Modeling BPMN 1.x in ARIS

ARIS

Version 9.8 - Service Release 2

October 2015

Product Documentation

Contents

1 Introduction

1.1 Purpose of this document

The BPMN specification defines a set of semantic objects which can be placed inside Business Process Diagrams. Each of those objects can define various attributes. All objects/attributes as a whole can be considered as the BPMN metamodel.

An application that is BPMN compliant has to provide methods to maintain all mandatory BPMN attributes (and therefore provide a way for the user to model any valid BPMN Metamodel)

The BPMN Specification does not specify how a tool should provide those mechanisms, it's the decision of the tool vendor how the (non graphical) BPMN attributes are represented in a BPMN compliant tool.

The following document describes the ARIS implementation (mapping of BPMN attributes to ARIS concepts) of the BPMN 1.0 (final adopted) specification.

For each BPMN Attribute the exact ARIS representation is described, this enables ARIS users to:

Model the complete BPMN semantic with ARIS

Determine the exact semantic of the BPMN models which are stored inside an ARIS repository

Transform the BPMN Models into BPEL XML via ARIS Reports

Export the BPMN Models / Metamodels via ARIS reports into a vendor independent format

In some situations BPMN attributes were mapped to more advanced ARIS modeling techniques. In particular various BPMN semantic definitions which are represented as BPMN Attribute are mapped to one or more of the following ARIS concepts:

ARIS connection which special semantic (e.g. Property definition, property assignments etc…)

Multiple occurrences of the same ARIS object definition (e.g. Link events, Referenced Sub Processes etc…)

Model assignments (e.g. Independent Sub Process modeling)

It is recommended that the reader of this document has a basic understanding of the above ARIS concepts.

1.2 Mapping Description

The following attribute tables were extracted from the BPMN 1.0 specification and were afterwards extended to describe the ARIS mapping.

The first column contains the BPMN Attribute definition.

If a BPMN Attribute is mapped directly to one/multiple ARIS attributes (that can be maintained at the corresponding ARIS object) the corresponding ARIS attribute names and their API-Names (together with their attribute number – which can be useful for report developers) are described in the second column.
If the mapping doesn't specify an explicit semantic for unmaintained ARIS attributes, the ARIS value **not maintained / empty** is the semantic equivalent to the BPMN default value of the corresponding attribute.

If a special BPMN attribute is mapped to one of the above advanced ARIS concepts the description of the mapping can be found in the third column.BPMN Element Mapping to ARIS Objects

1.3 Business Process Diagram Attributes

The BPMN Element **Business Process Diagram** is represented by the ARIS Model **Business process diagram (BPMN)**

The attributes of a BPD are represented as ARIS model attributes.

Table 1.3-1: Business Process Diagram Attributes

BPMN Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Id : Object		ARIS Global Unique Identifier (GUID) for the Model
Name : String	**Name**, AT_NAME, (1)	
Version (0-1) : String	**Release**, AT_REL_1 (92)	
Author (0-1) : String	**Author**, AT_AUTH (46)	
Language (0-1) : String	**Languages**, AT_LNG (379)	
ExpressionLanguage (0-1) : String	**Expression language**, AT_BPMN_EXPRESSION_LANG (3409)	
QueryLanguage (0-1) : String	**Query language** AT_BPMN_QUERY_LANG (3410)	
CreationDate (0-1) : Date	**Time of generation** AT_CREAT_TIME_STMP (1008)	
ModificationDate (0-1) : Date	**Last change** AT_LAST_CHNG_2 (1175)	

BPMN Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Pools (1-n) : Pool		1. Pools with a visible border: The Pool Objects (AT_BPMN_POOL) are placed inside the BPD diagram. 2. The invisible pool inside a BPD diagram D1: A Pool object (P1) is placed inside a superior model (D2) and a model assignment connection is created between the pool P1 object and the BPD diagram D1.
Documentation (0-1) : String	**Description / Definition** AT_DESC (9)	

1.4 Process Attributes

Inside ARIS the BPMN Construct **Process** is mapped to the concrete ARIS object **Pool**

An ARIS pool object is therefore a combination of the BPMN Objects **Process**, **Pool** and **Participant**

This construct is valid because a pool can have exactly one process and each process MUST HAVE a pool.

The BPMN process attributes which are mapped 1:1 to ARIS attributes are maintained at the pool object which contains the process.

Table 1.4-1: Process Attributes

Attributes	**ARIS Attribute Name (en) API_NAME (Attr. Number)**	**Other Semantic equivalent in ARIS**
Id : Object		ARIS Global Unique Identifier (GUID) for the Pool which contains the process.
Name : String	**Process Name** AT_BPMN_PROC_NAME (3424)	
ProcessType (None | Private | Abstract | Collaboration) None : String	**Process Type** AT_BPMN_POOL_TYPE (2220)	
Status (None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed) None : String		This Attribute is not relevant at the design time, it is only relevant while the process is executed by an engine and is therefore not maintainable in ARIS

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
GraphicalElements (0-n) : Object		The graphical Elements are placed inside the BPD diagram which contains this process (P1). Because a BPD can display more than one process the following mapping is essential: **1) If the process is part of a pool that has visible borders:** Each element which is directly or indirectly connected to the pool (that contains the process) via **belongs to** connections (embedded or explicit, directly or via lanes of the pool) is part of the process **2) If the process is part of a Pool which has invisible borders:** All objects which are placed inside the BPD and are not part of other processes (as stated under 1) are part of this process.
Assignments (0-n) : Assignment		Assignments are modeled inside the BPMN Allocation diagram which is assigned to the defining Flow Object. A detailed explanation of Assignment modeling in ARIS can be found in the Section Property Assignment Modeling in ARIS (Page 76).

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Properties (0-n) : Property		Process properties are modeled as properties which are connected towards the pool object that contains the process with a **has** connection. This connection has to be modeled inside an assigned BPMN allocation diagram. Further information about property modeling can be found in chapter Property Modeling in ARIS (Page 75)
AdHoc False : Boolean	**Ad Hoc** AT_BPMN_ADHOC (2148)	
[AdHoc = True only] **AdHocOrdering** (0-1) (Sequential | Parallel) Parallel : String	**Ad Hoc Ordering** AT_BPMN_AD_HOC_ORDER (3423)	
[AdHoc = True only] **AdHocCompletionCondition** (0-1) : Expression	**Ad Hoc Completion Condition** AT_BPMN_COMPLETION_CONDI (2149)	
SuppressJoinFailure False : Boolean	**Suppress Join Failure** AT_BPEL_SUPPRESS_JOIN_FAILURE (2384)	
EnableInstanceCompensation False : Boolean	**Enable Instance Compensation** AT_BPMN_ENABLE_INST_COMPENS (3419)	

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Categories (0-n) : String		Categories are represented as ARIS BPMN Category Objects. The category object is connected with all objects that are a member of a category. Those **belongs to** connections can be modeled inside one or many BPMN Allocation diagrams (which should be assigned to the object for which the categories are specified). If a modeller wants to use multiple diagrams to model members of the same category he SHALL USE an occurrence copy of the category object.
Documentation (0-1) : String	**Description / Definition** AT_DESC (9)	

1.5 Common Graphical Object Attributes

The following table displays a set of common attributes for BPMN graphical objects (Flow Objects, Swim lanes, Artefacts and

Connecting Objects).

Table 1.5-1: Common Graphical Object Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Id : Object		ARIS GUID at the representing ARIS objects
Categories (0-n) : String		Categories are represented as ARIS BPMN Category Objects. The category object is connected with all objects that are a member of a category. Those **belongs to** connections can be modeled inside one or many BPMN Allocation diagrams (which should be assigned to the object for which the categories are specified). If a modeller wants to use multiple diagrams to model members of the same category he SHALL USE an occurrence copy of the category object. (This convention enables a user to quickly find all elements which are part of a category – all objects that are connected to a category object are part of a category)
Documentation (0-1) : String	**Description / Definition** AT_DESC (9)	

1.6 Common Flow Object Attributes

The following table displays the set of attributes common to BPMN Flow Objects (Events, Activities, and Gateways), and which extends the set of common graphical object attributes (see Table 1.5-1).

Table 1.6-1: Common Flow Object Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Name** AT_NAME (1)	
Assigments (0-n) : Assignment		Assignments are modeled inside the BPMN Allocation diagram which is assigned to the defining Flow Object. A detailed explanation of assignment modeling in ARIS can be found in the Section Property Assignment Modeling in ARIS (Page 76).
Pool : Pool		**If the objects has a belongs to connection (usually this connection would be realised via an embedding (invisible connection) towards a pool or a lane which is part of a pool:** The target of the belongs to connection from the current flow object towards a pool or lane specifies the pool of this flow object. If the object has no explicit belongs to connection (only possible reason would be that the object is placed inside an invisible pool): The pool of an object is defined as the Pool object which is the assignment parent of the current BPD. So to resolve the pool that contains this flow object a flow engine would need to find the Pool object which has an assignment to the current BPD.

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Lanes (0-n) : Lane		**If the objects has a belongs to connection towards a lane:** The target(s) of the belongs to connection(s) from the current flow object towards the lane(s) specifies the lane of this flow object. Objects which are part of an invisible Pool and which have no belongs to connection towards a Lane are part of the Lane which has the same name than the Pool (implicit lane assignment towards the implicit lane of a pool.)

1.7 Events

1.7.1 Common Event Attributes

The following table displays the set of attributes common to the three types of Events, and which extends the set of common

Flow Object attributes (see Table 1.6-1).

Table 1.7-1: Common Event Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
EventType (Start | End | Intermediate) Start : String		There are three ARIS symbols which represent unspecified Start/Intermediate/End Events: **Start Event**: Type Number 841 API Name: ST_BPMN_START_EV **Intermediate Event**: Type Number 842 API Name: ST_BPMN_INTERMEDIATE_EV **End Event:** Type Number 843 API Name: ST_BPMN_END_EV There are other (trigger specific ARIS symbols, e.g. **Message Start Event**) See Trigger Specific event symbol numbers (Page 24)

1.7.2 Start Event

The following table displays the set of attributes of a Start Event, which extends the set of common Event elements (see Table 1.7-1).

The corresponding ARIS attributes are maintained at the corresponding ARIS event objects (ST_BPMN_START_EV)

Table 1.7-2:Start Event Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
Trigger (None | Message | Timer | Rule | Link | Multiple) None : String	**Trigger / Result** AT_BPMN_EV_TYPE (2137)	There are two possible ways to describe the trigger of an event – the first one (which defines the trigger effectively at the object definition level) is using the ARIS Attribute AT_BPMN_EV_TYPE. The ARIS value **not maintained** has the BPMN Semantic **None** The second approach uses separate ARIS symbol (at occurrence level) to distinguish the different event types. Both approaches have their advantages; however the modeling is more convenient with the symbol-number based approach. See Trigger Specific event symbol numbers (Page 24)

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Message Trigger only] **Message** : Message	**Message** AT_BPMN_MESSAGE (2138) DEPRECATED: **The usage of this attribute is not recommended because it doesn't allow you to reference a complete ARIS Message Element. You should use the mechanism described in the next column instead.**	Message events must have an incoming ARIS connection **receives** or an outgoing ARIS connection **sends**. Those connections may lead towards an ARIS Message object (ST_BPMN_MESSAGE). The message of a Message Trigger is therefore defined by the message object which is placed at the other side of the **sends** or **receives** connection. If the message is not specified with the above mechanism an empty (no properties defined) message will be assumed. The Name of this empty unspecified message is defined by the Attribute **Connection role** (AT_CXN_ROLE) which can be maintained directly at the ARIS Message flow connection
[Message Trigger only] **Implementation** (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	
[Timer Trigger only] **TimeDate** (0-1) : Date	**Time date** AT_BPMN_TIMEDATE (2140)	
[Timer Trigger only] **TimeCycle** (0-1) : String	**Time cycle** AT_BPMN_TIMECYCLE (2141)	

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Rule Trigger only] **RuleName** : Rule	The Abstract BPMN Type **Rule** is defined by a rule name and a rule expression: **Rule name**: AT_BPMN_RULE_ATG (3397) **Rule expression:** AT_BPMN_RULE_EXPRESSION (2142)	
[Link Trigger only] **LinkId** : String	**Link ID** AT_BPMN_LINK_NAME (2143)	The LinkID is being used to create a virtual connection between two separate link events. This connection can be created in ARIS in a more convenient way with two occurrences of the same object definition. Whenever the token reaches one of the occurrences it is automatically transferred to the other occurrence. (a modeler SHALL place exactly two occurrences of the same link event definition)
[Link Trigger only] **ProcessRef** : Process	**Process reference** AT_BPMN_PROC_REF (2155) The process has to be specified with the GUID of the pool that contains the process! DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence Copy approach described in the next column	If you model a link event with two occurrences of the same object definition the process of the second link event can be discovered automatically because the second occurrence has a **belongs to** connection towards its embedding pool and from there on towards the embedding process.

