<AutomationML/>

The Glue for Seamless Automation Engineering

Application Recommendation: AAS Representation

Document Identifier: AR AAS, V 1.0.0

State: November 2019

Version 1.0.0, November 2019

Contact: www.automationml.org

Table of contents

List of figures

List of tables

1 Introduction

An essential feature of the Industry 4.0 approach is the representation of functional and non-functional assets within a production system through an asset administration shell. Within the life cycle of the production system this asset administration shell needs to be exchanged as visualized in the following figure.

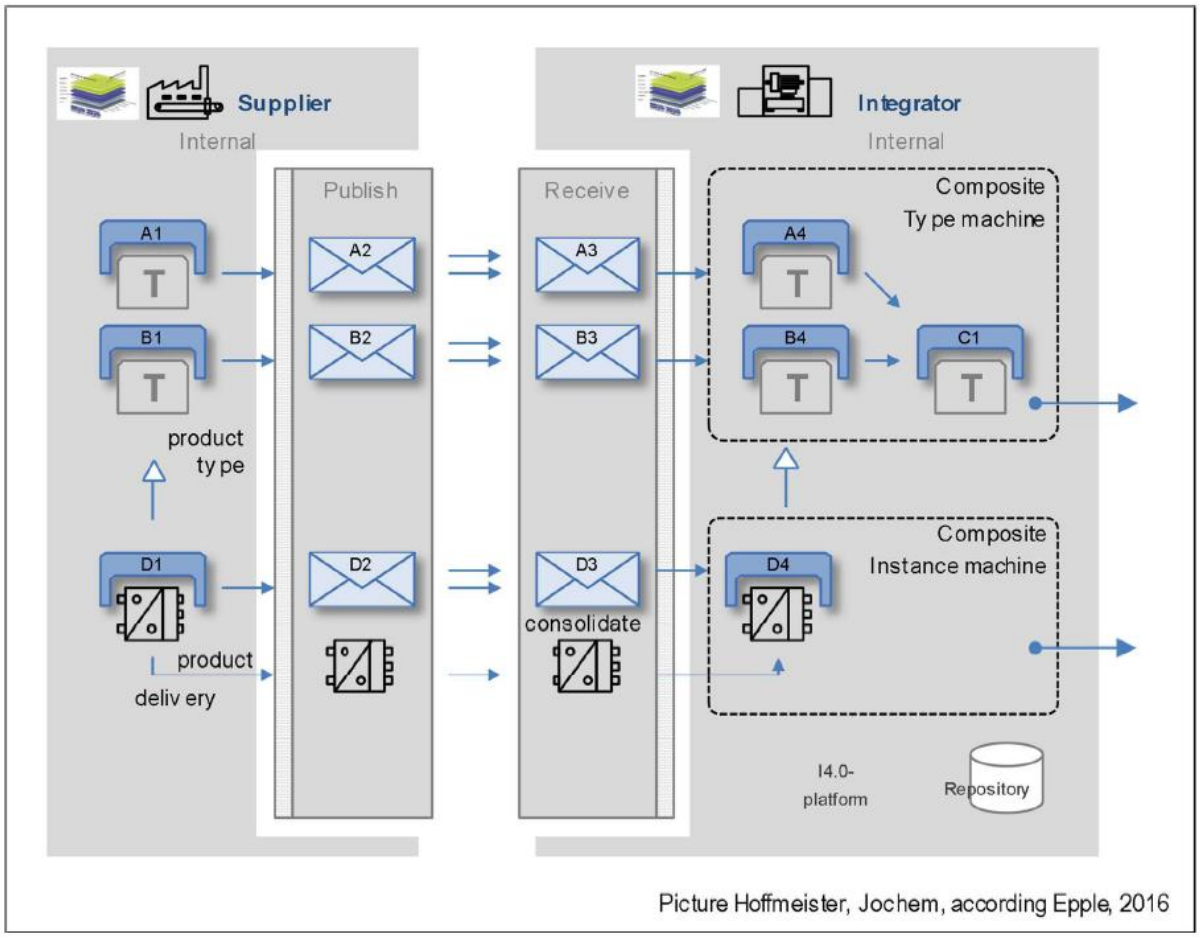

Figure 1 Exchange of information between partners in the value chain

According to *[VWSiD2.0]* with the "Supplier" (going to provide some products) and the "Integrator" (going to utilize these products in order to build a machine) at least two partners are involved in the information exchange transferring two sets of information related to types of assets and related to actual asset instances.

Essential within this information exchange of an asset administration shell is the need of a proper data structuring and the use of proper data storage formats. In *[VWSiD2.0]* the data structuring has been presented by providing a Metamodel for the asset administration shell data. The aim of this document is to specify the way of use of the data exchange format AutomationML for representing, storing, and exchanging the asset administration shell following the named Metamodel.

Therefore, this document provides a serialization of the asset administration shell Metamodel using AutomationML by defining appropriate serialization rules and role, interface and system unit classes related to them.

AutomationML is best suited to be used as the serialization format in the engineering phase.

This document was created in a collaboration of Plattform Industrie 4.0 and AutomationML association. The mapping rules are also published in *[VWSiD2.0]*.

1.1 Basics

The data exchange format AutomationML which is standardised as per IEC 62714 standard, is a neutral, free, and XML-based data format. It has been developed in order to support the data exchange between engineering tools in a heterogeneous engineering tool landscape.

Due to the different aspects of AutomationML the IEC 62714 consists of different parts.

The table below lists the parts which are relevant for this document.

Table 1 – Overview of AutomationML parts

Part / Document Identifier	Title	Description
Part 1 / WP Arch, V 2.0.0	Architecture and general requirements	This part specifies the general AutomationML architecture, the modelling of the engineering data, classes, instances, relations, references, hierarchies, basic AutomationML libraries and extended AutomationML concepts.
Part 2 / WP Lib V 2.0.0	Role class libraries	This part specifies additional AutomationML libraries.
Whitepaper / WP eClass V 1.0.0	AutomationML and eCl@ss integration	This Whitepaper describes the integration of eCl@ss in AutomationML
Best Practice Recommendation / BPR MlingExp V 1.0.0	Multilingual expressions in AutomationML	This best practice recommendation describes the handling of different texts for different languages in AutomationML
Best Practice Recommendation / BPR EDRef V 1.0.0	External Data Reference	This best practice recommendation describes how to reference external data from AutomationML
Application Recommendation AutomationProject Configuration / AR APC V 1.2.0	AutomationML and ECAD integration	This application recommendation describes the data exchange between ECAD and PLC systems

1.2 Scope

This application recommendation provides a modelling methodology for representing the information content of an asset administration shell following *[VWSiD2.0]* by the engineering data format AutomationML. It will describe the recommended use of role, interface, and system unit classes as well as the recommended structures to be considered within the instance hierarchy as well as the system unit class libraries of an AutomationML project.

This application recommendation is intended to enable the implementation of proper asset administration shell serialization following the described modelling methodology to be applied especially during the engineering time of a production system and its components.

1.3 References

The following documents are referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Extensible Markup Language (XML) 1.0:2004, W3C Recommendation (available at <http://www.w3.org/TR/2004/REC-xml-20040204/>)

IEC 62424:2008, Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools

Whitepaper AutomationML Part 1 – AutomationML Architecture, April 2016

Whitepaper AutomationML Part 2 – AutomationML Role Libraries, October 2014

Whitepaper AutomationML – AutomationML and eCl@ss Integration, November 2015

Best Practice Recommendation Multilingual expressions in AutomationML, March 2017

[BPR EDRef1.0.0] Best Practice Recommendation External Data Reference, July 2016

[AR APC1.2.0] Application Recommendation Automation Project Configuration, V1.2.0 November 2019

Whitepaper AutomationML – Automation Component Description, V1.0.0, upcoming

[VWSiD2.0] Plattform Industrie 4.0 (2019) Details of the Asset Administration Shell. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 2.0). Federal Ministry for Economic Affairs and Energy (BMWi)(Hrsg.), ZVEI & Plattform Industrie 4.0. https://www.plattform-i40.de/PI40/Navigation/EN/InPractice/Online-Library/online-library.html. upcoming

[ETFA2017] Wagner C., Grothoff J., Epple U., Drath R., Malakuti S., Grüner S., Hoffmeister M., Zimmermann P.: The role of the Industry 4.0 Asset Administration Shell and the Digital Twin during the life cycle of a plant. In: Proceedings of the 2017 IEEE Conference on Emerging Technologies Factory Automation (ETFA 2017), Limassol, Cyprus, 2017.

2 AutomationML and Asset Administration Shell

2.1 AutomationML

AutomationML is a free and neutral file format. It has been initiated 2006 by the Daimler AG in the domain of manufacturing industry. It aims for modelling various engineering information of different domains and is standardized in the IEC62714.

AutomationML has a lean and distributed file architecture. It combines existing established XML data formats which are proven in use for their specific domain. The AutomationML standard does not define any new file format, instead it defines how the sub data formats are interconnected. This is why the normative part of the IEC62714-1:2018 document consists of 32 pages only. The data formats for the following modelling domains are:

- object topologies: CAEX according to IEC62424
- geometries and kinematics: COLLADA 1.4.1 and 1.5.0 (ISO/PAS 17506:2012)
- discrete behavior: PLCopen XML 2.0 and 2.0.1

CAEX according to IEC62424 forms the base of AutomationML. It stores object oriented engineering information, e.g. a plant hierarchy structure. Each CAEX object can reference external files that contain geometry, kinematics, logics information and even any “other” data as Word/Excel/PDF/binary file. This enables cross domain modelling and is designed for future extension.

The backbone format of AutomationML CAEX provides strong flexibility to model any kind of data. Hence, it allows for modeling data and libraries for a wide range of domains and is not bound to a specific industry. Two key properties of AutomationML are substantial in its application in a heterogeneous tool landscape:

- The modelling of *mixed semantics* – this means that standard classes and proprietary classes are modelled in the same way, and can be stored together in the same AutomationML file. This allows to standardize where needed and still model and transport proprietary data that can only be interpreted by prepared target tools.
- The labelling of AutomationML files – this means that every AutomationML file gets a meta information block containing information about the creator of the file. This allows any receiver to distinguish AutomationML files from different tools and to identify the needed semantics or dialect.

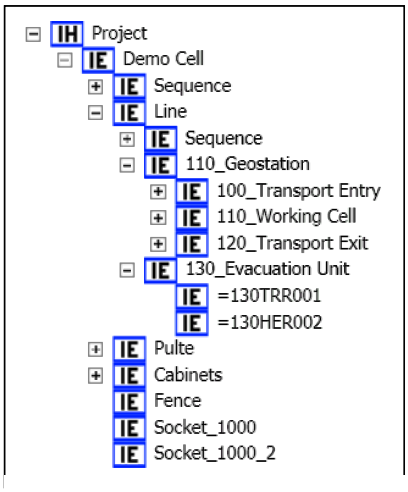

Figure 2: Example plant topology

Object hierarchies in CAEX form the core of AutomationML. An CAEX object is a data representation of any asset. It can model physical assets, e.g. a motor, a robot, a tank; or abstract assets as a function block or a folder. CAEX allows to combine those objects to systems, since every physical or logical system is characterized by internal elements (objects) which may contain further internal elements, and all elements may have interfaces, attributes and connections with each other. Finally, CAEX allows to model any plant topology, communication topology, process topology, resource topology etc., see Figure 2.

The same mechanisms can be used to model the topology of a single device: its inner topology, the signals, related behavior, and can reference all related documents for the device. In the result, AutomationML allows to define a powerful information model that is nothing less than a comprehensive self-description of the related device.

Meanwhile, AutomationML has reached industrial application in a wide range of applications, e.g. in virtual commissioning, in data synchronization across multiple engineering tool platforms, in the modelling of communication networks, in the extended modelling of PCE requests, in the modeling of electrical or pneumatical interfaces or in the comprehensive modelling of any automation components.

