\mathsf{has\_service\_task\_in\_message\_ref}$

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.

*AX_*218 has_service_task_in_message_ref has domain SERVICE_TASK

*AX_*219 has_service_task_in_message_ref has range MESSAGE

*AX_*220 SERVICE_TASK $\sqsubseteq$ $(= 1)$has_service_task_out_message_ref

Property: has_service_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_*221 has_service_task_out_message_ref has domain SERVICE_TASK

*AX_*222 has_service_task_out_message_ref has range MESSAGE

Property: has_service_task_implementation

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_*223 has_service_task_implementation has domain SERVICE_TASK

*AX_*224 has_service_task_implementation has range *xsd:string*{"Web_Service","Other","Unspecified"}

Class: RECEIVE_TASK

Label: Receive Task

Description:

*AX_*225 RECEIVE_TASK $\sqsubseteq$ $(= 1)$has_receive_task_message_ref

Property: has_receive_task_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.

*AX_*226 has_receive_task_message_ref has domain RECEIVE_TASK

*AX_*227 has_receive_task_message_ref has range MESSAGE

*AX_*228 RECEIVE_TASK $\sqsubseteq$ $(= 1)$has_receive_task_instantiate

Property: has_receive_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.

AX_229 has_receive_task_instantiate has domain RECEIVE_TASK

AX_230 has_receive_task_instantiate has range *xsd:boolean*

Property: has_receive_task_implementation

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_implementation has domain RECEIVE_TASK

AX_232 has_receive_task_implementation has range *xsd:string*{"Web_Service","Other","Unspecified"}

Class: SEND_TASK

Label: Send Task

Description:

AX_233 SEND_TASK $\sqsubseteq (= 1)$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_implementation

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_implementation has domain SEND_TASK

AX_237 has_send_task_implementation has range *xsd:string*{"Web_Service","Other","Unspecified"}

Class: USER_TASK

Label: User Task

Description:

AX_238 USER_TASK $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (= 1)$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

AX_251 has_reference_task_task_ref has range 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 GATEWAY $\sqsubseteq (= 1)$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 has_gateway_gateway_type has domain GATEWAY

AX_254 has_gateway_gateway_type 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. esclusive – 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 GATEWAY_TYPES $\equiv \{$*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 $(\neg\{exclusive\})(inclusive)$

AX_257 $(\neg\{exclusive\})(complex)$

AX_258 $(\neg\{exclusive\})(parallel)$

AX_259 $(\neg\{inclusive\})(complex)$

AX_260 $(\neg\{inclusive\})(parallel)$

AX_261 $(\neg\{complex\})(parallel)$

AX_262 EXCLUSIVE_GATEWAY $\equiv$ GATEWAY $\sqcap$ $\exists$has_gateway_gateway_type.$\{exclusive\}$

AX_263 INCLUSIVE_GATEWAY $\equiv$ GATEWAY $\sqcap$ $\exists$has_gateway_gateway_type.$\{inclusive\}$

AX_264 PARALLEL_GATEWAY $\equiv$ GATEWAY $\sqcap$ $\exists$has_gateway_gateway_type.$\{parallel\}$

AX_265 COMPLEX_GATEWAY $\equiv$ GATEWAY $\sqcap$ $\exists$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 $\sqsubseteq$ $(=1)$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 $\equiv$ $\{data\_exclusive\_type, event\_exclusive\_type\}$

Instance: *data_exclusive_type*
Label: data
Description: data – Data-Based

Instance: *event_exclusive_type*
Label: event
Description: event – Event-based
AX_272 $(\neg\{\textit{data\_exclusive\_type}\})(\textit{event\_exclusive\_type})$
AX_273 DATA_BASED_EXCLUSIVE_GATEWAY $\equiv$ EXCLUSIVE_GATEWAY $\sqcap$ $\exists \mathsf{has\_exclusive\_gateway\_exclusive\_type}.\{\textit{data\_exclusive\_type}\}$
AX_274 EVENT_BASED_EXCLUSIVE_GATEWAY $\equiv$ EXCLUSIVE_GATEWAY $\sqcap$ $\exists \mathsf{has\_exclusive\_gateway\_exclusive\_type}.\{\textit{event\_exclusive\_type}\}$

Class: DATA_BASED_EXCLUSIVE_GATEWAY

Label: Data Based Exclusive Gateway
Description: Data Based Exclusive Gateway
AX_275 DATA_BASED_EXCLUSIVE_GATEWAY $\sqsubseteq (=1)\mathsf{has\_data\_based\_exclusive\_gateway\_marker\_visible}$

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 has_data_based_exclusive_gateway_marker_visible has domain DATA_BASED_EXCLUSIVE_GATEWAY
AX_277 has_data_based_exclusive_gateway_marker_visible has range *xsd:boolean*
AX_278 DATA_BASED_EXCLUSIVE_GATEWAY $\sqsubseteq (\geq 1)\mathsf{has\_data\_based\_exclusive\_gateway\_default\_gate}$

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 has_data_based_exclusive_gateway_default_gate has domain DATA_BASED_EXCLUSIVE_GATEWAY
AX_280 has_data_based_exclusive_gateway_default_gate has range GATE

Class: EVENT_BASED_EXCLUSIVE_GATEWAY

Label: Event Based Exclusive Gateway
Description: Event Based Exclusive Gateway
AX_281 EVENT_BASED_EXCLUSIVE_GATEWAY $\sqsubseteq (=1)\mathsf{has\_event\_based\_exclusive\_gateway\_instantiate}$

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 has_event_based_exclusive_gateway_instantiate has domain EVENT_BASED_EXCLUSIVE_GATEWAY

*AX*_283 has_event_based_exclusive_gateway_instantiate has range *xsd:boolean*

Class: INCLUSIVE_GATEWAY

Label: Inclusive Gateway

Description: Inclusive Gateway

*AX*_284 INCLUSIVE_GATEWAY $\sqsubseteq (\geq 1)$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 $\sqsubseteq (\geq 1)$has_complex_gateway_incoming_condition