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Multiple Trigger only] **Triggers** (2-n) : Trigger		Multiple Triggers are modelled as a combination of other triggers. A possible Scenario would be: two incoming message connection for two possible start messages an incoming message and a maintained Time date attribute (process is started if the message arrives or at the given timestamp) a link event and a possible starting message any combination of the above

1.7.3 End Event

The following table displays the set of attributes of an End Event, which extends the set of common Event elements (see Table 1.7-1).

The corresponding ARIS attributes are maintained at the corresponding ARIS event objects (ST_BPMN_END_EV)

Table 1.7-3: End Event Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
Result (None | Message | Error | Cancel | Compensation | Link | Terminate | Multiple) None : String	**Trigger / Result** AT_BPMN_EV_TYPE (2137)	There are two possible ways to describe the trigger of an event – the first one (which defines the trigger effectively at the object definition level) is using the ARIS Attribute AT_BPMN_EV_TYPE. The ARIS value **not maintained** has the BPMN Semantic **None** The second approach uses separate ARIS symbol (at occurrence level) to distinguish the different event types. Both approaches have their advantages; however the modeling is more convenient with the symbol-number based approach. See Trigger Specific event symbol numbers (Page 24)
[Message Result only] **Message** : Message	Same behavior as the Message attribute at the Start Event	Same behavior as the message attribute at the Start Event
[Message Trigger only] **Implementation** (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Error Result only] **ErrorCode** : String	**Error code** AT_BPMN_ERROR_CODE (2145)	
[Compensation Result only] **Activity** : Object	**Compensation** AT_BPMN_COMPENSATION (2146) If maintained the Attribute value MUST be the ARIS GUID of the activity which needs to be compensated. DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence Copy approach described in the next column	This BPMN attribute specifies the activity that needs to be compensated. Because Compensation Results are always paired with a Compensation Trigger the recommended approach is to use 2 occurrences of the same ARIS Event definition. The first occurrence is an intermediate or an end event which can be reached via a normal sequence flow (incoming **occurs before** or **activates** connection – this event is the trigger that activates a compensation flow. The second occurrence is an intermediate event which is attached to the boundary of an activity with a **can trigger** connection. Because the two occurrence of the event have an internal ARIS connection the connection from the triggering event towards the activity that needs to be compensated is created (without the need to use AT_BPMN_COMPENSATION)
[Link Result only] **LinkId** : String	**Link ID** AT_BPMN_LINK_NAME (2143)	The LinkID is being used to create a virtual connection between two separate link events. This connection can be created in ARIS in a more convenient way with two occurrences of the same object definition. Whenever the token reaches one of the occurrences it is automatically transferred to the other occurrence. (a modeler SHALL only place two occurrences of the same link event definition)

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Link Result only] **ProcessRef** : Process	**Process reference** AT_BPMN_PROC_REF (2155) The process has to be specified with the GUID of the pool that contains the process! DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence Copy approach described in the next column	Same behavior as the ProcessRef Attribute at the Start Event
[Multiple Result only] **Results** (2-n) : Result		Multiple Results are modeled as a combination of other Results. A possible scenario would be: two outgoing message connection to represent that two messages are send a combination of a second occurrence (link event) and an outgoing message to represent a combination of a link result and a message result ...

1.7.4 Intermediate Event

The following table displays the set of attributes of an Intermediate Event, which extends the set of common Event elements (see Table Table 1.7-1: Common Event Attributes).

The corresponding ARIS attributes are maintained at the corresponding ARIS event objects

Table 1.7-4: Intermediate Event Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
Trigger (None | Message | Timer | Error | Cancel | Link | Compensation | Rule | Multiple) Message : String	**Trigger / Result** AT_BPMN_EV_TYPE (2137)	There are two possible ways to describe the trigger of an event – the first one (which defines the trigger effectively at the object definition level) is using the ARIS Attribute AT_BPMN_EV_TYPE. The ARIS value **not maintained** has the BPMN Semantic **None** The second approach uses separate ARIS symbol (at occurrence level) to distinguish the different event types. Both approaches have advantages; however the modeling is more convenient with the symbol-number based approach. See Trigger Specific event symbol numbers (Page 24)
Target (0-1) : Object		This BPMN Attribute defines the Activity or Sub Process which may throw the Compensation / Timeout or Error Event that is represented by this Intermediate Event. This activity is specified inside ARIS with a **can trigger** connection from the activity towards the intermediate event

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Message Trigger only] **Message** : Message		Same behavior as the Message attribute at the Start Event (The direction of the ARIS Connection defines if the message is being sent or received.
[Message Trigger only] **Implementation** (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	
[Timer Trigger only] **Timedate** (0-1) : Date	**Time date** AT_BPMN_TIMEDATE (2140)	
[Timer Trigger only] **TimeCycle** (0-1) : String	**Time cycle** AT_BPMN_TIMECYCLE (2141)	
[Error Trigger only] **ErrorCode** : String	**Error code** AT_BPMN_ERROR_CODE	The error code is being used to link an error **throwing** end event with an intermediate event that **catches** the error. This connection can also be established with two occurrences of the same ARIS event (The first occurrence would be the triggering end event; the second occurrence represents the catching intermediate event). If multiple error IDs should be catched by the same intermediate event the error code field of the catching intermediate event SHALL be empty.

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Compensation Trigger only] **Activity** : Object	**Compensation** AT_BPMN_COMPENSATION (2146) DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence Copy approach described in the next column	The connection between an intermediate event that triggers the compensation and the intermediate event that is the starting point of the compensation (attached to the activity boundary) should be created via ARIS occurrence copies. The activity that needs to be compensated is therefore specified by a **can trigger** connection from the activity towards the second (attached) intermediated event.
[Rule Trigger only] **RuleName** : Rule	The Abstract BPMN Type **Rule** is defined by a rule name and a rule expression: **Rule name**: AT_BPMN_RULE_ATG (3397) **Rule expression:** AT_BPMN_RULE_EXPRESSION (2142)	
[Link Trigger only] **LinkId** : String	**Link ID** AT_BPMN_LINK_NAME (2143) DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence copy approach described in the next column	The LinkID is being used to create a virtual connection between two separate link events. This connection can be created in ARIS in a more convenient way with two occurrences of the same object definition. Whenever the token reach's one of the occurrences it is automatically transferred to the other occurrence. (This only works if you place exactly two occurrences of the link event!)

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
[Link Trigger only] **ProcessRef** : Process	**Process reference** AT_BPMN_PROC_REF DEPRECATED: The usage of the attribute is not recommended because the same semantic can be archived with the ARIS Occurrence copy approach described in the next column	If you model a link event with two occurrences of the same object definition the process of the second link event can be discovered automatically because the second occurrence has a connection towards its embedding pool and from there on towards the embedding process.
[Multiple Trigger only] **Triggers** (2-n) : Trigger		Multiple Triggers are modeled as a combination of other triggers. A possible Scenario for an intermediate event would be: two incoming message connection for two possible trigger messages an incoming message and a maintained Time date attribute (event is triggered if the message arrives OR at the given timestamp) a link event and a possible starting message

1.7.5 Trigger Specific event symbol numbers (Updated with the last Service Release)

The above specified mechanism for the event trigger selection via the ARIS attribute AT_BPMN_EV_TYPE (Message, Link, Multiple etc…) has been extended with a new symbol number specific approach. Additionally to the above mentioned general event symbol numbers the following new (trigger specific) event symbols were added to the latest ARIS version:

BPMN event name	ARIS Symbol Number (API-Name)
Message Start Event	ST_BPMN_MSG_SE
Message Intermediate Event	ST_BPMN_MSG_IE
Message End Event	ST_BPMN_MSG_EE
Link Start Event	ST_BPMN_LINK_SE
Link Intermediate Event	ST_BPMN_LINK_IE
Link End Event	ST_BPMN_LINK_EE
Multiple Start Event	ST_BPMN_MULTIPLE_SE
Multiple Intermediate Event	ST_BPMN_MULTIPLE_IE
Multiple End Event	ST_BPMN_MULTIPLE_EE
Error Intermediate Event	ST_BPMN_ERROR_IE
Error End Event	ST_BPMN_ERROR_EE
Cancel Intermediate Event	ST_BPMN_CANCEL_IE
Cancel End Event	ST_BPMN_CANCEL_EE
Compensation Intermediate Event	ST_BPMN_COMPENSATION_IE
Compensation End Event	ST_BPMN_COMPENSATION_EE
Terminate End Event	ST_BPMN_TERMINATE_EE

A modeling project should stick to one of the two trigger definition approaches (attribute based or symbol number based).

The symbol number based approach is more convinient for the modeler because he doesn't

have to maintain the trigger type attribute for each placed event. The attribute based approach has the advantage that ARIS ensures the same event type for every occurrence of the same object definition.

The provided BPMN semantic checks can handle both trigger definition approaches.

If the symbol number specific approach is being used the attribute AT_BPMN_EV_TYPE should either be set to the correct event type or should be left unmaintained (because the export logic can use the symbol number to determine the effective event type).

However, if the symbol number specific approach is being used the ARIS Attribute **AT_BPMN_EV_TYPE** MUST NOT be set to a conflicting event type.

1.8 Activities

1.8.1 Common Activity Attributes

The following table displays the set of attributes common to both a Sub-Process and a Task, and which extends the set of common Flow Object attributes (see Table 1.6-1).

Table 1.8-1: Common Activity Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
ActivityType (Task | Sub- Process) Task : String		The Activity Type is defined via the ARIS Symbol Type Number that is being used for the Activity. **Task**: Type Number 335 API: ST_FUNC **Sub-Process**: **(Collapsed)** Type Number 1213 API: ST_BPMN_SUB_PROC_COLLAPSED **Sub Process**: **(expanded)** Type Number 1214 API: ST_BPMN_SUB_PROC_EXPANDED
Status (None | Ready | Active | Cancelled | Aborting | Aborted | Completing | Completed) None : String		The Status of the Activity is only relevant during the process execution and is therefore not maintained in ARIS

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
Properties (0-n) : Property		The properties of activities are modeled the same way as for processes and messages. Further information can be found in chapter Property Modeling in ARIS (Page 75)
InputSets (0-n) : Input		The modeling of InputSets, OutputSets and IORules is described in the Chapter Input Sets, Output Sets and IORules in ARIS (Page 78)
[Input: for InputSets only] **Inputs** (1-n) : Artifact		The modeling of InputSets, OutputSets and IORules is described in the Chapter Input Sets, Output Sets and IORules in ARIS (Page 78)
		The modeling of InputSets, OutputSets and IORules is described in the Chapter Input Sets, Output Sets and IORules in ARIS (Page 78)
[Output: for OutputSets only] **Outputs** (1-n) : Artifact		The modeling of InputSets, OutputSets and IORules is described in the Chapter Input Sets, Output Sets and IORules in ARIS (Page 78)
IORules (0-n) : Expression		The modeling of InputSets, OutputSets and IORules is described in the Chapter Input Sets, Output Sets and IORules in ARIS (Page 78)
StartQuantity 1 : Integer	**Start quantity** AT_BPMN_START_QUANT (3411)	

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
LoopType (None | Standard | MultiInstance) None : String	**Loop type** AT_BPMN_LOOP_TYPE (2163)	The loop type attribute has visual representations (display the attribute as an icon)

1.8.1.1 Standard Loop Attributes

The following are additional attributes of a Standard Loop Activity (where the LoopType attribute is set to **Standard**), which extends the set of common activity attributes (see Table 1.8-1).