2.2 Asset Administration Shell

Industry 4.0 aims for introducing internet technology into production in order to utilize the benefits and concepts of the internet technology for industrial purposes. The *Asset Administration Shell* (AAS) is a core concept of this strategy: it is a *digital representation* of the *Asset* of an *Industry 4.0 component* within an *Industry 4.0 System*. What does this mean? Two key aspects characterize the AAS.

1. The AAS is a software layer, similar to a printer driver. While a printer driver connects a printer to an operating system, the AAS connects a proprietary device (asset) with an Industry 4.0 system. The AAS makes any device an Industry 4.0 component. For this, the AAS must, on the one hand, communicate with the device utilizing its proprietary communication channels. On the other hand, it must provide standardized Industry 4.0 software interfaces that are compatible to the Industry 4.0 system. The AAS makes a device searchable, explorable and controllable by means of Industry 4.0 services. The main feature of the AAS concept is that all devices across multiple vendors can be connected, explored and treated in the same way on-line across the Industry 4.0 network. This allows a huge range of innovations. The possibilities are initially visible in the IoT world of connected devices (mobile phones, home devices, connected cars etc.).
2. The AAS contains or references information about the device: e.g. parameters, geometry data, handbooks, function blocks, certificates, simulation models, simply any kind of information that are of value during its life cycle in engineering, operation, maintenance. Those information are called sub-models. This is a key difference between the AAS and a printer driver that usually does not contain any data about the printer.

In the result, the AAS can represent the device, it becomes a digital twin. In best case, it can comprehensively represent the device even when the physical device is not present. This helps to test and develop industrial systems before the real devices are in place. Figure 3 summarizes the term AAS and explains related terms.

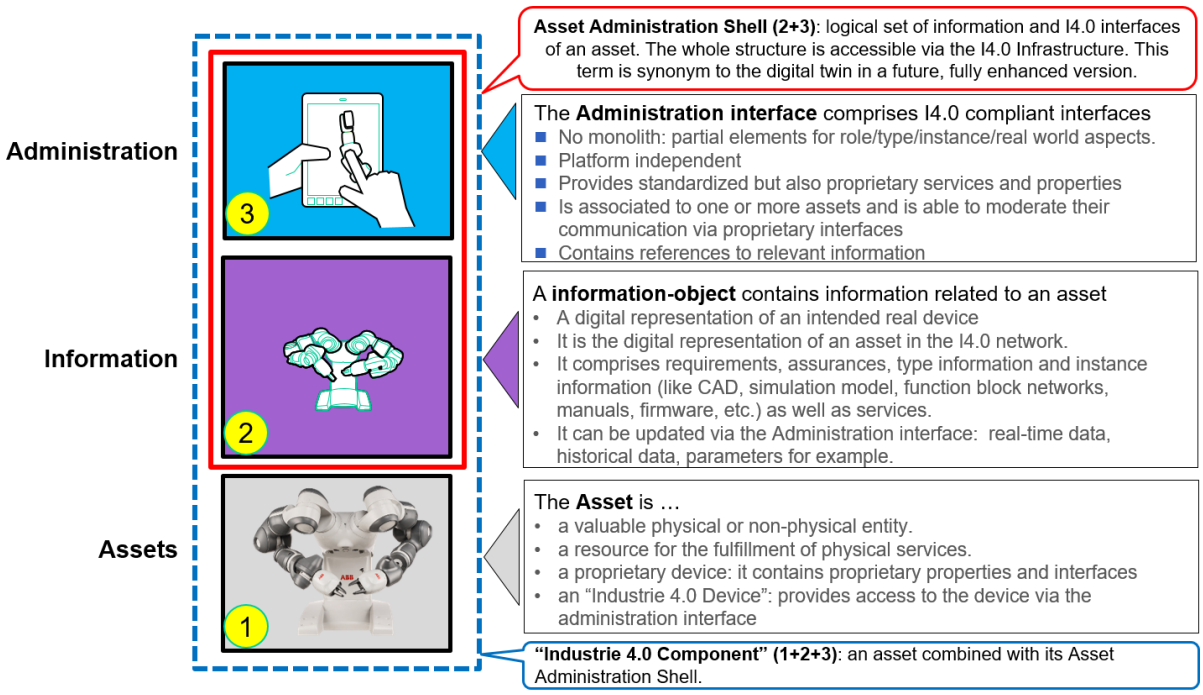

Figure 3: Asset Administration Shell and related terms (Source: [ETFA2017])

3 Asset Administration Shell Metamodel

In this clause a coarse overview of the main concepts of the Asset Administration Shell metamodel is presented. This clause as well as the detailed description of the metamodel can be found in *[VWSiD2.0]*.

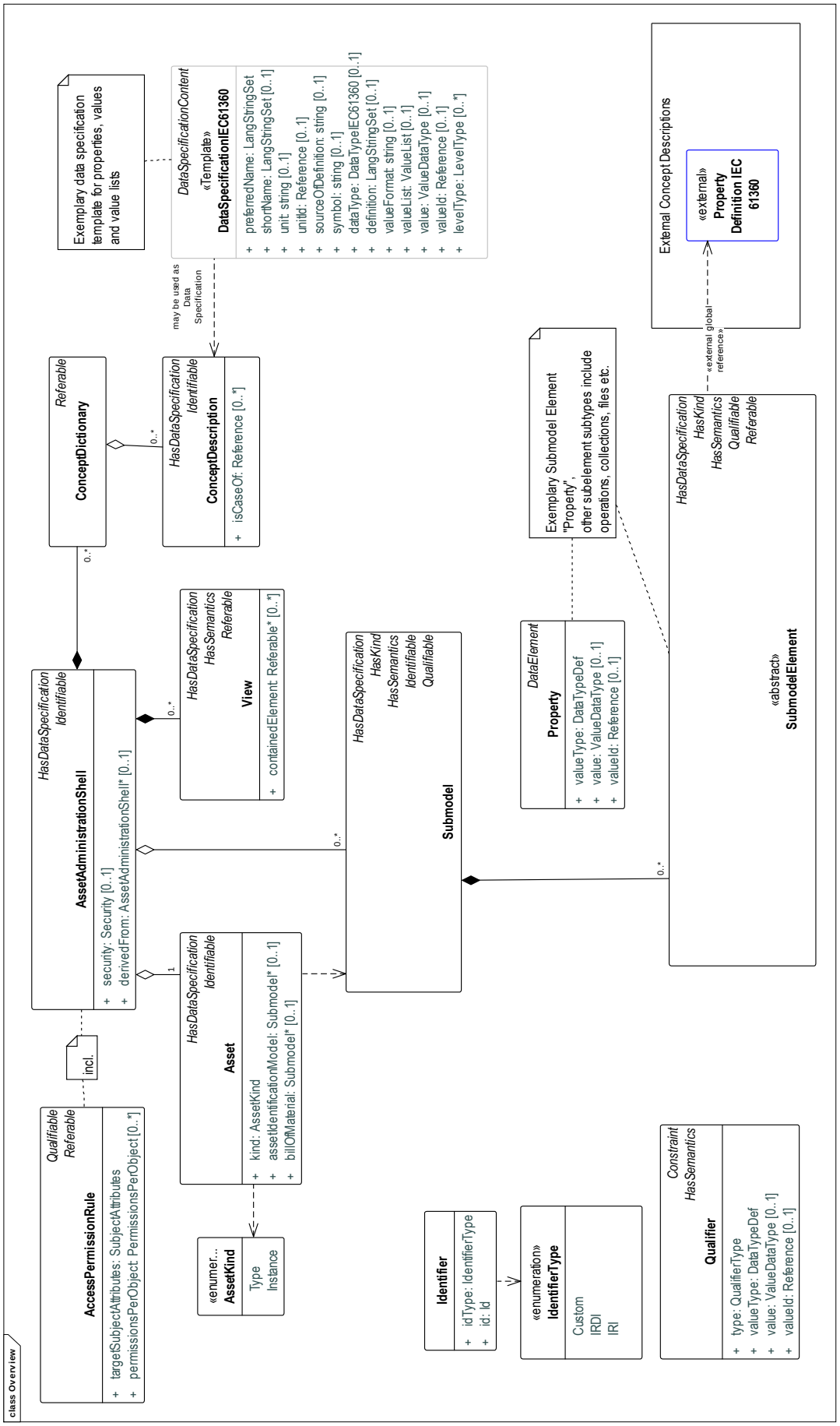
Class Overview
Exemplary data specification template for properties, values and value lists
DataSpecificationContent
«Template»
DataSpecificationIEC61360
+ preferredName: LangStringSet
+ shortName: LangStringSet [0..1]
+ unit: string [0..1]
+ unitId: Reference [0..1]
+ sourceOfDefinition: string [0..1]
+ symbol: string [0..1]
+ dataType: DataTypeIEC61360 [0..1]
+ definition: LangStringSet [0..1]
+ valueFormat: string [0..1]
+ valueList: ValueList [0..1]
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
+ levelType: LevelType [0..*]
may be used as Data Specification
External Concept Descriptions
«external» Property Definition IEC 61360
«external global reference»
Referable
ConceptDictionary
0..*
HasDataSpecification
Identifiable
ConceptDescription
+ isCaseOf: Reference [0..*]
Exemplary Submodel Element "Property", other subelement subtypes include operations, collections, files etc.
HasDataSpecification
HasKind
HasSemantics
Qualifiable
Referable
«abstract»
SubmodelElement
HasDataSpecification
HasSemantics
Referable
View
+ containedElement: Referable* [0..*]
HasDataSpecification
HasKind
HasSemantics
Identifiable
Qualifiable
Submodel
DataElement
Property
+ valueType: DataTypeDef
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
HasDataSpecification
Identifiable
AssetAdministrationShell
+ security: Security [0..1]
+ derivedFrom: AssetAdministrationShell* [0..1]
incl.
Qualifiable
Referable
AccessPermissionRule
+ targetSubjectAttributes: SubjectAttributes
+ permissionsPerObject: PermissionsPerObject [0..*]
HasDataSpecification
Identifiable
Asset
+ kind: AssetKind
+ assetIdentificationModel: Submodel* [0..1]
+ billOfMaterial: Submodel* [0..1]
«enumer...
AssetKind
Type
Instance
Identifier
+ idType: IdentifierType
+ id: Id
«enumeration»
IdentifierType
Custom
IRDI
IRI
Constraint
HasSemantics
Qualifier
+ type: QualifierType
+ valueType: DataTypeDef
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
1
0..*

Figure 4 Coarse overview Metamodel of the Asset Administration Shell

The main parts of an Asset Administration Shell (AAS) is the asset it is representing as well as the submodels. Optionally, dictionaries and views may be part of the AAS. A dictionary contains concept descriptions used to describe the semantics of submodels. Views define a set of elements selected for a specific stakeholder.

An AAS represents exactly one asset. Asset types and asset instances are distinguished by setting the attribute "*kind*".

Note: the UML modelling uses so-called abstract classes for denoting reused concepts like "HasSemantics", "Qualifiable" etc.

In case of an AAS of an instance asset, a reference to the AAS may be added, which represents either the asset type or another asset instance it was derived from (*derivedFrom*). The same holds for AAS of an asset type: also types can be derived from other types.

An asset typically may be represented by several different identification properties like for example the serial number, its RFID code etc. Such local identification properties are defined in the asset identification submodel (*assetIdentificationModel*).

AASs, assets, submodels and concept descriptions need to be globally uniquely identifiable (*Identifiable*). Other elements like for example properties, single local dictionaries just need to be referable within the model and thus only need a local identifier (*idShort* from *Referable*).

*Submodel*s consist of a set of submodel elements. Submodel elements may be qualified by a so-called *Qualifier*. There are different subtypes of submodel elements like properties, operations, collections etc. A typical submodel element is shown in the overview figure: a property. A property is a data submodel element that has a value of simple type like string, date etc.