*AX*_288 COMPLEX_GATEWAY $\sqsubseteq (\geq 1)$has_sequence_flow_target_ref_inv$\sqcup((\leq 2)$has_sequence_flow_target_ref_inv$\sqcap$ $\exists$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 $\sqsubseteq (\geq 1)$has_complex_gateway_outgoing_condition

*AX*_292 COMPLEX_GATEWAY $\sqsubseteq (\geq 1)$has_sequence_flow_source_ref_inv$\sqcup((\leq 2)$has_sequence_flow_source_ref_inv$\sqcap$ $\exists$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 SWIMLANE $\equiv$ POOL $\sqcup$ LANE

AX_296 POOL $\sqsubseteq \neg$LANE

AX_297 SWIMLANE $\sqsubseteq (= 1)$has_swimlane_name

Property: has_swimlane_name

Label: Name

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

AX_298 has_swimlane_name has domain SWIMLANE

AX_299 has_swimlane_name has range *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 POOL $\sqsubseteq (\geq 1)$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 has_pool_process_ref has domain POOL

AX_302 has_pool_process_ref has range PROCESS

AX_303 POOL $\sqsubseteq (= 1)$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 has_pool_participant_ref has domain POOL

AX_305 has_pool_participant_ref has range PARTICIPANT

AX_306 POOL $\sqsubseteq (\leq 1)$has_pool_lanes

Property: has_pool_lanes

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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\textsc{artifact} \equiv \textsc{data\_object} \sqcup (\textsc{group} \sqcup \textsc{annotation})$

AX_318 $\textsc{data\_object} \sqsubseteq \neg\textsc{group}$

AX_319 $\textsc{data\_object} \sqsubseteq \neg\textsc{annotation}$

AX_320 $\textsc{group} \sqsubseteq \neg\textsc{annotation}$

AX_321 $\textsc{artifact} \sqsubseteq (= 1)\mathsf{has\_artifact\_type}$

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 $\textsc{artifact}$

AX_323 has_artifact_type has range $\textsc{artifact\_types}$

AX_324 $\textsc{artifact\_types} \equiv \{\mathit{data\_object\_artifact\_type}, \mathit{group\_artifact\_type}, \mathit{annotation\_artifact\_type}\}$

Instance: *data_object_artifact_type*

Label: Data Object

Instance: *group_artifact_type*

Label: Group

Instance: *annotation_artifact_type*

Label: Annotation

AX_325 $\textsc{data\_object} \equiv \textsc{artifact} \sqcap \exists\mathsf{has\_artifact\_type}.\{\mathit{data\_object\_artifact\_type}\}$

AX_326 $\textsc{group} \equiv \textsc{artifact} \sqcap \exists\mathsf{has\_artifact\_type}.\{\mathit{group\_artifact\_type}\}$

AX_327 $\textsc{annotation} \equiv \textsc{artifact} \sqcap \exists\mathsf{has\_artifact\_type}.\{\mathit{annotation\_artifact\_type}\}$

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 $\textsc{data\_object} \sqsubseteq (= 1)\mathsf{has\_data\_object\_name}$

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 $\textsc{data\_object}$

AX_330 has_data_object_name has range *xsd:string*

AX_331 $\textsc{data\_object} \sqsubseteq (\geq 1)\mathsf{has\_data\_object\_state}$

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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ $(= 1)$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 $\equiv$ SEQUENCE_FLOW $\sqcup$ (MESSAGE_FLOW $\sqcup$ ASSOCIATION)

AX_345 SEQUENCE_FLOW $\sqsubseteq \neg$MESSAGE_FLOW

AX_346 SEQUENCE_FLOW $\sqsubseteq \neg$ASSOCIATION

AX_347 MESSAGE_FLOW $\sqsubseteq \neg$ASSOCIATION

AX_348 CONNECTING_OBJECT $\sqsubseteq (\geq 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\textsc{sequence\_flow} \sqsubseteq (=1)\mathsf{has\_sequence\_flow\_condition\_type}$

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 $\textsc{sequence\_flow}$

AX_361 has_sequence_flow_condition_type has range *xsd:string*{"None","Expression","Default"}

AX_362 $\textsc{sequence\_flow} \sqsubseteq (\neg\exists\mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcup ((\exists\mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcap ((=1)\mathsf{has\_sequence\_flow\_condition\_expression}))$

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–Subject to any constraints imposed by a Source that is a Gateway.

AX_363 has_sequence_flow_condition_expression has domain $\textsc{sequence\_flow}$

AX_364 has_sequence_flow_condition_expression has range $\textsc{expression}$

AX_365 $\mathsf{has\_sequence\_flow\_source\_ref} \sqsubseteq \mathsf{has\_connecting\_object\_source\_ref}$

AX_366 $\mathsf{has\_sequence\_flow\_target\_ref} \sqsubseteq \mathsf{has\_connecting\_object\_target\_ref}$

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 $\textsc{sequence\_flow}$

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.

AX_368 has_sequence_flow_target_ref has domain SEQUENCE_FLOW

AX_369 has_sequence_flow_source_ref_inv = $\text{has\_sequence\_flow\_source\_ref}^{-1}$

AX_370 has_sequence_flow_target_ref_inv = $\text{has\_sequence\_flow\_target\_ref}^{-1}$

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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq$ has_connecting_object_source_ref

AX_375 has_message_flow_target_ref $\sqsubseteq$ 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 = $\text{has\_message\_flow\_source\_ref}^{-1}$

AX_379 has_message_flow_target_ref_inv = $\text{has\_message\_flow\_target\_ref}^{-1}$

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 ASSOCIATION $\sqsubseteq (=1)$has_association_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 has_association_direction has domain ASSOCIATION