Table 1.8-2: Standard Loop Activity Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
LoopCondition : Expression	**Loop condition** AT_BPMN_LOOP_CONDITION (2164)	
LoopCounter : Integer		The LoopCounter attribute is used at runtime to count the number of loops and is therefore not maintainable in ARIS
LoopMaximum (0-1) : Integer	**Loop maximum** AT_BPMN_MAX_LOOP (2165)	
TestTime (Before | After) After : String	**Test before [Boolean]** AT_BPMN_LOOP_TEST_TIME (2166) The ARIS boolean value TRUE represents the BPMN value Before	

1.8.1.2 Multi-Instance Loop Attributes

The following are additional attributes of a Multi-Instance Loop Activity (where the LoopType attribute is set to **MultiInstance**), which extends the set of common activity attributes (see Table 1.8-1).

The attributes are maintained at the ARIS Function / Sub Process / Transaction Objects.

Table 1.8-3: Multi-Instance Loop Activity Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
MI_Condition : Expression	**MI_Condition** AT_BPMN_MI_CONDITION (3412)	
LoopCounter : Integer		The LoopCounter attribute is used at runtime to count the number of activities that are already executed and is therefore not maintainable in ARIS
MI_Ordering (Sequential | Parallel) Sequential : String	**Parallel MI_Ordering** AT_BPMN_LOOP_INST_GEN (2167) The ARIS Boolean value TRUE represents the BPMN Value Parallel	
[Parallel MI_Ordering only] **MI_FlowCondition** (None | One | All | Complex) All : String	**MI_FlowCondition** AT_BPMN_LOOP_FLOW_CONDI (2168)	
[Complex MI_FlowCondition only] **ComplexMI_FlowCondition** (0-1) : Expression	**ComplexMI_FlowCondition** AT_BPMN_LOOP_FLOW_CNDI_COMPLEX (2169)	

1.8.2 Sub-Process

Sub processes are modeled inside the BPD with the two ARIS Function symbols **Sub Process (collapsed)** and **Sub Process (expanded)** – the BPMN Attributes of a sub process are maintainable at the corresponding ARIS objects.

The following table displays the set of attributes of a Sub-Process, which extends the set of common activity attributes (see Table 1.8-1).

Table 1.8-4: Sub-Process Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
SubProcessType (Embedded | Independent | Reference) Embedded : String	**Sub process** AT_BPMN_SUB_PROC_TYPE (2154)	The ARIS Attribute does not provide a value for the Reference Sub Process Type. Referenced Sub Processes are defined with the ARIS multiple occurrence Mechanism. An second ARIS occurrence of an independent Sub Process has the same semantic than a BPMN Referenced Sub Process
IsATransaction False : Boolean	**Transaction** AT_BPMN_TRANSACTION (2159)	Additionally ARIS provides a new activity symbols Transaction (expanded) and Transaction (collapsed). If the modeler is using those symbols he SHALL set the Transaction attribute to true

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
Transaction (0-1) : Transaction	The Abstract BPMN Structure Transaction is defined by the following 3 ARIS attributes: **Transaction ID:** AT_BPMN_TRANSACTION_ID (2160) **Transaction protocol:** AT_BPMN_TRANSACTION_PROTOCOL (2161) **Transaction method:** AT_BPMN_TRANSACTION_METHOD (2162)	

1.8.2.1 Embedded Sub-Process

The following are additional attributes of an Embedded Sub-Process (where the Sub process type attribute is set to **Embedded**), which extends the set of Sub-Process attributes (see Table 1.8-4).

Table 1.8-5: Embedded Sub-Process Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
GraphicalElements (0-n) : Object		**For collapsed embedded Sub Processes:** The elements which are placed inside a BPMN Model that is assigned to a BPMN Sub Process Symbol are part of the sub-process. For expanded embedded Sub Processes: All elements which have an ARIS belong to connection towards the sub-process (usually via an invisible embedding connection) are part of the sub-process.
AdHoc False : Boolean	**Ad hoc** AT_BPMN_ADHOC (2148)	
[AdHoc = True only] **AdHocOrdering** (0-1) (Sequential | Parallel) Parallel : String	**Ad hoc ordering** AT_BPMN_AD_HOC_ORDER (3423)	

1.8.2.2 Independent Sub-Process Attributes

The following are additional attributes of an Independent Sub-Process (where the SubProcessType attribute is set to **Independent**), which extends the set of Sub-Process attributes (see Table 1.8-4).

Table 1.8-6: Independent Sub-Process Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
DiagramRef : Business Process Diagram		The diagram is defined via a ARIS Assignment Connection from the Sub Process Symbol towards the BPD Model that contains the process that should be executed
ProcessRef : Process	**Process Reference** AT_BPMN_PROC_REF (2155) Because the BPD Diagram specified with the DiagramRef Attribute can specify more than one process the exact process inside the BPD has to be defined. This Attribute must be maintained with the ARIS GUID of the Pool Object that contains Process that should be executed.	(Depending on your BPEL Export logic it may be possible to leave this attribute unmaintained if the target diagram(=assigned model) contains ONLY ONE process – in this case the target process can be discovered automatically via the assigned model)

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
InputPropertyMaps (0-n) : Expression		The InputPropertyMaps Attribute specifies which properties of the parent process are copied into which properties of the sub process (OrderProcess.CustomerName -> CreditCardPaymentProcess.Debitor). The modeling of this mapping is described in the chapter Input / Output Property Maps modeling in ARIS (Page 75)
OutputPropertyMaps (0-n) : Expression		The OutputPropertyMaps Attribute specifies which Properties of the independent sub process are copied into which properties of the parent process (after the sub process has finished its execution) CreditCardPaymentProcess.Debitor -> OrderProcess.CustomerName. The modeling of this mapping is described in the chapter Input / Output Property Maps modeling in ARIS (Page 75)

1.8.2.3 Reference Sub-Process Attributes

The following table displays the set of attributes of a Reference Sub-Process (where the SubProcessType attribute is set to **Reference**), which extends the set of Sub-Process attributes (see Table 1.8-4).

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
SubProcessRef : Task		Referenced Sub Processes are modeled inside ARIS with a second occurrence copy of an independent sub process object.

1.8.3 Task

The following table displays the set of attributes of a Task, which extends the set of common activity object attributes (see

Table 1.8-1).

Table 1.8-7: Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
TaskType (Service | Receive | Send | User | Script | Abstract | Manual | Reference | None) None : String	**Task Type** AT_BPMN_TASK_TYPE (2170)	The ARIS Attribute does not provide a value for the **Reference** Task Type. Referenced Task Types are defined with the ARIS multiple occurrence mechanism. An second ARIS occurrence of a Task has the same semantic as a BPMN Referenced Task

1.8.3.1 Service Task Attributes

The following table displays the set of attributes of a Service Task (where the Task Type attribute is set to **Service**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-8: Service Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
InMessage : Message		In Message specifies the message that is send towards a Web Service at the beginning of the task. This semantic is modeled with an outgoing **sends** connection from the task towards a message (and from there on towards the web service / task that is the receiver of this message).
OutMessage : Message		Out Message specifies the message that has to be received from a Web Service at the end of the task. This semantic is modeled with an incoming **is received by** connection from the message (and from there on towards the sender) towards the task.
Implementation (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	

1.8.3.2 Receive Task Attributes

The following table displays the set of attributes of a Receive Task (where the Task Type attribute is set to **Receive**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-9: Receive Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Message : Message		This Semantic is modeled with an incoming message flow **is received from** connection from a message object towards the Receive Task
Instantiate False **: Boolean**	**Instantiate** AT_BPMN_TASK_INSTANTIATE (2171)	
Implementation (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	

1.8.3.3 Send Task Attributes

The following table displays the set of attributes of a Send Task (where the Task Type attribute is set to **Send**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-10: Send Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Message : Message		This Semantic is modeled with an outgoing message flow **sends** connection from the sending task towards a message object (and from there on towards the receiver)
Implementation (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	

1.8.3.4 User Task Attributes

The following table displays the set of attributes of a User Task (where the Task Type attribute is set to **User**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-11: User Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Performers (1-n) : String		Performers are modeled inside an assigned BPMN allocation Diagram. The performers of an user task are specified with one or multiple **executes** connections towards the human resource (or another ARIS object that represents a person or a job-role) that performs the task.
InMessage : Message		In Message specifies the message that is send towards a Web Service at the beginning of the task. This semantic is modeled with an outgoing **sends** connection from the task towards a message (and from there on towards the pool / task that is the receiver of this message).
OutMessage : Message		Out Message specifies the message that is received from a Web Service at the end of the task. This semantic is modeled with an incoming **is received by** connection from the message towards a message (and from there on towards the sender of the message).

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Implementation (Web Service | Other | Unspecified) Web Service : String	**Implementation** AT_BPMN_IMPLEMENTATION (3398)	

1.8.3.5 Script Task Attributes

The following table displays the set of attributes of a Script Task (where the TaskType attribute is set to **Script**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-12: Script Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Script (0-1) : String	**Script** AT_BPMN_SCRIPT (3422)	

1.8.3.6 Manual Task Attributes

The following table displays the set of attributes of a Manual Task (where the Task Type attribute is set to **Manual**), which extends the set of Task attributes (see Table 1.8-7):

Table 1.8-13:Manual Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Performers (0-n) : String		Performers are modeled inside an assigned BPMN allocation Diagram. The performers of manual task are specified with one or multiple **executes** connections towards the human resource (or another ARIS object that represents a person or a job-role) that performs the task.

1.8.3.7 Reference Task Attributes

The following table displays the set of attributes of a Reference Task (where the TaskType attribute is set to **Reference**), which extends the set of Task attributes (see Table 1.8-7).

Table 1.8-14: Reference Task Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
TaskRef : Task		The ARIS Attribute does not provide a value for the **Reference** Task Type. Referenced Task Types are defined with the ARIS multiple occurrence mechanism. **An second ARIS occurrence of a Task has the same semantic as a BPMN Referenced Task**

1.9 Gateways

1.9.1 Common Gateway Attributes

The following table displays the attributes common to Gateways, and which extends the set of common Flow Object attributes

(see Table 1.6-1).

Table 1.9-1: Common Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS Description
GatewayType (XOR | OR | Complex | AND) XOR : String	**GatewayType** AT_BPMN_GATEWAY_TYPE (2147)	

1.9.2 Exclusive Gateways (XOR)

1.9.2.1 Data-Based

The following table displays the attributes for a Data-Based Exclusive Gateway. These attributes only apply if the

GatewayType attribute is set to XOR. The following attributes extend the set of common Gateway attributes (see Table 1.9-1).

Table 1.9-2: Data-Based Exclusive Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
XORType (Data | Event) Data : String	**GatewayType** AT_BPMN_GATEWAY_TYPE (2147)	The BPMN Attributes Gateway Type and XOR Type are combined into the ARIS GatewayType Attribute (Data-based or Event-based value description)
MarkerVisible False : Boolean		This attribute maps to the fact if the attribute **GatewayType** is a placed attribute in ARIS
Gates (0-n) : Gate		Usually BPMN gates are not modeled explicitly in ARIS. The semantic of BPMN gates (expression / default flow) is usually maintained at the ARIS connections which represent the BPMN Sequence Flow.
[Gate] **OutgoingSequenceFlow** : SequenceFlow		This Sequence flow is represented by an outgoing sequence flow connection which has The ARIS Attribute **Condition** AT_BPMN_SEQ_FLOW_CONDITION (2174) set to **Expression**.

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
[Gate] **Assigments** (0-n) : Assignment		The following modeling technique enables a user to model property assignments for individual gates: - Create an ARIS assignment relationship from the Sequence Flow connection which represents the gate towards a separate BPMN Allocation Diagram - Place a **Gate** Object inside this Allocation Diagram - Create an **assigns** connection from the gate object towards the property which should be the target of the assignment. - Maintain the connection attributes **from** and **assign time** at the connection from the gate object towards the property. Further information about property assignment can be found in chapter Property Assignment Modeling in ARIS
DefaultGate (0-1) : Gate		The default gate is represented by a sequence flow which has its **Condition** attribute set to **Default**
[Gate] **OutgoingSequenceFlow** : SequenceFlow		This Sequence flow is represented by an outgoing sequence flow connection which has The ARIS Attribute **Condition** AT_BPMN_SEQ_FLOW_CONDITION (2174) set to **Default**) .