Every submodel element needs a semantic definition *(semanticId* in *HasSemantics*). The submodel element might either refer directly to a corresponding semantic definition provided by an external reference (e.g. to an eCl@ss or IEC CDD property definition) or it may reference a submodel element of *kind* = *Template* that defines the semantics of submodel elements of *kind* = *Instance*.

The AAS itself can also define its own dictionary that contains semantic definitions of its submodel elements. These semantic definitions are called concept descriptions (*ConceptDescription*). It is optional whether an AAS defines its own concept dictionary (*ConceptDictionary*) or not.

The concept dictionary may contain copies of property definitions of external standards. In this case a semantic definition to the external standard shall be added (*isCaseOf). isCaseOf* is a more formal definition of *sourceOfDefinition* that is just text.

The concept dictionary may also contain proprietary definitions. In this case the provider of the AAS shall be aware that no interoperability with other AAS can be ensured.

Data Specification Templates (*dataSpecification*) can be used to define which attributes (besides those predefined by the metamodel) are used to define a submodel element or a concept description. For the concept description of properties typically the Data Specification Template following IEC 61360 is used. For denoting recommended Data Specification Templates to be used the <<*template*>>-dependency is used.

Some Data Specification Templates like the template for IEC 61360 property definitions (*DataSpecificationIEC61360*) are explicitly predefined and recommended to be used by the Plattform Industrie 4.0. If proprietary templates are used, again, interoperability with other AAS cannot be ensured.

Besides submodel elements including properties and concept descriptions also other identifiable elements may use additional templates (*dataSpecification*).

Submodel elements and the submodels themselves may have additional qualifiers (*Qualifiable*). Per *Qualifiable* there might be more than one qualifier.

Additionally, Views can be defined within an AAS. Views may consist of any elements that are referable (*containedElement*). A "Safety View", for example, contains all properties or operations that are safety relevant and need special treatment. A View definition can also be used in different life

cycle stages. For example, there could be a view for engineering and all referenced artefacts are deleted before delivering the AAS to the customer.

For every AAS security aspects need to be considered (*security*). In *[VWSiD2.0]* the aspect of access control is covered in more detail. So-called access permission rules are defined, that define which permission a specific authenticated subject has on which object.

Figure 4 gives an overview of the core elements defined in the metamodel excluding security. Security topics are not yet included in this first mapping of AAS to AML.

4 Mapping Rules

The rules for mapping of the AAS information model to AutomationML information model are given in the following. In subsequent clauses examples for the rules are given. For reasons of simplicity the term AML element is used for either InternalElements or for SystemUnitClasses or interfaces (or all of them) depending on the context.

4.1 Generic Rules for mapping:

(1) If present, AML role class and attribute name are taken from the AAS metamodel.

(2) If present, AML element names are the same as the value of idShort information from the AAS. If not present, a sufficiently unique element name is to be generated.

(3) Attributes of AAS Classes are modelled as attributes of AML elements.
The morphology of the AML and AAS information is in principle the same.

(4) Semantics of AML attributes is given by the AML RefSemantic attribute. Semantics of AML elements are defined via AML **role** and **interface classes**.

Example for RefSemantic values: AAS:Qualifier, AAS:Qualifier/type, AAS:Qualifier/value. For a complete list of all RefSemantic values see Appendix A.

Note: RefSemantic is not identical to semanticId in AAS. The difference is explained in clause 6.2.5.

(5) Attributes on AML Elements are created only if required.
Attributes, such as category, are only created for AML elements, if values are present in the AAS and vice versa.

(6) Values of Attributes in AAS which are of type "Reference" are serialized as string.

- The rules for serialization can be found in *[VWSiD2.0].*
- Example: *(Submodel)(local)[IRI]*www.myuri.de
- **Every internal element that represents information of the AAS metamodel.** A Tool with AASX-Import would search only for the AAS roles in the AML file. However, an AutomationML tool could also export/write a file containing AAS roles and other roles/models.

4.2 Rules for elements other than SubmodelElements of the AAS:

(7) Qualifiers are mapped on instance level to a complex attribute in AML
- Name of top-most hierarchy attribute: **qualifier:<value of AAS:Qualifier/type>=<value of Qualifier/value>**. Example: "*Qualifier:PredicateRelation=GREATER_THAN_0*"
- Subordinate attributes of the qualifier are mapped to subordinate attributes of the attribute in AML, e.g. for qualifier type, value with according AML RefSemantic.

(8) View as InternalElement with RoleClass View groups mirror elements
The internal elements with the assigned role class *"View"* group one to many mirror elements as a child. The target of the mirror element is the AAS entity identified by the AAS View as a contained element. A mirror element points with its *RefBaseSystemUnitPath* attribute to the UUID of the corresponding AML element.

4.3 Rules for subtypes of AAS SubmodelElement:

(9) All **AAS SubmodelElements** are mapped to AML InternalElements with an associated role class equal to the respective SubmodelElement subtype (e.g. AAS Property to AML Role Class Property).

(10) For submodel element **File** an interface **FileDataReference with its predefined attributes refURI and MIMEType** is used for referencing the file. Attribute value of submodel element File is not needed.

(11) For submodel elements **ReferenceElement** and **RelationshipElement** the interface "**ReferableReference**" is used to reference to the corresponding objects. In case of a local reference (AAS:Key/local = True) the interface "ReferableReference" needs to be set and is pointing to the corresponding element within AutomationML via an InternalLink. In this case, the value is empty. In case of an external reference (AAS:Key/local = False) no InternalLink is set for the interface. Instead, the *value* attribute carries a serialization of the AAS Reference.

(12) For **Operation**, the Operation is mapped to an InternalElement in AML with Role Class *Operation*.

- *input/output/inoutputVariable* attributes of AAS Operation are mapped to InternalElements in AML with RoleClass *OperationInputVariables, OperationOutputVariables* resp. *OperationInoutputVariables.* These InternalElements contain subordinated InternalElements for the submodel elements.

(13) **SubmodelElementCollection** is mapped to an InternalElement in AML with RoleClass *SubmodelElementCollection*. It contains subordinated InternalElements for contained elements.

4.4 Rules for the instance hierarchy of AML:

(14) For an AASX Tool with AML Export typically an instance hierarchy needs to be created. This instance hierarchy needs a name. If there is no other naming convention or no existing instance hierarchy that shall be used a possible name for the InstanceHierarchy is "*AssetAdministrationShellInstanceHierarchy*". It contains the asset administration shells containing elements of kind=Instance. Elements within this hierarchy have the role "*AssetAdministrationShell*".

Note: The AML import to an AASX Tool just needs to look for the role "*AssetAdministrationShell*", not for the names of the instance hierarchies.

(15) For an AAS with concept descriptions an AASX Tool with AML Export needs to create an instance for the concept descriptions. If there is no other naming convention or no existing concept description instance hierarchy, then a possible name for the InstanceHierarchy is "*AssetAdministrationShellConceptDescription*s". It contains the concept descriptions used or that can be used within the asset administration shells. The role of the elements contained in this instance hierarchy is "*ConceptDescription*".

Note: The AML import to an AASX Tool just needs to look for the role "*ConceptDescription*", not for the names of the instance hierarchies.

(16) For each AAS related element within an instance with role "*AssetAdministrationShell*" the corresponding role within *AssetAdministrationShellRoleClassLib* is assigned.

(17) In case of an AAS element having a data specification additionally an instance of the corresponding template with Role **DataSpecificationConcent** within the corresponding SystemUnitClassLib is assigned. Thus, all attributes defined within the template are additionally predefined for the element.

(18) The name of the element within a concept description being instantiated by a data specification template, i.e. an element with role "*DataSpecifiationConent*", is "*EmbeddedDataSpecification*". If more than one data specification template is used, then the element containing the different elements has the name "**EmbeddedDataSpecifications**" (no role) and its sub-elements are names "**EmbeddedDataSpecificication_<Number>"** because the names need to be unique.

(19) In case of a concept description AAS has a predefined data specification template (Role *DataSpecification*) called "*DataSpecificationIEC61360Template*". This template is used for a concept description by instantiating its content, i.e. element with role "*DataSpecificationContent"*.

(20) The name of the elements within a library needs to be unique. This is why for concept descriptions within the library for concept descriptions the name is chosen as follows:

<value of AAS:ConceptDescription/idShort>__<value of AAS:ConceptDescription/identification/idType>_<value of AAS:ConceptDescription/identification/id>

4.5 Rules for the Role Class Library of AML:

(21) **There is a predefined RoleClassLibrary with name "AssetAdministrationShellRoleClassLib".**
It contains all roles specific for asset administration shells.

(22) **All AAS Referables (and thus Identifiables) are mapped to specific Role Classes in AML.**
The name is identical to the name in AAS.

(23) **A small number of role classes are required for entities that have cardinality > 1 and different names like "***OperationInputVariables***", "***OperationOutputVariables***" and "***OperationInoutputVariables***" of Operation**

4.6 Rules for System Unit Class Libraries:

(24) For an AASX Tool with AML Export a system unit class library needs to be created if the AASX contains submodels or submodel elements of kind=Template[1]. This system unit class library needs a name. If there is no other naming convention or no other system unit class lib that can be used a possible name for the library is "*AssetAdministrationShellSystemUnitClasses*". It contains the asset administration shells containing elements of kind=Type. Elements within this library have the role "*AssetAdministrationShell*".

Note: The AML import to an AASX Tool just needs to look for the role "*AssetAdministrationShell*", not for the names of the system unit class libraries.

(25) If an AAS contains a submodel of kind=Template, then a corresponding SystemUnitClass is created for:

- the AAS itself, and
- the submodels within the AAS with kind=Template

(26) The same roles as for AAS with submodels of kind=Instance are assigned.

(27) There is a predefined SystemUnitClassLibrary with name "AssetAdministrationShellDataSpecificationTemplates".
It contains the predefined data specification templates as defined in Asset Administration Shell in Detail. These data specifications have the role "DataSpecification".

(28) A Data Specification has an internal element with role "DataSpecificationContent" defining all the attributes available when using the data specification.

[1] In AAS V1.0 „Template" was named „Type".

(29) In case there is the need to assign more than one data specification template to an element a system unit class containing the needed data specification content elements needs to be defined**.**

4.7 Rules for Interface Libraries:

(1) There is a predefined InterfaceClassLibrary with name "**AssetAdministrationShellInterfaceClassLib**".
It contains the predefined interfaces used within the asset administration shell.

(2) FileDataReference is an interface derived from the AutomationML interface "**ExternalDataReference**". It is used as interface for submodel element "File".

(3) **ReferableReference** is an interface for realizing references as used within submodel elements "ReferenceElement" and "RelationshipElement" (and its subclasses) within asset administration shells

5 AutomationML Libraries

This clause defines and standardizes the AutomationML libraries that shall be used for this mapping.

5.1 Role Class Library AssetAdministrationShellRoleClassLib

5.1.1 RoleClass AssetAdministrationShell

Class name	AssetAdministrationShell	
Description	An Asset Administration Shell.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/AssetAdministrationShell	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	identification	Abstract attribute for identification. Has the subattributes id and idType.

		RefSemantic: AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id
	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration “IdentifierType”. IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/version
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	derivedFrom	The derivedFrom attribute is used to establish a relationship between two Asset Administration

	(xs:string)	Shells that are derived from each other. *RefSemantic:* AAS:AssetAdministrationShell/derivedFrom
Interfaces	None	

5.1.2 RoleClass Asset

Class name	Asset	
Description	An Asset describes meta data of an asset that is represented by an AAS. The asset may either represent an asset type or an asset instance. The asset has a globally unique identifier plus – if needed – additional domain specific (proprietary) identifiers.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/Asset	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages.