AX_382 has_association_direction has range *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 276), Roles (see Section B.11.16 on page 276), Conditions (see Section B.11.5 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 SUPPORTING_ELEMENT $\equiv$ PROCESS $\sqcup$ MESSAGE $\sqcup$ CONDITION $\sqcup$ EVENT_DETAIL $\sqcup$ ASSIGNMENT $\sqcup$ EXPRESSION $\sqcup$ PROPERTY $\sqcup$ TRANSACTION $\sqcup$ GATE $\sqcup$ WEB_SERVICE $\sqcup$ ROLE $\sqcup$ ENTITY $\sqcup$ PARTICIPANT $\sqcup$ CATEGORY $\sqcup$ OUTPUT_SET $\sqcup$ INPUT_SET

AX_384 PROCESS $\sqsubseteq \neg$MESSAGE

AX_385 PROCESS $\sqsubseteq \neg$CONDITION

AX_386 PROCESS $\sqsubseteq \neg$EVENT_DETAIL

AX_387 PROCESS $\sqsubseteq \neg$ASSIGNMENT

AX_388 PROCESS $\sqsubseteq \neg$EXPRESSION

AX_389 PROCESS $\sqsubseteq \neg$PROPERTY

AX_390 PROCESS $\sqsubseteq \neg$TRANSACTION

AX_391 PROCESS $\sqsubseteq \neg$GATE

AX_392 PROCESS $\sqsubseteq \neg$WEB_SERVICE

AX_393 PROCESS $\sqsubseteq \neg$ROLE

AX_394 PROCESS $\sqsubseteq \neg$ENTITY

AX_395 PROCESS $\sqsubseteq \neg$PARTICIPANT

AX_396 PROCESS $\sqsubseteq \neg$CATEGORY

AX_397 PROCESS $\sqsubseteq \neg$OUTPUT_SET

AX_398 PROCESS $\sqsubseteq \neg$INPUT_SET

AX_399 MESSAGE $\sqsubseteq \neg$CONDITION

AX_400 MESSAGE $\sqsubseteq \neg$EVENT_DETAIL

AX_401 MESSAGE $\sqsubseteq \neg$ASSIGNMENT

AX_402 MESSAGE $\sqsubseteq \neg$EXPRESSION

AX_403 MESSAGE $\sqsubseteq \neg$PROPERTY

AX_404 MESSAGE $\sqsubseteq \neg$TRANSACTION

AX_405 MESSAGE $\sqsubseteq \neg$GATE

AX_406 MESSAGE $\sqsubseteq \neg$WEB_SERVICE

AX_407 MESSAGE $\sqsubseteq \neg$ROLE

AX_408 MESSAGE $\sqsubseteq \neg$ENTITY

AX_409 MESSAGE $\sqsubseteq \neg$PARTICIPANT

AX_410 MESSAGE $\sqsubseteq \neg$CATEGORY

AX_411 MESSAGE $\sqsubseteq \neg$OUTPUT_SET

AX_412 MESSAGE $\sqsubseteq \neg$INPUT_SET

AX_413 CONDITION $\sqsubseteq \neg$EVENT_DETAIL

AX_414 CONDITION $\sqsubseteq \neg$ASSIGNMENT

AX_415 CONDITION $\sqsubseteq \neg$EXPRESSION

AX_416 CONDITION $\sqsubseteq \neg$PROPERTY

AX_417 CONDITION $\sqsubseteq \neg$TRANSACTION

AX_418 CONDITION $\sqsubseteq \neg$GATE

AX_419 CONDITION $\sqsubseteq \neg$WEB_SERVICE

AX_420 CONDITION $\sqsubseteq \neg$ROLE

AX_421 CONDITION $\sqsubseteq \neg$ENTITY

AX_422 CONDITION $\sqsubseteq \neg$PARTICIPANT

AX_423 CONDITION $\sqsubseteq \neg$CATEGORY

AX_424 CONDITION $\sqsubseteq \neg$OUTPUT_SET

AX_425 CONDITION $\sqsubseteq \neg$INPUT_SET

AX_426 EVENT_DETAIL $\sqsubseteq \neg$ASSIGNMENT

AX_427 EVENT_DETAIL $\sqsubseteq \neg$EXPRESSION

AX_428 EVENT_DETAIL $\sqsubseteq \neg$PROPERTY

AX_429 EVENT_DETAIL $\sqsubseteq \neg$TRANSACTION

AX_430 EVENT_DETAIL $\sqsubseteq \neg$GATE

AX_431 EVENT_DETAIL $\sqsubseteq \neg$WEB_SERVICE

AX_432 EVENT_DETAIL $\sqsubseteq \neg$ROLE

AX_433 EVENT_DETAIL $\sqsubseteq \neg$ENTITY

AX_434 EVENT_DETAIL $\sqsubseteq \neg$PARTICIPANT

AX_435 EVENT_DETAIL $\sqsubseteq \neg$CATEGORY

AX_436 EVENT_DETAIL $\sqsubseteq \neg$OUTPUT_SET

AX_437 EVENT_DETAIL $\sqsubseteq \neg$INPUT_SET

AX_438 ASSIGNMENT $\sqsubseteq \neg$EXPRESSION

AX_439 ASSIGNMENT $\sqsubseteq \neg$PROPERTY

AX_440 ASSIGNMENT $\sqsubseteq \neg$TRANSACTION

AX_441 ASSIGNMENT $\sqsubseteq$ ¬GATE

AX_442 ASSIGNMENT $\sqsubseteq$ ¬WEB_SERVICE

AX_443 ASSIGNMENT $\sqsubseteq$ ¬ROLE

AX_444 ASSIGNMENT $\sqsubseteq$ ¬ENTITY

AX_445 ASSIGNMENT $\sqsubseteq$ ¬PARTICIPANT

AX_446 ASSIGNMENT $\sqsubseteq$ ¬CATEGORY

AX_447 ASSIGNMENT $\sqsubseteq$ ¬OUTPUT_SET

AX_448 ASSIGNMENT $\sqsubseteq$ ¬INPUT_SET

AX_449 EXPRESSION $\sqsubseteq$ ¬PROPERTY

AX_450 EXPRESSION $\sqsubseteq$ ¬TRANSACTION

AX_451 EXPRESSION $\sqsubseteq$ ¬GATE

AX_452 EXPRESSION $\sqsubseteq$ ¬WEB_SERVICE

AX_453 EXPRESSION $\sqsubseteq$ ¬ROLE

AX_454 EXPRESSION $\sqsubseteq$ ¬ENTITY

AX_455 EXPRESSION $\sqsubseteq$ ¬PARTICIPANT

AX_456 EXPRESSION $\sqsubseteq$ ¬CATEGORY

AX_457 EXPRESSION $\sqsubseteq$ ¬OUTPUT_SET

AX_458 EXPRESSION $\sqsubseteq$ ¬INPUT_SET

AX_459 PROPERTY $\sqsubseteq$ ¬TRANSACTION

AX_460 PROPERTY $\sqsubseteq$ ¬GATE

AX_461 PROPERTY $\sqsubseteq$ ¬WEB_SERVICE

AX_462 PROPERTY $\sqsubseteq$ ¬ROLE

AX_463 PROPERTY $\sqsubseteq$ ¬ENTITY

AX_464 PROPERTY $\sqsubseteq$ ¬PARTICIPANT

AX_465 PROPERTY $\sqsubseteq$ ¬CATEGORY