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
[Gate] **Assigments** (0-n) : Assignment		The modeling technique for gate assignments is described above (see the explanation of assignments for non-default gates that are part of a data-based exclusive gateway)

1.9.2.2 Event-Based

The following table displays the attributes for an Event-Based Exclusive Gateway. These attributes only apply if the

GatewayType attribute is set to XOR. The following attributes extend the set of common Gateway attributes (see Table 1.9-1).

Table 1.9-3: Event-Based Exclusive Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
XORType (Data | Event) Event : String		The BPMN Attributes Gateway Type and XORType are combined into the ARIS GatewayType Attribute (Data-based or Event-based value description)
Gates (2-n) : Gate		Usually BPMN gates are not modeled explicitly in ARIS. The semantic of BPMN gates (expression / default flow) is usually maintained at the ARIS connections which represent the BPMN SequenceFlow.
[Gate] **OutgoingSequenceFlow** : SequenceFlow		This Sequence flow is represented by an outgoing sequence flow connection from the gateway towards an EVENT. The sequence flow connection must have the ARIS Attribute Condition AT_BPMN_SEQ_FLOW_CONDITION (2174) set to None. **None** is represented by an unmaintained ARIS Attribute)
[Gate] **Assigments** (0-n) : Assignment		The modeling technique for gate assignments is described above (see the explanation of assignments for non-default gates that are part of a data-based exclusive gateway)

1.9.3 Inclusive Gateways (OR)

These attributes only apply if the GatewayType attribute is set to OR. The following attributes extend the set of common Gateway attributes (see Table 1.9-1).

Inclusive Gateways are modeled in ARIS with a Gateway object that has its Gateway Type attribute set to **OR**

Table 1.9-4: Inclusive Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Gates (0-n) : Gate		Same modeling as for the data based XOR gateway
[Gate] **OutgoingSequenceFlow** : SequenceFlow		Same modeling as for the data based XOR gateway
[Gate] **Assigments** (0-n) : Assignment		Same modeling as for the data based XOR gateway.
DefaultGate (0-1) : Gate		Same modeling as for the data based XOR gateway
[Gate] **OutgoingSequenceFlow** : SequenceFlow		Same modeling as for the data based XOR gateway
[Gate] **Assigments** (0-n) : Assignment		Same modeling as for the data based XOR gateway

1.9.4 Complex Gateways

The following table displays the attributes for a Complex Gateway. These attributes only apply if the GatewayType attribute is

set to Complex. The following attributes extend the set of common Gateway attributes (see Table 1.9-1).

A complex gateway is represented in ARIS as a **Gateway** Object which has its Attribute **Gateway Type** set to **Complex**

Table 1.9-5: Complex Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Gates (0-n) : Gate		BPMN gates are not modeled explicitly in ARIS. The semantic of BPMN gates (expression / default flow) is usually maintained at the ARIS connections which represent the BPMN SequenceFlow.
[Gate] **OutgoingSequenceFlow** : SequenceFlow		Same modeling as for the data based XOR gateway
[Gate] **Assigments** (0-n) : Assignment		The modeling technique for gate assignments is described above (see the explanation of assignments for non-default gates that are part of a data-based exclusive gateway)
IncomingCondition (0-1) : Expression	**Incoming Condition** AT_BPMN_INCOMING_COND (2379)	
OutgoingCondition (0-1) : Expression	**Outgoing Condition** AT_BPMN_OUTGOING_COND (2380)	

1.9.5 Parallel Gateways (AND)

The following table displays the attributes for a Parallel Gateway. These attributes only apply if the GatewayType attribute is

set to AND (Parallel). The following attributes extend the set of common Gateway attributes (see Table 1.9-1).

A parallel Gateway is represented in ARIS as an **Gateway** Object which has its Attribute **Gateway Type** set to **AND**

Table 1.9-6: Parallel Gateway Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Gates (0-n) : Gate		Same modeling as for the data based XOR gateway
[Gate] **OutgoingSequenceFlow** : SequenceFlow		Same modeling as for the data based XOR gateway
[Gate] **Assigments** (0-n) : Assignment		Same modeling as for the data based XOR gateway

1.10 Swim lanes (Pools and Lanes)

1.10.1 Common Swimlane Attributes

The following table displays a set of common attributes for Swim lanes (Pools and Lanes), and which extends the set of common graphical object attributes (see Table 1.5-1).

The ARIS attributes are maintained at the corresponding ARIS objects **Pool** and **Lane**

Table 1.10-1: Common Swimlane Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Name** AT_NAME (1)	

1.10.2 Pool

The following table displays the identified attributes of a Pool, and which extends the set of common Swimlane attributes (see

Table 1.10-1).

Table 1.10-2: Pool Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Process (0-1) : Process		Because each Pool may have a process and each process MUST have a pool the abstract BPMN construct **Process** was combined with the ARIS Object Type **Pool**. The Process is part of the Pool; the Attributes of the process are maintainable in the attribute group **Process** at the pool Object.
Participant : Participant	The attributes of the participant are maintained in the Attribute Group **Participant** at the ARIS Pool Object. **Participant Type**: AT_BPMN_PARTICIPANT_TYPE (3426) **Role / Entity:** AT_BPMN_ROLE_ENTITY (3427)	
Lanes (1-n) : Lane		Lanes of a pool are defined as following: All Lane Objects have a direct (implicit or explicit) ARIS **belongs to** connections from the Lane towards this pool are part of this pool.

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
BoundaryVisible True : Boolean		Only Visible Pools are placed inside a BPMN Model. **Invisible pools are modeled as following:** A visible pool object P1 is placed inside a separate model M1 (M1 may be a BPD or an other model type) and the content of the pool is placed inside a separate BPD M2. An Assignment connection from the P1 object towards the BPD M2 specifies that all objects which are placed inside M2 and which are NOT part of another pool are part of P1

1.10.3 Lane

The following table displays the identified attributes of a Lane, and which extends the set of common Swimlane attributes (see

Table 1.10-1).

Table 1.10-3: Lane Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
ParentPool : Pool		The parent pool is defined as the pool which can be reached via **belongs to** connections (either directly or via intermediate parent lane objects.
ParentLane (0-1) : Lane		The parent lane of lane is defined as the lane which can be reached directly via a **belongs to** connection – only one of those connections may be drawn

1.11 Artifacts

1.11.1 Common Artefact Attributes

The following table displays the identified attributes common to Artifacts, and which extends the set of common graphical object attributes (see Table 1.5-1).

Table 1.11-1: Common Artefact Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
ArtifactType (DataObject | Group | Annotation) DataObject : String		The artefact type is specified by the ARIS symbol number of the placed artefact. **DataObject:** SymboltypNum: ST_BPMN_ARTIFACT (955) **Group** SymbolNum: ST_BPMN_GROUPING (1216) **Annotation** SymbolNum ST_BPMN_ANNOTATION (1219)

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Pool (0-1) : Pool		**If the artefact has a belongs to connection towards a pool or a lane which is part of a pool:** The target of the belongs to connection from the current flow object towards a pool or lane specifies the pool of this artefact. If the artefact has no explicit belongs to connection (only possible reason would be that the artefact is placed inside an invisible pool): The pool of an object is defined as the Pool object which is the assignment parent of the current BPD. So to resolve the pool that contains this flow object a flow engine would need to find the Pool object which has an assignment to the current BPD.
Lanes (0-n) : Lane		The lanes of artefacts are defined as the lanes which can be reached directly via a **belongs to** connection

1.11.2 Data Object

The following table displays the attributes for Data Objects, and which extends the set of common Artifact attributes (see Table 1.11-1).

Data Objects are represented in ARIS with the dedicated **DataObject** Symbol which is Information carrier.

Table 1.11-2: Data Object Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Name** AT_NAME (1)	
State (0-1) : String	**State** AT_BPMN_STATE (3428)	
Properties (0-n) : Properties		The properties of DataObjects are modeled the same way as the properties of activities – More information can be found in the chapter Property Modeling in ARIS (Page 75)
RequiredForStart True : Boolean	**Required for Start** (at the outgoing connection towards the consumer of the data object. AT_BPMN_REQU_4_START (3420)	
ProducedAtCompletion True : Boolean	**Produced at Completion** (at the incoming connection from the producer of the artefact) AT_BPMN_PRODUCED_AT_COMPLET (3421)	

1.11.3 Text Annotation

Text Annotations are modeled in ARIS with the dedicated **Annotation** Object which has the symbol number 1219 (ST_BPMN_ANNOTATION) The BPMN Attributes are maintained at those Object

The following table displays the attributes for Annotations, and which extends the set of common Artifact attributes (see Table 1.11-1).

Table 1.11-3: Text Annotation Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Text : String	AT_DESC (9) or AT_NAME(1)	

1.11.4 Group

Groups are modeled in ARIS with the dedicated **Group** Object (Symbol number 1216 - ST_BPMN_GROUPING)

The following table displays the attributes for Groups, and which extends the set of common Artifact attributes (see Table 1.11-1).

Table 1.11-4: Group Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name (0-1) : String	AT_NAME (1)	

1.12 Graphical Connecting Objects

Graphical Connecting Objects are represented as various types ARIS Connection types.

Table 1.12-1: Common Connecting Object Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Connection role** AT_CXN_ROLE (25)	
Source : Object		The source and target of ARIS connection are maintained internally (they are semantically defined by the endpoints of a connection) and are usually not maintained by the modeler in form of attributes – they are automatically maintained if the modeler creates / changes a connection endpoint.
Target : Object		The source and target of an ARIS connection are maintained internally (they are semantically defined by the endpoints of a connection) and are usually not maintained by the modeler in form of attributes – they are automatically maintained if the modeler creates / changes a connection endpoint.

1.12.1 Sequence Flow

The following table displays the set of attributes of a Sequence Flow, and which extends the set of common Connecting Object attributes (see Table 1.12-1).

Sequence Flows are represented in ARIS with the following ARIS connection types:

Connection Name (English)	API Name	Connection API Number	Connected Objects
occurs before	CT_SUCCEED	477	Event -> Event
activates	CT_ACTIV_1	43	Event -> Function
is evaluated by	CT_IS_EVAL_BY_1	48	Event -> Gateway
creates	CT_CRT_1	44	Function -> Event
is predecessor of	CT_IS_PREDEC_OF_1	118	Function -> Function
leads to	CT_LEADS_TO_1	116	Function -> Gateway
leads to	CT_LEADS_TO_2	117	Gateway -> Event
activates	CT_ACTIV_1	43	Gateway -> Function
links	CT_LNK_2	54	Gateway -> Gateway

The sequence flow attributes are maintained directly at the corresponding ARIS connections.

Table 1.12-2: Sequence Flow Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
ConditionType (None | Expression | Default) None : String	**Condition** AT_BPMN_SEQ_FLOW_CONDITION (2174) The condition type None is represented by a non maintained attribute value.	
[ConditionType is set to Expression only] **ConditionExpression** : Expression	**Condition Expression** AT_BPMN_CONDITION_EXPRESSION (2175)	
Quantity 1 : Integer	**Quantity** AT_BPMN_QUANT (3429)	

1.12.2 Message Flow

ows are represented in ARIS with the following ARIS connection types:

message flow	CT_BPMN_MESSAGE_FLOW (689)	Sender -> Receiver
sends	CT_SENDS_2 (407)	Sender -> Message Object
is received by	CT_IS_RECEIVED (408)	Message Object → Receiver

The Message Object is of type ST_BPMN_MESSAGE(866)

g table displays the identified attributes of a Message Flow, and which extends the set of common bject attributes (see Table 1.12-1).