		RefSemantic: AAS:Referable/description
	identification	Abstract attribute for identification. Has the subattributes id and idType. *RefSemantic:* AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id
	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration “IdentifierType”. IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/version
	dataSpecification (xs:string)	Global reference to the data specification template used by the element.

		RefSemantic: AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the asset: either type or instance. *RefSemantic:* AAS:Asset/kind
Interfaces	None	

5.1.3 RoleClass Submodel

Class name	Submodel	
Description	A Submodel defines a specific aspect of the asset represented by the AAS. A submodel is used to structure the virtual representation and technical functionality of an Administration Shell into distinguishable parts. Each submodel refers to a well-defined domain or subject matter. Submodels can become standardized and thus become submodels templates. Submodels can have different life-cycles.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/Submodel	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:*

		AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	identification	Abstract attribute for identification. Has the subattributes id and idType. *RefSemantic:* AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id
	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration “IdentifierType”. IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration.

		RefSemantic: AAS:AdministrativeInformation/version
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS:HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE].

		RefSemantic: AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
Interfaces	None	

5.1.4 RoleClass SubmodelElementCollection

Class name	SubmodelElementCollection	
Description	A submodel element collection is a set or list of submodel elements.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/SubmodelElementCollection	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category

	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded

		value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
	allowDuplicates (xs:boolean)	If allowDuplicates=true, then it is allowed that the collection contains the same element several times. *RefSemantic:* AAS:SubmodulElementCollection/allowDuplicates
	ordered (xs:boolean)	If ordered=false, then the elements in the property collection are not ordered. If ordered=true, then the elements in the collection are ordered. Default = false. Note: An ordered submodel element collection is typically implemented as an indexed array. *RefSemantic:* AAS:SubmodulElementCollection/ordered
	value (xs:string)	Submodel element contained in the collection. *RefSemantic:* AAS:SubmodulElementCollection/value
Interfaces	None	

5.1.5 RoleClass Blob

Class name	Blob
Description	A BLOB is a data element that represents a file that is contained with its source code in the value attribute.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AssetAdministrationShellRoleClassLib/Blob

Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the

		child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
	mimeType (xs:string)	Mime type of the content of the BLOB. The mime type states which file extension the file has. Valid values are e.g. “application/json”, “application/xls”, ”image/jpg”. The allowed values are defined as in RFC2046. *RefSemantic:* AAS:Blob/mimeType

	value (xs:string)	The value of the BLOB instance of a blob data element. Note: In contrast to the file property the file content is stored directly as value in the Blob data element. *RefSemantic:* AAS:Blob/value
Interfaces	None	

5.1.6 RoleClass Capability

Class name	Capability	
Description	A capability is the implementation-independent description of the potential of an asset to achieve a certain effect in the physical or virtual world.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/Capability	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages.

		RefSemantic: AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute

		from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
Interfaces	None	

5.1.7 RoleClass File

Class name	File	
Description	A role class for a File that a data element that represents an address to a file. It is derived from the AutomationML role class ExternalData that is an role type for a document type and the base class for all document type roles. It describes different document types. ExternalData is defined in AutomationML BPR_005E_ExternalDataReference_v1.0.0_2.	
Parent class	AutomationMLBPRRoleClassLib/ExternalData	
Path for element reference	AssetAdministrationShellRoleClassLib/File	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints.

		RefSemantic: AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE].

		RefSemantic: AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
Interfaces	FileDataReference (FileDataReference)	An ExternalInterface with attributes for refURI and mimeType for referencing files. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.

5.1.8 RoleClass Property

Class name	Property	
Description	A property is a data element that has a single value.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/Property	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is

		similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value.

		RefSemantic: AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
	value (xs:string)	The value of the property instance. *RefSemantic:* AAS:Property/value
	valueId (xs:string)	Reference to the global unique id if a coded value. *RefSemantic:* AAS:Property/valueId
Interfaces	None	

5.1.9 RoleClass ReferenceElement

Class name	ReferenceElement
Description	A reference element is a data element that defines a logical reference to another element within the same or another AAS or a reference to an external object or entity.

Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/ReferenceElement	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId	Description or comments on the element. The

	(xs:string)	description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
Interfaces	ReferableReference (ReferableReference)	An ExternalInterface for referencing any other referable element of the same of any other AAS or a reference to an external object or entity. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.

5.1.10 RoleClass RelationshipElement

Class name	RelationshipElement	
Description	A relationship element is used to define a relationship between two referable elements.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/RelationshipElement	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification

	kind (xs:string)	Kind of the element: either type or instance. *RefSemantic:* AAS:Asset/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:*

		AAS:Qualifier/valueId
Interfaces	first (ReferableReference)	First ExternalInterface of RelationshipElement for referencing any other referable element of the same of any other AAS or a reference to an external object or entity. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.
	second (ReferableReference)	Second ExternalInterface of RelationshipElement for referencing any other referable element of the same of any other AAS or a reference to an external object or entity. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.

5.1.11 RoleClass AnnotatedRelationshipElement

Class name	AnnotatedRelationshipElement	
Description	An annotated relationship element is an relationship element that can be annotated with additional data elements.	
Parent class	AssetAdministrationShellRoleClassLib/RelationshipElement	
Path for element reference	AssetAdministrationShellRoleClassLib/AnnotatedRelationshipElement	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:*

		AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS: HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute alue. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type

	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
	annotation (xs:string)	Annotations that hold for the relationships between the two elements. *RefSemantic:* AAS:AnnotatedRelationshipElement/annotation
Interfaces	first (ReferableReference)	First ExternalInterface of RelationshipElement for referencing any other referable element of the same of any other AAS or a reference to an external object or entity. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.
	second (ReferableReference)	Second ExternalInterface of RelationshipElement for referencing any other referable element of the same of any other AAS or a reference to an external object or entity. See detailed description for this InterfaceClass in AssetAdministrationShellInterfaceClassLib.

5.1.12 RoleClass Operation

Class name	Operation	
Description	An operation is a submodel element with input and output variables.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/Operation	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for

		referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	kind (xs:string)	Kind of the element: either template or instance. *RefSemantic:* AAS:HasKind/kind
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance with RFC5646. At it, the values of the child attributes are the labels within the respective language.

		RefSemantic: AAS:HasSemantics/semanticId
	qualifier:[TYPE]=[VALUE] (xs:string)	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element. [TYPE] is the value of the attribute type and [VALUE] is the value of the attribute value. *RefSemantic:* AAS:Qualifier/qualifier
	type (xs:string)	The type describes the type of the qualifier that is applied to the element. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/type
	value (xs:string)	The qualifier value is the value of the qualifier. Constraint AASd-006: if both, the value and the valueId are present then the value needs to be identical to the value of the referenced coded value in valueId. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/value
	valueId (xs:string)	Reference to the global unique id of a coded value. This attribute is a subattribute from qualifier: [TYPE]=[VALUE]. *RefSemantic:* AAS:Qualifier/valueId
Interfaces	None	

5.1.13 RoleClass OperationInputVariables

Class name	OperationInputVariables
Description	The list of AAS OperationVariableIn entities. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the Operation role.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AssetAdministrationShellRoleClassLib/OperationInputVariables

Attributes	None
Interfaces	None

5.1.14 RoleClass OperationOutputVariables

Class name	OperationOutputVariables
Description	The list of AAS OperationVariableOut entities. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the Operation role.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AssetAdministrationShellRoleClassLib/OperationOutputVariables
Attributes	None
Interfaces	None

5.1.15 RoleClass OperationInoutputVariables

Class name	OperationInoutputVariables
Description	The list of AAS OperationVariableInOut entities. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the Operation role.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AssetAdministrationShellRoleClassLib/OperationInoutputVariables
Attributes	None
Interfaces	None

5.1.16 RoleClass View

Class name	View	
Description	A view is a collection of referable elements w.r.t. to a specific viewpoint of one or more stakeholders.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group	
Path for element reference	AssetAdministrationShellRoleClassLib/View	
Attributes	AssociatedFacet (xs:string)	The attribute AssociatedFacet shall be used for the definition of the name of the corresponding Facet. It is part of the role class group from the AutomationMLBaseLibraries and not affiliated

		with the Asset Administration Shell. Example: AssociatedFacet = „PLCFacet“
	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	semanticId (xs:string)	Description or comments on the element. The description can be provided in several languages. This attribute has the name of the label and has a value with the label written in the default language. The individual languages are modelled as child attributes. The names of the child attributes are the prefix “aml-lang=” with the expression of the language in compliance

		with RFC5646. At it, the values of the child attributes are the labels within the respective language. *RefSemantic:* AAS:HasSemantics/semanticId
Interfaces	None	

5.1.17 RoleClass ConceptDictionary

Class name	ConceptDictionary	
Description	A dictionary contains elements that can be reused.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/ConceptDictionary	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description

Interfaces	None

5.1.18 RoleClass ConceptDescription

Class name	ConceptDescription	
Description	Explanation: The semantics of a property or other elements that may have a semantic description is defined by a concept description. The description of the concept should follow a standardized schema (realized as data specification template).	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/ConceptDescription	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:Referable/idShort
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints. *RefSemantic:* AAS:Referable/category
	description (xs:string)	Description or comments on the element. The description can be provided in several languages. *RefSemantic:* AAS:Referable/description
	identification	Abstract attribute for identification. Has the subattributes id and idType. *RefSemantic:*

		AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id
	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration “IdentifierType”. IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/version
	dataSpecification (xs:string)	Global reference to the data specification template used by the element. *RefSemantic:* AAS:HasDataSpecification/dataSpecification
	isCaseOf (xs:string)	Global reference to an external definition the concept is compatible to or was derived from. *RefSemantic:*

		AAS:ConceptDescription/isCaseOf
Interfaces	None	

5.1.19 RoleClass DataSpecification

Class name	DataSpecification	
Description	Role class of an element that has a data specification template. A template defines the additional attributes an element may or shall have.	
Parent class	AutomationMLBaseRoleClassLib/ AutomationMLBaseRole	
Path for element reference	AssetAdministrationShellRoleClassLib/ DataSpecification	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the “BrowserPath” in OPC UA. *RefSemantic:* AAS:HasSemantics/semanticId
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints.
	description (xs:string)	Description or comments on the element. The description can be provided in several languages.
	identification	Abstract attribute for identification. Has the subattributes id and idType. *RefSemantic:* AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id

	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration “IdentifierType”. IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/version
Interfaces	None	

5.1.20 RoleClass DataSpecificationContent

Class name	DataSpecificationContent
Description	Role class for the content of a DataSpecification.
Parent class	AutomationMLBaseRoleClassLib/ AutomationMLBaseRole
Path for element reference	AssetAdministrationShellRoleClassLib/ DataSpecificationContent
Attributes	None
Interfaces	None

5.2 Interface Class Library AssetAdministrationShellInterfaceClassLib

5.2.1 InterfaceClass FileDataReference

Class name	FileDataReference
Description	A FileDataReference represents the address to a File. FileDataReference is derived from the AutomationML Interface Class ExternalDataReference that is defined in AutomationML BPR_005E_ExternalDataReference_v1.0.0_2:The

	interface class “ExternalDataReference” shall be used in order to reference external documents out of the scope of AutomationML.	
Parent class	AutomationMLBPRInterfaceClassLib/ExternalDataReference	
Path for element reference	AssetAdministrationShellInterfaceClassLib/FileDataReference	
Attributes	refURI (xs:anyURI)	The attribute refURI is an URI that can represent an absolute or relative path to an L document. An added fragment (with #) references inside the document
	MIMEType (xs:string)	Mime type of the content of the File.