AX_466 PROPERTY $\sqsubseteq$ ¬OUTPUT_SET

AX_467 PROPERTY $\sqsubseteq$ ¬INPUT_SET

AX_468 TRANSACTION $\sqsubseteq$ ¬GATE

AX_469 TRANSACTION $\sqsubseteq$ ¬WEB_SERVICE

AX_470 TRANSACTION $\sqsubseteq$ ¬ROLE

AX_471 TRANSACTION $\sqsubseteq$ ¬ENTITY

AX_472 TRANSACTION $\sqsubseteq$ ¬PARTICIPANT

AX_473 TRANSACTION $\sqsubseteq$ ¬CATEGORY

AX_474 TRANSACTION $\sqsubseteq$ ¬OUTPUT_SET

AX_475 TRANSACTION $\sqsubseteq$ ¬INPUT_SET

AX_476 GATE $\sqsubseteq$ ¬WEB_SERVICE

AX_477 GATE $\sqsubseteq$ ¬ROLE

AX_478 GATE $\sqsubseteq$ ¬ENTITY

AX_479 GATE $\sqsubseteq$ ¬PARTICIPANT

AX_480 GATE $\sqsubseteq$ ¬CATEGORY

AX_481 GATE $\sqsubseteq$ ¬OUTPUT_SET

AX_482 GATE $\sqsubseteq$ ¬INPUT_SET

AX_483 WEB_SERVICE $\sqsubseteq \neg$ROLE

AX_484 WEB_SERVICE $\sqsubseteq \neg$ENTITY

AX_485 WEB_SERVICE $\sqsubseteq \neg$PARTICIPANT

AX_486 WEB_SERVICE $\sqsubseteq \neg$CATEGORY

AX_487 WEB_SERVICE $\sqsubseteq \neg$OUTPUT_SET

AX_488 WEB_SERVICE $\sqsubseteq \neg$INPUT_SET

AX_489 ROLE $\sqsubseteq \neg$ENTITY

AX_490 ROLE $\sqsubseteq \neg$PARTICIPANT

AX_491 ROLE $\sqsubseteq \neg$CATEGORY

AX_492 ROLE $\sqsubseteq \neg$OUTPUT_SET

AX_493 ROLE $\sqsubseteq \neg$INPUT_SET

AX_494 ENTITY $\sqsubseteq \neg$PARTICIPANT

AX_495 ENTITY $\sqsubseteq \neg$CATEGORY

AX_496 ENTITY $\sqsubseteq \neg$OUTPUT_SET

AX_497 ENTITY $\sqsubseteq \neg$INPUT_SET

AX_498 PARTICIPANT $\sqsubseteq \neg$CATEGORY

AX_499 PARTICIPANT $\sqsubseteq \neg$OUTPUT_SET

AX_500 PARTICIPANT $\sqsubseteq \neg$INPUT_SET

AX_501 CATEGORY $\sqsubseteq \neg$OUTPUT_SET

AX_502 CATEGORY $\sqsubseteq \neg$INPUT_SET

AX_503 OUTPUT_SET $\sqsubseteq \neg$INPUT_SET

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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq (\geq 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$has_condition_name $\sqcup (= 1)$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_condition_expression has domain CONDITION

AX_532 has_condition_condition_expression 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 $\sqsubseteq (= 1)$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 $\equiv$ {*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 $\sqsubseteq (= 1)$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 EventTypes (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
$AX\_540\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{compensation\_event\_detail\_type})$
$AX\_541\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{link\_event\_detail\_type})$
$AX\_542\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{error\_event\_detail\_type})$
$AX\_543\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{conditional\_event\_detail\_type})$
$AX\_544\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{message\_event\_detail\_type})$
$AX\_545\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$
$AX\_546\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$
$AX\_547\ (\neg\{\mathit{cancel\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$
$AX\_548\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{link\_event\_detail\_type})$
$AX\_549\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{error\_event\_detail\_type})$
$AX\_550\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{conditional\_event\_detail\_type})$
$AX\_551\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{message\_event\_detail\_type})$
$AX\_552\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$
$AX\_553\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$
$AX\_554\ (\neg\{\mathit{compensation\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$
$AX\_555\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{error\_event\_detail\_type})$
$AX\_556\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{conditional\_event\_detail\_type})$
$AX\_557\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{message\_event\_detail\_type})$
$AX\_558\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$
$AX\_559\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$
$AX\_560\ (\neg\{\mathit{link\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$
$AX\_561\ (\neg\{\mathit{error\_event\_detail\_type}\})(\mathit{conditional\_event\_detail\_type})$