Table 1.12-3: Message Flow Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Message (0-1) : Message		A message flow that specifies a dedicated message is modeled as following: Source → Message Object → Target The message object (AT_CLST (14)) is placed **between** source and target object. (See Figure 1: Explicit modeled message object

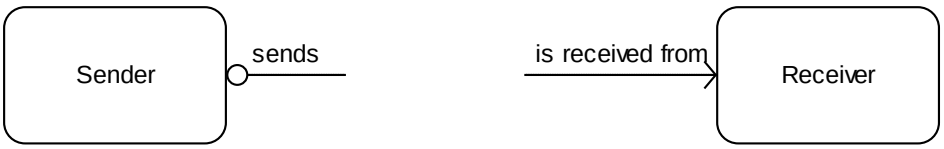


Figure 1: Explicit modeled message object

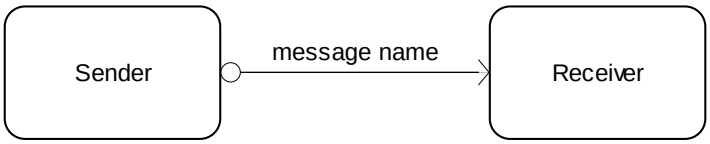


Figure 2: Implicit modeled message object (no message structure defined)

1.12.3 Association

The following table displays the identified attributes of an Association, and which extends the set of common Connecting Object attributes (see Table 1.12-1).

Associations are modeled with multiple ARIS connection types. (The concrete connection number depends on the semantic of the association). All ARIS Connections which are not classified as sequence flows or as message flows can be classified as BPMN Association.

Table 1.12-4: Association Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Direction (None | To | From | Both) None : String		The direction of an ARIS Connection is maintained internally and is usually not maintained via attributes. The user can define the direction of an association (if a connection can be created in both directions) during the placement of the ARIS connection.

1.13 Supporting Types

1.13.1 Assignment

BPMN Assignments are represented in ARIS by an **assigns** connection from the assigning flow object towards the property which represents the target of the assignment. These connections are modeled inside a BPMN Allocation Diagram.

The source of this **assigns** connection has to be the flow object which performs the assignment. Further information can be found in the Chapter Property Assignment Modeling in ARIS (Page 76).

The following table displays the set of attributes of an Assignment, which is used in the definition of attributes for Process, Activities, Events, Gateways, and Gates.

Table 1.13-1: Assignment Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
To : Property		The target property is defined by the property which is the target of the ARIS **assigns** connection which represents the assignment
From : Expression	**From** (at the **assigns** connection) AT_BPMN_ASSIGN_EXPRESSION (2151)	
Assign Time (0-1) (Start | End) Start : String	**Assign Time** (at the **assigns** connection) AT:BPMN_ASSIGN_TIME (2150)	

1.13.2 Entity

The following table displays the set of attributes of an Entity, which is used in the definition of attributes for a Participant.

Entities are part of a Participant which is a member of a Pool Object. Therefore the entity attributes are maintained at the ARIS Pool Object which represents the given participant

Table 1.13-2: Entity Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
	Role / Entity	
Name : String	AT_BPMN_ROLE_ENTITY (3427)	

1.13.3 Expression

The following table displays the set of attributes of an Expression, which is used in the definition of attributes for Start Event, Intermediate Event, Activity, Complex Gateway, and Sequence Flow.

Because the BPMN Object **Expression** can be directly represented as one String Attribute the Expression Object is not explicitly modeled in ARIS. Whenever an Expression Object is needed as an Attribute of another object the Expression String is directly entered at the object which references the Expression

Table 1.13-3: Expression Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Expression : String	Multiple Attributes – depending on the usage of the Expression:	The abstract BPMN construct Expression is defined by its only member **Expression** – this member is maintained directly at all objects which can have a Expression – so there is no need to define an explicit Expression object in ARIS

1.13.4 Message

The following table displays the set of attributes of a Message, which is used in the definition of attributes for a Start Event, End Event, Intermediate Event, Task, and Message Flow.

Messages have a 1:1 Representation in ARIS models. They are modeled as objects that have the symbol number ST_BPMN_MESSAGE (866)

Table 1.13-4: Message Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Name** AT_NAME(1)	
Properties (0-n) : Property		The properties of messages are modeled the same way as for processes and activities. Further information can be found in chapter Property Modeling in ARIS (Page 75)
From : Participant		The source of a message is defined by the source of a **sends** connection from the sending object towards a message object. In an ARIS model participants are **embedded** inside the **Pool** Objects. If the direct source of a message flow is NOT a pool object the source participant is defined as the pool object which is the embedding container of the actual source (The pool object which can be reached via a **belongs to** connection from the actual message source).

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
To : Participant		The target of a message is defined by the target of an **is received by** connection from the message object towards the receiving participant. In an ARIS model participants are included inside the **Pool** Objects. If the direct target of a message flow is NOT a pool object the target participant is defined as the pool object which is the embedding container of the actual target. (The pool object which can be reached via a **belongs to** connection from the actual message target.

1.13.5 Object

The following table displays the set of attributes of an Object, which is used in the definition of attributes for all graphical elements.

Table 1.13-5: Object Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Id : String		The ID maps directly to the ARIS GUID which is automatically maintained for each object.

1.13.6 Participant

The definition of a Pool Participant is done with ARIS attributes that can be maintained directly at the pool object.

Table 1.13-6: Participant Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
ParticipantType (Role | Entity) Role : String	**Participant Type** AT_BPMN_PARTICIPANT_TYPE (3426)	
[ParticipantType = “Role” only] **Role** (0-1) : Role	**Role / Entity** AT_BPMN_ROLE_ENTITY (3427)	
[ParticipantType = “Entity” only] **Entity** (0-1) : Entity	**Role / Entity** AT_BPMN_ROLE_ENTITY (3427)	

1.13.7 Property

Properties are modeled with dedicated Property objects in ARIS

Those property objects can be placed inside the BPMN Allocation Diagram and have a connection towards their defining object (Process or Activity)

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

Table 1.13-7: Property Attributes

Attributes	**ARIS Attribute Name (en)** **API_NAME (Attr. Number)**	**Other Semantic equivalent in ARIS**
Name : String	**Name** AT_NAME (1)	
Type : String	**Type** AT_TYPE (972)	
[Type = “Set” only] **Correlation** (0-1) False : Boolean	**Correlation** AT_BPMN_CORRELATION (3430)	

1.13.8 Role

The following table displays the set of attributes of a Role, which is used in the definition of attributes for a Participant.

Roles are part of a Participant which is a member of a Pool Object. Therefore the attributes are maintained at the ARIS Pool Object which represents the given participant

Table 1.13-8: Role Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
	Role / Entity	
Name : String	AT_BPMN_ROLE_ENTITY (3427)	

1.13.9 Rule

Rule objects are not directly modeled in ARIS – the attributes are maintained directly at the event objects which reference the Rule object (a rule start event for example).

The following table displays the set of attributes of a Rule, which is used in the definition of attributes for Start Event and

Intermediate Event.

Table 1.13-9: Rule Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Name : String	**Rule Name** AT_BPMN_RULE_ATG (3397)	
RuleExpression (0-1) : Expression	**Rule expression** AT_BPMN_RULE_EXPRESSION (2142)	

1.13.10 Transaction

Transaction objects are not modeled directly in ARIS. The attributes are maintained directly at the sub process objects which references the transaction.

The following table displays the set of attributes of a Transaction, which is used in the definition of attributes for a Sub- Process.

Table 1.13-10: Transaction Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
TransactionId : String	**Transaction ID** AT_BPMN_TRANSACTION_ID (2160)	
TransactionProtocol : String	**Transaction protocol** AT_BPMN_TRANSACTION_PROTOCOL (2161)	
TransactionMethod (Compensate | Store | Image) Compensate : String	**Transaction method** AT_BPMN_TRANSACTION_METHOD (2162)	

1.13.11 Web Service

There are two possible solutions to model the details of a web-service. The (not recommended) first version is using ARIS attributes at the Object which references the web-service.

For this first solution the attributes are maintainable at the ARIS objects Task and Event (if the task/event is triggered by an incoming sequence flow the web service is being executed, the target of the outgoing message flow is only used for the target participant of the message)

The attribute mapping for this solution is described in the following table.

Table 1.13-11: Web Service Attributes

Attributes	ARIS Attribute Name (en) API_NAME (Attr. Number)	Other Semantic equivalent in ARIS
Participant : Participant	**Participant Type** AT_BPMN_PARTICIPANT_TYPE (3426) Role / Entity AT_BPMN_ROLE_ENTITY (3427)	
Interface : String	**Interface** AT_BPMN_WS_INTERFACE (3400)	
Operation : String	**Operation** AT_BPMN_WS_OPERATION (3401)	

The second, more advanced approach is using the ARIS objects PartnerLink, Interface and Operation (inside an assigned BPMN Allocation Diagram). Those objects are part of the standard ARIS BPEL modeling technique. This second approach enables a user to reuse already defined ARIS BPEL content (for example which operation belongs to which interface and which interfaces are available at which partner links). An ARIS connection **is supported by** from the BPMN Task towards the BPEL operation (which is executed by the Task) enables a process engine/BPEL export engine to determine the exact BPEL operation (and their interface/partner link)

In this second approach the operation, interface and partner-link attributes of the web service

are fetched from the target of the message flow (the event which receives the message flow has connections towards the operation)

This second approach is described inside chapter BPMN Web Service .

The BPEL Export implementation should define this definition in more detail.

2 Examples / Explanations

2.1 Property Modeling in ARIS

Properties can be attached to a various BPMN elements (Process, Activity, Message etc.)

The BPMN properties of an object are modeled inside the BPMN Allocation Diagram (which MAY be assigned to the property owner).

Properties are modeled as separate ARIS objects which are connected with a **has** connection towards the object which encompasses the properties. Process properties are connected with the pool object which represents the BPMN/BPEL Process.

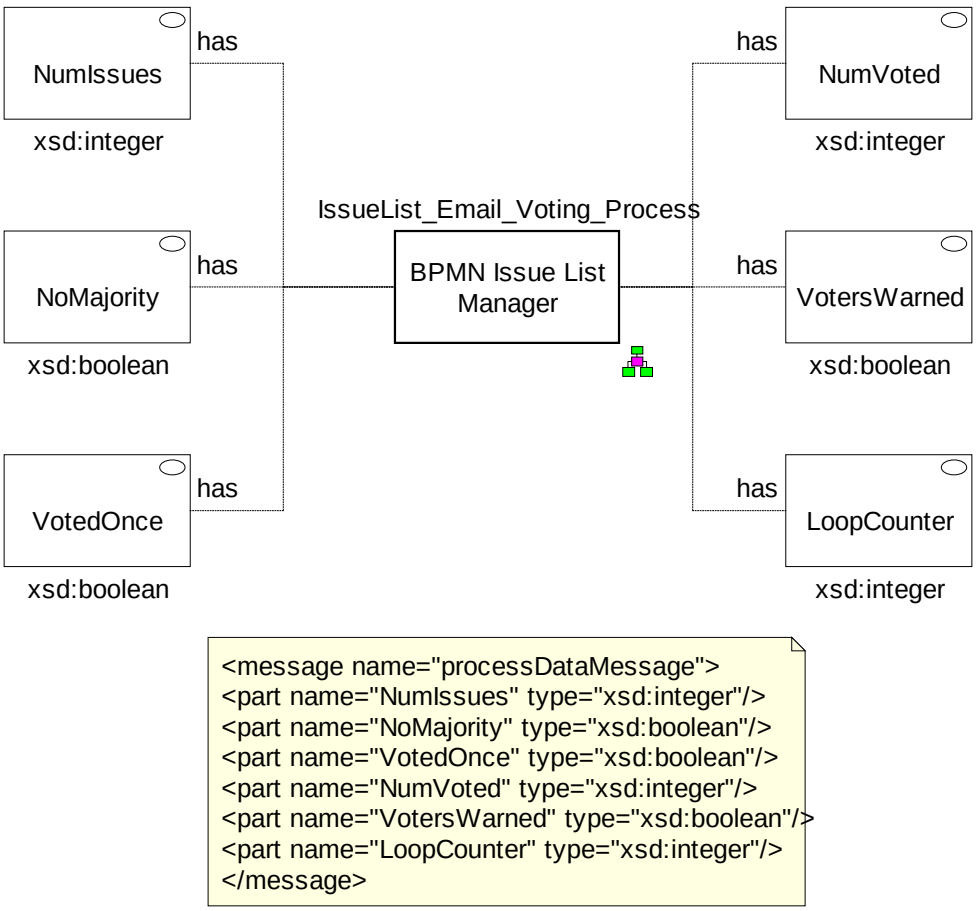


Figure 3: Process properties and their data types for the BPMN 1.0 Spec Example Process

2.2 Input / Output Property Maps modeling in ARIS

Property mapping defines which properties of the calling process should be transferred into which properties of the called sub-process. This definition is done graphically inside a BPMN

allocation diagram which may be assigned to the Independent sub process object.