5.2.2 InterfaceClass ReferableReference

Class name	ReferableReference	
Description	Reference to any other referable element of the same of any other AAS or a reference to an external object or entity. For local references inside the same Asset Administration Shell an InternalLink between two objects with this interface "ReferableReference" shall be set. In this case the attribute value has to be empty. For references between different Asset Administration Shells or external objects or entities the attribute value shall be used and no InternalLink shall be set.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Path for element reference	AssetAdministrationShellInterfaceClassLib/ReferableReference	
Attributes	value (xs:string)	Reference to any other referable element of any other AAS or a reference to an external object or entity. Note: For references to any other referable element of the same AAS InternalLinks are used and this attribute value shall be empty. *RefSemantic:* AAS:ReferenceElement/value

5.3 SystemUnitClassLibrary AssetAdministrationShellTemplates

5.3.1 SystemUnitClass DataSpecificationIEC61360Template

Class name	DataSpecificationIEC61360Template
Description	An AAS Data Specification template for IEC61369. A template consists of the DataSpecificationContent containing the additional attributes to be added to the element instance that references the data specification template and meta information about the template itself (this is why DataSpecification inherits from Identifiable). In UML these are two separated classes.

Role class	AssetAdministrationShellRoleClassLib/DataSpecification	
Path for element reference	AssetAdministrationShellTemplates/ DataSpecificationIEC61360Template	
Attributes	idShort (xs:string)	Identifying string of the element within its name space. Constraint AASd-001: In case of a referable element not being an identifiable element this id is mandatory and used for referring to the element in its name space. Constraint AASd-002: idShort shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. Constraint AASd-003: idShort shall be matched case-insensitive. Note: In case of an identifiable element idShort is optional but recommended to be defined. It can be used for unique reference in its name space and thus allows better usability and a more performant implementation. In this case it is similar to the "BrowserPath" in OPC UA. *RefSemantic:* AAS:HasSemantics/semanticId
	category (xs:string)	The category is a value that gives further meta information w.r.t. to the class of the element. It affects the expected existence of attributes and the applicability of constraints.
	description (xs:string)	Description or comments on the element. The description can be provided in several languages.
	identification	Abstract attribute for identification. Has the subattributes id and idType. *RefSemantic:* AAS:Identifiable/identification
	id (xs:string)	Identifier of the element. Its type is defined in idType. Id is a subproperty of identification. *RefSemantic:* AAS:Identifier/id
	idType (xs:string)	Type of the Identifier, e.g. IRI, IRDI etc. The supported Identifier types are defined in the enumeration "IdentifierType". IdType is a subproperty of identification. *RefSemantic:* AAS:Identifier/idType
	administration	Abstract attribute for administration. has the subattributes revision and version. *RefSemantic:* AAS:Identifiable/administration
	revision (xs:string)	Revision of the element. Constraint AASd-005: A revision requires a version. This means, if there is

		no version there is no revision neither. Revision is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/revision
	version (xs:string)	Version of the element. Version is a subproperty of administration. *RefSemantic:* AAS:AdministrativeInformation/version
Interfaces	None	

5.3.2 SystemUnitClass DataSpecificationIEC61360

Class name	DataSpecificationIEC61360	
Description	The content of an AAS Data Specification template for IEC61360.	
Role class	AssetAdministrationShellRoleClassLib/DataSpecificationContent	
Path for element reference	AssetAdministrationShellTemplates/ DataSpecificationIEC61360Template/ DataSpecificationIEC61360	
Attributes	preferredName (xs:string)	Identifies the attribute hierarchy for preferredName in above attribute hierarchy. Subordinate attributes are designated by the country code information (see aml-lang literal). *RefSemantic:* IEC:DataSpecificationIEC61360/preferredName
	shortName (xs:string)	Identifies the attribute for shortName in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/shortName
	unit (xs:string)	Identifies the attribute for unit in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/unit
	unitId (xs:string)	Identifies the attribute for unitId in above attribute hierarchy in its string serialization. *RefSemantic:* IEC:DataSpecificationIEC61360/unitId
	sourceOfDefinition (xs:string)	Identifies the attribute for sourceOfDefinition in above attribute hierarchy. Subordinate attributes

		are designated by the country code information (see aml-lang literal). *RefSemantic:* IEC:DataSpecificationIEC61360/sourceOfDefinition
	symbol (xs:string)	Identifies the attribute for symbol in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/symbol
	dataType (xs:string)	Identifies the attribute for dataType in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/dataType
	definition (xs:string)	Identifies the attribute for definition in above attribute hierarchy. Subordinate attributes are designated by the country code information (see aml-lang literal). *RefSemantic:* IEC:DataSpecificationIEC61360/definition
	valueFormat (xs:string)	Identifies the attribute for valueFormat in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/valueFormat
	valueList (xs:string)	Identifies the attribute for valueList in above attribute hierarchy. *RefSemantic:* IEC:DataSpecificationIEC61360/valueList
	value (xs:string)	The attribute value. *RefSemantic:* IEC:DataSpecificationIEC61360/value
	valueId (xs:string)	The id for the value. *RefSemantic:* IEC:DataSpecificationIEC61360/valueId
Interfaces	None	

6 Practical Examples

6.1 Unified Example

The following example is used to demonstrate the main features of the mapping of the metamodel to AutomationML. In [VWSiD2.0] this example is used for the mapping to each of the different data formats. Intention is to motivate the equivalency of information in different representations.

It shows an AAS with three submodels: *TechnicalData*, *OperationalData* and *Documentation*. The asset, a motor, that it is representing has in this example the global ID http://customer.com/assets/KHBVZJSQKIY.

The *TechnicalData* submodel contains data that is available at engineering time: the maximum rotation speed measured in 1/min. Its semanticId is 0173-1#02-BAA120#008. It is an eCl@ss IRDI. However, in this example a copy of the eCl@ss entry values are copied to a corresponding concept description with the same IRDI. The unit “1/min” has also a unique id, “0173-1#05-AAA650#002”.

The third submodel “Documentation” contains a pdf document, the operating manual.

The AutomationML serialization of this example can be found in UnifiedAASExampleVWSiD2.0.aml which is distributed together with this document on the AutomationML website: https://www.automationml.org/o.red.c/dateien.html

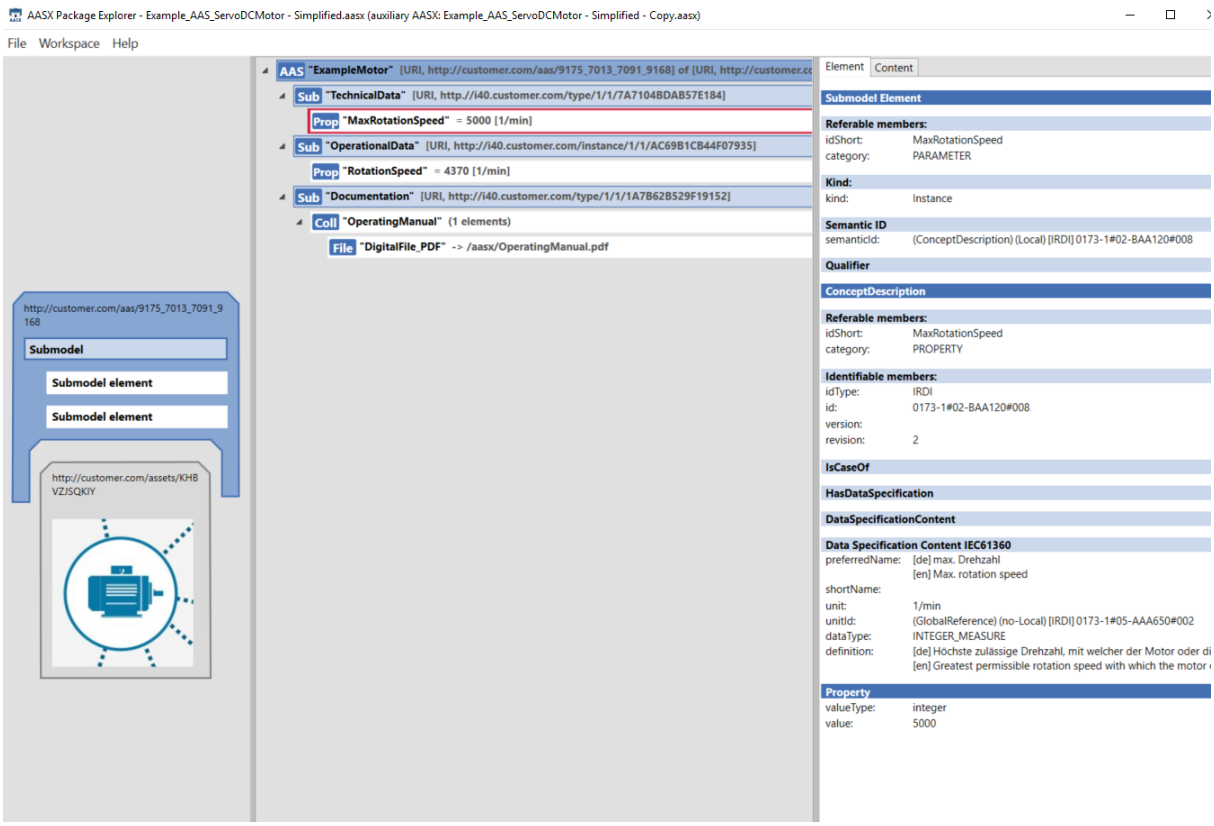

Figure 5 Unified Example for ExampleMotor

In Figure 6 the example is shown in the AutomationML Editor. Details are explained in the following clauses.

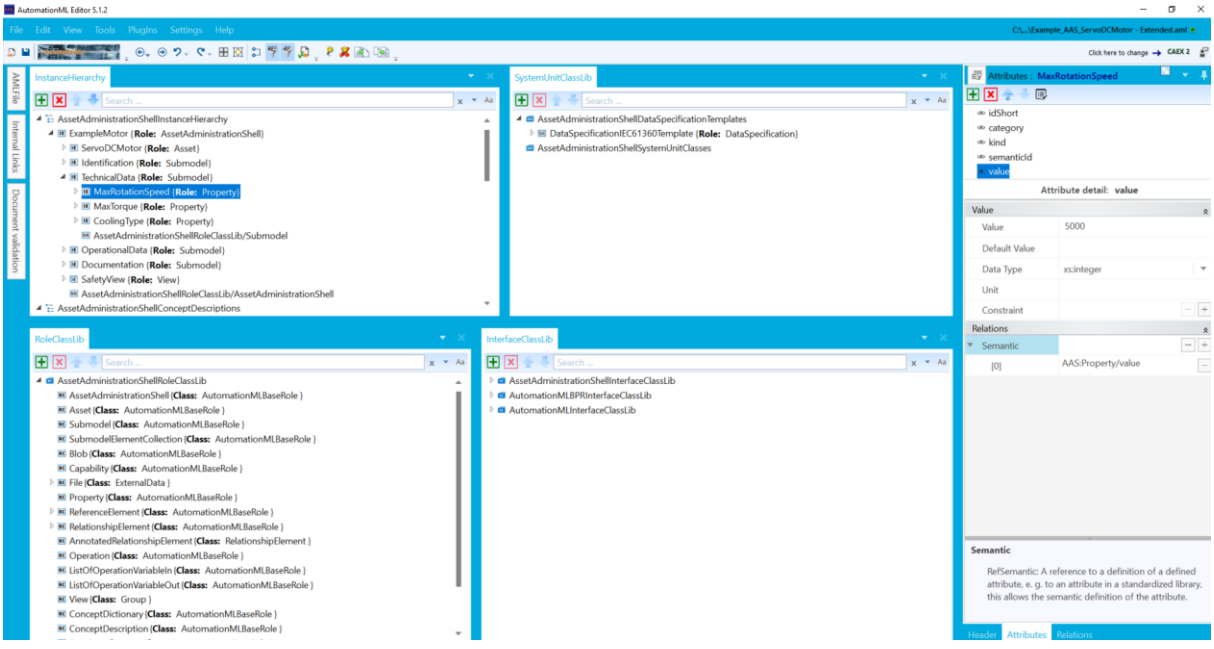

Figure 6 Example in Automation ML Editor

6.2 Detailed Examples

In this chapter the example above is taken and extended to describe the features and rules of the mapping in a detailed way.