AX_562 $(\neg\{\mathit{error\_event\_detail\_type}\})(\mathit{message\_event\_detail\_type})$

AX_563 $(\neg\{\mathit{error\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$

AX_564 $(\neg\{\mathit{error\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$

AX_565 $(\neg\{\mathit{error\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$

AX_566 $(\neg\{\mathit{conditional\_event\_detail\_type}\})(\mathit{message\_event\_detail\_type})$

AX_567 $(\neg\{\mathit{conditional\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$

AX_568 $(\neg\{\mathit{conditional\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$

AX_569 $(\neg\{\mathit{conditional\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$

AX_570 $(\neg\{\mathit{message\_event\_detail\_type}\})(\mathit{terminate\_event\_detail\_type})$

AX_571 $(\neg\{\mathit{message\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$

AX_572 $(\neg\{\mathit{message\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$

AX_573 $(\neg\{\mathit{terminate\_event\_detail\_type}\})(\mathit{timer\_event\_detail\_type})$

AX_574 $(\neg\{\mathit{terminate\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$

AX_575 $(\neg\{\mathit{timer\_event\_detail\_type}\})(\mathit{signal\_event\_detail\_type})$

AX_576 CANCEL_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap\, \exists \mathsf{has\_event\_detail\_type}.\{\mathit{cancel\_event\_detail\_type}\}$

Class: CANCEL_EVENT_DETAIL

Label: Cancel Event Detail

AX_577 CONDITIONAL_EVENT_DETAIL $\equiv$ EVENT_DETAIL$\sqcap\exists \mathsf{has_event_detail_type}.\{\mathit{conditional_event_detail_type}\}$

Class: CONDITIONAL_EVENT_DETAIL

Label: Conditional Event Detail

AX_578 CONDITIONAL_EVENT_DETAIL $\sqsubseteq (=1)\mathsf{has\_conditional\_event\_condition\_ref}$

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 $\equiv$ EVENT_DETAIL$\sqcap\exists \mathsf{has_event_detail_type}.\{\mathit{compensation_event_detail_type}\}$

Class: COMPENSATION_EVENT_DETAIL

Label: Compensation Event Detail

AX_582 COMPENSATION_EVENT_DETAIL $\sqsubseteq (\geq 1)\mathsf{has\_activity\_ref}$

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 has_activity_ref has domain COMPENSATION_EVENT_DETAIL

AX_584 has_activity_ref has range ACTIVITY

AX_585 ERROR_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.$\{error\_event\_detail\_type\}$

Class: ERROR_EVENT_DETAIL

Label: Error Event Detail

AX_586 ERROR_EVENT_DETAIL $\sqsubseteq (\geq 1)$has_error_detail_error_code

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 has_error_detail_error_code has domain ERROR_EVENT_DETAIL

AX_588 has_error_detail_error_code has range *xsd:string*

AX_589 LINK_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.$\{link\_event\_detail\_type\}$

Class: LINK_EVENT_DETAIL

Label: Link Event Detail

AX_590 LINK_EVENT_DETAIL $\sqsubseteq (= 1)$has_link_event_name

Property: has_link_event_name

Label: Name

Description: If the Trigger is a Link, then the Name MUST be entered.

AX_591 has_link_event_name has domain LINK_EVENT_DETAIL

AX_592 has_link_event_name has range *xsd:string*

AX_593 MESSAGE_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.$\{message\_event\_detail\_type\}$

Class: MESSAGE_EVENT_DETAIL

Label: Message Event Detail

AX_594 MESSAGE_EVENT_DETAIL $\sqsubseteq (= 1)$has_message_event_message_ref

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 $\sqsubseteq$ $(= 1)$has_message_event_implementation

Property: has_message_event_implementation

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_implementation has domain MESSAGE_EVENT_DETAIL

*AX*_599 has_message_event_implementation has range *xsd:string*{"Web_Service","Other","Unspecified"}

*AX*_600 SIGNAL_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.{*signal_event_detail_type*}

Class: SIGNAL_EVENT_DETAIL

Label: Signal Event Detail

*AX*_601 SIGNAL_EVENT_DETAIL $\sqsubseteq$ $(= 1)$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 $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.{*terminate_event_detail_type*}

Class: TERMINATE_EVENT_DETAIL

Label: Terminate Event Detail

*AX*_605 TIMER_EVENT_DETAIL $\equiv$ EVENT_DETAIL $\sqcap$ $\exists$has_event_detail_type.{*timer_event_detail_type*}

Class: TIMER_EVENT_DETAIL

Label: Timer Event Detail

*AX*_606 TIMER_EVENT_DETAIL $\sqsubseteq$ $(= 1)$has_timer_event_time_date $\sqcup$ $(= 1)$has_timer_event_time_cycle

Property: has_timer_event_time_date

Label: TimeDate

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

AX_607 has_timer_event_time_date has domain TIMER_EVENT_DETAIL

AX_608 has_timer_event_time_date has range TIME_DATE_EXPRESSION

Property: has_timer_event_time_cycle

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).

AX_609 has_timer_event_time_cycle has domain TIMER_EVENT_DETAIL

AX_610 has_timer_event_time_cycle has range 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).

AX_611 EXPRESSION $\sqsubseteq (= 1)$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.

AX_612 has_expression_expression_body has domain EXPRESSION

AX_613 has_expression_expression_body has range *xsd:string*

AX_614 EXPRESSION $\sqsubseteq (= 1)$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.

AX_615 has_expression_expression_language has domain EXPRESSION

AX_616 has_expression_expression_language has range *xsd:string*

AX_617 TIME_DATE_EXPRESSION $\sqsubseteq$ EXPRESSION

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).