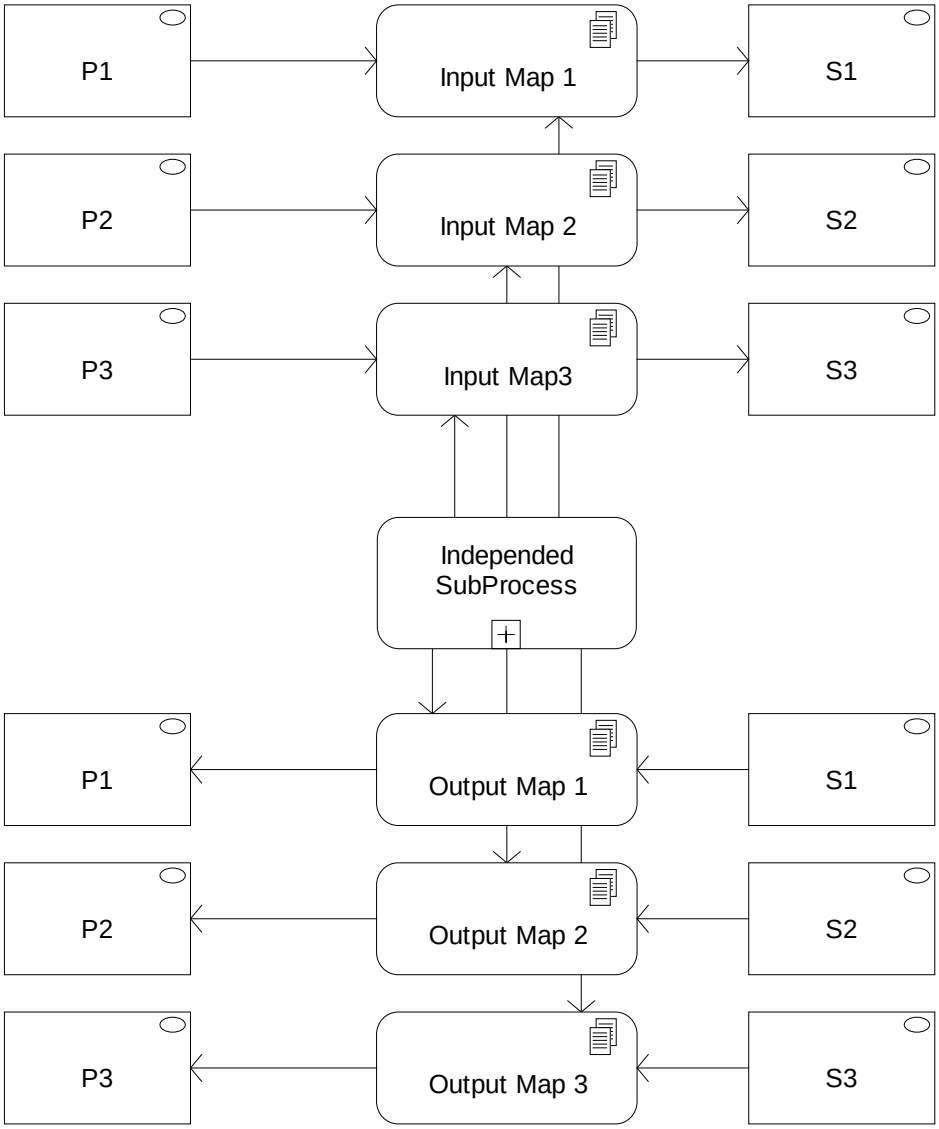


Figure 4: The Property Input/Output behavior of an independent sub process. P1, P2 and P3 are Occurrence copies of properties that are part of the parent process, S1-S3 are property objects that are part of the Independent Sub Process.
The properties P1, P2 and P3 are transferred into the properties S1, S2 and S3 of the Sub Process. After the Sub Process has been completed the content S1 is copied into P1, S2 is copied into P2 and S3 is copied (mapped) to P3.

2.3 Property Assignment Modeling in ARIS

BPMN Assignments specify which value should be assigned to which property (and have therefore a completely different semantic as the ARIS model assignments).

BPMN assignments are modeled with the **assigns** connection which creates a connection from the assigning object towards the target property. The actual value which should be assigned is defined at the connection attribute **from** (AT_BPMN_ASSIGN_EXPRESSION), the assign-time specifies if the assignment should be made before the start of the activity or after it was completed.

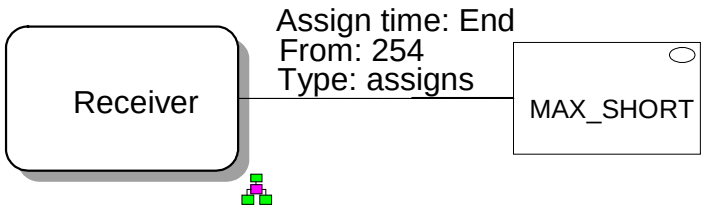


Figure 5: The activity "Receiver" assigns the value "254" to the property "MAX_SHORT" after "Receiver" has finished its processing

One Activity/Sub Process / Gateway / etc can have multiple property assignments. Each assignment is modeled with a separate connection towards the target property.

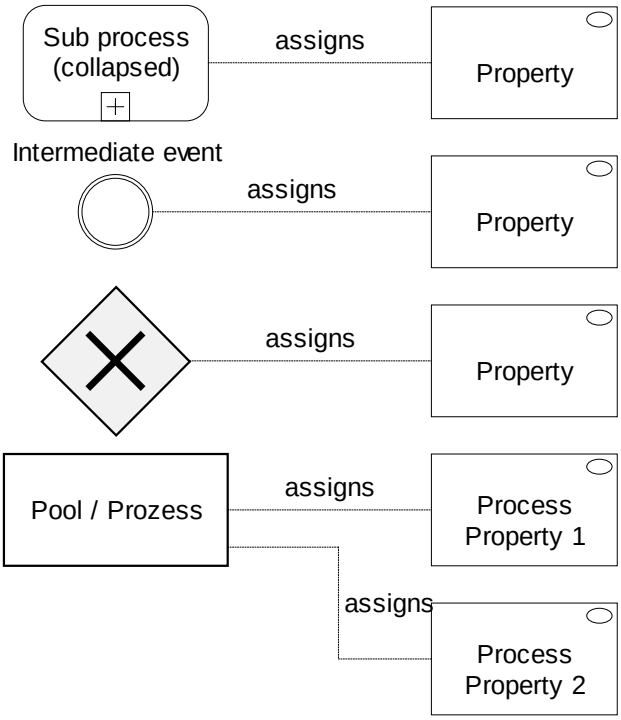


Figure 6: The properties can be attached to Activities, Events, Gateways and to the Process (represented as a pool)

2.4 Input Sets, Output Sets and IORules in ARIS

2.4.1 Explicit IORules

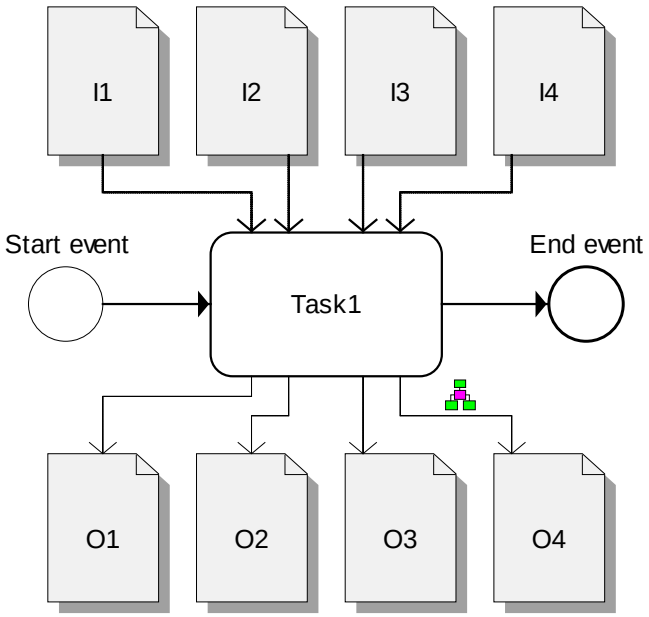


Figure 7: The example process - in the BPMN Diagram – Task 1 reads I1, I2, I3 and I4 and produces O1, O2, O3 and O4 - This diagram DOES NOT define which input creates which output and which input attributes can/must be available to execute the activity

The BPMN Specification defines

Each Activity can have [0-n] InputSets, each input Set is mapped to exactly one Output Set.

Input and Output Sets can contain [1-n] Artefacts, which Output Set is created by which InputSet is defined via [0-n] IORules.

This detailed IO behavior semantic is modeled inside a **BPMN Allocation Diagram, which is assigned to the Task for which the IO behavior should be specified** as following:

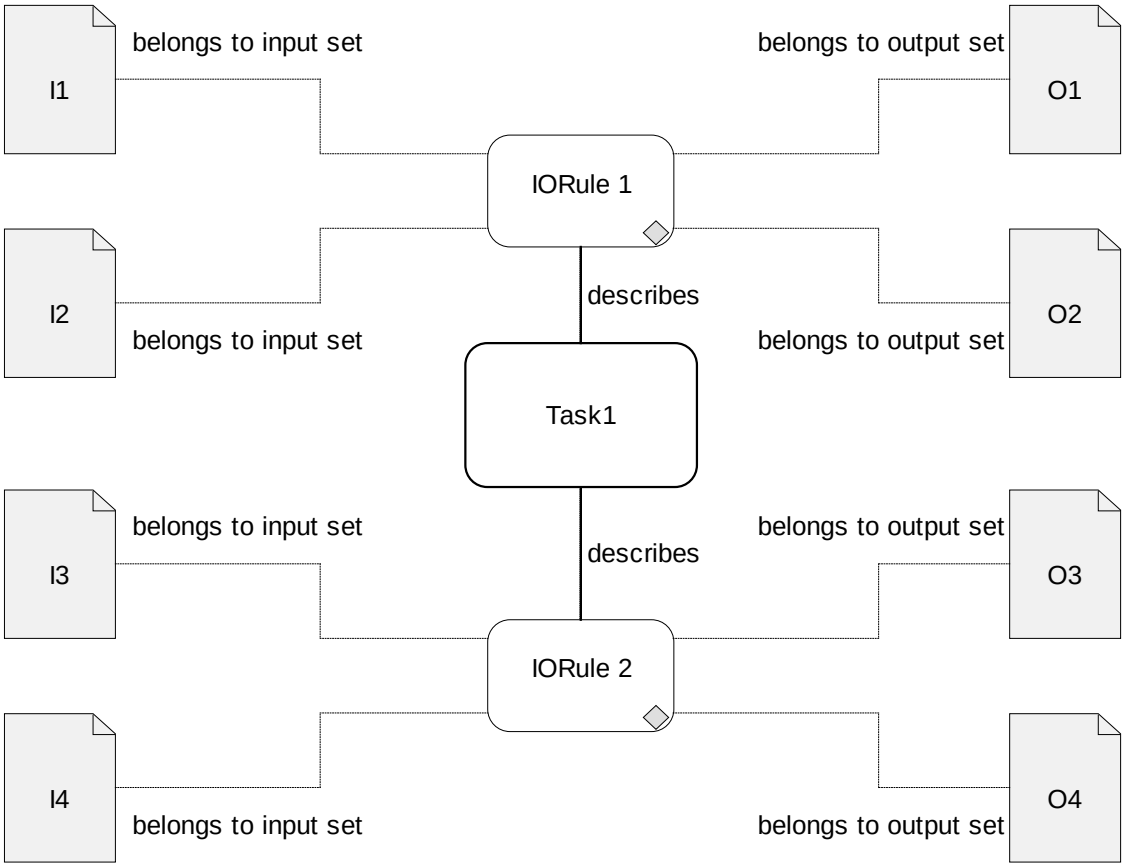


Figure 8: A Task with two input Sets ({I1, I2} {I3, I4}, two output sets ({O1, O2} {O3, O4} and two corresponding IORules (IORule 1, IORule 2) . Semantic: Task1 can be instantiated if I1 AND I2 are available OR if I3 AND I4 are available. If the Task is instantiated with I1 AND I2 it will produce O1 AND O2, if the Task was instantiated with I3 AND I4 it will produce O3 AND O4.