6.2.1 Example Property and Concept Description

Figure 7 shows the property “MaxRotationSpeed”.

Please note: The Unit “1/min” could have been added to the AML field “Unit” of attribute value. However, it is available only indirectly via its semanticId (see Figure 8).

The value of attribute “semanticId” uses the reference serialization as defined in *[VWSiD2.0]*.

MaxRotationSpeed - Attributes

Name	Value	Unit
semanticId	(ConceptDescription)(local)[IRDI] 0173-1#02-BAA120#008	--
value	5000	--
kind	Instance	--
idShort	MaxRotationSpeed	--
category	PARAMETER	--

Figure 7 Example Property MaxRotationSpeed

Attributes

Attribute-Table

Name	Value	Unit
preferredName		
aml-lang=de	max. Drehzahl	
aml-lang=en	Max. rotation speed	
Attribute		
shortName		
unit	1/min	
unitId	(GlobalReference)(no-local)[IRDI] 0173-1#05-AAA650#002	
dataType	INTEGER_MEASURE	
definition		
aml-lang=de	Höchste zulässige Drehzahl, mit welcher der Motor oder die Speiseinheit betrieben werden darf	
aml-lang=en	Greatest permissible rotation speed with which the motor or feeding unit may be operated	

Header | Attributes | Internal Links

Figure 8 Example DataSpecificationContent of Concept Description MaxRotationSpeed conformant to template DataSpecificationIEC61360Template

6.2.2 Example Attributes of Attributes

Complex Types are realized as attributes with sub-attributes (see Figure 9).

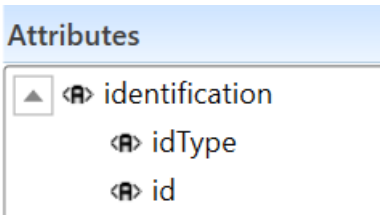

Figure 9 Example Identification with two sub-attributes

6.2.3 Example Language Tagged Strings

For attributes of type langString or langStringSet the predefined AML attribute “aml-lang” is used (see Figure 10).

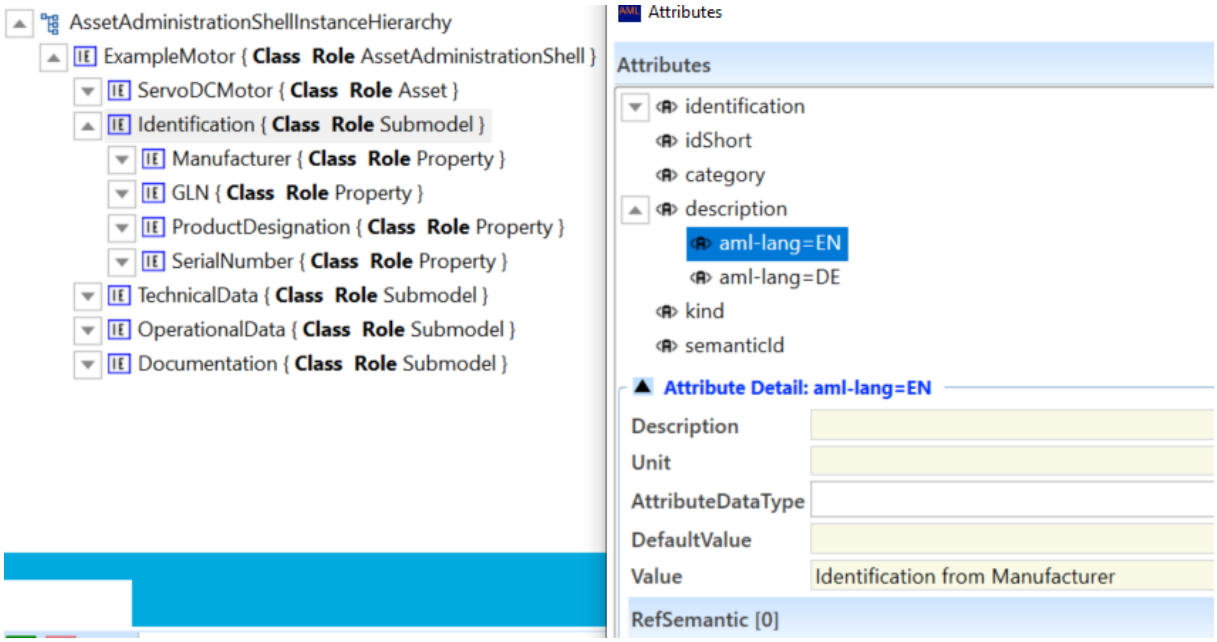

Figure 10 Example for attribute value in multiple languages

6.2.4 Example Asset

In Figure 11 the asset represented by an asset administration shell is shown: it has the role “Asset” assigned to it.

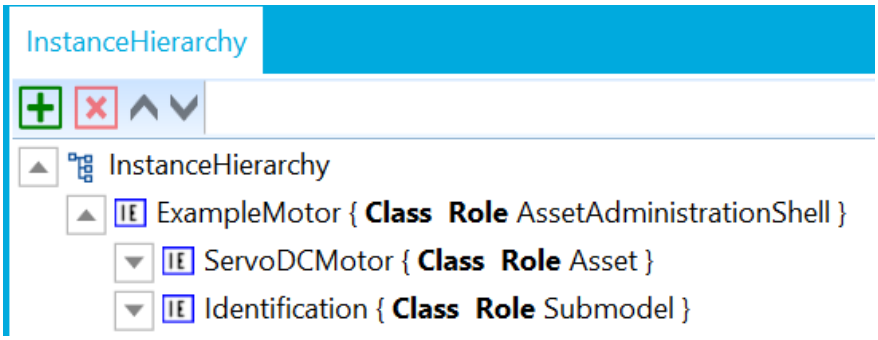

Figure 11 Example Asset in Instance Hierarchy

6.2.5 Example RefSemantic

Note: *RefSemantic*[2] is an attribute of AutomationML whereas semanticId is an attribute of Asset Administration Shell.

RefSemantic describes the semantics of the metamodel. Thus the values reference to a description within the Asset Administration Shell Specification.

SemanticId describes the semantics of an instance.

RefSemantic for the AML attribute “semanticId” of an AML element “MaxRotationSpeed” for example says that it has the semantics as defined for the attribute “semanticId” within the AAS specification. The semanticId of the property says that it is the maximum rotation speed (see Figure 12Figure 12).

For a complete list of all RefSemantic see Appendix A.

[2] In the tooling it is sometimes just denoted as „Semantic“.

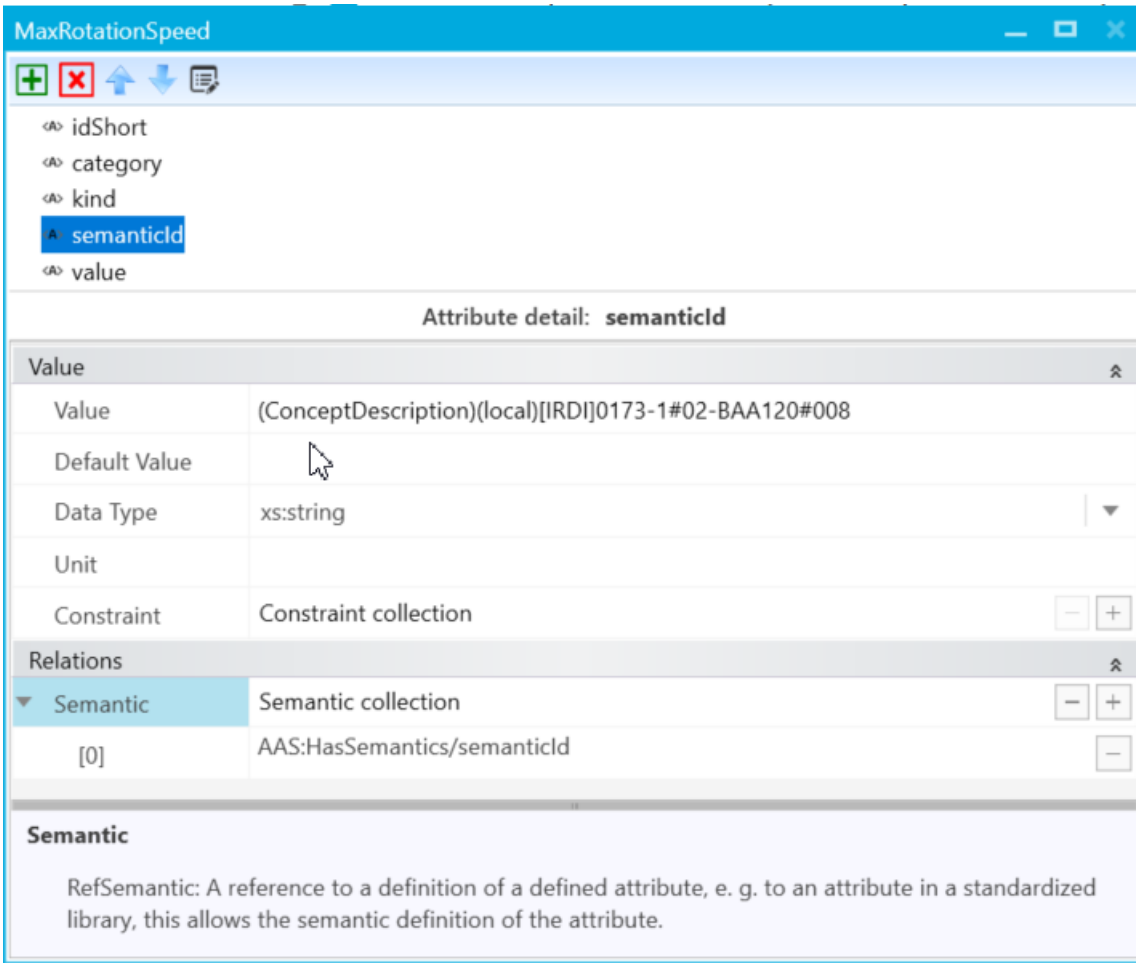

Figure 12 Example for RefSemantic and semanticId of the Property "MaxRotationSpeed"

6.2.6 Example References

References are serialized into a single string. See example in Figure 13: The value of the semanticId is typed as "Reference" and is serialized as string.
Exceptions: ReferenceElement and RelationshipElement including its subtypes as well as Views.

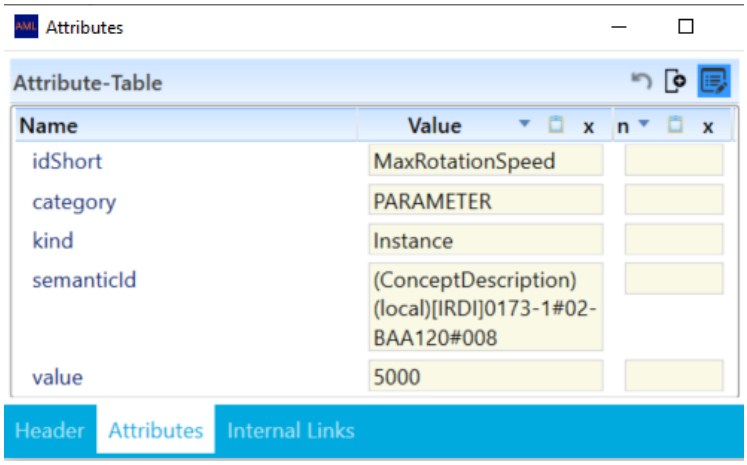

Figure 13 Example for serialized reference as value for attribute semanticId

6.2.7 Example ReferenceElement

In Figure 14 an example for a reference element is given: it references another element within the AAS by using the interface “ReferableReference”. The blue dotted line represents a InternalLink to the target referenced element (not visible here).