AX_618 $\textsc{gate} \sqsubseteq (= 1)\mathsf{has\_gate\_outgoing\_sequence\_flow\_ref}$

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 $\textsc{input\_set} \sqsubseteq (\exists \mathsf{has\_input\_set\_artifact\_input}.\textsc{artifact\_input}) \sqcup (\exists \mathsf{has\_input\_set\_property\_input}.\textsc{property})$

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 ArtifactInput 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 MESSAGE $\sqsubseteq (= 1)$has_message_name

Property: has_message_name

Label: Name

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

AX_629 has_message_name has domain MESSAGE

AX_630 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 has_message_property has domain MESSAGE

AX_632 has_message_property has range PROPERTY

AX_633 MESSAGE $\sqsubseteq (= 1)$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 has_message_from_ref has domain MESSAGE

AX_635 has_message_from_ref has range PARTICIPANT

AX_636 MESSAGE $\sqsubseteq (= 1)$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 has_message_to_ref has domain MESSAGE

AX_638 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 OBJECT $\sqsubseteq (= 1)$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.

*AX*_640 has_object_id has range *xsd:string*

*AX*_641 has_object_id has domain OBJECT

Class: OUTPUT_SET

Label: Output Set

Description: 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 OUTPUT_SET $\sqsubseteq$ ($\exists$has_output_set_artifact_output.ARTIFACT_OUTPUT) $\sqcup$ ($\exists$has_output_set_property_output.PROPERTY)

Property: has_output_set_artifact_output

Label: ArtifactOutput

Description: 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 has_output_set_artifact_output has domain OUTPUT_SET

*AX*_644 has_output_set_artifact_output has range ARTIFACT_OUTPUT

Property: has_output_set_property_output

Label: PropertyOutput

Description: 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 has_output_set_property_output has domain OUTPUT_SET

*AX*_646 has_output_set_property_output has range PROPERTY

Class: PARTICIPANT

Label: Participant

Description: 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 PARTICIPANT $\sqsubseteq$ $(= 1)$has_participant_participant_type

Property: has_participant_participant_type

Label: ParticipantType

Description:

*AX*_648 has_participant_participant_type has range *xsd:string*{"Role","Entity"}

AX_649 has_participant_participant_type has domain PARTICIPANT

AX_650 PARTICIPANT $\sqsubseteq$ ($\exists$has_participant_participant_type.$\{$"Role"$\}$ $\sqcap$ $(= 1)$has_participant_role_ref) $\sqcup$ ($\exists$has_participant_participant_type.$\{$"Entity"$\}$ $\sqcap$ $(= 1)$has_participant_entity_ref)

Property: has_participant_role_ref

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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ $(= 1)$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 $\sqsubseteq$ (≥ 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 $\sqsubseteq$ ($\geq$ 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 has_property_correlation has domain PROPERTY

*AX*_666 has_property_correlation 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 ROLE $\sqsubseteq$ (= 1)has_role_name

Property: has_role_name

Label: Name

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

*AX*_668 has_role_name has domain ROLE

*AX*_669 has_role_name 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 SIGNAL $\sqsubseteq$ (= 1)has_signal_name

Property: has_signal_name

Label: Name

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

*AX*_671 has_signal_name has domain SIGNAL

*AX*_672 has_signal_name 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 has_signal_property has domain SIGNAL

*AX*_674 has_signal_property 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 TRANSACTION $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (\leq 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (= 1)$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 $\sqsubseteq (=1)$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 $\sqsubseteq (\exists$has_process_ad_hoc.$\{$"false"$\}) \sqcup (\exists$has_process_ad_hoc.$\{$"true"$\} \sqcap$
$(=1)$has_process_ad_hoc_ordering $\sqcap (=1)$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 [?]

AX_722 SEQUENCE_FLOW $\sqsubseteq \forall$has_connecting_object_source_ref.(INTERMEDIATE_EVENT $\sqcup$ START_EVENT $\sqcup$ TASK $\sqcup$ SUB_PROCESS $\sqcup$ GATEWAY)

AX_723 SEQUENCE_FLOW $\sqsubseteq \forall$has_connecting_object_target_ref.(INTERMEDIATE_EVENT $\sqcup$ END_EVENT $\sqcup$ TASK $\sqcup$ SUB_PROCESS $\sqcup$ GATEWAY)

AX_724 $\textsc{message\_flow} \sqsubseteq \forall \mathsf{has\_connecting\_object\_source\_ref}.((\textsc{intermediate\_event} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{message\_event\_detail}) \sqcup (\textsc{end\_event} \sqcap \exists \mathsf{has\_end\_event\_result}.\textsc{message\_event\_detail}) \sqcup \textsc{task} \sqcup \textsc{sub\_process} \sqcup \textsc{pool})$

AX_725 $\textsc{message\_flow} \sqsubseteq \forall \mathsf{has\_connecting\_object\_target\_ref}.((\textsc{intermediate\_event} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{message\_event\_detail}) \sqcup (\textsc{start\_event} \sqcap \exists \mathsf{has\_start\_event\_trigger}.\textsc{message\_event\_detail}) \sqcup \textsc{task} \sqcup \textsc{sub\_process} \sqcup \textsc{pool})$

AX_726 $\textsc{activity} \sqsubseteq (\forall \mathsf{has\_flow\_object\_assignment}.(\exists \mathsf{has\_assignment\_assign\_time}.\{\text{"Start"}\} \sqcup \exists \mathsf{has\_assignment\_assign\_time}.\{\text{"End"}\}))$

AX_727 $\textsc{start\_event} \sqsubseteq \exists \mathsf{has\_connecting\_object\_source\_ref\_inv}.(\textsc{sequence\_flow})$

AX_728 $\textsc{start\_event} \sqsubseteq \forall \mathsf{has\_connecting\_object\_source\_ref\_inv}.(\textsc{sequence\_flow} \sqcap \exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"None"}\})$

AX_729 $\textsc{none\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap \neg \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{event\_detail}$

AX_730 $\textsc{cancel\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{cancel\_event\_detail}$

AX_731 $\textsc{compensation\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{compensation\_event\_detail}$

AX_732 $\textsc{link\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{link\_event\_detail}$