2.4.2 Implicit IO Rules

As a shortcut to this exact but complex modeling convention the following definitions allows a modeler to create well defined data IO definitions without the need to create an extra BPMN allocation diagram:

If an Artefact (I5) is connected directly towards a Task with an **provides input for** connection and that task is NOT connected towards an IORule object with an **belongs to input set** connection an implicit connection **belongs to** towards an invisible IORule object **Implicit IO Rule** is assumed.

If a Task is connected directly towards a Artefact (I6) with an **provides output to** connection and that Artefact is NOT connected towards an IORule object with an **belongs to output set** connection an implicit connection **belongs to output set** towards an invisible IORule object **Implicit IO Rule** is assumed.

In that case Figure 9 and Figure 10 represent the same IO behavior.

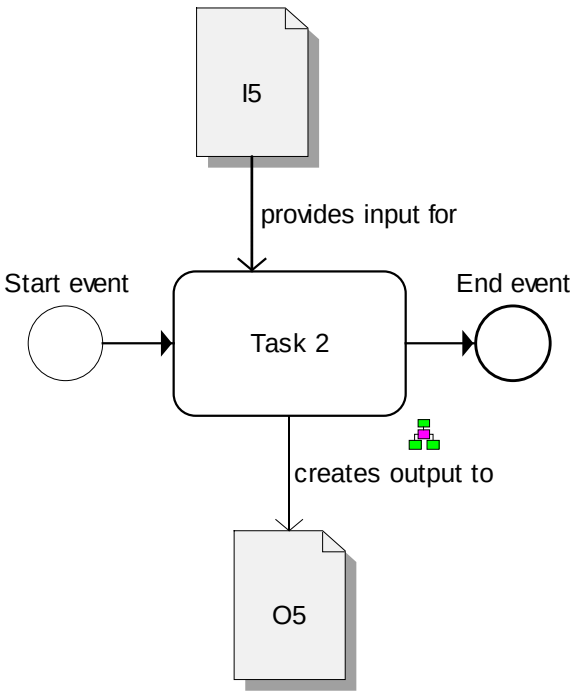


Figure 9: Simple Data Definition inside a BPMN Diagram – This diagram represents the same semantic than Figure 4, but can be modeled without an extra Allocation Diagram.

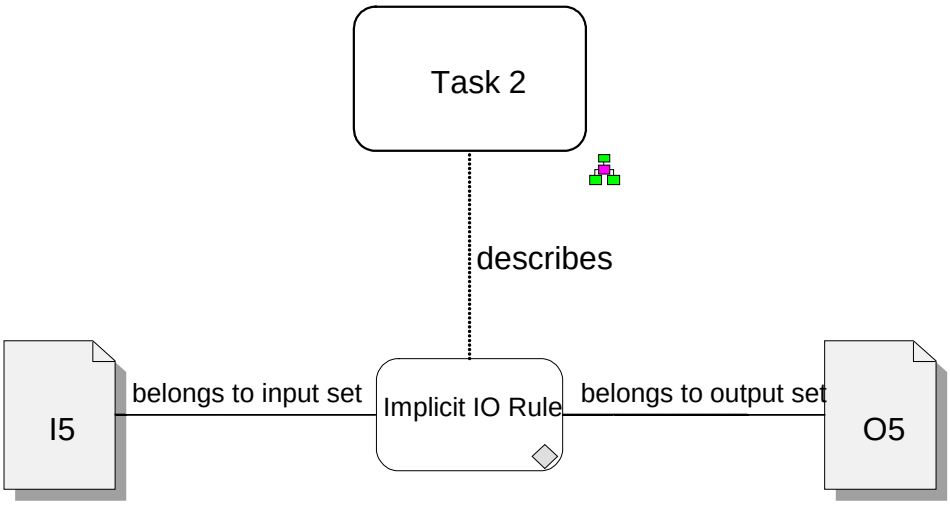


Figure 10: Exact modeling of Figure 11 inside a BPMN Allocation Diagram – this diagram represents the same semantic as Figure 12

The Input Property Map defines which properties of the parent process are copied to which properties of the independent target sub process (before the target process is executed).

The Output Property Map defines which properties of the independent sub process are copied to which properties of the calling parent process (after the target process has been completed).

ARIS does this mapping graphically inside a BPMN Allocation Diagram which has to be assigned to the sub process symbol.

2.5 BPMN Web Service Modeling (Version with dedicated ARIS objects)

The BPMN Specification defines a Web Service as a combination of a participant, an interface and an operation. Because those objects have their origin in the BPEL world the ARIS BPEL objects are being reused to represent Participant, Interface and Operation.

The concrete mapping (which BPEL Operation is represented by which BPMN Task) is done inside a BPMN Allocation Diagram. Figure 13 is the overall view of a BPMN Task that is a representation of a BPEL Operation.

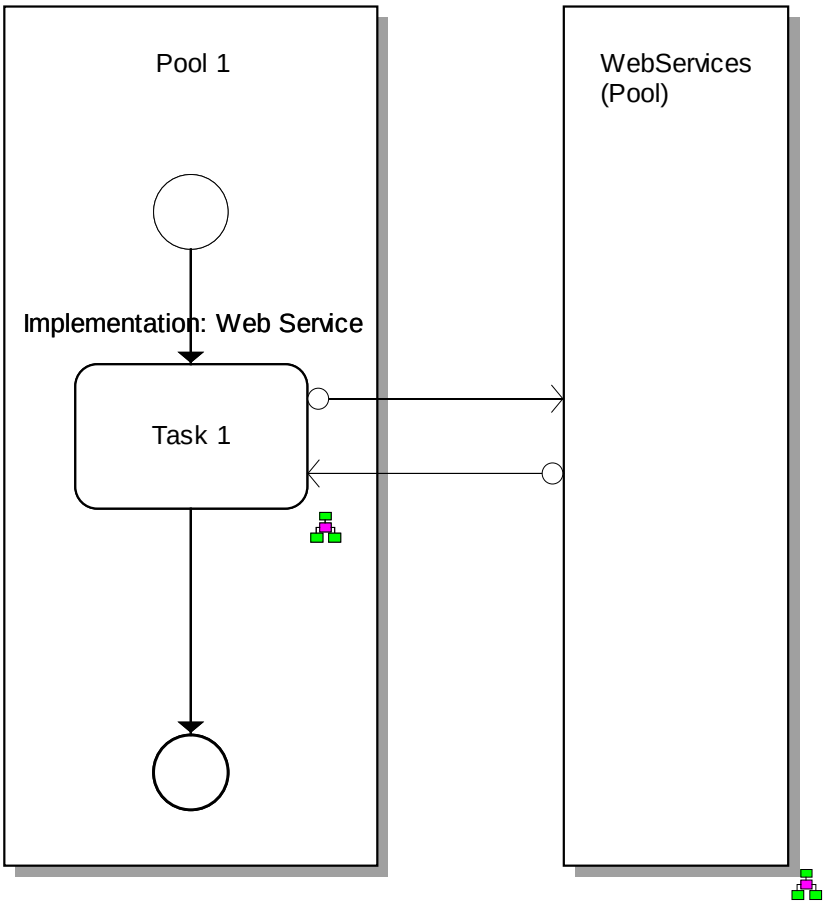


Figure 13: Task 1 has a Web Service Implementation; The Participant that provides the Web Service is **WebServices (Pool)** – the definition which operation is being executed is done inside the allocation diagram which is assigned behind Task1 (Figure 14)

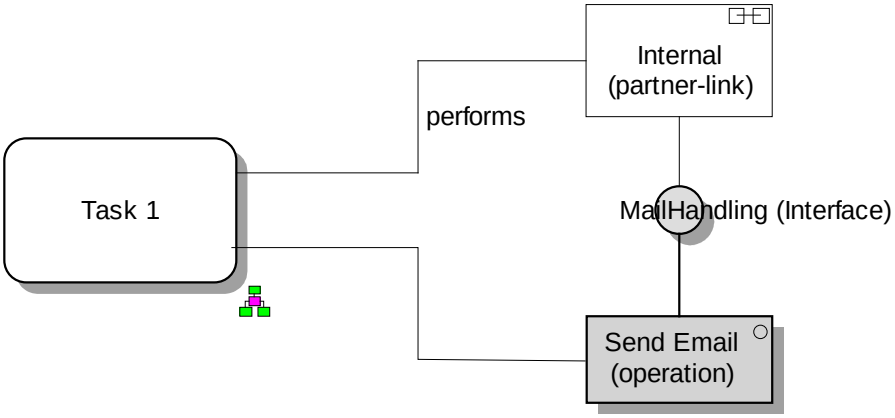


Figure 14: The allocation Diagram behind Task1 – Task 1 executes the Operation **SendEMail** from the Interface **MailHandling** on the PartnerLink **internal**

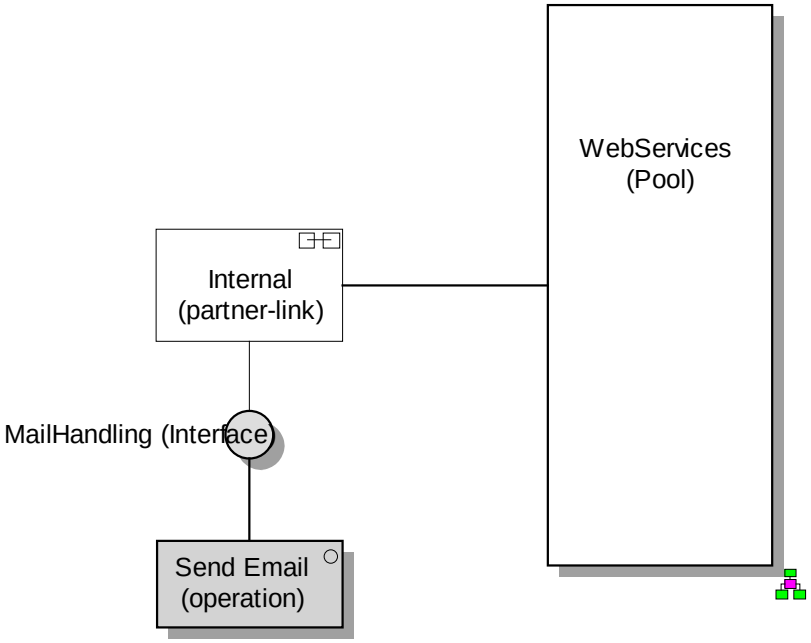


Figure 15: The BPMN Web Service Pool Object provides the **Internal** PartnerLink, this partner Link provides the Interface **MailHandling**, which provides the operation **SendEMail** This is modeled inside the BPMN Allocation Diagram which has to be assigned behind the WebService(Pool) Object.

3 Semantic Checks

Various ARIS semantic checks were developed to validate an ARIS BPMN models against the BPMN specification. Those semantic checks (and a semantic check profile which contains them) are currently distributed in separate files and can be imported via the usual import mechanism.

The following list contains the BPMN constrains which are being validated by those semantic checks.

3.1 Process / Pool

Structure Rules / Attributes general process

1	All Flow Objects (Events, Activities and Gateways) MUST HAVE a (semantically) connection towards a pool (or towards a Lane that is a sub-part of a pool) (it must be possible for an export writer to determine the pool of all objects) If an object is not placed inside a pool (has a belongs to connection towards a pool or a lane that is part of a pool) it may only be place inside a model which has a superior pool object. (the object inherits the pool through the ARIS assignment))	
2	if a process has one or more start events then there MUST NOT be other Activities inside the process that have no incoming sequence flows	(this may be a complex check which may need some clarification) → postpone
3	Pools which have the Attribute **Adhoc** set to True must have a maintained CompletionCondition Attribute	

Structure Lanes

4	Each lane must have a parent pool The parent pool of the lane can be either specified by the target of a belongs to connection (towards a pool) or by a superior pool object (a pool object is placed inside a superior model, the current model is assigned behind this pool object)	

3.2 Sequence Flow / Message Flow

Attribute Rules

5	For Sequence Flows which have the **Condition** Type set to **Expression** the attribute **Condition Expression** must be maintained	AT_BPMN_SEQ_FLOW_COND ITION AT_BPMN_CONDITION_EXPR ESSION
6	If a sequence flow has its attribute **Condition** set to **Default** / not maintained the Condition Expression MUST NOT be maintained	
7	A sequence flow which originates from an Event MUST NOT have a maintained **Condition** Attribute.	