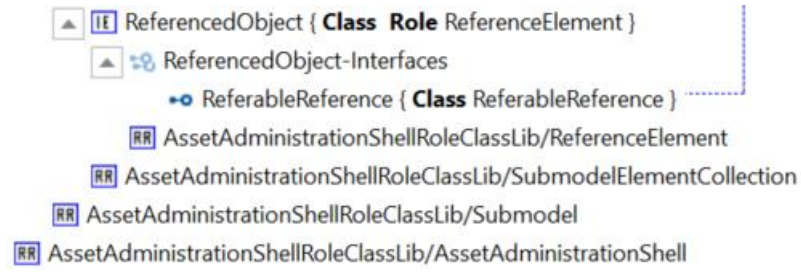

Figure 14 Example for ReferenceElement with Interface

6.2.8 Example File

In Figure 15 an AAS File submodel element is shown. It is realized with the predefined AML interface FileDataReference from Interface Class Library “AssetAdministrationShellInterfaceClassLib”. It is derived from the AML interface “ExternalDataReference” from Interface Class Library “AutomationMLBPRInterfaceClassLib” that again is derived from the AML interface “ExternalDataConnector” as defined in the Interface Class Library “AutomationMLInterfaceClassLib”. The interface *FileDataReference* already has a MIMEType inherited that is conformant to AAS:/File/mimeType.

Figure 15 Example File

6.2.9 Example Operation

Operation is mapped to InternalElement in AML with Role Class Operation.

InoputVariables/OutputVariables/InoutputVariables are mapped to InternalElements in AML with RoleClass OperationInputVariables resp. ~Output~. These InternalElements contain subordinated InternalElements for the submodel elements.

The in or out element can be empty, i.e. does not need to have child elements.

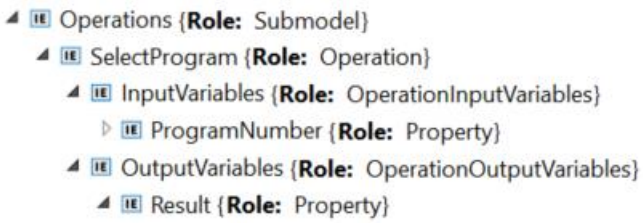

Figure 16 Example Operation "SelectProgram" with input variables

6.2.10 Example Qualifier

The example in Figure 17 shows a property with two qualifiers. One of the qualifiers is of type “ExpressionsSemantic”, the other of type “PredicateRelation”. The qualifier value of the “PredicateRelation” qualifier is “GREATER_THAN_0”.

Qualifiers are not referable, i.e. they do not have an idShort attribute. This is why a name of the AML attribute was generated as follows: **qualifier:<value of AAS:Qualifier/type>=<value of Qualifier/value>**.

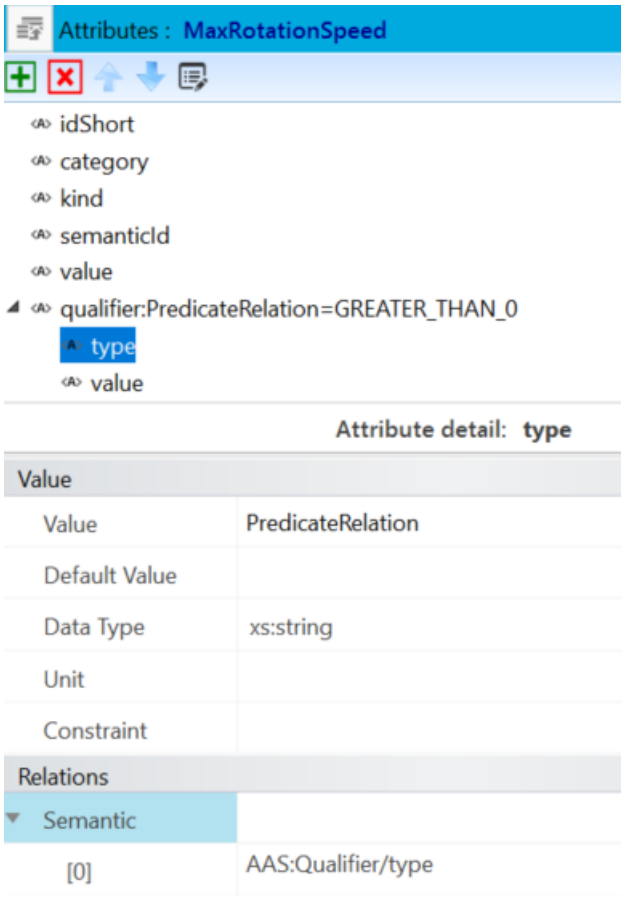

Figure 17 Example Qualifier "PredicateRelation" with qualifier value "GREATER_THAN_0" for a Property

6.2.11 Example Concept Descriptions

Concept Descriptions are stored in an Instance Hierarchy. The default name is "AssetAdministrationShellConceptDescriptions".

The name of the elements within an instance hierarchy needs to be unique. This is why the name is chosen as follows:

<value of AAS:ConceptDescription/idShort>_<value of AAS:ConceptDescription/identification/idType>__<value of AAS:ConceptDescription/identification/id>

An example concept description for max. rotation speed using the predefined data specification template “DataSpecificationIEC61360” is shown in Figure 18 and Figure 19.

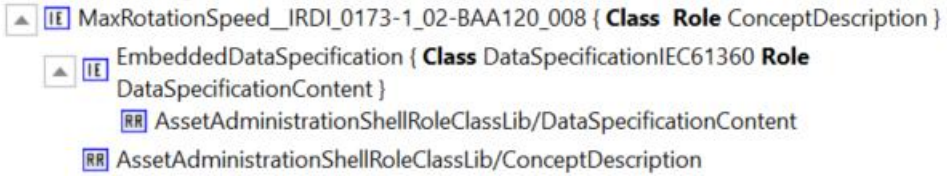

Figure 18 Example Concept Description using predefined data specification template IEC61360

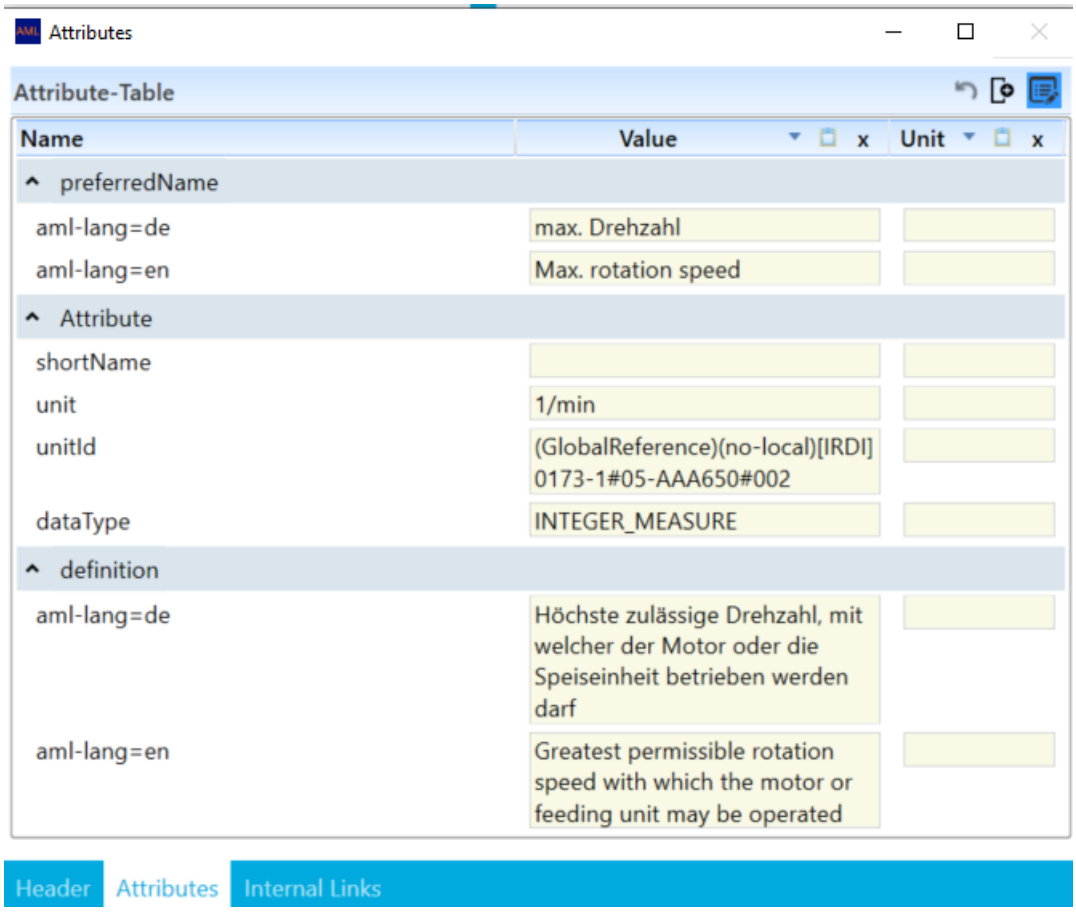

Figure 19 Example Embedded Data Specification IEC61360 of Concept Description for Property “MaxRotationSpeed”

6.2.12 Example View

In Figure 20Figure 20 an example view with name “SafetyView” is shown that contains a reference to the property RotationSpeed.

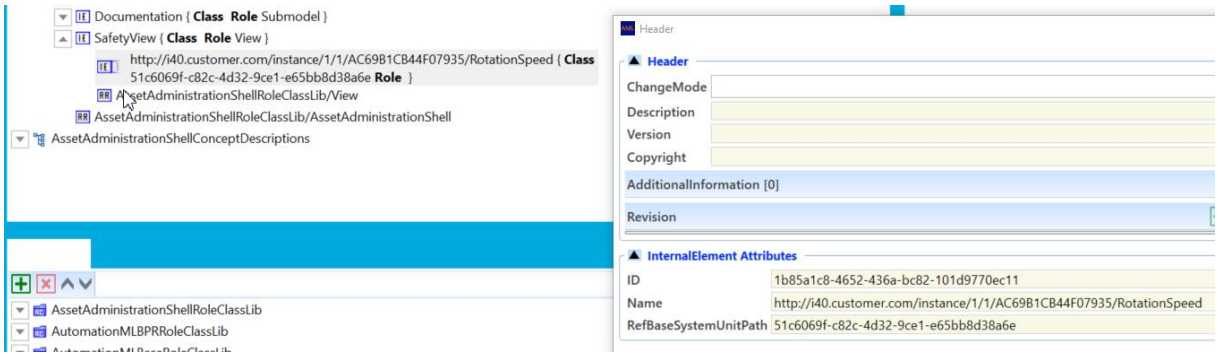

Figure 20 Example SafetyView

6.2.13 Example Submodels of kind=Template

Submodel templates (i.e. submodels and elements with kind = Template) are modelled as System Unit Classes. They are part of a System Unit Class Library with default name “AssetAdministrationShellSystemUnitClasses”.