AX_733 $\textsc{error\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{error\_event\_detail}$

AX_734 $\textsc{conditional\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{conditional\_event\_detail}$

AX_735 $\textsc{message\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{message\_event\_detail}$

AX_736 $\textsc{timer\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{timer\_event\_detail}$

AX_737 $\textsc{signal\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (=1)\mathsf{has\_intermediate\_event\_trigger} \sqcap \exists \mathsf{has\_intermediate\_event\_trigger}.\textsc{signal\_event\_detail}$

AX_738 $\textsc{multiple\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap (\leq 2)\mathsf{has\_intermediate\_event\_trigger}$

AX_739 $\textsc{activity\_boundary\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap \exists \mathsf{has\_intermediate\_event\_target}.\textsc{activity}$

AX_740 $\textsc{not\_activity\_boundary\_intermediate\_event} \equiv \textsc{intermediate\_event} \sqcap \neg \exists \mathsf{has\_intermediate\_event\_target}.\textsc{activity}$

AX_741 $\textsc{activity\_boundary\_intermediate\_event} \sqsubseteq (\textsc{cancel\_intermediate\_event} \sqcup \textsc{compensation\_intermediate\_event} \sqcup \textsc{error\_intermediate\_event} \sqcup \textsc{conditional\_intermediate\_event} \sqcup \textsc{message\_intermediate\_event} \sqcup \textsc{timer\_intermediate\_event} \sqcup \textsc{signal\_intermediate\_event} \sqcup \textsc{multiple\_intermediate\_event})$

AX_742 $\textsc{activity\_boundary\_intermediate\_event} \sqsubseteq (\exists \mathsf{has\_intermediate\_event\_target}.(\textsc{sub\_process} \sqcap \exists \mathsf{has\_sub\_process\_is\_a\_transaction}.\{\text{"true"}\})) \sqcup ((\neg \exists \mathsf{has\_intermediate\_event\_target}.(\textsc{sub\_process} \sqcap \exists \mathsf{has\_sub\_process\_is\_a\_transaction}.\{\text{"true"}\})) \sqcap (\neg \textsc{cancel\_intermediate\_event}))$

AX_743 $\textsc{activity\_boundary\_intermediate\_event} \sqsubseteq \neg \exists \mathsf{has\_connecting\_object\_target\_ref\_inv}.\textsc{sequence\_flow}$

AX_744 $\textsc{activity\_boundary\_intermediate\_event} \sqsubseteq (\neg \textsc{compensation\_intermediate\_event} \sqcap ((=1)\mathsf{has\_sequence\_flow\_source\_ref\_inv})) \sqcup (\textsc{compensation\_intermediate\_event} \sqcap \neg \exists \mathsf{has\_sequence\_flow\_source\_ref\_inv}.\textsc{sequence\_flow})$

AX_745 $\textsc{not\_activity\_boundary\_intermediate\_event} \sqsubseteq (\textsc{none\_intermediate\_event} \sqcup \textsc{compensation\_intermediate\_event} \sqcup \textsc{link\_intermediate\_event} \sqcup \textsc{conditional\_intermediate\_event} \sqcup$

MESSAGE_INTERMEDIATE_EVENT ⊔ TIMER_INTERMEDIATE_EVENT ⊔ SIGNAL_INTERMEDIATE_EVENT)

AX_746 NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT ⊑ (¬(NONE_INTERMEDIATE_EVENT ⊔ COMPENSATION_INTERMEDIATE_EVENT)⊓(≥ 1)has_sequence_flow_target_ref_inv)⊔((NONE_INTERMEDIATE_EVENT⊔ COMPENSATION_INTERMEDIATE_EVENT) ⊓ (= 1)has_sequence_flow_target_ref_inv)

AX_747 NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT ⊑ (LINK_INTERMEDIATE_EVENT) ⊔ (¬LINK_INTERMEDIATE_EVENT ⊓ (= 1)has_sequence_flow_source_ref_inv)

AX_748 NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT ⊑ (¬LINK_INTERMEDIATE_EVENT) ⊔ (LINK_INTERMEDIATE_EVENT ⊓ (¬(∃has_sequence_flow_source_ref_inv.SEQUENCE_FLOW ⊓ ∃has_sequence_flow_target_ref_inv.SEQUENCE_FLOW)))

AX_749 INTERMEDIATE_EVENT ⊑ ((¬MESSAGE_INTERMEDIATE_EVENT⊓(= 0)has_message_flow_source_ref_inv⊓ (= 0)has_message_flow_target_ref_inv)⊔(MESSAGE_INTERMEDIATE_EVENT⊓(((≥ 1)has_message_flow_source_ref_inv⊓ (= 0)has_message_flow_target_ref_inv)⊔((= 0)has_message_flow_source_ref_inv⊓(≥ 1)has_message_flow_target_ref_inv))))

AX_750 END_EVENT ⊑ (¬∃has_end_event_result.ERROR_EVENT_DETAIL)⊔(∃has_end_event_result.(ERROR_EVENT_DETAIL⊓ (= 1)has_error_detail_error_code))

AX_751 NOT_ACTIVITY_BOUNDARY_INTERMEDIATE_EVENT ⊑ (¬ERROR_INTERMEDIATE_EVENT) ⊔ (∃has_intermediate_event_trigger.(ERROR_EVENT_DETAIL ⊓ (= 1)has_error_detail_error_code))

AX_752 RECEIVE_TASK ⊑ (∃has_receive_task_instantiate.{"false"}) ⊔ (∃has_receive_task_instantiate.{"true"} ⊓ ¬∃has_activity_loop_type.LOOP_TYPES)

AX_753 RECEIVE_TASK ⊑ ¬∃has_connecting_object_source_ref_inv.MESSAGE_FLOW

AX_754 SEND_TASK ⊑ ¬∃has_connecting_object_target_ref_inv.MESSAGE_FLOW

AX_755 SCRIPT_TASK ⊑ ¬(∃has_connecting_object_target_ref_inv.MESSAGE_FLOW ⊔ ∃has_connecting_object_source_ref_inv.MESSAGE_FLOW)