Structure Rules

8	A Sequence Flow MUST NOT cross the boundary of a (sub)process (Source and Target of a Sequence Flow MUST not cross Pool boundaries (but they can cross lane boundaries)	(both elements are part of the same pool (direct or via lanes that belong to a pool) or both elements are part of the inherited pool (pool is placed in a superior model) For the expanded SubProcess: Both elements must be part of the same subprocess element.

9	A message flow MUST cross Pool boundaries	Inverse logic of the sequence flows.
10	If an activity has only one outgoing sequence flow this flow MUST NOT be conditional.	
11	Compensation, Cancel and Error Events which have an incoming **may trigger** connection must overlap the source of the **may trigger** connection. (share some pixels) The source of a normal **is created by** Sequence Flow Connections MUST NOT overlap its target. (share some pixels with the target object)	(from an activity / sub process towards an intermediate event)

3.3 Events

Attribute Rules events

12	Start Event must be of type Message, Timer, Rule, Link or Multiple Event or NONE (MUST not be Error, Compensation or Terminate)	
13	Intermediate Event must not be a Terminate Event	
14	End Event must NOT be a Timer or a Rule Event	
15	If the EndEvent is of type "ERROR" the ErrorID attribute must be maintained	(an error end event throws the error – therefore the errorID must be maintained)
16	If the result is of type "Link" the linkID must be maintained	
17	If an event is of type **Rule** the RuleExpression must be maintained	
18	For intermediate/start Timer Events TimeDate OR TimeCycle MUST be maintained (only one of both, not both together)	

Structure Rules Events

19	An intermediate compensation event must have an connection of type **can trigger** from an Activity towards the Intermediate Event and exactly one connection of type **activates** from the Event towards a Compensation Task	AT_BPMN_COMPENSATION_ACTIVITY
20	An intermediate error/cancel event must have a connection of type **can trigger** from a Transaction sub process towards the Intermediate Event.	
21	if a start event is of type "message" an incoming message MUST BE supplied.	
22	A **can trigger** connection MUST NOT have Intermediate Events of type **none** and **link** as their targets	
23	if a start event has one incoming message flows it MUST BE of type "message" or of type "multiple".	

24	if a start event has more than one incoming message flow it MUST BE of type "multiple"	
25	A start event MUST be the source of a sequence flow	
26		
27	The Cancel Result (end event) MUST NOT be used unless it is placed inside a SubProcess of type "Transaction"	Superior object has to be a transaction, or it has to be placed inside a (expanded) transaction
28	if there is an start event inside a BPD which is not placed inside a sub-process there MUST BE at least one end event (which is also not placed inside a sub-process)	
29	if an end event is used there MUST NOT be other activities that have no outgoing sequence flow connection (exception: compensation activities/compensation sub-processes (which MUST NOT have an outgoing sequence flow (see separate check)))	(this rule does not consider events which are embedded in sub-processes)
30	end events of type "message" MUST have an outgoing message connection	
31	End events MUST have an incoming sequence flow	
32	Intermediate events of type "NONE" and "LINK" must not be attached to the border of a sub process	(no incoming **can trigger** connection)
33	Intermediate Events of type "MULTIPLE" and "CANCEL" MUST NOT be used in normal sequence flow	(no incoming **leads to** ,**creates** and **occurs before** connection)
34	for intermediate events of type "Message" an incoming Message MUST be provided	(incoming message flow, direct or via a message object mandatory)
35	Intermediate Events of Type **Error** which doesn't have an incoming **can trigger** connection MUST HAVE the ErrorID Attribute maintained	

36	Intermediate events which have an incoming **can trigger** connection MUST NOT have other incoming sequence flow connections AT THE SAME OCCURENCE. (they may have one other incoming sequence flows at their second occurrences)	**leads to** ,**creates** or **occurs before** connection
37	All intermediate events may have a maximum of 1 incoming sequence flows (**leads to**, **creates** and **occurs before** connections), Intermediate Events of type **None** and **Error** MUST have exactly one incoming sequence flow	**leads to** ,**creates** or **occurs before** connection
38	An intermediate event MUST have exactly one outgoing sequence flow (at least at definition level)	
38b	Compensation and Link Event must have exactly 2 occurrences	(the flow is transferred from one occurrence towards the other one)

3.4 Activity

Structure Rules Activity

39	A Task of type **Receive** MUST NOT have an outgoing message flow AND must have an incoming message flow	(at definition level)
40	A Task of type **Send** MUST NOT have an incoming message flow and MUST HAVE an outgoing message flow	(at definition level)
41	A Task Type of **Script**, **Manual** or **None** MUST NOT have an incoming or an outgoing Message Flow	(at definition level)
42	A Task Type of **Service** OR **User** MUST HAVE an incoming and an outgoing message flow	(at definition level)
44	A Task MUST NOT have more than one incoming message flows	
45	A Compensation Task/Sub Process MUST NOT have any outgoing sequence flow connections and MUST have a incoming sequence flow connection from an compensation intermediate event.	

Attributes Rules Activity

46	The Activity Attribute **StartQuantity** MUST NOT be less than 1	
47	The Attribute **MI_FlowCondition** MUST NOT be maintained if the LoopType of an Activity is != Multiple Instance	
48	The Attribute **MI_Ordering** MUST not be maintained the LoopType is != multiple instance	
49	The attribute **Complex MI_FlowCondition** must not be maintained if the MI_FlowCondition is != Complex	
50	The Attribute **Complex MI_FlowCondition** MUST BE maintained if the MI_FlowCondition Attribute is set to **Complex**	
51	Embedded sub processes MUST NOT have a **performs** connection towards a Mapping symbol (inside a BPMN allocation diagram)	
52	A Task of Type Receive that has its Instantiate attribute set to True MUST NOT have a Standard or MultiInstance LoopType	
53	if an Sub Process is of the type **AdHoc** (Adhoc Attribute==true) the Attribute **AdHocCompletionCondition** must be maintained	
54	If the Attribute AT_BPMN_LOOP_TYPE is set to **Standard** the attribute AT_BPMN_LOOP_CONDITION must be maintained	(same for sub-processes)
55	If the Attribute AT_BPMN_LOOP_TYPE is set to **Multiple Instance** the attribute **AT_BPMN_MI_CONDITION** must be maintained	

3.5 Gateways

Attribute Rules Gateways

56	The attribute **instantiate** at gateways and tasks may only be set to TRUE if the element is the first element in a process, or if the element is the second element in a process and the first element is a start event.

Structure Rules Gateways (Splits and Joins)

57	An AND join must have at least 2 incoming and 1 outgoing sequence flow connection An AND split must have at least 1 incoming and 2 outgoing sequence flow connections	Splits and joins are the same ARIS objects. (2 or more incoming connections, == join, 2 or ore outgoing connections == split)
58	An event based gateway which acts as a split MUST lead to >1 events (of type message, timer, rule or link) OR towards >1 Task that are of the type **receive** OR to a combination of (one ore more) tasks (of type receive) and events (those events MUST not be message events if a receive task is part of the destination objects)	(set of target objects MUST not contain message events AND receive tasks, valid event types are message, timer, rule and link.)
59	For all outgoing connections of an data based XOR Split gateway the attribute **Condition** must be maintained, only one of those connections may have the **Condition** set to **Default**, the others MUST have an **expression** condition.	
60	All outgoing sequence flow connections of an event based gateway MUST HAVE their Condition unmaintained (==NONE condition)	
61	Each event based gateway must have a minimum of two outgoing sequence flow connections.	
62	If a Complex Gateway has multiple incoming sequence flow connection the attribute **IncomingCondition** must be maintained.	

63	If a Complex Gateway has multiple outgoing sequence flow connections the attribute **OutgoingCondition** must be maintained.	
65	If a gateway has only one outgoing sequence flow connection (JOIN) the attribute **Condition** must be unmaintained at all incoming and outgoing sequence flows AND it MUST have multiple incoming sequence flows	AT_BPMN_SEQ_FLOW_CONDITION
66	All outgoing sequence flow connection of an parallel gateway (AND-split) must have their condition attribute unmaintained	AT_BPMN_SEQ_FLOW_CONDITION

Property Modeling

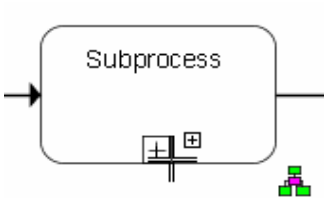
67	Each Property MUST be part of a pool or a Sub Process (there MUST BE exactly one **has** connection" from a Pool or Sub Process towards the property)

4 Collapsing / Expanding Embedded Sub-Processes

4.1 Expanding a Sub Process

The Expand feature provides an easy / quick way to expand the content of an embedded, assigned sub-process into the current BPD.

The features is accessible via a double-click on the plus sign inside the Expanded Sub-Process Symbol.



The Expand Sub-Process Features checks the content of the assigned BPMN model (assigned behind a collapsed sub-process symbol).

If all elements of the assigned model are embeddable into an Expanded Sub Process (pools and lanes are not embeddable) the expand feature changes the symbol of the collapsed sub-process into an expanded sub-process and transfers the content of the assigned model into this expanded sub-process.

Because the content of the assigned model needs usually much more space than the collapsed sub-process symbol the symbol is enlarged (so that it can surround the content of the assigned model).

After the symbol has been enlarged the content of the assigned model is transferred into the expanded sub-process symbol.

Each element is automatically embedded into the expanded sub-process symbol with a belongs to connection.

After all elements are embedded into the Expanded Sub Process the assigned model is deleted (the content of this model is now part of the expanded sub process symbol)

Figure 16, Figure 17 and Figure 18 are showing the transformation from a collapsed sub process with an assigned BPMN model towards an expanded sub process symbol.

4.2 Collapsing a Sub Process

The Collapse Sub Process feature provides an easy way to collapse the content of an expanded sub-process into a separate (assigned) BPMN diagram. It provides therefore the reverse functionality of the expand feature. The collapse feature is accessible via a double-click on the “minus” sign of the expanded sub-process symbol.

During the collapse procedure a new BPMN Model is created and automatically assigned behind the collapsed sub-process. All elements which were embedded into the expanded sub-process (with a belongs to connection) are transferred into the new (assigned) model.

After the transfer into the assigned model the belongs to connections between the transferred elements and their sub-process is deleted.

Because the collapsed sub-process symbol needs usually less modeling space than the expanded version the features tries to remove the free space around the collapsed symbol – however this is not always possible (because there is not enough free space that can be removed).

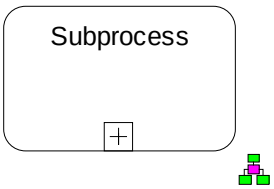


Figure 16: Collapsed Sub-Process with an assigned BPMN Diagram

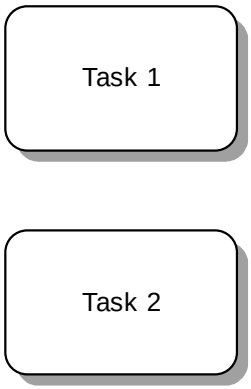


Figure 17: Content of the collapsed sub process in a separate (assigned) diagram.

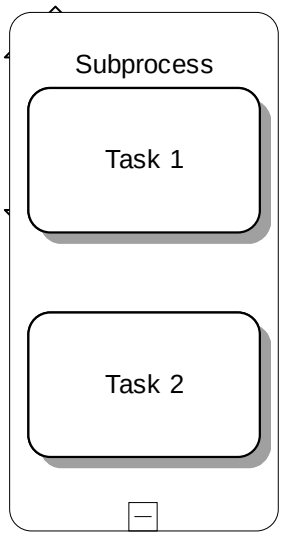


Figure 18: Expanded Sub Process with 2 Embedded Tasks