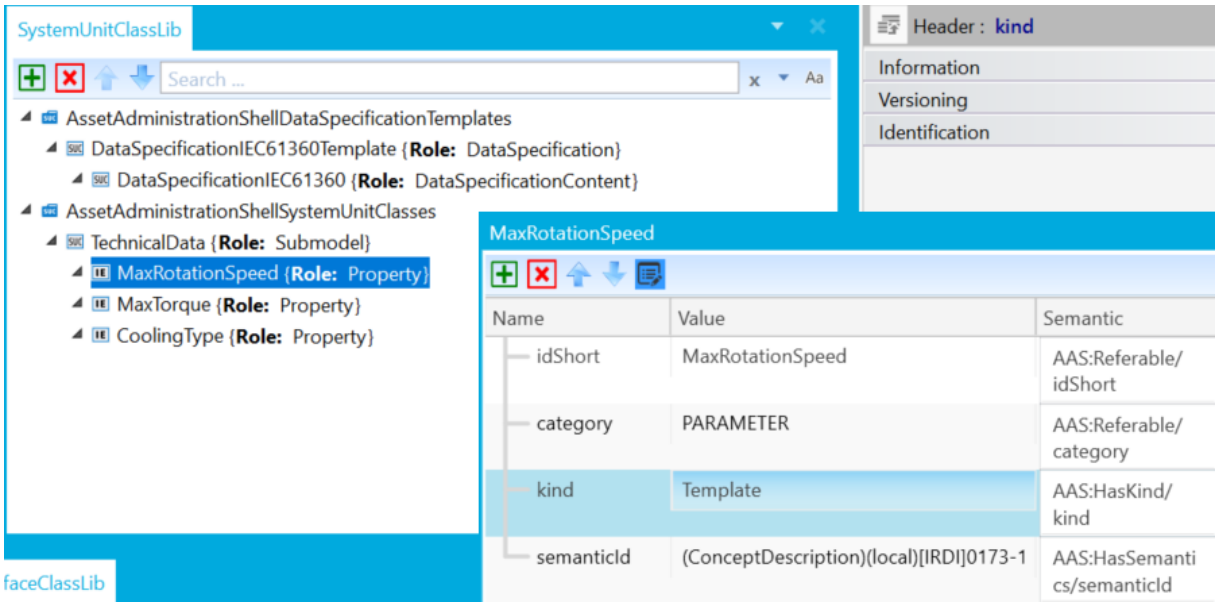

Figure 21 Example for System Unit Class with a Submodel Template for Technical Data

7 Outlook

In this document the first version of the mapping of the Asset Administration Shell metamodel to AutomationML is described. The metamodel as well as this mapping will be continuously evolving and this document will be updated and extended.

This clause describes some of the topics that will be addressed in upcoming versions.

An essential aspect of digitization is security. The metamodel of the Asset Administration Shell as in *[VWSiD2.0]* has already integrated security aspects in the design. These aspects will also be integrated in future version of the mapping described in this document.

AutomationML has already and is continuing to define semantic models for various use case and data exchange scenarios such as AR APC *[AR APC1.2.0]*, the upcoming Automation Component Description and others. Furthermore, AutomationML models adapting these concepts exist in the field.

It is essential to enable the interleaving of the Asset Administration Shell concepts with existing AutomationML models. It should be possible to reuse the semantics, references and relations that are already available in the AutomationML model also in the AAS. Furthermore, redundant data within the models shall be prevented. The basics have been with the here described mapping. The AAS-related Role Classes can already be used together with existing libraries. The handling of semantics, reference and relations will be evolved in the future version of this document.

In a next version the topic of reusable submodels of kind=instance in multiple asset administration shells will be discussed.

Appendix A RefSemantic and Roles for AAS

The following symbolic identifications (literals) are used by the AutomationML mapping. Without loss of generality, these literals are recommended to be used also by other functionalities, unless not otherwise stated by this specification.

Literals to be used in names in namespace “AAS:”	Description
AssetAdministrationShellRoleClassLib	Name of the root node element for the AAS role classes in the AML RoleClassLib.
AssetAdministrationShellInterfaceClassLib	Name of the root node element for the AAS information in the AML InterfaceClassLib.
AssetAdministrationShellDataSpecifications	Name of the root node element for the predefined AAS data specification templates in the AML SystemUnitClassLib.

The following symbolic identifications (literals) are used by the AutomationML mapping in case no other names or existing libraries are selected for export.

Literals to be used in names in namespace “AAS:”	Description
AssetAdministrationShell-InstanceHierarchy	Default name of the root node element for the AAS information in the AML InstanceHierarchy.
AssetAdministrationShell-SystemUnitClasses	Default name of the root node element for the AAS information in the AML SystemUnitClassLib.
AssetAdministrationShell-ConceptDescriptions	Default name of the root node element for the AAS concept descriptions in the AML InstanceHierarchy.

The following literals are used to identify attributes of AAS metamodel elements. In AutomationML, this is done via the "RefSemantic" mechanism and the literal is preceded by the string "AAS:" (e.g. "AAS:Asset"). The rules for creating these values are described in *[VWSiD2.0]* .

Literals to identify attributes[3] in namespace “AAS:” (value of RefSemantic)	Description
AAS:Referable/ idShort	Identifies the *idShort* attribute within an AAS Referable, such as Property.
AAS:Referable/ category	As above for *category*
AAS:Referable/ description	As above for *description*
AAS:Asset/ kind	Identifies the *kind* attribute within various AAS entities.

[3] Note: the blank within the path is to be ignored, it is just used for better formatting of the table. I.e. *AAS:Referable/ idShort* needs to be *AAS:Referable/idShort* as value of the RefSemantics attribute.

AAS:HasKind/ kind	Identifies the *kind* attribute of submodel and submodel element entities.
AAS:HasSemantics/ semanticId	Identifies the *semanticId* attribute within various AAS entities.
AAS:Identifiable/ identification	Identifies the attribute hierarchy for *identification* information within AAS Identifiables.
AAS:Identifier/ idType	Identifies the attribute for *idType* in above attribute hierarchy.
AAS:Identifier/ id	Identifies the attribute for *id* in above attribute hierarchy.
AAS:Identifiable/ administration	Identifies the attribute hierarchy for *administrative information* within AAS Identifiables.
AAS:AdministrativeInformation/ version	Identifies the attribute for *version* in above attribute hierarchy.
AAS:AdministrativeInformation/ revision	Identifies the attribute for *revision* in above attribute hierarchy.
AAS:HasDataSpecification/ dataSpecification	Identifies an attribute containing the *dataSpecification* reference in its string serialization.
AAS:Qualifiable/ qualifier	Identifies the attribute hierarchy for an AAS *Qualifiable* in total.
AAS:Qualifier/ type	Identifies the attribute for *qualifier type* in above attribute hierarchy.
AAS:Qualifier/ value	Identifies the attribute for *qualifier value* in above attribute hierarchy.
AAS:Qualifier/ valueId	Identifies the attribute for qualifier *valueId* in above attribute hierarchy.
AAS:AssetAdministrationShell/ derivedFrom	Identifies the attribute containing the *derivedFrom* reference in its string serialization
AAS:Asset/ assetIdentificationModel	Identifies the attribute containing the *assetIdentificationModel* reference for AAS Asset in its string serialization
AAS:Property/ value AAS:MultiLanguageProperty/ value AAS:Blob/ value AAS:File/ value AAS:ReferenceElement/ value	Identifies the *value* attribute in various AAS SubmodelElements.
AAS:Property/ valueId	Identifies the *valueId* attribute in the property element.
AAS:Blob/ mimeType	Identifies the *mimeType* attribute in various AAS SubmodelElements. For SubmodelElement File the *MIMEType* attribute of the predefined *FileDataReference* interface is used. That has the same semantics as AAS:File/ mimeType
AAS:AnnotationRelationshipElement/ annotations	Annotations of the AAS SubmodelElement Annotated Relationship. The references to first and second object in the relationship is realized via ReferableReference interfaces.
AAS:ConceptDescription/ isCaseOf	Identifies the attribute containing the *isCaseOf* reference

	within AAS ConceptDescription in its string serialization.
AAS:ConceptDescription/ dataSpecification	Identifies the attribute containing the *dataSpecification* reference within AAS EmbeddedDataSpecification in its string serialization.

The following literals are used to identify the predefined templates as defined in *[VWSiD2.0].* So far only the IEC61360 template for concept descriptions is supported.

Literals to identify system unit classes in AssetAdministrationShellDataSpecificationTemplates	Description
DataSpecificationIEC61360Template	System Unit class for the predefined data specification template representing IEC61360 attributes for properties etc. It has the role DataSpecification.
DataSpecificationIEC61360	System Unit class for the content of the predefined data specification template DataSpecificationIEC61360Template It has the role DataSpecificationContent.

The AutomationML mapping supports the template *"DataSpecificationIEC616360"*. The attribute literals are defined in the following table. The name space qualifier in this case is "IEC". Example: *"IEC:DataSpecificationIEC61360/preferredName"*

Literals to identify attributes n namespace "IEC:" (value of RefSemantic)	Description
IEC:DataSpecificationIEC61360/ preferredName	Identifies the attribute hierarchy for *preferredName* in above attribute hierarchy. Subordinate attributes are designated by the country code information (see aml-lang literal).
IEC:DataSpecificationIEC61360/ shortName	Identifies the attribute for *shortName* in above attribute hierarchy.
IEC:DataSpecificationIEC61360/ unit	Identifies the attribute for *unit* in above attribute hierarchy.
IEC:DataSpecificationIEC61360/ unitId	Identifies the attribute for *unitId* in above attribute hierarchy in its string serialization.
IEC:DataSpecificationIEC61360/ sourceOfDefinition	Identifies the attribute for *sourceOfDefinition* in above attribute hierarchy. Subordinate attributes are designated by the country code information (see aml-lang literal).
IEC:DataSpecificationIEC61360/ symbol	Identifies the attribute for *symbol* in above attribute hierarchy.
IEC:DataSpecificationIEC61360/ valueFormat	Identifies the attribute for *valueFormat* in above attribute hierarchy.
IEC:DataSpecificationIEC61360/ dataType	Identifies the attribute for *dataType* in above attribute hierarchy.
IEC:DataSpecificationIEC61360/ definition	Identifies the attribute for *definition* in above attribute hierarchy. Subordinate attributes are designated by the country code

	information (see aml-lang literal).
IEC:DataSpecificationIEC61360/ valueFormat	Identifies the attribute for *valueFormat* in above attribute hierarchy.

The following literals are used to identify AAS entities. In AutomationML, this is done via the "Role" mechanism and the role name is preceeded by " AssetAdministrationShellRoleClassLib/" in order to identify the according role class lib (e.g. "AssetAdministrationShellRoleClassLib/ Asset").

Literals to identify AAS entities	Inherits from	Description
Asset	AutomationMLBaseRole	The AAS Asset entity. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the *AssetAdministrationShell* role.
View	Group	The AAS View entity. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the *AssetAdministrationShell* role.
AssetAdministrationShell	AutomationMLBaseRole	The AssetAdministrationShell entity. In AML, the corresponding InternalElement is child of the instance hierarchy named *AssetAdministrationShellInstanceHierarchy*.
Submodel	AutomationMLBaseRole	The AAS Submodel entity. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the *AssetAdministrationShell* role.
SubmodelElementCollection, Operation	AutomationMLBaseRole	One of the AAS SubmodelElement entities. In AML, the corresponding element is always an InternalElement with this role and is a child of the InternalElement with the *Submodel* role.
Property, Blob, Capability	AutomationMLBaseRole	One of the AAS SubmodelElement entities. In AML, the corresponding element is always an InternalElement with this role and is a child of the InternalElement with the *Submodel* role.
ReferenceElement	AutomationMLBaseRole	The value of an ReferenceElement is realized as an interface *ReferableReference*.
RelationshipElement, AnnotatedRelationshipElement	AutomationMLBaseRole	The attributes *first* and *second* of a relationship element are realized with interface *ReferableReference*.
File	ExternalData	The AAS File SubmodelElement is realized as ExternalData with interface *FileDataReference*.
OperationInputVariables, OperationOutputVariables, OperationInoutputVariables	AutomationMLBaseRole	The list of AAS OperationVariable entities. In AML, the corresponding InternalElement with this role is a child of the InternalElement with the *Operation* role.
ConceptDescription	AutomationMLBaseRol	The AAS ConceptDescription entity. In AML,

	e	the corresponding InternalElement is child of the instance hierarchy library named *AssetAdministrationShellConceptDescriptions*.