AX_756 MANUAL_TASK ⊑ ¬(∃has_connecting_object_target_ref_inv.MESSAGE_FLOW ⊔ ∃has_connecting_object_source_ref_inv.MESSAGE_FLOW)

AX_757 GATEWAY ⊑ (≤ 2)has_sequence_flow_target_ref_inv⊔((≥ 1)has_sequence_flow_target_ref_inv⊓(≤ 2)has_gateway_gate)

AX_758 EVENT_BASED_EXCLUSIVE_GATEWAY ⊑ (≤ 2)has_gateway_gate

AX_759 has_gateway_gate_inv = has_gateway_gate^{-1}

AX_760 has_inclusive_gateway_default_gate_inv = has_inclusive_gateway_default_gate^{-1}

AX_761 has_data_based_exclusive_gateway_default_gate_inv = has_data_based_exclusive_gateway_default_gate^{-1}

AX_762 GATE ⊑ (= 1)has_gateway_gate_inv

AX_763 GATE ⊑ (∃has_gateway_gate_inv.(¬EVENT_BASED_EXCLUSIVE_GATEWAY)) ⊔ (∃has_gateway_gate_inv.EVENT_BASED_EXCLUSIVE_GATEWAY ⊓ ∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"None"}))

AX_764 GATE ⊑ (∃has_gateway_gate_inv.(¬COMPLEX_GATEWAY))⊔(∃has_gateway_gate_inv.COMPLEX_GATEWAY⊓ ∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"None"}))

AX_765 GATE ⊑ (∃has_gateway_gate_inv.(¬PARALLEL_GATEWAY))⊔(∃has_gateway_gate_inv.PARALLEL_GATEWAY⊓ ∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"None"}))

AX_766 GATE ⊑ (∃has_gateway_gate_inv.(¬INCLUSIVE_GATEWAY))⊔(∃has_gateway_gate_inv.INCLUSIVE_GATEWAY⊓ ((= 1)has_gateway_gate_inv⊓∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"None"}))⊔ ((≤ 2)has_gateway_gate_inv⊓∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"Expression"})))

AX_767 GATE ⊑ (∃has_gateway_gate_inv.(¬DATA_BASED_EXCLUSIVE_GATEWAY)) ⊔ (∃has_gateway_gate_inv.DATA_BASED_EXCLUSIVE_GATEWAY ⊓ ((= 1)has_gateway_gate_inv ⊓ ∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"None"}))⊔((≤ 2)has_gateway_gate_inv⊓ ∃has_gate_outgoing_sequence_flow_ref.(∃has_sequence_flow_condition_type.{"Expression"})))

AX_768 EVENT_BASED_EXCLUSIVE_GATEWAY ⊑ (∀has_gateway_gate.(∃has_gate_outgoing_sequence_flow_ref. (∃has_connecting_object_target_ref.(RECEIVE_TASK⊔TIMER_INTERMEDIATE_EVENT⊔SIGNAL_INTERMEDIATE_EVENT))))⊔

$(\forall \mathsf{has\_gateway\_gate}.(\exists \mathsf{has\_gate\_outgoing\_sequence\_flow\_ref}.(\exists \mathsf{has\_connecting\_object\_target\_ref}.$ $(\textsc{message\_intermediate\_event} \sqcup \textsc{timer\_intermediate\_event} \sqcup \textsc{signal\_intermediate\_event}))))$

AX_769 $\textsc{sequence\_flow} \sqsubseteq (\neg\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcup$ $((\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcap \forall \mathsf{has\_connecting\_object\_source\_ref}.(\neg\textsc{event}))$

AX_770 $\textsc{sequence\_flow} \sqsubseteq (\neg\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcup$ $((\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcap \forall \mathsf{has\_connecting\_object\_source\_ref}.(\neg\textsc{parallel\_gateway}))$

AX_771 $\textsc{activity} \sqsubseteq (\neg\exists \mathsf{has\_sequence\_flow\_source\_ref\_inv}.(\textsc{sequence\_flow}$ $\sqcap \exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\})) \sqcup ((\exists \mathsf{has\_sequence\_flow\_source\_ref\_inv}.(\textsc{sequence\_flow} \sqcap$ $\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\})) \sqcap (\leq 2)\mathsf{has\_sequence\_flow\_source\_ref\_inv})$

AX_772 $\textsc{sequence\_flow} \sqsubseteq (\exists \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{data\_based\_exclusive\_gateway} \sqcup$ $\textsc{inclusive\_gateway}) \sqcap \neg\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"None"}\}) \sqcup$ $(\neg\exists \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{data\_based\_exclusive\_gateway} \sqcup \textsc{inclusive\_gateway}))$

AX_773 $\textsc{sequence\_flow} \sqsubseteq (\neg\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcup$ $((\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Expression"}\}) \sqcap \forall \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{task} \sqcup \textsc{sub\_process} \sqcup$ $\textsc{data\_based\_exclusive\_gateway} \sqcup \textsc{inclusive\_gateway}))$

AX_774 $\textsc{sequence\_flow} \sqsubseteq (\neg\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Default"}\}) \sqcup$ $((\exists \mathsf{has\_sequence\_flow\_condition\_type}.\{\text{"Default"}\}) \sqcap \forall \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{activity} \sqcup$ $\textsc{data\_based\_exclusive\_gateway}))$

AX_775 $\textsc{association} \sqsubseteq (\exists \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{artifact}) \sqcap \exists \mathsf{has\_connecting\_object\_target\_ref}.$ $(\textsc{flow\_object})) \sqcup (\exists \mathsf{has\_connecting\_object\_target\_ref}.(\textsc{artifact}) \sqcap \exists \mathsf{has\_connecting\_object\_source\_ref}.(\textsc{flow\_object}))$